```
                        ;
                        ;--------------------------------------------------------------
                        ; <CAVEAT EMPTOR>:
                        ;
                        ;       THE  BIOS ROUTINES ARE MEANT TO BE ACCESSED THROUGH
                        ;       SOFTWARE INTERRUPTS ONLY.  ANY ADDRESSES PRESENT IN
                        ;       THE LISTINGS  ARE INCLUDED  ONLY FOR  COMPLETENESS,
                        ;       NOT FOR  REFERENCE.   APPLICATIONS WHICH  REFERENCE
                        ;       ABSOLUTE  ADDRESSES  WITHIN THIS  CODE  VIOLATE THE
                        ;       STRUCTURE AND DESIGN OF BIOS.
                        ;--------------------------------------------------------------
                        ;-----------------------------------------
                        ;                EQUATES
                        ;-----------------------------------------
= 0060                  PORT_A          EQU     60H     ; 8255 PORT A ADDR
= 0038                  CPUREG          EQU     38H     ; MASK FOR CPU REG BITS
= 0007                  CRTREG          EQU     7       ; MASK FOR CRT REG BITS
= 0061                  PORT_B          EQU     61H     ; 8255 PORT B ADDR
= 0062                  PORT_C          EQU     62H     ; 8255 PORT C ADDR
= 0063                  CMD_PORT        EQU     63H
= 0089                  MODE_8255       EQU     10001001B
= 0020                  INTA00          EQU     20H     ; 8259 PORT
= 0021                  INTA01          EQU     21H     ; 8259 PORT
= 0020                  EOI             EQU     20H
= 0040                  TIMER           EQU     40H
= 0043                  TIM_CTL         EQU     43H     ; 8253 TIMER CONTROL PORT ADDR
= 0040                  TIMER0          EQU     40H     ; 8253 TIMER/CNTER 0 PORT ADDR
= 0061                  KB_CTL          EQU     61H     ; CONTROL BITS FOR KEYBOARD
= 03DA                  VGA_CTL         EQU     3DAH    ; VIDEO GATE ARRAY CONTROL PORT
= 00A0                  NMI_PORT        EQU     0A0H    ; NMI CONTROL PORT
= 0080                  PORT_80         EQU     080H
= 03DF                  PAGREG          EQU     03DFH   ; CRT/CPU PAGE REGISTER
= 0060                  KBPORT          EQU     060H    ; KEYBOARD PORT
= 4000                  DIAG_TABLE_PTR  EQU     4000H
= 2000                  MINI            EQU     2000H
                        ;-----------------------------------------
                        ;          DISKETTE EQUATES
                        ;-----------------------------------------
= 00F2                  NEC_CTL         EQU     0F2H    ; CONTROL PORT FOR THE DISKETTE
= 0080                  FDC_RESET       EQU     80H     ; RESETS THE NEC (FLOPPY DISK
                                                        ; CONTROLLER).  0 RESETS,
                                                        ; 1 RELEASES THE RESET
= 0020                  WD_ENABLE       EQU     20H     ; ENABLES WATCH DOG TIMER IN NEC
= 0040                  WD_STROBE       EQU     40H     ; STROBES WATCHDOG TIMER
= 0001                  DRIVE_ENABLE    EQU     01H     ; SELECTS AND ENABLES DRIVE

= 00F4                  NEC_STAT        EQU     0F4H    ; STATUS REGISTER FOR THE NEC
= 0020                  BUSY_BIT        EQU     20H     ; BIT = 0 AT END OF EXECUTION PHASE
= 0040                  DIO             EQU     40H     ; INDICATES DIRECTION OF TRANSFER
= 0080                  RQM             EQU     80H     ; REQUEST FOR MASTER
= 00F5                  NEC_DATA        EQU     0F5H    ; DATA PORT FOR THE NEC
                        ;-----------------------------------------
                        ;          8088 INTERRUPT LOCATIONS
                        ;-----------------------------------------
0000                    ABSO    SEGMENT AT 0
0008                            ORG     2*4
0008                    NMI_PTR         LABEL    WORD
000C                            ORG     3*4
000C                    INT3_PTR        LABEL   WORD
0014                            ORG     5*4
0014                    INT5_PTR        LABEL   WORD
0020                            ORG     8*4
0020                    INT_PTR         LABEL   DWORD
0040                            ORG     10H*4
0040                    VIDEO_INT       LABEL   WORD
0070                            ORG     1CH*4
0070                    INT1C_PTR       LABEL   WORD
0074                            ORG     1DH*4
0074                    PARM_PTR        LABEL   DWORD   ; POINTER TO VIDEO PARMS
0060                            ORG     18H*4
0060                    BASIC_PTR       LABEL   WORD    ; ENTRY POINT FOR CASSETTE BASIC
0078                            ORG     01EH*4          ; INTERRUPT 1EH
0078                    DISK_POINTER    LABEL   DWORD
007C                            ORG     01FH*4          ; LOCATION OF POINTER
007C                    EXT_PTR LABEL   DWORD           ; POINTER TO EXTENSION
0110                            ORG     044H*4
0110                    CSET_PTR        LABEL   DWORD   ; POINTER TO DOT PATTERNS
0120                            ORG     048H*4
0120                    KEY62_PTR       LABEL   WORD    ; POINTER TO 62 KEY KEYBOARD CODE
0124                            ORG     049H*4
0124                    EXST            LABEL   WORD    ; POINTER TO EXT. SCAN TABLE
0204                            ORG     081H*4
0204                    INT81   LABEL   WORD
0208                            ORG     082H*4
0208                    INT82   LABEL   WORD
0224                            ORG     089H*4
0224                    INT89   LABEL   WORD
0400                            ORG     400H
0400                    DATA_AREA       LABEL   BYTE    ; ABSOLUTE LOCATION OF DATA SEGMENT
0400                    DATA_WORD       LABEL   WORD
7C00                            ORG     7C00H
7C00                    BOOT_LOCN       LABEL   FAR
7C00                    ABSO    ENDS
```

```
                                ;--------------------------------------------
                                ; STACK -- USED DURING INITIALIZATION ONLY
                                ;--------------------------------------------
0000                            STACK   SEGMENT AT 30H
0000    80 [                            DW      128 DUP(?)
            ????
                ]

0100                            TOS     LABEL   WORD
0100                            STACK   ENDS
                                ;--------------------------------------
                                ;       ROM BIOS DATA AREAS
                                ;--------------------------------------
0000                            DATA    SEGMENT AT 40H
0000    04 [                    RS232_BASE      DW      4 DUP(?) ; ADDRESSES OF RS232 ADAPTERS
            ????
                ]

0008    04 [                    PRINTER_BASE    DW      4 DUP(?) ; ADDRESSES OF PRINTERS
            ????
                ]

0010  ????                      EQUIP_FLAG      DW      ?       ; INSTALLED HARDWARE
0012  ??                        KBD_ERR         DB      ?       ; COUNT OF KEYBOARD TRANSMIT ERRORS
0013  ????                      MEMORY_SIZE     DW      ?       ; USABLE MEMORY SIZE IN K BYTES
0015  ????                      TRUE_MEM        DW      ?       ; REAL MEMORY SIZE IN K BYTES
                                ;--------------------------------------
                                ;       KEYBOARD DATA AREAS
                                ;--------------------------------------
0017  ??                        KB_FLAG         DB      ?
                                ;----- SHIFT FLAG EQUATES WITHIN KB_FLAG
= 0040                          CAPS_STATE      EQU     40H     ; CAPS LOCK STATE HAS BEEN TOGGLED
= 0020                          NUM_STATE       EQU     20H     ; NUM LOCK STATE HAS BEEN TOGGLED
= 0008                          ALT_SHIFT       EQU     08H     ; ALTERNATE SHIFT KEY DEPRESSED
= 0004                          CTL_SHIFT       EQU     04H     ; CONTROL SHIFT KEY DEPRESSED
= 0002                          LEFT_SHIFT      EQU     02H     ; LEFT SHIFT KEY DEPRESSED
= 0001                          RIGHT_SHIFT     EQU     01H     ; RIGHT SHIFT KEY DEPRESSED
0018  ??                        KB_FLAG_1       DB      ?       ; SECOND BYTE OF KEYBOARD STATUS
= 0080                          INS_SHIFT       EQU     80H     ; INSERT KEY IS DEPRESSED
= 0040                          CAPS_SHIFT      EQU     40H     ; CAPS LOCK KEY IS DEPRESSED
= 0020                          NUM_SHIFT       EQU     20H     ; NUM LOCK KEY IS DEPRESSED
= 0010                          SCROLL_SHIFT    EQU     10H     ; SCROLL LOCK KEY IS DEPRESSED
= 0008                          HOLD_STATE      EQU     08H     ; SUSPEND KEY HAS BEEN TOGGLED
= 0004                          CLICK_ON        EQU     04H     ; INDICATES THAT AUDIO FEEDBACK IS
                                                                ; ENABLED
= 0002                          CLICK_SEQUENCE  EQU     02H     ; OCURRNCE OF ALT-CTRL-CAPSLOCK HAS
                                                                ; OCCURED
0019  ??                        ALT_INPUT       DB      ?       ; STORAGE FOR ALTERNATE KEYPAD
                                                                ; ENTRY
001A  ????                      BUFFER_HEAD     DW      ?       ; POINTER TO HEAD OF KEYBOARD BUFF
001C  ????                      BUFFER_TAIL     DW      ?       ; POINTER TO TAIL OF KEYBOARD BUFF
001E    10 [                    KB_BUFFER       DW      16 DUP(?) ; ROOM FOR 15 ENTRIES
            ????
                ]

                                ;------ HEAD = TAIL INDICATES THAT THE BUFFER IS EMPTY
= 0045                          NUM_KEY         EQU     69      ; SCAN CODE FOR NUMBER LOCK
= 0046                          SCROLL_KEY      EQU     70      ; SCROLL LOCK KEY
= 0038                          ALT_KEY         EQU     56      ; ALTERNATE SHIFT KEY SCAN CODE
= 001D                          CTL_KEY         EQU     29      ; SCAN CODE FOR CONTROL KEY
= 003A                          CAPS_KEY        EQU     58      ; SCAN CODE FOR SHIFT LOCK
= 002A                          LEFT_KEY        EQU     42      ; SCAN CODE FOR LEFT SHIFT
= 0036                          RIGHT_KEY       EQU     54      ; SCAN CODE FOR RIGHT SHIFT
= 0052                          INS_KEY         EQU     82      ; SCAN CODE FOR INSERT KEY
= 0053                          DEL_KEY         EQU     83      ; SCAN CODE FOR DELETE KEY
                                ;--------------------------------------
                                ;       DISKETTE DATA AREAS
                                ;--------------------------------------
003E  ??                        SEEK_STATUS     DB      ?       ; DRIVE RECALIBRATION STATUS
                                                                ; BIT 0 = DRIVE NEEDS RECAL BEFORE
                                                                ;   NEXT SEEK IF BIT IS = 0
003F  ??                        MOTOR_STATUS    DB      ?       ; MOTOR STATUS
                                                                ; BIT 0 = DRIVE 0 IS CURRENTLY
                                                                ;   RUNNING
0040  ??                        MOTOR_COUNT     DB      ?       ; TIME OUT COUNTER FOR DRIVE
                                                                ; TURN OFF
= 0025                          MOTOR_WAIT      EQU     37      ; 2 SECS OF COUNTS FOR MOTOR
                                                                ; TURN OFF
0041  ??                        DISKETTE_STATUS DB      ?       ; RETURN CODE STATUS BYTE
= 0080                          TIME_OUT        EQU     80H     ; ATTACHMENT FAILED TO RESPOND
= 0040                          BAD_SEEK        EQU     40H     ; SEEK OPERATION FAILED
= 0020                          BAD_NEC         EQU     20H     ; NEC CONTROLLER HAS FAILED
= 0010                          BAD_CRC         EQU     10H     ; BAD CRC ON DISKETTE READ
= 0009                          DMA_BOUNDARY    EQU     09H     ; ATTEMPT TO DMA ACROSS 64K
                                                                ; BOUNDARY
= 0008                          BAD_DMA         EQU     08H     ; DMA OVERRUN ON OPERATION
= 0004                          RECORD_NOT_FND  EQU     04H     ; REQUESTED SECTOR NOT FOUND
= 0003                          WRITE_PROTECT   EQU     03H     ; WRITE ATTEMPTED ON WRITE
                                                                ; PROTECTED DISK
= 0002                          BAD_ADDR_MARK   EQU     02H     ; ADDRESS MARK NOT FOUND
= 0001                          BAD_CMD         EQU     01H     ; BAD COMMAND GIVEN TO DISKETTE I/O
0042    07 [                    NEC_STATUS      DB      7 DUP(?) ; STATUS BYTES FROM NEC
            ??
                ]

= 0020                          SEEK_END        EQU     20H
= 012C                          THRESHOLD       EQU     300     ; NUMBER OF TIMER-0 TICKS TILL
                                                                ; ENABLE
= 00AF                          PARM0           EQU     0AFH    ; PARAMETER 0 IN THE DISK_PARM
                                                                ; TABLE
= 0003                          PARM1           EQU     3       ; PARAMETER 1
= 0019                          PARM9           EQU     25      ; PARAMETER 9
= 0004                          PARM10          EQU     4       ; PARAMETER 10
```

```
                                ;--------------------------------------
                                ;       VIDEO DISPLAY DATA AREA
                                ;--------------------------------------
0049  ??                        CRT_MODE        DB      ?       ; CURRENT CRT MODE
004A  ????                      CRT_COLS        DW      ?       ; NUMBER OF COLUMNS ON SCREEN
004C  ????                      CRT_LEN         DW      ?       ; LENGTH OF REGEN IN BYTES
004E  ????                      CRT_START       DW      ?       ; STARTING ADDRESS IN REGEN BUFFER
0050      08 [                  CURSOR_POSN     DW      8 DUP(?) ; CURSOR FOR EACH OF UP TO 8 PAGES
                ????
                    ]

0060  ????                      CURSOR_MODE     DW      ?       ; CURRENT CURSOR MODE SETTING
0062  ??                        ACTIVE_PAGE     DB      ?       ; CURRENT PAGE BEING DISPLAYED
0063  ????                      ADDR_6845       DW      ?       ; BASE ADDRESS FOR ACTIVE DISPLAY
                                                                ; CARD
0065  ??                        CRT_MODE_SET    DB      ?       ; CURRENT SETTING OF THE
                                                                ; CRT MODE REGISTER
0066  ??                        CRT_PALLETTE    DB      ?       ; CURRENT PALETTE MASK SETTING
                                ;--------------------------------------
                                ;       CASSETTE DATA AREA
                                ;--------------------------------------
0067  ????                      EDGE_CNT        DW      ?       ; TIME COUNT AT DATA EDGE
0069  ????                      CRC_REG         DW      ?       ; CRC REGISTER
006B  ??                        LAST_VAL        DB      ?       ; LAST INPUT VALUE
                                ;--------------------------------------
                                ;       TIMER DATA AREA
                                ;--------------------------------------
006C  ????                      TIMER_LOW       DW      ?       ; LOW WORD OF TIMER COUNT
006E  ????                      TIMER_HIGH      DW      ?       ; HIGH WORD OF TIMER COUNT
0070  ??                        TIMER_OFL       DB      ?       ; TIMER HAS ROLLED OVER SINCE LAST
                                                                ; READ
                                ;--------------------------------------
                                ;       SYSTEM DATA AREA
                                ;--------------------------------------
0071  ??                        BIOS_BREAK      DB      ?       ; BIT 7=1 IF BREAK KEY HAS BEEN HIT
0072  ????                      RESET_FLAG      DW      ?       ; WORD=1234H IF KEYBOARD RESET
                                                                ; UNDERWAY
                                ;--------------------------------------
                                ;       EXTRA DISKETTE DATA AREAS
                                ;--------------------------------------
0074  ??                        TRACK0          DB      ?
0075  ??                        TRACK1          DB      ?
0076  ??                        TRACK2          DB      ?
0077  ??                                        DB      ?
                                ;--------------------------------------
                                ;       PRINTER AND RS232 TIME-OUT VARIABLES
                                ;--------------------------------------
0078      04 [                  PRINT_TIM_OUT   DB      4 DUP(?)
                ??
                    ]

007C      04 [                  RS232_TIM_OUT   DB      4 DUP(?)
                ??
                    ]


                                ;--------------------------------------
                                ;       ADDITIONAL KEYBOARD DATA AREA
                                ;--------------------------------------
0080  ????                      BUFFER_START    DW      ?
0082  ????                      BUFFER_END      DW      ?
0084  ??                        INTR_FLAG       DB      ?       ; FLAG TO INDICATE AN INTERRUPT
                                                                ; HAPPENED
                                ;--------------------------------------
                                ;  62 KEY KEYBOARD DATA AREA
                                ;--------------------------------------
0085  ??                        CUR_CHAR        DB      ?       ; CURRENT CHARACTER FOR TYPAMATIC
0086  ??                        VAR_DELAY       DB      ?       ; DETERMINES WHEN INITIAL DELAY IS
                                                                ; OVER
= 000F                          DELAY_RATE      EQU     0FH     ; INCREASES INITIAL DELAY
0087  ??                        CUR_FUNC        DB      ?       ; CURRENT FUNCTION
0088  ??                        KB_FLAG_2       DB      ?       ; 3RD BYTE OF KEYBOARD FLAGS
= 0004                          RANGE           EQU     4       ; NUMBER OF POSITIONS TO SHIFT
                                                                ; DISPLAY
                                ;--------------------------------------
                                ;  BIT ASSIGNMETS FOR KB_FLAG_2
                                ;--------------------------------------
= 0080                          FN_FLAG         EQU     80H
= 0040                          FN_BREAK        EQU     40H
= 0020                          FN_PENDING      EQU     20H
= 0010                          FN_LOCK         EQU     10H
= 0008                          TYPE_OFF        EQU     08H
= 0004                          HALF_RATE       EQU     04H
= 0002                          INIT_DELAY      EQU     02H
= 0001                          PUTCHAR         EQU     01H
0089  ??                        HORZ_POS        DB      ?       ; CURRENT VALUE OF HORIZONTAL
                                                                ; START PARM
008A  ??                        PAGDAT          DB      ?       ; IMAGE OF DATA WRITTEN TO PAGREG
008B                            DATA    ENDS
                                ;--------------------------------------
                                ;       EXTRA DATA AREA
                                ;--------------------------------------
0000                            XXDATA  SEGMENT AT 50H
0000  ??                        STATUS_BYTE     DB      ?
                                ; THE FOLLOWING AREA IS USED ONLY DURING DIAGNOSTICS
                                ;  (POST AND ROM RESIDENT)
0001  ??                        DCP_MENU_PAGE   DB      ?       ; TO CURRENT PAGE FOR DIAG. MENU
0002  ????                      DCP_ROW_COL     DW      ?       ; CURRENT ROW/COLUMN COORDINATES
                                                                ; FOR DIAG MENU
0004  ??                        WRAP_FLAG       DB      ?       ; INTERNAL/EXTERNAL 8250 WRAP
                                                                ; INDICATOR
```

```
0005  ??                    MFG_TST         DB      ?       ; INITIALIZATION FLAG
0006  ????                  MEM_TOT         DW      ?       ; WORD EQUIV. TO HIGHEST SEGMENT IN
                                                            ; MEMORY
0008  ????                  MEM_DONES       DW      ?       ; CURRENT SEGMENT VALUE FOR
                                                            ; BACKGROUND MEM TEST
000A  ????                  MEM_DONEO       DW      ?       ; CURRENT OFFSET VALUE FOR
                                                            ; BACKGROUND MEM TEST
000C  ????                  INT1CO          DW      ?       ; SAVE AREA FOR INTERRUPT 1C
                                                            ; ROUTINE
000E  ????                  INT1CS          DW      ?
0010  ??                    MENU_UP         DB      ?       ; FLAG TO INDICATE WHETHER MENU IS
                                                            ; ON SCREEN (FF=YES, 0=NO)
0011  ??                    DONE128         DB      ?       ; COUNTER TO KEEP TRACK OF 128 BYTE
                                                            ; BLOCKS TESTED BY BGMEM
0012  ????                  KBDONE          DW      ?       ; TOTAL K OF MEMORY THAT HAS BEEN
                                                            ; TESTED BY BACKGROUND MEM TEST
                            ;---------------------------------------
                            ;       POST DATA AREA
                            ;---------------------------------------
0014  ????                  IO_ROM_INIT     DW      ?       ; POINTR TO OPTIONAL I/O ROM INIT
                                                            ; ROUTINE
0016  ????                  IO_ROM_SEG      DW      ?       ; POINTER TO IO ROM SEGMENT
0018  ??                    POST_ERR        DB      ?       ; FLAG TO INDICATE ERROR OCCURRED
                                                            ; DURING POST
0019     09 [               MODEM_BUFFER    DB      9 DUP(?) ; MODEM RESPONSE BUFFER
               ??
                  ]

                                                            ; (MAX 9 CHARS)
0022  ????                  MFG_RTN         DW      ?       ; POINTER TO MFG. OUTPUT ROUTINE
0024  ????                                  DW      ?
                            ;---------------------------------------
                            ;      SERIAL PRINTER DATA
                            ;---------------------------------------
0026  ????                  SP_FLAG         DW      ?
0028  ??                    SP_CHAR         DB      ?
                                                            ; THE FOLLOWING SIX ENTRIES ARE
                                                            ; DATA PERTAINING TO NEW STICK
0029  ????                  NEW_STICK_DATA  DW      ?       ; RIGHT STICK DELAY
002B  ????                                  DW      ?       ; RIGHT BUTTON A DELAY
002D  ????                                  DW      ?       ; RIGHT BUTTON B DELAY
002F  ????                                  DW      ?       ; LEFT STICK DELAY
0031  ????                                  DW      ?       ; LEFT BUTTON A DELAY
0033  ????                                  DW      ?       ; LEFT BUTTON B DELAY
0035  ????                                  DW      ?       ; RIGHT STICK LOCATION
0037  ????                                  DW      ?       ; UNUSED
0039  ????                                  DW      ?       ; UNUSED
003B  ????                                  DW      ?       ; LEFT STICK POSTITON
003D                        XXDATA  ENDS
                            ;---------------------------------------
                            ;      DISKETTE DATA AREA
                            ;---------------------------------------
0000                        DKDATA  SEGMENT AT 60H
0000  ??                    NUM_DRIVE       DB      ?
0001  ??                    DUAL            DB      ?
0002  ??                    OPERATION       DB      ?
0003  ??                    DRIVE           DB      ?
0004  ??                    TRACK           DB      ?
0005  ??                    HEAD            DB      ?
0006  ??                    SECTOR          DB      ?
0007  ??                    NUM_SECTOR      DB      ?
0008  ??                    SEC             DB      ?
                            ;    FORMAT ID
0009     08 [               TK_HD_SC        DB      8 DUP(0,0,0,0)  ;TRACK,HEAD,SECTOR,NUM OF
               00
               00
               00
               00
                  ]

                                                            ; SECTOR
                            ;    BUFFER FOR READ AND WRITE OPERATION
= 0200                      DK_BUF_LEN      EQU     512     ; 512 BYTES/SECTOR
0029  0200 [                READ_BUF        DB      DK_BUF_LEN DUP(0)
               00
                  ]

0229  0100 [                WRITE_BUF       DB      (DK_BUF_LEN/2) DUP(6DH,0BH)
               6D
               0B
                  ]

                            ;    INFO FLAGS
0429  ??                    REQUEST_IN      DB      ?       ;SELECTION CHARACTER
042A  ??                    DK_EXISTED      DB      ?
042B  ??                    DK_FLAG         DB      ?
042C  ????                  RAN_NUM         DW      ?
042E  ????                  SEED            DW      ?
                            ;    SPEED TEST VARIABLES
0430  ????                  DK_SPEED        DW      ?
0432  ????                  TIM_1           DW      ?
0434  ????                  TIM_1_1         DW      ?
0436  ????                  TIM_2           DW      ?
0438  ????                  TIM_1_2         DW      ?
043A  ????                  FRACT_H         DW      ?
043C  ????                  FRACT_L         DW      ?
043E  ????                  PART_CYCLE      DW      ?
0440  ????                  WHOLE_CYCLE     DW      ?
0442  ????                  HALF_CYCLE      DW      ?
```

```
                                ;       ERROR PARAMETERS
0444  ??                        DK_ER_OCCURED   DB      ?       ;ERROR HAS OCCURRED
0445  ??                        DK_ER_L1        DB      ?       ;CUSTOMER ERROR LEVEL
0446  ??                        DK_ER_L2        DB      ?       ;SERVICE ERROR LEVEL
0447  ??                        ER_STATUS_BYTE  DB      ?       ;STATUS BYTE RETURN FROM INT 13H
                                ;       LANGUAGE TABLE
0448  ??                        LANG_BYTE       DB      ?       ; PORT 80 TO DETERMINE WHICH
                                                                ; LANGAGE TO USE
0449                            DKDATA  ENDS
                                ;--------------------------------------
                                ;       VIDEO DISPLAY BUFFER
                                ;--------------------------------------
0000                            VIDEO_RAM       SEGMENT AT 0B800H
0000  4000 [                           DB      16384 DUP(?)
              ??
                 ]

4000                            VIDEO_RAM       ENDS
                                ;--------------------------------------
                                ;       ROM RESIDENT CODE               :
                                ;--------------------------------------
0000                            CODE    SEGMENT PAGE
                                        ASSUME  CS:CODE,DS:ABS0,ES:NOTHING,SS:STACK

0000  31 35 30 34 30 33                 DB      '1504036 COPR. IBM 1981,1983'   ; COPYRIGHT NOTICE
      36 20 43 4F 50 52
      2E 20 49 42 4D 20
      31 39 38 31 2C 31
      39 38 33

001B  0149 R                    Z1      DW      L12             ; RETURN POINTERS FOR RTNS CALLED
001D  0157 R                            DW      L14             ; BEFORE STACK INITIALIZED
001F  016D R                            DW      L16
0021  0186 R                            DW      L19
0023  01BA R                            DW      L24
0025  20 4B 42                 F3B     DB      ' KB'
0028  0A47 R                   EX_0    DW      OFFSET  EB0
002A  0A47 R                            DW      OFFSET  EB0
002C  0ABB R                            DW      OFFSET  TOTLTP0
002E  0AB4 R                    EX1     DW      OFFSET  M01
                                ;--------------------------------------
                                ;       MESSAGE AREA FOR POST           :
                                ;--------------------------------------
0030  45 52 52 4F 52            ERROR_ERR       DB      'ERROR' ;GENERAL ERROR PROMPT
0035  41                        MEM_ERR         DB      'A'     ;MEMORY ERROR
0036  42                        KEY_ERR         DB      'B'     ;KEYBOARD ERROR MSG
0037  43                        CASS_ERR        DB      'C'     ;CASSETTE ERROR MESSAGE
0038  44                        COM1_ERR        DB      'D'     ;ON-BOARD SERIAL PORT ERR. MSG
0039  45                        COM2_ERR        DB      'E'     ;SERIAL PORTION OF MODEM ERROR
003A  46                        ROM_ERR         DB      'F'     ;OPTIONAL GENERIC BIOS ROM ERROR
003B  47                        CART_ERR        DB      'G'     ;CARTRIDGE ERROR
003C  48                        DISK_ERR        DB      'H'     ;DISKETTE ERR
                                ;
003D                            F4      LABEL   WORD            ; PRINTER SOURCE TABLE
003D  0378                              DW      378H
003F  0278                              DW      278H
0041                            F4E     LABEL   WORD
0041                            IMASKS  LABEL       BYTE        ; INTERRUPT MASKS FOR 8259
                                                                ; INTERRUPT CONTROLLER
0041  EF                                DB          0EFH        ; MODEM INTR MASK
0042  F7                                DB          0F7H        ; SERIAL PRINTER INTR MASK
                                ;--------------------------------------
                                ; SETUP                         :
                                ;       DISABLE NMI, MASKABLE INTS.     :
                                ;       SOUND CHIP, AND VIDEO.          :
                                ;       TURN DRIVE 0 MOTOR OFF          :
                                ;--------------------------------------
                                        ASSUME  CS:CODE,DS:ABS0,ES:NOTHING,SS:STACK
0043                            RESET   LABEL   FAR
0043  B0 00                     START:  MOV     AL,0
0045  E6 A0                             OUT     0A0H,AL         ; DISABLES NMI
0047  FE C8                             DEC     AL              ; SEND FF TO MFG_TESTER
0049  E6 10                             OUT     10H,AL
004B  E4 A0                             IN      AL,0A0H         ; RESET NMI F/F
004D  FA                                CLI                     ; DISABLES MASKABLE INTERRUPTS
                                                                ; DISABLE ATTENUATION IN SOUND CHIP
004E  B8 108F                           MOV     AX,108FH        ; REG ADDRESS IN AH, ATTENUATOR OFF
                                                                ; IN AL
0051  BA 00C0                           MOV     DX,00C0H        ; ADDRESS OF SOUND CHIP
0054  B9 0004                           MOV     CX,4            ; 4 ATTENUATORS TO DISABLE
0057  0A C4                     L1:     OR      AL,AH           ; COMBINE REG ADDRESS AND DATA
0059  EE                               OUT     DX,AL
005A  80 C4 20                          ADD     AH,20H          ; POINT TO NEXT REG
005D  E2 F8                             LOOP    L1
005F  B0 A0                             MOV     AL,WD_ENABLE+FDC_RESET  ; TURN DRIVE 0 MOTOR OFF,
                                                                ; ENABLE TIMER
0061  E6 F2                             OUT     0F2H,AL
0063  BA 03DA                           MOV     DX,VGA_CTL      ; VIDEO GATE ARRAY CONTROL
0066  EC                               IN      AL,DX           ; SYNC VGA TO ACCEPT REG
0067  B0 04                             MOV     AL,4            ; SET VGA RESET REG
0069  EE                               OUT     DX,AL           ; SELECT IT
006A  B0 01                             MOV     AL,1            ; SET ASYNC RESET
006C  EE                               OUT     DX,AL           ; RESET VIDEO GATE ARRAY
                                ;--------------------------------------
                                ; TEST 1                        :
                                ;       8088 PROCESSOR TEST             :
                                ; DESCRIPTION                   :
                                ;       VERIFY 8088 FLAGS, REGISTERS    :
                                ;       AND CONDITIONAL JUMPS           :
                                ;                                       :
                                ; MFG. ERROR CODE 0001H          :
                                ;--------------------------------------
```

```
006D  B4 D5                     MOV     AH,0D5H         ; SET SF, CF, ZF, AND AF FLAGS ON
006F  9E                        SAHF
0070  73 4C                     JNC     L4              ; GO TO ERR ROUTINE IF CF NOT SET
0072  75 4A                     JNZ     L4              ; GO TO ERR ROUTINE IF ZF NOT SET
0074  7B 48                     JNP     L4              ; GO TO ERR ROUTINE IF PF NOT SET
0076  79 46                     JNS     L4              ; GO TO ERR ROUTINE IF SF NOT SET
0078  9F                        LAHF                    ; LOAD FLAG IMAGE TO AH
0079  B1 05                     MOV     CL,5            ; LOAD CNT REG WITH SHIFT CNT
007B  D2 EC                     SHR     AH,CL           ; SHIFT AF INTO CARRY BIT POS
007D  73 3F                     JNC     L4              ; GO TO ERR ROUTINE IF AF NOT SET
007F  B0 40                     MOV     AL,40H          ; SET THE OF FLAG ON
0081  D0 E0                     SHL     AL,1            ; SETUP FOR TESTING
0083  71 39                     JNO     L4              ; GO TO ERR ROUTINE IF OF NOT SET
0085  32 E4                     XOR     AH,AH           ; SET AH = 0
0087  9E                        SAHF                    ; CLEAR SF, CF, ZF, AND PF
0088  76 34                     JBE     L4              ; GO TO ERR ROUTINE IF CF ON
                                                        ; GO TO ERR ROUTINE IF ZF ON
008A  78 32                     JS      L4              ; GO TO ERR ROUTINE IF SF ON
008C  7A 30                     JP      L4              ; GO TO ERR ROUTINE IF PF ON
008E  9F                        LAHF                    ; LOAD FLAG IMAGE TO AH
008F  B1 05                     MOV     CL,5            ; LOAD CNT REG WITH SHIFT CNT
0091  D2 EC                     SHR     AH,CL           ; SHIFT 'AF' INTO CARRY BIT POS
0093  72 29                     JC      L4              ; GO TO ERR ROUTINE IF ON
0095  D0 E4                     SHL     AH,1            ; CHECK THAT 'OF' IS CLEAR
0097  70 25                     JO      L4              ; GO TO ERR ROUTINE IF ON
                        ;----- READ/WRITE THE 8088 GENERAL AND SEGMENTATION REGISTERS
                        ;       WITH ALL ONE'S AND ZEROES'S.
0099  B8 FFFF                   MOV     AX,0FFFFH       ; SETUP ONE'S PATTERN IN AX
009C  F9                        STC
009D  8E D8             L2:     MOV     DS,AX           ; WRITE PATTERN TO ALL REGS
009F  8C DB                     MOV     BX,DS
00A1  8E C3                     MOV     ES,BX
00A3  8C C1                     MOV     CX,ES
00A5  8E D1                     MOV     SS,CX
00A7  8C D2                     MOV     DX,SS
00A9  8B E2                     MOV     SP,DX
00AB  8B EC                     MOV     BP,SP
00AD  8B F5                     MOV     SI,BP
00AF  8B FE                     MOV     DI,SI
00B1  73 07                     JNC     L3
00B3  33 C7                     XOR     AX,DI           ; PATTERN MAKE IT THRU ALL REGS
00B5  75 07                     JNZ     L4              ; NO - GO TO ERR ROUTINE
00B7  F8                        CLC
00B8  EB E3                     JMP     L2
00BA  0B C7             L3:     OR      AX,DI           ; ZERO PATTERN MAKE IT THRU?
00BC  74 0C                     JZ      L5              ; YES - GO TO NEXT TEST
00BE  BA 0010           L4:     MOV     DX,0010H        ; HANDLE ERROR
00C1  B0 00                     MOV     AL,0
00C3  EE                        OUT     DX,AL           ; ERROR 0001
00C4  42                        INC     DX
00C5  EE                        OUT     DX,AL
00C6  FE C0                     INC     AL
00C8  EE                        OUT     DX,AL
00C9  F4                        HLT                     ; HALT
00CA                    L5:
                        ;--------------------------------------
                        ; TEST 2                             :
                        ;       8255 INITIALIZATION AND TEST :
                        ; DESCRIPTION                        :
                        ;       FIRST INITIALIZE 8255 PROG.  :
                        ;       PERIPHERAL INTERFACE. PORTS A&B :
                        ;       ARE LATCHED OUTPUT           :
                        ;       BUFFERS. C IS INPUT.         :
                        ; MFG. ERR. CODE =0002H              :
                        ;--------------------------------------
00CA  B0 FE                     MOV     AL,0FEH         ; SEND FE TO MFG
00CC  E6 10                     OUT     10H,AL
00CE  B0 89                     MOV     AL,MODE_8255
00D0  E6 63                     OUT     CMD_PORT,AL     ; CONFIGURES I/O PORTS
00D2  2B C0                     SUB     AX,AX           ; TEST PATTERN SEED = 0000
00D4  8A C4             L6:     MOV     AL,AH
00D6  E6 60                     OUT     PORT_A,AL       ; WRITE PATTERN TO PORT A
00D8  E4 60                     IN      AL,PORT_A       ; READ PATTERN FROM PORT A
00DA  E6 61                     OUT     PORT_B,AL       ; WRITE PATTERN TO PORT B
00DC  E4 61                     IN      AL,PORT_B       ; READ OUTPUT PORT
00DE  3A C4                     CMP     AL,AH           ; DATA AS EXPECTED?
00E0  75 06                     JNE     L7              ; IF NOT, SOMETHING IS WRONG
00E2  FE C4                     INC     AH              ; MAKE NEW DATA PATTERN
00E4  75 EE                     JNZ     L6              ; LOOP TILL 255 PATTERNS DONE
00E6  EB 05                     JMP     SHORT L8        ; CONTINUE IF DONE
00E8  B3 02             L7:     MOV     BL,02H          ; SET ERROR FLAG (BH=00 NOW)
00EA  E9 09BC R                 JMP     E_MSG           ; GO ERROR ROUTINE
00ED  32 C0             L8:     XOR     AL,AL           ;
00EF  E6 60                     OUT     KBPORT,AL       ; CLEAR KB PORT
00F1  E4 62                     IN      AL,PORT_C       ;
00F3  24 08                     AND     AL,00001000B    ; 64K CARD PRESENT?
00F5  B0 1B                     MOV     AL,1BH          ; PORT SETTING FOR 64K SYS
00F7  75 02                     JNZ     L9              ;
00F9  B0 3F                     MOV     AL,3FH          ; PORT SETTING FOR 128K SYS
00FB  BA 03DF           L9:     MOV     DX,PAGREG       ;
00FE  EE                        OUT     DX,AL           ;
00FF  B0 0D                     MOV     AL,00001101B            INITIALIZE OUTPUT PORTS
0101  E6 61                     OUT     PORT_B,AL
```

```
                        ;--------------------------------------------------------
                        ; PART 3
                        ;       SET UP VIDEO GATE ARRAY AND 6845 TO GET MEMORY WORKING
                        ;
                        ;--------------------------------------------------------
0103  B0 FD                     MOV     AL,0FDH
0105  E6 10                     OUT     10H,AL          ;
0107  BA 03D4                   MOV     DX,03D4H        ; SET ADDRESS OF 6845
010A  BB F0A4 R                 MOV     BX,OFFSET VIDEO_PARMS ; POINT TO 6845 PARMS
010D  B9 0010 90                MOV     CX,M0040        ; SET PARM LEN
0111  32 E4                     XOR     AH,AH           ; AH IS REG #
0113  8A C4             L10:    MOV     AL,AH           ; GET 6845 REG #
0115  EE                        OUT     DX,AL
0116  42                       INC     DX              ; POINT TO DATA PORT
0117  FE C4                    INC     AH              ; NEXT REG VALUE
0119  2E: 8A 07                 MOV     AL,CS:[BX]      ; GET TABLE VALUE
011C  EE                        OUT     DX,AL           ; OUT TO CHIP
011D  43                       INC     BX              ; NEXT IN TABLE
011E  4A                       DEC     DX              ; BACK TO POINTER REG
011F  E2 F2                     LOOP    L10
                            START VGA WITHOUT VIDEO ENABLED
0121  BA 03DA                   MOV     DX,VGA_CTL      ; SET ADDRESS OF VGA
0124  EC                        IN      AL,DX           ; BE SURE ADDR/DATA FLAG IS
                                                        ; IN THE PROPER STATE
0125  B9 0005                   MOV     CX,5            ; # OF REGISTERS
0128  32 E4                     XOR     AH,AH           ; AH IS REG COUNTER
012A  8A C4             L11:    MOV     AL,AH           ; GET REG #
012C  EE                        OUT     DX,AL           ; SELECT IT
012D  32 C0                     XOR     AL,AL           ; SET ZERO FOR DATA
012F  EE                        OUT     DX,AL
0130  FE C4                     INC     AH              ; NEXT REG
0132  E2 F6                     LOOP    L11
                        ;--------------------------------------------------
                        ; TEST 4                                         :
                        ;       PLANAR BOARD ROS CHECKSUM TEST           :
                        ; DESCRIPTION                                    :
                        ;       A CHECKSUM TEST IS DONE FOR EACH ROS     :
                        ;       MODULE ON THE PLANAR BOARD TO.           :
                        ;       MFG ERROR CODE =0003H MODULE AT ADDRESS  :
                        ;                       F000:0000 ERROR          :
                        ;                       0004H MODULE AT ADDRESS  :
                        ;                       F800:0000 ERROR          ;
                        ;--------------------------------------------------
0134  B0 FC                     MOV     AL,0FCH
0136  E6 10                     OUT     10H,AL          ; MFG OUT=FC
                        ; CHECK MODULE AT F000:0 (LENGTH 32K)
0138  33 F6                     XOR     SI,SI           ; INDEX OFFSET WITHIN SEGMENT OF
                                                        ; FIRST BYTE
013A  8C C8                     MOV     AX,CS           ; SET UP STACK SEGMENT
013C  8E D0                     MOV     SS,AX
013E  8E D8                     MOV     DS,AX           ; LOAD DS WITH SEGMENT OF ADDRESS
                                                        ; SPACE OF BIOS/BASIC
0140  B9 8000                   MOV     CX,8000H        ; NUMBER OF BYTES TO BE TESTED, 32K
0143  BC 001B R                 MOV     SP,OFFSET Z1    ; SET UP STACK POINTER SO THAT
                                                        ; RETURN WILL COME HERE
0146  E9 FEEB R                 JMP     ROS_CHECKSUM    ; JUMP TO ROUTINE WHICK PERFORMS
                                                        ; CRC CHECK
0149  74 06             L12:    JZ      L13             ; MODULE AT F000:0 OK, GO CHECK
                                                        ; OTHER MODULE AT F000:8000
014B  BB 0003                   MOV     BX,0003H        ; SET ERROR CODE
014E  E9 09BC R                 JMP     E_MSG           ; INDICATE ERROR
0151  B9 8000           L13:    MOV     CX,8000H        ; LOAD COUNT (SI POINTING TO START
0154  E9 FEEB R                 JMP     ROS_CHECKSUM    ; OF NEXT MODULE AT THIS POINT)
0157  74 06             L14:    JZ      L15             ; PROCEED IF NO ERROR
0159  BB 0004                   MOV     BX,0004H        ; INDICATE ERROR
015C  E9 09BC R                 JMP     E_MSG           ;
015F                    L15:
                        ;--------------------------------------------------
                        ; TEST 5
                        ;       BASE  2K READ/WRITE STORAGE TEST
                        ; DESCRIPTION
                        ;       WRITE/READ/VERIFY DATA PATTERNS
                        ;       AA,55, AND 00 TO 1ST 2K OF STORAGE
                        ;       AND THE 2K JUST BELOW 64K (CRT BUFFER)
                        ;       VERIFY STORAGE ADDRESSABILITY.
                        ;       ON EXIT SET CRT PAGE TO 3. SET
                        ;       TEMPORARY STACK ALSO.
                        ;  MFG. ERROR CODE 04XX FOR SYSTEM BOARD MEM.
                        ;                   05XX FOR 64K ATTRIB. CD. MEM
                        ;                   06XX FOR ERRORS IN BOTH
                        ;                   (XX= ERROR BITS)
                        ;--------------------------------------------------:
015F  B0 FB                     MOV     AL,0FBH
0161  E6 10                     OUT     10H,AL          ; SET MFG FLAG=FB
0163  B9 0400                   MOV     CX,0400H        ; SET FOR 1K WORDS, 2K BYTES
0166  33 C0                     XOR     AX,AX
0168  8E C0                     MOV     ES,AX           ; LOAD ES WITH 0000 SEGMENT
016A  E9 0B59 R                 JMP     PODSTG
016D  75 19             L16:    JNZ     L20             ; BAD STORAGE FOUND
016F  B0 FA                     MOV     AL,0FAH         ; MFG OUT=FA
0171  E6 10                     OUT     10H,AL
0173  B9 0400                   MOV     CX,400H         ; 1024 WORDS TO BE TESTED IN THE
                                                        ; REGEN BUFFER
0176  E4 60                     IN      AL,PORT_A       ; WHERE IS THE REGEN BUFFER?
0178  3C 1B                     CMP     AL,1BH          ; TOP OF 64K?
017A  B8 0F80                   MOV     AX,0F80H        ; SET POINTER TO THERE IF IT IS
017D  74 02                     JE      L18
017F  B4 1F                     MOV     AH,1FH          ; OR SET POINTER TO TOP OF 128K
0181  8E C0             L18:    MOV     ES,AX
0183  E9 0B59 R                 JMP     PODSTG
0186  74 23             L19:    JZ      L23
```

```
0188  B7 04                   L20:    MOV     BH,04H          ; ERROR 04....
018A  E4 62                             IN      AL,PORT_C       ; GET CONFIG BITS
018C  24 08                             AND     AL,00001000B    ; TEST FOR ATTRIB CARD PRESENT
018E  74 06                             JZ      L21             ; WORRY ABOUT ODD/EVEN IF IT IS
0190  8A D9                             MOV     BL,CL
0192  0A DD                             OR      BL,CH           ; COMBINE ERROR BITS IF IT ISN'T
0194  EB 12                             JMP     SHORT L22       ;
0196  80 FC 02                 L21:    CMP     AH,02           ; EVEN BYTE ERROR? ERR 04XX
0199  8A D9                             MOV     BL,CL
019B  74 08                             JE      L22
019D  FE C7                             INC     BH              ; MAKE INTO 05XX ERR
019F  0A DD                             OR      BL,CH           ; MOVE AND POSSIBLY COMBINE
                                                                ; ERROR BITS
01A1  80 FC 01                          CMP     AH,1            ; ODD BYTE ERROR
01A4  74 02                             JE      L22
01A6  FE C7                             INC     BH              ; MUST HAVE BEEN BOTH
                                                                ; - MAKE INTO 06XX
01A8  E9 09BC R                 L22:    JMP     E_MSG           ; JUMP TO ERROR OUTPUT ROUTINE
                                ; RETEST HIGH 2K USING B8000 ADDRESS PATH
01AB  B0 F9                     L23:    MOV     AL,0F9H         ; MFG OUT =F9
01AD  E6 10                             OUT     10H,AL
01AF  B9 0400                           MOV     CX,0400H        ; 1K WORDS
01B2  B8 B880                           MOV     AX,0B880H       ; POINT TO AREA JUST TESTED WITH
                                                                ; DIRECT ADDRESSING
01B5  8E C0                             MOV     ES,AX
01B7  E9 0B59 R                         JMP     PODSTG
01BA  74 06                     L24:    JZ      L25
01BC  B8 0005                           MOV     BX,0005H        ; ERROR 0005
01BF  E9 09BC R                         JMP     E_MSG
                                ;----- SETUP STACK SEG AND SP
01C2  B8 0030                   L25:    MOV     AX,0030H        ; GET STACK VALUE
01C5  8E D0                             MOV     SS,AX           ; SET THE STACK UP
01C7  BC 0100 R                         MOV     SP,OFFSET TOS   ; STACK IS READY TO GO
01CA  33 C0                             XOR     AX,AX           ; SET UP DATA SEG
01CC  8E D8                             MOV     DS,AX
                                ;----- SETUP CRT PAGE
01CE  C7 06 0462 R 0007                 MOV     DATA_WORD[ACTIVE_PAGE-DATA],07
                                ;----- SET PRELIMINARY MEMORY SIZE WORD
01D4  BB 0040                           MOV     BX,64
01D7  E4 62                             IN      AL,PORT_C       ;
01D9  24 08                             AND     AL,08H          ; 64K CARD PRESENT?
01DB  B0 1B                             MOV     AL,1BH          ; PORT SETTING FOR 64K SYSTEM
01DD  75 05                             JNZ     L26             ; SET TO 64K IF NOT
01DF  83 C3 40                          ADD     BX,64           ; ELSE SET FOR 128K
01E2  B0 3F                             MOV     AL,3FH          ; PORT SETTING FOR 128K SYSTEM
01E4  89 1E 0415 R              L26:    MOV     DATA_WORD[TRUE_MEM-DATA],BX
01E8  A2 048A R                         MOV     DATA_AREA[PAGDAT-DATA],AL
                                ;--------------------------------------------------------
                                ; PART 6
                                ;       INTERRUPTS
                                ; DESCRIPTION
                                ;       32 INTERRUPTS ARE INITIALIZED TO POINT TO A
                                ;       DUMMY HANDLER. THE BIOS INTERRUPTS ARE LOADED.
                                ;       DIAGNOSTIC INTERRUPTS ARE LOADED
                                ;       SYSTEM CONFIGURATION WORD IS PUT IN MEMORY.
                                ;       THE DUMMY INTERRUPT HANDLER RESIDES HERE.
                                ;--------------------------------------------------------
                                        ASSUME  DS:XXDATA
01EB  B8  ---- R                        MOV     AX,XXDATA
01EE  8E D8                             MOV     DS,AX
01F0  C6 06 0005 R F8                   MOV     MFG_TST,0F8H    ; SET UP MFG CHECKPOINT FROM THIS
                                                                ; POINT
01F5  E8 E6D8 R                         CALL    MFG_UP          ; UPDATE MFG CHECKPOINT
01F8  C7 06 0022 R 0A61 R               MOV     MFG_RTN,OFFSET MFG_OUT
01FE  8C C8                             MOV     AX,CS
0200  A3 0024 R                         MOV     MFG_RTN+2,AX    ; SET DOUBLEWORD POINTER TO MFG.
                                                                ; ERROR OUTPUT ROUTINE SO DIAGS.
                                                                ; DON'T HAVE TO DUPLICATE CODE
                                        ASSUME  CS:CODE,DS:ABS0
0203  B8 0000                           MOV     AX,0
0206  8E D8                             MOV     DS,AX
                                ;----- SET UP THE INTERRUPT VECTORS TO TEMP INTERRUPT
0208  B9 00FF                           MOV     CX,255          ; FILL ALL INTERRUPTS
020B  2B FF                             SUB     DI,DI           ; FIRST INTERRUPT LOCATION IS 0000
020D  8E C7                             MOV     ES,DI           ; SET ES=0000 ALSO
020F  B8 F815 R                 D3:     MOV     AX,OFFSET D11   ; MOVE ADDR OF INTR PROC TO TBL
0212  AB                                STOSW
0213  8C C8                             MOV     AX,CS           ; GET ADDR OF INTR PROC SEG
0215  AB                                STOSW
0216  E2 F7                             LOOP    D3              ; VECTBLO
0218  C7 06 0124 R 109D R               MOV     EXST,OFFSET EXTAB ; SET UP EXT. SCAN TABLE
                                ; SET UP BIOS INTERRUPTS
021E  BF 0040 R                         MOV     DI,OFFSET VIDEO_INT ; SET UP VIDEO INT
0221  0E                                PUSH    CS
0222  1F                                POP     DS              ; PLACE CS IN DS
0223  BE FF03 R                         MOV     SI,OFFSET VECTOR_TABLE+16
0226  B9 0010                           MOV     CX,16
0229  A5                       D4:     MOVSW                   ; MOVE INTERRUPT VECTOR TO LOW
                                                                ; MEMORY
022A  47                                INC     DI
022B  47                                INC     DI              ; POINT TO NEXT VECTOR ENTRY
022C  E2 FB                             LOOP    D4              ; REPEAT FOR ALL 16 BIOS INTERRUPTS
                                ; SET UP DIAGNOSTIC INTERRUPTS
022E  BF 0200                           MOV     DI,0200H        ; START WITH INT. 80H
0231  BE 4000                           MOV     SI,DIAG_TABLE_PTR ; POINT TO ENTRY POINT TABLE
0234  B9 0010                           MOV     CX,16           ; 16 ENTRIES
0237  A5                       D5:     MOVSW                   ; MOVE INTERRUPT VECTOR TO LOW
                                                                ; MEMORY
```

```
0238  47                            INC     DI
0239  47                            INC     DI                ; POINT TO NEXT VECTOR ENTRY
023A  E2 FB                         LOOP    D5                ; REPEAT FOR ALL 16 BIOS INTERRUPTS
023C  8E D9                         MOV     DS,CX             ; SET DS TO ZERO
023E  C7 06 0204 R 1B63 R           MOV     INT81,OFFSET LOCATE1
0244  C7 06 0208 R 1A2A R           MOV     INT82,OFFSET PRNT3
024A  C7 06 0224 R 1BA5 R           MOV     INT89,OFFSET JOYSTICK

                              ;----- SET UP DEFAULT EQUIPMENT DETERMINATION WORD
                              ;       BIT 15,14 = NUMBER OF PRINTERS ATTACHED
                              ;       BIT 13 = 1 = SERIAL PRINTER PRESENT
                              ;       BIT 12 = GAME I/O ATTACHED
                              ;       BIT 11,10,9 = NUMBER OF RS232 CARDS ATTACHED
                              ;       BIT 8 = DMA (0=DMA PRESENT, 1=NO DMA ON SYSTEM
                              ;       BIT 7,6 = NUMBER OF DISKETTE DRIVES
                              ;               00=1, 01=2, 10=3, 11=4 ONLY IF BIT 0 = 1
                              ;       BIT 5,4 = INITIAL VIDEO MODE
                              ;                       00 - UNUSED
                              ;                       01 - 40X25 BW USING COLOR CARD
                              ;                       10 - 80X25 BW USING COLOR CARD
                              ;                       11 - 80X25 BW USING BW CARD
                              ;       BIT 3,2 = PLANAR RAM SIZE (10=48K,11=64K)
                              ;       BIT 1 NOT USED
                              ;       BIT 0 = 1 (IPL DISKETTE INSTALLED)
                              ;--------------------------------------------
                                      ASSUME  CS:CODE,DS:ABS0
0250  BB 1118                         MOV     BX,1118H          ;DEFAULT GAME10,40X25,NO DMA,48K ON
                                                                ; PLANAR
0253  E4 62                           IN      AL,PORT_C
0255  24 08                           AND     AL,08H            ; 64K CARD PRESENT
0257  75 03                           JNZ     D55               ; NO, JUMP
0259  80 CB 04                        OR      BL,4              ; SET 64K ON PLANAR
025C  89 1E 0410 R                D55:  MOV     DATA_WORD[EQUIP_FLAG-DATA],BX
                              ;--------------------------------------------------------
                              ; TEST 7
                              ;       INITIALIZE AND TEST THE 8259 INTERRUPT CONTROLLER CHIP
                              ; MFG ERR. CODE 07XX (XX=00, DATA PATH OR INERNAL FAILURE,
                              ;       XX=ANY OTHER BITS ON=UNEPECTED INTERRUPTS
                              ;--------------------------------------------------------
0260  E8 E6D8 R                        CALL    MFG_UP            ; MFG CODE=F7
                                      ASSUME  DS:ABS0,CS:CODE
0263  B0 13                           MOV     AL,13H            ; ICW1 - RESET EDGE SENSE CIRCUIT,
                                                                ;SET SINGLE 8259 CHIP AND ICW4 READ
0265  E6 20                           OUT     INTA00,AL
0267  B0 08                           MOV     AL,8              ; ICW2 - SET INTERRUPT TYPE 8 (8-F)
0269  E6 21                           OUT     INTA01,AL
026B  B0 09                           MOV     AL,9              ; ICW4 - SET BUFFERED MODE/SLAVE
                                                                ;   AND 8086 MODE
026D  E6 21                           OUT     INTA01,AL
                              ;--------------------------------------------------------
                              ;       TEST ABILITY TO WRITE/READ THE MASK REGISTER
                              ;--------------------------------------------------------
026F  B0 00                           MOV     AL,0              ; WRITE ZEROES TO IMR
0271  8A D8                           MOV     BL,AL             ; PRESET ERROR INDICATOR
0273  E6 21                           OUT     INTA01,AL         ; DEVICE INTERRUPTS ENABLED
0275  E4 21                           IN      AL,INTA01         ; READ IMR
0277  0A C0                           OR      AL,AL             ; IMR = 0?
0279  75 18                           JNZ     GERROR            ; NO - GO TO ERROR ROUTINE
027B  B0 FF                           MOV     AL,0FFH           ; DISABLE DEVICE INTERRUPTS
027D  E6 21                           OUT     INTA01,AL         ; WRITE ONES TO IMR
027F  E4 21                           IN      AL,INTA01         ; READ IMR
0281  04 01                           ADD     AL,1              ; ALL IMR BITS ON?
                                                                ; (ADD SHOULD PRODUCE 0)
0283  75 0E                           JNZ     GERROR            ; NO - GO TO ERROR ROUTINE
                              ;--------------------------------------------------------
                              ;       CHECK FOR HOT INTERRUPTS
                              ;--------------------------------------------------------
                              ;       INTERRUPTS ARE MASKED OFF.  NO INTERRUPTS SHOULD OCCUR.
0285  FB                            STI                       ; ENABLE EXTERNAL INTERRUPTS
0286  B9 0050                       MOV     CX,50H
0289  E2 FE                  HOT1:  LOOP    HOT1              ; WAIT FOR ANY INTERRUPTS
028B  8A 1E 0484 R                  MOV     BL,DATA_AREA[INTR_FLAG-DATA]  ; DID ANY INTERRUPTS
                                                                ; OCCUR?
028F  0A DB                           OR      BL,BL
0291  74 05                           JZ      END_TESTG         ; NO - GO TO NEXT TEST
0293  B7 07                    GERROR: MOV     BH,07H            ; SET 07 SECTION OF ERROR MSG
0295  E9 09BC R                       JMP     E_MSG
0298                           END_TESTG:
                              ; FIRE THE DISKETTE WATCHDOG TIMER
029A  B0 E0                           MOV     AL,WD_ENABLE+WD_STROBE+FDC_RESET
029A  E6 F2                           OUT     0F2H,AL
029C  B0 A0                           MOV     AL,WD_ENABLE+FDC_RESET
029E  E6 F2                           OUT     0F2H,AL
                                      ASSUME  CS:CODE,DS:ABS0
                              ;--------------------------------------------------------------------
                              ;       8253 TIMER CHECKOUT
                              ; DESCRIPTION
                              ;       VERIFY THAT THE TIMERS (0, 1, AND 2) FUNCTION PROPERLY.
                              ;       THIS INCLUDES CHECKING FOR STUCK BITS IN ALL THE TIMERS,
                              ;       THAT TIMER 1 RESPONDS TO TIMER 0 OUTPUTS, THAT TIMER 0
                              ;       INTERRUPTS WHEN IT SHOULD, AND THAT TIMER 2'S OUTPUT WORKS
                              ;       AS IT SHOULD.
                              ;       THERE ARE 7 POSSIBLE ERRORS DURING THIS CHECKOUT.
                              ;       BL VALUES FOR THE CALL TO E_MSG INCLUDE:
                              ;       0)  STUCK BITS IN TIMER 0
                              ;       1)  TIMER 1 DOES NOT RESPOND TO TIMER 0 OUTPUT
                              ;       2)  TIMER 0 INTERRUPT DOES NOT OCCUR
                              ;       3)  STUCK BITS IN TIMER 1
                              ;       4)  TIMER 2 OUTPUT INITIAL VALUE IS NOT LOW
                              ;       5)  STUCK BITS IN TIMER 2
                              ;       6)  TIMER 2 OUTPUT DOES NOT GO HIGH ON TERMINAL COUNT
```

```
                        ;--------------------------------------------------------------------------
                        ;       INITIALIZE TIMER 1 AND TIMER 0 FOR TEST
                        ;--------------------------------------------------------------------------
02A0  E8 E6D8 R                 CALL    MFG_UP          ; MFG CKPOINT=F6
02A3  B8 0176                   MOV     AX,0176H        ; SET TIMER 1 TO MODE 3 BINARY
02A6  BB FFFF                   MOV     BX,OFFFFH       ; INITIAL COUNT OF FFFF
02A9  E8 FFE0 R                 CALL    INIT_TIMER      ; INITIALIZE TIMER 1
02AC  B8 0036                   MOV     AX,0036H        ; SET TIMER 0 TO MODE 3 BINARY
                                                        ; INITIAL COUNT OF FFFF
02AF  E8 FFE0 R                 CALL    INIT_TIMER      ; INITIALIZE TIMER 0
                        ;--------------------------------------------------------------------------
                        ;       SET BIT 5 OF PORT A0 SO TIMER 1 CLOCK WILL BE PULSED BY THE
                        ;       TIMER 0 OUTPUT RATHER THAN THE SYSTEM CLOCK.
                        ;--------------------------------------------------------------------------
02B2  B0 20                     MOV     AL,00100000B
02B4  E6 A0                     OUT     0A0H,AL
                        ;--------------------------------------------------------------------------
                        ;       CHECK IF ALL BITS GO ON AND OFF IN TIMER 0 (CHECK FOR STUCK
                        ;         BITS)
                        ;--------------------------------------------------------------------------
02B6  B4 00                     MOV     AH,0            ; TIMER 0
02B8  E8 036C R                 CALL    BITS_ON_OFF     ; LET SUBROUTINE CHECK IT
02BB  73 05                     JNB     TIMER1_NZ       ;NO STUCK BITS (CARRY FLAG NOT SET)
02BD  B3 00                     MOV     BL,0            ; STUCK BITS IN TIMER 0
02BF  E9 0362 R                 JMP     TIMER_ERROR
                        ;--------------------------------------------------------------------------
                        ;       SINCE TIMER 0 HAS COMPLETED AT LEAST ONE COMPLETE CYCLE,
                        ;       TIMER 1 SHOULD BE NON-ZERO.  CHECK THAT THIS IS THE CASE.
                        ;--------------------------------------------------------------------------
02C2                    TIMER1_NZ:
02C2  E4 41                     IN      AL,TIMER+1      ; READ LSB OF TIMER 1
02C4  8A E0                     MOV     AH,AL           ; SAVE LSB
02C6  E4 41                     IN      AL,TIMER+1      ; READ MSB OF TIMER 1
02C8  3D FFFF                   CMP     AX,0FFFFH       ; STILL FFFF?
02CB  75 05                     JNE     TIMER0_INTR     ; NO - TIMER 1 HAS BEEN BUMPED
02CD  B3 01                     MOV     BL,1            ; TIMER 1 WAS NOT BUMPED BY TIMER 0
02CF  E9 0362 R                 JMP     TIMER_ERROR
                        ;--------------------------------------------------------------------------
                        ;       CHECK FOR TIMER 0 INTERRUPT
                        ;--------------------------------------------------------------------------
02D2                    TIMER0_INTR:
02D2  FB                        STI                     ; ENABLE MASKABLE EXT INTERRUPTS
02D3  E4 21                     IN      AL,INTA01
02D5  24 FE                     AND     AL,0FEH         ; MASK ALL INTRS EXCEPT LVL 0
02D7  20 06 0484 R              AND     DATA_AREA[INTR_FLAG-DATA],AL ; CLEAR INT RECEIVED
02DB  E6 21                     OUT     INTA01,AL       ; WRITE THE 8259 IMR
02DD  B9 FFFF                   MOV     CX,0FFFFH       ; SET LOOP COUNT
02E0                    WAIT_INTR_LOOP:
02E0  F6 06 0484 R 01           TEST    DATA_AREA[INTR_FLAG-DATA],1  ; TIMER 0 INT OCCUR?
02E5  75 06                     JNE     RESET_INTRS     ; YES - CONTINUE
02E7  E2 F7                     LOOP    WAIT_INTR_LOOP  ; WAIT FOR INTR FOR SPECIFIED TIME
02E9  B3 02                     MOV     BL,2            ; TIMER 0 INTR DIDN'T OCCUR
02EB  EB 75                     JMP     SHORT TIMER_ERROR
                        ;--------------------------------------------------------------------------
                        ;       HOUSEKEEPING FOR TIMER 0 INTERRUPTS
                        ;--------------------------------------------------------------------------
02ED                    RESET_INTRS:
02ED  FA                        CLI
                        ; SET TIMER INT. TO POINT TO MFG. HEARTBEAT ROUTINE IF IN MFG MODE
02EE  BA 0201                   MOV     DX,201H
02F1  EC                        IN      AL,DX           ; GET MFG. BITS
02F2  24 F0                     AND     AL,0F0H
02F4  3C 10                     CMP     AL,10H          ; SYS TEST MODE?
02F6  74 04                     JE      D6
02F8  0A C0                     OR      AL,AL           ; OR BURN-IN MODE
02FA  75 11                     JNZ     TIME_1
02FC  C7 06 0020 R 188D R   D6: MOV     INT_PTR,OFFSET MFG_TICK ; SET TO POINT TO MFG.
                                                        ; ROUTINE
0302  C7 06 0070 R 188D R       MOV     INT1C_PTR,OFFSET MFG_TICK ; ALSO SET USER TIMER INT
                                                        ; FOR DIAGS. USE
0308  B0 FE                     MOV     AL,0FEH
030A  E6 21                     OUT     INTA01,AL
030C  FB                        STI
                        ;--------------------------------------------------------------------------
                        ;       RESET D5  OF PORT A0 SO THAT THE TIMER 1 CLOCK WILL BE
                        ;       PULSED BY THE SYSTEM CLOCK.
                        ;--------------------------------------------------------------------------
030D  B0 00             TIME_1: MOV     AL,0            ; MAKE AL = 00
030F  E6 A0                     OUT     0A0H,AL
                        ;--------------------------------------------------------------------------
                        ;       CHECK FOR STUCK BITS IN TIMER 1
                        ;--------------------------------------------------------------------------
0311  B4 01                     MOV     AH,1            ; TIMER 1
0313  E8 036C R                 CALL    BITS_ON_OFF
0316  73 04                     JNB     TIMER2_INIT     ; NO STUCK BITS
0318  B3 03                     MOV     BL,3            ; STUCK BITS IN TIMER 1
031A  EB 46                     JMP     SHORT TIMER_ERROR
                        ;--------------------------------------------------------------------------
                        ;       INITIALIZE TIMER 2
                        ;--------------------------------------------------------------------------
031C                    TIMER2_INIT:
031C  B8 02B6                   MOV     AX,02B6H        ; SET TIMER 2 TO MODE 3 BINARY
031F  BB FFFF                   MOV     BX,OFFFFH       ; INITIAL COUNT
0322  E8 FFE0 R                 CALL    INIT_TIMER
                        ;--------------------------------------------------------------------------
                        ;       SET PB0 OF PORT_B OF 8255 (TIMER 2 GATE)
                        ;--------------------------------------------------------------------------
0325  E4 61                     IN      AL,PORT_B       ; CURRENT STATUS
0327  0C 01                     OR      AL,00000001B    ; SET BIT 0 - LEAVE OTHERS ALONE
0329  E6 61                     OUT     PORT_B,AL
```

```
                        ;--------------------------------------------------------------------------
                        ;       CHECK FOR STUCK BITS IN TIMER 2
                        ;--------------------------------------------------------------------------
032B  B4 02                     MOV     AH,2            ; TIMER 2
032D  E8 036C R                 CALL    BITS_ON_OFF
0330  73 04                     JNB     REINIT_T2       ; NO STUCK BITS
0332  B3 05                     MOV     BL,5            ; STUCK BITS IN TIMER 2
0334  EB 2C                     JMP     SHORT TIMER_ERROR
                        ;--------------------------------------------------------------------------
                        ;       RE_INITIALIZE TIMER 2 WITH MODE 0 AND A SHORT COUNT
                        ;--------------------------------------------------------------------------
0336                    REINIT_T2:
                        ; DROP GATE TO TIMER 2
0336  E4 61                     IN      AL,PORT_B       ; CURRENT STATUS
0338  24 FE                     AND     AL,11111110B    ; RESET BIT 0 - LEAVE OTHERS ALONE
033A  E6 61                     OUT     PORT_B,AL
033C  B8 02B0                   MOV     AX,02B0H        ; SET TIMER 2 TO MODE 0 BINARY
033F  BB 000A                   MOV     BX,000AH        ; INITIAL COUNT OF 10
0342  E8 FFE0 R                 CALL    INIT_TIMER
                        ;--------------------------------------------------------------------------
                        ;       CHECK PC5 OF PORT_C OF 8255 TO SEE IF THE OUTPUT OF TIMER 2
                        ;       IS LOW
                        ;--------------------------------------------------------------------------
0345  E4 62                     IN      AL,PORT_C       ; CURRENT STATUS
0347  24 20                     AND     AL,00100000B    ; MASK OFF OTHER BITS
0349  74 04                     JZ      CK2_ON          ; IT'S LOW
034B  B3 04                     MOV     BL,4            ; PC5 OF PORT_C WAS HIGH WHEN IT
034D  EB 13                     JMP     SHORT TIMER_ERROR ; SHOULD HAVE BEEN LOW
                        ; TURN GATE BACK ON
034F  E4 61             CK2_ON: IN      AL,PORT_B       ; CURRENT STATUS
0351  0C 01                     OR      AL,00000001B    ; SET BIT 0 - LEAVE OTHERS ALONE
0353  E6 61                     OUT     PORT_B,AL
                        ;--------------------------------------------------------------------------
                        ;       CHECK PC5 OF PORT_C TO SEE IF THE OUTPUT OF TIMER 2 GOES
                        ;       HIGH
                        ;--------------------------------------------------------------------------
0355  B9 000A                   MOV     CX,000AH        ; WAIT FOR OUTPUT GO HIGH, SHOULD
0358  E2 FE             CK2_LO: LOOP    CK2_LO          ; BE LONGER THAN INITIAL COUNT
035A  E4 62                     IN      AL,PORT_C       ; CURRENT STATUS
035C  24 20                     AND     AL,00100000B    ; MASK OFF ALL OTHER BITS
035E  75 57                     JNZ     POD13_END       ; IT'S HIGH - WE'RE DONE!
0360  B3 06                     MOV     BL,6            ; TIMER 2 OUTPUT DID NOT GO HIGH
                        ;--------------------------------------------------------------------------
                        ;       8253 TIMER ERROR OCCURRED.  SET BH WITH MAJOR ERROR
                        ;       INDICATOR AND CALL E_MSG TO INFORM THE SYSTEM OF THE ERROR.
                        ;       (BL ALREADY CONTAINS THE MINOR ERROR INDICATOR TO TELL
                        ;       WHICH PART OF THE TEST FAILED.)
                        ;--------------------------------------------------------------------------
0362                    TIMER_ERROR:
0362  B7 08                     MOV     BH,8            ; TIMER ERROR INDICATOR
0364  E8 09BC R                 CALL    E_MSG
0367  EB 4E                     JMP     SHORT POD13_END
                        ;--------------------------------------------------------------------------
                        ;       BITS ON/OFF SUBROUTINE - USED FOR DETERMINING IF A
                        ;       PARTICULAR TIMER'S BITS GO ON AND OFF AS THEY SHOULD.
                        ;       THIS ROUTINE ASSUMES THAT THE TIMER IS USING BOTH THE LSB
                        ;       AND THE MSB.
                        ;   CALLING PARAMETER:
                        ;       (AH) = TIMER NUMBER (0, 1, OR 2)
                        ;   RETURNS:
                        ;       (CF) = 1 IF FAILED
                        ;       (CF) = 0 IF PASSED
                        ;       REGISTERS AX, BX, CX, DX, DI, AND SI ARE ALTERED.
                        ;--------------------------------------------------------------------------
0369                    LATCHES LABEL   BYTE
0369  00                       DB      00H             ; LATCH MASK FOR TIMER 0
036A  40                       DB      40H             ; LATCH MASK FOR TIMER 1
036B  80                       DB      80H             ; LATCH MASK FOR TIMER 2
                        ;
036C                    BITS_ON_OFF     PROC  NEAR
036C  33 DB                     XOR     BX,BX           ; INITIALIZE BX REGISTER
036E  33 F6                     XOR     SI,SI           ; 1ST PASS - SI = 0
0370  BA 0040                   MOV     DX,TIMER        ; BASE PORT ADDRESS FOR TIMERS
0373  02 D4                     ADD     DL,AH
0375  BF 0369 R                 MOV     DI,OFFSET LATCHES ; SELECT LATCH MASK
0378  32 C0                     XOR     AL,AL           ; CLEAR AL
037A  86 C4                     XCHG    AL,AH           ; AH -> AL
037C  03 F8                     ADD     DI,AX           ; TIMER LATCH MASK INDEX
                        ; 1ST PASS - CHECKS FOR ALL BITS TO COME ON
                        ; 2ND PASS - CHECKS FOR ALL BITS TO GO OFF
037E                    OUTER_LOOP:
037E  B9 0008                   MOV     CX,8            ; OUTER LOOP COUNTER
0381                    INNER_LOOP:
0381  51                        PUSH    CX              ; SAVE OUTER LOOP COUNTER
0382  B9 FFFF                   MOV     CX,0FFFFH       ; INNER LOOP COUNTER
0385                    TST_BITS:
0385  2E: 8A 05                 MOV     AL,CS:[DI]      ; TIMER LATCH MASK
0388  E6 43                     OUT     TIM_CTL,AL      ; LATCH TIMER
038A  50                        PUSH    AX              ; PAUSE
038B  58                        POP     AX
038C  EC                        IN      AL,DX           ; READ TIMER LSB
038D  0B F6                     OR      SI,SI
038F  75 0D                     JNE     SECOND          ; SECOND PASS
0391  0C 01                     OR      AL,01H          ; TURN LS BIT ON
0393  0A D8                     OR      BL,AL           ; TURN 'ON' BITS ON
0395  EC                        IN      AL,DX           ; READ TIMER MSB
0396  0A F8                     OR      BH,AL           ; TURN 'ON' BITS ON
0398  81 FB FFFF                CMP     BX,0FFFFH       ; ARE ALL TIMER BITS ON?
039C  EB 07                     JMP     SHORT TST_CMP   ; DON'T CHANGE FLAGS
```

```
039E                            SECOND:
039E  22 D8                             AND     BL,AL           ; CHECK FOR ALL BITS OFF
03A0  EC                                IN      AL,DX           ; READ MSB
03A1  22 F8                             AND     BH,AL           ; TURN OFF BITS
03A3  0B DB                             OR      BX,BX           ; ALL OFF?
03A5                            TST_CMP:
03A5  74 07                             JE      CHK_END         ; YES - SEE IF DONE
03A7  E2 DC                             LOOP    TST_BITS        ; KEEP TRYING
03A9  59                                POP     CX              ; RESTORE OUTER LOOP COUNTER
03AA  E2 D5                             LOOP    INNER_LOOP      ; TRY AGAIN
03AC  F9                                STC                     ; ALL TRIES EXHAUSTED - FAILED TEST
03AD  C3                                RET
03AE                            CHK_END:
03AE  59                                POP     CX              ; POP FORMER OUTER LOOP COUNTER
03AF  46                                INC     SI
03B0  83 FE 02                          CMP     SI,2
03B3  75 C9                             JNE     OUTER_LOOP      ; CHECK FOR ALL BITS TO GO OFF
03B5  F8                                CLC                     ; TIMER BITS ARE WORKING PROPERLY
03B6  C3                                RET
03B7                            BITS_ON_OFF     ENDP
03B7                            POD13_END:
                                ;--------------------------------------------------------
                                ;                       CRT ATTACHMENT TEST
                                ;
                                ; 1. INIT CRT TO 40X25 - BW
                                ; 2. CHECK FOR VERTICAL AND VIDEO ENABLES, AND CHECK
                                ;    TIMING OF SAME
                                ; 3. CHECK VERTICAL INTERRUPT
                                ; 4. CHECK RED, BLUE, GREEN, AND INTENSIFY DOTS
                                ; 5. INIT TO 40X25 - COLOR
                                ;   MFG. ERROR CODE 09XX (XX-SEE COMMENETS IN CODE)
                                ;--------------------------------------------------------
= A0AC                          MAVT    EQU     0A0ACH          ; MAXIMUM TIME FOR VERT/VERT
                                                                ; (NOMINAL + 10%)
= C460                          MIVT    EQU     0C460H          ; MINIMUM TIME FOR VERT/VERT
                                                                ; (NOMINAL - 10%)
                                ;  NOMINAL TIME IS 8286H FOR 60 hz.
= 00C8                          EPF     EQU     200             ; NUMBER OF ENABLES PER FRAME

03B7  E8 E6D8 R                         CALL    MFG_UP          ; MFG CHECKPOINT= F5
03BA  FA                                CLI
03BB  B0 70                             MOV     AL,01110000B    ; SET TIMER 1 TO MODE 0
03BD  E6 43                             OUT     TIM_CTL,AL
03BF  B9 8000                           MOV     CX,8000H
03C2  E2 FE                     Q1:     LOOP    Q1              ; WAIT FOR MODE SET TO "TAKE"
03C4  B0 00                             MOV     AL,00H
03C6  E6 41                             OUT     TIMER+1,AL      ; SEND FIRST BYTE TO TIMER
03C8  2B C0                             SUB     AX,AX           ; SET MODE 40X25 - BW
03CA  CD 10                             INT     10H
03CC  B8 0507                           MOV     AX,0507H        ; SET TO VIDEO PAGE 7
03CF  CD 10                             INT     10H             ;
03D1  BA 03DA                           MOV     DX,03DAH        ; SET ADDRESSING TO VIDEO ARRAY
03D4  2B C9                             SUB     CX,CX           ;
                                ; LOOK FOR VERTICAL
03D6  EC                        Q2:     IN      AL,DX           ; GET STATUS
03D7  A8 08                             TEST    AL,00001000B    ; VERTICAL THERE YET?
03D9  75 06                             JNE      Q3             ; CONTINUE IF IT IS
03DB  E2 F9                             LOOP    Q2              ; KEEP LOOKING TILL COUNT EXHAUSTED
03DD  B3 00                             MOV     BL,00           ;
03DF  EB 4C                             JMP     SHORT Q115      ; NO VERTICAL = ERROR 0900
                                ; GOT VERTICAL - START TIMER
03E1  32 C0                     Q3:     XOR     AL,AL           ;
03E3  E6 41                             OUT     TIMER+1,AL      ; SEND 2ND BYTE TO TIMER TO START
03E5  2B DB                             SUB     BX,BX           ; INIT. ENABLE COUNTER
                                ; WAIT FOR VERTICAL TO GO AWAY
03E7  33 C9                             XOR     CX,CX
03E9  EC                        Q4:     IN      AL,DX           ; GET STATUS
03EA  A8 08                             TEST    AL,00001000B    ; VERTICAL STILL THERE?
03EC  74 06                             JZ      Q5              ; CONTINUE IF IT'S GONE
03EE  E2 F9                             LOOP    Q4              ; KEEP LOOKING TILL COUNT EXHAUSTED
03F0  B3 01                             MOV     BL,01H          ;
03F2  EB 39                             JMP     SHORT Q115      ; VERTICAL STUCK ON = ERROR 0901
                                ; NOW START LOOKING FOR ENABLE TRANSITIONS
03F4  2B C9                     Q5:     SUB     CX,CX
03F6  EC                        Q6:     IN      AL,DX           ; GET STATUS
03F7  A8 01                             TEST    AL,00000001B    ; ENABLE ON YET?
03F9  75 0A                             JNE     Q7              ; GO ON IF IT IS
03FB  A8 08                             TEST    AL,00001000B    ; VERTICAL ON AGAIN?
03FD  75 22                             JNE     Q11             ; CONTINUE IF IT IS
03FF  E2 F5                             LOOP    Q6              ; KEEP LOOKING IF NOT
0401  B3 02                             MOV     BL,02H          ;
0403  EB 28                             JMP     SHORT Q115      ; ENABLE STUCK OFF = ERROR 0902
                                ; MAKE SURE VERTICAL WENT OFF WITH ENABLE GOING ON
0405  A8 08                     Q7:     TEST    AL,00001000B    ; VERTICAL OFF?
0407  74 04                             JZ      Q8              ; GO ON IF IT IS
0409  B3 03                             MOV     BL,03H          ;
040B  EB 20                             JMP     SHORT Q115      ; VERTICAL STUCK ON = ERROR 0903
                                ; NOW WAIT FOR ENABLE TO GO OFF
040D  2B C9                     Q8:     SUB     CX,CX
040F  EC                        Q9:     IN      AL,DX           ; GET STATUS
0410  A8 01                             TEST    AL,00000001B    ; ENABLE OFF YET?
0412  74 06                             JE      Q10             ; PROCEED IF IT IS
0414  E2 F9                             LOOP    Q9              ; KEEP LOOKING IF NOT YET LOW
0416  B3 04                             MOV     BL,04H          ;
0418  EB 13                             JMP     SHORT Q115      ; ENABLE STUCK ON = ERROR 0904
                                ; ENABLE HAS TOGGLED, BUMP COUNTER AND TEST FOR NEXT VERTICAL
041A  43                        Q10:    INC     BX              ; BUMP ENABLE COUNTER
041B  74 04                             JZ      Q11             ; IF COUNTER WRAPS, ERROR
041D  A8 08                             TEST    AL,00001000B    ; DID ENABLE GO LOW BECAUSE OF
                                                                ; VERTICAL?
041F  74 D3                             JZ      Q5              ; IF NOT, LOOK FOR ANOTHER ENABLE
                                                                ;   TOGGLE
```

```
                        ; HAVE HAD COMPLETE VERTICAL-VERTICAL CYCLE, NOW TEST RESULTS
0421  B0 40             Q11:    MOV     AL,40H          ; LATCH TIMER1
0423  E6 43                     OUT     TIM_CTL,AL      ;
0425  81 FB 00C8                CMP     BX,EPF          ; NUMBER OF ENABLES BETWEEN
                                                        ; VERTICALS O.K.?
0429  74 04                     JE      Q12             ;
042B  B3 05                     MOV     BL,05H          ;
042D  EB 74             Q115:   JMP     SHORT Q22       ; WRONG # ENABLES = ERROR 0905
042F  E4 41             Q12:    IN      AL,TIMER+1      ; GET TIMER VALUE LOW
0431  8A E0                     MOV     AH,AL           ; SAVE IT
0433  90                        NOP                     ;
0434  E4 41                     IN      AL,TIMER+1      ; GET TIMER HIGH
0436  86 E0                     XCHG    AH,AL           ;
0438  FB                        STI                     ; INTERRUPTS BACK ON
0439  90                        NOP
043A  3D A0AC                   CMP     AX,MAVT         ;
043D  7D 04                     JGE     Q13             ;
043F  B3 06                     MOV     BL,06H          ;
0441  EB 60                     JMP     SHORT Q22       ; VERTICALS TOO FAR APART
                                                        ; = ERROR 0906
0443  3D C460           Q13:    CMP     AX,MIVT         ;
0446  7E 04                     JLE     Q14             ;
0448  B3 07                     MOV     BL,07H          ;
044A  EB 57                     JMP     SHORT Q22       ; VERTICALS TOO CLOSE TOGETHER
                                                        ;   = ERROR 0907
                        ; TIMINGS SEEM O.K., NOW CHECK VERTICAL INTERRUPT (LEVEL 5)
044C  2B C9             Q14:    SUB     CX,CX           ; SET TIMEOUT REG
044E  E4 21                     IN      AL,INTA01       ;
0450  24 DF                     AND     AL,11011111B    ; UNMASK INT. LEVEL 5
0452  E6 21                     OUT     INTA01,AL       ;
0454  20 06 0484 R              AND     DATA_AREA[INTR_FLAG-DATA],AL
0458  FB                        STI                     ; ENABLE INTS.
0459  F6 06 0484 R 20   Q15:    TEST    DATA_AREA[INTR_FLAG-DATA],00100000B ; SEE IF INTR.
                                                        ;    5 HAPPENED YET
045E  75 06                     JNZ     Q16             ; GO ON IF IT DID
0460  E2 F7                     LOOP    Q15             ; KEEP LOOKING IF IT DIDN'T
0462  B3 08                     MOV     BL,08H          ;
0464  EB 3D                     JMP     SHORT Q22       ; NO VERTICAL INTERRUPT
                                                        ; = ERROR 0908
0466  E4 21             Q16:    IN      AL,INTA01       ; DISABLE INTERRUPTS FOR LEVEL 5
0468  0C 20                     OR      AL,00100000B    ;
046A  E6 21                     OUT     INTA01,AL       ;
                        ; SEE IF RED, GREEN, BLUE AND INTENSIFY DOTS WORK
                        ; FIRST, SET A LINE OF REVERSE VIDEO, INTENSIFIED BLANKS INTO VIDEO
                        ; BUFFER
046C  B8 09DB                   MOV     AX,09DBH        ; WRITE CHARS, BLOCKS
046F  BB 077F                   MOV     BX,077FH        ; PAGE 7, REVERSE VIDEO,
                                                        ;    HIGH INTENSITY
0472  B9 0028                   MOV     CX,40           ; 40 CHARACTERS
0475  CD 10                     INT     10H             ;
0477  33 C0                     XOR     AX,AX           ; START WITH BLUE DOTS
0479  2B C9             Q17:    SUB     CX,CX           ;
047B  EE                        OUT     DX,AL           ; SET VIDEO ARRAY ADDRESS FOR DOTS
                        ; SEE IF DOT COMES ON
047C  EC                Q18:    IN      AL,DX           ; GET STATUS
047D  A8 10                     TEST    AL,00010000B    ; DOT THERE?
047F  75 08                     JNZ     Q19             ; GO LOOK FOR DOT TO TURN OFF
0481  E2 F9                     LOOP    Q18             ; CONTINUE TESTING FOR DOT ON
0483  B3 10                     MOV     BL,10H          ;
0485  0A DC                     OR      BL,AH           ; OR IN DOT BEING TESTED
0487  EB 1A                     JMP     SHORT Q22       ; DOT NOT COMING ON = ERROR 091X
                                                        ; ( X=0, BLUE; X=1, GREEN;
                                                        ; X=2, RED; X=3, INTENSITY)
                        ; SEE IF DOT GOES OFF
0489  2B C9             Q19:    SUB     CX,CX
048B  EC                Q20:    IN      AL,DX           ; GET STATUS
048C  A8 10                     TEST    AL,00010000B    ; IS DOT STILL ON?
048E  74 08                     JE      Q21             ; GO ON IF DOT OFF
0490  E2 F9                     LOOP    Q20             ; ELSE, KEEP WAITING FOR DOT
                                                        ; TO GO OFF
0492  B3 20                     MOV     BL,20H          ;
0494  0A DC                     OR      BL,AH           ; OR IN DOT BEING TESTED
0496  EB 0B                     JMP     SHORT Q22       ; DOT STUCK ON = ERROR 092X
                                                        ; (X=0, BLUE; X=1, GREEN;
                                                        ;  X=2, RED; X=3, INTENSITY)
                        ; ADJUST TO POINT TO NEXT DOT
0498  FE C4             Q21:    INC     AH              ;
049A  80 FC 04                  CMP     AH,4            ; ALL 4 DOTS DONE?
049D  74 09                     JE      Q23             ; GO END
049F  8A C4                     MOV     AL,AH           ;
04A1  EB D6                     JMP     Q17             ; GO LOOK FOR ANOTHER DOT
04A3  B7 09             Q22:    MOV     BH,09H          ; SET MSB OF ERROR CODE
04A5  E9 09BC R                 JMP     E_MSG
                        ; DONE WITH TEST RESET TO 40X25 - COLOR
                                ASSUME  DS:DATA
04A8  E8 138B R         Q23:    CALL    DDS
04AB  B8 0001                   MOV     AX,0001H        ; INIT TO 40X25 - COLOR
04AE  CD 10                     INT     10H
04B0  B8 0507                   MOV     AX,0507H        ; SET TO VIDEO PAGE 7
04B3  CD 10                     INT     10H
04B5  81 3E 0072 R 1234         CMP     RESET_FLAG,1234H ; WARM START?
04BB  74 03                     JE      Q24             ; BYPASS PUTTING UP POWER-ON SCREEN
04BD  E8 0C21 R                 CALL    PUT_LOGO        ; PUT LOGO ON SCREEN
```

```
04BD  E8 0C21 R                       CALL    PUT_LOGO          ; PUT LOGO ON SCREEN
04C0  B0 76                   Q24:    MOV     AL,01110110B      ; RE-INIT TIMER 1
04C2  E6 43                           OUT     TIM_CTL,AL        ;
04C4  B0 00                           MOV     AL,00H
04C6  E6 41                           OUT     TIMER+1,AL
04C8  90                              NOP
04C9  90                              NOP
04CA  E6 41                           OUT     TIMER+1,AL
                                      ASSUME  DS:ABS0
04CC  E8 E6D8 R                       CALL    MFG_UP            ; MFG CHECKPOINT=F4
04CF  33 C0                           XOR     AX,AX
04D1  8E D8                           MOV     DS,AX
04D3  C7 06 0008 R 0F78 R             MOV     NMI_PTR,OFFSET KBDNMI ; SET INTERRUPT VECTOR
04D9  C7 06 0120 R F06B R             MOV     KEY62_PTR,OFFSET KEY_SCAN_SAVE  ; SET VECTOR FOR
                                                                    ; POD INT HANDLER
04DF  0E                              PUSH    CS
04E0  58                              POP     AX
04E1  A3 0122 R                       MOV     KEY62_PTR+2,AX
                                      ASSUME  DS:DATA
04E4  E8 138B R                       CALL    DDS               ; SET DATA SEGMENT
04E7  BE 001E R                       MOV     SI,OFFSET KB_BUFFER ; SET KEYBOARD PARMS
04EA  89 36 001A R                    MOV     BUFFER_HEAD,SI
04EE  89 36 001C R                    MOV     BUFFER_TAIL,SI
04F2  89 36 0080 R                    MOV     BUFFER_START,SI
04F6  83 C6 20                        ADD     SI,32             ; SET DEFAULT BUFFER OF 32 BYTES
04F9  89 36 0082 R                    MOV     BUFFER_END,SI
04FD  E4 A0                           IN      AL,0A0H           ; CLEAR NMI F/F
04FF  B0 80                           MOV     AL,80H            ; ENABLE NMI
0501  E6 A0                           OUT     0A0H,AL           ;
                              ;  IF A KEY IS STUCK, THE BUFFER SHOULD FILL WITH THAT KEY'S CODE
                              ;  THIS WILL BE CHECKED LATER
                              ;--------------------------------------------
                              ;
                              ;          MEMORY SIZE DETERMINE AND TEST
                              ;  THIS ROUTINE WILL DETERMINE HOW MUCH MEM
                              ;  IS ATTACHED TO THE SYSTEM (UP TO 640KB)
                              ;  AND SET "MEMORY_SIZE" AND "REAL_MEMORY"
                              ;  WORDS IN THE DATA AREA.
                              ;
                              ;  AFTER THIS, MEMORY WILL BE EITHER TESTED
                              ;  OR CLEARED, DEPENDING ON THE CONTENTS OF
                              ;  "RESET_FLAG".
                              ;   MFG. ERROR CODES   -0AXX PLANAR BD ERROR
                              ;                      -0BXX 64K CD ERROR
                              ;                      -0CXX ERRORS IN BOTH
                              ;                       ODD AND EVEN BYTES
                              ;                       IN A 128K SYS
                              ;                      -1YXX MEMORY ABOVE 128K
                              ;                       Y=SEGMENT HAVING TROUBLE
                              ;                       XX= ERROR BITS
                              ;--------------------------------------------
                                      ASSUME  DS:DATA
0503  E8 E6D8 R                       CALL    MFG_UP            ; MFG CHECKPOINT=F3
0506  BB 0040                         MOV     BX,64             ; START WITH BASE 64K
0509  E4 62                           IN      AL,PORT_C         ; GET CONFIG BYTE
050B  A8 08                           TEST    AL,00001000B      ; SEE IF 64K CARD INSTALLED
050D  75 03                           JNE     Q25               ; (BIT 4 WILL BE 0 IF CARD PLUGGED)
050F  83 C3 40                        ADD     BX,64             ; ADD 64K
0512  53                      Q25:    PUSH    BX                ; SAVE K COUNT
0513  83 EB 10                        SUB     BX,16             ; SUBTRACT 16K CRT REFRESH SPACE
0516  89 1E 0013 R                    MOV     [MEMORY_SIZE],BX  ; LOAD "CONTIGUOUS MEMORY" WORD
051A  5B                              POP     BX
051B  BA 2000                         MOV     DX,2000H          ; SET POINTER TO JUST ABOVE 128K
051E  2B FF                           SUB     DI,DI             ; SET DI TO POINT TO BEGINNING
0520  B9 AA55                         MOV     CX,0AA55H         ; LOAD DATA PATTERN
0523  8E C2                   Q26:    MOV     ES,DX             ; SET SEGMENT TO POINT TO MEMORY
                                                                ; SPACE
0525  26: 89 0D                       MOV     ES:[DI],CX        ; SET DATA PATTERN TO MEMORY
0528  B0 0F                           MOV     AL,0FH            ; SET AL TO ODD VALUE
052A  26: 8B 05                       MOV     AX,ES:[DI]        ; GET DATA PATTERN BACK FROM MEM
052D  33 C1                           XOR     AX,CX             ; SEE IF DATA MADE IT BACK
052F  75 0C                           JNZ     Q27               ; NO? THEN END OF MEM HAS BEEN
                                                                ; REACHED
0531  81 C2 1000                      ADD     DX,1000H          ; POINT TO BEGINNING OF NEXT 64K
0535  83 C3 40                        ADD     BX,64             ; ADJUST TOTAL MEM. COUNTER
0538  80 FE A0                        CMP     DH,0A0H           ; PAST 640K YET?
053B  75 E6                           JNE     Q26               ; CHECK FOR ANOTHER BLOCK IF NOT
053D  89 1E 0015 R            Q27:    MOV     [TRUE_MEM],BX     ; LOAD "TOTAL MEMORY" WORD
                              ; SIZE HAS BEEN DETERMINED, NOW TEST OR CLEAR ALL OF MEMORY
0541  B8 0004                         MOV     AX,4              ; 4 KB KNOWN OK AT THIS POINT
0544  E8 05BC R                       CALL    Q35
0547  BA 0080                         MOV     DX,0080H          ; SET POINTER TO JUST ABOVE
                                                                ; LOWER 2K
054A  B9 7800                         MOV     CX,7800H          ; TEST 30K WORDS (60KB)
054D  8E C2                   Q28:    MOV     ES,DX
054F  51                              PUSH    CX
0550  53                              PUSH    BX
0551  50                              PUSH    AX
0552  E8 0B59 R                       CALL    PODSTG            ; TEST OR FILL MEM
0555  74 03                           JZ      Q29
0557  E9 0603 R                       JMP     Q39               ; JUMP IF ERROR
055A  58                      Q29:    POP     AX
055B  5B                              POP     BX
055C  59                              POP     CX                ; RECOVER
055D  80 FD 78                        CMP     CH,78H            ; WAS THIS A 60 K PASS
0560  9C                              PUSHF
0561  05 003C                         ADD     AX,60             ; BUMP GOOD STORAGE BY 60 KB
0564  9D                              POPF
0565  74 03                           JE      Q30
0567  05 0002                         ADD     AX,2              ;ADD 2 FOR A 62K PASS
056A  E8 05BC R               Q30:    CALL    Q35
056D  3B C3                           CMP     AX,BX             ; ARE WE DONE YET?
056F  75 03                           JNE     Q31
0571  E9 0640 R                       JMP     Q43               ; ALL DONE, IF SO
```

```
0574  3D 0080                Q31:    CMP     AX,128          ; DONE WITH 1ST 128K?
0577  74 1E                          JE      Q32             ; GO FINISH REST OF MEM.
0579  BA 0F80                        MOV     DX,0F80H        ; SET POINTER TO FINISH 1ST 64 KB
057C  B9 0400                        MOV     CX,0400H
057F  8E C2                          MOV     ES,DX
0581  50                             PUSH    AX
0582  53                             PUSH    BX
0583  52                             PUSH    DX
0584  E8 0B59 R                      CALL    PODSTG          ; GO TEST/FILL
0587  75 7A                          JNZ     Q39             ;
0589  5A                             POP     DX
058A  5B                             POP     BX
058B  58                             POP     AX
058C  05 0002                        ADD     AX,2            ; UPDATE GOOD COUNT
058F  BA 1000                        MOV     DX,1000H        ; SET POINTER TO 2ND 64K BLOCK
0592  B9 7C00                        MOV     CX,7C00H        ; 62K WORTH
0595  EB 86                          JMP     Q28             ; GO TEST IT
0597  BA 2000                Q32:    MOV     DX,2000H        ; POINT TO BLOCK ABOVE 128K
059A  3B D8                   Q33:    CMP     BX,AX           ; COMPARE GOOD MEM TO TOTAL MEM
059C  75 03                           JNE     Q34
059E  E9 0640 R                       JMP     Q43             ; EXIT IF ALL DONE
05A1  B9 4000                 Q34:    MOV     CX,4000H        ; SET FOR 32KB BLOCK
05A4  8E C2                           MOV     ES,DX
05A6  50                              PUSH    AX
05A7  53                              PUSH    BX
05A8  52                              PUSH    DX
05A9  E8 0B59 R                       CALL    PODSTG          ; GO TEST/FILL
05AC  75 55                           JNZ     Q39             ;
05AE  5A                              POP     DX
05AF  5B                              POP     BX
05B0  58                              POP     AX
05B1  05 0020                         ADD     AX,32           ; BUMP GOOD MEMORY COUNT
05B4  E8 05BC R                       CALL    Q35             ; DISPLAY CURRENT GOOD MEM
05B7  80 C6 08                        ADD     DH,08H          ; SET POINTER TO NEXT 32K
05BA  EB DE                           JMP     Q33             ; AND MAKE ANOTHER PASS
                              ;---------------------------------------
                              ;    SUBROUTINE FOR PRINTING TESTED
                              ;    MEMORY OK MSG ON THE CRT
                              ; CALL PARMS: AX = K OF GOOD MEMORY
                              ;              (IN HEX)
                              ;---------------------------------------
05BC                          Q35     PROC    NEAR
05BC  E8 138B R                       CALL    DDS             ; ESTABLISH ADDRESSING
05BF  81 3E 0072 R 1234               CMP     RESET_FLAG,1234H ; WARM START?
05C5  74 3B                           JE      Q35E            ; NO PRINT ON WARM START
05C7  53                              PUSH    BX
05C8  51                              PUSH    CX
05C9  52                              PUSH    DX
05CA  50                              PUSH    AX              ; SAVE WORK REGS
05CB  B4 02                           MOV     AH,2            ; SET CURSOR TOWARD THE END OF
05CD  BA 1421                         MOV     DX,1421H        ; ROW 20 (ROW 20, COL. 33)
05D0  B7 07                           MOV     BH,7            ; PAGE 7
05D2  CD 10                           INT     10H
05D4  58                              POP     AX              ;
05D5  50                              PUSH    AX
05D6  BB 000A                         MOV     BX,10           ; SET UP FOR DECIMAL CONVERT
05D9  B9 0003                         MOV     CX,3            ; OF 3 NIBBLES
05DC  33 D2                   Q36:    XOR     DX,DX           ;
05DE  F7 F3                           DIV     BX              ; DEVIDE BY 10
05E0  80 CA 30                        OR      DL,30H          ; MAKE INTO ASCII
05E3  52                              PUSH    DX              ; SAVE
05E4  E2 F6                           LOOP    Q36             ;
05E6  B9 0003                         MOV     CX,3            ;
05E9  58                      Q37:    POP     AX              ; RECOVER A NUMBER
05EA  E8 18BA R                       CALL    PRT_HEX
05ED  E2 FA                           LOOP    Q37
05EF  B9 0003                         MOV     CX,3
05F2  BE 0025 R                       MOV     SI,OFFSET F3B   ; PRINT " KB"
05F5  2E: 8A 04               Q38:    MOV     AL,CS:[SI]
05F8  46                              INC     SI
05F9  E8 18BA R                       CALL    PRT_HEX
05FC  E2 F7                           LOOP    Q38
05FE  58                              POP     AX
05FF  5A                              POP     DX
0600  59                              POP     CX
0601  5B                              POP     BX
0602  C3                    Q35E:   RET
0603                          Q35     ENDP
                              ; ON ENTRY TO MEMORY ERROR ROUTINE, CX HAS ERROR BITS
                              ; AH HAS ODD/EVEN INFO, OTHER USEFUL INFO ON THE STACK
0603  5A                      Q39:    POP     DX              ; POP SEGMENT POINTER TO DX
                                                              ; (HEADING DOWNHILL, DON'T CARE
                                                              ; ABOUT STACK)
0604  81 FA 2000                       CMP     DX,2000H        ; ABOVE 128K (THE SIMPLE CASE)
0608  7C 0E                           JL      Q40             ; GO DO ODD/EVEN-LESS THAN 128K
060A  8A D9                           MOV     BL,CL           ; FORM ERROR BITS ("XX")
060C  0A DD                           OR      BL,CH
060E  B1 04                           MOV     CL,4            ; ROTATE MOST SIGNIFIGANT
                                                              ; NIBBLE OF SEGMET
0610  D2 EE                           SHR     DH,CL           ; TO LOW NIBBLE OF DH
0612  B7 10                           MOV     BH,10H          ;
0614  0A FE                           OR      BH,DH           ; FORM "1Y" VALUE
0616  EB 20                           JMP     SHORT Q42
0618  B7 0A                   Q40:    MOV     BH,0AH          ; ERROR 0A....
061A  E4 62                           IN      AL,PORT_C       ; GET CONFIG BITS
061C  24 08                           AND     AL,00001000B    ; TEST FOR ATTRIB CARD PRESENT
061E  74 06                           JZ      Q41             ; WORRY ABOUT ODD/EVEN IF IT IS
0620  8A D9                           MOV     BL,CL
0622  0A DD                           OR      BL,CH           ; COMBINE ERROR BITS IF IT ISN'T
0624  EB 12                           JMP     SHORT Q42       ;
```

```
0626  80 FC 02                  Q41:    CMP     AH,02           ; EVEN BYTE ERROR? ERR 0AXX
0629  8A D9                            MOV     BL,CL
062B  74 0B                      JE      Q42
062D  FE C7                      INC     BH              ; MAKE INTO 0BXX ERR
062F  0A DD                      OR      BL,CH           ; MOVE AND COMBINE ERROR BITS
0631  80 FC 01                   CMP     AH,1            ; ODD BYTE ERROR
0634  74 02                      JE      Q42
0636  FE C7                      INC     BH              ; MUST HAVE BEEN BOTH
                                                         ; - MAKE INTO 0CXX
0638  BE 0035 R                  Q42:    MOV     SI,OFFSET MEM_ERR
063B  E8 09BC R                  CALL    E_MSG           ; LET ERROR ROUTINE FIGURE OUT
                                                         ; WHAT TO DO
063E  FA                         CLI                     ;
063F  F4                         HLT
0640                             Q43:
                                 ;--------------------------------------------
                                 ;       KEYBOARD TEST
                                 ; DESCRIPTION
                                 ;       NMI HAS BEEN ENABLED FOR QUITE A FEW
                                 ;       SECONDS NOW. CHECK THAT NO SCAN CODES
                                 ;       HAVE SHOWN UP IN THE BUFFER. (STUCK
                                 ;       KEY) IF THEY HAVE, DISPLAY THEM AND
                                 ;       POST ERROR.
                                 ;       MFG ERR CODE
                                 ;       2000 STRAY NMI INTERRUPTS OR KEYBOARD
                                 ;       RECEIVE ERRORS
                                 ;       21XX  CARD FAILURE
                                 ;           XX=01, KB DATA STUCK HIGH
                                 ;           XX=02, KB DATA STUCK LOW
                                 ;           XX=03, NO NMI INTERRUPT
                                 ;       22XX STUCK KEY (XX=SCAN CODE)
                                 ;--------------------------------------------
                                         ASSUME  DS:DATA
                                 ;----- CHECK FOR STUCK KEYS
0640  E8 E6DB R                          CALL    MFG_UP          ; MFG CODE=F2
0643  E8 138B R                          CALL    DDS             ; ESTABLISH ADDRESSING
0646  BB 001E R                          MOV     BX,OFFSET KB_BUFFER
0649  8A 07                             MOV     AL,[BX]         ; CHECK FOR STUCK KEYS
064B  0A C0                             OR      AL,AL           ; SCAN CODE = 0?
064D  74 06                             JE      F6_Y            ; YES - CONTINUE TESTING
064F  B7 22                             MOV     BH,22H          ; 22XX ERROR CODE
0651  8A D8                             MOV     BL,AL           ;
0653  EB 0A                             JMP     SHORT F6
0655  80 3E 0012 R 00           F6_Y:   CMP     KBD_ERR,00H     ; DID NMI'S HAPPEN WITH NO SCAN
                                                                ; CODE PASSED?
065A  74 1C                             JE      F7              ; (STRAYS) - CONTINUE IF NONE
065C  BB 2000                           MOV     BX,2000H        ; SET ERROR CODE 2000
065F  BE 0036 R                 F6:     MOV     SI,OFFSET KEY_ERR ; GET MSG ADDR
0662  81 3E 0072 R 4321                 CMP     RESET_FLAG,4321H ; WARM START TO DIAGS
0668  74 0B                             JE      F6_Z            ; DO NOT PUT UP MESSAGE
066A  81 3E 0072 R 1234                 CMP     RESET_FLAG,1234H ; WARM SYSTEM START
0670  74 03                             JE      F6_Z            ; DO NOT PUT UP MESSAGE
0672  E8 09BC R                         CALL    E_MSG           ; PRINT MSG ON SCREEN
0675  E9 06FF R                 F6_Z:   JMP     F6_X
                                 ; CHECK LINK CARD, IF PRESENT
0678  BA 0201                   F7:     MOV     DX,0201H
067B  EC                                IN      AL,DX           ; CHECK FOR BURN-IN MODE
067C  24 F0                             AND     AL,0F0H
067E  74 7F                             JZ      F6_X            ; BYPASS CHECK IN BURN-IN MODE
0680  E4 62                             IN      AL,PORT_C       ; GET CONFIG. PORT DATA
0682  24 80                             AND     AL,10000000B    ; KEYBOARD CABLE ATTACHED?
0684  74 79                             JZ      F6_X            ; BYPASS  TEST IF IT IS
0686  E4 61                             IN      AL,PORT_B       ;
0688  24 FC                             AND     AL,11111100B    ; DROP SPEAKER DATA
068A  E6 61                             OUT     PORT_B,AL       ;
068C  B0 B6                             MOV     AL,0B6H         ; MODE SET TIMER 2
068E  E6 43                             OUT     TIM_CTL,AL      ;
0690  B0 40                             MOV     AL,040H         ; DISABLE NMI
0692  E6 A0                             OUT     0A0H,AL         ;
0694  B0 20                             MOV     AL,32           ; LSB TO TIMER 2
                                                                ; (APPROX. 40Khz VALUE)
0696  BA 0042                           MOV     DX,TIMER+2
0699  EE                               OUT     DX,AL
069A  2B C0                             SUB     AX,AX
069C  8B C8                             MOV     CX,AX
069E  EE                                OUT     DX,AL           ; MSB TO TIMER 2 (START TIMER)
069F  E4 61                             IN      AL,PORT_B       ;
06A1  0C 01                             OR      AL,1
06A3  E6 61                             OUT     PORT_B,AL       ; ENABLE TIMER 2
06A5  E4 62                     F7_0:   IN      AL,PORT_C       ; SEE IF KEYBOARD DATA ACTIVE
06A7  24 40                             AND     AL,01000000B    ;
06A9  75 06                             JNZ     F7_1            ; EXIT LOOP IF DATA SHOWED UP
06AB  E2 F8                             LOOP    F7_0
06AD  B3 02                             MOV     BL,02H          ; SET NO KEYBOARD DATA ERROR
06AF  EB 49                             JMP     SHORT F6_1
06B1  06                        F7_1:   PUSH    ES              ; SAVE ES
06B2  2B C0                             SUB     AX,AX           ; SET UP SEGMENT REG
06B4  8E C0                             MOV     ES,AX           ; *
06B6  26: C7 06 0008 R F815 R           MOV     ES:[NMI_PTR],OFFSET D11 ; SET UP NEW NMI VECTOR
06BD  A2 0084 R                         MOV     INTR_FLAG,AL    ; RESET INTR FLAG
06C0  E4 61                             IN      AL,PORT_B       ; DISABLE INTERNAL BEEPER TO
06C2  0C 30                             OR      AL,00110000B    ; PREVENT ERROR BEEP
06C4  E6 61                             OUT     PORT_B,AL
06C6  B0 C0                             MOV     AL,0C0H
06C8  E6 A0                             OUT     0A0H,AL         ; ENABLE NMI
06CA  B9 0100                           MOV     CX,0100H        ;
```

```
06CD  E2 FE                   F6_0:   LOOP    F6_0            ; WAIT A BIT
06CF  E4 61                           IN      AL,PORT_B       ; RE-ENABLE BEEPER
06D1  24 CF                           AND     AL,11001111B
06D3  E6 61                           OUT     PORT_B,AL
06D5  A0 0084 R                       MOV     AL,INTR_FLAG    ; GET INTR FLAG
06D8  0A C0                           OR      AL,AL           ; WILL BE NON-ZERO IF NMI HAPPENED
06DA  B3 03                           MOV     BL,03H          ; SET POSSIBLE ERROR CODE
06DC  26: C7 06 0008 R 0F78 R         MOV     ES:[NMI_PTR],OFFSET KBDNMI ; RESET NMI VECTOR
06E3  07                              POP     ES              ; RESTORE ES
06E4  74 14                           JZ      F6_1            ; JUMP IF NO NMI
06E6  B0 00                           MOV     AL,00H          ; DISABLE FEEDBACK CKT
06E8  E6 A0                           OUT     0A0H,AL         ;
06EA  E4 61                           IN      AL,PORT_B       ;
06EC  24 FE                           AND     AL,11111110B    ; DROP GATE TO TIMER 2
06EE  E6 61                           OUT     PORT_B,AL       ;
06F0  E4 62                   F6_2:   IN      AL,PORT_C       ; SEE IF KEYBOARD DATA ACTIVE
06F2  24 40                           AND     AL,01000000B    ;
06F4  74 09                           JZ      F6_X            ; EXIT LOOP IF DATA WENT LOW
06F6  E2 F8                           LOOP    F6_2
06F8  B3 01                           MOV     BL,01H          ; SET KEYBOARD DATA STUCK HIGH ERR
06FA  B7 21                   F6_1:   MOV     BH,21H          ; POST ERROR  "21XX"
06FC  E9 065F R                       JMP     F6              ;
06FF  B0 00                   F6_X:   MOV     AL,00H          ; DISABLE FEEDBACK CKT
0701  E6 A0                           OUT     0A0H,AL         ;
                              ;--------------------------------------------------------
                              ;       CASSETTE INTERFACE TEST
                              ; DESCRIPTION
                              ;       TURN CASSETTE MOTOR OFF. WRITE A BIT OUT TO THE
                              ;       CASSETTE DATA BUS. VERIFY THAT CASSETTE DATA
                              ;       READ IS WITHIN A VALID RANGE.
                              ;  MFG. ERROR CODE=2300H (DATA PATH ERROR)
                              ;                  23FF (RELAY FAILED TO PICK)
                              ;--------------------------------------------------------
= 0A9A                        MAX_PERIOD      EQU     0A9AH   ;NOM.+10%
= 08AD                        MIN_PERIOD      EQU     08ADH   ;NOM -10%
                              ;----- TURN THE CASSETTE MOTOR OFF
0703  E8 E608 R                       CALL    MFG_UP          ; MFG CODE=F1
0706  E4 61                           IN      AL,PORT_B
0708  0C 09                           OR      AL,00001001B    ; SET TIMER 2 SPK OUT, AND CASSETTE
070A  E6 61                           OUT     PORT_B,AL       ; OUT BITS ON, CASSETTE MOT OFF
                              ;----- WRITE A BIT
070C  E4 21                           IN      AL,INTA01
070E  0C 01                           OR      AL,01H          ; DISABLE TIMER INTERRUPTS
0710  E6 21                           OUT     INTA01,AL
0712  B0 B6                           MOV     AL,0B6H         ; SEL TIM 2, LSB, MSB, MD 3
0714  E6 43                           OUT     TIMER+3,AL      ; WRITE 8253 CMD/MODE REG
0716  B8 04D2                       MOV     AX,1234         ; SET TIMER 2 CNT FOR 1000 USEC
0719  E6 42                           OUT     TIMER+2,AL      ; WRITE TIMER 2 COUNTER REG
071B  8A C4                           MOV     AL,AH           ; WRITE MSB
071D  E6 42                           OUT     TIMER+2,AL
071F  2B C9                           SUB     CX,CX           ; CLEAR COUNTER FOR LONG DELAY
0721  E2 FE                           LOOP    $               ; WAIT FOR COUNTER TO INIT
                              ;----- READ CASSETTE INPUT
0723  E4 62                           IN      AL,PORT_C       ; READ VALUE OF CASS IN BIT
0725  24 10                           AND     AL,10H          ; ISOLATE FROM OTHER BITS
0727  A2 006B R                       MOV     LAST_VAL,AL
072A  E8 F96F R                       CALL    READ_HALF_BIT   ; TO SET UP CONDITIONS FOR CHECK
072D  E8 F96F R                       CALL    READ_HALF_BIT
0730  E3 3E                           JCXZ    F8              ; CAS_ERR
0732  53                             PUSH    BX              ; SAVE HALF BIT TIME VALUE
0733  E8 F96F R                       CALL    READ_HALF_BIT
0736  58                             POP     AX              ; GET TOTAL TIME
0737  E3 37                           JCXZ    F8              ; CAS_ERR
0739  03 C3                           ADD     AX,BX
073B  3D 0A9A                       CMP     AX,MAX_PERIOD
073E  73 30                           JNC     F8              ; CAS_ERR
0740  3D 08AD                       CMP     AX,MIN_PERIOD
0743  72 28                           JC      F8
0745  BA 0201                       MOV     DX,201H
0748  EC                             IN      AL,DX
0749  24 F0                           AND     AL,0F0H         ; DETERMINE MODE
074B  3C 10                           CMP     AL,00010000B    ; MFG?
074D  74 04                           JE      F9
074F  3C 40                           CMP     AL,01000000B    ; SERVICE?
0751  75 26                           JNE     T13_END         ; GO TO NEXT TEST IF NOT
                              ; CHECK THAT CASSETTE RELAY IS PICKING (CAN'T DO TEST IN NORMAL
                              ; MODE BECAUSE OF POSSIBILITY OF WRITING ON CASSETTE IF "RECORD"
                              ; BUTTON IS DEPRESSED.)
0753  E4 61                   F9:     IN      AL,PORT_B
0755  8A D0                           MOV     DL,AL           ; SAVE PORT B CONTENTS
0757  24 E5                           AND     AL,11100101B    ; SET CASSETTE MOTOR ON
0759  E6 61                           OUT     PORT_B,AL       ;
075B  33 C9                           XOR     CX,CX
075D  E2 FE                   F91:    LOOP    F91             ; WAIT FOR RELAY TO SETTLE
075F  E8 F96F R                       CALL    READ_HALF_BIT
0762  E8 F96F R                       CALL    READ_HALF_BIT
0765  8A C2                           MOV     AL,DL           ; DROP RELAY
0767  E6 61                           OUT     PORT_B,AL
0769  E3 0E                           JCXZ    T13_END         ; READ_HALF_BIT SHOULD TIME OUT IN
                                                              ; THIS SITUATION
076B  BB 23FF                         MOV     BX,23FFH        ; ERROR 23FF
076E  EB 03                           JMP     SHORT F81
0770                          F8:                             ; CAS_ERR
0770  BB 2300                         MOV     BX,2300H        ; ERR. CODE 2300H
0773  BE 0037 R               F81:    MOV     SI,OFFSET CASS_ERR ; CASSETTE WRAP FAILED
0776  E8 09BC R                       CALL    E_MSG           ; GO PRINT ERROR MSG
0779  E4 21                   T13_END:IN      AL,INTA01
077B  24 FE                           AND     AL,0FEH         ; ENABLE TIMER INTS
077D  E6 21                           OUT     INTA01,AL
077F  E4 A0                           IN      AL,NMI_PORT     ; CLEAR NMI FLIP/FLOP
0781  B0 80                           MOV     AL,80H          ; ENABLE NMI INTERUPTS
0783  E6 A0                           OUT     NMI_PORT,AL
```

```
                        ;--------------------------------------------------------------------------
                        ;       SERIAL PRINTER AND MODEM POWER ON DIAGNOSTIC
                        ;   DESCRIPTION:
                        ;       VERIFIES THAT THE SERIAL PRINTER UART FUNCTIONS PROPERLY.
                        ;       CHECKS IF THE MODEM CARD IS ATTACHED.  IF IT'S NOT, EXITS.
                        ;       VERIFIES THAT THE MODEM UART FUNCTIONS PROPERLY.
                        ;       ERROR CODES RETURNED BY 'UART' RANGE FROM 1 TO 1FH AND ARE
                        ;       REPORTED VIA REGISTER BL.  SEE LISTING OF 'UART' (POD27)
                        ;       FOR POSSIBLE ERRORS.
                        ;       MFG. ERR. CODES  23XX FOR SERIAL PRINTER
                        ;                        24XX FOR MODEM
                        ;--------------------------------------------------------------------------
                                ASSUME  CS:CODE,DS:DATA
                        ;--------------------------------------------------------------------------
                        ;       TEST SERIAL PRINTER INS8250 UART
                        ;--------------------------------------------------------------------------
0785  E8 E6D8 R                  CALL    MFG_UP          ; MFG ROUTINE INDICATOR=F0
0788  BA 02F8                   MOV     DX,02F8H        ; ADDRESS OF SERIAL PRINTER CARD
078B  E8 E831 R                  CALL    UART            ; ASYNCH. COMM. ADAPTER POD
078E  73 06                     JNC     TM              ; PASSED
0790  BE 0038 R                  MOV     SI,OFFSET COM1_ERR ; CODE FOR DISPLAY
0793  E8 098C R                  CALL    E_MSG           ; REPORT ERROR
                        ;--------------------------------------------------------------------------
                        ;       TEST MODEM INS8250 UART
                        ;--------------------------------------------------------------------------
0796  E8 E6D8 R         TM:      CALL    MFG_UP          ; MFG ROUTINE INDICATOR = EF
0799  E4 62                     IN      AL,PORT_C       ; TEST FOR MODEM CARD PRESENT
079B  24 02                     AND     AL,00000010B    ; ONLY CONCERNED WITH BIT 1
079D  75 0E                     JNE     TM1             ; IT'S NOT THERE - DONE WITH TEST
079F  BA 03F8                   MOV     DX,03F8H        ; ADDRESS OF MODEM CARD
07A2  E8 E831 R                  CALL    UART            ; ASYNCH. COMM. ADAPTER POD
07A5  73 06                     JNC     TM1             ; PASSED
07A7  BE 0039 R                  MOV     SI,OFFSET COM2_ERR ; MODEM ERROR
07AA  E8 098C R                  CALL    E_MSG           ; REPORT ERROR
07AD                    TM1:
                        ;-----------------------------------------
                        ;       SETUP HARDWARE INT. VECTOR TABLE
                        ;-----------------------------------------
                                ASSUME  CS:CODE,DS:ABS0
07AD  2B C0                     SUB     AX,AX
07AF  8E C0                     MOV     ES,AX
07B1  B9 0008                   MOV     CX,08           ; GET VECTOR CNT
07B4  0E                        PUSH    CS              ; SETUP DS SEG REG
07B5  1F                        POP     DS
07B6  BE FEF3 R                  MOV     SI,OFFSET VECTOR_TABLE
07B9  BF 0020 R                  MOV     DI,OFFSET INT_PTR
07BC  A5                F7A:     MOVSW
07BD  47                        INC     DI              ; SKIP OVER SEGMENT
07BE  47                        INC     DI
07BF  E2 FB                     LOOP    F7A
                        ;----- SET UP OTHER INTERRUPTS AS NECESSARY
                                ASSUME  DS:ABS0
07C1  8E D9                     MOV     DS,CX
07C3  C7 06 0014 R FF54 R       MOV     INT5_PTR,OFFSET PRINT_SCREEN ; PRINT SCREEN
07C9  C7 06 0120 R 10C6 R       MOV     KEY62_PTR,OFFSET KEY62_INT ; 62 KEY CONVERSION
                                                                   ; ROUTINE
07CF  C7 06 0110 R FA6E R       MOV     CSET_PTR,OFFSET CRT_CHAR_GEN ; DOT TABLE
07D5  C7 06 0060 R FFCB R       MOV     BASIC_PTR,OFFSET BAS_ENT ; CASSETTE BASIC ENTRY
07DB  0E                        PUSH    CS
07DC  58                        POP     AX
07DD  A3 0062 R                  MOV     WORD PTR BASIC_PTR+2,AX ; CODE SEGMENT FOR CASSETTE
                        ;--------------------------------------------------------------------------
                        ; CHECK FOR OPTIONAL ROM FROM C0000 TO F0000 IN 2K BLOCKS
                        ;       (A VALID MODULE HAS '55AA' IN THE FIRST 2 LOCATIONS,
                        ;       LENGTH INDICATOR (LENGTH/512) IN THE 3D LOCATION AND
                        ;       TEST/INIT. CODE STARTING IN THE 4TH LOCATION.)
                        ;       MFG ERR CODE 25XX (XX=MSB OF SEGMENT THAT HAS CRC CHECK)
                        ;--------------------------------------------------------------------------
07E0  B0 01                     MOV     AL,01H
07E2  E6 13                     OUT     13H,AL
07E4  E8 E6D8 R                  CALL    MFG_UP          ; MFG ROUTINE = EE
07E7  BA C000                   MOV     DX,0C000H       ; SET BEGINNING ADDRESS
07EA                    ROM_SCAN_1:
07EA  8E DA                     MOV     DS,DX
07EC  2B DB                     SUB     BX,BX           ; SET BX=0000
07EE  8B 07                     MOV     AX,[BX]         ; GET 1ST WORD FROM MODULE
07F0  53                        PUSH    BX
07F1  5B                        POP     BX              ; BUS SETTLING
07F2  3D AA55                   CMP     AX,0AA55H       ; = TO ID WORD?
07F5  75 05                     JNZ     NEXT_ROM        ; PROCEED TO NEXT ROM IF NOT
07F7  E8 EB51 R                  CALL    ROM_CHECK       ; GO CHECK OUT MODULE
07FA  EB 04                     JMP     SHORT ARE_WE_DONE ; CHECK FOR END OF ROM SPACE
07FC                    NEXT_ROM:
07FC  81 C2 0080                ADD     DX,0080H        ; POINT TO NEXT 2K ADDRESS
0800                    ARE_WE_DONE:
0800  81 FA F000                CMP     DX,0F000H       ; AT F0000 YET?
0804  7C E4                     JL      ROM_SCAN_1      ; GO CHECK ANOTHER ADD. IF NOT
```

```
                        ;--------------------------------------------------------------------
                        ;       DISKETTE ATTACHMENT TEST
                        ; DESCRIPTION
                        ;       CHECK IF IPL DISKETTE DRIVE IS ATTACHED TO SYSTEM.  IF
                        ;       ATTACHED, VERIFY STATUS OF NEC FDC AFTER A RESET.  ISSUE
                        ;       A RECAL AND SEEK CMD TO FDC AND CHECK STATUS.  COMPLETE
                        ;       SYSTEM INITIALIZATION THEN PASS CONTROL TO THE BOOT
                        ;       LOADER PROGRAM.
                        ;   MFG ERR CODES: 2601 RESET TO DISKETTE CONTROLLER CD. FAILED
                        ;                  2602 RECALIBRATE TO DISKETTE DRIVE FAILED
                        ;                  2603 WATCHDOG TIMER FAILED
                        ;--------------------------------------------------------------------
                                ASSUME  CS:CODE,DS:DATA
0806  E8 E6D8 R                   CALL    MFG_UP                  ; MFG ROUTINE = ED
0809  E8 138B R                   CALL    DDS                     ; POINT TO DATA AREA
080C  B0 FF                       MOV     AL,0FFH
080E  A2 0074 R                   MOV     TRACK0,AL               ; INIT DISKETTE SCRATCHPADS
0811  A2 0075 R                   MOV     TRACK1,AL
0814  A2 0076 R                   MOV     TRACK2,AL
0817  E4 62                       IN      AL,PORT_C               ; DISKETTE PRESENT?
0819  24 04                       AND     AL,00000100B
081B  74 03                       JZ      F10_0
081D  E9 08A3 R                   JMP     F15                     ; NO - BYPASS DISKETTE TEST
0820  80 0E 0010 R 01     F10_0:  OR      BYTE PTR EQUIP_FLAG,01H ; SET IPL DISKETTE
                                                                  ; INDICATOR IN EQUIP. FLAG
0825  83 3E 0072 R 00             CMP     RESET_FLAG,0            ; RUNNING FROM POWER-ON STATE?
082A  75 0E                       JNE     F10                     ; BYPASS WATCHDOG TEST
082C  B0 0A                       MOV     AL,00001010B            ; READ INT. REQUEST REGISTER CMD
082E  E6 20                       OUT     INTA00,AL               ;
0830  E4 20                       IN      AL,INTA00
0832  24 40                       AND     AL,01000000B            ; HAS WATCHDOG GONE OFF?
0834  75 04                       JNZ     F10                     ; PROCEED IF IT HAS
0836  B3 03                       MOV     BL,03H                  ; SET ERROR CODE
0838  EB 33                       JMP     SHORT F13
083A  B0 80               F10:    MOV     AL,FDC_RESET
083C  E6 F2                       OUT     0F2H,AL                 ; DISABLE WATCHDOG TIMER
083E  B4 00                       MOV     AH,0                    ; RESET NEC FDC
0840  8A D4                       MOV     DL,AH                   ; SET FOR DRIVE 0
0842  CD 13                       INT     13H                     ; VERIFY STATUS AFTER RESET
0844  F6 C4 FF                   TEST    AH,0FFH                 ; STATUS OK?
0847  B3 01                       MOV     BL,01H                  ; SET UP POSSIBLE ERROR CODE
0849  75 22                       JNZ     F13                     ; NO - FDC FAILED
                        ;----- TURN DRIVE 0 MOTOR ON
084B  B0 81                       MOV     AL,DRIVE_ENABLE+FDC_RESET ; TURN MOTOR ON,DRIVE 0
084D  E6 F2                       OUT     0F2H,AL                 ; WRITE FDC CONTROL REG
084F  2B C9                       SUB     CX,CX
0851  E2 FE               F11:    LOOP    F11                     ; WAIT FOR 1 SECOND
0853  E2 FE               F12:    LOOP    F12
0855  33 D2                       XOR     DX,DX                   ; SELECT DRIVE 0
0857  B5 01                       MOV     CH,1                    ; SELECT TRACK 1
0859  88 16 003E R                MOV     SEEK_STATUS,DL
085D  E8 E9F8 R                   CALL    SEEK                    ; RECALIBRATE DISKETTE
0860  B3 02                       MOV     BL,02H                  ; ERROR CODE
0862  72 09                       JC      F13                     ; GO TO ERR SUBROUTINE IF ERR
0864  B5 22                       MOV     CH,34                   ; SELECT TRACK 34
0866  E8 E9FB R                   CALL    SEEK                    ; SEEK TO TRACK 34
0869  73 0A                       JNC     F14                     ; OK, TURN MOTOR OFF
086B  B3 02                       MOV     BL,02H
086D  B7 26               F13:    MOV     BH,26H                  ; DSK_ERR:(26XX)
086F  8E 003C R                   MOV     SI,OFFSET DISK_ERR ; GET ADDR OF MSG
0872  E8 09BC R                   CALL    E_MSG                   ; GO PRINT ERROR MSG
0875  B0 82               F14:    MOV     AL,FDC_RESET+02H
0877  E6 F2                       OUT     0F2H,AL
0879  E4 E2                       IN      AL,0E2H
087B  24 06                       AND     AL,00000110B
087D  3C 02                       CMP     AL,00000010B
087F  75 1E                       JNE     F14_1
0881  B0 84                       MOV     AL,FDC_RESET+04H
0883  E6 F2                       OUT     0F2H,AL
0885  E4 E2                       IN      AL,0E2H
0887  24 06                       AND     AL,00000110B
0889  3C 04                       CMP     AL,00000100B
088B  75 12                       JNE     F14_1
088D  E4 E2                       IN      AL,0E2H
088F  24 30                       AND     AL,00110000B
0891  74 0C                       JZ      F14_1
0893  3C 10                       CMP     AL,00010000B
0895  B4 40                       MOV     AH,01000000B
0897  74 02                       JE      F14_2
0899  B4 80                       MOV     AH,10000000B
089B  08 26 0010 R        F14_2:  OR      BYTE PTR EQUIP_FLAG,AH
                        ;----- TURN DRIVE 0 MOTOR OFF
089F  B0 80               F14_1:  MOV     AL,FDC_RESET    ; TURN DRIVE 0 MOTOR OFF
08A1  E6 F2                       OUT     0F2H,AL
08A3  C6 06 0084 R 00     F15:    MOV     INTR_FLAG,00H   ; SET STRAY INTERRUPT FLAG = 00
08A8  BF 0078 R                   MOV     DI,OFFSET PRINT_TIM_OUT ;SET DEFAULT PRT TIMEOUT
08AB  1E                          PUSH    DS
08AC  07                          POP     ES
08AD  B8 1414                     MOV     AX,1414H        ; DEFAULT=20
08B0  AB                          STOSW
08B1  AB                          STOSW
08B2  B8 0101                     MOV     AX,0101H        ;RS232 DEFAULT=01
08B5  AB                          STOSW
08B6  AB                          STOSW
08B7  E4 21                       IN      AL,INTA01
08B9  24 FE                       AND     AL,0FEH         ; ENABLE TIMER INT. (LVL 0)
08BB  E6 21                       OUT     INTA01,AL
                                  ASSUME  DS:XXDATA
08BD  1E                          PUSH    DS
08BE  B8  ---- R                  MOV     AX,XXDATA
08C1  8E D8                       MOV     DS,AX
```

```
08C3  80 3E 0018 R 00                 CMP     POST_ERR,00H    ; CHECK FOR "POST_ERR" NON-ZERO
                                      ASSUME  DS:DATA
08C8  1F                              POP     DS
08C9  74 10                           JE      F15A_0          ; CONTINUE IF NO ERROR
08CB  B2 02                           MOV     DL,2            ; 2 SHORT BEEPS (ERROR)
08CD  E8 1A0C R                       CALL    ERR_BEEP
08D0                            ERR_WAIT:
08D0  B4 00                           MOV     AH,00
08D2  CD 16                           INT     16H             ; WAIT FOR "ENTER" KEY
08D4  80 FC 1C                        CMP     AH,1CH
08D7  75 F7                           JNE     ERR_WAIT
08D9  EB 05                           JMP     SHORT F15C
08DB  B2 01                     F15A_0: MOV   DL,1            ; 1 SHORT BEEP (NO ERRORS)
08DD  E8 1A0C R                       CALL    ERR_BEEP
                                ;----- SETUP PRINTER AND RS232 BASE ADDRESSES IF DEVICE ATTACHED
08E0  BD 003D R                 F15C: MOV     BP,OFFSET F4    ; PRT_SRC_TBL
08E3  33 F6                           XOR     SI,SI
08E5                            F16:                          ; PRT_BASE:
08E5  2E: 8B 56 00                    MOV     DX,CS:[BP]      ; GET PRINTER BASE ADDR
08E9  B0 AA                           MOV     AL,0AAH         ; WRITE DATA TO PORT A
08EB  EE                              OUT     DX,AL
08EC  1E                              PUSH    DS              ; BUS SETTLING
08ED  EC                              IN      AL,DX           ; READ PORT A
08EE  1F                              POP     DS
08EF  3C AA                           CMP     AL,0AAH         ; DATA PATTERN SAME
08F1  75 06                           JNE     F17             ; NO - CHECK NEXT PRT CD
08F3  89 94 0008 R                    MOV     PRINTER_BASE[SI],DX ; YES - STORE PRT BASE ADDR
08F7  46                              INC     SI              ; INCREMENT TO NEXT WORD
08F8  46                              INC     SI
08F9  45                        F17:  INC     BP              ; POINT TO NEXT BASE ADDR
08FA  45                              INC     BP
08FB  83 FD 41                        CMP     BP,OFFSET F4E   ; ALL POSSIBLE ADDRS CHECKED?
08FE  75 E5                           JNE     F16             ; PRT_BASE
0900  33 DB                           XOR     BX,BX           ; SET ADDRESS BASE
0902  BA 03FA                         MOV     DX,03FAH        ; POINT TO INT ID REGISTER
0905  EC                              IN      AL,DX           ; READ PORT
0906  A8 F8                           TEST    AL,0F8H         ; SEEM TO BE AN 8250
0908  75 08                           JNZ     F18
090A  C7 87 0000 R 03F8               MOV     RS232_BASE[BX],3F8H ; SETUP RS232 CD #1 ADDR
0910  43                              INC     BX
0911  43                              INC     BX
0912  C7 87 0000 R 02F8         F18:  MOV     RS232_BASE[BX],2F8H ; SETUP RS232 #2
0918  43                              INC     BX              ; (ALWAYS PRESENT)
0919  43                              INC     BX
                                ;----- SET UP EQUIP FLAG TO INDICATE NUMBER OF PRINTERS AND RS232
                                ;       CARDS
091A  8B C6                           MOV     AX,SI           ; SI HAS 2* NUMBER OF PRINTERS
091C  B1 03                           MOV     CL,3            ; SHIFT COUNT
091E  D2 C8                           ROR     AL,CL           ; ROTATE RIGHT 3 POSITIONS
0920  0A C3                           OR      AL,BL           ; OR IN THE RS232 COUNT
0922  08 06 0011 R                    OR      BYTE PTR EQUIP_FLAG+1,AL ; STORE AS SECOND BYTE
                                ;----- SET EQUIP. FLAG TO INDICATE PRESENCE OF SERIAL PRINTER
                                ; ATTACHED TO ON BOARD RS232 PORT. ---ASSUMPTION--"RTS" IS TIED TO
                                ; "CARRIER DETECT" IN THE CABLE PLUG FOR THIS SPECIFIC PRINTER.
0926  8B C8                           MOV     CX,AX           ; SAVE PRINTER COUNT IN CX
0928  BB 02FE                         MOV     BX,2FEH         ; SET POINTER TO MODEM STATUS REG
092B  BA 02FC                         MOV     DX,2FCH         ; POINT TO MODEM CONTROL REG
092E  2A C0                           SUB     AL,AL
0930  EE                              OUT     DX,AL           ; CLEAR IT
0931  EB 00                           JMP     $+2             ; DELAY
0933  87 D3                           XCHG    DX,BX           ; POINT TO MODEM STATUS REG
0935  EC                              IN      AL,DX           ; CLEAR IT
0936  EB 00                           JMP     $+2             ; DELAY
0938  B0 02                           MOV     AL,02H          ; BRING UP RTS
093A  87 D3                           XCHG    DX,BX           ; POINT TO MODEM CONTROL REG
093C  EE                              OUT     DX,AL           ;
093D  EB 00                           JMP     $+2             ; DELAY
093F  87 D3                           XCHG    DX,BX           ; POINT TO MODEM STATUS REG
0941  EC                              IN      AL,DX           ; GET CONTENTS
0942  A8 08                           TEST    AL,00001000B    ; HAS CARRIER DETECT CHANGED?
0944  74 23                           JZ      F19_A           ; NO, THEN NO PRINTER
0946  A8 01                           TEST    AL,00000001B    ; DID CTS CHANGE? (AS WITH WRAP
                                                              ; CONNECTOR INSTALLED)
0948  75 1F                           JNZ     F19_A           ; WRAP CONNECTOR ON IF IT DID
094A  2A C0                           SUB     AL,AL           ; SET RTS OFF
094C  87 D3                           XCHG    DX,BX           ; POINT TO MODEM CONTROL REG
094E  EE                              OUT     DX,AL           ; DROP RTS
094F  EB 00                           JMP     $+2             ; DELAY
0951  87 D3                           XCHG    DX,BX           ; MODEM STATUS REG
0953  EC                              IN      AL,DX           ; GET STATUS
0954  24 08                           AND     AL,00001000B    ; HAS CARRIER DETECT CHANGED?
0956  74 11                           JZ      F19_A           ; NO, THEN NO PRINTER
                                ; CARRIER DETECT IS FOLLOWING RTS-INDICATE SERIAL PRINTER ATTACHED
0958  80 C9 20                        OR      CL,00100000B    ;
095B  F6 C1 C0                        TEST    CL,11000000B    ; CHECK FOR NO PARALLEL PRINTERS
095E  75 09                           JNZ     F19_A           ; DO NOTHING IF PARALLEL PRINTER
                                                              ; ATTACHED
0960  80 C9 40                        OR      CL,01000000B    ; INDICATE 1 PRINTER ATTACHED
0963  C7 06 0008 R 02F8               MOV     PRINTER_BASE,2F8H ; STORE ON-BOARD RS232 BASE IN
                                                              ; PRINTER BASE
0969  08 0E 0011 R              F19_A: OR      BYTE PTR EQUIP_FLAG+1,CL ; STORE AS SECOND BYTE
096D  33 D2                           XOR     DX,DX           ; POINT TO FIRST SERIAL PORT
096F  F6 C1 40                        TEST    CL,040H         ; SERIAL PRINTER ATTACHED?
0972  74 18                           JZ      F19_C           ; NO, SKIP INIT
0974  81 3E 0000 R 02F8               CMP     RS232_BASE,02F8H ; PRINTER IN FIRST SERIAL PORT
097A  74 01                           JE      F19_B           ; YES, JUMP
097C  42                              INC     DX              ; NO POINT TO SECOND SERIAL PORT
097D  B8 0087                  F19_B: MOV     AX,87H          ; INIT SERIAL PRINTER
0980  CD 14                           INT     14H
0982  F6 C4 1E                        TEST    AH,1EH          ; ERROR?
0985  75 05                           JNZ     F19_C           ; YES, JUMP
0987  B8 0118                         MOV     AX,0118H        ; SEND CANCEL COMMAND TO
098A  CD 14                           INT     14H             ; ..SERIAL PRINTER
```

```
098C  BA 0201                 F19_C:  MOV     DX,0201H
098F  EC                              IN      AL,DX                   ; GET MFG. / SERVICE  MODE INFO
0990  24 F0                           AND     AL,0F0H                 ; IS HIGH ORDER NIBBLE = 0?
0992  75 03                           JNZ     F19_1                   ; (BURN-IN MODE)
0994  E9 0043 R              F19_0:  JMP     START                   ; ELSE GO TO BEGINNING OF POST
0997  3C 20                  F19_1:  CMP     AL,00100000B            ; SERVICE MODE LOOP?
0999  74 F9                           JE      F19_0                   ; BRANCH TO START
099B  81 3E 0072 R 4321               CMP     RESET_FLAG,4321H        ; DIAG. CONTROL PROGRAM RESTART?
09A1  74 0C                           JE      F19_3                   ; NO,  GO BOOT
09A3  3C 10                           CMP     AL,00010000B            ; MFG DCP RUN REQUEST
09A5  74 08                           JE      F19_3
09A7  C7 06 0072 R 1234               MOV     RESET_FLAG,1234H        ; SET WARM START INDICATOR IN CASE
                                                                      ; OF CARTRIDGE RESET
09AD  CD 19                           INT     19H                     ; GO TO THE BOOT LOADER
                                      ASSUME  DS:ABS0
09AF  FA                      F19_3:  CLI
09B0  2B C0                           SUB     AX,AX
09B2  8E D8                           MOV     DS,AX                   ; RESET TIMER INT.
09B4  C7 06 0020 R FEA5 R             MOV     INT_PTR,OFFSET TIMER_INT
09BA  CD 80                           INT     80H                     ; ENTER DCP THROUGH INT. 80H
                              ;-----------------------------------------------------------------------
                              ;       THIS SUBROUTINE IS THE GENERAL ERROR HANDLER FOR THE POST
                              ;
                              ; ENTRY REQUIREMENTS:
                              ;       SI = OFFSET(ADDRESS) OF MESSAGE BUFFER
                              ;       BX= ERROR CODE FOR MANUFACTURING OR SERVICE MODE
                              ;       REGISTERS ARE NOT PRESERVED
                              ;       LOCATION "POST_ERR" IS SET NON-ZERO IF AN ERROR OCCURS IN
                              ;       CUSTOMER MODE
                              ;       SERVICE/MANUFACTURING FLAGS AS FOLLOWS: (HIGH NIBBLE OF
                              ;       PORT 201)
                              ;       0000 = MANUFACTURING (BURN-IN) MODE
                              ;       0001 = MANUFACTURING (SYSTEM TEST) MODE
                              ;       0010 = SERVICE MODE (LOOP POST)
                              ;       0100 = SERVICE MODE (SYSTEM TEST)
                              ;-----------------------------------------------------------------------
09BC                          E_MSG   PROC    NEAR
09BC  BA 0201                         MOV     DX,201H
09BF  EC                              IN      AL,DX                   ; GET MODE BITS
09C0  24 F0                           AND     AL,0F0H                 ; ISOLATE BITS OF INTEREST
09C2  75 03                           JNZ     EM0
09C4  E9 0A61 R                       JMP     MFG_OUT                 ; MANUFACTURING MODE (BURN-IN)
09C7  3C 10                  EM0:     CMP     AL,00010000B            ;
09C9  75 03                           JNE     EM1
09CB  E9 0A61 R                       JMP     MFG_OUT                 ; MFG. MODE (SYSTEM TEST)
09CE  8A F0                  EM1:     MOV     DH,AL                   ; SAVE MODE
09D0  80 FF 0A                        CMP     BH,0AH                  ; ERROR CODE ABOVE 0AH (CRT STARTED
                                                                      ;  DISPLAY POSSIBLE)?
09D3  7C 63                           JL      BEEPS                   ; DO BEEP OUTPUT IF BELOW 10H
09D5  53                              PUSH    BX                      ; SAVE ERROR AND MODE FLAGS
09D6  56                              PUSH    SI
09D7  52                              PUSH    DX
09D8  B4 02                           MOV     AH,2                    ; SET CURSOR
09DA  BA 1521                         MOV     DX,1521H                ; ROW 21, COL. 33
09DD  B7 07                           MOV     BH,7                    ; PAGE 7
09DF  CD 10                           INT     10H
09E1  BE 0030 R                       MOV     SI,OFFSET ERROR_ERR
09E4  B9 0005                         MOV     CX,5                    ; PRINT WORD "ERROR"
09E7  2E: 8A 04              EM_0:    MOV     AL,CS:[SI]
09EA  46                              INC     SI
09EB  E8 18BA R                       CALL    PRT_HEX
09EE  E2 F7                           LOOP    EM_0
                              ; LOOK FOR A BLANK SPACE TO POSSIBLY PUT CUSTOMER LEVEL ERRORS (IN
                              ; CASE OF MULTI ERROR)
09F0  B6 16                           MOV     DH,16H
09F2  B4 02                  EM_1:    MOV     AH,2                    ; SET CURSOR
09F4  CD 10                           INT     10H                     ; ROW 22, COL33 (OR ABOVE, IF
                                                                      ; MULTIPLE ERRS)
09F6  B4 08                           MOV     AH,8                    ; READ CHARACTER THIS POSITION
09F8  CD 10                           INT     10H
09FA  FE C2                           INC     DL                      ; POINT TO NEXT POSTION
09FC  3C 20                           CMP     AL,' '                  ; BLANK?
09FE  75 F2                           JNE     EM_1                    ; GO CHECK NEXT POSITION, IF NOT
0A00  5A                              POP     DX                      ; RECOVER ERROR POINTERS
0A01  5E                              POP     SI
0A02  5B                              POP     BX
0A03  80 FE 20                        CMP     DH,00100000B            ; SERVICE MODE?
0A06  74 21                           JE      SERV_OUT                ;
0A08  80 FE 40                        CMP     DH,01000000B            ;
0A0B  74 1C                           JE      SERV_OUT
0A0D  2E: 8A 04                       MOV     AL,CS:[SI]              ; GET ERROR CHARACTER
0A10  E8 18BA R                       CALL    PRT_HEX                 ; DISPLAY IT
0A13  80 FF 20                        CMP     BH,20H                  ; ERROR BELOW 20? (MEM TROUBLE?)
0A16  7D 03                           JNL     EM_2
0A18  E9 0ABB R                       JMP     TOTLTP0                 ; HALT SYSTEM IF SO.
                                      ASSUME  DS:XXDATA               ;
0A1B  1E                      EM_2:   PUSH    DS
0A1C  50                              PUSH    AX
0A1D  B8  ---- R                      MOV     AX,XXDATA
0A20  8E D8                           MOV     DS,AX
0A22  88 3E 0018 R                    MOV     POST_ERR,BH             ; SET ERROR FLAG NON-ZERO
0A26  58                              POP     AX
0A27  1F                              POP     DS
                                      ASSUME  DS:NOTHING
0A28  C3                              RET                             ; RETURN TO CALLER
```

```
0A29                            SERV_OUT:
0A29  8A C7                             MOV     AL,BH           ; PRINT MSB
0A2B  53                                PUSH    BX
0A2C  E8 18A9 R                         CALL    XPC_BYTE        ; DISPLAY IT
0A2F  5B                                POP     BX
0A30  8A C3                             MOV     AL,BL           ; PRINT LSB
0A32  E8 18A9 R                         CALL    XPC_BYTE
0A35  E9 0A88 R                         JMP     TOTLTPO
0A38  FA                        BEEPS:  CLI
0A39  8C C8                             MOV     AX,CS           ; SET CODE SEG= STACK SEG
0A3B  8E D0                             MOV     SS,AX           ; (STACK IS LOST, BUT THINGS ARE
                                                                ; OVER, ANYWAY)
0A3D  B2 02                             MOV     DL,2            ; 2 BEEPS
0A3F  BC 0028 R                         MOV     SP,OFFSET EX_O  ; SET DUMMY RETURN
0A42  B3 01                     EB:     MOV     BL,1            ; SHORT BEEP
0A44  E9 FF31 R                         JMP     BEEP            ;
0A47  E2 FE                     EBO:    LOOP    EBO             ; WAIT (BEEPER OFF)
0A49  FE CA                             DEC     DL              ; DONE YET?
0A4B  75 F5                             JNZ     EB              ; LOOP IF NOT
0A4D  80 FF 05                          CMP     BH,05H          ; 64K CARD ERROR?
0A50  75 69                             JNE     TOTLTPO         ; END IF NOT
0A52  80 FE 20                          CMP     DH,00100000B    ; SERVICE MODE?
0A55  74 05                             JE      EB1
0A57  80 FE 40                          CMP     DH,01000000B    ;
0A5A  75 5F                             JNE     TOTLTPO         ; END IF NOT
0A5C  B3 01                     EB1:    MOV     BL,1            ; ONE MORE BEEP FOR 64K ERROR IF IN
                                                                ; SERVICE MODE
0A5E  E9 FF31 R                         JMP     BEEP
0A61                            MFG_OUT:
0A61  FA                                CLI
0A62  E4 61                             IN      AL,PORT_B
0A64  24 FC                             AND     AL,0FCH
0A66  E6 61                             OUT     PORT_B,AL
0A68  BA 0011                           MOV     DX,11H          ; SEND DATA TO  ADDRESSES 11,12
0A6B  8A C7                             MOV     AL,BH           ;
0A6D  EE                                OUT     DX,AL           ; SEND HIGH BYTE
0A6E  42                                INC     DX
0A6F  8A C3                             MOV     AL,BL           ;
0A71  EE                                OUT     DX,AL           ; SEND LOW BYTE
                                ; INIT. ON-BOARD RS232 PORT FOR COMMUNICATIONS W/MFG MONITOR
                                        ASSUME  DS:XXDATA
0A72  B8  ---- R                        MOV     AX,XXDATA
0A75  8E D8                             MOV     DS,AX           ; POINT TO DATA SEGMENT CONTAINING
                                                                ; CHECKPOINT #
0A77  8C C8                             MOV     AX,CS
0A79  8E D0                             MOV     SS,AX           ; SET STACK FOR RTN
0A7B  BC 002E R                         MOV     SP,OFFSET EX1
0A7E  BA 02FB                           MOV     DX,02FBH        ; LINE CONTROL REG. ADDRESS
0A81  E9 F085 R                         JMP     S8250           ; GO SET UP FOR 9600, ODD, 2 STOP
                                                                ; BITS, 8 BITS
0A84  8B CA                     M01:    MOV     CX,DX           ; DX CAME BACK WITH XMIT REG
                                                                ; ADDRESS IN IT
0A86  BA 02FC                           MOV     DX,02FCH        ; MODEM CONTROL REG
0A89  2A C0                             SUB     AL,AL           ; SET DTR AND RTS LOW SO POSSIBLE
                                                                ; WRAP PLUG WON'T CONFUSE THINGS
0A8B  EE                                OUT     DX,AL
0A8C  BA 02FE                           MOV     DX,02FEH        ; MODEM STATUS REG
0A8F  EC                        M02:    IN      AL,DX
0A90  24 10                             AND     AL,00010000B    ; CTS UP YET?
0A92  74 FB                             JZ      M02             ; LOOP TILL IT IS
0A94  4A                                DEC     DX              ; SET DX=2FD (LINE STATUS REG)
0A95  87 D1                             XCHG    DX,CX           ; POINT TO XMIT. DATA REG
0A97  A0 0005 R                         MOV     AL,MFG_TST      ; GET MFG ROUTINE ERROR INDICATOR
0A9A  EE                                OUT     DX,AL           ; (MAY BE WRONG FOR EARLY ERRORS)
0A9B  EB 00                             JMP     $+2             ; DELAY
0A9D  87 D1                             XCHG    DX,CX           ; POINT DX=2FD
0A9F  EC                        M03:    IN      AL,DX           ; TRANSMIT EMPTY?
0AA0  24 20                             AND     AL,00100000B
0AA2  EB 00                             JMP     $+2             ; DELAY
0AA4  74 F9                             JZ      M03             ; LOOP TILL IT IS
0AA6  87 D1                             XCHG    DX,CX
0AA8  8A C7                             MOV     AL,BH           ; GET MSB OF ERROR WORD
0AAA  EE                                OUT     DX,AL
0AAB  EB 00                             JMP     $+2             ; DELAY
0AAD  87 D1                             XCHG    DX,CX
0AAF  EC                        M04:    IN      AL,DX           ; WAIT FOR XMIT EMPTY
0AB0  24 20                             AND     AL,00100000B
0AB2  EB 00                             JMP     $+2             ; DELAY
0AB4  74 F9                             JZ      M04
0AB6  8A C3                             MOV     AL,BL           ; GET LSB OF ERROR WORD
0AB8  87 D1                             XCHG    DX,CX
0ABA  EE                                OUT     DX,AL
0ABB                            TOTLTPO:
0ABB  FA                                CLI                     ; DISABLE INTS.
0ABC  2A C0                             SUB     AL,AL           ;
0ABE  E6 F2                             OUT     0F2H,AL         ; STOP DISKETTE MOTOR
0AC0  E6 A0                             OUT     0A0H,AL         ; DISABLE NMI
0AC2  F4                                HLT                     ; HALT
0AC3  C3                                RET
0AC4                            E_MSG   ENDP
```

```
                        ;--------------------------------------------------------
                        ;   SUBROUTINE TO INITIALIZE INS8250 PORTS TO THE MASTER RESET
                        ;       STATUS.  THIS ROUTINE ALSO TESTS THE PORTS' PERMANENT
                        ;       ZERO BITS.
                        ;   EXPECTS TO BE PASSED:
                        ;       (DX) = ADDRESS OF THE 8250 TRANSMIT/RECEIVE BUFFER
                        ;   UPON RETURN:
                        ;       (CF) = 1  IF ONE OF THE PORTS' PERMANENT ZERO BITS WAS NOT
                        ;                 ZERO (ERR)
                        ;                 (DX) = PORT ADDRESS THAT FAILED TEST
                        ;                 (AL) = MEANINGLESS
                        ;                 (BL) = 2  INTR ENBL REG  BITS NOT 0
                        ;                        3  INTR ID REG BITS NOT 0
                        ;                        4  MODEM CTRL REG  BITS NOT 0
                        ;                        5  LINE STAT REG BITS NOT 0
                        ;              0  IF ALL PORTS' PERMANENT ZERO BITS WERE ZERO
                        ;                 (DX) = TRANSMIT/RECEIVE BUFFER ADDRESS
                        ;                 (AL) = LAST VALUE READ FROM RECEIVER BUFFER
                        ;                 (BL) = 5 (MEANINGLESS)
                        ;       PORTS SET UP AS FOLLOWS ON ERROR-FREE RETURN:
                        ;        XF9 - INTR ENBL REG  = 0          ALL INTERRUPTS DISABLED
                        ;        XFA - INTR ID REG    = 00000001B  NO INTERRUPTS PENDING
                        ;        XFB - LINE CTRL REG  = 0          ALL BITS LOW
                        ;        XFC - MODEM CTRL REG = 0          ALL BITS LOW
                        ;        XFD - LINE STAT REG  = 01100000B  TRANSMITTER HOLDING
                        ;                                          REGISTER AND TRANSMITTER EMPTY ON
                        ;        XFE - MODEM STAT REG = XXXX0000B  WHERE X 'S REPRESENT
                        ;                                          INPUT SIGNALS
                        ;   REGISTERS DX, AL, AND BL ARE ALTERED.  NO OTHER REGISTERS USED.
                        ;--------------------------------------------------------
0AC4                    I8250   PROC    NEAR
0AC4  EC                        IN      AL,DX           ; READ RECVR BUFFER BUT IGNORE
                                                        ; CONTENTS
0AC5  B3 02                     MOV     BL,2            ; ERROR INDICATOR
0AC7  E8 FE9F R                 CALL    RR2             ; READ INTR ENBL REG
0ACA  24 F0                     AND     AL,11110000B    ; BITS 4-7 OFF?
0ACC  75 28                     JNE     AT20            ; NO - ERROR
0ACE  E8 FE9A R                 CALL    RR1             ; READ INTR ID REG
0AD1  24 F8                     AND     AL,11111000B    ; BITS 3-7 OFF?
0AD3  75 21                     JNE     AT20            ; NO
0AD5  42                        INC     DX              ; LINE CTRL REG
0AD6  E8 FE9A R                 CALL    RR1             ; READ MODEM CTRL REG
0AD9  24 E0                     AND     AL,11100000B    ; BITS 5-7 OFF?
0ADB  75 19                     JNE     AT20            ; NO
0ADD  E8 FE9A R                 CALL    RR1             ; READ LINE STAT REG
0AE0  24 80                     AND     AL,10000000B    ; BIT 7 OFF?
0AE2  75 12                     JNE     AT20            ; NO
0AE4  B0 60                     MOV     AL,60H
0AE6  EE                        OUT     DX,AL
0AE7  EB 00                     JMP     $+2             ; I/O DELAY
0AE9  42                        INC     DX              ; MODEM STAT REG
0AEA  32 C0                     XOR     AL,AL
0AEC  EE                        OUT     DX,AL           ; WIRED BITS WILL BE HIGH
0AED  E8 FEA0 R                 CALL    RR3             ; CLEAR BITS 0-3 IN CASE THEY'RE ON
                                                        ; AFTER WRITING TO STATUS REG
0AF0  83 EA 06                  SUB     DX,6            ; RECEIVER BUFFER
0AF3  EC                        IN      AL,DX           ; IN CASE WRITING TO PORTS CAUSED
                                                        ; DATA READY TO GO HIGH!
0AF4  F8                        CLC
0AF5  C3                        RET
0AF6  F9                AT20:   STC                     ; ERROR RETURN
0AF7  C3                        RET
0AF8                    I8250   ENDP
                        ;--------------------------------------------------------
                        ;   SUBROUTINE TO TEST A PARTICULAR 8250 INTERRUPT.  PASS IT THE
                        ;       (BIT # + 1) OF THE STATUS REGISTER THAT IS TO BE TESTED.
                        ;       THIS ROUTINE SETS THAT BIT AND CHECKS TO SEE IF THE CORRECT
                        ;       8250 INTERRUPT IS GENERATED.
                        ;   IT EXPECTS TO BE PASSED:
                        ;       (AH) = BIT # TO BE TESTED
                        ;       (BL) = INTERRUPT IDENTIFIER
                        ;           (0) = RECEIVED DATA AVAILABLE OR TRANSMITTER HOLDING
                        ;                 REGISTER EMPTY INTERRUPT TEST
                        ;           (1) = RECEIVER LINE STATUS OR MODEM STATUS INTERRUPT
                        ;                 TEST
                        ;       (BH) = BITS WHICH DETERMINE WHICH INTERRUPT IS TO BE
                        ;              CHECKED
                        ;           (0) = MODEM STATUS
                        ;           (2) = TRANSMITTER HOLDING REGISTER EMPTY
                        ;           (4) = RECEIVED DATA AVAILABLE
                        ;           (6) = RECEIVER LINE STATUS
                        ;       (CX) = VALUE TO SUBTRACT AND ADD IN ORDER TO REFERENCE THE
                        ;              INTERRUPT IDENTIFICATION REGISTER
                        ;           (3) = RECEIVED DATA AVAILABLE, TRANSMITTER HOLDING
                        ;                 REGISTER AND RECEIVER LINE STATUS INTERRUPTS
                        ;           (4) = MODEM STATUS INTERRUPT
                        ;       (DX) = ADDRESS OF THE LINE STATUS OR MODEM STATUS REGISTER
                        ;              IT RETURNS:
                        ;       (AL) = OFFH IF TEST FAILS - EITHER NO INTERRUPT OCCURRED OR
                        ;              THE WRONG INTERRUPT OCCURRED
                        ;    OR
                        ;       (AL) = CONTENTS OF THE INTERRUPT ID REGISTER FOR RECEIVED
                        ;          DATA AVAILABLE AND TRANSMITTER HOLDING REGISTER
                        ;          EMPTY INTERRUPTS
                        ;          -OR-
                        ;          CONTENTS OF THE LINE STATUS OR MODEM STATUS REGISTER
                        ;              DEPENDING ON WHICH ONE WAS TESTED.
                        ;       (DX) = ADDRESS OF INTERRUPT ID REGISTER FOR RECEIVED DATA
                        ;              AVAILABLE OR TRANSMITTER HOLDING REGISTER EMPTY
                        ;              INTERRUPTS
                        ;    OR
                        ;       (DX) = ADDRESS OF THE LINE STATUS OR DATA SET STATUS
                        ;              REGISTER (DEPENDING ON WHICH INTERRUPT WAS TESTED)
                        ;   NO OTHER REGISTERS ARE ALTERED.
                        ;--------------------------------------------------------
```

```
0AF8                            ICT     PROC    NEAR
0AF8  EC                                IN      AL,DX           ; READ STATUS REGISTER
0AF9  EB 00                             JMP     $+2             ; I/O DELAY
0AFB  0A C4                             OR      AL,AH           ; SET TEST BIT
0AFD  EE                                OUT     DX,AL           ; WRITE IT TO THE STATUS REGISTER
0AFE  2B D1                             SUB     DX,CX           ; POINT TO INTERRUPT ID REGISTER
0B00  51                                PUSH    CX
0B01  2B C9                             SUB     CX,CX           ; WAIT FOR 8250 INTERRUPT TO OCCUR
0B03  EC                        AT21:   IN      AL,DX           ; READ INTR ID REG
0B04  A8 01                             TEST    AL,1            ; INTERRUPT PENDING?
0B06  74 02                             JE      AT22            ; YES -RETURN W/ INTERRUPT ID IN AL
0B08  E2 F9                             LOOP    AT21            ; NO - TRY AGAIN
0B0A  59                        AT22:   POP     CX              ; AL = 1 IF NO INTERRUPT OCCURRED
0B0B  3A C7                             CMP     AL,BH           ; INTERRUPT WE'RE LOOKING FOR?
0B0D  75 09                             JNE     AT23            ; NO
0B0F  0A DB                             OR      BL,BL           ; DONE WITH TEST FOR THIS INTERRUPT
0B11  74 07                             JE      AT24            ; RETURN W/ CONTENTS OF INTR ID REG
0B13  03 D1                             ADD     DX,CX           ; READ STATUS REGISTER TO CLEAR THE
0B15  EC                                IN      AL,DX           ; INTERRUPT (WHEN BL=1)
0B16  EB 02                             JMP     SHORT AT24      ; RETURN CONTENTS OF STATUS REG
0B18  B0 FF                     AT23:   MOV     AL,0FFH         ; SET ERROR INDICATOR
0B1A  C3                        AT24:   RET
0B1B                            ICT     ENDP
                                ;--- INT 19 -----------------------------------------
                                ; BOOT STRAP LOADER
                                ;       TRACK 0, SECTOR 1 IS READ INTO THE
                                ;       BOOT LOCATION (SEGMENT 0, OFFSET 7C00)
                                ;       AND CONTROL IS TRANSFERRED THERE.
                                ;
                                ;       IF THE DISKETTE IS NOT PRESENT OR HAS A
                                ;       PROBLEM LOADING (E.G., NOT READY), AN INT.
                                ;       18H IS EXECUITED. IF A CARTRIDGE HAS VECTORED
                                ;       INT. 18H TO ITSELF, CONTROL WILL BE PASSED TO
                                ;       THE CARTRIDGE.
                                ;-------------------------------------------------
                                        ASSUME  CS:CODE,DS:ABS0
0B1B                            BOOT_STRAP      PROC    NEAR
0B1B  FB                                STI                     ; ENABLE INTERRUPTS
0B1C  2B C0                             SUB     AX,AX           ; SET 40X25 B&W MODE ON CRT
0B1E  CD 10                             INT     10H             ;
0B20  2B C0                             SUB     AX,AX           ; ESTABLISH ADDRESSING
0B22  8E D8                             MOV     DS,AX
                                ;----- SEE IF DISKETTE PRESENT
0B24  E4 62                             IN      AL,PORT_C       ; GET CONFIG BITS
0B26  24 04                             AND     AL,00000100B    ; IS DISKETTE PRESENT?
0B28  75 28                             JNZ     H3              ; NO, THEN ATTEMPT TO GO TO CART.
                                ;----- RESET THE DISK PARAMETER TABLE VECTOR
0B2A  C7 06 0078 R EFC7 R               MOV     WORD PTR DISK_POINTER, OFFSET DISK_BASE
0B30  8C 0E 007A R                      MOV     WORD PTR DISK_POINTER+2,CS
                                ;-----  LOAD SYSTEM FROM DISKETTE -- CX HAS RETRY COUNT
0B34  B9 0004                           MOV     CX,4            ; SET RETRY COUNT
0B37  51                        H1:     PUSH    CX              ; SAVE RETRY COUNT
0B38  B4 00                             MOV     AH,0            ; RESET THE DISKETTE SYSTEM
0B3A  CD 13                             INT     13H             ; DISKETTE_IO
0B3C  72 0F                             JC      H2              ; IF ERROR, TRY AGAIN
0B3E  B8 0201                           MOV     AX,201H         ; READ IN THE SINGLE SECTOR
0B41  2B D2                             SUB     DX,DX           ; TO THE BOOT LOCATION
0B43  8E C2                             MOV     ES,DX
0B45  BB 7C00 R                         MOV     BX,OFFSET BOOT_LOCN
                                                                ; DRIVE 0, HEAD 0
0B48  B9 0001                           MOV     CX,1            ; SECTOR 1, TRACK 0
0B4B  CD 13                             INT     13H             ; DISKETTE_IO
0B4D  59                        H2:     POP     CX              ; RECOVER RETRY COUNT
0B4E  73 04                             JNC     H3A             ; CF SET BY UNSUCCESSFUL READ
0B50  E2 E5                             LOOP    H1              ; DO IT FOR RETRY TIMES
                                ;----- UNABLE TO IPL FROM THE DISKETTE
0B52  CD 18                     H3:     INT     18H             ; GO TO BASIC OR CARTRIDGE
                                ;----- IPL WAS SUCCESSFUL
0B54  EA 7C00 ---- R            H3A:    JMP     BOOT_LOCN
0B59                            BOOT_STRAP      ENDP
                                ;--------------------------------------------------------
                                ; THIS ROUTINE PERFORMS A READ/WRITE TEST ON A BLOCK OF
                                ; STORAGE (MAX. SIZE = 32KB). IF "WARM START", FILL
                                ; BLOCK WITH 0000 AND RETURN.
                                ; DATA PATTERNS USED:
                                ;       0->FF ON ONE BYTE TO TEST DATA BUS
                                ;       AAAA,5555,00FF,FF00 FOR ALL WORDS
                                ;       FILL WITH 0000 BEFORE EXIT
                                ; ON ENTRY:
                                ;   ES = ADDRESS OF STORAGE TO BE TESTED
                                ;   DS = ADDRESS OF STORAGE TO BE TESTED
                                ;   CX = WORD COUNT OF STORAGE BLOCK TO BE TESTED
                                ;         (MAX. = 8000H (32K WORDS))
                                ; ON EXIT:
                                ;   ZERO FLAG = OFF IF STORAGE ERROR
                                ;   IF ZERO FLAG = OFF, THEN CX = XOR'ED BIT PATTERN
                                ;   OF THE EXPECTED DATA PATTERN VS. THE ACTUAL DATA
                                ;   READ. (I.E., A BIT "ON" IN AL IS THE BIT IN ERROR)
                                ;   AH=03 IF BOTH BYTES OF WORD HAVE ERRORS
                                ;   AH=02 IF LOW (EVEN) BYTE HAS ERROR
                                ;   AH=01 IF HI (ODD) BYTE HAS ERROR
                                ; AX,BX,CX,DX,DI,SI ARE ALL DESTROYED.
                                ;--------------------------------------------------------
```

```
0B59                            PODSTG  PROC    NEAR
                                        ASSUME  DS:ABS0
0B59  FC                                CLD                     ; SET DIRECTION TO INCREMENT
0B5A  2B FF                             SUB     DI,DI           ; SET DI=0000 REL. TO START OF SEG
0B5C  2B C0                             SUB     AX,AX           ; INITIAL DATA PATTERN FOR 00-FF
                                                                ; TEST
0B5E  8E D8                             MOV     DS,AX           ; SET DS TO ABS0
0B60  8B 1E 0472 R                      MOV     BX,DATA_WORD[RESET_FLAG-DATA] ; WARM START?
0B64  81 FB 1234                        CMP     BX,1234H
0B68  8C C2                             MOV     DX,ES
0B6A  8E DA                             MOV     DS,DX           ; RESTORE DS
0B6C  75 08                             JNE     P1
0B6E  F3/ AB                    P12:    REP     STOSW           ; SIMPLE FILL WITH 0 ON WARM-START
0B70  8E D8                             MOV     DS,AX
0B72  89 1E 0472 R                      MOV     DATA_WORD[RESET_FLAG-DATA],BX
0B76  8E DA                             MOV     DS,DX           ; RESTORE DS
0B78  C3                                RET                     ; AND EXIT
0B79  81 FB 4321                P1:     CMP     BX,4321H        ; DIAG. RESTART?
0B7D  74 EF                             JE      P12             ; DO FILL WITH ZEROS
0B7F  88 05                     P2:     MOV     [DI],AL         ; WRITE TEST DATA
0B81  8A 05                             MOV     AL,[DI]         ; GET IT BACK
0B83  32 C4                             XOR     AL,AH           ; COMPARE TO EXPECTED
0B85  74 03                             JZ      PY
0B87  E9 0C0C R                         JMP     P8              ; ERROR EXIT IF MISCOMPARE
0B8A  FE C4                     PY:     INC     AH              ; FORM NEW DATA PATTERN
0B8C  8A C4                             MOV     AL,AH           ;
0B8E  75 EF                             JNZ     P2              ; LOOP TILL ALL 256 DATA PATTERNS
                                                                ;  DONE
0B90  8B E9                             MOV     BP,CX           ; SAVE WORD COUNT
0B92  B8 AAAA                           MOV     AX,0AAAAH       ; LOAD DATA PATTERN
0B95  8B D8                             MOV     BX,AX
0B97  BA 5555                           MOV     DX,05555H       ; LOAD OTHER DATA PATTERN
0B9A  F3/ AB                            REP     STOSW           ; FILL WORDS FROM LOW TO HIGH
                                                                ; WITH AAAA
0B9C  4F                                DEC     DI              ; POINT TO LAST WORD WRITTEN
0B9D  4F                                DEC     DI
0B9E  FD                                STD                     ; SET DIRECTION FLAG TO GO DOWN
0B9F  8B F7                             MOV     SI,DI           ; SET INDEX REGS. EQUAL
0BA1  8B CD                             MOV     CX,BP           ; RECOVER WORD COUNT
0BA3                            P3:                             ; GO FROM HIGH TO LOW
0BA3  AD                                LODSW                   ; GET WORD FROM MEMORY
0BA4  33 C3                             XOR     AX,BX           ; EQUAL WHAT S/B THERE?
0BA6  75 64                             JNZ     P8              ; GO ERROR EXIT IF NOT
0BA8  8B C2                             MOV     AX,DX           ; GET 55 DATA PATTERN
0BAA  AB                                STOSW                   ;  STORE IT IN LOCATION JUST READ
0BAB  E2 F6                             LOOP    P3              ; LOOP TILL ALL BYTES DONE
0BAD  8B CD                             MOV     CX,BP           ; RECOVER WORD COUNT
0BAF  FC                                CLD                     ; BACK TO INCREMENT
0BB0  46                                INC     SI              ; ADJUST PTRS
0BB1  46                                INC     SI
0BB2  8B FE                             MOV     DI,SI
0BB4  8B DA                             MOV     BX,DX           ; S/B DATA PATTERN TO BX
0BB6  BA 00FF                           MOV     DX,00FFH        ; DATA FOR CHECKERBOARD PATTERN
0BB9  AD                        PX:     LODSW                   ; GET WORD FROM MEMORY
0BBA  33 C3                             XOR     AX,BX           ; EQUAL WHAT S/B THERE?
0BBC  75 4E                             JNZ     P8              ; GO ERROR EXIT IF NOT
0BBE  8B C2                             MOV     AX,DX           ; GET OTHER PATTERN
0BC0  AB                                STOSW                   ; STORE IT IN LOCATION JUST READ
0BC1  E2 F6                             LOOP    PX              ; LOOP TILL ALL BYTES DONE
0BC3  8B CD                             MOV     CX,BP           ; RECOVER WORD COUNT
0BC5  FD                                STD                     ; DECREMENT
0BC6  4E                                DEC     SI              ; ADJUST PTRS
0BC7  4E                                DEC     SI
0BC8  8B FE                             MOV     DI,SI
0BCA  8B DA                             MOV     BX,DX           ; S/B DATA PATTERN TO BX
0BCC  F7 D2                             NOT     DX              ; MAKE PATTERN FF00
0BCE  0A D2                             OR      DL,DL           ; FIRST PASS?
0BD0  74 E7                             JZ      PX
0BD2  FC                                CLD                     ; INCREMENT
0BD3  83 C6 04                          ADD     SI,4
0BD6  F7 D2                             NOT     DX
0BD8  8B FE                             MOV     DI,SI
0BDA  8B CD                             MOV     CX,BP
0BDC                            P4:                             ; LOW TO HIGH
0BDC  AD                                LODSW                   ; GET A WORD
0BDD  33 C2                             XOR     AX,DX           ; SHOULD COMPARE TO DX
0BDF  75 2B                             JNZ     P8              ; GO ERROR IF NOT
0BE1  AB                                STOSW                   ; WRITE 0000 BACK TO LOCATION
                                                                ; JUST READ
0BE2  E2 F8                             LOOP    P4              ; LOOP TILL DONE
0BE4  FD                                STD                     ; BACK TO DECREMENT
0BE5  4E                                DEC     SI              ; ADJUST POINTER DOWN TO LAST WORD
0BE6  4E                                DEC     SI              ; WRITTEN
                                ; CHECK IF IN SERVICE/MFG MODES, IF SO, PERFORM REFRESH CHECK
0BE7  BA 0201                           MOV     DX,201H         ;
0BEA  EC                                IN      AL,DX           ; GET OPTION BITS
0BEB  24 F0                             AND     AL,0F0H         ;
0BED  3C F0                             CMP     AL,0F0H         ; ALL BITS HIGH=NORMAL MODE
0BEF  74 10                             JE      P6
0BF1  8C C9                             MOV     CX,CS
0BF3  8C D3                             MOV     BX,SS
0BF5  3B CB                             CMP     CX,BX           ; SEE IF IN PRE-STACK MODE
0BF7  74 08                             JE      P6              ; BYPASS RETENTION TEST IF SO
0BF9  B0 18                             MOV     AL,24           ; SET OUTER LOOP COUNT
                                ; WAIT ABOUT 6-8 SECONDS WITHOUT ACCESSING MEMORY
                                ; IF REFRESH IS NOT WORKING PROPERLY, THIS SHOULD
                                ; BE ENOUGH TIME FOR SOME DATA TO GO SOUR.
```

```
0BFB  E2 FE                     P5:     LOOP    P5
0BFD  FE C8                             DEC     AL
0BFF  75 FA                             JNZ     P5
0C01  8B CD                     P6:     MOV     CX,BP           ; RECOVER WORD COUNT
0C03  AD                        P7:     LODSW                   ; GET WORD
0C04  0B C0                             OR      AX,AX           ; = TO 0000
0C06  75 04                             JNZ     P8              ; ERROR IF NOT
0C08  E2 F9                             LOOP    P7              ; LOOP TILL DONE
0C0A  EB 13                             JMP     SHORT P11       ; THEN EXIT
0C0C  8B C8                     P8:     MOV     CX,AX           ; SAVE BITS IN ERROR
0C0E  32 E4                             XOR     AH,AH
0C10  0A ED                             OR      CH,CH           ; HIGH BYTE ERROR?
0C12  74 02                             JZ      P9
0C14  FE C4                             INC     AH              ; SET HIGH BYTE ERROR
0C16  0A C9                     P9:     OR      CL,CL           ; LOW BYTE ERROR?
0C18  74 03                             JZ      P10
0C1A  80 C4 02                          ADD     AH,2
0C1D  0A E4                     P10:    OR      AH,AH           ; SET ZERO FLAG=0 (ERROR INDICATION
0C1F  FC                        P11:    CLD                     ; SET DIR FLAG BACK TO INCREMENT
0C20  C3                                RET                     ; RETURN TO CALLER
0C21                            PODSTG  ENDP
                                ;--------------------------------------------------
                                ; PUT_LOGO PROCEDURE
                                ;       THIS PROC SETS UP POINTERS AND CALLS THE SCREEN
                                ;   OUTPUT ROUTINE SO THAT THE IBM LOGO, A MESSAGE,
                                ;   AND A COLOR BAR ARE PUT UP ON THE SCREEN.
                                ;   AX,BX, AND DX ARE DESTROYED. ALL OTHERS ARE SAVED
                                ;--------------------------------------------------
0C21                            PUT_LOGO PROC   NEAR
0C21  1E                                PUSH    DS
0C22  55                                PUSH    BP
0C23  50                                PUSH    AX
0C24  53                                PUSH    BX
0C25  51                                PUSH    CX
0C26  52                                PUSH    DX
0C27  BD 0C4A R                         MOV     BP,OFFSET LOGO  ;
0C2A  BA 8000                           MOV     DX,8000H        ;POINT DH DL AT ROW,COLUMN 0,0
0C2D  B3 1F                             MOV     BL,00011111B    ;ATTRIBUTE OF CHARACTERS TO BE
                                                                ;WRITTEN
0C2F  CD 82                             INT     82H             ;CALL OUTPUT ROUTINE
0C31  B3 00                             MOV     BL,00000000B    ; INITIALIZE ATTRIBUTE
0C33  B2 00                             MOV     DL,0            ; INITIALIZE COLUMN
0C35  B6 94                     AGAIN:  MOV     DH,94H          ; SET LINE
0C37  BD 0CDD R                         MOV     BP,OFFSET COLOR ; OUTPUT GIVEN COLOR BAR
0C3A  CD 82                             INT     82H             ;CALL OUTPUT ROUTINE
0C3C  FE C3                             INC     BL              ; INCREMENT ATTRIBUTE
0C3E  80 FA 20                          CMP     DL,32           ; IS THE COLUMN COUNTER POINTING
                                                                ; PAST 40?
0C41  7C F2                             JL      AGAIN           ; IF NOT, DO IT AGAIN
0C43  5A                                POP     DX
0C44  59                                POP     CX
0C45  5B                                POP     BX
0C46  58                                POP     AX
0C47  5D                                POP     BP              ; RESTORE BP
0C48  1F                                POP     DS              ; RESTORE DS
0C49  C3                                RET
0C4A                            PUT_LOGO ENDP
0C4A  03                        LOGO    DB      LOGO_E - LOGO
0C4B  20 DC                             DB      ' ',220
= 0C4D                          LOGO_E  =       $
0C4D  28 FB                             DB      40,-5
0C4F  28 FB                             DB      40,-5
0C51  02 07 01 09 03 04                 DB      2,7,1,9,3,4,9,4,1,-5
      09 04 01 FB
0C5B  02 07 01 0A 02 05                 DB      2,7,1,10,2,5,7,5,1,-5
      07 05 01 FB
0C65  02 07 01 0B 01 06                 DB      2,7,1,11,1,6,5,6,1,-5
      05 06 01 FB
0C6F  04 03 05 03 03 03                 DB      4,3,5,3,3,3,3,5,3,5,3,-5
      03 05 03 05 03 FB
0C7B  04 03 05 03 03 03                 DB      4,3,5,3,3,3,3,6,1,6,3,-5
      03 06 01 06 03 FB
0C87  04 03 05 08 04 0D                 DB      4,3,5,8,4,13,3,-5
      03 FB
0C8F  04 03 05 07 05 0D                 DB      4,3,5,7,5,13,3,-5
      03 FB
0C97  04 03 05 08 04 0D                 DB      4,3,5,8,4,13,3,-5
      03 FB
0C9F  04 03 05 03 03 03                 DB      4,3,5,3,3,3,3,13,3,-5
      03 0D 03 FB
0CA9  04 03 05 03 03 03                 DB      4,3,5,3,3,3,3,3,1,5,1,3,3,-5
      03 03 01 05 01 03
      03 FB
0CB7  02 07 01 0B 01 05                 DB      2,7,1,11,1,5,2,3,2,5,1,-5
      02 03 02 05 01 FB
0CC3  02 07 01 0A 02 05                 DB      2,7,1,10,2,5,3,1,3,5,1,-5
      03 01 03 05 01 FB
0CCF  02 07 01 09 03 05                 DB      2,7,1,9,3,5,7,5,1,-5
      07 05 01 FB
0CD9  28 FB                             DB      40,-5
0CDB  28 FC                             DB      40,-4
0CDD  02                        COLOR   DB      COLOR_E - COLOR
0CDE  DB                                DB      219
= 0CDF                          COLOR_E =       $
0CDF  02 77 02 77 02 77                 DB      2,121-2,2,121-2,2,121-2,2,121-2,2,-4
      02 77 02 FC

                                        ASSUME  DS:DATA
```

```
;--- INT 10 ------------------------------------------------------------
; VIDEO_IO
;       THESE ROUTINES PROVIDE THE CRT INTERFACE
;       THE FOLLOWING FUNCTIONS ARE PROVIDED:
;       (AH)=0  SET MODE (AL) CONTAINS MODE VALUE
;               (AL)=0 40X25 BW (POWER ON DEFAULT)
;               (AL)=1 40X25 COLOR
;               (AL)=2  80X25 BW
;               (AL)=3  80X25 COLOR
;               GRAPHICS MODES
;               (AL)=4  320X200 4 COLOR
;               (AL)=5  320X200 BW 4 SHADES
;               (AL)=6  640X200 BW 2 SHADES
;               (AL)=7  NOT VALID
;        **** EXTENDED MODES ***
;               (AL)=8  160X200 16 COLOR
;               (AL)=9  320X200 16 COLOR
;               (AL)=A  640X200 4 COLOR
;               *** NOTE BW MODES OPERATE SAME AS COLOR MODES, BUT
;                       COLOR BURST IS NOT ENABLED
;               *** NOTE IF HIGH ORDER BIT IN AL IS SET, THE REGEN
;                       BUFFER IS NOT CLEARED.
;       (AH)=1  SET CURSOR TYPE
;               (CH) =  BITS 4-0 = START LINE FOR CURSOR
;               ** HARDWARE WILL ALWAYS CAUSE BLINK
;               ** SETTING BIT 5 OR 6 WILL CAUSE ERRATIC
;                  BLINKING OR NO CURSOR AT ALL
;               ** IN GRAPHICS MODES, BIT 5 IS FORCED ON TO
;                  DISABLE THE CURSOR
;               (CL) =  BITS 4-0 = END LINE FOR CURSOR
;       (AH)=2  SET CURSOR POSITION
;               (DH,DL) = ROW,COLUMN  (0,0) IS UPPER LEFT
;               (BH) = PAGE NUMBER (MUST BE 0 FOR GRAPHICS MODES)
;       (AH)=3  READ CURSOR POSITION
;               (BH) = PAGE NUMBER (MUST BE 0 FOR GRAPHICS MODES)
;               ON EXIT (DH,DL) = ROW,COLUMN OF CURRENT CURSOR
;                       (CH,CL) = CURSOR MODE CURRENTLY SET
;       (AH)=4  READ LIGHT PEN POSITION
;               ON EXIT:
;            (AH) = 0 -- LIGHT PEN SWITCH NOT DOWN/NOT TRIGGERED
;            (AH) = 1 -- VALID LIGHT PEN VALUE IN REGISTERS
;                        (DH,DL) = ROW,COLUMN OF CHARACTER LP POSN
;                        (CH) = RASTER LINE (0-199)
;                        (BX) = PIXEL COLUMN (0-319,639)
;       (AH)=5  SELECT ACTIVE DISPLAY PAGE (VALID ONLY FOR
;                       ALPHA MODES)
;               (AL)=NEW PAGE VALUE (0-7 FOR MODES 0&1, 0-3 FOR
;                       MODES 2&3)
;               IF BIT 7 (80H) OF AL=1
;                       READ/WRITE CRT/CPU PAGE REGISTERS
;               (AL) = 80H READ CRT/CPU PAGE REGISTERS
;               (AL) = 81H SET CPU PAGE REGISTER
;                       (BL) = VALUE TO SET
;               (AL) = 82H SET CRT PAGE REGISTER
;                       (BH) = VALUE TO SET
;               (AL) = 83H SET BOTH CRT AND CPU PAGE REGISTERS
;                       (BL) = VALUE TO SET IN CPU PAGE REGISTER
;                       (BH) = VALUE TO SET IN CRT PAGE REGISTER
;               IF BIT 7 (80H) OF AL=1
;                  ALWAYS  RETURNS (BH) = CONTENTS OF CRT PAGE REG
;                                  (BL) = CONTENTS OF CPU PAGE REG
;       (AH)=6  SCROLL ACTIVE PAGE UP
;               (AL) = NUMBER OF LINES, INPUT LINES BLANKED AT
;                      BOTTOM OF WINDOW, AL = 0 MEANS BLANK
;                      ENTIRE WINDOW
;               (CH,CL) = ROW,COLUMN OF UPPER LEFT CORNER OF
;                         SCROLL
;               (DH,DL) = ROW,COLUMN OF LOWER RIGHT CORNER OF
;                          SCROLL
;               (BH) = ATTRIBUTE TO BE USED ON BLANK LINE
;       (AH)=7  SCROLL ACTIVE PAGE DOWN
;               (AL) = NUMBER OF LINES, INPUT LINES BLANKED AT TOP
;                      OF WINDOW, AL=0 MEANS BLANK ENTIRE WINDOW
;               (CH,CL) = ROW,COLUMN OF UPPER LEFT CORNER OF
;                         SCROLL
;               (DH,DL) = ROW,COLUMN OF LOWER RIGHT CORNER OF
;                         SCROLL
;               (BH) = ATTRIBUTE TO BE USED ON BLANK LINE
;
;       CHARACTER HANDLING ROUTINES
;       (AH) = 8 READ ATTRIBUTE/CHARACTER AT CURRENT CURSOR POSITION
;               (BH) = DISPLAY PAGE (VALID FOR ALPHA MODES ONLY)
;               ON EXIT:
;               (AL) = CHAR READ
;               (AH) = ATTRIBUTE OF CHARACTER READ (ALPHA MODES
;                      ONLY)
;       (AH) = 9 WRITE ATTRIBUTE/CHARACTER AT CURRENT CURSOR
;              POSITION
;               (BH) = DISPLAY PAGE (VALID FOR ALPHA MODES ONLY)
;               (CX) = COUNT OF CHARACTERS TO WRITE
;               (AL) = CHAR TO WRITE
;               (BL) = ATTRIBUTE OF CHARACTER (ALPHA)/COLOR OF
;                      CHARACTER (GRAPHICS). SEE NOTE ON WRITE
;                      DOT FOR BIT 7 OF BL = 1.
;       (AH) = 10 (0AH) WRITE CHARACTER ONLY AT CURRENT CURSOR
;                       POSITION
;               (BH) = DISPLAY PAGE (VALID FOR ALPHA MODES ONLY)
;               (CX) = COUNT OF CHARACTERS TO WRITE
;               (AL) = CHAR TO WRITE
;               (BL) = COLOR OF CHAR (GRAPHICS)
;                       SEE NOTE ON WRITE DOT FOR BIT 7 OF BL = 1.
```

```
;       FOR READ/WRITE CHARACTER INTERFACE WHILE IN GRAPHICS MODE,
;               THE CHARACTERS ARE FORMED FROM A CHARACTER
;               GENERATOR IMAGE MAINTAINED IN THE SYSTEM ROM.
;               INTERRUPT 44H (LOCATION 00110H) IS USED TO
;               POINT TO THE 1K BYTE TABLE CONTAINING THE
;               FIRST 128 CHARS (0-127).
;               INTERRUPT 1FH (LOCATION 0007CH) IS USED TO
;               POINT TO THE 1K BYTE TABLE CONTAINING THE SECOND
;               128 CHARS (128-255).
;       FOR WRITE CHARACTER INTERFACE IN GRAPHICS MODE, THE
;               REPLICATION FACTOR CONTAINED IN (CX) ON ENTRY WILL
;               PRODUCE VALID RESULTS ONLY FOR CHARACTERS
;               CONTAINED ON THE SAME ROW.  CONTINUATION TO
;               SUCCEEDING LINES WILL NOT PRODUCE CORRECTLY.
;
;       GRAPHICS INTERFACE
;       (AH) = 11 (0BH) SET COLOR PALETTE
;               (BH) = PALETTE COLOR ID BEING SET (0-127)
;               (BL) = COLOR VALUE TO BE USED WITH THAT COLOR ID
;                       COLOR ID = 0 SELECTS THE BACKGROUND
;                                    COLOR (0-15)
;                       COLOR ID = 1 SELECTS THE PALETTE TO BE
;                          USED:
;                          2 COLOR MODE:
;                               0 = WHITE FOR COLOR 1
;                               1 = BLACK FOR COLOR 1
;                          4 COLOR MODES:
;                               0 = GREEN, RED, BROWN FOR
;                                   COLORS 1,2,3
;                               1 = CYAN, MAGENTA, WHITE FOR
;                                   COLORS 1,2,3
;                          16 COLOR MODES:
;                               ALWAYS SETS UP PALETTE AS:
;                               BLUE FOR COLOR 1
;                               GREEN FOR COLOR 2
;                               CYAN FOR COLOR 3
;                               RED FOR COLOR 4
;                               MAGENTA FOR COLOR 5
;                               BROWN FOR COLOR 6
;                               LIGHT GRAY FOR COLOR 7
;                               DARK GRAY FOR COLOR 8
;                               LIGHT BLUE FOR COLOR 9
;                               LIGHT GREEN FOR COLOR 10
;                               LIGHT CYAN FOR COLOR 11
;                               LIGHT RED FOR COLOR 12
;                               LIGHT MAGENTA FOR COLOR 13
;                               YELLOW FOR COLOR 14
;                               WHITE FOR COLOR 15
;                IN 40X25 OR 80X25 ALPHA MODES, THE VALUE SET
;                       FOR PALETTE COLOR 0 INDICATES THE BORDER
;                       COLOR TO BE USED. IN GRAPHIC MODES, IT
;                       INDICATES THE BORDER COLOR AND THE
;                       BACKGROUND COLOR.
;       (AH) = 12 (0CH) WRITE DOT
;               (DX) = ROW NUMBER
;               (CX) = COLUMN NUMBER
;               (AL) = COLOR VALUE
;                  IF BIT 7 OF AL = 1, THEN THE COLOR VALUE IS
;                  EXCLUSIVE OR'D WITH THE CURRENT CONTENTS OF
;                  THE DOT
;       (AH) = 13 (0DH) READ DOT
;               (DX) = ROW NUMBER
;               (CX) = COLUMN NUMBER
;               (AL) RETURNS THE DOT READ
;
; ASCII TELETYPE ROUTINE FOR OUTPUT
;       (AH) = 14 (0EH) WRITE TELETYPE TO ACTIVE PAGE
;               (AL) = CHAR TO WRITE
;               (BL) = FOREGROUND COLOR IN GRAPHICS MODE
;               NOTE -- SCREEN WIDTH IS CONTROLLED BY PREVIOUS
;                       MODE SET
;       (AH) = 15 (0FH) CURRENT VIDEO STATE
;               RETURNS THE CURRENT VIDEO STATE
;               (AL) = MODE CURRENTLY SET (SEE AH=0 FOR
;                      EXPLANATION)
;               (AH) = NUMBER OF CHARACTER COLUMNS ON SCREEN
;               (BH) = CURRENT ACTIVE DISPLAY PAGE
;       (AH) = 16 (10H) SET PALETTE REGISTERS
;               (AL) = 0 SET PALETTE REGISTER
;                       (BL) = PALETTE REGISTER TO SET (00H - 0FH)
;                       (BH) = VALUE TO SET
;               (AL) = 1 SET BORDER COLOR REGISTER
;                       (BH) = VALUE TO SET
;               (AL) = 2 SET ALL PALETTE REGISTERS AND BORDER
;                        REGISTER
;                       ES:DX POINTS TO A 17 BYTE LIST
;                       BYTES 0 THRU 15 ARE VALUES FOR PALETTE
;                               REGISTERS 0 THRU 15
;                       BYTE 16 IS THE VALUE FOR THE BORDER
;                               REGISTER
;       NOTE:
;       IN MODES USING A 32K REGEN (9 AND A), ACCESS THROUGH THE CPU
;       REGISTER BY USE OF B800H SEGMENT VALUE ONLY REACHES THE
;       FIRST 16K. BIOS USES THE CONTENTS OF THE CPU PAGE REG
;       (BITS 3,4, & 5 OF PAGDAT IN BIOS DATA AREA) TO DERIVE THE
;       PROPER SEGMENT VALUE.
;
;       CS,SS,DS,ES,BX,CX,DX PRESERVED DURING CALL
;       ALL OTHERS DESTROYED
```

```
                        ;--------------------------------------------------
                        ; VIDEO GATE ARRAY REGISTERS
                        ;
                        ;  PORT 3DA OUTPUT
                        ;       REG 0   MODE CONTROL 1 REGISTER
                        ;       01H     +HI BANDWIDTH/-LOW BANDWIDTH
                        ;       02H     +GRAPHICS/-ALPHA
                        ;       04H     +B&W
                        ;       08H     +VIDEO ENABLE
                        ;       10H     +16 COLOR GRAPHICS
                        ;
                        ;       REG 1   PALETTE MASK REISTER
                        ;       01H     PALETTE MASK 0
                        ;       02H     PALETTE MASK 1
                        ;       04H     PALETTE MASK 2
                        ;       08H     PALETTE MASK 3
                        ;
                        ;       REG 2   BORDER COLOR REGISTER
                        ;       01H     BLUE
                        ;       02H     GREEN
                        ;       04H     RED
                        ;       08H     INTENSITY
                        ;
                        ;       REG 3   MODE CONTROL 2 REGISTER
                        ;       01H     RESERVED -- MUST BE ZERO
                        ;       02H     +ENABLE BLINK
                        ;       04H     RESERVED -- MUST BE ZERO
                        ;       08H     +2 COLOR GRAPHICS (640X200 2 COLOR ONLY)
                        ;
                        ;       REG 4   RESET REGISTER
                        ;       01H     +ASYNCHRONOUS RESET
                        ;       02H     +SYNCHRONOUS RESET
                        ;
                        ;       REGS 10 TO 1F   PALETTE REGISTERS
                        ;       01H     BLUE
                        ;       02H     GREEN
                        ;       04H     RED
                        ;       08H     INTENSITY
                        ;
                        ;  VIDEO GATE ARRAY STATUS
                        ;    PORT 3DA INPUT
                        ;       01H     +DISPLAY ENABLE
                        ;       02H     +LIGHT PEN TRIGGER SET
                        ;       04H     -LIGHT PEN SWITCH MADE
                        ;       08H     +VERTICAL RETRACE
                        ;       10H     +VIDEO DOTS
                                ASSUME  CS:CODE,DS:DATA,ES:VIDEO_RAM
0CE9                    M0010   LABEL   WORD    ;  TABLE OF ROUTINES WITHIN VIDEO I/O
0CE9  0DA5 R                    DW      OFFSET  SET_MODE
0CEB  E45E R                    DW      OFFSET  SET_CTYPE
0CED  E488 R                    DW      OFFSET  SET_CPOS
0CEF  E52D R                    DW      OFFSET  READ_CURSOR
0CF1  F751 R                    DW      OFFSET  READ_LPEN
0CF3  E4B3 R                    DW      OFFSET  ACT_DISP_PAGE
0CF5  E5D3 R                    DW      OFFSET  SCROLL_UP
0CF7  E63F R                    DW      OFFSET  SCROLL_DOWN
0CF9  F0E4 R                    DW      OFFSET  READ_AC_CURRENT
0CFB  F113 R                    DW      OFFSET  WRITE_AC_CURRENT
0CFD  F12C R                    DW      OFFSET  WRITE_C_CURRENT
0CFF  E543 R                    DW      OFFSET  SET_COLOR
0D01  F187 R                    DW      OFFSET  WRITE_DOT
0D03  F146 R                    DW      OFFSET  READ_DOT
0D05  1992 R                    DW      OFFSET  WRITE_TTY
0D07  E5B1 R                    DW      OFFSET  VIDEO_STATE
0D09  E685 R                    DW      OFFSET  SET_PALLETTE
= 0022                  M0010L  EQU     $-M0010

0D0B                    VIDEO_IO        PROC    NEAR
0D0B  FB                        STI                     ; INTERRUPTS BACK ON
0D0C  FC                        CLD                     ; SET DIRECTION FORWARD
0D0D  06                        PUSH    ES
0D0E  1E                        PUSH    DS              ; SAVE SEGMENT REGISTERS
0D0F  52                        PUSH    DX
0D10  51                        PUSH    CX
0D11  53                        PUSH    BX
0D12  56                        PUSH    SI
0D13  57                        PUSH    DI
0D14  50                        PUSH    AX              ; SAVE AX VALUE
0D15  8A C4                     MOV     AL,AH           ; GET INTO LOW BYTE
0D17  32 E4                     XOR     AH,AH           ; ZERO TO HIGH BYTE
0D19  D1 E0                     SAL     AX,1            ; *2 FOR TABLE LOOKUP
0D1B  8B F0                     MOV     SI,AX           ; PUT INTO SI FOR BRANCH
0D1D  3D 0022                   CMP     AX,M0010L       ; TEST FOR WITHIN RANGE
0D20  72 04                     JB      C1              ; BRANCH AROUND BRANCH
0D22  58                        POP     AX              ; THROW AWAY THE PARAMETER
0D23  E9 0F70 R                 JMP     VIDEO_RETURN    ; DO NOTHING IF NOT IN RANGE
0D26  E8 138B R         C1:     CALL    DDS
0D29  B8 B800                   MOV     AX,0B800H       ; SEGMENT FOR COLOR CARD
0D2C  80 3E 0049 R 09           CMP     CRT_MODE,9      ; IN MODE USING 32K REGEN
0D31  72 09                     JC      C2              ; NO,JUMP
0D33  8A 26 008A R              MOV     AH,PAGDAT       ; GET COPY OF PAGE REGS
0D37  80 E4 38                  AND     AH,CPUREG       ; ISOLATE CPU REG
0D3A  D0 EC                     SHR     AH,1            ; SHIFT TO MAKE INTO SEGMENT VALUE
0D3C  8E C0             C2:     MOV     ES,AX           ; SET UP TO POINT AT VIDEO RAM AREA
0D3E  58                        POP     AX              ; RECOVER VALUE
0D3F  8A 26 0049 R              MOV     AH,CRT_MODE     ; GET CURRENT MODE INTO AH
0D43  2E: FF A4 0CE9 R          JMP     WORD PTR CS:[SI+OFFSET M0010]
0D48                    VIDEO_IO        ENDP
```

```
                                ;--------------------------------------------
                                ; SET_MODE
                                ;       THIS ROUTINE INITIALIZES THE ATTACHMENT TO
                                ;       THE SELECTED MODE.  THE SCREEN IS BLANKED.
                                ; INPUT
                                ;       (AL) = MODE SELECTED (RANGE 0-B)
                                ; OUTPUT
                                ;       NONE
                                ;--------------------------------------------
0D48                            M0050   LABEL   WORD                    ; TABLE OF REGEN LENGTHS
0D48  0800                              DW      2048                    ; MODE 0 40X25  BW
0D4A  0800                              DW      2048                    ; MODE 1 40X25  COLOR
0D4C  1000                              DW      4096                    ; MODE 2 80X25  BW
0D4E  1000                              DW      4096                    ; MODE 3 80X25  COLOR
0D50  4000                              DW      16384                   ; MODE 4 320X200 4 COLOR
0D52  4000                              DW      16384                   ; MODE 5 320X200 4 COLOR
0D54  4000                              DW      16384                   ; MODE 6 640X200 BW
0D56  0000                              DW      0                       ; MODE 7 INVALID
0D58  4000                              DW      16384                   ; MODE 8 160X200 16 COLOR
0D5A  8000                              DW      32768                   ; MODE 9 320X200 16 COLOR
0D5C  8000                              DW      32768                   ; MODE A 640X200 4 COLOR
                                ;------ COLUMNS
0D5E                            M0060   LABEL   BYTE
0D5E  28 28 50 50 28 28                 DB      40,40,80,80,40,40,80,0,20,40,80
      50 00 14 28 50

                                ;------- TABLE OF GATE ARRAY PARAMATERS FOR MODE SETTING
0D69                            M0070   LABEL   BYTE
                                ;------ SET UP FOR 40X25 BW      MODE 0
0D69  0C 0F 00 02                       DB      0CH,0FH,0,2             ; GATE ARRAY PARMS
= 0004                          M0070L  EQU     $-M0070
                                ;------ SET UP FOR 40X25 COLOR   MODE 1
0D6D  08 0F 00 02                       DB      08H,0FH,0,2             ; GATE ARRAY PARMS
                                ;------ SET UP FOR 80X25 BW      MODE 2
0D71  0D 0F 00 02                       DB      0DH,0FH,0,2             ; GATE ARRAY PARMS
                                ;------ SET UP FOR 80X25 COLOR   MODE 3
0D75  09 0F 00 02                       DB      09H,0FH,0,2             ; GATE ARRAY PARMS
                                ;------ SET UP FOR 320X200  4 COLOR      MODE 4
0D79  0A 03 00 00                       DB      0AH,03H,0,0             ; GATE ARRAY PARMS
                                ;------ SET UP FOR 320X200  BW           MODE 5
0D7D  0E 03 00 00                       DB      0EH,03H,0,0             ; GATE ARRAY PARMS
                                ;------ SET UP FOR 640X200  BW           MODE 6
0D81  0E 01 00 08                       DB      0EH,01H,0,8             ; GATE ARRAY PARMS
                                ;------ SET UP FOR    INVALID           MODE 7
0D85  00 00 00 00                       DB      00H,00H,0,0             ; GATE ARRAY PARMS
                                ;------ SET UP FOR 160X200  16 COLOR    MODE 8
0D89  1A 0F 00 00                       DB      1AH,0FH,0,0             ; GATE ARRAY PARMS
                                ;------ SET UP FOR 320X200  16 COLOR    MODE 9
0D8D  1B 0F 00 00                       DB      1BH,0FH,0,0             ; GATE ARRAY PARMS
                                ;------ SET UP FOR 640X200   4 COLOR    MODE A
0D91  0B 03 00 00                       DB      0BH,03H,0,0             ; GATE ARRAY PARMS
                                ;--------------- TABLES OF PALETTE COLORS FOR 2 AND 4 COLOR MODES
                                ;-----  2 COLOR, SET 0
0D95                            M0072   LABEL   BYTE
0D95  00 0F 00 00                       DB      0,0FH,0,0
= 0004                          M0072L  EQU     $-M0072         ;ENTRY LENGTH
                                ;-----  2 COLOR, SET 1
0D99  0F 00 00 00                       DB      0FH,0,0,0
                                ;-----  4 COLOR, SET 0
0D9D                            M0074   LABEL   BYTE
0D9D  00 02 04 06                       DB      0,2,4,6
                                ;-----  4 COLOR, SET 1
0DA1                            M0075   LABEL   BYTE
0DA1  00 03 05 0F                       DB      0,3,5,0FH
0DA5                            SET_MODE        PROC    NEAR
0DA5  50                                PUSH    AX              ;SAVE INPUT MODE ON STACK
0DA6  24 7F                             AND     AL,7FH          ;REMOVE CLEAR REGEN SWITCH
0DA8  3C 07                             CMP     AL,7            ;CHECK FOR VALID MODES
0DAA  74 04                             JE      C3              ;MODE 7 IS INVALID
0DAC  3C 0B                             CMP     AL,0BH
0DAE  72 02                             JC      C4              ;GREATER THAN A IS INVALID
0DB0  B0 00                     C3:     MOV     AL,0            ;DEFAULT TO MODE 0
0DB2  3C 02                     C4:     CMP     AL,2            ;CHECK FOR MODES NEEDING 128K
0DB4  74 08                             JE      C5
0DB6  3C 03                             CMP     AL,3
0DB8  74 04                             JE      C5
0DBA  3C 09                             CMP     AL,09H
0DBC  72 0A                             JC      C6
0DBE  81 3E 0015 R 0080         C5:     CMP     TRUE_MEM,128    ;DO WE HAVE 128K?
0DC4  73 02                             JNC     C6              ;YES, JUMP
0DC6  B0 00                             MOV     AL,0            ;NO, DEFAULT TO MODE 0
0DC8  BA 03D4                   C6:     MOV     DX,03D4H        ; ADDRESS OF COLOR CARD
0DCB  8A E0                             MOV     AH,AL           ; SAVE MODE IN AH
0DCD  A2 0049 R                         MOV     CRT_MODE,AL     ; SAVE IN GLOBAL VARIABLE
0DD0  89 16 0063 R                      MOV     ADDR_6845,DX    ; SAVE ADDRESS OF BASE
0DD4  8B F8                             MOV     DI,AX           ; SAVE MODE IN DI
0DD6  BA 03DA                           MOV     DX,VGA_CTL      ; POINT TO CONTROL REGISTER
0DD9  EC                                IN      AL,DX           ; SYNC CONTROL REG TO ADDRESS
0DDA  32 C0                             XOR     AL,AL           ; SET VGA REG 0
0DDC  EE                                OUT     DX,AL           ; SELECT IT
0DDD  A0 0065 R                         MOV     AL,CRT_MODE_SET ; GET LAST MODE SET
0DE0  24 F7                             AND     AL,0F7H         ; TURN OFF VIDEO
0DE2  EE                                OUT     DX,AL           ; SET IN GATE ARRAY
```

```
                        ;----- SET DEFAULT PALETTES
0DE3  8B C7                     MOV     AX,DI           ; GET MODE
0DE5  B4 10                     MOV     AH,10H          ; SET PALETTE REG 0
0DE7  BB 0D95 R                 MOV     BX,OFFSET M0072 ; POINT TO TABLE ENTRY
0DEA  3C 06                     CMP     AL,6            ; 2 COLOR MODE?
0DEC  74 0F                     JE      C7              ; YES, JUMP
0DEE  BB 0DA1 R                 MOV     BX,OFFSET M0075 ; POINT TO TABLE ENTRY
0DF1  3C 05                     CMP     AL,5            ; CHECK FOR 4 COLOR MODE
0DF3  74 08                     JE      C7              ; YES, JUMP
0DF5  3C 04                     CMP     AL,4            ; CHECK FOR 4 COLOR MODE
0DF7  74 04                     JE      C7              ; YES JUMP
0DF9  3C 0A                     CMP     AL,0AH          ; CHECK FOR 4 COLOR MODE
0DFB  75 11                     JNE     C9              ; NO, JUMP
0DFD  B9 0004           C7:     MOV     CX,4            ; NUMBER OF REGS TO SET
0E00  8A C4             C8:     MOV     AL,AH           ; GET REG NUMBER
0E02  EE                        OUT     DX,AL           ; SELECT IT
0E03  2E: 8A 07                 MOV     AL,CS:[BX]      ; GET DATA
0E06  EE                        OUT     DX,AL           ; SET IT
0E07  FE C4                     INC     AH              ; NEXT REG
0E09  43                        INC     BX              ; NEXT TABLE VALUE
0E0A  E2 F4                     LOOP    C8
0E0C  EB 0B                     JMP     SHORT C11
                        ;----- SET PALETTES FOR DEFAULT 16 COLOR
0E0E  B9 0010           C9:     MOV     CX,16           ; NUMBER OF PALETTES, AH IS REG
                                                        ; COUNTER
0E11  8A C4             C10:    MOV     AL,AH           ; GET REG NUMBER
0E13  EE                        OUT     DX,AL           ; SELECT IT
0E14  EE                        OUT     DX,AL           ; SET PALETTE VALUE
0E15  FE C4                     INC     AH              ; NEXT REG
0E17  E2 F8                     LOOP    C10
                        ;----- SET UP M0 & M1 IN PAGREG
0E19  8B C7             C11:    MOV     AX,DI           ; GET CURRENT MODE
0E1B  32 DB                     XOR     BL,BL           ; SET UP FOR ALPHA MODE
0E1D  3C 04                     CMP     AL,4            ; IN ALPHA MODE
0E1F  72 08                     JC      C12             ; YES, JUMP
0E21  B3 40                     MOV     BL,40H          ; SET UP FOR 16K REGEN
0E23  3C 09                     CMP     AL,09H          ; MODE USE 16K
0E25  72 02                     JC      C12             ; YES, JUMP
0E27  B3 C0                     MOV     BL,0C0H         ; SET UP FOR 32K REGEN
0E29  BA 03DF           C12:    MOV     DX,PAGREG       ; SET PORT ADDRESS OF PAGREG
0E2C  A0 008A R                 MOV     AL,PAGDAT       ; GET LAST DATA OUTPUT
0E2F  24 3F                     AND     AL,3FH          ; CLEAR M0 & M1 BITS
0E31  0A C3                     OR      AL,BL           ; SET NEW BITS
0E33  EE                        OUT     DX,AL           ; STUFF BACK IN PORT
0E34  A2 008A R                 MOV     PAGDAT,AL       ; SAVE COPY IN RAM
                        ;----- ENABLE VIDEO AND CORRECT PORT SETTING
0E37  8B C7                     MOV     AX,DI           ; GET CURRENT MODE
0E39  32 E4                     XOR     AH,AH           ;   INTO AX REG
0E3B  B9 0004                   MOV     CX,M0070L       ; SET TABLE ENTRY LENGTH
0E3E  F7 E1                     MUL     CX              ; TIMES MODE FOR OFFSET INTO TABLE
0E40  8B D8                     MOV     BX,AX           ; TABLE OFFSET IN BX
0E42  81 C3 0D69 R              ADD     BX,OFFSET M0070 ; ADD TABLE START TO OFFSET
0E46  2E: 8A 27                 MOV     AH,CS:[BX]      ; SAVE MODE SET AND PALETTE
0E49  2E: 8A 47 02              MOV     AL,CS:[BX + 2]  ;   TILL WE CAN PUT THEM IN RAM
0E4D  8B F0                     MOV     SI,AX
0E4F  FA                        CLI                     ; DISABLE INTERRUPTS
0E50  E8 E675 R                 CALL    MODE_ALIVE      ; KEEP MEMORY DATA VALID
0E53  B0 10                     MOV     AL,10H          ; DISABLE NMI AND HOLD REQUEST
0E55  E6 A0                     OUT     NMI_PORT,AL     ;
0E57  BA 03DA                   MOV     DX,VGA_CTL      ;
0E5A  B0 04                     MOV     AL,4            ; POINT TO RESET REG
0E5C  EE                        OUT     DX,AL           ; SEND TO GATE ARRAY
0E5D  B0 02                     MOV     AL,2            ; SET SYNCHRONOUS RESET
0E5F  EE                        OUT     DX,AL           ; DO IT
                        ; WHILE THE GATE ARRAY IS IN RESET STATE, WE CANNOT ACCESS RAM
0E60  8B C6                     MOV     AX,SI           ; RESTORE NEW MODE SET
0E62  80 E4 F7                  AND     AH,0F7H         ; TURN OFF VIDEO ENABLE
0E65  32 C0                     XOR     AL,AL           ; SET UP TO SELECT VGA REG 0
0E67  EE                        OUT     DX,AL           ; SELECT IT
0E68  86 E0                     XCHG    AH,AL           ; AH IS VGA REG COUNTER
0E6A  EE                        OUT     DX,AL           ; SET MODE
0E6B  B0 04                     MOV     AL,4            ; SET UP TO SELECT VGA REG 4
0E6D  EE                        OUT     DX,AL           ; SELECT IT
0E6E  32 C0                     XOR     AL,AL           ;
0E70  EE                        OUT     DX,AL           ; REMOVE RESET FROM VGA
                        ; NOW OKAY TO ACCESS RAM AGAIN
0E71  B0 80                     MOV     AL,80H          ; ENABLE NMI AGAIN
0E73  E6 A0                     OUT     NMI_PORT,AL     ;
0E75  E8 E675 R                 CALL    MODE_ALIVE      ; KEEP MEMORY DATA VALID
0E78  FB                        STI                     ; ENABLE INTERRUPTS
0E79  EB 07                     JMP     SHORT C14
0E7B  8A C4             C13:    MOV     AL,AH           ; GET VGA REG NUMBER
0E7D  EE                        OUT     DX,AL           ; SELECT REG
0E7E  2E: 8A 07                 MOV     AL,CS:[BX]      ; GET TABLE VALUE
0E81  EE                        OUT     DX,AL           ; PUT IN VGA REG
0E82  43                C14:    INC     BX              ; NEXT IN TABLE
0E83  FE C4                     INC     AH              ; NEXT REG
0E85  E2 F4                     LOOP    C13             ; DO ENTIRE ENTRY
                        ;---- SET UP CRT AND CPU PAGE REGS ACCORDING TO MODE & MEMORY SIZE
0E87  BA 03DF                   MOV     DX,PAGREG       ; SET IO ADDRESS OF PAGREG
0E8A  A0 008A R                 MOV     AL,PAGDAT       ; GET LAST DATA OUTPUT
0E8D  24 C0                     AND     AL,0C0H         ; CLEAR REG BITS
0E8F  B3 36                     MOV     BL,36H          ; SET UP FOR GRAPHICS MODE WITH 32K
                                                        ; REGEN
0E91  A8 80                     TEST    AL,80H          ; IN THIS MODE?
0E93  75 0C                     JNZ     C15             ; YES, JUMP
0E95  B3 3F                     MOV     BL,3FH          ; SET UP FOR 16K REGEN AND 128K
                                                        ; MEMORY
0E97  81 3E 0015 R 0080         CMP     TRUE_MEM,128    ; DO WE HAVE 128K?
0E9D  73 02                     JNC     C15             ; YES, JUMP
0E9F  B3 1B                     MOV     BL,1BH          ; SET UP FOR 16K REGEN AND 64K
                                                        ; MEMORY
```

```
0EA1  0A C3                     C15:    OR      AL,BL           ; COMBINE MODE BITS AND REG VALUES
0EA3  EE                                OUT     DX,AL           ; SET PORT
0EA4  A2 008A R                         MOV     PAGDAT,AL       ; SAVE COPY IN RAM
0EA7  8B C6                             MOV     AX,SI           ; PUT MODE SET & PALETTE IN RAM
0EA9  8B 26 0065 R                      MOV     CRT_MODE_SET,AH
0EAD  A2 0066 R                         MOV     CRT_PALLETTE,AL
0EB0  E4 61                             IN      AL,PORT_B       ; GET CURRENT VALUE OF 8255 PORT B
0EB2  24 FB                             AND     AL,0FBH         ; SET UP GRAPHICS MODE
0EB4  F6 C4 02                          TEST    AH,2            ; JUST SET ALPHA MODE IN VGA?
0EB7  75 02                             JNZ     C16             ; YES, JUMP
0EB9  0C 04                             OR      AL,4            ; SET UP ALPHA MODE
0EBB  E6 61                     C16:    OUT     PORT_B,AL       ; STUFF BACK IN 8255
                                ;------- SET UP 6845
0EBD  1E                                PUSH    DS              ; SAVE DATA SEGMENT VALUE
0EBE  33 C0                             XOR     AX,AX           ; SET UP FOR ABSO SEGMENT
0EC0  8E D8                             MOV     DS,AX           ; ESTABLISH VECTOR TABLE ADDRESSING
                                        ASSUME  DS:ABS0
0EC2  C5 1E 0074 R                      LDS     BX,PARM_PTR     ; GET POINTER TO VIDEO PARMS
                                        ASSUME  DS:CODE
0EC6  8B C7                             MOV     AX,DI           ; GET CURRENT MODE IN AX
0EC8  B9 0010 90                        MOV     CX,M0040        ; LENGTH OF EACH ROW OF TABLE
0ECC  80 FC 02                          CMP     AH,2            ; DETERMINE WHICH TO USE
0ECF  72 10                             JC      C17             ; MODE IS 0 OR 1
0ED1  03 D9                             ADD     BX,CX           ; MOVE TO NEXT ROW OF INIT TABLE
0ED3  80 FC 04                          CMP     AH,4
0ED6  72 09                             JC      C17             ; MODE IS 2 OR 3
0ED8  03 D9                             ADD     BX,CX           ; MOVE TO GRAPHICS ROW OF
                                                                ; INIT_TABLE
0EDA  80 FC 09                          CMP     AH,9
0EDD  72 02                             JC      C17             ; MODE IS 4, 5, 6, 8, OR 9
0EDF  03 D9                             ADD     BX,CX           ; MOVE TO NEXT GRAPHICS ROW OF
                                                                ; INIT_TABLE
                                ;----- BX POINTS TO CORRECT ROW OF INITIALIZATION TABLE
0EE1  50                        C17:    PUSH    AX              ; SAVE MODE IN AH
0EE2  8A 47 02                          MOV     AL,DS:[BX+2]    ; GET HORZ. SYNC POSITION
0EE5  8B 7F 0A                          MOV     DI,WORD PTR DS:[BX+10]  ; GET CURSOR TYPE
0EE8  1E                                PUSH    DS
0EE9  E8 138B R                         CALL    DDS
                                        ASSUME  DS:DATA
0EEC  A2 0089 R                         MOV     HORZ_POS,AL     ; SAVE HORZ. SYNC POSITION VARIABLE
0EEF  89 3E 0060 R                      MOV     CURSOR_MODE,DI  ; SAVE CURSOR MODE
0EF3  50                                PUSH    AX
0EF4  A0 0086 R                         MOV     AL,VAR_DELAY    ; SET DEFAULT OFFSET
0EF7  24 0F                             AND     AL,0FH
0EF9  A2 0086 R                         MOV     VAR_DELAY,AL
0EFC  58                                POP     AX
                                        ASSUME  DS:CODE
0EFD  1F                                POP     DS
0EFE  32 E4                             XOR     AH,AH           ; AH WILL SERVE AS REGISTER NUMBER
                                                                ; DURING LOOP
0F00  BA 03D4                           MOV     DX,03D4H        ; POINT TO 6845
                                ;LOOP THROUGH TABLE, OUTPUTTING REG ADDRESS, THEN VALUE FROM TABLE
0F03  8A C4                     C18:    MOV     AL,AH           ; GET 6845 REGISTER NUMBER
0F05  EE                                OUT     DX,AL
0F06  42                                INC     DX              ; POINT TO DATA PORT
0F07  FE C4                             INC     AH              ; NEXT REGISTER VALUE
0F09  8A 07                             MOV     AL,[BX]         ; GET TABLE VALUE
0F0B  EE                                OUT     DX,AL           ; OUT TO CHIP
0F0C  43                                INC     BX              ; NEXT IN TABLE
0F0D  4A                                DEC     DX              ; BACK TO POINTER REGISTER
0F0E  E2 F3                             LOOP    C18             ; DO THE WHOLE TABLE
0F10  58                                POP     AX              ; GET MODE BACK
0F11  1F                                POP     DS              ; RECOVER SEGMENT VALUE
                                        ASSUME  DS:DATA
                                ;------ FILL REGEN AREA WITH BLANK
0F12  33 FF                             XOR     DI,DI           ; SET UP POINTER FOR REGEN
0F14  89 3E 004E R                      MOV     CRT_START,DI    ; START ADDRESS SAVED IN GLOBAL
0F18  C6 06 0062 R 00                   MOV     ACTIVE_PAGE,0   ; SET PAGE VALUE
0F1D  5A                                POP     DX              ; GET ORIGINAL INPUT BACK
0F1E  80 E2 80                          AND     DL,80H          ; NO CLEAR OF REGEN ?
0F21  75 1C                             JNZ     C21             ; SKIP CLEARING REGEN
0F23  BA B800                           MOV     DX,0B800H       ; SET UP SEGMENT FOR 16K REGEN AREA
0F26  B9 2000                           MOV     CX,8192         ; NUMBER OF WORDS TO CLEAR
0F29  3C 09                             CMP     AL,09H          ; REQUIRE 32K BYTE REGEN ?
0F2B  72 05                             JC      C19             ; NO, JUMP
0F2D  D1 E1                             SHL     CX,1            ; SET 16K WORDS TO CLEAR
0F2F  BA 1800                           MOV     DX,1800H        ; SET UP SEGMENT FOR 32K REGEN AREA
0F32  8E C2                     C19:    MOV     ES,DX           ; SET REGEN SEGMENT
0F34  3C 04                             CMP     AL,4            ; TEST FOR GRAPHICS
0F36  B8 0F20                           MOV     AX,' '+15*256   ; FILL CHAR FOR ALPHA
0F39  72 02                             JC      C20             ; NO_GRAPHICS_INIT
0F3B  33 C0                             XOR     AX,AX           ; FILL FOR GRAPHICS MODE
0F3D  F3/ AB                    C20:    REP     STOSW           ; FILL THE REGEN BUFFER WITH BLANKS
                                ;----- ENABLE VIDEO
0F3F  BA 03DA                   C21:    MOV     DX,VGA_CTL      ; SET PORT ADDRESS OF VGA
0F42  32 C0                             XOR     AL,AL           ;
0F44  EE                                OUT     DX,AL           ; SELECT VGA REG 0
0F45  A0 0065 R                         MOV     AL,CRT_MODE_SET ; GET MODE SET VALUE
0F48  EE                                OUT     DX,AL           ; SET MODE
                                ;----- DETERMINE NUMBER OF COLUMNS, BOTH FOR ENTIRE DISPLAY
                                ;----- AND THE NUMBER TO BE USED FOR TTY INTERFACE
0F49  32 FF                             XOR     BH,BH
0F4B  8A 1E 0049 R                      MOV     BL,CRT_MODE
0F4F  2E: 8A 87 005E R                  MOV     AL,CS:[BX + OFFSET M0060]
0F54  32 E4                             XOR     AH,AH
0F56  A3 004A R                         MOV     CRT_COLS,AX     ; NUMBER OF COLUMNS IN THIS SCREEN
```

```
                        ;----- SET CURSOR POSITIONS
0F59  D1 E3                     SHL     BX,1                    ; WORD OFFSET INTO CLEAR LENGTH
                                                                ; TABLE
0F5B  2E: 8B 8F 0048 R          MOV     CX,CS:[BX + OFFSET M0050]  ; LENGTH TO CLEAR
0F60  89 0E 004C R              MOV     CRT_LEN,CX              ; SAVE LENGTH OF CRT
0F64  B9 0008                   MOV     CX,8                    ; CLEAR ALL CURSOR POSITIONS
0F67  BF 0050 R                 MOV     DI,OFFSET CURSOR_POSN
0F6A  1E                        PUSH    DS                      ; ESTABLISH SEGMENT
0F6B  07                        POP     ES                      ;   ADDRESSING
0F6C  33 C0                     XOR     AX,AX
0F6E  F3/ AB                    REP     STOSW                   ; FILL WITH ZEROES
                        ;----- NORMAL RETURN FROM ALL VIDEO RETURNS
0F70                    VIDEO_RETURN:
0F70  5F                        POP     DI
0F71  5E                        POP     SI
0F72  5B                        POP     BX
0F73  59                C22:    POP     CX
0F74  5A                        POP     DX
0F75  1F                        POP     DS
0F76  07                        POP     ES                      ; RECOVER SEGMENTS
0F77  CF                        IRET                            ; ALL DONE
0F78                    SET_MODE        ENDP
                        ;--------------------------------------------------------
                        ;
                        ;
                        ; KBDNMI - KEYBOARD NMI INTERRUPT ROUTINE
                        ;
                        ;       THIS ROUTINE OBTAINS CONTROL UPON AN NMI INTERRUPT, WHICH
                        ;       OCCURS UPON A KEYSTROKE FROM THE KEYBOARD.
                        ;
                        ;       THIS ROUTINE WILL DE-SERIALIZE THE BIT STREAM IN ORDER TO
                        ;       GET THE KEYBOARD SCAN CODE ENTERED.  IT THEN ISSUES INT 41
                        ;       PASSING THE SCAN CODE IN AL TO THE KEY PROCESSOR. UPON RETURN
                        ;       IT RE-ENABLES NMI AND RETURNS TO SYSTEM (IRET).
                        ;
                        ;--------------------------------------------------------
                                        ASSUME CS:CODE,DS:DATA
0F78                    KBDNMI  PROC    FAR
                        ;-----------DISABLE INTERRUPTS
0F78  FA                        CLI
                        ;-----------SAVE REGS & DISABLE NMI
0F79  56                        PUSH    SI
0F7A  57                        PUSH    DI
0F7B  50                        PUSH    AX                      ; SAVE REGS
0F7C  53                        PUSH    BX
0F7D  51                        PUSH    CX
0F7E  52                        PUSH    DX
0F7F  1E                        PUSH    DS
0F80  06                        PUSH    ES
                        ;-----------INIT COUNTERS
0F81  BE 0008                   MOV     SI,8                    ; SET UP # OF DATA BITS
0F84  32 DB                     XOR     BL,BL                   ; INIT. PARITY COUNTER
                        ;-----------SAMPLE 5 TIMES TO  VALIDATE START BIT
0F86  32 E4                     XOR     AH,AH
0F88  B9 0005                   MOV     CX,5                    ; SET COUNTER
0F8B  E4 62             I1:     IN      AL,PORT_C               ; GET SAMPLE
0F8D  A8 40                     TEST    AL,40H                  ; TEST IF 1
0F8F  74 02                     JZ      I2                      ; JMP IF 0
0F91  FE C4                     INC     AH                      ; KEEP COUNT OF 1'S
0F93  E2 F6             I2:     LOOP    I1                      ; KEEP SAMPLING
0F95  80 FC 03                  CMP     AH,3                    ; VALID START BIT ?
0F98  73 03                     JNB     I25                     ; JUMP IF OK
0F9A  EB 5D 90                  JMP     I8                      ; INVALID (SYNC ERROR) NO AUDIO
                                                                ; OUTPUT
                        ;-----------VALID START BIT, LOOK FOR TRAILING EDGE
0F9D  B9 0032           I25:    MOV     CX,50                   ; SET UP WATCHDOG TIMEOUT
0FA0  E4 62             I3:     IN      AL,PORT_C               ; GET SAMPLE
0FA2  A8 40                     TEST    AL,40H                  ; TEST IF 0
0FA4  74 05                     JZ      I5                      ; JMP IF TRAILING EDGE FOUND
0FA6  E2 F8                     LOOP    I3                      ; KEEP LOOKING FOR TRAILING EDGE
0FA8  EB 4F 90                  JMP     I8                      ; SYNC ERROR (STUCK ON 1'S)
                        ;-----------READ CLOCK TO SET START OF BIT TIME
0FAB  B0 40             I5:     MOV     AL,40H                  ; READ CLOCK
0FAD  E6 43                     OUT     TIM_CTL,AL              ; *
0FAF  90                        NOP                             ; *
0FB0  90                        NOP                             ; *
0FB1  E4 41                     IN      AL,TIMER+1              ; *
0FB3  8A E0                     MOV     AH,AL                   ; *
0FB5  E4 41                     IN      AL,TIMER+1              ; *
0FB7  86 E0                     XCHG    AH,AL                   ; *
0FB9  8B F8                     MOV     DI,AX                   ; SAVE CLOCK TIME IN DI
                        ;-----------VERIFY VALID TRANSITION
0FBB  B9 0004                   MOV     CX,4                    ; SET COUNTER
0FBE  E4 62             I6:     IN      AL,PORT_C               ; GET SAMPLE
0FC0  A8 40                     TEST    AL,40H                  ; TEST IF 0
0FC2  75 35                     JNZ     I8                      ; JMP IF INVALID TRANSITION (SYNC)
0FC4  E2 F8                     LOOP    I6                      ; KEEP LOOKING FOR VALID TRANSITION
                        ;-----------SET UP DISTANCE TO MIDDLE OF 1ST DATA BIT
0FC6  BA 0220                   MOV     DX,544                  ; 310 USEC AWAY (.838 US / CT)
                        ;-------START LOOKING FOR TIME TO READ DATA BITS AND ASSEMBLE BYTE
0FC9  E8 1031 R         I7:     CALL    I30
0FCC  BA 020E                   MOV     DX,526                  ; SET NEW DISTANCE TO NEXT HALF BIT
0FCF  50                        PUSH    AX                      ; SAVE 1ST HALF BIT
0FD0  E8 1031 R                 CALL    I30
0FD3  8A C8                     MOV     CL,AL                   ; PUT 2ND HALF BIT IN CL
0FD5  58                        POP     AX                      ; RESTORE 1ST HALF BIT
0FD6  3A C8                     CMP     CL,AL                   ; ARE THEY OPPOSITES ?
0FD8  74 2A                     JE      I9                      ; NO, PHASE ERROR
```

```
                        ;-----------VALID DATA BIT, PLACE IN SCAN BYTE
OFDA  D0 EF                     SHR     BH,1                    ; SHIFT PREVIOUS BITS
OFDC  0A F8                     OR      BH,AL                   ; OR IN NEW DATA BIT
OFDE  4E                        DEC     SI                      ; DECREMENT DATA BIT COUNTER
OFDF  75 E8                     JNZ     I7                      ; CONTINUE FOR MORE DATA BITS
                        ;-----------WAIT FOR TIME TO SAMPLE PARITY BIT
OFE1  E8 1031 R                 CALL    I30
OFE4  50                        PUSH    AX                      ; SAVE 1ST HALF BIT
OFE5  E8 1031 R                 CALL    I30
OFE8  8A C8                     MOV     CL,AL                   ; PUT 2ND HALF BIT IN CL
OFEA  58                        POP     AX                      ; RESTORE 1ST HALF BIT
OFEB  3A C8                     CMP     CL,AL                   ; ARE THEY OPPOSITES ?
OFED  74 15                     JE      I9                      ; NO, PHASE ERROR
                        ;-----------VALID PARITY BIT, CHECK PARITY
OFEF  80 E3 01                  AND     BL,1                    ; CHECK IF ODD PARITY
OFF2  74 10                     JZ      I9                      ; JMP IF PARITY ERROR
                        ;-----------VALID CHARACTER, SEND TO CHARACTER PROCESSING
OFF4  FB                        STI                             ; ENABLE INTERRUPTS
OFF5  8A C7                     MOV     AL,BH                   ; PLACE SCAN CODE IN AL
OFF7  CD 48                     INT     48H                     ; CHARACTER PROCESSING
                        ;-----------RESTORE REGS AND RE-ENABEL NMI
OFF9  07                I8:     POP     ES                      ; RESTORE REGS
OFFA  1F                        POP     DS
OFFB  5A                        POP     DX
OFFC  59                        POP     CX
OFFD  5B                        POP     BX
OFFE  E4 A0                     IN      AL,0A0H                 ; ENABLE NMI
1000  58                        POP     AX
1001  5F                        POP     DI
1002  5E                        POP     SI
1003  CF                        IRET                            ; RETURN TO SYSTEM
                        ;-----------PARITY, SYNCH OR PHASE ERROR. OUTPUT MISSED KEY BEEP
1004  E8 138B R         I9:     CALL    DDS                     ; SETUP ADDRESSING
1007  83 FE 08                  CMP     SI,8                    ; ARE WE ON THE FIRST DATA BIT?
100A  74 ED                     JE      I8                      ; NO AUDIO FEEDBACK (MIGHT BE A
                                                                ; ..GLITCH)
100C  F6 06 0018 R 01           TEST    KB_FLAG_1,01H           ; CHECK IF TRANSMISSION ERRORS
                                                                ; ..ARE TO BE REPORTED
1011  75 18                     JNZ     I10                     ; 1=DO NOT BEEP, 0=BEEP
1013  BB 0080                  MOV     BX,080H                 ; DURATION OF ERROR BEEP
1016  B9 0048                  MOV     CX,048H                 ; FREQUENCY OF ERROR BEEP
1019  E8 E035 R                CALL    KB_NOISE                ; AUDIO FEEDBACK
101C  80 26 0017 R F0           AND     KB_FLAG,0F0H            ; CLEAR ALT,CLRL,LEFT AND RIGHT
                                                                ; SHIFTS
1021  80 26 0018 R 0F           AND     KB_FLAG_1,0FH           ; CLEAR POTENTIAL BREAK OF INS,CAPS
                                                                ; NUM AND SCROLL SHIFT
1026  80 26 0088 R 1F           AND     KB_FLAG_2,1FH           ; CLEAR FUNCTION STATES
102B  FE 06 0012 R       I10:    INC     KBD_ERR                 ; KEEP TRACK OF KEYBOARD ERRORS
102F  EB C8                     JMP     SHORT I8                ; RETURN FROM INTERRUPT
1031                     KBDNMI  ENDP
1031                     I30     PROC    NEAR
1031  B0 40              I31:    MOV     AL,40H                  ; READ CLOCK
1033  E6 43                     OUT     TIM_CTL,AL              ; *
1035  90                        NOP                             ; *
1036  90                        NOP                             ; *
1037  E4 41                     IN      AL,TIMER+1              ; *
1039  8A E0                     MOV     AH,AL                   ; *
103B  E4 41                     IN      AL,TIMER+1              ; *
103D  86 E0                     XCHG    AH,AL                   ; *
103F  8B CF                     MOV     CX,DI                   ; GET LAST CLOCK TIME
1041  2B C8                     SUB     CX,AX                   ; SUB CURRENT TIME
1043  3B CA                     CMP     CX,DX                   ; IS IT TIME TO SAMPLE ?
1045  72 EA                     JC      I31                     ; NO, KEEP LOOKING AT TIME
1047  2B CA                     SUB     CX,DX                   ; UPDATE # OF COUNTS OFF
1049  8B F8                     MOV     DI,AX                   ; SAVE CURRENT TIME AS LAST TIME
104B  03 F9                     ADD     DI,CX                   ; ADD DIFFERENCE FOR NEXT TIME
                        ;-----------START SAMPLING DATA BIT (5 SAMPLES)
104D  B9 0005                   MOV     CX,5                    ; SET COUNTER
                        ;--------------------------------------------------------
                        ;
                        ; SAMPLE LINE
                        ;
                        ;       PORT_C IS SAMPLED CX TIMES AND IF THER ARE 3 OR MORE 1"S
                        ;       THEN 80H IS RETURNED IN AL, ELSE 00H IS RETURNED IN AL.
                        ;       PARITY COUNTER IS MAINTAINED IN ES.
                        ;
                        ;--------------------------------------------------------
1050  32 E4                     XOR     AH,AH                   ; CLEAR COUNTER
1052  E4 62              I32:    IN      AL,PORT_C               ; GET SAMPLE
1054  A8 40                     TEST    AL,40H                  ; TEST IF 1
1056  74 02                     JZ      I33                     ; JMP IF 0
1058  FE C4                     INC     AH                      ; KEEP COUNT OF 1'S
105A  E2 F6              I33:    LOOP    I32                     ; KEEP SAMPLING
105C  80 FC 03                  CMP     AH,3                    ; VALID 1 ?
105F  72 05                     JB      I34                     ; JMP IF NOT VALID 1
1061  B0 80                     MOV     AL,080H                 ; RETURN 80H IN AL (1)
1063  FE C3                     INC     BL                      ; INCREMENT PARITY COUNTER
1065  C3                        RET                             ; RETURN TO CALLER
1066  32 C0              I34:    XOR     AL,AL                   ; RETURN 0 IN AL (0)
1068  C3                        RET                             ; RETURN TO CALLER
1069                     I30     ENDP
```

```
                                ;--------------------------------------------------------------------------
                                ;KEY62_INT
                                ;       THE PURPOSE OF THIS ROUTINE IS TO TRANSLATE SCAN CODES AND
                                ;       SCAN CODE COMBINATIONS FROM THE 62 KEY KEYBOARD TO THEIR
                                ;       EQUIVILENTS ON THE 83 KEY KEYBOARD.  THE SCAN CODE IS
                                ;       PASSED IN AL. EACH SCAN CODE PASSED EITHER TRIGGERS ONE OR
                                ;       MORE CALLS TO INTERRUPT 9 OR SETS FLAGS TO RETAIN KEYBOARD
                                ;       STATUS.  WHEN INTERRUPT 9 IS CALLED THE TRANSLATED SCAN
                                ;       CODES ARE PASSED TO IT IN AL.  THE INTENT OF THIS CODE WAS
                                ;       TO KEEP INTERRUPT 9 INTACT FROM ITS ORIGIN IN THE PC FAMILY
                                ;       THIS ROUTINE IS IN THE FRONT END OF INTERRUPT 9 AND
                                ;       TRANSFORMS A 62 KEY KEYBOARD TO LOOK AS IF IT WERE AN 83
                                ;       KEY VERSION.
                                ;       IT IS ASSUMED THAT THIS ROUTINE IS CALLED FROM THE NMI
                                ;       DESERIALIZATION ROUTINE AND THAT ALL REGISTERS WERE SAVED
                                ;       IN THE CALLING ROUTINE.  AS A CONSEQUENCE ALL REGISTERS ARE
                                ;       DESTROYED.
                                ;--------------------------------------------------------------------------
                                ;EQUATES
= 0080                          BREAK_BIT       EQU     80H
= 0054                          FN_KEY          EQU     54H
= 0055                          PHK             EQU     FN_KEY+1
= 0056                          EXT_SCAN        EQU     PHK+1   ; BASE CODE FOR SCAN CODES
                                                                ; EXTENDING BEYOND 83
= 00FF                          AND_MASK        EQU     OFFH    ; USED TO SELECTIVELY REMOVE BITS
= 001F                          CLEAR_FLAGS     EQU     AND_MASK - (FN_FLAG+FN_BREAK+FN_PENDING)
                                ;SCAN CODES
= 0030                          B_KEY           EQU     48
= 0010                          Q_KEY           EQU     16
= 0019                          P_KEY           EQU     25
= 0012                          E_KEY           EQU     18
= 001F                          S_KEY           EQU     31
= 0031                          N_KEY           EQU     49
= 0048                          UP_ARROW        EQU     72
= 0050                          DOWN_ARROW      EQU     80
= 004B                          LEFT_ARROW      EQU     75
= 004D                          RIGHT_ARROW     EQU     77
= 000C                          MINUS           EQU     12
= 000D                          EQUALS          EQU     13
= 000B                          NUM_0           EQU     11
                                ;NEW TRANSLATED SCAN CODES
                                ;-------------------------------------------------------
                                ;NOTE:
                                ;       BREAK, PAUSE, ECHO, AND PRT_SCREEN ARE USED AS OFFSETS
                                ;       INTO THE TABLE 'SCAN'. OFFSET = TABLE POSITION + 1.
                                ;-------------------------------------------------------
= 0001                          ECHO            EQU     01
= 0002                          BREAK           EQU     02
= 0003                          PAUSE           EQU     03
= 0004                          PRT_SCREEN      EQU     04
= 0046                          SCROLL_LOCK     EQU     70
= 0045                          NUM_LOCK        EQU     69
= 0047                          HOME            EQU     71
= 004F                          END_KEY         EQU     79
= 0049                          PAGE_UP         EQU     73
= 0051                          PAGE_DOWN       EQU     81
= 004A                          KEYPAD_MINUS    EQU     74
= 004E                          KEYPAD_PLUS     EQU     78
                                        ASSUME CS:CODE,DS:DATA
                                ;----TABLE OF VALID SCAN CODES
1069                            KBO     LABEL BYTE
1069  30 10 12 19 1F 31                 DB B_KEY, Q_KEY, E_KEY, P_KEY, S_KEY, N_KEY
106F  48 50 4B 4D 0C                   DB UP_ARROW, DOWN_ARROW, LEFT_ARROW, RIGHT_ARROW, MINUS
1074  0D                               DB EQUALS
= 000C                          KBOLEN          EQU $ - KBO
                                ;----TABLE OF NEW SCAN CODES
1075                            KB1     LABEL BYTE
1075  02 03 01 04 46 45                 DB BREAK, PAUSE, ECHO, PRT_SCREEN, SCROLL_LOCK, NUM_LOCK
107B  47 4F 49 51 4A 4E                 DB HOME,END_KEY,PAGE_UP,PAGE_DOWN,KEYPAD_MINUS,KEYPAD_PLUS
                                ;-----------------------------------------------
                                ;NOTE:  THERE IS A ONE TO ONE CORRESPONDENCE BETWEEN
                                ;       THE SIZE OF KBO AND KB1.
                                ;-----------------------------------------------
                                ;TABLE OF NUMERIC KEYPAD SCAN CODES
                                ;       THESE SCAN CODES WERE NUMERIC KEYPAD CODES ON
                                ;       THE 83 KEY KEYBOARD.
                                ;-------------------------------------------------
1081                            NUM_CODES LABEL BYTE
1081  4F 50 51 4B 4C 4D                 DB 79,80,81,75,76,77,71,72,73,82
      47 48 49 52

                                ;-------------------------------------------------
                                ;TABLE OF SIMULATED KEYSTROKES
                                ;       THIS TABLE REPRESENTS A 4*2 ARRAY.  EACH ROW
                                ;       CONSISTS OF A SEQUENCE OF SCAN CODES WHICH
                                ;       WOULD HAVE BEEN GENERATED ON AN 83 KEY KEYBOARD
                                ;       TO CAUSE THE FOLLOWING FUNCTIONS:
                                ;               ROW 1=ECHO CRT OUTPUT TO THE PRINTER
                                ;               ROW 2=BREAK
                                ;       THE TABLE HAS BOTH MAKE AND BREAK SCAN CODES.
                                ;-------------------------------------------------------
1088                            SCAN    LABEL BYTE
1088  1D 37 B7 9D                       DB 29,55,183,157        ; CTRL + PRTSC
108F  1D 46 C6 9D                       DB 29,70,198,157        ; CTRL + SCROLL-LOCK
```

```
                                ;--------------------------------------------------
                                ;TABLE OF VALID ALT SHIFT SCAN CODES
                                ;       THIS TABLE CONTAINS SCAN CODES FOR KEYS ON THE
                                ;       62 KEY KEYBOARD.  THESE CODES ARE USED IN
                                ;       COMBINATION WITH THE ALT KEY TO PRODUCE SCAN CODES
                                ;       FOR KEYS NOT FOUND ON THE 62 KEY KEYBOARD.
                                ;--------------------------------------------------
1093                            ALT_TABLE       LABEL   BYTE
1093  35 28 34 1A 1B                    DB 53,40,52,26,27
= 0005                          ALT_LEN  EQU $ - ALT_TABLE
                                ;--------------------------------------------------
                                ;TABLE OF TRANSLATED SCAN CODES WITH ALT SHIFT
                                ;       THIS TABLE CONTAINS THE SCAN CODES FOR THE
                                ;       KEYS WHICH ARE NOT ON THE 62 KEY KEYBOARD AND
                                ;       WILL BE TRANSLATED WITH ALT SHIFT.  THERE IS A
                                ;       ONE TO ONE CORRESPONDENCE BETWEEN THE SIZES
                                ;       OF ALT_TABLE AND NEW_ALT.
                                ;       THE FOLLOWING TRANSLATIONS ARE MADE:
                                ;               ALT+ / = \
                                ;               ALT+ ' = `
                                ;               ALT+ [ = ;
                                ;               ALT+ ] = ~
                                ;               ALT+ . = *
                                ;--------------------------------------------------
1098                            NEW_ALT LABEL BYTE
1098  28 29 37 2B 29                    DB 43,41,55,43,41
                                ;--------------------------------------------------
                                ;EXTAB
                                ;       TABLE OF SCAN CODES FOR MAPPING EXTENDED SET
                                ;       OF SCAN CODES (SCAN CODES > 85).  THIS TABLE
                                ;       ALLOWS OTHER DEVICES TO USE THE KEYBOARD INTERFACE.
                                ;       IF THE DEVICE GENERATES A SCAN CODE > 85 THIS TABLE
                                ;       CAN BE USED TO MAP THE DEVICE TO THE KEYBOARD.  THE
                                ;       DEVICE ALSO HAS THE OPTION OF HAVING A UNIQUE SCAN
                                ;       CODE PUT IN THE KEYBOARD BUFFER (INSTEAD OF MAPPING
                                ;       TO THE KEYBOARD). THE EXTENDED SCAN CODE PUT IN THE
                                ;       BUFFER WILL BE CONTINUOUS BEGINNING AT 150.  A ZERO
                                ;       WILL BE USED IN PLACE OF AN ASCII CODE. (E.G.  A
                                ;       DEVICE GENERATING SCAN CODE 86 AND NOT MAPPING 86
                                ;       TO THE KEYBOARD WILL HAVE A [150,0] PUT IN THE
                                ;       KEYBOARD BUFFER)
                                ;       TABLE FORMAT:
                                ;       THE FIRST BYTE IS A LENGTH INDICATING THE NUMBER
                                ;       OF SCAN CODES MAPPED TO THE KEYBOARD.  THE REMAINING
                                ;       ENTRIES ARE WORDS. THE FIRST BYTE (LOW BYTE) IS A
                                ;       SCAN CODE AND THE SECOND BYTE (HIGH BYTE) IS ZERO.
                                ;       A DEVICE GENERATING N SCAN CODES IS ASSUMED TO GENERATE THE
                                ;       FOLLOWING STREAM 86,87,88,...,86+(N-1).  THE SCAN CODE BYTES
                                ;       IN THE TABLE CORRESPOND TO THIS SET WITH THE FIRST DATA
                                ;       BYTE MATCHING 86, THE SECOND MATCHING 87 ETC.
                                ;   NOTES:
                                ;       (1) IF A DEVICE GENERATES A BREAK CODE, NOTHING IS
                                ;           PUT IN THE BUFFER.
                                ;       (2) A LENGTH OF 0 INDICATES THAT ZERO SCAN CODES HAVE BEEN
                                ;           MAPPED TO THE KEYBOARD AND ALL EXTENDED SCAN CODES WILL
                                ;           BE USED.
                                ;       (3) A DEVICE CAN MAP SOME OF ITS SCAN CODES TO THE KEYBOARD
                                ;           AND HAVE SOME ITS SCAN CODES IN THE EXTENDED SET.
                                ;--------------------------------------------------
109D                            EXTAB   LABEL BYTE
109D  14                                DB      20                      ; LENGTH OF TABLE
109E  0048 0049 004D 0051               DW      72,73,77,81,80,79,75,71,57,28
      0050 004F 004B 0047
      0039 001C
10B2  0011 0012 001F 002D               DW      17,18,31,45,44,43,30,16,15,1
      002C 002B 001E 0010
      000F 0001
10C6                            KEY62_INT PROC FAR
10C6  FB                                 STI
10C7  FC                                 CLD                            ; FORWARD DIRECTION
10C8  E8 138B R                          CALL    DDS                    ; SET UP ADDRESSING
10CB  8A E0                              MOV     AH,AL                  ; SAVE SCAN CODE
10CD  E8 131E R                          CALL    TPM                    ; ADJUST OUTPUT FOR USER
                                                                        ; MODIFICATION
10D0  73 01                              JNC     KBX0                   ; JUMP IF OK TO CONTINUE
10D2  CF                                 IRET                           ; RETURN FROM INTERRUPT.
                                ;----EXTENDED SCAN CODE CHECK
10D3  3C FF                     KBX0:    CMP     AL,0FFH                ; IS THIS AN OVERRUN CHAR?
10D5  74 6C                              JE      KBO_1                  ; PASS IT TO INTERRUPT 9
10D7  24 7F                              AND     AL,AND_MASK-BREAK_BIT  ; TURN OFF BREAK BIT
10D9  3C 56                              CMP     AL,EXT_SCAN            ; IS THIS A SCAN CODE > 83
10DB  7C 5F                              JL      KBX4                   ; REPLACE BREAK BIT
                                ;----SCAN CODE IS IN EXTENDED SET
10DD  1E                                 PUSH    DS
10DE  33 F6                              XOR     SI,SI
10E0  8E DE                              MOV     DS,SI
                                         ASSUME  DS:ABS0
10E2  C4 3E 0124 R                       LES     DI,DWORD PTR EXST ; GET THE POINTER TO THE EXTENDED
                                                                   ; SET
10E6  26: 8A 0D                          MOV     CL,BYTE PTR ES:[DI] ; GET LENGTH BYTE
10E9  1F                                 POP     DS
                                         ASSUME  DS:DATA
                                ;----DOES SCAN CODE GET MAPPED TO KEYBOARD OR TO NEW EXTENDED SCAN
                                ;     CODES?
10EA  2C 56                              SUB     AL,EXT_SCAN            ; CONVERT TO BASE OF NEW SET
10EC  FE C9                              DEC     CL                     ; LENGTH - 1
10EE  3A C1                              CMP     AL,CL                  ; IS CODE IN TABLE?
10F0  7F 10                              JG      KBX1                   ; JUMP IF SCAN CODE IS NOT IN TABLE
```

```
                                ;----GET SCAN CODE FROM TABLE
10F2  47                                INC     DI                      ; POINT DI PAST LENGTH BYTE
10F3  8B D8                             MOV     BX,AX
10F5  32 FF                             XOR     BH,BH                   ; PREPARE FOR ADDING TO 16 BIT
                                                                        ; REGISTER
10F7  D1 E3                             SHL     BX,1
10F9  03 FB                             ADD     DI,BX                   ; OFFSET TO CORRECT TABLE ENTRY
10FB  26: 8A 05                         MOV     AL,BYTE PTR ES:[DI] ; TRANSLATED SCAN CODE IN AL
10FE  3C 56                             CMP     AL,EXT_SCAN             ; IS CODE IN KEYBOARD SET?
1100  7C 3A                            JL      KBX4                    ; IN KEYBOARD SET, CHECK FOR BREAK
                                ;----SCAN CODE GETS MAPPED TO EXTENDED SCAN CODES
1102  F6 C4 80                    KBX1:   TEST    AH,BREAK_BIT            ; IS THIS A BREAK CODE?
1105  74 01                             JZ      KBX2                    ; MAKE CODE, PUT IN BUFFER
1107  CF                                IRET                            ; BREAK CODE, RETURN FROM INTERRUPT
1108  80 C4 40                  KBX2:   ADD     AH,64                   ; EXTENDED SET CODES BEGIN AT 150
110B  32 C0                             XOR     AL,AL                   ; ZERO OUT ASCII VALUE (NUL)
110D  8B 1E 001C R                      MOV     BX,BUFFER_TAIL          ; GET TAIL POINTER
1111  8B F3                             MOV     SI,BX                   ; SAVE POINTER TO TAIL
1113  E8 144F R                         CALL    K4                      ; INCREMENT TAIL VALUE
1116  3B 1E 001A R                      CMP     BX,BUFFER_HEAD          ; IS BUFFER FULL?
111A  75 19                             JNE     KBX3                    ; PUT CONTENTS OF AX IN BUFFER
                                ;----BUFFER IS FULL, BEEP AND CLEAR FLAGS
111C  BB 0080                           MOV     BX,80H                  ; FREQUENCY OF BEEP
111F  B9 0048                           MOV     CX,48H                  ; DURATION OF BEEP
1122  E8 E035 R                         CALL    KB_NOISE                ; BUFFER FULL BEEP
1125  80 26 0017 R F0                   AND     KB_FLAG,0F0H            ; CLEAR ALT, CTRL, LEFT AND RIGHT
                                                                        ; SHIFTS
112A  80 26 0018 R 0F                   AND     KB_FLAG_1,0FH           ; CLEAR MAKE OF INS,CAPS_LOCK,NUM
                                                                        ; AND SCROLL
112F  80 26 0088 R 1F                   AND     KB_FLAG_2,1FH           ; CLEAR FUNCTION STATES
1134  CF                                IRET                            ; DONE WITH INTERRUPT
1135  89 04                     KBX3:   MOV     [SI],AX                 ; PUT CONTENTS OF AX IN BUFFER
1137  89 1E 001C R                      MOV     BUFFER_TAIL,BX          ; ADVANCE BUFFER TAIL
113B  CF                                IRET                            ; RETURN FROM INTERRUPT
113C  80 E4 80                  KBX4:   AND     AH,BREAK_BIT            ; MASK BREAK BIT ON ORIGINAL SCAN
113F  0A C4                             OR      AL,AH                   ; UPDATE NEW SCAN CODE
1141  8A E0                             MOV     AH,AL                   ; SAVE AL IN AH AGAIN
                                ;----83 KEY KEYBOARD FUNCTIONS SHIFT+PRTSC AND CTRL+NUMLOCK
1143  3C 45                     KB0_1:  CMP     AL,NUM_KEY              ; IS THIS A NUMLOCK?
1145  75 14                             JNE     KB0_3                   ; CHECK FOR PRTSC
1147  F6 06 0017 R 04                   TEST    KB_FLAG,CTL_SHIFT ; IS CTRL KEY BEING HELD DOWN?
114C  74 0A                             JZ      KB0_2                   ; NUMLOCK WITHOUT CTRL, CONTINUE
114E  F6 06 0017 R 08                   TEST    KB_FLAG,ALT_SHIFT ; IS ALT KEY HELD CONCURRENTLY?
1153  75 03                             JNZ     KB0_2                   ; PASS IT ON
1155  E9 12EB R                         JMP     KB16_1                  ; PUT KEYBOARD IN HOLD STATE
1158  E9 125C R                 KB0_2:  JMP     CONT_INT                ; CONTINUE WITH INTERRUPT 48H
                                ;----CHECK FOR PRTSC
115B  3C 37                     KB0_3:  CMP     AL,55                   ; IS THIS A PRTSC KEY?
115D  75 11                             JNZ     KB1_1                   ; NOT A PRTSC KEY
115F  F6 06 0017 R 03                   TEST    KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT ; EITHER SHIFT
                                                                        ; ACTIVE?
1164  74 F2                             JZ      KB0_2                   ; PROCESS SCAN IN INT9
1166  F6 06 0017 R 04                   TEST    KB_FLAG,CTL_SHIFT ; IS THE CTRL KEY PRESSED?
116B  75 EB                             JNZ     KB0_2                   ; NOT A VALID PRTSC (PC COMPATIBLE)
116D  E9 1301 R                         JMP     PRTSC                   ; HANDLE THE PRINT SCREEN FUNCTION
                                ;----ALTERNATE SHIFT TRANSLATIONS
1170  8A E0                     KB1_1:  MOV     AH, AL                  ; SAVE CHARACTER
1172  24 7F                             AND     AL, AND_MASK - BREAK_BIT ; MASK BREAK BIT
1174  F6 06 0017 R 08                   TEST    KB_FLAG,ALT_SHIFT ; IS THIS A POTENTIAL TRANSLATION
1179  74 39                             JZ      KB2
                                ;----TABLE LOOK UP
117B  0E                                PUSH    CS
117C  07                                POP     ES                      ; INITIALIZE SEGMENT FOR TABLE LOOK
                                                                        ; UP
117D  BF 1093 R                         MOV     DI,OFFSET ALT_TABLE
1180  B9 0005                           MOV     CX,ALT_LEN              ; GET READY FOR TABLE LOOK UP
1183  F2/ AE                           REPNE   SCASB                   ; SERACH TABLE
1185  75 2D                             JNE     KB2                     ; JUMP IF MATCH IS NOT FOUND
1187  B9 1094 R                         MOV     CX,OFFSET ALT_TABLE + 1
118A  2B F9                             SUB     DI,CX                   ; UPDATE DI TO INDEX SCAN CODE
118C  2E: 8A 85 1098 R                  MOV     AL,CS:NEW_ALT[DI] ; TRANSLATE SCAN CODE
                                ;----CHECK FOR BREAK CODE
1191  8A 1E 0017 R                      MOV     BL,KB_FLAG              ; SAVE KB_FLAG STATUS
1195  80 36 0017 R 08                   XOR     KB_FLAG,ALT_SHIFT ; MASK OFF ALT SHIFT
119A  F6 C4 80                          TEST    AH,BREAK_BIT            ; IS THIS A BREAK CHARACTER?
119D  74 02                             JZ      KB1_2                   ; JUMP IF SCAN IS A MAKE
119F  0C 80                             OR      AL,BREAK_BIT            ; SET BREAK BIT
                                ;----MAKE CODE, CHECK FOR SHIFT SEQUENCE
11A1  83 FF 03                  KB1_2:  CMP     DI,3                    ; IS THIS A SHIFT SEQUENCE
11A4  7C 05                             JL      KB1_3                   ; JUMP IF NOT SHIFT SEQUENCE
11A6  80 0E 0017 R 02                   OR      KB_FLAG,LEFT_SHIFT ; TURN ON SHIFT FLAG
11AB  E6 60                     KB1_3:  OUT     KBPORT,AL
11AD  CD 09                             INT     9H                      ; ISSUE INT TO PROCESS SCAN CODE
11AF  88 1E 0017 R                      MOV     KB_FLAG,BL              ; RESTORE ORIGINAL FLAG STATES
11B3  CF                                IRET
                                ;----FUNCTION KEY HANDLER
11B4  3C 54                     KB2:    CMP     AL, FN_KEY              ; CHECK FOR FUNCTION KEY
11B6  75 23                             JNZ     KB4                     ; JUMP IF NOT FUNCTION KEY
11B8  F6 C4 80                          TEST    AH, BREAK_BIT   ; IS THIS A FUNCTION BREAK
11BB  75 0B                             JNZ     KB3                     ; JUMP IF FUNCTION BREAK
11BD  80 26 0088 R 1F                   AND     KB_FLAG_2,CLEAR_FLAGS ; CLEAR ALL PREVIOUS
                                                                        ; FUNCTIONS
11C2  80 0E 0088 R A0                   OR      KB_FLAG_2, FN_FLAG + FN_PENDING
11C7  CF                                IRET                            ; RETURN FROM INTERRUPT
                                ;----FUNCTION BREAK
11C8  F6 06 0088 R 20           KB3:    TEST    KB_FLAG_2,FN_PENDING
11CD  75 06                             JNZ     KB3_1                   ; JUMP IF FUNCTION IS PENDING
11CF  80 26 0088 R 1F                   AND     KB_FLAG_2,CLEAR_FLAGS ; CLEAR ALL FLAGS
11D4  CF                                IRET
11D5  80 0E 0088 R 40           KB3_1:  OR      KB_FLAG_2,FN_BREAK ; SET BREAK FLAG
11DA  CF                        KB3_2:  IRET                            ; RETURN FROM INTERRUPT
```

```
                        ;----CHECK IF FUNCTION FLAG ALREADY SET
11DB  3C 55             KB4:    CMP     AL,PHK          ; IS THIS A PHANTOM KEY?
11DD  74 FB                     JZ      KB3_2           ; JUMP IF PHANTOM SEQUENCE
11DF  F6 06 0088 R 90   KB4_0:  TEST    KB_FLAG_2,FN_FLAG+FN_LOCK ; ARE WE IN FUNCTION
                                                        ; STATE?
11E4  75 21                     JNZ     KB5
                        ;----CHECK IF NUM_STATE IS ACTIVE
11E6  F6 06 0017 R 20           TEST    KB_FLAG,NUM_STATE
11EB  74 16                     JZ      KB4_1           ; JUMP IF NOT IN NUM_STATE
11ED  3C 0B                     CMP     AL,NUM_0        ; ARE WE IN NUMERIC KEYPAD REGION?
11EF  77 12                     JA      KB4_1           ; JUMP IF NOT IN KEYPAD
11F1  FE C8                     DEC     AL              ; CHECK LOWER BOUND OF RANGE
11F3  74 0E                     JZ      KB4_1           ; JUMP IF NOT IN RANGE (ESC KEY)
                        ;----TRANSLATE SCAN CODE TO NUMERIC KEYPAD
11F5  FE C8                     DEC     AL              ; AL IS OFFSET INTO TABLE
11F7  BB 1081 R                 MOV     BX,OFFSET NUM_CODES
11FA  2E: D7                    XLAT    CS:NUM_CODES    ; NEW SCAN CODE IS IN AL
11FC  80 E4 80                  AND     AH,BREAK_BIT    ; ISOLATE BREAK BIT ON ORIGINAL
                                                        ; SCAN CODE
11FF  0A C4                     OR      AL,AH           ; UPDATE KEYPAD SCAN CODE
1201  EB 59                     JMP     SHORT CONT_INT  ; CONTINUE WITH INTERRUPT
1203  8A C4            KB4_1:  MOV     AL,AH           ; GET BACK BREAK BIT IF SET
1205  EB 55                     JMP     SHORT CONT_INT
                        ;----CHECK FOR VALID FUNCTION KEY
1207  3C 0B             KB5:    CMP     AL, NUM_0       ; CHECK FOR RANGE OF INTEGERS
1209  77 2D                     JA      KB7             ; JUMP IF NOT IN RANGE
120B  FE C8                     DEC     AL              ; CHECK FOR ESC KEY (=1)
120D  75 25                     JNZ     KB6             ; NOT ESCAPE KEY, RANGE OF INTEGERS
                        ;----ESCAPE KEY, LOCK KEYBOARD IN FUNCTION LOCK
120F  F6 C4 80                  TEST    AH,BREAK_BIT    ; IS THIS A BREAK CODE?
1212  75 30                     JNZ     KB8             ; NO PROCESSING FOR ESCAPE BREAK
1214  F6 06 0088 R 80           TEST    KB_FLAG_2,FN_FLAG ; TOGGLES ONLY WHEN FN HELD
                                                        ; CONCURRENTLY
1219  74 29                     JZ      KB8             ; NOT HELD CONCURRENTLY
121B  F6 06 0088 R 40           TEST    KB_FLAG_2,FN_BREAK ; HAS THE FUNCTION KEY BEEN
                                                        ; RELEASED?
1220  75 22                     JNZ     KB8             ; CONTINUE IF RELEASED.  PROCESS AS
                                                        ; ESC
1222  F6 06 0017 R 03           TEST    KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT ; EITHER SHIFT?
1227  74 1B                     JZ      KB8             ; NOT HELD DOWN
1229  80 36 0088 R 10           XOR     KB_FLAG_2,FN_LOCK ; TOGGLE STATE
122E  80 26 0088 R 1F           AND     KB_FLAG_2,CLEAR_FLAGS ; TURN OFF OTHER STATES
1233  CF                        IRET                    ; RETURN FROM INTERUPT
                        ;----SCAN CODE IN RANGE 1 -> 0
1234  04 3A             KB6:    ADD     AL, 58          ; GENERATE CORRECT SCAN CODE
1236  EB 3E                     JMP     SHORT KB12      ; CLEAN-UP BEFORE RETURN TO KB_INT
                        ;----CHECK TABLE FOR OTHER VALID SCAN CODES
1238  0E                KB7:    PUSH    CS
1239  07                        POP     ES              ; ESTABLISH ADDRESS OF TABLE
123A  BF 1069 R                 MOV     DI, OFFSET KB0  ; BASE OF TABLE
123D  B9 000C                   MOV     CX, KBOLEN      ; LENGTH OF TABLE
1240  F2/ AE                    REPNE   SCASB           ; SEARCH TABLE FOR A MATCH
1242  74 1D                     JE      KB10            ; JUMP IF MATCH
                        ;----ILLEGAL CHARACTER
1244  F6 06 0088 R 40    KB8:    TEST    KB_FLAG_2,FN_BREAK ; HAS BREAK OCCURED?
1249  74 0F                     JZ      KB9             ; FUNCTION KEY HAS NOT BEEN
                                                        ; RELEASED
124B  F6 C4 80                  TEST    AH,BREAK_BIT    ; IS THIS A BREAK OF AN ILLEGAL
124E  75 0A                     JNZ     KB9             ; DON'T RESET FLAGS ON ILLEGAL
                                                        ; BREAK
1250  80 26 0088 R 1F   KB8S:   AND     KB_FLAG_2,CLEAR_FLAGS ; NORMAL STATE
1255  C6 06 0087 R 00           MOV     CUR_FUNC,0      ; RETRIEVE ORIGINAL SCAN CODE
                        ;----FUNCTION BREAK IS NOT SET
125A  8A C4             KB9:    MOV     AL,AH           ; RETRIEVE ORIGINAL SCAN CODE
125C                    CONT_INT:
125C  E6 60                     OUT     KBPORT,AL
125E  CD 09                     INT     9H              ; ISSUE KEYBOARD INTERRUPT
1260                    RET_INT:
1260  CF                        IRET
                        ;----BEFORE TRANSLATION CHECK FOR ALT+FN+N_KEY AS NUM LOCK
1261  3C 31             KB10:   CMP     AL,N_KEY        ; IS THIS A POTENTIAL NUMLOCK?
1263  75 07                     JNE     KB10_1          ; NOT A NUMKEY, TRANSLATE IT
1265  F6 06 0017 R 08           TEST    KB_FLAG,ALT_SHIFT ; ALT HELD DOWN ALSO?
126A  74 D8                     JZ      KB8             ; TREAT AS ILLEGAL COMBINATION
126C  B9 106A R         KB10_1: MOV     CX, OFFSET KB0 + 1 ; GET OFFSET TO TABLE
126F  2B F9                     SUB     DI, CX          ; UPDATE INDEX TO NEW SCAN CODE
                                                        ; TABLE
1271  2E: 8A 85 1075 R          MOV     AL, CS:KB1[DI]  ; MOV NEW SCAN CODE INTO REGISTER
                        ;----TRANSLATED CODE IN AL OR AN OFFSET TO THE TABLE "SCAN"
1276  F6 C4 80           KB12:   TEST    AH,BREAK_BIT    ; IS THIS A BREAK CHAR?
1279  74 35                     JZ      KB13            ; JUMP IF MAKE CODE
                        ;----CHECK FOR TOGGLE KEY
127B  3C 45                     CMP     AL,NUM_LOCK     ; IS THIS A NUM LOCK?
127D  74 04                     JZ      KB12_1          ; JUMP IF TOGGLE KEY
127F  3C 46                     CMP     AL,SCROLL_LOCK  ; IS THIS A SCROLL LOCK?
1281  75 08                     JNZ     KB12_2          ; JUMP IF NOT A TOGGLE KEY
1283  0C 80           KB12_1:  OR      AL,80H          ; TURN ON BREAK BIT
1285  E6 60                     OUT     KBPORT,AL
1287  CD 09                     INT     9H              ; TOGGLE STATE
1289  24 7F                     AND     AL,AND_MASK-BREAK_BIT ; TURN OFF BREAK BIT
128B  F6 06 0088 R 40   KB12_2: TEST    KB_FLAG_2,FN_BREAK ; HAS FUNCTION BREAK OCCURED?
1290  74 11                     JZ      KB12_3          ; JUMP IF BREAK HAS NOT OCCURED
1292  3A 06 0087 R              CMP     AL,CUR_FUNC     ; IS THIS A BREAK OF OLD VALID
                                                        ; FUNCTION
1296  75 C8                     JNE     RET_INT         ; ALLOW FURTHER CURRENT FUNCTIONS
1298  80 26 0088 R 1F           AND     KB_FLAG_2,CLEAR_FLAGS
129D                    KB12_20:
129D  C6 06 0087 R 00           MOV     CUR_FUNC,0      ; CLEAR CURRENT FUNCTION
12A2  CF                        IRET                    ; RETURN FROM INTERRUPT
```

```
12A3  3A 06 0087 R            KB12_3: CMP     AL,CUR_FUNC     ; IS THIS BREAK OF FIRST FUNCTION?
12A7  75 B7                           JNE     RET_INT         ; IGNORE
12A9  80 26 0088 R DF                 AND     KB_FLAG_2,AND_MASK-FN_PENDING ; TURN OFF PENDING
                                                              ; FUNCTION
12AE  EB ED                           JMP     KB12_20         ; CLEAR CURRENT FUNCTION AND RETURN
                              ;----VALID MAKE KEY HAS BEEN PRESSED
12B0  F6 06 0088 R 40         KB13:   TEST    KB_FLAG_2,FN_BREAK ; CHECK IF FUNCTION KEY HAS BEEN
                                                              ; PRESSED
12B5  74 0D                           JZ      KB14_1          ; JUMP IF NOT SET
                              ;----FUNCTION BREAK HAS ALREADY OCCURED
12B7  80 3E 0087 R 00                 CMP     CUR_FUNC,0      ; IS THIS A NEW FUNCTION?
12BC  74 06                           JZ      KB14_1          ; INITIALIZE NEW FUNCTION
12BE  38 06 0087 R                    CMP     CUR_FUNC,AL     ; IS THIS NON-CURRENT FUNCTION
12C2  75 8C                           JNZ     KB85            ;JUMP IF NO FUNCTION IS PENDING
                                                              ;..TO RETRIEVE ORIGINAL SCAN CODE
                              ;----CHECK FOR SCAN CODE GENERATION SEQUENCE
12C4  A2 0087 R               KB14_1: MOV     CUR_FUNC,AL     ; INITIALIZE CURRENT FN
12C7  3C 04                   KB16:   CMP     AL,PRT_SCREEN   ; IS THIS A SIMULATED SEQUENCE?
12C9  7F 91                           JG      CONT_INT        ; JUMP IF THIS IS A SIMPLE
                                                              ; TRANSLATION
12CB  74 34                           JZ      PRTSC           ; DO THE PRINT SCREEN FUNCTION
12CD  3C 03                           CMP     AL,PAUSE        ; IS THIS THE HOLD FUNCTION?
12CF  74 1A                           JZ      KB16_1          ; DO THE PAUSE FUNCTION
                              ;----BREAK OR ECHO
12D1  FE C8                           DEC     AL              ; POINT AT BASE
12D3  D0 E0                           SHL     AL,1
12D5  D0 E0                           SHL     AL,1            ; MULTIPLY BY 4
12D7  98                              CBW
12D8  2E: 8D 36 1088 R                LEA     SI,SCAN         ; ADDRESS SEQUENCE OF SIMULATED
                                                              ; KEYSTROKES
12DD  03 F0                           ADD     SI,AX           ; UPDATE TO POINT AT CORRECT SET
12DF  B9 0004                         MOV     CX,4            ; LOOP COUNTER
12E2                          GENERATE:
12E2  2E: AC                          LODS    SCAN            ; GET SCAN CODE FROM TABLE
12E4  E6 60                           OUT     KBPORT,AL
12E6  CD 09                           INT     9H              ; PROCESS IT
12E8  E2 F8                           LOOP    GENERATE        ; GET NEXT
12EA  CF                              IRET
                              ;----PUT KEYBOARD IN HOLD STATE
12EB  F6 06 0018 R 08         KB16_1: TEST    KB_FLAG_1,HOLD_STATE ; CANNOT GO IN HOLD STATE IF
                                                              ; ITS ACTIVE
12F0  75 0E                           JNZ     KB16_2          ; DONE WITH INTERRUPT
12F2  80 0E 0018 R 08                 OR      KB_FLAG_1,HOLD_STATE ; TURN ON HOLD FLAG
12F7  E4 A0                           IN      AL,NMI_PORT     ; RESET KEYBOARD LATCH
12F9  F6 06 0018 R 08         HOLD:   TEST    KB_FLAG_1,HOLD_STATE ; STILL IN HOLD STATE?
12FE  75 F9                           JNZ     HOLD            ; CONTINUE LOOPING UNTIL KEY IS
                                                              ; PRESSED
1300  CF                      KB16_2: IRET                    ; RETURN FROM INTERRUPT 48H
                              ;----PRINT SCREEN FUNCTION
1301  F6 06 0018 R 08        PRTSC:  TEST    KB_FLAG_1,HOLD_STATE ; IS HOLD STATE IN PROGRESS?
1306  74 06                           JZ      KB16_3          ; OK TO CONTINUE WITH PRTSC
1308  80 26 0018 R F7                 AND     KB_FLAG_1,OFFH-HOLD_STATE ; TURN OFF FLAG
130D  CF                              IRET
130E  83 C4 06                KB16_3: ADD     SP,3*2          ; GET RID OF CALL TO INTERRUPT 48H
1311  07                             POP     ES              ; POP REGISTERS THAT AREN'T
                                                              ; MODIFIED IN INT5
1312  1F                              POP     DS
1313  5A                              POP     DX
1314  59                              POP     CX
1315  5B                              POP     BX
1316  E4 A0                           IN      AL,NMI_PORT     ; RESET KEYBOARD LATCH
1318  CD 05                           INT     5H              ; ISSUE INTERRUPT
131A  58                              POP     AX
131B  5F                              POP     DI
131C  5E                              POP     SI              ; POP THE REST
131D  CF                              IRET
131E                          KEY62_INT ENDP
                              ;--------------------------------------------------------------------
                              ;TYPAMATIC
                              ;       THIS ROUTINE WILL CHECK KEYBOARD STATUS BITS IN KB_FLAG_2
                              ;       AND DETERMINE WHAT STATE THE KEYBOARD IS IN.  APPROPRIATE
                              ;       ACTION WILL BE TAKEN.
                              ;INPUT
                              ;       AL= SCAN CODE OF KEY WHICH TRIGGERED NON-MASKABLE INTERRUPT
                              ;OUTPUT
                              ;       CARRY BIT = 1 IF NO ACTION IS TO BE TAKEN.
                              ;       CARRY BIT = 0 MEANS SCAN CODE IN AL SHOULD BE PROCESSED
                              ;                     FURTHER.
                              ;       MODIFICATIONS TO THE VARIABLES CUR_CHAR AND VAR_DELAY ARE
                              ;       MADE.  ALSO THE PUTCHAR BIT IN KB_FLAG_2 IS TOGGLED WHEN
                              ;       THE KEYBOARD IS IN HALF RATE MODE.
                              ;--------------------------------------------------------------------
131E                          TPM     PROC    NEAR
131E  53                              PUSH    BX
131F  38 06 0085 R                    CMP     CUR_CHAR,AL     ; IS THIS A NEW CHARACTER?
1323  74 31                           JZ      TP2             ; JUMP IF SAME CHARACTER
                              ;----NEW CHARACTER CHECK FOR BREAK SEQUENCES
1325  A8 80                           TEST    AL,BREAK_BIT    ; IS THE NEW KEY A BREAK KEY?
1327  74 12                           JZ      TP0             ; JUMP IF NOT A BREAK
1329  24 7F                           AND     AL,07FH         ; CLEAR BREAK BIT
132B  38 06 0085 R                    CMP     CUR_CHAR,AL     ; IS NEW CHARACTER THE BREAK OF
                                                              ; LAST MAKE?
132F  8A C4                           MOV     AL,AH           ; RETRIEVE ORIGINAL CHARACTER
1331  75 05                           JNZ     TP              ; JUMP IF NOT THE SAME CHARACTER
1333  C6 06 0085 R 00                 MOV     CUR_CHAR,00     ; CLEAR CURRENT CHARACTER
1338  F8                      TP:     CLC                     ; CLEAR CARRY BIT
1339  5B                              POP     BX
133A  C3                              RET                     ; RETURN
```

```
                                ;----INITIALIZE A NEW CHARACTER
133B  A2 0085 R                 TP0:    MOV     CUR_CHAR,AL     ; SAVE NEW CHARACTER
133E  80 26 0086 R F0                   AND     VAR_DELAY,0F0H  ; CLEAR VARIABLE DELAY
1343  80 26 0088 R FE                   AND     KB_FLAG_2,0FEH  ; INITIAL PUTCHAR BIT AS ZERO
1348  F6 06 0088 R 02                   TEST    KB_FLAG_2,INIT_DELAY ; ARE WE INCREASING THE
                                                                ; INITIAL DELAY?
134D  74 E9                             JZ      TP              ; DEFAULT DELAY
134F  80 0E 0086 R 0F                   OR      VAR_DELAY,DELAY_RATE ; INCREASE DELAY BY 2X
1354  EB E2                             JMP     SHORT TP
                                ;----CHECK IF WE ARE IN TYPAMATIC MODE AND IF DELAY IS OVER
1356  F6 06 0088 R 08           TP2:    TEST    KB_FLAG_2,TYPE_OFF ; IS TYPAMATIC TURNED OFF?
135B  75 2B                             JNZ     TP4             ; JUMP IF TYPAMATIC RATE IS OFF
135D  8A 1E 0086 R                      MOV     BL,VAR_DELAY    ; GET VAR_DEALY
1361  80 E3 0F                          AND     BL,0FH          ; MASK OFF HIGH ORDER(SCREEN RANGE)
1364  0A DB                            OR      BL,BL           ; IS INITIAL DELAY OVER?
1366  74 0D                            JZ      TP3             ; JUMP IF DELAY IS OVER
1368  FE CB                            DEC     BL              ; DECREASE DELAY WAIT BY ANOTHER
                                                                ; CHARACTER
136A  80 26 0086 R F0                   AND     VAR_DELAY,0F0H
136F  08 1E 0086 R                      OR      VAR_DELAY,BL
1373  EB 13                             JMP     SHORT TP4
                                ;----CHECK IF TIME TO OUTPUT CHAR
1375  F6 06 0088 R 04           TP3:    TEST    KB_FLAG_2,HALF_RATE ; ARE WE IN HALF RATE MODE
137A  74 BC                             JZ      TP              ; JUMP IF WE ARE IN NORMAL MODE
137C  80 36 0088 R 01                   XOR     KB_FLAG_2,PUTCHAR ; TOGGLE BIT
1381  F6 06 0088 R 01                   TEST    KB_FLAG_2,PUTCHAR ; IS IT TIME TO PUT OUT A CHAR
1386  75 B0                             JNZ     TP              ; NOT TIME TO OUTPUT CHARACTER
1388                            TP4:                            ; SKIP THIS CHARACTER
1388  F9                                STC                     ; SET CARRY FLAG
1389  5B                                POP     BX
138A  C3                                RET
138B                            TPM     ENDP
                                ;--------------------------------------------------------------
                                ; THIS SUBROUTINE SETS DS TO POINT TO THE BIOS DATA AREA
                                ; INPUT: NONE
                                ; OUTPUT: DS IS SET
                                ;--------------------------------------------------------------
138B                            DDS     PROC    NEAR
138B  50                                PUSH    AX
138C  B8 0040                           MOV     AX,40H
138F  8E D8                             MOV     DS,AX
1391  58                                POP     AX
1392  C3                                RET
1393                            DDS     ENDP
                                ;--- INT 1A ---------------------------------------------------
                                ; TIME_OF_DAY/SOUND SOURCE SELECT
                                ;       THIS ROUTINE ALLOWS THE CLOCK TO BE SET/READ.
                                ;       AN INTERFACE FOR SETTING THE MULTIPLEXER FOR
                                ;       AUDIO SOURCE IS ALSO PROVIDED
                                ;
                                ; INPUT
                                ;   (AH) = 0    READ THE CURRENT CLOCK SETTING
                                ;               RETURNS CX = HIGH PORTION OF COUNT
                                ;                       DX = LOW PORTION OF COUNT
                                ;                       AL = 0 IF TIMER HAS NOT PASSED 24 HOURS
                                ;                       SINCE LAST READ. <> 0 IF ON ANOTHER DAY
                                ;   (AH) = 1    SET THE CURRENT CLOCK
                                ;       CX = HIGH PORTION OF COUNT
                                ;       DX = LOW PORTION OF COUNT
                                ;   (AH) = 80H  SET UP SOUND MULTIPLEXER
                                ;       AL =(SOURCE OF SOUND) --> "AUDIO OUT" OR RF MODULATOR
                                ;            00 = 8253 CHANNEL 2
                                ;            01 = CASSETTE INPUT
                                ;            02 = "AUDIO IN" LINE ON I/O CHANNEL
                                ;            03 = COMPLEX SOUND GENERATOR CHIP
                                ;
                                ; NOTE: COUNTS OCCUR AT THE RATE OF 1193180/65536 COUNTS/SEC
                                ;       (OR ABOUT 18.2 PER SECOND -- SEE EQUATES BELOW)
                                ;--------------------------------------------------------------
                                        ASSUME  CS:CODE,DS:DATA
1393                            TIME_OF_DAY     PROC    FAR
1393  FB                                STI                     ; INTERRUPTS BACK ON
1394  1E                                PUSH    DS              ; SAVE SEGMENT
1395  E8 138B R                         CALL    DDS
1398  80 FC 80                          CMP     AH,80H          ; AH=80
139B  74 2E                             JE      T4A             ; MUX_SET-UP
139D  0A E4                             OR      AH,AH           ; AH=0
139F  74 07                             JZ      T2              ; READ_TIME
13A1  FE CC                             DEC     AH              ; AH=1
13A3  74 16                             JZ      T3              ; SET_TIME
13A5  FB                        T1:     STI                     ; INTERRUPTS BACK ON
13A6  1F                                POP     DS              ; RECOVER SEGMENT
13A7  CF                                IRET                    ; RETURN TO CALLER
13A8  FA                        T2:     CLI                     ; NO TIMER INTERRUPTS WHILE READING
13A9  A0 0070 R                         MOV     AL,TIMER_OFL
13AC  C6 06 0070 R 00                   MOV     TIMER_OFL,0     ; GET OVERFLOW, AND RESET THE FLAG
13B1  8B 0E 006E R                      MOV     CX,TIMER_HIGH
13B5  8B 16 006C R                      MOV     DX,TIMER_LOW
13B9  EB EA                             JMP     T1              ; TOD_RETURN
13BB  FA                        T3:     CLI                     ; NO INTERRUPTS WHILE WRITING
13BC  89 16 006C R                      MOV     TIMER_LOW,DX
13C0  89 0E 006E R                      MOV     TIMER_HIGH,CX   ; SET THE TIME
13C4  C6 06 0070 R 00                   MOV     TIMER_OFL,0     ; RESET OVERFLOW
13C9  EB DA                             JMP     T1              ; TOD_RETURN
```

```
13CB  51                      T4A:    PUSH    CX
13CC  B1 05                           MOV     CL,5                ;
13CE  D2 E0                           SAL     AL,CL               ; SHIFT PARM BITS LEFT 5 POSITIONS
13D0  86 C4                           XCHG    AL,AH               ; SAVE PARM
13D2  E4 61                           IN      AL,PORT_B           ; GET CURRENT PORT SETTINGS
13D4  24 9F                           AND     AL,10011111B        ; ISOLATE MUX BITS
13D6  0A C4                           OR      AL,AH               ; COMBINE PORT BITS/PARM BITS
13D8  E6 61                           OUT     PORT_B,AL           ; SET PORT TO NEW VALUE
13DA  59                              POP     CX
13DB  EB C8                           JMP     T1                  ; TOD_RETURN
13DD                          TIME_OF_DAY     ENDP
                              ;---- INT 16 ------------------------------------------------------
                              ; KEYBOARD I/O
                              ;       THESE ROUTINES PROVIDE KEYBOARD SUPPORT
                              ; INPUT
                              ;       (AH)=0  READ THE NEXT ASCII CHARACTER STRUCK FROM THE
                              ;               KEYBOARD, RETURN THE RESULT IN (AL), SCAN CODE IN
                              ;               (AH)
                              ;       (AH)=1  SET THE Z FLAG TO INDICATE IF AN ASCII CHARACTER IS
                              ;               AVAILABLE TO BE READ.
                              ;               (ZF)=1 -- NO CODE AVAILABLE
                              ;               (ZF)=0 -- CODE IS AVAILABLE
                              ;               IF ZF = 0, THE NEXT CHARACTER IN THE BUFFER TO BE
                              ;               READ IS IN AX, AND THE ENTRY REMAINS IN THE BUFFER
                              ;       (AH)=2  RETURN THE CURRENT SHIFT STATUS IN AL REGISTER
                              ;               THE BIT SETTINGS FOR THIS CODE ARE INDICATED IN THE
                              ;               THE EQUATES FOR KB_FLAG
                              ;       (AH)=3  SET TYPAMATIC RATES.  THE TYPAMATIC RATE CAN BE
                              ;               CHANGED USING THE FOLLOWING FUNCTIONS:
                              ;               (AL)=0      RETURN TO DEFAULT.  RESTORES ORIGINAL
                              ;                           STATE. I.E. TYPAMATIC ON, NORMAL INITIAL
                              ;                           DELAY, AND NORMAL TYPAMATIC RATE.
                              ;               (AL)=1      INCREASE INITIAL DELAY.  THIS IS THE
                              ;                           DELAY BETWEEN THE FIRST CHARACTER AND
                              ;                           THE BURST OF TYPAMATIC CHARS.
                              ;               (AL)=2      HALF_RATE.  SLOWS TYPAMATIC CHARACTERS
                              ;                           BY ONE HALF.
                              ;               (AL)=3      COMBINES AL=1 AND AL=2.  INCREASES
                              ;                           INITIAL DELAY AND SLOWS TYPAMATIC
                              ;                           CHARACTERS BY ONE HALF.
                              ;               (AL)=4      TURN OFF TYPAMATIC CHARACTERS.  ONLY THE
                              ;                           FIRST CHARACTER IS HONORED.  ALL OTHERS
                              ;                           ARE IGNORED.
                              ;               AL IS RANGE CHECKED.  IF AL<0 OR AL>4 THE STATE
                              ;               REMAINS THE SAME.
                              ;               ***NOTE***  EACH TIME THE TYPAMATIC RATES ARE
                              ;               CHANGED ALL PREVIOUS STATES ARE REMOVED.  I.E. IF
                              ;               THE KEYBOARD IS IN THE HALF RATE MODE AND YOU WANT
                              ;               TO ADD AN INCREASE IN TYPAMATIC DELAY, YOU MUST
                              ;               CALL THIS ROUTINE WITH AH=3 AND AL=3.
                              ;       (AH)=4  ADJUST KEYBOARD BY THE VALUE IN AL AS FOLLOWS:
                              ;               (AL)=0      TURN OFF KEYBOARD CLICK.
                              ;               (AL)=1      TURN ON KEYBOARD CLICK.
                              ;               AL IS RANGE CHECKED.  THE STATE IS UNALTERED IF
                              ;               AL <> 1,0.
                              ; OUTPUT
                              ;       AS NOTED ABOVE, ONLY AX AND FLAGS CHANGED
                              ;       ALL REGISTERS RETAINED
                              ;--------------------------------------------------------------------
13DD                          KEYBOARD_IO     PROC    FAR
                              ASSUME  CS:CODE,DS:DATA
13DD  FB                              STI                         ; INTERRUPTS BACK ON
13DE  1E                              PUSH    DS                  ; SAVE CURRENT DS
13DF  53                              PUSH    BX                  ; SAVE BX TEMPORARILY
13E0  E8 1388 R                       CALL    DDS                 ; POINT DS AT BIOS DATA SEGMENT
13E3  0A E4                           OR      AH,AH               ; AH=0
13E5  74 0A                           JZ      K1                  ; ASCII_READ
13E7  FE CC                           DEC     AH                  ; AH=1
13E9  74 1E                           JZ      K2                  ; ASCII_STATUS
13EB  FE CC                           DEC     AH                  ; AH=2
13ED  74 2B                           JZ      K3                  ; SHIFT_STATUS
13EF  EB 2E                           JMP     SHORT   K3_1
                              ;------ READ THE KEY TO FIGURE OUT WHAT TO DO
13F1                          K1:                                 ; ASCII READ
13F1  FB                              STI                         ; INTERRUPTS BACK ON DURING LOOP
13F2  90                              NOP                         ; ALLOW AN INTERRUPT TO OCCUR
13F3  FA                              CLI                         ; INTERRUPTS BACK OFF
13F4  8B 1E 001A R                    MOV     BX,BUFFER_HEAD      ; GET POINTER TO HEAD OF BUFFER
13F8  3B 1E 001C R                    CMP     BX,BUFFER_TAIL      ; TEST END OF BUFFER
13FC  74 F3                           JZ      K1                  ; LOOP UNTIL SOMETHING IN BUFFER
13FE  8B 07                           MOV     AX,[BX]             ; GET SCAN CODE AND ASCII CODE
1400  E8 144F R                       CALL    K4                  ; MOVE POINTER TO NEXT POSITION
1403  89 1E 001A R                    MOV     BUFFER_HEAD,BX      ; STORE VALUE IN VARIABLE
1407  EB 43                          JMP     SHORT   RET_INT16
                              ;------ ASCII STATUS
1409  FA                      K2:     CLI                         ; INTERRUPTS OFF
140A  8B 1E 001A R                    MOV     BX,BUFFER_HEAD      ; GET HEAD POINTER
140E  3B 1E 001C R                    CMP     BX,BUFFER_TAIL      ; IF EQUAL (Z=1) THEN NOTHING THERE
1412  8B 07                           MOV     AX,[BX]
1414  FB                              STI                         ; INTERRUPTS BACK ON
1415  5B                              POP     BX                  ; RECOVER REGISTER
1416  1F                              POP     DS                  ; RECOVER SEGMENT
1417  CA 0002                         RET     2                   ; THROW AWAY FLAGS
                              ;------ SHIFT STATUS
141A  A0 0017 R               K3:     MOV     AL,KB_FLAG          ; GET THE SHIFT STATUS FLAGS
141D  EB 2D                           JMP     SHORT   RET_INT16
```

```
                                ;------- ADJUST KEY CLICK
141F  FE CC                     K3_1:   DEC     AH
1421  74 1A                             JZ      K3_3                    ; AH=3, ADJUST TYPAMATIC
1423  FE CC                             DEC     AH                      ; RANGE CHECK FOR AH=4
1425  75 25                             JNZ     RET_INT16               ; ILLEGAL FUNCTION CALL
1427  0A C0                             OR      AL,AL                   ; TURN OFF KEYBOARD CLICK?
1429  75 07                             JNZ     K3_2                    ; JUMP FOR RANGE CHECK
142B  80 26 0018 R FB                   AND     KB_FLAG_1,AND_MASK-CLICK_ON ; TURN OFF CLICK
1430  EB 1A                             JMP     SHORT RET_INT16
1432  3C 01                     K3_2:   CMP     AL,1                    ; RANGE CHECK
1434  75 16                             JNE     RET_INT16               ; NOT IN RANGE, RETURN
1436  80 0E 0018 R 04                   OR      KB_FLAG_1,CLICK_ON ; TURN ON KEYBOARD CLICK
143B  EB 0F                             JMP     SHORT   RET_INT16
                                ;------ SET TYPAMATIC
143D  3C 04                     K3_3:   CMP     AL,4                    ; CHECK FOR CORRECT RANGE
143F  7F 0B                             JG      RET_INT16               ; IF ILLEGAL VALUE IN AL IGNORE
1441  80 26 0088 R F1                   AND     KB_FLAG_2,0F1H  ; MASK OFF ANY OLD TYPAMATIC STATES
1446  D0 E0                             SHL     AL,1                    ; SHIFT TO PROPER POSITION
1448  08 06 0088 R                      OR      KB_FLAG_2,AL
144C                            RET_INT16:
144C  5B                               POP     BX                      ; RECOVER REGISTER
144D  1F                               POP     DS                      ; RECOVER REGISTER
144E  CF                               IRET                            ; RETURN TO CALLER
144F                            KEYBOARD_IO     ENDP
                                ;------ INCREMENT A BUFFER POINTER
144F                            K4      PROC    NEAR
144F  43                                INC     BX                      ; MOVE TO NEXT WORD IN LIST
1450  43                                INC     BX
1451  3B 1E 0082 R                      CMP     BX,BUFFER_END   ; AT END OF BUFFER?
1455  75 04                             JNE     K5                      ; NO, CONTINUE
1457  8B 1E 0080 R                      MOV     BX,BUFFER_START ; YES, RESET TO BUFFER BEGINNING
145B  C3                        K5:     RET
145C                            K4      ENDP
                                ;------ TABLE OF SHIFT KEYS AND MASK VALUES
145C                            K6      LABEL   BYTE
145C  52                                DB      INS_KEY                 ; INSERT KEY
145D  3A 45 46 38 1D                    DB      CAPS_KEY,NUM_KEY,SCROLL_KEY,ALT_KEY,CTL_KEY
1462  2A 36                             DB      LEFT_KEY,RIGHT_KEY
= 0008                          K6L     EQU     $-K6
                                ;------ SHIFT_MASK_TABLE
1464                            K7      LABEL   BYTE
1464  80                                DB      INS_SHIFT               ; INSERT MODE SHIFT
1465  40 20 10 08 04                    DB   CAPS_SHIFT,NUM_SHIFT,SCROLL_SHIFT,ALT_SHIFT,CTL_SHIFT
146A  02 01                             DB      LEFT_SHIFT,RIGHT_SHIFT
                                ;------ SCAN CODE TABLES
146C  1B FF 00 FF FF FF         K8              DB      27,-1,0,-1,-1,-1,30,-1
      1E FF
1474  FF FF FF 1F FF 7F                   DB      -1,-1,-1,31,-1,127,-1,17
      FF 11
147C  17 05 12 14 19 15                   DB      23,5,18,20,25,21,9,15
      09 0F
1484  10 1B 1D 0A FF 01                   DB      16,27,29,10,-1,1,19
      13
148B  04 06 07 08 0A 0B                   DB      4,6,7,8,10,11,12,-1,-1
      0C FF FF
1494  FF FF 1C 1A 18 03                   DB      -1,-1,28,26,24,3,22,2
      16 02
149C  0E 0D FF FF FF FF                   DB      14,13,-1,-1,-1,-1,-1,-1
      FF FF
14A4  20 FF                               DB      ' ',-1
                                ;-------- CTL TABLE SCAN
14A6                            K9      LABEL   BYTE
14A6  5E 5F 60 61 62 63                   DB      94,95,96,97,98,99,100,101
      64 65
14AE  66 67 FF FF 77 FF                   DB      102,103,-1,-1,119,-1,132,-1
      84 FF
14B6  73 FF 74 FF 75 FF                   DB      115,-1,116,-1,117,-1,118,-1
      76 FF
14BE  FF                                  DB      -1
                                ;------- LC TABLE
14BF                            K10     LABEL   BYTE
14BF  1B 31 32 33 34 35                   DB      01BH,'1234567890-=',08H,09H
      36 37 38 39 30 2D
      3D 08 09
14CE  71 77 65 72 74 79                   DB      'qwertyuiop[]',0DH,-1,'asdfghjkl;',027H
      75 69 6F 70 5B 5D
      0D FF 61 73 64 66
      67 68 6A 6B 6C 3B
      27
14E7  60 FF 5C 7A 78 63                   DB      60H,-1,5CH,'zxcvbnm,./',-1,'*',-1,' '
      76 62 6E 6D 2C 2E
      2F FF 2A FF 20
14F8  FF                                  DB      -1
                                ;------ UC TABLE
14F9                            K11     LABEL   BYTE
14F9  1B 21 40 23 24 25                   DB      27,'!@#$',37,05EH,'&*()_+',08H,0
      5E 26 2A 28 29 5F
      2B 08 00
1508  51 57 45 52 54 59                   DB      'QWERTYUIOP{}',0DH,-1,'ASDFGHJKL:"'
      55 49 4F 50 7B 7D
      0D FF 41 53 44 46
      47 48 4A 4B 4C 3A
      22
1521  7E FF 7C 5A 58 43                   DB      07EH,-1,'|ZXCVBNM<>?',-1,0,-1,' ',-1
      56 42 4E 4D 3C 3E
      3F FF 00 FF 20 FF
```

```
                                ;------ UC TABLE SCAN
1533                            K12     LABEL   BYTE
1533  54 55 56 57 58 59                        DB      84,85,86,87,88,89,90
      5A
153A  5B 5C 5D                                  DB      91,92,93
                                ;------ ALT TABLE SCAN
153D                            K13     LABEL   BYTE
153D  68 69 6A 6B 6C                            DB      104,105,106,107,108
1542  6D 6E 6F 70 71                            DB      109,110,111,112,113
                                ;------ NUM STATE TABLE
1547                            K14     LABEL   BYTE
1547  37 38 39 2D 34 35                        DB      '789-456+1230.'
      36 2B 31 32 33 30
      2E

                                ;------ BASE CASE TABLE
1554                            K15     LABEL   BYTE
1554  47 48 49 FF 4B FF                        DB      71,72,73,-1,75,-1,77
      4D
1558  FF 4F 50 51 52 53                        DB      -1,79,80,81,82,83
                                ;------ KEYBOARD INTERRUPT ROUTINE
1561                            KB_INT  PROC    FAR
1561  FB                                STI                     ; ALLOW FURTHER INTERRUPTS
1562  50                                PUSH    AX
1563  53                                PUSH    BX
1564  51                                PUSH    CX
1565  52                                PUSH    DX
1566  56                                PUSH    SI
1567  57                                PUSH    DI
1568  1E                                PUSH    DS
1569  06                                PUSH    ES
156A  FC                                CLD                     ; FORWARD DIRECTION
156B  E8 1388 R                         CALL    DDS
156E  8A E0                             MOV     AH,AL           ; SAVE SCAN CODE IN AH
                                ;------ TEST FOR OVERRUN SCAN CODE FROM KEYBOARD
1570  3C FF                            CMP     AL,0FFH         ; IS THIS AN OVERRUN CHAR?
1572  75 1B                            JNZ     K16             ; NO, TEST FOR SHIFT KEY
1574  BB 0080                          MOV     BX,80H          ; DURATION OF ERROR BEEP
1577  B9 0048                          MOV     CX,48H          ; FREQUENCY OF TONE
157A  E8 E035 R                         CALL    KB_NOISE        ; BUFFER FULL BEEP
157D  80 26 0017 R F0                   AND     KB_FLAG,0F0H    ; CLEAR ALT,CLRL,LEFT AND RIGHT
                                                                ; SHIFTS
1582  80 26 0018 R 0F                   AND     KB_FLAG_1,0FH   ; CLEAR POTENTIAL BREAK OF INS,CAPS
                                                                ; ,NUM AND SCROLL SHIFT
1587  80 26 0088 R 1F                   AND     KB_FLAG_2,1FH   ; CLEAR FUNCTION STATES
158C  E9 164A R                         JMP     K26             ; END OF INTERRUPT
                                ;------ TEST FOR SHIFT KEYS
158F                            K16:                            ; TEST_SHIFT
158F  24 7F                             AND     AL,07FH         ; TURN OFF THE BREAK BIT
1591  0E                                PUSH    CS
1592  07                                POP     ES              ; ESTABLISH ADDRESS OF SHIFT TABLE
1593  BF 145C R                         MOV     DI,OFFSET K6    ; SHIFT KEY TABLE
1596  B9 0008                          MOV     CX,K6L          ; LENGTH
1599  F2/ AE                           REPNE   SCASB           ; LOOK THROUGH THE TABLE FOR A
                                                                ; MATCH
159B  8A C4                             MOV     AL,AH           ; RECOVER SCAN CODE
159D  74 03                             JE      K17             ; JUMP IF MATCH FOUND
159F  E9 163A R                         JMP     K25             ; IF NO MATCH, THEN SHIFT NOT FOUND
                                ;------ SHIFT KEY FOUND
15A2  81 EF 145D R              K17:    SUB     DI,OFFSET K6+1  ; ADJUST PTR TO SCAN CODE MATCH
15A6  2E: 8A A5 1464 R                  MOV     AH,CS:K7[DI]    ; GET MASK INTO AH
15AB  A8 80                             TEST    AL,80H          ; TEST FOR BREAK KEY
15AD  75 51                             JNZ     K23             ; BREAK_SHIFT_FOUND
                                ;------ SHIFT MAKE FOUND, DETERMINE SET OR TOGGLE
15AF  80 FC 10                          CMP     AH,SCROLL_SHIFT
15B2  73 07                             JAE     K18             ; IF SCROLL SHIFT OR ABOVE, TOGGLE
                                                                ; KEY
                                ;------ PLAIN SHIFT KEY, SET SHIFT ON
15B4  08 26 0017 R                      OR      KB_FLAG,AH      ; TURN ON SHIFT BIT
15B8  E9 164A R                         JMP     K26             ; INTERRUPT_RETURN
                                ;------ TOGGLED SHIFT KEY, TEST FOR 1ST MAKE OR NOT
15BB                            K18:                            ; SHIFT-TOGGLE
15BB  F6 06 0017 R 04                   TEST    KB_FLAG, CTL_SHIFT ; CHECK CTL SHIFT STATE
15C0  75 78                             JNZ     K25             ; JUMP IF CTL STATE
15C2  3C 52                             CMP     AL, INS_KEY     ; CHECK FOR INSERT KEY
15C4  75 22                             JNZ     K22             ; JUMP IF NOT INSERT KEY
15C6  F6 06 0017 R 08                   TEST    KB_FLAG, ALT_SHIFT ; CHECK FOR ALTERNATE SHIFT
15CB  75 6D                             JNZ     K25             ; JUMP IF ALTERNATE SHIFT
15CD  F6 06 0017 R 20                   TEST    KB_FLAG, NUM_STATE ; CHECK FOR BASE STATE
15D2  75 0D                             JNZ     K21             ; JUMP IF NUM LOCK IS ON
15D4  F6 06 0017 R 03                   TEST    KB_FLAG, LEFT_SHIFT+ RIGHT_SHIFT        ;
15D9  74 0D                             JZ      K22             ; JUMP IF BASE STATE
15DB                            K20:                            ; NUMERIC ZERO, NOT INSERT KEY
15DB  B8 5230                           MOV     AX, 5230H       ; PUT OUT AN ASCII ZERO
15DE  E9 17EC R                         JMP     K57             ; BUFFER_FILL
15E1                            K21:                            ; MIGHT BE NUMERIC
15E1  F6 06 0017 R 03                   TEST    KB_FLAG, LEFT_SHIFT+ RIGHT_SHIFT
15E6  74 F3                             JZ      K20             ; JUMP NUMERIC, NOT INSERT
15E8                            K22:                            ; SHIFT TOGGLE KEY HIT; PROCESS IT
15E8  84 26 0018 R                      TEST    AH,KB_FLAG_1    ; IS KEY ALREADY DEPRESSED
15EC  75 5C                             JNZ     K26             ; JUMP IF KEY ALREADY DEPRESSED
15EE  08 26 0018 R                      OR      KB_FLAG_1,AH    ; INDICATE THAT THE KEY IS
                                                                ; DEPRESSED
15F2  30 26 0017 R                      XOR     KB_FLAG,AH      ; TOGGLE THE SHIFT STATE
15F6  3C 52                             CMP     AL,INS_KEY      ; TEST FOR 1ST MAKE OF INSERT KEY
15F8  75 50                             JNE     K26             ; JUMP IF NOT INSERT KEY
15FA  B8 5200                           MOV     AX,INS_KEY*256  ; SET SCAN CODE INTO AH, 0 INTO AL
15FD  E9 17EC R                         JMP     K57             ; PUT INTO OUTPUT BUFFER
```

```
                                ;------ BREAK SHIFT FOUND
1600                            K23:                            ; BREAK-SHIFT-FOUND
1600  80 FC 10                          CMP     AH,SCROLL_SHIFT ; IS THIS A TOGGLE KEY
1603  73 1A                             JAE     K24             ; YES, HANDLE BREAK TOGGLE
1605  F6 D4                             NOT     AH              ; INVERT MASK
1607  20 26 0017 R                      AND     KB_FLAG,AH      ; TURN OFF SHIFT BIT
160B  3C B8                             CMP     AL,ALT_KEY+80H  ; IS THIS ALTERNATE SHIFT RELEASE
160D  75 3B                             JNE     K26             ; INTERRUPT_RETURN
                                ;------ ALTERNATE SHIFT KEY RELEASED, GET THE VALUE INTO BUFFER
160F  A0 0019 R                         MOV     AL,ALT_INPUT
1612  32 E4                             XOR     AH,AH           ; SCAN CODE OF 0
1614  88 26 0019 R                      MOV     ALT_INPUT,AH    ; ZERO OUT THE FIELD
1618  0A C0                             OR      AL,AL           ; WAS THE INPUT=0?
161A  74 2E                             JE      K26             ; INTERRUPT_RETURN
161C  E9 17F5 R                         JMP     K58             ; IT WASN'T, SO PUT IN BUFFER
161F                            K24:                            ; BREAK-TOGGLE
161F  3C BA                             CMP     AL,CAPS_KEY+BREAK_BIT ; SPECIAL CASE OF TOGGLE KEY
1621  75 0F                             JNE     K24_1           ; JUMP AROUND POTENTIAL UPDATE
1623  F6 06 0018 R 02                   TEST    KB_FLAG_1,CLICK_SEQUENCE
1628  74 08                             JZ      K24_1           ; JUMP IF NOT SPECIAL CASE
162A  80 26 0018 R FD                   AND     KB_FLAG_1,AND_MASK-CLICK_SEQUENCE ; MASK OFF MAKE
                                                                ; OF CLICK
162F  EB 19 90                          JMP     K26             ; INTERRUPT IS OVER
                                ;------ BREAK OF NORMAL TOGGLE
1632  F6 D4                     K24_1:  NOT     AH              ; INVERT MASK
1634  20 26 0018 R                      AND     KB_FLAG_1,AH    ; INDICATE NO LONGER DEPRESSED
1638  EB 10                             JMP     SHORT K26       ; INTERRUPT_RETURN
                                ;------ TEST FOR HOLD STATE
163A                            K25:                            ; NO-SHIFT-FOUND
163A  3C 80                             CMP     AL,80H          ; TEST FOR BREAK KEY
163C  73 0C                             JAE     K26             ; NOTHING FOR BREAK CHARS FROM HERE
                                                                ; ON
163E  F6 06 0018 R 08                   TEST    KB_FLAG_1,HOLD_STATE ; ARE WE IN HOLD STATE?
1643  74 0E                             JZ      K28             ; BRANCH AROUND TEST IF NOT
1645  80 26 0018 R F7                   AND     KB_FLAG_1,NOT HOLD_STATE ; TURN OFF THE HOLD STATE
                                                                ; BIT
164A                            K26:                            ; INTERRUPT-RETURN
164A  07                                POP     ES
164B  1F                                POP     DS
164C  5F                                POP     DI
164D  5E                                POP     SI
164E  5A                                POP     DX
164F  59                                POP     CX
1650  5B                                POP     BX
1651  58                                POP     AX              ; RESTORE STATE
1652  CF                                IRET                    ; RETURN, INTERRUPTS BACK ON WITH
                                                                ; FLAG CHANGE
                                ;------ NOT IN  HOLD STATE, TEST FOR SPECIAL CHARS
1653                            K28:                            ; NO-HOLD-STATE
1653  F6 06 0017 R 08                   TEST    KB_FLAG,ALT_SHIFT ; ARE WE IN ALTERNATE SHIFT
1658  75 03                             JNZ     K29             ; JUMP IF ALTERNATE SHIFT
165A  E9 1749 R                         JMP     K38             ; JUMP IF NOT ALTERNATE
                                ;------ TEST FOR ALT+CTRL KEY SEQUENCES
165D                            K29:                            ; TEST-RESET
165D  F6 06 0017 R 04                   TEST    KB_FLAG,CTL_SHIFT ; ARE WE IN CONTROL SHIFT ALSO
1662  74 69                             JZ      K31             ; NO_RESET
1664  3C 53                             CMP     AL,DEL_KEY      ; SHIFT STATE IS THERE, TEST KEY
1666  75 09                             JNE     K29_1           ; NO_RESET
                                ;------ CTL-ALT-DEL HAS BEEN FOUND, DO I/O CLEANUP
1668  C7 06 0072 R 1234                MOV     RESET_FLAG, 1234H ; SET FLAG FOR RESET FUNCTION
166E  E9 0043 R                         JMP     NEAR PTR RESET  ; JUMP TO POWER ON DIAGNOSTICS
1671  3C 52                     K29_1:  CMP     AL,INS_KEY      ; CHECK FOR RESET WITH DIAGNOSTICS
1673  75 09                             JNE     K29_2           ; CHECK FOR OTHER
                                                                ; ALT-CTRL-SEQUENCES
                                ;------ ALT-CTRL-INS HAS BEEN FOUND
1675  C7 06 0072 R 4321                MOV     RESET_FLAG,4321H ; SET FLAG FOR DIAGNOSTICS
167B  E9 0043 R                         JMP     NEAR PTR RESET  ; LEVEL 1 DIAGNOSTICS
167E  3C 3A                     K29_2:  CMP     AL,CAPS_KEY     ; CHECK FOR KEYBORAD CLICK TOGGLE
1680  75 13                             JNE     K29_3           ; CHECK FOR SCREEN ADJUSTMENT
                                ;------ ALT+CTRL+CAPSLOCK HAS BEEN FOUND
1682  F6 06 0018 R 02                   TEST    KB_FLAG_1,CLICK_SEQUENCE
1687  75 C1                             JNZ     K26             ; JUMP IF SEQUENCE HAS ALREADY
                                                                ; OCCURED
1689  80 36 0018 R 04                   XOR     KB_FLAG_1,CLICK_ON ; TOGGLE BIT FOR AUDIO KEYSTROKE
                                                                ; FEEDBACK
168E  80 0E 0018 R 02                   OR      KB_FLAG_1,CLICK_SEQUENCE ; SET CLICK_SEQUENCE STATE
1693  EB B5                             JMP     SHORT K26       ; INTERRUPT IS OVER
1695  3C 4D                     K29_3:  CMP     AL,RIGHT_ARROW  ; ADJUST SCREEN TO THE RIGHT?
1697  75 12                             JNE     K29_4           ; LOOK FOR RIGHT ADJUSTMENT
1699  E8 186E R                         CALL    GET_POS         ; GET THE # OF POSITIONS SCREEN IS
                                                                ; SHIFTED
169C  3C FC                             CMP     AL,0-RANGE      ; IS SCREEN SHIFTED AS FAR AS
                                                                ; POSSIBLE?
169E  7C AA                             JL      K26             ; OUT OF RANGE
16A0  FE 0E 0089 R                      DEC     HORZ_POS        ; SHIFT VALUE TO THE RIGHT
16A4  FE C8                             DEC     AL              ; DECREASE RANGE VALUE
16A6  E8 187A R                         CALL    PUT_POS         ; RESTORE STORAGE LOCATION
16A9  EB 14                             JMP     SHORT K29_5     ; ADJUST
16AB  3C 4B                     K29_4:  CMP     AL,LEFT_ARROW   ; ADJUST SREEN TO THE LEFT?
16AD  75 1E                             JNE     K31             ; NOT AN ALT_CTRL SEQUENCE
16AF  E8 186E R                         CALL    GET_POS         ; GET NUMBER OF POSITIONS SCREEN IS
                                                                ; SHIFTED
16B2  3C 04                             CMP     AL,RANGE        ; IS SCREEN SHIFTED AS FAR AS
                                                                ; POSSIBLE?
16B4  7F 94                             JG      K26
16B6  FE 06 0089 R                      INC     HORZ_POS        ; SHIFT SCREEN TO THE LEFT
16BA  FE C0                             INC     AL              ; INCREASE NUMBER OF POSITIONS
                                                                ; SCREEN IS SHIFTED
16BC  E8 187A R                         CALL    PUT_POS         ; PUT POSTION BACK IN STORAGE
```

```
16BF  B0 02                   K29_5:  MOV     AL,2            ; ADJUST
16C1  BA 03D4                         MOV     DX,3D4H         ; ADDRESS TO CRT CONTROLLER
16C4  EE                              OUT     DX,AL
16C5  A0 0089 R                       MOV     AL,HORZ_POS     ; COLUMN POSITION
16C8  42                              INC     DX              ; POINT AT DATA REGISTER
16C9  EE                              OUT     DX,AL           ; MOV POSITION
16CA  E9 164A R                       JMP     K26
                                    ;------ IN ALTERNATE SHIFT, RESET NOT FOUND
16CD                                K31:                    ; NO-RESET
16CD  3C 39                           CMP     AL,57           ; TEST FOR SPACE KEY
16CF  75 29                           JNE     K32             ; NOT THERE
16D1  B0 20                           MOV     AL,' '          ; SET SPACE CHAR
16D3  E9 17EC R                       JMP     K57             ; BUFFER_FILL
                                    ;----- ALT-INPUT-TABLE
16D6                                K30     LABEL   BYTE
16D6  52 4F 50 51 4B 4C             DB      82,79,80,81,75,76,77
      4D
16DD  47 48 49                      DB      71,72,73        ; 10 NUMBERS ON KEYPAD
                                    ;------ SUPER-SHIFT-TABLE
16E0  10 11 12 13 14 15             DB      16,17,18,19,20,21,22,23 ; A-Z TYPEWRITER CHARS
      16 17
16E8  18 19 1E 1F 20 21             DB      24,25,30,31,32,33,34,35
      22 23
16F0  24 25 26 2C 2D 2E             DB      36,37,38,44,45,46,47,48
      2F 30
16F8  31 32                         DB      49,50
                                    ;------ LOOK FOR KEY PAD ENTRY
16FA                                K32:                    ; ALT-KEY-PAD
16FA  BF 16D6 R                       MOV     DI,OFFSET K30   ; ALT-INPUT-TABLE
16FD  B9 000A                         MOV     CX,10           ; LOOK FOR ENTRY USING KEYPAD
1700  F2/ AE                         REPNE   SCASB           ; LOOK FOR MATCH
1702  75 13                          JNE     K33             ; NO_ALT_KEYPAD
1704  81 EF 16D7 R                   SUB     DI,OFFSET K30+1 ; DI NOW HAS ENTRY VALUE
1708  A0 0019 R                     MOV     AL,ALT_INPUT    ; GET THE CURRENT BYTE
170B  B4 0A                         MOV     AH,10           ; MULTIPLY BY 10
170D  F6 E4                         MUL     AH
170F  03 C7                         ADD     AX,DI           ; ADD IN THE LATEST ENTRY
1711  A2 0019 R                     MOV     ALT_INPUT,AL    ; STORE IT AWAY
1714  E9 164A R                     JMP     K26             ; THROW AWAY THAT KEYSTROKE
                                    ;------ LOOK FOR SUPERSHIFT ENTRY
1717                                K33:                    ; NO-ALT-KEYPAD
1717  C6 06 0019 R 00               MOV     ALT_INPUT,0     ; ZERO ANY PREVIOUS ENTRY INTO
                                                            ; INPUT
171C  B9 001A                       MOV     CX,26           ; DI,ES ALREADY POINTING
171F  F2/ AE                        REPNE   SCASB           ; LOOK FOR MATCH IN ALPHABET
1721  75 05                        JNE     K34             ; NOT FOUND, FUNCTION KEY OR OTHER
1723  32 C0                        XOR     AL,AL           ; ASCII CODE OF ZERO
1725  E9 17EC R                     JMP     K57             ; PUT IT IN THE BUFFER
                                    ;------ LOOK FOR TOP ROW OF ALTERNATE SHIFT
1728                                K34:                    ; ALT-TOP-ROW
1728  3C 02                        CMP     AL,2            ; KEY WITH '1' ON IT
172A  72 0C                        JB      K35             ;  NOT ONE OF INTERESTING KEYS
172C  3C 0E                        CMP     AL,14           ;  IS IT IN THE REGION?
172E  73 08                        JAE     K35             ;  ALT-FUNCTION
1730  80 C4 76                     ADD     AH,118          ;  CONVERT PSUEDO SCAN CODE TO
                                                            ; RANGE
1733  32 C0                        XOR     AL,AL           ;  INDICATE AS SUCH
1735  E9 17EC R                     JMP     K57             ;  BUFFER_FILL
                                    ;------ TRANSLATE ALTERNATE SHIFT PSEUDO SCAN CODES
1738                                K35:                    ; ALT-FUNCTION
1738  3C 3B                        CMP     AL,59           ; TEST FOR IN TABLE
173A  73 03                        JAE     K37             ; ALT-CONTINUE
173C                                K36:                    ; CLOSE-RETURN
173C  E9 164A R                     JMP     K26             ; IGNORE THE KEY
173F                                K37:                    ; ALT-CONTINUE
173F  3C 47                        CMP     AL,71           ; IN KEYPAD REGION
1741  73 F9                        JAE     K36             ; IF SO, IGNORE
1743  BB 153D R                     MOV     BX,OFFSET K13   ; ALT SHIFT PSEUDO SCAN TABLE
1746  E9 1863 R                     JMP     K63             ; TRANSLATE THAT
                                    ;------ NOT IN ALTERNATE SHIFT
1749                                K38:                    ; NOT-ALT-SHIFT
1749  F6 06 0017 R 04               TEST    KB_FLAG,CTL_SHIFT ; ARE WE IN CONTROL SHIFT?
174E  74 34                        JZ      K44             ; NOT-CTL-SHIFT
                                    ;------ CONTROL SHIFT, TEST SPECIAL CHARACTERS
                                    ;------ TEST FOR BREAK AND PAUSE KEYS
1750  3C 46                        CMP     AL,SCROLL_KEY   ; TEST FOR BREAK
1752  75 19                        JNE     K41             ; NO-BREAK
1754  8B 1E 001A R                  MOV     BX,BUFFER_HEAD  ; GET CURRENT BUFFER HEAD
1758  C6 06 0071 R 80               MOV     BIOS_BREAK,80H  ; TURN ON BIOS_BREAK BIT
175D  CD 1B                        INT     1BH             ; BREAK INTERRUPT VECTOR
175F  2B C0                        SUB     AX,AX           ; PUT OUT DUMMY CHARACTER
1761  89 07                        MOV     [BX],AX         ; PUT DUMMY CHAR AT BUFFER HEAD
1763  E8 144F R                     CALL    K4              ; UPDATE BUFFER POINTER
1766  89 1E 001C R                  MOV     BUFFER_TAIL,BX  ; UPDATE TAIL
176A  E9 164A R                     JMP     K26             ; DONE WITH INTERUPT
176D                                K41:                    ; NO-PAUSE
                                    ;------ TEST SPECIAL CASE KEY 55
176D  3C 37                        CMP     AL,55
176F  75 06                        JNE     K42             ; NOT-KEY-55
1771  B8 7200                      MOV     AX,114*256      ; START/STOP PRINTING SWITCH
1774  EB 76 90                     JMP     K57             ; BUFFER_FILL
```

```
                                ;------ SET UP TO TRANSLATE CONTROL SHIFT
1777                            K42:                            ; NOT-KEY-55
1777  BB 146C R                         MOV     BX,OFFSET K8    ; SET UP TO TRANSLATE CTL
177A  3C 3B                             CMP     AL,59           ; IS IT IN TABLE?
177C  72 6A                             JB      K56             ; YES, GO TRANSLATE CHAR
                                                                ; CTL-TABLE-TRANSLATE
177E  BB 14A6 R                         MOV     BX,OFFSET K9    ; CTL TABLE SCAN
1781  E9 1863 R                         JMP     K63             ; TRANSLATE_SCAN
                                ;------ NOT IN CONTROL SHIFT
1784                            K44:                            ; NOT-CTL-SHIFT
1784  3C 47                             CMP     AL,71           ; TEST FOR KEYPAD REGION
1786  73 1F                             JAE     K48             ; HANDLE KEYPAD REGION
1788  F6 06 0017 R 03                   TEST    KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT
178D  74 4E                             JZ      K54             ; TEST FOR SHIFT STATE
                                ;------ UPPER CASE, HANDLE SPECIAL CASES
178F  3C 0F                             CMP     AL,15           ; BACK TAB KEY
1791  75 05                             JNE     K46             ; NOT-BACK-TAB
1793  B8 0F00                           MOV     AX,15*256       ; SET PSEUDO SCAN CODE
1796  EB 54                             JMP     SHORT K57       ; BUFFER_FILL
1798                            K46:                            ; NOT-PRINT-SCREEN
1798  3C 3B                             CMP     AL,59           ; FUNCTION KEYS
179A  72 06                             JB      K47             ; NOT-UPPER-FUNCTION
179C  BB 1533 R                         MOV     BX,OFFSET K12   ; UPPER CASE PSEUDO SCAN CODES
179F  E9 1863 R                         JMP     K63             ; TRANSLATE_SCAN
17A2                            K47:                            ; NOT-UPPER-FUNCTION
17A2  BB 14F9 R                         MOV     BX,OFFSET K11   ; POINT TO UPPER CASE TABLE
17A5  EB 41                             JMP     SHORT K56       ; OK, TRANSLATE THE CHAR
                                ;------ KEYPAD KEYS, MUST TEST NUM LOCK FOR DETERMINATION
17A7                            K48:                            ; KEYPAD-REGION
17A7  F6 06 0017 R 20                   TEST    KB_FLAG,NUM_STATE ; ARE WE IN NUM_LOCK?
17AC  75 21                             JNZ     K52             ; TEST FOR SURE
17AE  F6 06 0017 R 03                   TEST    KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT ; ARE WE IN SHIFT
                                                                ; STATE
17B3  75 21                             JNZ     K53             ; IF SHIFTED, REALLY NUM STATE
                                ;------ BASE CASE FOR KEYPAD
17B5                            K49:                            ; BASE-CASE
17B5  3C 4A                             CMP     AL,74           ; SPECIAL CASE FOR A COUPLE OF KEYS
17B7  74 0C                             JE      K50             ; MINUS
17B9  3C 4E                             CMP     AL,78
17BB  74 0D                             JE      K51
17BD  2C 47                             SUB     AL,71           ; CONVERT ORIGIN
17BF  BB 1554 R                         MOV     BX,OFFSET K15   ; BASE CASE TABLE
17C2  E9 1865 R                         JMP     K64             ; CONVERT TO PSEUDO SCAN
17C5  B8 4A2D                   K50:    MOV     AX,74*256+'-'   ; MINUS
17C8  EB 22                             JMP     SHORT K57       ; BUFFER_FILL
17CA  B8 4E2B                   K51:    MOV     AX,78*256+'+'   ; PLUS
17CD  EB 1D                             JMP     SHORT K57       ; BUFFER_FILL
                                ;------ MIGHT BE NUM LOCK, TEST SHIFT STATUS
17CF                            K52:                            ; ALMOST-NUM-STATE
17CF  F6 06 0017 R 03                   TEST    KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT
17D4  75 DF                             JNZ     K49             ; SHIFTED TEMP OUT OF NUM STATE
17D6                            K53:                            ; REALLY_NUM_STATE
17D6  2C 46                             SUB     AL,70           ; CONVERT ORIGIN
17D8  BB 1547 R                         MOV     BX,OFFSET K14   ; NUM STATE TABLE
17DB  EB 0B                             JMP     SHORT K56       ; TRANSLATE_CHAR
                                ;------ PLAIN OLD LOWER CASE
17DD                            K54:                            ; NOT-SHIFT
17DD  3C 3B                             CMP     AL,59           ; TEST FOR FUNCTION KEYS
17DF  72 04                             JB      K55             ; NOT-LOWER-FUNCTION
17E1  32 C0                             XOR     AL,AL           ; SCAN CODE IN AH ALREADY
17E3  EB 07                             JMP     SHORT K57       ; BUFFER_FILL
17E5                            K55:                            ; NOT-LOWER-FUNCTION
17E5  BB 14BF R                         MOV     BX,OFFSET K10   ; LC TABLE
                                ;------ TRANSLATE THE CHARACTER
17E8                            K56:                            ; TRANSLATE-CHAR
17E8  FE C8                             DEC     AL              ; CONVERT ORIGIN
17EA  2E: D7                            XLAT    CS:K11          ; CONVERT THE SCAN CODE TO ASCII
                                ;------ PUT CHARACTER INTO BUFFER
17EC                            K57:                            ; BUFFER-FILL
17EC  3C FF                             CMP     AL,-1           ; IS THIS AN IGNORE CHAR?
17EE  74 1F                             JE      K59             ; YES, DO NOTHING WITH IT
17F0  80 FC FF                          CMP     AH,-1           ; LOOK FOR -1 PSEUDO SCAN
17F3  74 1A                             JE      K59             ; NEAR_INTERRUPT_RETURN
                                ;------ HANDLE THE CAPS LOCK PROBLEM
17F5                            K58:                            ; BUFFER-FILL-NOTEST
17F5  F6 06 0017 R 40                   TEST    KB_FLAG,CAPS_STATE ; ARE WE IN CAPS LOCK STATE?
17FA  74 20                             JZ      K61             ; SKIP IF NOT
                                ;------ IN CAPS LOCK STATE
17FC  F6 06 0017 R 03                   TEST    KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT ; TEST FOR SHIFT
                                                                ; STATE
1801  74 0F                             JZ      K60             ; IF NOT SHIFT, CONVERT LOWER TO
                                                                ; UPPER
                                ;------ CONVERT ANY UPPER CASE TO LOWER CASE
1803  3C 41                             CMP     AL,'A'          ; FIND OUT IF ALPHABETIC
1805  72 15                             JB      K61             ; NOT_CAPS_STATE
1807  3C 5A                             CMP     AL,'Z'
1809  77 11                             JA      K61             ; NOT_CAPS_STATE
180B  04 20                             ADD     AL,'a'-'A'      ; CONVERT TO LOWER CASE
180D  EB 0D                             JMP     SHORT K61       ; NOT_CAPS_STATE
180F                            K59:                            ; NEAR-INTERRUPT-RETURN
180F  E9 164A R                         JMP     K26             ; INTERRUPT_RETURN
                                ;------ CONVERT ANY LOWER CASE TO UPPER CASE
1812                            K60:                            ; LOWER-TO-UPPER
1812  3C 61                             CMP     AL,'a'          ; FIND OUT IF ALPHABETIC
1814  72 06                             JB      K61             ; NOT_CAPS_STATE
1816  3C 7A                             CMP     AL,'z'
1818  77 02                             JA      K61             ; NOT_CAPS_STATE
181A  2C 20                             SUB     AL,'a'-'A'      ; CONVERT TO UPPER CASE
```

```
181C                            K61:                                    ; NOT-CAPS-STATE
181C  8B 1E 001C R                      MOV     BX,BUFFER_TAIL  ; GET THE END POINTER TO THE BUFFER
1820  8B F3                            MOV     SI,BX           ; SAVE THE VALUE
1822  E8 144F R                        CALL    K4              ; ADVANCE THE TAIL
1825  3B 1E 001A R                     CMP     BX,BUFFER_HEAD  ; HAS THE BUFFER WRAPPED AROUND?
1829  75 1D                           JNE     K61_1           ; BUFFER_FULL_BEEP
182B  53                              PUSH    BX              ; SAVE BUFFER_TAIL
182C  BB 0080                         MOV     BX,080H         ; DURATION OF ERROR BEEP
182F  B9 0048                         MOV     CX,48H          ; FREQUENCY OF ERROR BEEP HALF TONE
1832  E8 E035 R                       CALL    KB_NOISE        ; OUTPUT NOISE
1835  80 26 0017 R F0                 AND     KB_FLAG,0F0H    ; CLEAR ALT,CLRL,LEFT AND RIGHT
                                                              ; SHIFTS
183A  80 26 0018 R 0F                 AND     KB_FLAG_1,0FH   ; CLEAR POTENTIAL BREAK OF INS,CAPS
                                                              ; ,NUM AND SCROLL SHIFT
183F  80 26 0088 R 1F                 AND     KB_FLAG_2,1FH   ; CLEAR FUNCTION STATES
1844  5B                             POP     BX              ; RETRIEVE BUFFER TAIL
1845  E9 164A R                       JMP     K26             ; RETURN FROM INTERRUPT
1848  F6 06 0018 R 04       K61_1:   TEST    KB_FLAG_1,CLICK_ON ; IS AUDIO FEEDBACK ENABLED?
184D  74 0B                           JZ      K61_2           ; NO, JUST PUT IN BUFFER
184F  53                              PUSH    BX              ; SAVE BUFFER_TAIL VALUE
1850  BB 0001                         MOV     BX,1H           ; DURATION OF CLICK
1853  B9 0010                         MOV     CX,10H          ; FREQUENCY OF CLICK
1856  E8 E035 R                       CALL    KB_NOISE        ; OUTPUT AUDIO FEEDBACK OF KEY
                                                              ; STROKE
1859  5B                             POP     BX              ; RETRIEVE BUFFER_TAIL VALUE
185A  89 04                   K61_2:  MOV     [SI],AX         ; STORE THE VALUE
185C  89 1E 001C R                    MOV     BUFFER_TAIL,BX  ; MOVE THE POINTER UP
1860  E9 164A R                       JMP     K26             ; INTERRUPT_RETURN
                              ;------ TRANSLATE SCAN FOR PSEUDO SCAN CODES
1863                          K63:                            ; TRANSLATE-SCAN
1863  2C 3B                           SUB     AL,59           ; CONVERT ORIGIN TO FUNCTION KEYS
1865                          K64:                            ; TRANSLATE-SCAN-ORGD
1865  2E: D7                          XLAT    CS:K9           ; CTL TABLE SCAN
1867  8A E0                           MOV     AH,AL           ; PUT VALUE INTO AH
1869  32 C0                           XOR     AL,AL           ; ZERO ASCII CODE
186B  E9 17EC R                       JMP     K57             ; PUT IT INTO THE BUFFER
186E                          KB_INT  ENDP
                              ;--------------------------------------------------------------------------
                              ;GET_POS
                              ;       THIS ROUTINE WILL SHIFT THE VALUE STORED IN THE HIGH NIBBLE
                              ;       OF THE VARIABLE VAR_DELAY TO THE LOW NIBBLE.
                              ;INPUT
                              ;       NONE.  IT IS ASSUMED THAT DS POINTS AT THE BIOS DATA AREA
                              ;OUTPUT
                              ;       AL CONTAINS THE SHIFTED VALUE.
                              ;--------------------------------------------------------------------------
186E                          GET_POS PROC    NEAR
186E  51                              PUSH    CX              ; SAVE SHIFT REGISTER
186F  A0 0086 R                       MOV     AL,BYTE PTR VAR_DELAY ; GET STORAGE LOCATION
1872  24 F0                           AND     AL,0F0H         ; MASK OFF LOW NIBBLE
1874  B1 04                           MOV     CL,4            ; SHIFT OF FOUR BIT POSITIONS
1876  D2 F8                           SAR     AL,CL           ; SHIFT THE VALUE SIGN EXTENDED
1878  59                              POP     CX              ; RESTORE THE VALUE
1879  C3                              RET
187A                          GET_POS ENDP
                              ;--------------------------------------------------------------------------
                              ;PUT_POS
                              ;       THIS ROUTINE WILL TAKE THE VALUE IN LOW ORDER NIBBLE IN
                              ;       AL AND STORE IT IN THE HIGH ORDER OF VAR_DELAY
                              ;INPUT
                              ;       AL CONTAINS THE VALUE FOR STORAGE
                              ;OUTPUT
                              ;       NONE.
                              ;--------------------------------------------------------------------------
187A                          PUT_POS PROC    NEAR
187A  51                              PUSH    CX              ; SAVE REGISTER
187B  B1 04                           MOV     CL,4            ; SHIFT COUNT
187D  D2 E0                           SHL     AL,CL           ; PUT IN HIGH ORDER NIBBLE
187F  8A 0E 0086 R                    MOV     CL,BYTE PTR VAR_DELAY ; GET DATA BYTE
1883  80 E1 0F                        AND     CL,0FH          ; CLEAR OLD VALUE IN HIGH NIBBLE
1886  0A C1                           OR      AL,CL           ; COMBINE HIGH AND LOW NIBBLES
1888  A2 0086 R                       MOV     BYTE PTR VAR_DELAY,AL ; PUT IN POSITION
188B  59                              POP     CX              ; RESTORE REGISTER
188C  C3                              RET
188D                          PUT_POS ENDP
                              ;--------------------------------------------------------------------------
                              ; MANUFACTURING ACTIVITY SIGNAL ROUTINE - INVOKED THROUGH THE TIMER
                              ; TICK ROUTINE DURING MANUFACTRUING ACTIVITIES . (ACCESSED THROUGH
                              ; INT 1CH)
                              ;--------------------------------------------------------------------------
188D                          MFG_TICK        PROC    FAR
188D  50                              PUSH    AX
188E  2B C0                           SUB     AX,AX           ; SEND A 00 TO PORT 13 AS A
                                                              ; ACTIVITY SIGNAL
1890  E6 13                           OUT     13H,AL
1892  E4 61                           IN      AL,PORT_B       ; FLIP SPEAKER DATA TO OPPOSITE
                                                              ; SENSE
1894  8A E0                           MOV     AH,AL           ; SAVE ORIG SETTING
1896  80 E4 9D                        AND     AH,10011101B    ; MAKE SURE MUX IS -> RIGHT AND
                                                              ; ISOLATE SPEAKER BIT
1899  F6 D0                           NOT     AL              ; FLIP ALL BITS
189B  24 02                           AND     AL,00000010B    ; ISOLATE SPEAKER DATA BIT (NOW IN
                                                              ; OPPOSITE SENSE)
189D  0A C4                           OR      AL,AH           ; COMBINE WITH ORIG. DATA FROM
                                                              ; PORT B
189F  0C 10                           OR      AL,00010000B    ; AND DISABLE INTERNAL SPEAKER
18A1  E6 61                           OUT     PORT_B,AL
18A3  B0 20                           MOV     AL,20H          ; EOI TO INTR. CHIP
18A5  E6 20                           OUT     20H,AL
18A7  58                              POP     AX
18A8  CF                              IRET
18A9                          MFG_TICK        ENDP
```

```
                        ;--------------------------------------------------------------------
                        ;       CONVERT AND PRINT ASCII CODE
                        ;
                        ;       AL MUST CONTAIN NUMBER TO BE CONVERTED.
                        ;       AX AND BX DESTROYED.
                        ;--------------------------------------------------------------------
18A9                    XPC_BYTE        PROC    NEAR
18A9  50                        PUSH    AX              ; SAVE FOR LOW NIBBLE DISPLAY
18AA  B1 04                     MOV     CL,4            ; SHIFT COUNT
18AC  D2 E8                     SHR     AL,CL           ; NIBBLE SWAP
18AE  E8 18B4 R                 CALL    XLAT_PR         ; DO THE HIGH NIBBLE DISPLAY
18B1  58                        POP     AX              ; RECOVER THE NIBBLE
18B2  24 0F                     AND     AL,0FH          ; ISOLATE TO LOW NIBBLE
                                                        ; FALL INTO LOW NIBBLE CONVERSION
18B4                    XLAT_PR PROC    NEAR            ; CONVERT 00-0F TO ASCII CHARACTER
18B4  04 90                     ADD     AL,090H         ; ADD FIRST CONVERSION FACTOR
18B6  27                        DAA                     ; ADJUST FOR NUMERIC AND ALPHA
                                                        ; RANGE
18B7  14 40                     ADC     AL,040H         ; ADD CONVERSION AND ADJUST LOW
                                                        ; NIBBLE
18B9  27                        DAA                     ; ADJUST HIGH NIBBLE TO ASCII RANGE
18BA                    PRT_HEX PROC    NEAR
18BA  53                        PUSH    BX
18BB  B4 0E                     MOV     AH,14           ; DISPLAY CHARACTER IN AL
18BD  B7 00                     MOV     BH,0
18BF  CD 10                     INT     10H             ; CALL VIDEO_IO
18C1  5B                        POP     BX
18C2  C3                        RET
18C3                    PRT_HEX ENDP
18C3                    XLAT_PR ENDP
18C3                    XPC_BYTE        ENDP
                        ;CONTROL IS PASSED HERE WHEN THERE ARE NO PARALLEL PRINTERS
                        ;ATTACHED.  CX HAS EQUIPMENT FLAG,DS POINTS AT DATA (40H)
                        ;DETERMINE WHICH RS232 CARD (0,1) TO USE
18C3                    REPRINT PROC NEAR
18C3  2B D2             B1_A:   SUB     DX,DX           ;ASSUME TO USE CARD 0
18C5  F6 C5 04                  TEST    CH,00000100B    ;UNLESS THERE ARE TWO CARDS
18C8  74 01                     JE      B10_1           ;IN WHICH CASE,
18CA  42                        INC     DX              ;USE CARD 1
                        ;DETERMINE WHICH FUNCTION IS BEING CALLED
18CB  0A E4             B10_1:  OR      AH,AH           ;TEST FOR AH = 0
18CD  74 41                     JZ      B12             ;GO PRINT CHAR
18CF  FE CC                     DEC     AH              ;TEST FOR AH = 1
18D1  74 1D                     JZ      B11             ;GO DO INIT
18D3  FE CC                     DEC     AH              ;TEST FOR AH = 2
18D5  75 16                     JNZ     SHORT B10_3     ;IF NOT VALID, RETURN
                                                        ;ELSE...
                        ;GET STATUS FROM RS232 PORT
18D7  50                        PUSH    AX              ;SAVE AL
18D8  B4 03                     MOV     AH,03H          ;USE THE GET COMMO PORT
18DA  CD 14                     INT     014H            ;STATUS FUNCTION OF INT14
18DC  E8 1925 R                 CALL    FAKE            ;FAKE WILL MAP ERROR BITS FROM
                                                        ;RS232 TO CORRESPONDING ONES
                                                        ;FOR THE PRINTER
18DF  58                        POP     AX              ;RESTORE AL
18E0  0A F6                     OR      DH,DH           ;CHECK IF ANY FLAGS WERE SET
18E2  74 07                     JZ      B10_2
18E4  8A E6                     MOV     AH,DH           ;MOVE FAKED ERROR CONDITION TO AH
18E6  80 E4 FE                  AND     AH,0FEH
18E9  EB 02                     JMP     SHORT B10_3     ;THEN RETURN
18EB  B4 90             B10_2:  MOV     AH,090H         ;MOVE IN STATUS FOR 'CORRECT'
                                                        ; RETURN
18ED  E9 F00D R         B10_3:  JMP     B1
                        ;INIT COMMO PORT    --- DX HAS WHICH CARD TO INIT.
                        ;MOVE TIME OUT VALUE FROM PRINTER TO RS232 TIME OUT VALUE
18F0  8B F2             B11:    MOV     SI,DX           ;SI GETS OFFSET INTO THE TABLE
18F2  A0 0078 R                 MOV     AL,PRINT_TIM_OUT
18F5  04 0A                     ADD     AL,0AH          ; INCREASE DELAY
18F7  88 84 007C R              MOV     RS232_TIM_OUT[SI],AL
18FB  50                        PUSH    AX              ;SAVE AL
18FC  B0 87                     MOV     AL,087H         ;SET INIT FOR:  1200 BAUD
                                                        ;               8 BIT WRD LNG
                                                        ;               NO PARITY
                                                        ;               2 STOP BITS
18FE  2A E4                     SUB     AH,AH           ;AH=0 IS COMMO INIT FUNCTION
1900  CD 14                     INT     014H            ;DO INIT
1902  E8 1925 R                 CALL    FAKE            ;FAKE WILL MAP ERROR BITS FROM
                                                        ;RS232 TO CORRESPONDING ONES
                                                        ;FOR THE PRINTER
1905  58                        POP     AX              ;RESTORE AL
1906  8A E6                     MOV     AH,DH           ;IF DH IS RETURNED ZERO, MEANING
1908  0A E4                     OR      AH,AH           ;NO ERRORS RETURN IT FOR THAT'S THE
                                                        ;'CORRECT' RETURN FROM AN ERROR
                                                        ; FREE INIT
190A  74 E1                     JE      B10_3
190C  B4 AB                     MOV     AH,0ABH
190E  EB DD                     JMP     SHORT B10_3     ;THEN RETURN
```

```
                        ;PRINT CHAR TO SERIAL PORT
                        ;DX = RS232 CARD TO BE USED: AL HAS CHAR TO BE PRINTED
1910  50                B12:    PUSH    AX              ;SAVE AL
1911  B4 01                     MOV     AH,01           ;1 IS SEND A CHAR DOWN COMMO LINE
1913  CD 14                     INT     014H            ;SEND THE CHAR
1915  E8 1925 R                 CALL    FAKE            ;FAKE WILL MAP ERROR BITS FROM
                                                        ;RS232 TO CORRESPONDING ONES
                                                        ;FOR THE PRINTER
1918  58                        POP     AX              ;RESTORE AL
1919  0A F6                     OR      DH,DH           ;SEE IF NO ERRORS WERE RETURNED
191B  74 04                     JZ      B12_1
191D  8A E6                     MOV     AH,DH           ;IF THERE WERE ERRORS, RETURN THEM
191F  EB CC                     JMP     SHORT B10_3     ;AND RETURN
1921  B4 10              B12_1:  MOV     AH,010H         ;PUT 'CORRECT' RETURN STATUS IN AH
1923  EB C8                     JMP     SHORT B10_3     ;AND RETURN
1925                    REPRINT ENDP
                        ;THIS PROC MAPS THE ERRORS RETURNED FROM A BIOS INT14 CALL
                        ;TO THOSE 'LIKE THAT' OF AN INT17 CALL
                        ;BREAK,FRAMING,PARITY,OVERRUN ERRORS ARE LOGGED AS I/O
                        ;ERRORS AND A TIME OUT IS MOVED TO THE APPROPIATE BIT
1925                    FAKE    PROC NEAR
1925  32 F6                     XOR     DH,DH           ;CLEAR FAKED STATUS FLAGS
1927  F6 C4 1E                  TEST    AH,011110B      ;CHECK FOR BREAK,FRAMING,PARITY
                                                        ;OVERRUN
192A  74 03                     JZ      B13_1           ;ERRORS. IF NOT THEN CHECK FOR
                                                        ; TIME OUT.
192C  B6 08                     MOV     DH,01000B       ;SET BIT 3 TO INDICATE 'I/O ERROR'
192E  C3                        RET                     ;AND RETURN
192F  F6 C4 80            B13_1:  TEST    AH,080H         ;TEST FOR TIME OUT ERROR RETURNED
1932  74 02                     JZ      B13_2           ;IF NOT TIME OUT, RETURN
1934  B6 09                     MOV     DH,09H          ;IF TIME OUT
1936  C3                  B13_2:  RET
1937                    FAKE    ENDP
                        ;--------------------------------------------------------------------
                        ;NEW_INT9
                        ;       THIS ROUTINE IS THE INTERRUPT 9 HANDLER WHEN THE MACHINE IS
                        ;       FIRST POWERED ON AND CASSETTE BASIC IS GIVEN CONTROL.  IT
                        ;       HANDLES THE FIRST KEYSTROKES ENTERED FROM THE KEYBOARD AND
                        ;       PERFORMS "SPECIAL" ACTIONS AS FOLLOWS:
                        ;               IF ESC IS THE FIRST KEY ENETERED MINI-WELCOME IS
                        ;                  EXECUTED
                        ;               IF CTRL-ESC IS THE FIRST SEQUENCE "LOAD CAS1:,R" IS
                        ;                  EXECUTED GIVING THE USER THE ABILITY TO BOOT
                        ;                  FROM CASSETTE
                        ;       AFTER THESE KEYSTROKES OR AFTER ANY OTHER KEYSTROKES THE
                        ;       INTERRUPT 9 VECTOR IS CHANGED TO POINT AT THE REAL
                        ;       INTERRUPT 9 ROUTINE.
                        ;--------------------------------------------------------------------
1937                    NEW_INT_9  PROC FAR
1937  3C 01                     CMP     AL,1            ; IS THIS AN ESCAPE KEY?
1939  74 10                     JE      ESC_KEY         ; JUMP IF AL=ESCAPE KEY
193B  3C 1D                     CMP     AL,29           ; ELSE, IS THIS A CONTROL KEY?
193D  74 06                     JE      CTRL_KEY        ; JUMP IF AL=CONTROL KEY
193F  E8 E01B R                 CALL    REAL_VECTOR_SETUP ; OTHERWISE, INITIALIZE REAL
                                                        ; INT 9 VECTOR
1942  CD 09                     INT     9H              ; PASS THE SCAN CODE IN AL
1944  CF                        IRET                    ; RETURN TO INTERRUPT 48H
1945                    CTRL_KEY:
1945  80 0E 0017 R 04           OR      KB_FLAG,04H     ; TURN ON CTRL SHIFT IN KB_FLAG
194A  CF                        IRET                    ; RETURN TO INTERRUPT
194B                    ESC_KEY:
194B  F6 06 0017 R 04           TEST    KB_FLAG,04H     ; HAS CONTROL SHIFT OCCURED?
1950  74 29                     JE      ESC_ONLY        ; NO.  ESCAPE ONLY
                        ;CONTROL ESCAPE HAS OCCURED, PUT MESSAGE IN BUFFER FOR CASSETTE
                        ; LOAD
1952  C6 06 0017 R 00           MOV     KB_FLAG,0       ; ZERO OUT CONTROL STATE
1957  1E                        PUSH    DS
1958  07                        POP     ES              ; INITIALIZE ES FOR BIOS DATA
1959  1E                        PUSH    DS              ; SAVE OLD DS
195A  0E                        PUSH    CS              ; POINT DS AT CODE SEGMENT
195B  1F                        POP     DS
195C  BE 1983 R                 MOV     SI,OFFSET CAS_LOAD ; GET MESSAGE
195F  BF 001E R                 MOV     DI,OFFSET KB_BUFFER ; POINT AT KEYBOARD BUFFER
1962  B9 000F 90                MOV     CX,CAS_LENGTH   ; LENGTH OF CASSETTE MESSAGE
1966  AC                  T_LOOP: LODSB                   ; GET ASCII CHARACTER FROM MESSAGE
1967  AB                        STOSW                   ; PUT IN KEYBOARD BUFFER
1968  E2 FC                     LOOP    T_LOOP
196A  1F                        POP     DS              ; RETRIEVE BIOS DATA SEGMENT
                        ;------ INITIALIZE QUEUE SO MESSAGE WILL BE REMOVED FROM BUFFER
196B  C7 06 001A R 001E R       MOV     BUFFER_HEAD,OFFSET KB_BUFFER
1971  C7 06 001C R 003C R       MOV     BUFFER_TAIL,OFFSET KB_BUFFER+(CAS_LENGTH*2)
                        ;--------------------------------------------------------------------
                        ;***NOTE***
                        ;       IT IS ASSUMED THAT THE LENGTH OF THE CASSETTE MESSAGE IS
                        ;       LESS THAN OR EQUAL TO THE LENGTH OF THE BUFFER.  IF THIS IS
                        ;       NOT THE CASE THE BUFFER WILL EVENTUALLY CONSUME MEMORY.
                        ;--------------------------------------------------------------------
1977  E8 E01B R                 CALL    REAL_VECTOR_SETUP
197A  CF                        IRET
197B                    ESC_ONLY:
197B  E8 E01B R                 CALL    REAL_VECTOR_SETUP
197E  B9 2000                   MOV     CX,MINI
1981  FF E1                     JMP     CX              ; ENTER THE WORLD OF KEYBOARD CAPER
                        ;------ MESSAGE FOR OUTPUT WHEN CONTROL-ESCAPE IS ENTERED AS FIRST
                        ;       KEY SEQUENCE
1983                    CAS_LOAD        LABEL   BYTE
1983  4C 4F 41 44 20 22         DB 'LOAD "CAS1:",R'
      43 41 53 31 3A 22
      2C 52
1991  0D                        DB  13
= 000F                  CAS_LENGTH EQU $ - CAS_LOAD
1992                    NEW_INT_9   ENDP
```

```
                        ;--------------------------------------------------
                        ; WRITE_TTY
                        ; THIS INTERFACE PROVIDES A TELETYPE LIKE INTERFACE TO THE
                        ;   VIDEO CARD.  THE INPUT CHARACTER IS WRITTEN TO THE CURRENT
                        ;   CURSOR POSITION, AND THE CURSOR IS MOVED TO THE NEXT POSITION.
                        ;   IF THE CURSOR LEAVES THE LAST COLUMN OF THE FIELD, THE COLUMN
                        ;   IS SET TO ZERO, AND THE ROW VALUE IS INCREMENTED.  IF THE ROW
                        ;   ROW VALUE LEAVES THE FIELD, THE CURSOR IS PLACED ON THE LAST
                        ;   ROW, FIRST COLUMN, AND THE ENTIRE SCREEN IS SCROLLED UP ONE
                        ;   LINE. WHEN THE SCREEN IS SCROLLED UP, THE ATTRIBUTE FOR FILLING
                        ;   THE NEWLY BLANKED LINE IS READ FROM THE CURSOR POSITION ON THE
                        ;   PREVIOUS LINE BEFORE THE SCROLL, IN CHARACTER MODE.  IN
                        ;   GRAPHICS MODE, THE 0 COLOR IS USED.
                        ; ENTRY --
                        ;   (AH) = CURRENT CRT MODE
                        ;   (AL) = CHARACTER TO BE WRITTEN
                        ;          NOTE THAT BACK SPACE, CAR RET, BELL AND LINE FEED ARE
                        ;          HANDLED AS COMMANDS RATHER THAN AS DISPLAYABLE GRAPHICS
                        ;   (BL) = FOREGROUND COLOR FOR CHAR WRITE IF CURRENTLY IN A
                        ;          GRAPHICS MODE
                        ; EXIT --
                        ;   ALL REGISTERS SAVED
                        ;--------------------------------------------------
                                ASSUME  CS:CODE,DS:DATA
1992                    WRITE_TTY       PROC    NEAR
1992  50                        PUSH    AX              ; SAVE REGISTERS
1993  50                        PUSH    AX              ; SAVE CHAR TO WRITE
1994  8A 3E 0062 R              MOV     BH,ACTIVE_PAGE  ; GET CURRENT PAGE SETTING
1998  53                        PUSH    BX              ; SAVE IT
1999  8A DF                     MOV     BL,BH           ; IN BL
199B  32 FF                     XOR     BH,BH
199D  D1 E3                     SAL     BX,1            ; CONVERT TO WORD OFFSET
199F  8B 97 0050 R              MOV     DX,[BX+OFFSET CURSOR_POSN] ; GET CURSOR POSITION
19A3  5B                        POP     BX              ; RECOVER CURRENT PAGE
19A4  58                        POP     AX              ; RECOVER CHAR
                        ;----- DX NOW HAS THE CURRENT CURSOR POSITION
19A5  3C 08                     CMP     AL,8            ; IS IT A BACKSPACE?
19A7  74 50                     JE      U8              ; BACK_SPACE
19A9  3C 0D                     CMP     AL,0DH          ; IS IT A CARRIAGE RETURN?
19AB  74 54                     JE      U9              ; CAR_RET
19AD  3C 0A                     CMP     AL,0AH          ; IS IT A LINE FEED
19AF  74 15                     JE      U10             ; LINE_FEED
19B1  3C 07                     CMP     AL,07H          ; IS IT A BELL
19B3  74 50                     JE      U11             ; BELL
                        ;----- WRITE THE CHAR TO THE SCREEN
19B5  B4 0A                     MOV     AH,10           ; WRITE CHAR ONLY
19B7  B9 0001                   MOV     CX,1            ; ONLY ONE CHAR
19BA  CD 10                     INT     10H             ; WRITE THE CHAR
                        ;----- POSITION THE CURSOR FOR NEXT CHAR
19BC  FE C2                     INC     DL
19BE  3A 16 004A R              CMP     DL,BYTE PTR CRT_COLS ; TEST FOR COLUMN OVERFLOW
19C2  75 31                     JNZ     U7              ; SET_CURSOR
19C4  32 D2                     XOR     DL,DL           ; COLUMN FOR CURSOR
                        ;---- LINE FEED
19C6                    U10:
19C6  80 FE 18                  CMP     DH,24
19C9  75 28                     JNZ     U6              ; SET_CURSOR_INC
                        ;----- SCROLL REQUIRED
19CB  B4 02                     MOV     AH,2
19CD  CD 10                     INT     10H             ; SET THE CURSOR
                        ;----- DETERMINE VALUE TO FILL WITH DURING SCROLL
19CF  A0 0049 R                 MOV     AL,CRT_MODE     ; GET THE CURRENT MODE
19D2  3C 04                     CMP     AL,4
19D4  72 04                     JC      U2              ; READ-CURSOR
19D6  32 FF                     XOR     BH,BH           ; FILL WITH BACKGROUND
19D8  EB 06                     JMP     SHORT U3        ; SCROLL-UP
19DA  B4 08             U2:     MOV     AH,8
19DC  CD 10                     INT     10H             ; READ CHAR/ATTR AT CURRENT CURSOR
19DE  8A FC                     MOV     BH,AH           ; STORE IN BH
19E0  B8 0601           U3:     MOV     AX,601H         ; SCROLL ONE LINE
19E3  2B C9                     SUB     CX,CX           ; UPPER LEFT CORNER
19E5  B6 18                     MOV     DH,24           ; LOWER RIGHT ROW
19E7  8A 16 004A R              MOV     DL,BYTE PTR CRT_COLS ; LOWER RIGHT COLUMN
19EB  FE CA                     DEC     DL
19ED  CD 10             U4:     INT     10H             ; SCROLL UP THE SCREEN
19EF  58                U5:     POP     AX              ; RESTORE THE CHARACTER
19F0  E9 0F70 R                 JMP     VIDEO_RETURN    ; RETURN TO CALLER
19F3  FE C6             U6:     INC     DH              ; NEXT ROW
19F5  B4 02             U7:     MOV     AH,2
19F7  EB F4                     JMP     U4              ; ESTABLISH THE NEW CURSOR
                        ;----- BACK SPACE FOUND
19F9  0A D2             U8:     OR      DL,DL           ; ALREADY AT END OF LINE
19FB  74 F8                     JE      U7              ; SET_CURSOR
19FD  FE CA                     DEC     DL              ; NO -- JUST MOVE IT BACK
19FF  EB F4                     JMP     U7              ; SET_CURSOR
                        ;----- CARRIAGE RETURN FOUND
1A01  32 D2             U9:     XOR     DL,DL           ; MOVE TO FIRST COLUMN
1A03  EB F0                     JMP     U7              ; SET_CURSOR
                        ;----- BELL FOUND
1A05  B3 02             U11:    MOV     BL,2            ; SET UP COUNT FOR BEEP
1A07  E8 FF31 R                 CALL    BEEP            ; SOUND THE POD BELL
1A0A  EB E3                     JMP     U5              ; TTY_RETURN
1A0C                    WRITE_TTY       ENDP
```

```
                        ;--------------------------------------------------------------------------
                        ;       THIS PROCEDURE WILL ISSUE SHORT TONES TO INDICATE FAILURES
                        ;       THAT  1: OCCUR BEFORE THE CRT IS STARTED, 2: TO CALL THE
                        ;       OPERATORS ATTENTION TO AN ERROR AT THE END OF POST, OR
                        ;       3: TO SIGNAL THE SUCCESSFUL COMPLETION OF POST
                        ; ENTRY PARAMETERS:
                        ;       DL = NUMBER OF APPROX. 1/2 SEC TONES TO SOUND
                        ;--------------------------------------------------------------------------
1A0C                    ERR_BEEP  PROC  NEAR
1A0C  9C                        PUSHF                   ; SAVE FLAGS
1A0D  53                        PUSH    BX
1A0E  FA                        CLI                     ; DISABLE SYSTEM INTERRUPTS
1A0F                    G3:                             ; SHORT_BEEP:
1A0F  B3 01                     MOV     BL,1            ; COUNTER FOR A SHORT BEEP
1A11  E8 FF31 R                 CALL    BEEP            ; DO THE SOUND
1A14  E2 FE             G4:     LOOP    G4              ; DELAY BETWEEN BEEPS
1A16  FE CA                     DEC     DL              ; DONE WITH SHORTS
1A18  75 F5                     JNZ     G3              ; DO SOME MORE
1A1A  E2 FE             G5:     LOOP    G5              ; LONG DELAY BEFORE RETURN
1A1C  E2 FE             G6:     LOOP    G6
1A1E  5B                        POP     BX              ; RESTORE ORIG CONTENTS OF BX
1A1F  9D                        POPF                    ; RESTORE FLAGS TO ORIG SETTINGS
1A20  C3                        RET                     ; RETURN TO CALLER
1A21                    ERR_BEEP        ENDP
                         LIST
                                ASSUME  CS:CODE,DS:DATA
E000                            ORG     0E000H
E000  31 35 30 34 30 33         DB      '1504037 COPR. IBM 1981,1983'  ; COPYRIGHT NOTICE
      37 20 43 4F 50 52
      2E 20 49 42 4D 20
      31 39 38 31 2C 31
      39 38 33


                        ;--------------------------------------------------------------------------
                        ;REAL_VECTOR_SETUP
                        ;
                        ;       THIS ROUTINE WILL INITIALIZE THE INTERRUPT 9 VECTOR TO
                        ;       POINT AT THE REAL INTERRUPT ROUTINE.
                        ;--------------------------------------------------------------------------
E01B                    REAL_VECTOR_SETUP       PROC      NEAR
E01B  50                        PUSH    AX              ; SAVE THE SCAN CODE
E01C  53                        PUSH    BX
E01D  06                        PUSH    ES
E01E  33 C0                     XOR     AX,AX           ; INITIALIZE TO POINT AT VECTOR
                                                        ; SECTOR(0)
E020  8E C0                     MOV     ES,AX
E022  BB 0024                   MOV     BX,9H*4H        ; POINT AT INTERRUPT 9
E025  26: C7 07 1561 R          MOV     WORD PTR ES:[BX],OFFSET KB_INT ; MOVE IN OFFSET OF
                                                        ; ROUTINE
E02A  43                        INC     BX              ; ADD 2 TO BX
E02B  43                        INC     BX
E02C  0E                        PUSH    CS              ; GET CODE SEGMENT OF BIOS (SEGMENT
                                                        ; RELOCATEABLE)
E02D  58                        POP     AX
E02E  26: 89 07                 MOV     WORD PTR ES:[BX],AX ; MOVE IN SEGMENT OF ROUTINE
E031  07                        POP     ES
E032  5B                        POP     BX
E033  58                        POP     AX
E034  C3                        RET
E035                    REAL_VECTOR_SETUP       ENDP
                        ;--------------------------------------------------------------------------
                        ;KB_NOISE
                        ;       THIS ROUTINE IS CALLED WHEN GENERAL BEEPS ARE REQUIRED FROM
                        ;       THE SYSTEM.
                        ;INPUT
                        ;       BX=LENGH OF THE TONE
                        ;       CX=CONTAINS THE FREQUENCY
                        ;OUTPUT
                        ;       ALL REGISTERS ARE MAINTAINED.
                        ;HINTS
                        ;       AS CX GETS LARGER THE TONE PRODUCED GETS LOWER IN PITCH.
                        ;
                        ;--------------------------------------------------------------------------
E035                    KB_NOISE        PROC   NEAR
E035  FB                        STI
E036  50                        PUSH   AX
E037  53                        PUSH   BX
E038  51                        PUSH   CX
E039  E4 61                     IN     AL,061H          ; GET CONTROL INFO
E03B  50                        PUSH   AX               ; SAVE
E03C                    LOOP01:
E03C  24 FC                     AND    AL,0FCH          ; TURN OFF TIMER GATE AND SPEAKER
                                                        ; DATA
E03E  E6 61                     OUT    061H,AL          ; OUTPUT TO CONTROL
E040  51                        PUSH   CX               ; HALF CYCLE TIME FOR TONE
E041  E2 FE             LOOP02: LOOP   LOOP02           ; SPEAKER OFF
E043  0C 02                     OR     AL,2             ; TURN ON SPEAKER BIT
E045  E6 61                     OUT    061H,AL          ; OUTPUT TO CONTROL
E047  59                        POP    CX
E048  51                        PUSH   CX               ; RETRIEVE FREQUENCY
E049  E2 FE             LOOP03: LOOP   LOOP03           ; ANOTHER HALF CYCLE
E04B  4B                        DEC    BX               ; TOTAL TIME COUNT
E04C  59                        POP    CX               ; RETRIEVE  FREQ.
E04D  75 ED                     JNZ    LOOP01           ; DO ANOTHER CYCLE
E04F  58                        POP    AX               ; RECOVER CONTROL
E050  E6 61                     OUT    061H,AL          ; OUTPUT THE CONTROL
E052  59                        POP    CX
E053  5B                        POP    BX
E054  58                        POP    AX
E055  C3                        RET
E056                    KB_NOISE        ENDP
E058                            ORG    0E05BH
E05B  E9 0043 R                 JMP    NEAR PTR RESET
```

```
                        ;--------------------------------------------------------
                        ;       CHARACTER GENERATOR GRAPHICS FOR 320X200 AND 640X200
                        ;       GRAPHICS FOR CHARACTERS 80H THROUGH FFH
                        ;--------------------------------------------------------

E05E                    CRT_CHARH       LABEL   BYTE
E05E  78 CC C0 CC 78 18                 DB      078H,0CCH,0C0H,0CCH,078H,018H,00CH,078H ; D_80
      0C 78
E066  00 CC 00 CC CC CC                 DB      000H,0CCH,000H,0CCH,0CCH,0CCH,07EH,000H ; D_81
      7E 00
E06E  1C 00 78 CC FC C0                 DB      01CH,000H,078H,0CCH,0FCH,0C0H,078H,000H ; D_82
      78 00
E076  7E C3 3C 06 3E 66                 DB      07EH,0C3H,03CH,006H,03EH,066H,03FH,000H ; D_83
      3F 00
E07E  CC 00 78 0C 7C CC                 DB      0CCH,000H,078H,00CH,07CH,0CCH,07EH,000H ; D_84
      7E 00
E086  E0 00 78 0C 7C CC                 DB      0E0H,000H,078H,00CH,07CH,0CCH,07EH,000H ; D_85
      7E 00
E08E  30 30 78 0C 7C CC                 DB      030H,030H,078H,00CH,07CH,0CCH,07EH,000H ; D_86
      7E 00
E096  00 00 78 C0 C0 78                 DB      000H,000H,078H,0C0H,0C0H,078H,00CH,038H ; D_87
      0C 38
E09E  7E C3 3C 66 7E 60                 DB      07EH,0C3H,03CH,066H,07EH,060H,03CH,000H ; D_88
      3C 00
E0A6  CC 00 78 CC FC C0                 DB      0CCH,000H,078H,0CCH,0FCH,0C0H,078H,000H ; D_89
      78 00
E0AE  E0 00 78 CC FC C0                 DB      0E0H,000H,078H,0CCH,0FCH,0C0H,078H,000H ; D_8A
      78 00
E0B6  CC 00 70 30 30 30                 DB      0CCH,000H,070H,030H,030H,030H,078H,000H ; D_8B
      78 00
E0BE  7C C6 38 18 18 18                 DB      07CH,0C6H,038H,018H,018H,018H,03CH,000H ; D_8C
      3C 00
E0C6  E0 00 70 30 30 30                 DB      0E0H,000H,070H,030H,030H,030H,078H,000H ; D_8D
      78 00
E0CE  C6 38 6C C6 FE C6                 DB      0C6H,038H,06CH,0C6H,0FEH,0C6H,0C6H,000H ; D_8E
      C6 00
E0D6  30 30 00 78 CC FC                 DB      030H,030H,000H,078H,0CCH,0FCH,0CCH,000H ; D_8F
      CC 00

E0DE  1C 00 FC 60 78 60                 DB      01CH,000H,0FCH,060H,078H,060H,0FCH,000H ; D_90
      FC 00
E0E6  00 00 7F 0C 7F CC                 DB      000H,000H,07FH,00CH,07FH,0CCH,07FH,000H ; D_91
      7F 00
E0EE  3E 6C CC FE CC CC                 DB      03EH,06CH,0CCH,0FEH,0CCH,0CCH,0CEH,000H ; D_92
      CE 00
E0F6  78 CC 00 78 CC CC                 DB      078H,0CCH,000H,078H,0CCH,0CCH,078H,000H ; D_93
      78 00
E0FE  00 CC 00 78 CC CC                 DB      000H,0CCH,000H,078H,0CCH,0CCH,078H,000H ; D_94
      78 00
E106  00 E0 00 78 CC CC                 DB      000H,0E0H,000H,078H,0CCH,0CCH,078H,000H ; D_95
      78 00
E10E  78 CC 00 CC CC CC                 DB      078H,0CCH,000H,0CCH,0CCH,0CCH,07EH,000H ; D_96
      7E 00
E116  00 E0 00 CC CC CC                 DB      000H,0E0H,000H,0CCH,0CCH,0CCH,07EH,000H ; D_97
      7E 00
E11E  00 CC 00 CC CC 7C                 DB      000H,0CCH,000H,0CCH,0CCH,07CH,00CH,0F8H ; D_98
      0C F8
E126  C3 18 3C 66 66 3C                 DB      0C3H,018H,03CH,066H,066H,03CH,018H,000H ; D_99
      18 00
E12E  CC 00 CC CC CC CC                 DB      0CCH,000H,0CCH,0CCH,0CCH,0CCH,078H,000H ; D_9A
      78 00
E136  18 18 7E C0 C0 7E                 DB      018H,018H,07EH,0C0H,0C0H,07EH,018H,018H ; D_9B
      18 18
E13E  38 6C 64 F0 60 E6                 DB      038H,06CH,064H,0F0H,060H,0E6H,0FCH,000H ; D_9C
      FC 00
E146  CC CC 78 FC 30 FC                 DB      0CCH,0CCH,078H,0FCH,030H,0FCH,030H,030H ; D_9D
      30 30
E14E  F8 CC CC FA C6 CF                 DB      0F8H,0CCH,0CCH,0FAH,0C6H,0CFH,0C6H,0C7H ; D_9E
      C6 C7
E156  0E 1B 18 3C 18 18                 DB      00EH,01BH,018H,03CH,018H,018H,0D8H,070H ; D_9F
      D8 70

E15E  1C 00 78 0C 7C CC                 DB      01CH,000H,078H,00CH,07CH,0CCH,07EH,000H ; D_A0
      7E 00
E166  38 00 70 30 30 30                 DB      038H,000H,070H,030H,030H,030H,078H,000H ; D_A1
      78 00
E16E  00 1C 00 78 CC CC                 DB      000H,01CH,000H,078H,0CCH,0CCH,078H,000H ; D_A2
      78 00
E176  00 1C 00 CC CC CC                 DB      000H,01CH,000H,0CCH,0CCH,0CCH,07EH,000H ; D_A3
      7E 00
E17E  00 F8 00 F8 CC CC                 DB      000H,0F8H,000H,0F8H,0CCH,0CCH,0CCH,000H ; D_A4
      CC 00
E186  FC 00 CC EC FC DC                 DB      0FCH,000H,0CCH,0ECH,0FCH,0DCH,0CCH,000H ; D_A5
      CC 00
E18E  3C 6C 6C 3E 00 7E                 DB      03CH,06CH,06CH,03EH,000H,07EH,000H,000H ; D_A6
      00 00
E196  38 6C 6C 38 00 7C                 DB      038H,06CH,06CH,038H,000H,07CH,000H,000H ; D_A7
      00 00
E19E  30 00 30 60 C0 CC                 DB      030H,000H,030H,060H,0C0H,0CCH,078H,000H ; D_A8
      78 00
E1A6  00 00 00 FC C0 C0                 DB      000H,000H,000H,0FCH,0C0H,0C0H,000H,000H ; D_A9
      00 00
E1AE  00 00 00 FC 0C 0C                 DB      000H,000H,000H,0FCH,00CH,00CH,000H,000H ; D_AA
      00 00
E1B6  C3 C6 CC DE 33 66                 DB      0C3H,0C6H,0CCH,0DEH,033H,066H,0CCH,00FH ; D_AB
      CC 0F
E1BE  C3 C6 CC DB 37 6F                 DB      0C3H,0C6H,0CCH,0DBH,037H,06FH,0CFH,003H ; D_AC
      CF 03
E1C6  18 18 00 18 18 18                 DB      018H,018H,000H,018H,018H,018H,018H,000H ; D_AD
      18 00
E1CE  00 33 66 CC 66 33                 DB      000H,033H,066H,0CCH,066H,033H,000H,000H ; D_AE
      00 00
E1D6  00 CC 66 33 66 CC                 DB      000H,0CCH,066H,033H,066H,0CCH,000H,000H ; D_AF
      00 00
```

```
E1DE  22 88 22 88 22 88                DB      022H,088H,022H,088H,022H,088H,022H,088H  ; D_B0
      22 88
E1E6  55 AA 55 AA 55 AA                DB      055H,0AAH,055H,0AAH,055H,0AAH,055H,0AAH  ; D_B1
      55 AA
E1EE  DB 77 DB EE DB 77                DB      0DBH,077H,0DBH,0EEH,0DBH,077H,0DBH,0EEH  ; D_B2
      DB EE
E1F6  18 18 18 18 18 18                DB      018H,018H,018H,018H,018H,018H,018H,018H  ; D_B3
      18 18
E1FE  18 18 18 18 F8 18                DB      018H,018H,018H,018H,0F8H,018H,018H,018H  ; D_B4
      18 18
E206  18 18 F8 18 F8 18                DB      018H,018H,0F8H,018H,0F8H,018H,018H,018H  ; D_B5
      18 18
E20E  36 36 36 36 F6 36                DB      036H,036H,036H,036H,0F6H,036H,036H,036H  ; D_B6
      36 36
E216  00 00 00 00 FE 36                DB      000H,000H,000H,000H,0FEH,036H,036H,036H  ; D_B7
      36 36
E21E  00 00 F8 18 F8 18                DB      000H,000H,0F8H,018H,0F8H,018H,018H,018H  ; D_B8
      18 18
E226  36 36 F6 06 F6 36                DB      036H,036H,0F6H,006H,0F6H,036H,036H,036H  ; D_B9
      36 36
E22E  36 36 36 36 36 36                DB      036H,036H,036H,036H,036H,036H,036H,036H  ; D_BA
      36 36
E236  00 00 FE 06 F6 36                DB      000H,000H,0FEH,006H,0F6H,036H,036H,036H  ; D_BB
      36 36
E23E  36 36 F6 06 FE 00                DB      036H,036H,0F6H,006H,0FEH,000H,000H,000H  ; D_BC
      00 00
E246  36 36 36 36 FE 00                DB      036H,036H,036H,036H,0FEH,000H,000H,000H  ; D_BD
      00 00
E24E  18 18 F8 18 F8 00                DB      018H,018H,0F8H,018H,0F8H,000H,000H,000H  ; D_BE
      00 00
E256  00 00 00 00 F8 18                DB      000H,000H,000H,000H,0F8H,018H,018H,018H  ; D_BF
      18 18

E25E  18 18 18 18 1F 00                DB      018H,018H,018H,018H,01FH,000H,000H,000H  ; D_C0
      00 00
E266  18 18 18 18 FF 00                DB      018H,018H,018H,018H,0FFH,000H,000H,000H  ; D_C1
      00 00
E26E  00 00 00 00 FF 18                DB      000H,000H,000H,000H,0FFH,018H,018H,018H  ; D_C2
      18 18
E276  18 18 18 18 1F 18                DB      018H,018H,018H,018H,01FH,018H,018H,018H  ; D_C3
      18 18
E27E  00 00 00 00 FF 00                DB      000H,000H,000H,000H,0FFH,000H,000H,000H  ; D_C4
      00 00
E286  18 18 18 18 FF 18                DB      018H,018H,018H,018H,0FFH,018H,018H,018H  ; D_C5
      18 18
E28E  18 18 1F 18 1F 18                DB      018H,018H,01FH,018H,01FH,018H,018H,018H  ; D_C6
      18 18
E296  36 36 36 36 37 36                DB      036H,036H,036H,036H,037H,036H,036H,036H  ; D_C7
      36 36
E29E  36 36 37 30 3F 00                DB      036H,036H,037H,030H,03FH,000H,000H,000H  ; D_C8
      00 00
E2A6  00 00 3F 30 37 36                DB      000H,000H,03FH,030H,037H,036H,036H,036H  ; D_C9
      36 36
E2AE  36 36 F7 00 FF 00                DB      036H,036H,0F7H,000H,0FFH,000H,000H,000H  ; D_CA
      00 00
E2B6  00 00 FF 00 F7 36                DB      000H,000H,0FFH,000H,0F7H,036H,036H,036H  ; D_CB
      36 36
E2BE  36 36 37 30 37 36                DB      036H,036H,037H,030H,037H,036H,036H,036H  ; D_CC
      36 36
E2C6  00 00 FF 00 FF 00                DB      000H,000H,0FFH,000H,0FFH,000H,000H,000H  ; D_CD
      00 00
E2CE  36 36 F7 00 F7 36                DB      036H,036H,0F7H,000H,0F7H,036H,036H,036H  ; D_CE
      36 36
E2D6  18 18 FF 00 FF 00                DB      018H,018H,0FFH,000H,0FFH,000H,000H,000H  ; D_CF
      00 00

E2DE  36 36 36 36 FF 00                DB      036H,036H,036H,036H,0FFH,000H,000H,000H  ; D_D0
      00 00
E2E6  00 00 FF 00 FF 18                DB      000H,000H,0FFH,000H,0FFH,018H,018H,018H  ; D_D1
      18 18
E2EE  00 00 00 00 FF 36                DB      000H,000H,000H,000H,0FFH,036H,036H,036H  ; D_D2
      36 36
E2F6  36 36 36 36 3F 00                DB      036H,036H,036H,036H,03FH,000H,000H,000H  ; D_D3
      00 00
E2FE  18 18 1F 18 1F 00                DB      018H,018H,01FH,018H,01FH,000H,000H,000H  ; D_D4
      00 00
E306  00 00 1F 18 1F 18                DB      000H,000H,01FH,018H,01FH,018H,018H,018H  ; D_D5
      18 18
E30E  00 00 00 00 3F 36                DB      000H,000H,000H,000H,03FH,036H,036H,036H  ; D_D6
      36 36
E316  36 36 36 36 FF 36                DB      036H,036H,036H,036H,0FFH,036H,036H,036H  ; D_D7
      36 36
E31E  18 18 FF 18 FF 18                DB      018H,018H,0FFH,018H,0FFH,018H,018H,018H  ; D_D8
      18 18
E326  18 18 18 18 F8 00                DB      018H,018H,018H,018H,0F8H,000H,000H,000H  ; D_D9
      00 00
E32E  00 00 00 00 1F 18                DB      000H,000H,000H,000H,01FH,018H,018H,018H  ; D_DA
      18 18
E336  FF FF FF FF FF FF                DB      0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH  ; D_DB
      FF FF
E33E  00 00 00 00 FF FF                DB      000H,000H,000H,000H,0FFH,0FFH,0FFH,0FFH  ; D_DC
      FF FF
E346  F0 F0 F0 F0 F0 F0                DB      0F0H,0F0H,0F0H,0F0H,0F0H,0F0H,0F0H,0F0H  ; D_DD
      F0 F0
E34E  0F 0F 0F 0F 0F 0F                DB      00FH,00FH,00FH,00FH,00FH,00FH,00FH,00FH  ; D_DE
      0F 0F
E356  FF FF FF FF 00 00                DB      0FFH,0FFH,0FFH,0FFH,000H,000H,000H,000H  ; D_DF
      00 00
```

```
E35E  00 00 76 DC C8 DC                DB      000H,000H,076H,0DCH,0C8H,0DCH,076H,000H  ; D_E0
      76 00
E366  00 78 CC F8 CC F8                DB      000H,078H,0CCH,0F8H,0CCH,0F8H,0C0H,0C0H  ; D_E1
      C0 C0
E36E  00 FC CC C0 C0 C0                DB      000H,0FCH,0CCH,0C0H,0C0H,0C0H,0C0H,000H  ; D_E2
      C0 00
E376  00 FE 6C 6C 6C 6C                DB      000H,0FEH,06CH,06CH,06CH,06CH,06CH,000H  ; D_E3
      6C 00
E37E  FC CC 60 30 60 CC                DB      0FCH,0CCH,060H,030H,060H,0CCH,0FCH,000H  ; D_E4
      FC 00
E386  00 00 7E DB DB DB                DB      000H,000H,07EH,0DBH,0DBH,0DBH,070H,000H  ; D_E5
      70 00
E38E  00 66 66 66 66 7C                DB      000H,066H,066H,066H,066H,07CH,060H,0C0H  ; D_E6
      60 C0
E396  00 76 DC 18 18 18                DB      000H,076H,0DCH,018H,018H,018H,018H,000H  ; D_E7
      18 00
E39E  FC 30 78 CC CC 78                DB      0FCH,030H,078H,0CCH,0CCH,078H,030H,0FCH  ; D_E8
      30 FC
E3A6  38 6C C6 FE C6 6C                DB      038H,06CH,0C6H,0FEH,0C6H,06CH,038H,000H  ; D_E9
      38 00
E3AE  38 6C C6 C6 6C 6C                DB      038H,06CH,0C6H,0C6H,06CH,06CH,0EEH,000H  ; D_EA
      EE 00
E3B6  1C 30 18 7C CC CC                DB      01CH,030H,018H,07CH,0CCH,0CCH,078H,000H  ; D_EB
      78 00
E3BE  00 00 7E DB DB 7E                DB      000H,000H,07EH,0DBH,0DBH,07EH,000H,000H  ; D_EC
      00 00
E3C6  06 0C 7E DB DB 7E                DB      006H,00CH,07EH,0DBH,0DBH,07EH,060H,0C0H  ; D_ED
      60 C0
E3CE  38 60 C0 F8 C0 60                DB      038H,060H,0C0H,0F8H,0C0H,060H,038H,000H  ; D_EE
      38 00
E3D6  78 CC CC CC CC CC                DB      078H,0CCH,0CCH,0CCH,0CCH,0CCH,0CCH,000H  ; D_EF
      CC 00

E3DE  00 FC 00 FC 00 FC                DB      000H,0FCH,000H,0FCH,000H,0FCH,000H,000H  ; D_F0
      00 00
E3E6  30 30 FC 30 30 00                DB      030H,030H,0FCH,030H,030H,000H,0FCH,000H  ; D_F1
      FC 00
E3EE  60 30 18 30 60 00                DB      060H,030H,018H,030H,060H,000H,0FCH,000H  ; D_F2
      FC 00
E3F6  18 30 60 30 18 00                DB      018H,030H,060H,030H,018H,000H,0FCH,000H  ; D_F3
      FC 00
E3FE  0E 1B 1B 18 18 18                DB      00EH,01BH,01BH,018H,018H,018H,018H,018H  ; D_F4
      18 18
E406  18 18 18 18 18 D8                DB      018H,018H,018H,018H,018H,0D8H,0D8H,070H  ; D_F5
      D8 70
E40E  30 30 00 FC 00 30                DB      030H,030H,000H,0FCH,000H,030H,030H,000H  ; D_F6
      30 00
E416  00 76 DC 00 76 DC                DB      000H,076H,0DCH,000H,076H,0DCH,000H,000H  ; D_F7
      00 00
E41E  38 6C 6C 38 00 00                DB      038H,06CH,06CH,038H,000H,000H,000H,000H  ; D_F8
      00 00
E426  00 00 00 18 18 00                DB      000H,000H,000H,018H,018H,000H,000H,000H  ; D_F9
      00 00
E42E  00 00 00 00 18 00                DB      000H,000H,000H,000H,018H,000H,000H,000H  ; D_FA
      00 00
E436  0F 0C 0C 0C EC 6C                DB      00FH,00CH,00CH,00CH,0ECH,06CH,03CH,01CH  ; D_FB
      3C 1C
E43E  78 6C 6C 6C 6C 00                DB      078H,06CH,06CH,06CH,06CH,000H,000H,000H  ; D_FC
      00 00
E446  70 18 30 60 78 00                DB      070H,018H,030H,060H,078H,000H,000H,000H  ; D_FD
      00 00
E44E  00 00 3C 3C 3C 3C                DB      000H,000H,03CH,03CH,03CH,03CH,000H,000H  ; D_FE
      00 00
E456  00 00 00 00 00 00                DB      000H,000H,000H,000H,000H,000H,000H,000H  ; D_FF
      00 00

                                       ASSUME  CS:CODE,DS:DATA
                                ;--------------------------------------------------
                                ; SET_CTYPE
                                ;       THIS ROUTINE SETS THE CURSOR VALUE
                                ; INPUT
                                ;       (CX) HAS CURSOR VALUE CH-START LINE, CL-STOP LINE
                                ; OUTPUT
                                ;       NONE
                                ;--------------------------------------------------
E45E                            SET_CTYPE       PROC    NEAR
E45E  80 FC 04                          CMP     AH,4            ; IN GRAPHICS MODE?
E461  72 03                             JC      C23X            ; NO, JUMP
E463  80 CD 20                          OR      CH,20H          ; YES, DISABLE CURSOR
E466  B4 0A                     C23X:   MOV     AH,10           ; 6845 REGISTER FOR CURSOR SET
E468  89 0E 0060 R                      MOV     CURSOR_MODE,CX  ; SAVE IN DATA AREA
E46C  E8 E472 R                         CALL    C23             ; OUTPUT CX REG
E46F  E9 0F70 R                         JMP     VIDEO_RETURN
                                ;THIS ROUTINE OUTPUTS THE CX REGISTER TO THE 6845 REGS NAMED IN AH
E472  8B 16 0063 R              C23:    MOV     DX,ADDR_6845    ; ADDRESS REGISTER
E476  8A C4                             MOV     AL,AH           ; GET VALUE
E478  EE                                OUT     DX,AL           ; REGISTER SET
E479  42                                INC     DX              ; DATA REGISTER
E47A  8A C5                             MOV     AL,CH           ; DATA
E47C  EE                                OUT     DX,AL
E47D  4A                                DEC     DX
E47E  8A C4                             MOV     AL,AH
E480  FE C0                             INC     AL              ; POINT TO OTHER DATA REGISTER
E482  EE                                OUT     DX,AL           ; SET FOR SECOND REGISTER
E483  42                                INC     DX
E484  8A C1                             MOV     AL,CL           ; SECOND DATA VALUE
E486  EE                                OUT     DX,AL
E487  C3                                RET                     ; ALL DONE
E488                            SET_CTYPE       ENDP
```

```
                        ;--------------------------------------------
                        ; SET_CPOS
                        ;       THIS ROUTINE SETS THE CURRENT CURSOR POSITION TO THE
                        ;       NEW X-Y VALUES PASSED
                        ; INPUT
                        ;       DX - ROW,COLUMN OF NEW CURSOR
                        ;       BH - DISPLAY PAGE OF CURSOR
                        ; OUTPUT
                        ;       CURSOR IS SET AT 6845 IF DISPLAY PAGE IS CURRENT DISPLAY
                        ;--------------------------------------------
E488                    SET_CPOS        PROC    NEAR
E488  8A CF                     MOV     CL,BH
E48A  32 ED                     XOR     CH,CH                   ; ESTABLISH LOOP COUNT
E48C  D1 E1                     SAL     CX,1                    ; WORD OFFSET
E48E  8B F1                     MOV     SI,CX                   ; USE INDEX REGISTER
E490  89 94 0050 R              MOV     [SI+OFFSET CURSOR_POSN],DX ; SAVE THE POINTER
E494  38 3E 0062 R              CMP     ACTIVE_PAGE,BH
E498  75 05                     JNZ     C24                     ; SET_CPOS_RETURN
E49A  8B C2                     MOV     AX,DX                   ; GET ROW/COLUMN TO AX
E49C  E8 E4A2 R                 CALL    C25                     ; CURSOR_SET
E49F  E9 0F70 R         C24:    JMP     VIDEO_RETURN
E4A2                    SET_CPOS        ENDP
                        ;------ SET CURSOR POSITION, AX HAS ROW/COLUMN FOR CURSOR
E4A2                    C25     PROC    NEAR
E4A2  E8 E5C2 R                 CALL    POSITION                ; DETERMINE LOCATION IN REGEN
                                                                ; BUFFER
E4A5  8B C8                     MOV     CX,AX
E4A7  03 0E 004E R              ADD     CX,CRT_START            ; ADD IN THE START ADDRESS FOR THIS
                                                                ; PAGE
E4AB  D1 F9                     SAR     CX,1                    ; DIVIDE BY 2 FOR CHAR ONLY COUNT
E4AD  B4 0E                     MOV     AH,14                   ; REGISTER NUMBER FOR CURSOR
E4AF  E8 E472 R                 CALL    C23                     ; OUTPUT THE VALUE TO THE 6845
E4B2  C3                        RET
E4B3                    C25     ENDP
                        ;--------------------------------------------------
                        ; ACT_DISP_PAGE
                        ;       THIS ROUTINE SETS THE ACTIVE DISPLAY PAGE, ALLOWING
                        ;       THE FULL USE OF THE RAM SET ASIDE FOR THE VIDEO ATTACHMENT
                        ; INPUT
                        ;       AL HAS THE NEW ACTIVE DISPLAY PAGE
                        ; OUTPUT
                        ;       THE 6845 IS RESET TO DISPLAY THAT PAGE
                        ;--------------------------------------------------
E4B3                    ACT_DISP_PAGE   PROC    NEAR
E4B3  A8 80                     TEST    AL,080H                 ; CRT/CPU PAGE REG FUNCTION
E4B5  75 24                     JNZ     SET_CRTCPU              ; YES, GO HANDLE IT
E4B7  A2 0062 R                 MOV     ACTIVE_PAGE,AL          ; SAVE ACTIVE PAGE VALUE
E4BA  8B 0E 004C R              MOV     CX,CRT_LEN              ; GET SAVED LENGTH OF REGEN BUFFER
E4BE  98                        CBW                             ; CONVERT AL TO WORD
E4BF  50                        PUSH    AX                      ; SAVE PAGE VALUE
E4C0  F7 E1                     MUL     CX                      ; DISPLAY PAGE TIMES REGEN LENGTH
E4C2  A3 004E R                 MOV     CRT_START,AX            ; SAVE START ADDRESS FOR LATER USE
E4C5  8B C8                     MOV     CX,AX                   ; START ADDRESS TO CX
E4C7  D1 F9                     SAR     CX,1                    ; DIVIDE BY 2 FOR 6845 HANDLING
E4C9  B4 0C                     MOV     AH,12                   ; 6845 REGISTER FOR START ADDRESS
E4CB  E8 E472 R                 CALL    C23
E4CE  5B                        POP     BX                      ; RECOVER PAGE VALUE
E4CF  D1 E3                     SAL     BX,1                    ; *2 FOR WORD OFFSET
E4D1  8B 87 0050 R              MOV     AX,[BX + OFFSET CURSOR_POSN] ; GET CURSOR FOR THIS
                                                                ; PAGE
E4D5  E8 E4A2 R                 CALL    C25                     ; SET THE CURSOR POSITION
E4D8  E9 0F70 R                 JMP     VIDEO_RETURN
                        ;--------------------------------------------------------
                        ; SET_CRTCPU
                        ;       THIS ROUTINE READS OR WRITES THE CRT/CPU PAGE REGISTERS
                        ;
                        ; INPUT
                        ;       (AL) = 83H      SET BOTH CRT AND CPU PAGE REGS
                        ;              (BH) =   VALUE TO SET IN CRT PAGE REG
                        ;              (BL) =   VALUE TO SET IN CPU PAGE REG
                        ;       (AL) = 82H      SET CRT PAGE REG
                        ;              (BH) =   VALUE TO SET IN CRT PAGE REG
                        ;       (AL) = 81H      SET CPU PAGE REG
                        ;              (BL) =   VALUE TO SET IN CPU PAGE REG
                        ;       (AL) = 80H      READ CURRENT VALUE OF CRT/CPU PAGE REGS
                        ;
                        ; OUTPUT
                        ;       ALL FUNCTIONS RETURN
                        ;              (BH) = CURRENT CONTENTS OF CRT PAGE REG
                        ;              (BL) = CURRENT CONTENTS OF CPU PAGE REG
                        ;--------------------------------------------------------
E4DB                    SET_CRTCPU:
E4DB  8A E0                     MOV     AH,AL                   ; SAVE REQUEST IN AH
E4DD  BA 03DA                   MOV     DX,VGA_CTL              ; SET ADDRESS OF GATE ARRAY
E4E0  EC                C26:    IN      AL,DX                   ; GET STATUS
E4E1  24 08                     AND     AL,08H                  ; VERTICAL RETRACE?
E4E3  74 FB                     JZ      C26                     ; NO, WAIT FOR IT
E4E5  BA 03DF                   MOV     DX,PAGREG               ; SET IO ADDRESS OF PAGE REG
E4E8  A0 008A R                 MOV     AL,PAGDAT               ; GET DATA LAST OUTPUT TO REG
E4EB  80 FC 80                  CMP     AH,80H                  ; READ FUNCTION REQUESTED?
E4EE  74 27                     JZ      C29                     ; YES, DON'T SET ANYTHING
E4F0  80 FC 84                  CMP     AH,84H                  ; VALID REQUEST?
E4F3  73 22                     JNC     C29                     ; NO, PRETEND IT WAS A READ REQUEST
E4F5  F6 C4 01                  TEST    AH,1                    ; SET CPU REG?
E4F8  74 0D                     JZ      C27                     ; NO, GO SEE ABOUT CRT REG
E4FA  D0 E3                     SHL     BL,1                    ; SHIFT VALUE TO RIGHT BIT POSITION
E4FC  D0 E3                     SHL     BL,1
E4FE  D0 E3                     SHL     BL,1
E500  24 C7                     AND     AL,NOT CPUREG           ; CLEAR OLD CPU VALUE
E502  80 E3 38                  AND     BL,CPUREG               ; BE SURE UNRELATED BITS ARE ZERO
E505  0A C3                     OR      AL,BL                   ; OR IN NEW VALUE
```

```
E507  F6 C4 02                C27:    TEST    AH,2            ; SET CRT REG?
E50A  74 07                           JZ      C28             ; NO, GO RETURN CURRENT SETTINGS
E50C  24 F8                           AND     AL,NOT CRTREG   ; CLEAR OLD CRT VALUE
E50E  80 E7 07                        AND     BH,CRTREG       ; BE SURE UNRELATED BITS ARE ZERO
E511  0A C7                           OR      AL,BH           ; OR IN NEW VALUE
E513  EE                      C28:    OUT     DX,AL           ; SET NEW VALUES
E514  A2 008A R                       MOV     PAGDAT,AL       ; SAVE COPY IN RAM
E517  8A D8                   C29:    MOV     BL,AL           ; GET CPU REG VALUE
E519  80 E3 38                        AND     BL,CPUREG       ; CLEAR EXTRA BITS
E51C  D0 FB                           SAR     BL,1            ; RIGHT JUSTIFY IN BL
E51E  D0 FB                           SAR     BL,1
E520  D0 FB                           SAR     BL,1
E522  8A F8                           MOV     BH,AL           ; GET CRT REG VALUE
E524  80 E7 07                        AND     BH,CRTREG       ; CLEAR EXTRA BITS
E527  5F                              POP     DI              ; RESTORE SOME REGS
E528  5E                              POP     SI
E529  58                              POP     AX              ; DISCARD SAVED BX
E52A  E9 0F73 R                       JMP     C22             ; RETURN
E52D                          ACT_DISP_PAGE   ENDP
                              ;--------------------------------------------------
                              ; READ_CURSOR
                              ;       THIS ROUTINE READS THE CURRENT CURSOR VALUE FROM THE
                              ;       6845, FORMATS IT, AND SENDS IT BACK TO THE CALLER
                              ; INPUT
                              ;       BH - PAGE OF CURSOR
                              ; OUTPUT
                              ;       DX - ROW, COLUMN OF THE CURRENT CURSOR POSITION
                              ;       CX - CURRENT CURSOR MODE
                              ;-----------------------------------------
E52D                          READ_CURSOR     PROC    NEAR
E52D  8A DF                           MOV     BL,BH
E52F  32 FF                           XOR     BH,BH
E531  D1 E3                           SAL     BX,1            ; WORD OFFSET
E533  8B 97 0050 R                    MOV     DX,[BX+OFFSET CURSOR_POSN]
E537  8B 0E 0060 R                    MOV     CX,CURSOR_MODE
E53B  5F                              POP     DI
E53C  5E                              POP     SI
E53D  5B                              POP     BX
E53E  58                              POP     AX              ; DISCARD SAVED CX AND DX
E53F  58                              POP     AX
E540  1F                              POP     DS
E541  07                              POP     ES
E542  CF                              IRET
E543                          READ_CURSOR     ENDP
                              ;-----------------------------------------------
                              ; SET COLOR
                              ;       THIS ROUTINE WILL ESTABLISH THE BACKGROUND COLOR, THE
                              ;       OVERSCAN COLOR, AND THE FOREGROUND COLOR SET FOR GRAPHICS
                              ; INPUT
                              ;       (BH) HAS COLOR ID
                              ;               IF BH=0, THE BACKGROUND COLOR VALUE IS SET
                              ;                       FROM THE LOW BITS OF BL (0-31)
                              ;                       IN GRAPHIC MODES, BOTH THE BACKGROUND AND
                              ;                       BORDER ARE SET. IN ALPHA MODES, ONLY THE
                              ;                       BORDER IS SET.
                              ;               IF BH=1, THE PALETTE SELECTION IS MADE
                              ;                       BASED ON THE LOW BIT OF BL:
                              ;                          2 COLOR MODE:
                              ;                               0 = WHITE FOR COLOR 1
                              ;                               1 = BLACK FOR COLOR 1
                              ;                          4 COLOR MODES:
                              ;                               0 = GREEN, RED, YELLOW FOR
                              ;                                   COLORS 1,2,3
                              ;                               1 = BLUE, CYAN, MAGENTA FOR
                              ;                                   COLORS 1,2,3
                              ;                          16 COLOR MODES:
                              ;                               ALWAYS SETS UP PALETTE AS:
                              ;                               BLUE FOR COLOR 1
                              ;                               GREEN FOR COLOR 2
                              ;                               CYAN FOR COLOR 3
                              ;                               RED FOR COLOR 4
                              ;                               MAGENTA FOR COLOR 5
                              ;                               BROWN FOR COLOR 6
                              ;                               LIGHT GRAY FOR COLOR 7
                              ;                               DARK GRAY FOR COLOR 8
                              ;                               LIGHT BLUE FOR COLOR 9
                              ;                               LIGHT GREEN FOR COLOR 10
                              ;                               LIGHT CYAN FOR COLOR 11
                              ;                               LIGHT RED FOR COLOR 12
                              ;                               LIGHT MAGENTA FOR COLOR 13
                              ;                               YELLOW FOR COLOR 14
                              ;                               WHITE FOR COLOR 15
                              ;       (BL) HAS THE COLOR VALUE TO BE USED
                              ; OUTPUT
                              ;       THE COLOR SELECTION IS UPDATED
                              ;-----------------------------------------
E543                          SET_COLOR       PROC    NEAR
E543  BA 03DA                         MOV     DX,VGA_CTL      ; I/O PORT FOR PALETTE
E546  EC                      C30:    IN      AL,DX           ; SYNC UP VGA FOR REG ADDRESS
E547  A8 08                           TEST    AL,8            ; IS VERTICAL RETRACE ON?
E549  74 FB                           JZ      C30             ; NO, WAIT UNTIL IT IS
E54B  0A FF                           OR      BH,BH           ; IS THIS COLOR 0?
E54D  75 19                           JNZ     C31             ; OUTPUT COLOR 1
```

```
                        ;------ HANDLE COLOR 0 BY SETTING THE BACKGROUND COLOR
                        ;       AND BORDER COLOR
E54F  80 3E 0049 R 04           CMP     CRT_MODE,4      ; IN ALPHA MODE?
E554  72 06                     JC      C305            ; YES, JUST SET BORDER REG
E556  B0 10                     MOV     AL,10H          ; SET PALETTE REG 0
E558  EE                        OUT     DX,AL           ; SELECT VGA REG
E559  8A C3                     MOV     AL,BL           ; GET COLOR
E55B  EE                        OUT     DX,AL           ; SET IT
E55C  B0 02             C305:   MOV     AL,2            ; SET BORDER REG
E55E  EE                        OUT     DX,AL           ; SELECT VGA BORDER REG
E55F  8A C3                     MOV     AL,BL           ; GET COLOR
E561  EE                        OUT     DX,AL           ; SET IT
E562  A2 0066 R                 MOV     CRT_PALLETTE,AL ; SAVE THE COLOR VALUE
E565  E9 0F70 R                 JMP     VIDEO_RETURN
                        ;------ HANDLE COLOR 1 BY CHANGING PALETTE REGISTERS
E568  A0 0049 R         C31:    MOV     AL,CRT_MODE     ; GET CURRENT MODE
E56B  B9 0D95 R                 MOV     CX,OFFSET M0072 ; POINT TO 2 COLOR TABLE ENTRY
E56E  3C 06                     CMP     AL,6            ; 2 COLOR MODE?
E570  74 0F                     JE      C33             ; YES, JUMP
E572  3C 04                     CMP     AL,4            ; 4 COLOR MODE?
E574  74 08                     JE      C32             ; YES, JUMP
E576  3C 05                     CMP     AL,5            ; 4 COLOR MODE?
E578  74 04                     JE      C32             ; YES, JUMP
E57A  3C 0A                     CMP     AL,0AH          ; 4 COLOR MODE?
E57C  75 20                     JNE     C36             ; NO, GO TO 16 COLOR SET UP
E57E  B9 0D9D R         C32:    MOV     CX,OFFSET M0074 ; POINT TO 4 COLOR TABLE ENTRY
E581  D0 CB             C33:    ROR     BL,1            ; SELECT ALTERNATE SET?
E583  73 03                     JNC     C34             ; NO, JUMP
E585  83 C1 04                  ADD     CX,M0072L       ; POINT TO NEXT ENTRY
E588  8B D9             C34:    MOV     BX,CX           ; TABLE ADDRESS IN BX
E58A  43                        INC     BX              ; SKIP OVER BACKGROUND COLOR
E58B  B9 0003                   MOV     CX,M0072L-1     ; SET NUMBER OF REGS TO FILL
E58E  B4 11                     MOV     AH,11H          ; AH IS REGISTER COUNTER
E590  8A C4             C35:    MOV     AL,AH           ; GET REG NUMBER
E592  EE                        OUT     DX,AL           ; SELECT IT
E593  2E: 8A 07                 MOV     AL,CS:[BX]      ; GET DATA
E596  EE                        OUT     DX,AL           ; SET IT
E597  FE C4                     INC     AH              ; NEXT REG
E599  43                        INC     BX              ; NEXT TABLE VALUE
E59A  E2 F4                     LOOP    C35
E59C  EB 0D                     JMP     SHORT C38
E59E  B4 11             C36:    MOV     AH,11H          ; AH IS REGISTER COUNTER
E5A0  B9 000F                   MOV     CX,15           ; NUMBER OF PALETTES
E5A3  8A C4             C37:    MOV     AL,AH           ; GET REG NUMBER
E5A5  EE                        OUT     DX,AL           ; SELECT IT
E5A6  EE                        OUT     DX,AL           ; SET PALETTE VALUE
E5A7  FE C4                     INC     AH              ; NEXT REG
E5A9  E2 F8                     LOOP    C37
E5AB  32 C0             C38:    XOR     AL,AL           ; SELECT LOW REG TO ENABLE VIDEO
                                                        ; AGAIN
E5AD  EE                        OUT     DX,AL
E5AE  E9 0F70 R                 JMP     VIDEO_RETURN
E5B1                    SET_COLOR       ENDP
                        ;--------------------------------------------
                        ; VIDEO STATE
                        ;   RETURNS THE CURRENT VIDEO STATE IN AX
                        ;   AH = NUMBER OF COLUMNS ON THE SCREEN
                        ;   AL = CURRENT VIDEO MODE
                        ;   BH = CURRENT ACTIVE PAGE
                        ;--------------------------------------------
E5B1                    VIDEO_STATE     PROC    NEAR
E5B1  8A 26 004A R              MOV     AH,BYTE PTR CRT_COLS ; GET NUMBER OF COLUMNS
E5B5  A0 0049 R                 MOV     AL,CRT_MODE     ; CURRENT MODE
E5B8  8A 3E 0062 R              MOV     BH,ACTIVE_PAGE  ; GET CURRENT ACTIVE PAGE
E5BC  5F                        POP     DI              ; RECOVER REGISTERS
E5BD  5E                        POP     SI              ;
E5BE  59                        POP     CX              ; DISCARD SAVED BX
E5BF  E9 0F73 R                 JMP     C22             ; RETURN TO CALLER
E5C2                    VIDEO_STATE     ENDP
                        ;-----------------------------------
                        ; POSITION
                        ;       THIS SERVICE ROUTINE CALCULATES THE REGEN BUFFER ADDRESS
                        ;       OF A CHARACTER IN THE ALPHA MODE
                        ; INPUT
                        ;       AX = ROW, COLUMN POSITION
                        ; OUTPUT
                        ;       AX = OFFSET OF CHAR POSITION IN REGEN BUFFER
                        ;-----------------------------------
E5C2                    POSITION        PROC    NEAR
E5C2  53                        PUSH    BX              ; SAVE REGISTER
E5C3  8B D8                     MOV     BX,AX
E5C5  8A C4                     MOV     AL,AH           ; ROWS TO AL
E5C7  F6 26 004A R              MUL     BYTE PTR CRT_COLS       ; DETERMINE BYTES TO ROW
E5CB  32 FF                     XOR     BH,BH
E5CD  03 C3                     ADD     AX,BX           ; ADD IN COLUMN VALUE
E5CF  D1 E0                     SAL     AX,1            ; * 2 FOR ATTRIBUTE BYTES
E5D1  5B                        POP     BX
E5D2  C3                        RET
E5D3                    POSITION        ENDP
                        ;--------------------------------------
                        ; SCROLL UP
                        ;       THIS ROUTINE MOVES A BLOCK OF CHARACTERS UP
                        ;       ON THE SCREEN
                        ; INPUT
                        ;       (AH) = CURRENT CRT MODE
                        ;       (AL) = NUMBER OF ROWS TO SCROLL
                        ;       (CX) = ROW/COLUMN OF UPPER LEFT CORNER
                        ;       (DX) = ROW/COLUMN OF LOWER RIGHT CORNER
                        ;       (BH) = ATTRIBUTE TO BE USED ON BLANKED LINE
                        ;       (DS) = DATA SEGMENT
                        ;       (ES) = REGEN BUFFER SEGMENT
                        ; OUTPUT
                        ;       NONE -- THE REGEN BUFFER IS MODIFIED
                        ;--------------------------------------
```

```
                        ASSUME  CS:CODE,DS:DATA,ES:DATA
E5D3                    SCROLL_UP       PROC    NEAR
E5D3  8A D8                     MOV     BL,AL                   ; SAVE LINE COUNT IN BL
E5D5  80 FC 04                  CMP     AH,4                    ; TEST FOR GRAPHICS MODE
E5D8  72 03                     JC      C39                     ; HANDLE SEPARATELY
E5DA  E9 F259 R                 JMP     GRAPHICS_UP
E5DD                    C39:                                    ; UP_CONTINUE
E5DD  53                        PUSH    BX                      ; SAVE FILL ATTRIBUTE IN BH
E5DE  8B C1                     MOV     AX,CX                   ; UPPER LEFT POSITION
E5E0  E8 E609 R                 CALL    SCROLL_POSITION         ; DO SETUP FOR SCROLL
E5E3  74 20                     JZ      C44                     ; BLANK_FIELD
E5E5  03 F0                     ADD     SI,AX                   ; FROM ADDRESS
E5E7  8A E6                     MOV     AH,DH                   ; # ROWS IN BLOCK
E5E9  2A E3                     SUB     AH,BL                   ; # ROWS TO BE MOVED
E5EB  E8 E62F R         C40:    CALL    C45                     ; MOVE ONE ROW
E5EE  03 F5                     ADD     SI,BP
E5F0  03 FD                     ADD     DI,BP                   ; POINT TO NEXT LINE IN BLOCK
E5F2  FE CC                     DEC     AH                      ; COUNT OF LINES TO MOVE
E5F4  75 F5                     JNZ     C40                     ; ROW_LOOP
E5F6  58                C41:    POP     AX                      ; RECOVER ATTRIBUTE IN AH
E5F7  B0 20                     MOV     AL,' '                  ; FILL WITH BLANKS
E5F9  E8 E638 R         C42:    CALL    C46                     ; CLEAR THE ROW
E5FC  03 FD                     ADD     DI,BP                   ; POINT TO NEXT LINE
E5FE  FE CB                     DEC     BL                      ; COUNTER OF LINES TO SCROLL
E600  75 F7                     JNZ     C42                     ; CLEAR_LOOP
E602  E9 0F70 R         C43:    JMP     VIDEO_RETURN
E605  8A DE             C44:    MOV     BL,DH                   ; GET ROW COUNT
E607  EB ED                     JMP     C41                     ; GO CLEAR THAT AREA
E609                    SCROLL_UP       ENDP
                        ;----- HANDLE COMMON SCROLL SET UP HERE
E609                    SCROLL_POSITION PROC    NEAR
E609  E8 E5C2 R                 CALL    POSITION                ; CONVERT TO REGEN POINTER
E60C  03 06 004E R              ADD     AX,CRT_START            ; OFFSET OF ACTIVE PAGE
E610  8B F8                     MOV     DI,AX                   ; TO ADDRESS FOR SCROLL
E612  8B F0                     MOV     SI,AX                   ; FROM ADDRESS FOR SCROLL
E614  2B D1                     SUB     DX,CX                   ; DX = #ROWS, #COLS IN BLOCK
E616  FE C6                     INC     DH
E618  FE C2                     INC     DL                      ; INCREMENT FOR 0 ORIGIN
E61A  32 ED                     XOR     CH,CH                   ; SET HIGH BYTE OF COUNT TO ZERO
E61C  8B 2E 004A R              MOV     BP,CRT_COLS             ; GET NUMBER OF COLUMNS IN DISPLAY
E620  03 ED                     ADD     BP,BP                   ; TIMES 2 FOR ATTRIBUTE BYTE
E622  8A C3                     MOV     AL,BL                   ; GET LINE COUNT
E624  F6 26 004A R              MUL     BYTE PTR CRT_COLS       ; DETERMINE OFFSET TO FROM
                                                                ; ADDRESS
E628  03 C0                     ADD     AX,AX                   ; *2 FOR ATTRIBUTE BYTE
E62A  06                        PUSH    ES                      ; ESTABLISH ADDRESSING TO REGEN
                                                                ; BUFFER
E62B  1F                        POP     DS                      ;  FOR BOTH POINTERS
E62C  0A DB                     OR      BL,BL                   ; 0 SCROLL MEANS BLANK FIELD
E62E  C3                        RET                             ; RETURN WITH FLAGS SET
E62F                    SCROLL_POSITION ENDP
                        ;------ MOVE_ROW
E62F                    C45     PROC    NEAR
E62F  8A CA                     MOV     CL,DL                   ; GET # OF COLS TO MOVE
E631  56                        PUSH    SI
E632  57                        PUSH    DI                      ; SAVE START ADDRESS
E633  F3/ A5                    REP     MOVSW                   ; MOVE THAT LINE ON SCREEN
E635  5F                        POP     DI
E636  5E                        POP     SI                      ; RECOVER ADDRESSES
E637  C3                        RET
E638                    C45     ENDP
                        ;------ CLEAR_ROW
E638                    C46     PROC    NEAR
E638  8A CA                     MOV     CL,DL                   ; GET # COLUMNS TO CLEAR
E63A  57                        PUSH    DI
E63B  F3/ AB                    REP     STOSW                   ; STORE THE FILL CHARACTER
E63D  5F                        POP     DI
E63E  C3                        RET
E63F                    C46     ENDP
                        ;--------------------------------------
                        ; SCROLL_DOWN
                        ;       THIS ROUTINE MOVES THE CHARACTERS WITHIN A DEFINED
                        ;       BLOCK DOWN ON THE SCREEN, FILLING THE TOP LINES
                        ;       WITH A DEFINED CHARACTER
                        ; INPUT
                        ;       (AH) = CURRENT CRT MODE
                        ;       (AL) = NUMBER OF LINES TO SCROLL
                        ;       (CX) = UPPER LEFT CORNER OF REGION
                        ;       (DX) = LOWER RIGHT CORNER OF REGION
                        ;       (BH) = FILL CHARACTER
                        ;       (DS) = DATA SEGMENT
                        ;       (ES) = REGEN SEGMENT
                        ; OUPUT
                        ;       NONE -- SCREEN IS SCROLLED
                        ;--------------------------------------
E63F                    SCROLL_DOWN     PROC    NEAR
E63F  FD                        STD                             ; DIRECTION FOR SCROLL DOWN
E640  8A D8                     MOV     BL,AL                   ; LINE COUNT TO BL
E642  80 FC 04                  CMP     AH,4                    ; TEST FOR GRAPHICS
E645  72 03                     JC      C47
E647  E9 F305 R                 JMP     GRAPHICS_DOWN
E64A  53                C47:    PUSH    BX                      ; SAVE ATTRIBUTE IN BH
E64B  8B C2                     MOV     AX,DX                   ; LOWER RIGHT CORNER
E64D  E8 E609 R                 CALL    SCROLL_POSITION         ; GET REGEN LOCATION
E650  74 1F                     JZ      C51
E652  2B F0                     SUB     SI,AX                   ; SI IS FROM ADDRESS
E654  8A E6                     MOV     AH,DH                   ; GET TOTAL # ROWS
E656  2A E3                     SUB     AH,BL                   ; COUNT TO MOVE IN SCROLL
```

```
E658  E8 E62F R           C48:    CALL    C45             ; MOVE ONE ROW
E65B  2B F5                       SUB     SI,BP
E65D  2B FD                       SUB     DI,BP
E65F  FE CC                       DEC     AH
E661  75 F5                       JNZ     C48
E663  58                  C49:    POP     AX              ; RECOVER ATTRIBUTE IN AH
E664  B0 20                       MOV     AL,' '
E666  E8 E638 R           C50:    CALL    C46             ; CLEAR ONE ROW
E669  2B FD                       SUB     DI,BP           ; GO TO NEXT ROW
E66B  FE CB                       DEC     BL
E66D  75 F7                       JNZ     C50
E66F  EB 91                       JMP     C43             ; SCROLL_END
E671  8A DE               C51:    MOV     BL,DH
E673  EB EE                       JMP     C49
E675                      SCROLL_DOWN     ENDP
                          ;--------------------------------------
                          ; MODE_ALIVE
                          ;       THIS ROUTINE READS 256 LOCATIONS IN MEMORY AS EVERY OTHER
                          ;       LOCATION IN 512 LOCATIONS.  THIS IS TO INSURE THE DATA
                          ;       INTEGRITY OF MEMORY DURING MODE CHANGES.
                          ;--------------------------------------
E675                      MODE_ALIVE      PROC    NEAR
E675  50                          PUSH    AX              ;SAVE USED REGS
E676  56                          PUSH    SI
E677  51                          PUSH    CX
E678  33 F6                       XOR     SI,SI
E67A  B9 0100                     MOV     CX,256
E67D  AC                  C52:    LODSB
E67E  46                          INC     SI
E67F  E2 FC                       LOOP    C52
E681  59                          POP     CX
E682  5E                          POP     SI
E683  58                          POP     AX
E684  C3                          RET
E685                      MODE_ALIVE      ENDP
                          ;--------------------------------------------------------
                          ; SET_PALLETTE
                          ;       THIS ROUTINE WRITES THE PALETTE REGISTERS
                          ; INPUT
                          ;       (AL) = 0        SET PALETTE REG
                          ;               (BH) =  VALUE TO SET
                          ;               (BL) =  PALETTE REG TO SET
                          ;       (AL) = 1        SET BORDER COLOR REG
                          ;               (BH) =  VALUE TO SET
                          ;       (AL) = 2        SET ALL PALETTE REGS AND BORDER REG
                          ;       NOTE: REGISTERS ARE WRITE ONLY.
                          ;--------------------------------------------------------
E685                      SET_PALLETTE    PROC    NEAR
E685  50                          PUSH    AX
E686  8B F4                       MOV     SI,SP
E688  36: 8B 44 0C                MOV     AX,SS:[SI+12]   ; GET SEG FROM STACK
E68C  8E C0                       MOV     ES,AX
E68E  8B F2                       MOV     SI,DX           ; OFFSET IN SI
E690  BA 03DA                     MOV     DX,VGA_CTL      ; SET VGA CONTROL PORT
E693  EC                  C53:    IN      AL,DX           ; GET VGA STATUS
E694  24 08                       AND     AL,08H          ; IN VERTICAL RETRACE?
E696  75 FB                       JNZ     C53             ; YES, WAIT FOR IT TO GO AWAY
E698  EC                  C54:    IN      AL,DX           ; GET VGA STATUS
E699  24 08                       AND     AL,08H          ; IN VERITCAL RETRACE?
E69B  74 FB                       JZ      C54             ; NO, WAIT FOR IT
E69D  58                          POP     AX
E69E  0A C0                       OR      AL,AL           ; SET PALETTE REG?
E6A0  74 0C                       JZ      C55             ; YES, GO DO IT
E6A2  3C 02                       CMP     AL,2            ; SET ALL REGS?
E6A4  74 17                       JE      C57             ;
E6A6  3C 01                       CMP     AL,1            ; SET BORDER COLOR REG?
E6A8  75 2B                       JNE     C59             ; NO, DON'T DO ANYTHING
E6AA  B0 02                       MOV     AL,2            ; SET BORDER COLOR REG NUMBER
E6AC  EB 06                       JMP     SHORT C56
E6AE  8A C3               C55:    MOV     AL,BL           ; GET DESIRED REG NUMBER IN AL
E6B0  24 0F                       AND     AL,0FH          ; STRIP UNUSED BITS
E6B2  0C 10                       OR      AL,10H          ; MAKE INTO REAL REG NUMBER
E6B4  EE                  C56:    OUT     DX,AL           ; SELECT REG
E6B5  8A C7                       MOV     AL,BH           ; GET DATA IN AL
E6B7  EE                          OUT     DX,AL           ; SET NEW DATA
E6B8  32 C0                       XOR     AL,AL           ; SET REG 0 SO DISPLAY WORKS AGAIN
E6BA  EE                          OUT     DX,AL
E6BB  EB 18                       JMP     SHORT C59
E6BD  B4 10               C57:    MOV     AH,10H          ; AH IS REG COUNTER
E6BF  8A C4               C58:    MOV     AL,AH           ; REG ADDRESS IN AL
E6C1  EE                          OUT     DX,AL           ; SELECT IT
E6C2  26: 8A 04                   MOV     AL,BYTE PTR ES:[SI]     ;GET DATA
E6C5  EE                          OUT     DX,AL           ; PUT IN VGA REG
E6C6  46                          INC     SI              ; NEXT DATA BYTE
E6C7  FE C4                       INC     AH              ; NEXT REG
E6C9  80 FC 20                    CMP     AH,20H          ; LAST PALETTE REG?
E6CC  72 F1                       JB      C58             ; NO, DO NEXT ONE
E6CE  B0 02                       MOV     AL,2            ; SET BORDER REG
E6D0  EE                          OUT     DX,AL           ; SELECT IT
E6D1  26: 8A 04                   MOV     AL,BYTE PTR ES:[SI]     ; GET DATA
E6D4  EE                          OUT     DX,AL           ; PUT IN VGA REG
```

```
E6D4  EE                        OUT     DX,AL           ; PUT IN VGA REG
E6D5  E9 0F70 R                 JMP     VIDEO_RETURN    ; ALL DONE
E6D8                            SET_PALLETTE    ENDP
E6D8                            MFG_UP  PROC    NEAR
E6D8  50                        PUSH    AX
E6D9  1E                        PUSH    DS
                                ASSUME  DS:XXDATA
E6DA  B8  ---- R                MOV     AX,XXDATA
E6DD  8E D8                     MOV     DS,AX
E6DF  A0 0005 R                 MOV     AL,MFG_TST      ; GET MFG CHECKPOINT
E6E2  E6 10                     OUT     10H,AL          ; OUTPUT IT TO TESTER
E6E4  FE C8                     DEC     AL              ; DROP IT BY 1 FOR THE NEXT TEST
E6E6  A2 0005 R                 MOV     MFG_TST,AL
                                ASSUME  DS:ABS0
E6E9  1F                        POP     DS
E6EA  58                        POP     AX
E6EB  C3                        RET
E6EC                            MFG_UP  ENDP
                                ASSUME  CS:CODE,DS:DATA
E6F2                            ORG     0E6F2H
E6F2  E9 0B1B R                 JMP     NEAR PTR BOOT_STRAP
                        ;--------------------------------------------------------------------------
                        ;   SUBROUTINE TO SET UP CONDITIONS FOR THE TESTING OF 8250 AND
                        ;       8259 INTERRUPTS.  ENABLES MASKABLE EXTERNAL INTERRUPTS,
                        ;       CLEARS THE 8259 INTR RECEIVED FLAG BIT, AND ENABLES THE
                        ;       DEVICE'S 8259 INTR (WHICHEVER IS BEING TESTED).
                        ;   IT EXPECTS TO BE PASSED:
                        ;       (DS) = ADDRESS OF SEGMENT WHERE INTR_FLAG IS DEFINED
                        ;       (DI) = OFFSET OF THE INTERRUPT BIT MASK
                        ;   UPON RETURN:
                        ;       INTR_FLAG BIT FOR THE DEVICE = 0
                        ;   NO REGISTERS ARE ALTERED.
                        ;--------------------------------------------------------------------------
E6F5                            SUI     PROC    NEAR
E6F5  50                        PUSH    AX
E6F6  FB                        STI                     ; ENABLE MASKABLE EXTERNAL
                                                        ; INTERRUPTS
E6F7  2E: 8A 25                 MOV     AH,CS:[DI]      ; GET INTERRUPT BIT MASK
E6FA  20 26 0084 R              AND     INTR_FLAG,AH    ; CLEAR 8259 INTERRUPT REC'D FLAG
                                                        ; BIT
E6FE  E4 21                     IN      AL,INTA01       ; CURRENT INTERRUPTS
E700  22 C4                     AND     AL,AH           ; ENABLE THIS INTERRUPT, TOO
E702  E6 21                     OUT     INTA01,AL       ; WRITE TO 8259 (INTERRUPT
                                                        ; CONTROLLER)
E704  58                        POP     AX
E705  C3                        RET
E706                            SUI     ENDP
                        ;--------------------------------------------------------------------------
                        ;   SUBROUTINE WHICH CHECKS IF A 8259 INTERRUPT IS GENERATED BY THE
                        ;       8250 INTERRUPT.
                        ;   IT EXPECTS TO BE PASSED:
                        ;       (DI) = OFFSET OF INTERRUPT BIT MASK
                        ;       (DS) = ADDRESS OF SEGMENT WHERE INTR_FLAG IS DEFINED.
                        ;   IT RETURNS:
                        ;       (CF) = 1 IF NO INTERRUPT IS GENERATED
                        ;              0 IF THE INTERRUPT OCCURRED
                        ;       (AL) = COMPLEMENT OF THE INTERRUPT MASK
                        ;   NO OTHER REGISTERS ARE ALTERED.
                        ;--------------------------------------------------------------------------
E706                            C5059   PROC    NEAR
E706  51                        PUSH    CX
E707  2B C9                     SUB     CX,CX           ; SET PROGRAM LOOP COUNT
E709  2E: 8A 05                 MOV     AL,CS:[DI]      ; GET INTERRUPT MASK
E70C  34 FF                     XOR     AL,0FFH         ; COMPLEMENT MASK SO ONLY THE INTR
                                                        ; TEST BIT IS ON
E70E  84 06 0084 R              AT25:   TEST    INTR_FLAG,AL    ; 8259 INTERRUPT OCCUR?
E712  75 03                     JNE     AT27            ; YES - CONTINUE
E714  E2 F8                     LOOP    AT25            ; WAIT SOME MORE
E716  F9                        STC                     ; TIME'S UP - FAILED
E717  59                        AT27:   POP     CX
E718  C3                        RET
E719                            C5059   ENDP
                        ;--------------------------------------------------------------------------
                        ;   SUBROUTINE TO WAIT FOR ALL ENABLED 8250 INTERRUPTS TO CLEAR (SO
                        ;       NO INTRS WILL BE PENDING).  EACH INTERRUPT COULD TAKE UP TO
                        ;       1 MILLISECOND TO CLEAR. THE INTERRUPT IDENTIFICATION
                        ;       REGISTER WILL BE CHECKED UNTIL THE INTERRUPT(S) IS CLEARED
                        ;       OR A TIMEOUT OCCURS.
                        ;   EXPECTS TO BE PASSED:
                        ;       (DX) = ADDRESS OF THE INTERRUPT ID REGISTER
                        ;   RETURNS:
                        ;       (AL) = CONTENTS OF THE INTR ID REGISTER
                        ;       (CF) = 1  IF INTERRUPTS ARE STILL PENDING
                        ;              0  IF NO INTERRUPTS ARE PENDING (ALL CLEAR)
                        ;   NO OTHER REGISTERS ARE ALTERED.
                        ;--------------------------------------------------------------------------
E719                            W8250C  PROC    NEAR
E719  51                        PUSH    CX
E71A  2B C9                     SUB     CX,CX
E71C  EC                        AT28:   IN      AL,DX           ; READ INTR ID REG
E71D  3C 01                     CMP     AL,1            ; INTERRUPTS STILL PENDING?
E71F  74 05                     JE      AT29            ; NO - GOOD FINISH
E721  E2 F9                     LOOP    AT28            ; KEEP TRYING
E723  F9                        STC                     ; TIME'S UP - ERROR
E724  EB 01                     JMP     SHORT AT30
E726  F8                        AT29:   CLC
E727  59                        AT30:   POP     CX
E728  C3                        RET
E729                            W8250C  ENDP
```

```
                        ;-----INT 14----------------------------------------------
                        ;RS232_IO
                        ;       THIS ROUTINE PROVIDES BYTE STREAM I/O TO THE COMMUNICATIONS
                        ;       PORT ACCORDING TO THE PARAMETERS:
                        ;       (AH)=0  INITIALIZE THE COMMUNICATIONS PORT
                        ;               (AL) HAS PARMS FOR INITIALIZATION
                        ;
                        ;---7-------6-------5-------4-------3-------2---------1--------0----
                        ;-------- BAUD RATE ---:::----PARITY----:::-STOPBIT-:::--WORD LENGTH--
                        ;
                        ;  000 - 110                X0 - NONE        0 - 1     10 - 7 BITS
                        ;  001 - 150                01 - ODD         1 - 2     11 - 8 BITS
                        ;  010 - 300                11 - EVEN
                        ;  011 - 600
                        ;  100 - 1200
                        ;  101 - 2400
                        ;  110 - 4800
                        ;  111 - 4800
                        ;               ON RETURN, THE RS232 INTERRUPTS ARE DISABLED AND
                        ;                       CONDITIONS ARE SET AS IN CALL TO COMMO
                        ;                       STATUS (AH=3)
                        ;       (AH)=1  SEND THE CHARACTER IN (AL) OVER THE COMMO LINE
                        ;               (AL) REGISTER IS PRESERVED
                        ;               ON EXIT, BIT 7 OF AH IS SET IF THE ROUTINE WAS
                        ;                       UNABLE TO TRANSMIT THE BYTE OF DATA OVER
                        ;                       THE LINE. IF BIT 7 OF AH IS NOT SET, THE
                        ;                       REMAINDER OF AH IS SET AS IN A STATUS
                        ;                       REQUEST, REFELECTING THE CURRENT STATUS OF
                        ;                       THE LINE.
                        ;       (AH)=2  RECEIVE A CHARACTER IN (AL) FROM COMMO LINE BEFORE
                        ;                       RETURNING TO CALLER
                        ;               ON EXIT, AH HAS THE CURRENT LINE STATUS, AS SET BY
                        ;                       THE STATUS ROUTINE, EXCEPT THAT THE ONLY
                        ;                       BITS LEFT ON, ARE THE ERROR BITS
                        ;                       (7,4,3,2,1). IN THIS CASE, THE TIME OUT BIT
                        ;                       INDICATES DATA SET READY WAS NOT RECEIVED.
                        ;                       THUS, AH IS NON ZERO ONLY WHEN AN ERROR
                        ;                       OCCURRED.(NOTE: IF THE TIME-OUT BIT IS SET,
                        ;                       OTHER BITS IN AH MAY NOT BE RELIABLE.)
                        ;       (AH)=3  RETURN THE COMMO PORT STATUS IN (AX)
                        ;               AH CONTAINS THE LINE CONTROL STATUS
                        ;               BIT 7 = TIME OUT
                        ;               BIT 6 = TRANS SHIFT REGISTER EMPTY
                        ;               BIT 5 = TRAN HOLDING REGISTER EMPTY
                        ;               BIT 4 = BREAK DETECT
                        ;               BIT 3 = FRAMING ERROR
                        ;               BIT 2 = PARITY ERROR
                        ;               BIT 1 = OVERRUN ERROR
                        ;               BIT 0 = DATA READY
                        ;               AL CONTAINS THE MODEM STATUS
                        ;               BIT 7 = RECIEVED LINE SIGNAL DETECT
                        ;               BIT 6 = RING INDICATOR
                        ;               BIT 5 = DATA SET READY
                        ;               BIT 4 = CLEAR TO SEND
                        ;               BIT 3 = DELTA RECEIVE LINE SIGNAL DETECT
                        ;               BIT 2 = TRAILING EDGE RING DETECTOR
                        ;               BIT 1 = DELTA DATA SET READY
                        ;               BIT 0 = DELTA CLEAR TO SEND
                        ;       (DX) = PARAMETER INDICATING WHICH RS232 CARD (0,1 ALLOWED)
                        ; DATA AREA RS232_BASE CONTAINS THE BASE ADDRESS OF THE 8250 ON THE
                        ;    CARD.  LOCATION 400H CONTAINS UP TO 4 RS232 ADDRESSES POSSIBLE
                        ; DATA AREA RS232_TIM_OUT (BYTE) CONTAINS OUTER LOOP COUNT
                        ;       VALUE FOR TIMEOUT (DEFAULT=1)
                        ;OUTPUT
                        ;               AX      MODIFIED ACCORDING TO PARMS OF CALL
                        ;               ALL OTHERS UNCHANGED
                        ;-----------------------------------------------------
                                ASSUME  CS:CODE,DS:DATA
E729                            ORG     0E729H
E729                    A1      LABEL   WORD
E729  03F9                      DW      1017    ; 110 BAUD      ; TABLE OF INIT VALUE
E72B  02EA                      DW      746     ; 150
E72D  0175                      DW      373     ; 300
E72F  00BA                      DW      186     ; 600
E731  005D                      DW      93      ; 1200
E733  002F                      DW      47      ; 2400
E735  0017                      DW      23      ; 4800
E737  0017                      DW      23      ; 4800
E739                    RS232_IO        PROC    FAR
                        ;------ VECTOR TO APPROPRIATE ROUTINE
E739  FB                        STI                     ; INTERRUPTS BACK ON
E73A  1E                        PUSH    DS              ; SAVE SEGMENT
E73B  52                        PUSH    DX
E73C  56                        PUSH    SI
E73D  57                        PUSH    DI
E73E  51                        PUSH    CX
E73F  53                        PUSH    BX
E740  8B F2                     MOV     SI,DX           ; RS232 VALUE TO SI
E742  8B FA                     MOV     DI,DX           ; AND TO DI (FOR TIMEOUTS)
E744  D1 E6                     SHL     SI,1            ; WORD OFFSET
E746  E8 138B R                 CALL    DDS             ; POINT TO BIOS DATA SEGMENT
E749  8B 94 0000 R              MOV     DX,RS232_BASE[SI] ; GET BASE ADDRESS
E74D  0B D2                     OR      DX,DX           ; TEST FOR 0 BASE ADDRESS
E74F  74 13                     JZ      A3              ; RETURN
E751  0A E4                     OR      AH,AH           ; TEST FOR (AH)=0
E753  74 16                     JZ      A4              ; COMMUN INIT
E755  FE CC                     DEC     AH              ; TEST FOR (AH)=1
E757  74 47                     JZ      A5              ; SEND AL
E759  FE CC                     DEC     AH              ; TEST FOR (AH)=2
E75B  74 6C                     JZ      A12             ; RECEIVE INTO AL
E75D  FE CC                     DEC     AH              ; TEST FOR (AH)=3
E75F  75 03                     JNZ     A3
E761  E9 E7F3 R                 JMP     A18             ; COMMUNICATION STATUS
```

```
E764                                A3:                             ; RETURN FROM RS232
E764  5B                                    POP     BX
E765  59                                    POP     CX
E766  5F                                    POP     DI
E767  5E                                    POP     SI
E768  5A                                    POP     DX
E769  1F                                    POP     DS
E76A  CF                                    IRET                    ; RETURN TO CALLER, NO ACTION
                                            ;------ INITIALIZE THE COMMUNICATIONS PORT
E76B  8A E0                         A4:     MOV     AH,AL           ; SAVE INIT PARMS IN AH
E76D  83 C2 03                              ADD     DX,3            ; POINT TO 8250 CONTROL REGISTER
E770  B0 80                                 MOV     AL,80H
E772  EE                                    OUT     DX,AL           ; SET DLAB=1
                                    ;------ DETERMINE BAUD RATE DIVISOR
E773  8A D4                                 MOV     DL,AH           ; GET PARMS TO DL
E775  B1 04                                 MOV     CL,4
E777  D2 C2                                 ROL     DL,CL
E779  81 E2 000E                            AND     DX,0EH          ; ISOLATE THEM
E77D  BF E729 R                             MOV     DI,OFFSET A1    ; BASE OF TABLE
E780  03 FA                                 ADD     DI,DX           ; PUT INTO INDEX REGISTER
E782  8B 94 0000 R                          MOV     DX,RS232_BASE[SI] ; POINT TO HIGH ORDER OF DIVISOR
E786  42                                    INC     DX
E787  2E: 8A 45 01                          MOV     AL,CS:[DI]+1    ; GET HIGH ORDER OF DIVISOR
E78B  EE                                    OUT     DX,AL           ; SET MS OF DIV TO 0
E78C  4A                                    DEC     DX
E78D  2E: 8A 05                             MOV     AL,CS:[DI]      ; GET LOW ORDER OF DIVISOR
E790  EE                                    OUT     DX,AL           ; SET LOW OF DIVISOR
E791  83 C2 03                              ADD     DX,3
E794  8A C4                                 MOV     AL,AH           ; GET PARMS BACK
E796  24 1F                                 AND     AL,01FH         ; STRIP OFF THE BAUD BITS
E798  EE                                    OUT     DX,AL           ; LINE CONTROL TO 8 BITS
E799  4A                                    DEC     DX
E79A  4A                                    DEC     DX
E79B  B0 00                                 MOV     AL,0
E79D  EE                                    OUT     DX,AL           ; INTERRUPT ENABLES ALL OFF
E79E  EB 53                                 JMP     SHORT A18       ; COM_STATUS
                                    ;------ SEND CHARACTER IN (AL) OVER COMMO LINE
E7A0                                A5:
E7A0  50                                    PUSH    AX              ; SAVE CHAR TO SEND
E7A1  83 C2 04                              ADD     DX,4            ; MODEM CONTROL REGISTER
E7A4  B0 03                                 MOV     AL,3            ; DTR AND RTS
E7A6  EE                                    OUT     DX,AL           ; DATA TERMINAL READY, REQUEST TO
                                                                    ; SEND
E7A7  42                                    INC     DX              ; MODEM STATUS REGISTER
E7A8  42                                    INC     DX
E7A9  B7 30                                 MOV     BH,30H          ; DATA SET READY & CLEAR TO SEND
E7AB  E8 E802 R                             CALL    WAIT_FOR_STATUS ; ARE BOTH TRUE?
E7AE  74 08                                 JE      A9              ; YES, READY TO TRANSMIT CHAR
E7B0  59                            A7:     POP     CX
E7B1  8A C1                                 MOV     AL,CL           ; RELOAD DATA BYTE
E7B3  80 CC 80                      A8:     OR      AH,80H          ; INDICATE TIME OUT
E7B6  EB AC                                 JMP     A3              ; RETURN
E7B8                                A9:                             ; CLEAR_TO_SEND
E7B8  4A                                    DEC     DX              ; LINE STATUS REGISTER
E7B9  B7 20                                 MOV     BH,20H          ; IS TRANSMITTER READY
E7BB  E8 E802 R                             CALL    WAIT_FOR_STATUS ; TEST FOR TRANSMITTER READY
E7BE  75 F0                                 JNZ     A7              ; RETURN WITH TIME OUT SET
E7C0  83 EA 05                              SUB     DX,5            ; DATA PORT
E7C3  59                                    POP     CX              ; RECOVER IN CX TEMPORARILY
E7C4  8A C1                                 MOV     AL,CL           ; MOVE CHAR TO AL FOR OUT, STATUS
                                                                    ; IN AH
E7C6  EE                                    OUT     DX,AL           ; OUTPUT CHARACTER
E7C7  EB 9B                                 JMP     A3              ; RETURN
                                    ;------ RECEIVE CHARACTER FROM COMMO LINE
E7C9  83 C2 04                      A12:    ADD     DX,4            ; MODEM CONTROL REGISTER
E7CC  B0 01                                 MOV     AL,1            ; DATA TERMINAL READY
E7CE  EE                                    OUT     DX,AL
E7CF  42                                    INC     DX              ; MODEM STATUS REGISTER
E7D0  42                                    INC     DX
E7D1  B7 20                                 MOV     BH,20H          ; DATA SET READY
E7D3  E8 E802 R                             CALL    WAIT_FOR_STATUS ; TEST FOR DSR
E7D6  75 DB                                 JNZ     A8              ; RETURN WITH ERROR
E7D8  4A                                    DEC     DX              ; LINE STATUS REGISTER
E7D9  EC                            A16:    IN      AL,DX
E7DA  A8 01                                 TEST    AL,1            ; RECEIVE BUFFER FULL
E7DC  75 09                                 JNZ     A17             ; TEST FOR REC. BUFF. FULL
E7DE  F6 06 0071 R 80                       TEST    BIOS_BREAK,80H  ; TEST FOR BREAK KEY
E7E3  74 F4                                 JZ      A16             ; LOOP IF NO BREAK KEY
E7E5  EB CC                                 JMP     A8              ; SET TIME OUT ERROR
E7E7  24 1E                         A17:    AND     AL,00011110B    ; TEST FOR ERROR CONDITIONS ON RECV
                                                                    ; CHAR
E7E9  8A E0                                 MOV     AH,AL
E7EB  8B 94 0000 R                          MOV     DX,RS232_BASE[SI] ; DATA PORT
E7EF  EC                                    IN      AL,DX           ; GET CHARACTER FROM LINE
E7F0  E9 E764 R                             JMP     A3              ; RETURN
                                    ;------ COMMO PORT STATUS ROUTINE
E7F3  8B 94 0000 R                  A18:    MOV     DX,RS232_BASE[SI]
E7F7  83 C2 05                              ADD     DX,5            ; CONTROL PORT
E7FA  EC                                    IN      AL,DX           ; GET LINE CONTROL STATUS
E7FB  8A E0                                 MOV     AH,AL           ; PUT IN AH FOR RETURN
E7FD  42                                    INC     DX              ; POINT TO MODEM STATUS REGISTER
E7FE  EC                                    IN      AL,DX           ; GET MODEM CONTROL STATUS
E7FF  E9 E764 R                             JMP     A3              ; RETURN
                                    ;--------------------------------------------
                                    ;       WAIT FOR STATUS ROUTINE
                                    ;ENTRY: BH=STATUS BIT(S) TO LOOK FOR,
                                    ;       DX=ADDR. OF STATUS REG
                                    ;EXIT:  ZERO FLAG ON = STATUS FOUND
                                    ;       ZERO FLAG OFF = TIMEOUT.
                                    ;       AH=LAST STATUS READ
                                    ;--------------------------------------------
```

```
E802                    WAIT_FOR_STATUS PROC    NEAR
E802  8A 9D 007C R              MOV     BL,RS232_TIM_OUT[DI] ;LOAD OUTER LOOP COUNT
E806  2B C9             WFS0:   SUB     CX,CX
E808  EC                WFS1:   IN      AL,DX           ;GET STATUS
E809  8A E0                     MOV     AH,AL           ; MOVE TO AH
E80B  22 C7                     AND     AL,BH           ;ISOLATE BITS TO TEST
E80D  3A C7                     CMP     AL,BH           ;EXACTLY = TO MASK
E80F  74 08                     JE      WFS_END         ;RETURN WITH ZERO FLAG ON
E811  E2 F5                     LOOP    WFS1            ;TRY AGAIN
E813  FE CB                     DEC     BL
E815  75 EF                     JNZ     WFS0
E817  0A FF                     OR      BH,BH           ;SET ZERO FLAG OFF
E819                    WFS_END:
E819  C3                        RET
E81A                    WAIT_FOR_STATUS ENDP
E81A                    RS232_IO        ENDP
                        ;--------------------------------------------------------------
                        ;THIS ROUTINE WILL READ TIMER1.  THE VALUE READ IS RETURNED IN AX.
                        ;--------------------------------------------------------------
E81A                    READ_TIME PROC NEAR
E81A  B0 40                     MOV     AL,40H          ; LATCH TIMER1
E81C  E6 43                     OUT     TIM_CTL,AL
E81E  50                        PUSH    AX              ; WAIT FOR 8253 TO INIT ITSELF
E81F  58                        POP     AX
E820  E4 41                     IN      AL,TIMER+1      ; READ LSB
E822  8A E0                     MOV     AH,AL           ; SAVE IT IN HIGH BYTE
E824  50                        PUSH    AX              ; WAIT FOR 8253 TO INIT ITSELF
E825  58                        POP     AX
E826  E4 41                     IN      AL,TIMER+1      ; READ MSB
E828  86 C4                     XCHG    AL,AH           ; PUT BYTES IN PROPER ORDER
E82A  C3                        RET
E82B                    READ_TIME       ENDP
E82E                            ORG     0E82EH
E82E  E9 13DD R                 JMP     NEAR PTR KEYBOARD_IO
                        ;--------------------------------------------------------------
                        ;ASYNCHRONOUS COMMUNICATIONS ADAPTER POWER ON DIAGNOSTIC TEST
                        ;DESCRIPTION:
                        ;   THIS SUBROUTINE PERFORMS A THOROUGH CHECK OUT OF AN INS8250 LSI
                        ;   CHIP.
                        ;   THE TEST INCLUDES:
                        ;   1)  INITIALIZATION OF THE CHIP TO ASSUME ITS MASTER RESET STATE.
                        ;   2)  READING REGISTERS FOR KNOWN PERMANENT ZERO BITS.
                        ;   3)  TESTING THE INS8250 INTERRUPT SYSTEM AND THAT THE 8250
                        ;       INTERRUPTS TRIGGER AN 8259 (INTERRUPT CONTROLLER) INTERRUPT.
                        ;   4)  PERFORMING THE LOOP BACK TEST:
                        ;       A)  TESTING WHAT WAS WRITTEN/READ AND THAT THE TRANSMITTER
                        ;           HOLDING REG EMPTY BIT AND THE RECEIVER INTERRUPT WORK
                        ;           PROPERLY.
                        ;       B)  TESTING IF CERTAIN BITS OF THE DATA SET CONTROL REGISTER
                        ;           ARE 'LOOPED BACK' TO THOSE IN THE DATA SET STATUS
                        ;           REGISTER.
                        ;       C)  TESTING THAT THE TRANSMITTER IS IDLE WHEN TRANSMISSION
                        ;           TEST IS FINISHED.
                        ; THIS SUBROUTINE EXPECTS TO HAVE THE FOLLOWING PARAMETER PASSED:
                        ;    (DX)= ADDRESS OF THE INS8250 CARD TO TEST.
                        ;    NOTE: THE ASSUMPTION HAS BEEN MADE THAT THE MODEM ADAPTER IS
                        ;          ----  LOCATED AT 03F8H; THE SERIAL PRINTER AT 02F8H.
                        ; IT RETURNS:
                        ;  (CF) = 1 IF ANY PORTION OF THE TEST FAILED
                        ;       = 0 IF TEST PASSED
                        ;  (BX) = FAILURE KEY FOR ERROR MESSAGE (ONLY VALID IF TEST FAILED)
                        ;     (BH) = 23H  SERIAL PRINTER ADAPTER TEST FAILURE
                        ;          = 24H  MODEM ADAPTER TEST FAILURE
                        ;     (BL) = 2   PERMANENT ZERO BITS IN INTERRUPT ENABLE REGISTER
                        ;                WERE INCORRECT
                        ;            3   PERMANENT ZERO BITS IN INTERRUPT IDENTIFICATION
                        ;                REGISTER WERE INCORRECT
                        ;            4   PERMANENT ZERO BITS IN DATA SET CONTROL REGISTER
                        ;                WERE INCORRECT
                        ;            5   PERMANENT ZERO BITS IN THE LINE STATUS REGISTER
                        ;                WERE INCORRECT
                        ;            6   RECEIVED DATA AVAILABLE INTERRUPT TEST FAILED
                        ;                (THE INTERRUPT WAS NOT GENERATED)
                        ;           16H  RECEIVED DATA AVAILABLE INTERRUPT FAILED TO CLEAR
                        ;            7   RESERVED FOR REPORTING THE TRANSMITTER HOLDING
                        ;                REGISTER EMPTY INTERRUPT TEST FAILED
                        ;                (NOT USED AT THIS TIME BECAUSE OF THE DIFFERENCES
                        ;                BETWEEN THE 8250'S WHICH WILL BE USED)
                        ;           17H  TRANSMITTER HOLDING REG EMPTY INTR FAILED TO CLEAR
                        ;           8-B  RECEIVER LINE STATUS INTERRUPT TEST FAILED
                        ;                (THE INTERRUPT WAS NOT GENERATED)
                        ;                8 - OVERRUN ERROR
                        ;                9 - PARITY ERROR
                        ;                A - FRAMING ERROR
                        ;                B - BREAK INTERRUPT ERROR
                        ;          18-1B RECEIVER LINE STATUS INTERRUPT FAILED TO CLEAR
                        ;           C-F  MODEM STATUS INTERRUPT TEST FAILED
                        ;                (THE INTERRUPT WAS NOT GENERATED)
                        ;                C - DELTA CLEAR TO SEND ERROR
                        ;                D - DELTA DATA SET READY ERROR
                        ;                E - TRAILING EDGE RING INDICATOR ERROR
                        ;                F - DELTA RECEIVE LINE SIGNAL DETECT ERROR
```

```
                        ;           1C-1F MODEM STATUS INTERRUPT FAILED TO CLEAR
                        ;            10H  AN 8250 INTERRUPT OCCURRED AS EXPECTED, BUT NO
                        ;                 8259 (INTR CONTROLLER) INTERRUPT WAS GENERATED
                        ;            11H  DURING THE TRANSMISSION TEST, THE TRANSMITTER
                        ;                 HOLDING REGISTER WAS NOT EMPTY WHEN IT SHOULD
                        ;                 HAVE BEEN.
                        ;            12H  DURING THE TRANSMISSION TEST, THE RECEIVED DATA
                        ;                 AVAILABLE INTERRUPT DIDN'T OCCUR.
                        ;            13H  TRANSMISSION ERROR - THE CHARACTER RECEIVED
                        ;                 DURING LOOP MODE WAS NOT THE SAME AS THE ONE
                        ;                 TRANSMITTED
                        ;            14H  DURING TRANSMISSION TEST, THE 4 DATA SET CONTROL
                        ;                 OUTPUTS WERE NOT THE SAME AS THE 4 DATA SET
                        ;                 CONTROL INPUTS.
                        ;            15H  THE TRANSMITTER WAS NOT IDLE AFTER THE TRANS-
                        ;                 MISSION TEST COMPLETED.
                        ;    ON EXIT:
                        ;        - THE MODEM OR SERIAL PRINTER'S 8259 INTERRUPT (WHICHEVER
                        ;          DEVICE WAS TESTED) IS DISABLED.
                        ;        - THE 8250 IS IN THE MASTER RESET STATE.
                        ;    ONLY THE DS REGISTER IS PRESERVED - ALL OTHERS ARE ALTERED.
                        ;--------------------------------------------------------------------
= 0084                  WRAP    EQU     84H             ; LOOP BACK TRANSMISSION TEST
                                                        ; INTERRUPT VECTOR ADDRESS
                                                        ; (IN DIAGNOSTICS)
                                ASSUME  CS:CODE,DS:DATA
E831                    UART    PROC    NEAR
E831  1E                        PUSH    DS
E832  E4 21                     IN      AL,INTA01       ; CURRENT ENABLED INTERRUPTS
E834  50                        PUSH    AX              ; SAVE FOR EXIT
E835  0C 01                     OR      AL,00000001B    ; DISABLE TIMER INTR DURING THIS
                                                        ; TEST
E837  E6 21                     OUT     INTA01,AL
E839  9C                        PUSHF                   ; SAVE CALLER'S FLAGS (SAVE INTR
                                                        ; FLAG)
E83A  52                        PUSH    DX              ; SAVE BASE ADDRESS OF ADAPTER CARD
E83B  E8 138B R                 CALL    DDS             ; SET UP 'DATA' AS DATA SEGMENT
                                                        ; ADDRESS
                        ;--------------------------------------------------------------------
                        ;       INITIALIZE PORTS FOR MASTER RESET STATES AND TEST PERMANENT
                        ;       ZERO DATA BITS FOR CERTAIN PORTS.
                        ;--------------------------------------------------------------------
E83E  E8 0AC4 R                 CALL    I8250
E841  73 03                     JNC     AT1             ; ALL OK
E843  E9 E94B R                 JMP     AT14            ; A PORT'S ZERO BITS WERE NOT ZERO!
                        ;--------------------------------------------------------------------
                        ;       INS8250 INTERRUPT SYSTEM TEST
                        ;       ONLY THE INTERRUPT BEING TESTED WILL BE ENABLED.
                        ;--------------------------------------------------------------------
                        ;       SET DI AND SI FOR CALLS TO 'SUI'
E846  BF 0041 R         AT1:    MOV     DI,OFFSET IMASKS ; BASE ADDRESS OF INTERRUPT MASKS
E849  33 F6                     XOR     SI,SI           ; MODEM INDEX
E84B  80 FE 02                  CMP     DH,2            ; OR SERIAL?
E84E  75 02                     JNE     AT2             ; NO - IT'S MODEM
E850  46                        INC     SI              ; IT'S SERIAL PRINTER
E851  47                        INC     DI              ; SERIAL PRINTER 8259 MASK ADDRESS
                        ;       RECEIVED DATA AVAILABLE INTERRUPT TEST
E852  E8 E6F5 R         AT2:    CALL    SUI             ; SET UP FOR INTERRUPTS
E855  FE C3                     INC     BL              ; ERROR REPORTER (INIT. IN I8250)
E857  42                        INC     DX              ; POINT TO INTERRUPT ENABLE
                                                        ; REGISTER
E858  B0 01                     MOV     AL,1            ; ENABLE RECEIVED DATA AVAILABLE
                                                        ; INTR
E85A  EE                        OUT     DX,AL
E85B  53                        PUSH    BX              ; SAVE ERROR REPORTER
E85C  83 C2 04                  ADD     DX,4            ; POINT TO LINE STATUS REGISTER
E85F  B4 01                     MOV     AH,1            ; SET RECEIVER DATA READY BIT
E861  BB 0400                   MOV     BX,0400H        ; INTR TO CHECK, INTR IDENTIFIER
E864  B9 0003                   MOV     CX,3            ; INTERRUPT ID REG 'INDEX'
E867  E8 0AF8 R                 CALL    ICT             ; PERFORM TEST FOR INTERRUPT
E86A  5B                        POP     BX              ; RESTORE ERROR INDICATOR
E86B  3C FF                     CMP     AL,0FFH         ; INTERRUPT ERROR OCCUR?
E86D  74 36                     JE      AT4             ; YES
E86F  E8 E706 R                 CALL    C5059           ; GENERATE 8259 INTERRUPT?
E872  72 33                     JC      AT5             ; NO
E874  4A                        DEC     DX
E875  4A                        DEC     DX              ; RESET INTR BY READING RECR BUFR
E876  EC                        IN      AL,DX           ; DON'T CARE ABOUT THE CONTENTS!
E877  42                        INC     DX
E878  42                        INC     DX              ; INTR ID REG
E879  E8 E719 R                 CALL    W8250C          ; WAIT FOR INTR TO CLEAR
E87C  73 03                     JNC     AT3             ; OK
E87E  E9 E94B R                 JMP     AT13            ; DIDN'T CLEAR
                        ;--------------------------------------------------------------------
                        ; TRANSMITTER HOLDING REGISTER EMPTY INTERRUPT TEST
                        ;       THIS TEST HAS BEEN MODIFIED BECAUSE THE DIFFERENT 8250'S
                        ;       THAT MAY BE USED IN PRODUCING THIS PRODUCT DO NOT FUNCTION
                        ;       THE SAME DURING THE STANDARD TEST OF THIS INTERRUPT
                        ;       (STANDARD BEING THE SAME METHOD FOR TESTING THE OTHER
                        ;       POSSIBLE 8250 INTERRUPTS).  IT IS STILL VALID FOR TESTING
                        ;       IF AN 8259 INTERRUPT IS GENERATED IN RESPONSE TO THE 8250
                        ;       INTERRUPT AND THAT THE 8250 INTERRUPT CLEARS AS IT SHOULD.
                        ;
                        ;       IF THE TRANSMITTER HOLDING REGISTER EMPTY INTERRUPT IS NOT
                        ;       GENERATED WHEN THAT INTERRUPT IS ENABLED, IT IS NOT TREATED
                        ;       AS AN ERROR.  HOWEVER, IF THE INTERRUPT IS GENERATED, IT
                        ;       MUST GENERATE AN 8259 INTERRUPT AND CLEAR PROPERLY TO PASS
                        ;       THIS TEST.
                        ;--------------------------------------------------------------------
```

```
E881  E8 E6F5 R          AT3:   CALL    SUI             ; SET UP FOR INTERRUPTS
E884  FE C3                     INC     BL              ; BUMP ERROR REPORTER
E886  4A                        DEC     DX              ; POINT TO INTERRUPT ENABLE
                                                        ; REGISTER
E887  B0 02                     MOV     AL,2            ; ENABLE XMITTER HOLDING REG EMPTY
                                                        ; INTR
E889  EE                        OUT     DX,AL
E88A  EB 00                     JMP     $+2             ; I/O DELAY
E88C  42                        INC     DX              ; INTR IDENTIFICATION REG
E88D  2B C9                     SUB     CX,CX
E88F  EC                 AT31:  IN      AL,DX           ; READ IT
E890  3C 02                     CMP     AL,2            ; XMITTER HOLDING REG EMPTY INTR?
E892  74 04                     JE      AT32            ; YES
E894  E2 F9                     LOOP    AT31
E896  EB 11                     JMP     SHORT AT6       ; THE INTR DIDN'T OCCUR - TRY NEXT
                                                        ; TEST
E898                     AT32:                          ; THE INTR DID OCCUR
E898  E8 E706 R                 CALL    C5059           ; GENERATE 8259 INTERRUPT?
E89B  72 0A                     JC      AT5             ; NO
E89D  E8 E719 R                 CALL    W8250C          ; WAIT FOR THE INTERRUPT TO CLEAR
                                                        ; (IT SHOULD ALREADY BE CLEAR
                                                        ; BECAUSE 'ICT' READ THE INTR ID
                                                        ; REG)
E8A0  73 07                     JNC     AT6             ; IT CLEARED
E8A2  E9 E948 R                 JMP     AT13            ; ERROR
E8A5  EB 7E              AT4:   JMP     SHORT AT11      ; AVOID OUT OF RANGE JUMPS
E8A7  EB 7A              AT5:   JMP     SHORT AT10
                        ;--------------------------------------------------------------
                        ; RECEIVER LINE STATUS INTERRUPT TEST
                        ;       THERE ARE 4 BITS WHICH COULD GENERATE THIS INTERRUPT.
                        ;       EACH ONE IS TESTED INDIVIDUALLY.
                        ;       WHEN:   AH      TESTING
                        ;               --      -------
                        ;                2      OVERRUN
                        ;                4      PARITY
                        ;                8      FRAMING
                        ;               10H     BREAK INTR
                        ;--------------------------------------------------------------
E8A9  4A                 AT6:   DEC     DX              ; POINT TO INTERRUPT ENABLE
                                                        ; REGISTER
E8AA  B0 04                     MOV     AL,4            ; ENABLE RECEIVER LINE STATUS INTR
E8AC  EE                        OUT     DX,AL
E8AD  83 C2 04                  ADD     DX,4            ; POINT TO LINE STATUS REGISTER
E8B0  B9 0003                   MOV     CX,3            ; INTR ID REG 'INDEX'
E8B3  BD 0004                   MOV     BP,4            ; LOOP COUNTER
E8B6  B4 02                     MOV     AH,2            ; INITIAL BIT TO BE TESTED
E8B8  E8 E6F5 R          AT7:   CALL    SUI             ; SET UP FOR INTERRUPTS
E8BB  FE C3                     INC     BL              ; BUMP ERROR REPORTER
E8BD  53                        PUSH    BX              ; SAVE IT
E8BE  BB 0601                   MOV     BX,0601H        ; INTR TO CHECK, INTR IDENTIFIER
E8C1  E8 0AF8 R                 CALL    ICT             ; PERFORM TEST FOR INTERRUPT
E8C4  5B                        POP     BX
E8C5  24 1E                     AND     AL,00011110B    ; MASK OUT BITS THAT DON'T MATTER
E8C7  3A C4                     CMP     AL,AH           ; TEST BIT ON?
E8C9  75 5A                     JNE     AT11            ; NO
E8CB  E8 E706 R                 CALL    C5059           ; GENERATE 8259 INTERRUPT?
E8CE  72 53                     JC      AT10            ; NO
E8D0  83 EA 03                  SUB     DX,3            ; INTR ID REG
E8D3  E8 E719 R                 CALL    W8250C          ; WAIT FOR THE INTR TO CLEAR
E8D6  72 70                     JC      AT13            ; IT DIDN'T
E8D8  4D                        DEC     BP              ; ALL FOUR BITS TESTED?
E8D9  74 07                     JE      AT8             ; YES - GO ON TO NEXT TEST
E8DB  D0 E4                     SHL     AH,1            ; GET READY FOR NEXT BIT
E8DD  83 C2 03                  ADD     DX,3            ; LINE STATUS REGISTER
E8E0  EB D6                     JMP     AT7             ; TEST NEXT BIT
                        ;--------------------------------------------------------------
                        ; MODEM STATUS INTERRUPT TEST
                        ;       THERE ARE 4 BITS WHICH COULD GENERATE THIS INTERRUPT.
                        ;       THEY ARE TESTED INDIVIDUALLY.
                        ;       WHEN:   AH      TESTING
                        ;               --      -------
                        ;                1      DELTA CLEAR TO SEND
                        ;                2      DELTA DATA SET READY
                        ;                4      TRAILING EDGE RING INDICATOR
                        ;                8      DELTA RECEIVE LINE SIGNAL DETECT
                        ;--------------------------------------------------------------
E8E2  83 C2 04           AT8:   ADD     DX,4            ; MODEM STATUS REGISTER
E8E5  EC                        IN      AL,DX           ; CLEAR DELTA BITS THAT MAY BE ON
                                                        ; BECAUSE OF DIFFERENCES AMONG
                                                        ; 8250'S.
E8E6  EB 00                     JMP     $+2             ; I/O DELAY
E8E8  83 EA 05                  SUB     DX,5            ; INTERRUPT ENABLE REGISTER
E8EB  B0 0B                     MOV     AL,8            ; ENABLE MODEM STATUS INTERRUPT
E8ED  EE                        OUT     DX,AL
E8EE  83 C2 05                  ADD     DX,5            ; POINT TO MODEM STATUS REGISTER
E8F1  B9 0004                   MOV     CX,4            ; INTR ID REG 'INDEX'
E8F4  BD 0004                   MOV     BP,4            ; LOOP COUNTER
E8F7  B4 01                     MOV     AH,1            ; INITIAL BIT TO BE TESTED
E8F9  E8 E6F5 R          AT9:   CALL    SUI             ; SET UP FOR INTERRUPTS
E8FC  FE C3                     INC     BL              ; BUMP ERROR INDICATOR
E8FE  53                        PUSH    BX              ; SAVE IT
E8FF  BB 0001                   MOV     BX,0001H        ; INTR TO CHECK, INTR IDENTIFIER
E902  E8 0AF8 R                 CALL    ICT             ; PERFORM TEST FOR INTERRUPT
E905  5B                        POP     BX
E906  24 0F                     AND     AL,00001111B    ; MASK OUT BITS THAT DON'T MATTER
E908  3A C4                     CMP     AL,AH           ; TEST BIT ON?
E90A  75 19                     JNE     AT11            ; NO
E90C  E8 E706 R                 CALL    C5059           ; GENERATE 8259 INTERRUPT?
E90F  72 12                     JC      AT10            ; NO
E911  83 EA 04                  SUB     DX,4            ; INTR ID REG
```

```
E914  E8 E719 R                 CALL    W8250C          ; WAIT FOR INTERRUPT TO CLEAR
E917  72 2F                     JC      AT13            ; IT DIDN'T
E919  4D                        DEC,    BP
E91A  74 0B                     JE      AT12            ; ALL FOUR BITS TESTED - GO ON
E91C  D0 E4                     SHL     AH,1            ; GET READY FOR NEXT BIT
E91E  83 C2 04                  ADD     DX,4            ; MODEM STATUS REGISTER
E921  EB D6                     JMP     AT9             ; TEST NEXT BIT
                        ;-------------------------------------------------------------------
                        ;       POSSIBLE 8259 INTERRUPT CONTROLLER PROBLEM
                        ;-------------------------------------------------------------------
E923  B3 10             AT10:   MOV     BL,10H          ; SET ERROR REPORTER
E925  EB 24             AT11:   JMP     SHORT AT14
                        ;-------------------------------------------------------------------
                        ;       SET 9600 BAUD RATE AND DEFINE DATA WORD AS HAVING 8
                        ;       BITS/WORD, 2 STOP BITS, AND ODD PARITY.
                        ;-------------------------------------------------------------------
E927  42                AT12:   INC     DX              ; LINE CONTROL REGISTER
E928  E8 F085 R                 CALL    S8250
                        ;-------------------------------------------------------------------
                        ;       SET DATA SET CONTROL WORD TO BE IN LOOP MODE
                        ;-------------------------------------------------------------------
E92B  83 C2 04                  ADD     DX,4
E92E  EC                        IN      AL,DX           ; CURRENT STATE
E92F  EB 00                     JMP     $+2             ; I/O DELAY
E931  0C 10                     OR      AL,00010000B    ; SET BIT 4 OF DATA SET CONTROL REG
E933  EE                        OUT     DX,AL
E934  EB 00                     JMP     $+2             ; I/O DELAY
E936  42                        INC     DX
E937  42                        INC     DX              ; MODEM STATUS REG
E938  EC                        IN      AL,DX           ; CLEAR POSSIBLE MODEM STATUS
                                                        ; INTERRUPT WHICH COULD BE CAUSED
                                                        ; BY THE OUTPUT BITS BEING LOOPED
                                                        ; TO THE INPUT BITS
E939  EB 00                     JMP     $+2             ; I/O DELAY
E93B  83 EA 06                  SUB     DX,6            ; RECEIVER BUFFER
E93E  EC                        IN      AL,DX           ; DUMMY READ TO CLEAR DATA READY
                                                        ; BIT IF IT WENT HIGH ON WRITE TO
                                                        ; MCR
                        ;-------------------------------------------------------------------
                        ;       PERFORM THE LOOP BACK TEST
                        ;-------------------------------------------------------------------
E93F  42                        INC     DX              ; INTR ENBL REG
E940  B0 00                     MOV     AL,0            ; SET FOR INTERNAL WRAP TEST
E942  CD 84                     INT     WRAP            ; DO LOOP BACK TRANSMISSION TEST
E944  B1 00                     MOV     CL,0            ; ASSUME NO ERRORS
E946  73 05                     JNC     AT15            ; WRAP TEST PASSED
E948  80 C3 10          AT13:   ADD     BL,10H          ; ERROR INDICATOR
                        ;-------------------------------------------------------------------
                        ;       AN ERROR WAS ENCOUNTERED SOMEWHERE DURING THE TEST
                        ;-------------------------------------------------------------------
E94B  B1 01             AT14:   MOV     CL,1            ; SET FAIL INDICATOR
                        ;-------------------------------------------------------------------
                        ;       HOUSEKEEPING:  RE-INITIALIZE THE 8250 PORTS (THE LOOP BIT
                        ;       WILL BE RESET), DISABLE THIS DEVICE INTERRUPT, SET UP
                        ;       REGISTER BH IF AN ERROR OCCURRED, AND SET OR RESET THE
                        ;       CARRY FLAG.
                        ;-------------------------------------------------------------------
E94D  5A                AT15:   POP     DX              ; GET BASE ADDRESS OF 8250 ADAPTER
E94E  53                        PUSH    BX              ; SAVE ERROR CODE
E94F  E8 0AC4 R                 CALL    I8250           ; RE-INITIALIZE 8250 PORTS
E952  5B                        POP     BX
E953  2E: 8A 25                 MOV     AH,CS:[DI]      ; GET DEVICE INTERRUPT MASK
E956  20 26 0084 R              AND     INTR_FLAG,AH    ; CLEAR DEVICE'S INTERRUPT FLAG BIT
E95A  80 F4 FF                  XOR     AH,0FFH         ; FLIP BITS
E95D  E4 21                     IN      AL,INTA01       ; GET CURRENT INTERRUPT PORT
E95F  0A C4                     OR      AL,AH           ; DISABLE THIS DEVICE INTERRUPT
E961  E6 21                     OUT     INTA01,AL
E963  9D                        POPF                    ; RE-ESTABLISH CALLER'S INTERRUPT
                                                        ; FLAG
E964  0A C9                     OR      CL,CL           ; ANY ERRORS?
E966  74 0C                     JE      AT17            ; NO
E968  B7 24                     MOV     BH,24H          ; ASSUME MODEM ERROR
E96A  80 FE 02                  CMP     DH,2            ; OR IS IT SERIAL?
E96D  75 02                     JNE     AT16            ; IT'S MODEM
E96F  B7 23                     MOV     BH,23H          ; IT'S SERIAL PRINTER
E971  F9                AT16:   STC                     ; SET CARRY FLAG TO INDICATE ERROR
E972  EB 01                     JMP     SHORT AT18
E974  F8                AT17:   CLC                     ; RESET CARRY FLAG - NO ERRORS
E975  58                AT18:   POP     AX              ; RESTORE ENTRY ENABLED INTERRUPTS
E976  E6 21                     OUT     INTA01,AL       ; DEVICE INTRS RE-ESTABLISHED
E978  1F                        POP     DS              ; RESTORE REGISTER
E979  C3                        RET
E97A                    UART    ENDP
E987                            ORG     0E987H
E987  E9 1561 R                 JMP     NEAR PTR KB_INT
                        ;--------------------------------------------------
                        ; NEC_OUTPUT
                        ;       THIS ROUTINE SENDS A BYTE TO THE NEC CONTROLLER
                        ;       AFTER TESTING FOR CORRECT DIRECTION AND CONTROLLER READY
                        ;       THIS ROUTINE WILL TIME OUT IF THE BYTE IS NOT ACCEPTED
                        ;       WITHIN A REASONABLE AMOUNT OF TIME, SETTING THE DISKETTE
                        ;       STATUS ON COMPLETION
                        ; INPUT
                        ;       (AH)    BYTE TO BE OUTPUT
                        ; OUTPUT
                        ;       CY = 0  SUCCESS
                        ;       CY = 1  FAILURE -- DISKETTE STATUS UPDATED
                        ;               IF A FAILURE HAS OCCURRED, THE RETURN IS MADE ONE
                        ;               LEVEL HIGHER THAN THE CALLER OF NEC_OUTPUT
                        ;               THIS REMOVES THE REQUIREMENT OF TESTING AFTER EVERY
                        ;               CALL OF NEC_OUTPUT
                        ;       (AL) DESTROYED
                        ;--------------------------------------------------
```

```
E98A                                ;
E98A                    NEC_OUTPUT      PROC    NEAR
E98A  52                                PUSH    DX              ; SAVE REGISTERS
E98B  51                                PUSH    CX
E98C  BA 00F4                           MOV     DX,NEC_STAT     ; STATUS PORT
E98F  33 C9                             XOR     CX,CX           ; COUNT FOR TIME OUT
E991  EC                        J23:    IN      AL,DX           ; GET STATUS
E992  A8 40                             TEST    AL,DIO          ; TEST DIRECTION BIT
E994  74 0C                             JZ      J25             ; DIRECTION OK
E996  E2 F9                             LOOP    J23
E998                            J24:                            ; TIME_ERROR
E998  80 0E 0041 R 80                   OR      DISKETTE_STATUS,TIME_OUT
E99D  59                                POP     CX
E99E  5A                                POP     DX              ; SET ERROR CODE AND RESTORE REGS
E99F  58                                POP     AX              ; DISCARD THE RETURN ADDRESS
E9A0  F9                                STC                     ; INDICATE ERROR TO CALLER
E9A1  C3                                RET                     ;
E9A2  33 C9                     J25:    XOR     CX,CX           ; RESET THE COUNT
E9A4  EC                        J26:    IN      AL,DX           ; GET THE STATUS
E9A5  A8 80                             TEST    AL,RQM          ; IS IT READY?
E9A7  75 04                             JNZ     J27             ; YES, GO OUTPUT
E9A9  E2 F9                             LOOP    J26             ; COUNT DOWN AND TRY AGAIN
E9AB  EB EB                             JMP     J24             ; ERROR CONDITION
E9AD                            J27:                            ; OUTPUT
E9AD  8A C4                             MOV     AL,AH           ; GET BYTE TO OUTPUT
E9AF  42                                INC     DX              ; DATA PORT IS 1 GREATER THAN
                                                                ; STATUS PORT
E9B0  EE                                OUT     DX,AL           ; OUTPUT THE BYTE
E9B1  59                                POP     CX              ; RECOVER REGISTERS
E9B2  5A                                POP     DX
E9B3  C3                                RET                     ; CY = 0 FROM TEST INSTRUCTION
E9B4                            NEC_OUTPUT      ENDP
                                ;--------------------------------------
                                ; GET_PARM
                                ;  THIS ROUTINE FETCHES THE INDEXED POINTER FROM
                                ;  THE DISK_BASE BLOCK POINTED AT BY THE DATA
                                ;  VARIABLE DISK_POINTER
                                ; A BYTE FROM THAT TABLE IS THEN MOVED INTO AH,
                                ;  THE INDEX OF THAT BYTE BEING THE PARM IN BX
                                ; ENTRY --
                                ;   BL = INDEX OF BYTE TO BE FETCHED * 2
                                ;        IF THE LOW BIT OF BL IS ON, THE BYTE IS IMMEDIATELY
                                ;        OUTPUT TO THE NEC CONTROLLER
                                ; EXIT --
                                ;   AH = THAT BYTE FROM BLOCK
                                ;   BX = DESTROYED
                                ;--------------------------------------------
E9B4                            GET_PARM        PROC    NEAR
E9B4  1E                                PUSH    DS              ; SAVE SEGMENT
E9B5  56                                PUSH    SI              ; SAVE REGISTER
E9B6  2B C0                             SUB     AX,AX           ; ZERO TO AX
E9B8  32 FF                             XOR     BH,BH           ; ZERO BH
E9BA  8E D8                             MOV     DS,AX
                                        ASSUME  DS:ABSO
E9BC  C5 36 0078 R                      LDS     SI,DISK_POINTER ; POINT TO BLOCK
E9C0  D1 EB                             SHR     BX,1            ; DIVIDE BX BY 2, AND SET FLAG FOR
                                                                ; EXIT
E9C2  9C                                PUSHF                   ; SAVE OUTPUT BIT
E9C3  8A 20                             MOV     AH,[SI+BX]      ; GET THE BYTE
E9C5  83 FB 01                          CMP     BX,1            ; IS THIS THE PARM WITH DMA
                                                                ; INDICATOR
E9C8  75 05                             JNZ     J27_1
E9CA  80 CC 01                          OR      AH,1            ; TURN ON NO DMA BIT
E9CD  EB 0C                             JMP     SHORT J27_2
E9CF  83 FB 0A                  J27_1:  CMP     BX,10           ; MOTOR STARTUP DELAY?
E9D2  75 07                             JNE     J27_2
E9D4  80 FC 04                          CMP     AH,4            ; GREATER THAN OR EQUAL TO 1/2 SEC?
E9D7  7D 02                             JGE     J27_2           ; YES, OKAY
E9D9  B4 04                             MOV     AH,4            ; NO, FORCE 1/2 SECOND DELAY
E9DB  9D                        J27_2:  POPF                    ; GET OUTPUT BIT
E9DC  5E                                POP     SI              ; RESTORE REGISTER
E9DD  1F                                POP     DS              ; RESTORE SEGMENT
                                        ASSUME  DS:DATA
E9DE  72 AA                             JC      NEC_OUTPUT      ; IF FLAG SET, OUTPUT TO CONTROLLER
E9E0  C3                                RET                     ; RETURN TO CALLER
E9E1                            GET_PARM        ENDP
                                ;--------------------------------------------------------
                                ; BOUND_SETUP
                                ;       THIS ROUTINE SETS UP BUFFER ADDRESSING FOR READ/WRITE/VERIFY
                                ;       OPERATIONS.
                                ; INPUT
                                ;       ES HAS ORIGINAL BUFFER SEGMENT VALUE
                                ;       BP POINTS AT BASE OF SAVED PARMETERS ON STACK
                                ; OUTPUT
                                ;       ES HAS SEGMENT WHICH WILL ALLOW 64K ACCESS.  THE
                                ;       COMBINATION ES:DI  AND DS:SI POINT TO THE BUFFER. THIS
                                ;       CALCULATED ADDRESS WILL ALWAYS ACCESS 64K OF MEMORY.
                                ;       BX DESTOYED
                                ;--------------------------------------------------------
```

```
E9E1                    BOUND_SETUP     PROC    NEAR
E9E1  51                        PUSH    CX                      ; SAVE REGISTERS
E9E2  8B 5E 0C                  MOV     BX,[BP+12]              ; GET OFFSET OF BUFFER FROM STACK
E9E5  53                        PUSH    BX                      ; SAVE OFFSET TEMPORARILY
E9E6  B1 04                     MOV     CL,4                    ; SHIFT COUNT
E9E8  D3 EB                     SHR     BX,CL                   ; SHIFT OFFSET FOR NEW SEGMENT
                                                                ; VALUE
E9EA  8C C1                     MOV     CX,ES                   ; PUT ES IN REGISTER SUITABLE FOR
                                                                ; ADDING TO
E9EC  03 CB                     ADD     CX,BX                   ; GET NEW VALUE FOR ES
E9EE  8E C1                     MOV     ES,CX                   ; UPDATE THE ES REGISTER
E9F0  5B                        POP     BX                      ; RECOVER ORIGINAL OFFSET
E9F1  81 E3 000F                AND     BX,0000FH               ; NEW OFFSET
E9F5  8B F3                     MOV     SI,BX                   ; DS:SI POINT AT BUFFER
E9F7  8B FB                     MOV     DI,BX                   ; ES:DI POINT AT BUFFER
E9F9  59                        POP     CX
E9FA  C3                        RET
E9FB                    BOUND_SETUP     ENDP
                        ;--------------------------------------------------
                        ; SEEK
                        ;       THIS ROUTINE WILL MOVE THE HEAD ON THE NAMED DRIVE
                        ;       TO THE NAMED TRACK.  IF THE DRIVE HAS NOT BEEN ACCESSED
                        ;       SINCE THE DRIVE RESET COMMAND WAS ISSUED, THE DRIVE WILL BE
                        ;       RECALIBRATED.
                        ; INPUT
                        ;       (DL) = DRIVE TO SEEK ON
                        ;       (CH) = TRACK TO SEEK TO
                        ; OUTPUT
                        ;       CY = 0 SUCCESS
                        ;       CY = 1 FAILURE -- DISKETTE_STATUS SET ACCORDINGLY
                        ;       (AX) DESTROYED
                        ;--------------------------------------------------------
E9FB                    SEEK    PROC    NEAR
E9FB  56                        PUSH    SI                      ; SAVE REGISTER
E9FC  53                        PUSH    BX                      ; SAVE REGISTER
E9FD  51                        PUSH    CX
E9FE  BE 0074 R                 MOV     SI,OFFSET TRACK0 ; BASE OF CURRENT HEAD POSITIONS
EA01  B0 01                     MOV     AL,1                    ; ESTABLISH MASK FOR RECAL
EA03  8A CA                     MOV     CL,DL                   ; USE DRIVE AS A SHIFT COUNT
EA05  81 E1 00FF                AND     CX,0FFH                 ; MASK OFF HIGH BYTE
EA09  03 F1                     ADD     SI,CX                   ; POINT SI AT CORRECT DRIVE
EA0B  D2 C0                     ROL     AL,CL                   ; GET MASK FOR DRIVE
                        ;------ SI CONTAINS OFFSET FOR CORRECT DRIVE, AL CONTAINS BIT MASK
                        ;       IN POSITION 0,1 OR 2
EA0D  59                        POP     CX                      ; RESTORE PARAMETER REGISTER
EA0E  8B EA66 R                 MOV     BX,OFFSET J32   ; SET UP ERROR RECOVERY ADDRESS
EA11  53                        PUSH    BX                      ; NEEDED FOR ROUTINE NEC_OUTPUT
EA12  84 06 003E R              TEST    SEEK_STATUS,AL  ; TEST DRIVE FOR RECAL
EA16  75 1B                     JNZ     J28                     ; NO_RECAL
EA18  08 06 003E R              OR      SEEK_STATUS,AL  ; TURN ON THE NO RECAL BIT IN FLAG
EA1C  80 3C 00                  CMP     BYTE PTR[SI],0  ; LAST REFERENCED TRACK=0?
EA1F  74 12                     JZ      J28                     ; YES IGNORE RECAL
EA21  B4 07                     MOV     AH,07H                  ; RECALIBRATE COMMAND
EA23  E8 E98A R                 CALL    NEC_OUTPUT
EA26  8A E2                     MOV     AH,DL                   ; RECAL REQUIRED ON DRIVE IN DL
EA28  E8 E98A R                 CALL    NEC_OUTPUT      ; OUTPUT THE DRIVE NUMBER
                        ;------ HEAD IS MOVING TO CORRECT TRACK
EA2B  E8 EA6F R                 CALL    CHK_STAT_2      ; GET THE STATUS OF RECALIBRATE
EA2E  72 39                     JC      J32_2           ; SEEK_ERROR
EA30  C6 04 00                  MOV     BYTE PTR[SI],0
                        ;----- DRIVE IS IN SYNCH WITH CONTROLLER, SEEK TO TRACK
EA33  8A 04             J28:    MOV     AL,BYTE PTR[SI] ; GET THE PCN
EA35  2A C5                     SUB     AL,CH           ; GET SEEK_WAIT VALUE
EA37  74 2C                     JZ      J31_1           ; ALREADY ON CORRECT TRACK
EA39  B4 0F                     MOV     AH,0FH          ; SEEK COMMAND TO NEC
EA3B  E8 E98A R                 CALL    NEC_OUTPUT
EA3E  8A E2                     MOV     AH,DL           ; DRIVE NUMBER
EA40  E8 E98A R                 CALL    NEC_OUTPUT
EA43  8A E5                     MOV     AH,CH           ; TRACK NUMBER
EA45  E8 E98A R                 CALL    NEC_OUTPUT
EA48  E8 EA6F R                 CALL    CHK_STAT_2      ; GET ENDING INTERRUPT AND SENSE
                                                        ; STATUS
                        ;----- WAIT FOR HEAD SETTLE
EA4B  9C                        PUSHF                   ; SAVE STATUS FLAGS
EA4C  51                        PUSH    CX              ; SAVE REGISTER
EA4D  B3 12                     MOV     BL,18           ; HEAD SETTLE PARAMETER
EA4F  E8 E984 R                 CALL    GET_PARM
EA52                    J29:                            ; HEAD_SETTLE
EA52  B9 0226                   MOV     CX,550          ; 1 MS LOOP
EA55  0A E4                     OR      AH,AH           ; TEST FOR TIME EXPIRED
EA57  74 06                     JZ      J31
EA59  E2 FE             J30:    LOOP    J30             ; DELAY FOR 1 MS
EA5B  FE CC                     DEC     AH              ; DECREMENT THE COUNT
EA5D  EB F3                     JMP     J29             ; DO IT SOME MORE
EA5F  59                J31:    POP     CX              ; RESTORE REGISTER
EA60  9D                        POPF
EA61  72 06                     JC      J32_2
EA63  88 2C                     MOV     BYTE PTR[SI],CH
EA65  5B                J31_1:  POP     BX              ; GET RID OF DUMMY RETURN
EA66                    J32:                            ; SEEK_ERROR
EA66  5B                        POP     BX              ; RESTORE REGISTER
EA67  5E                        POP     SI              ; UPDATE CORRECT
EA68  C3                        RET                     ; RETURN TO CALLER
EA69  C6 04 FF          J32_2:  MOV     BYTE PTR[SI],0FFH ; UNKNOWN STATUS ABOUT SEEK
                                                        ; OPERATION
EA6C  5B                        POP     BX              ; GET RID OF DUMMY RETURN
EA6D  EB F7                     JMP     SHORT J32
EA6F                    SEEK    ENDP
```

```
                                ;--------------------------------------------------
                                ; CHK_STAT_2
                                ;       THIS ROUTINE HANDLES THE INTERRUPT RECEIVED AFTER
                                ;       A RECALIBRATE, SEEK, OR RESET TO THE ADAPTER.
                                ;       THE INTERRUPT IS WAITED FOR, THE INTERRUPT STATUS SENSED,
                                ;       AND THE RESULT RETURNED TO THE CALLER.
                                ; INPUT
                                ;       NONE
                                ; OUTPUT
                                ;       CY = 0 SUCCESS
                                ;       CY = 1 FAILURE -- ERROR IS IN DISKETTE_STATUS
                                ;       (AX) DESTROYED
                                ;--------------------------------------------------
EA6F                            CHK_STAT_2      PROC    NEAR
EA6F  53                                PUSH    BX                      ; SAVE REGISTERS
EA70  56                                PUSH    SI
EA71  33 DB                             XOR     BX,BX                   ; NUMBER OF SENSE INTERRUPTS TO
                                                                        ; ISSUE
EA73  BE EA88 R                         MOV     SI,OFFSET J33_3 ; SET UP DUMMY RETURN FROM
                                                                        ; NEC_OUTPUT
EA76  56                                PUSH    SI                      ; PUT ON STACK
EA77  B4 08                     J33_2:  MOV     AH,08H                  ; SENSE INTERUPT STATUS
EA79  E8 E98A R                         CALL    NEC_OUTPUT              ; ISSUE SENSE INTERUPT STATUS
EA7C  E8 EAA0 R                         CALL    RESULTS                 ;
EA7F  72 10                             JC      J35                     ; NEC TIME OUT, FLAGS SET IN
                                                                        ; RESULTS
EA81  A0 0042 R                         MOV     AL,NEC_STATUS   ; GET STATUS
EA84  A8 20                             TEST    AL,SEEK_END     ; IS SEEK OR RECAL OPERATION DONE?
EA86  75 0D                             JNZ     J35_1                   ; JUMP IF EXECUTION OF SEEK OR
                                                                        ; RECAL DONE
EA88  4B                        J33_3:  DEC     BX                      ; DEC LOOP COUNTER
EA89  75 EC                             JNZ     J33_2                   ; DO ANOTHER LOOP
EA8B  80 0E 0041 R 80                   OR      DISKETTE_STATUS,TIME_OUT
EA90  F9                        J34:    STC                             ; RETURN ERROR INDICATION FOR
                                                                        ; CALLER
EA91  5E                        J35:    POP     SI                      ; RESTORE REGISTERS
EA92  5E                                POP     SI
EA93  5B                                POP     BX
EA94  C3                                RET
                                ;-----SEEK END HAS OCCURED, CHECK FOR NORMAL TERMINATION
EA95  24 C0                     J35_1:  AND     AL,0C0H                 ; MASK NORMAL TERMINATION BITS
EA97  74 F8                             JZ      J35                     ; JUMP IF NORMAL TERMINATION
EA99  80 0E 0041 R 40                   OR      DISKETTE_STATUS,BAD_SEEK
EA9E  EB F0                             JMP     J34
EAA0                            CHK_STAT_2      ENDP
                                ;--------------------------------------------
                                ; RESULTS
                                ;       THIS ROUTINE WILL READ ANYTHING THAT THE NEC CONTROLLER
                                ;       HAS TO SAY FOLLOWING AN INTERRUPT.
                                ;       IT IS ASSUMED THAT THE NEC DATA PORT = NEC STATUS PORT + 1.
                                ; INPUT
                                ;       NONE
                                ; OUTPUT
                                ;       CY = 0  SUCCESSFUL TRANSFER
                                ;       CY = 1  FAILURE -- TIME OUT IN WAITING FOR STATUS
                                ;       NEC_STATUS AREA HAS STATUS BYTE LOADED INTO IT
                                ;       (AH) DESTROYED
                                ;--------------------------------------------
EAA0                            RESULTS PROC    NEAR
EAA0  FC                                CLD
EAA1  BF 0042 R                         MOV     DI,OFFSET NEC_STATUS ; POINTER TO DATA AREA
EAA4  51                                PUSH    CX                      ; SAVE COUNTER
EAA5  52                                PUSH    DX
EAA6  53                                PUSH    BX
EAA7  B3 07                             MOV     BL,7                    ; MAX STATUS BYTES
                                ;------ WAIT FOR REQUEST FOR MASTER
EAA9                            J38:                                    ; INPUT_LOOP
EAA9  33 C9                             XOR     CX,CX                   ; COUNTER
EAAB  BA 00F4                           MOV     DX,NEC_STAT             ; STATUS PORT
EAAE                            J39:                                    ; WAIT_FOR_MASTER
EAAE  EC                                IN      AL,DX                   ; GET STATUS
EAAF  A8 80                             TEST    AL,080H                 ; MASTER READY
EAB1  75 0C                             JNZ     J40A                    ; TEST_DIR
EAB3  E2 F9                             LOOP    J39                     ; WAIT_MASTER
EAB5  80 0E 0041 R 80                   OR      DISKETTE_STATUS,TIME_OUT
EABA                            J40:                                    ; RESULTS_ERROR
EABA  F9                                STC                             ; SET ERROR RETURN
                                ;------ RESULT OPERATION IS DONE
EABB  5B                        J44:    POP     BX
EABC  5A                                POP     DX
EABD  59                                POP     CX
EABE  C3                                RET
                                ;------ TEST THE DIRECTION BIT
EABF  EC                        J40A:   IN      AL,DX                   ; GET STATUS REG AGAIN
EAC0  A8 40                             TEST    AL,040H                 ; TEST DIRECTION BIT
EAC2  75 07                             JNZ     J42                     ; OK TO READ STATUS
EAC4                            J41:                                    ; NEC_FAIL
EAC4  80 0E 0041 R 20                   OR      DISKETTE_STATUS,BAD_NEC
EAC9  EB EF                             JMP     J40                     ; RESULTS_ERROR
                                ;------ READ IN THE STATUS
EACB                            J42:                                    ; INPUT_STAT
EACB  42                                INC     DX                      ; POINT AT DATA PORT
EACC  EC                                IN      AL,DX                   ; GET THE DATA
EACD  88 05                             MOV     [DI],AL                 ; STORE THE BYTE
EACF  47                                INC     DI                      ; INCREMENT THE POINTER
EAD0  B9 000A                           MOV     CX,10                   ; LOOP TO KILL TIME FOR NEC
EAD3  E2 FE                     J43:    LOOP    J43
EAD5  4A                                DEC     DX                      ; POINT AT STATUS PORT
EAD6  EC                                IN      AL,DX                   ; GET STATUS
EAD7  A8 10                             TEST    AL,010H                 ; TEST FOR NEC STILL BUSY
EAD9  74 E0                             JZ      J44                     ; RESULTS DONE
EADB  FE CB                             DEC     BL                      ; DECREMENT THE STATUS COUNTER
EADD  75 CA                             JNZ     J38                     ; GO BACK FOR MORE
EADF  EB E3                             JMP     J41                     ; CHIP HAS FAILED
```

```
                        ;--------------------------------------------------
                        ; NUM_TRANS
                        ;       THIS ROUTINE CALCULATES THE NUMBER OF SECTORS THAT
                        ;       WERE ACTUALLY TRANSFERRED TO/FROM THE DISKETTE
                        ; INPUT
                        ;       (CH) = CYLINDER OF OPERATION
                        ;       (CL) = START SECTOR OF OPERATION
                        ; OUTPUT
                        ;       (AL) = NUMBER ACTUALLY TRANSFERRED
                        ;       NO OTHER REGISTERS MODIFIED
                        ;--------------------------------------------------
EAE1                    NUM_TRANS       PROC    NEAR
EAE1  A0 0045 R                 MOV     AL,NEC_STATUS+3 ; GET CYLINDER ENDED UP ON
EAE4  3A 46 08                  CMP     AL,[BP+11]      ; SAME AS WE STARTED
EAE7  A0 0047 R                 MOV     AL,NEC_STATUS+5 ; GET ENDING SECTOR
EAEA  74 07                     JZ      J45             ; IF ON SAME CYL, THEN NO ADJUST
EAEC  B3 08                     MOV     BL,8
EAEE  E8 E9B4 R                 CALL    GET_PARM        ; GET EOT VALUE
EAF1  8A C4                     MOV     AL,AH           ;  INTO AL
EAF3  FE C0             J45:    INC     AL              ; USE EOT+1 FOR CALCULATION
EAF5  2A 46 0A                  SUB     AL,[BP]+10      ; SUBTRACT START FROM END
EAF8  88 46 0E                  MOV     [BP+14],AL
EAFB  C3                        RET
EAFC                    NUM_TRANS       ENDP
EAFC                    RESULTS ENDP
                        ;--------------------------------------------------
                        ; DISABLE
                        ;       THIS ROUTINE WILL DISABLE ALL INTERRUPTS EXCEPT FOR
                        ;       INTERRUPT 6 SO WATCH DOG TIME OUT CAN OCCUR IN ERROR
                        ;       CONDITIONS.
                        ; INPUT
                        ;       NONE
                        ; OUTPUT
                        ;       NONE
                        ;       ALL REGISTERS REMAIN INTACT
                        ;--------------------------------------------------
EAFC                    DISABLE         PROC    NEAR
EAFC  50                        PUSH    AX
                        ;------ DISABLE ALL INTERRUPTS AT THE 8259 LEVEL EXCEPT DISKETTE
EAFD  E4 21                     IN      AL,INTA01       ; READ CURRENT MASK
EAFF  89 46 10                  MOV     [BP+16],AX      ; SAVE MASK ON THE SPACE ALLOCATED
                                                        ; ON THE STACK
EB02  B0 BF                     MOV     AL,0BFH         ; MASK OFF ALL INTERRUPTS EXCEPT
                                                        ; DISKETTE
EB04  E6 21                     OUT     INTA01,AL       ; OUTPUT MASK TO THE 8259
EB06  E8 E9E1 R                 CALL    BOUND_SETUP     ; SETUP REGISTERS TO ACCESS BUFFER
EB09  58                        POP     AX
EB0A  C3                        RET
EB0B                    DISABLE         ENDP
                        ;--------------------------------------------------
                        ; ENABLE
                        ;       THIS PROC ENABLES ALL INTERRUPTS.  IT ALSO SETS THE 8253 TO
                        ;       THE MODE REQUIRED FOR KEYBOARD DATA DESERIALIZATION.
                        ;       BEFORE THE LATCH FOR KEYBOARD DATA IS RESET, BIT 0 OF THE
                        ;       8255 IS READ TO DETERMINE WHETHER ANY KEYSTROKES OCCURED
                        ;       WHILE THE SYSTEM WAS MASKED OFF.
                        ; INPUT
                        ;       NONE
                        ; OUTPUT
                        ;       AL=1 MEANS A KEY WAS STRUCK DURING DISKETTE I/O.  (OR NOISE
                        ;            ON THE LINE)
                        ;       AL=0 MEANS THAT NO KEY WAS PRESSED.
                        ;       AX IS DESTROYED.  ALL OTHER REGISTERS REMAIN INTACT.
                        ;--------------------------------------------------
EB0B                    ENABLE          PROC    NEAR
EB0B  52                        PUSH    DX              ; SAVE DX
                        ;------ RETURN TIMER1 TO STATE NEEDED FOR KEYBOARD I/O
EB0C  B0 76                     MOV     AL,01110110B    ;
EB0E  E6 43                     OUT     TIM_CTL,AL
EB10  50                        PUSH    AX
EB11  58                        POP     AX              ; WAIT FOR 8253 TO INITIALIZE
                                                        ; ITSELF
EB12  B0 FF                     MOV     AL,0FFH         ; INITIAL VALUE FOR 8253
EB14  E6 41                     OUT     TIMER+1,AL      ; LSB
EB16  50                        PUSH    AX
EB17  58                        POP     AX              ; WAIT
EB18  E6 41                     OUT     TIMER+1,AL      ; MSB
                        ;------ CHECK IF ANY KEYSTROKES OCCURED DURING DISKETTE TRANSFER
EB1A  8E 46 10                  MOV     ES,[BP+16]      ; GET ORIGINAL ES VALUE FROM THE
                                                        ; STACK
EB1D  E4 62                     IN      AL,62H          ; READ PORT C OF 8255
EB1F  24 01                     AND     AL,01H          ; BIT=1 MEANS KESTROKE HAS OCCURED
EB21  50                        PUSH    AX              ; SAVE IT ON THE STACK
                        ;------ ENABLE NMI INTERRUPTS
EB22  E4 A0                     IN      AL,NMI_PORT     ; RESET LATCH
EB24  B0 80                     MOV     AL,80H          ; MASK TO ENABLE NMI
EB26  E6 A0                     OUT     NMI_PORT,AL     ; ENABLE NMI
                        ;-----  ENABLE ALL INTERRUPTS WHICH WERE ENABLED BEFORE TRANSFER
EB28  8B 46 10                  MOV     AX,[BP+16]      ; GET MASK FROM THE STACK
EB2B  E6 21                     OUT     INTA01,AL
EB2D  58                        POP     AX              ; PASS BACK KEY STROKE FLAG
EB2E  5A                        POP     DX
EB2F  FB                        STI
EB30  C3                        RET
EB31                    ENABLE          ENDP
```

```
                                ;--------------------------------------------------
                                ;CLOCK_WAIT
                                ;       THIS PROCEDURE IS CALLED WHEN THE TIME OF DAY
                                ;       IS BEING UPDATED.  IT WAITS IF TIMER0 IS ALMOST
                                ;       READY TO WRAP UNTIL IT IS SAFE TO READ AN ACCURATE
                                ;       TIMER1.
                                ;INPUT
                                ;       NONE.
                                ;OUTPUT
                                ;       NONE. AX IS DESTROYED.
                                ;--------------------------------------------------
EB31                            CLOCK_WAIT      PROC    NEAR
EB31  32 C0                             XOR     AL,AL                   ; READ MODE TIMER0 FOR 8253
EB33  E6 43                             OUT     TIM_CTL,AL              ; OUTPUT TO THE 8253
EB35  50                                PUSH    AX
EB36  58                                POP     AX                      ; WAIT FOR 8253 TO INITIALIZE
                                                                        ; ITSELF
EB37  E4 40                             IN      AL,TIMER0               ; READ LEAST SIGNIFICANT BYTE
EB39  86 C4                             XCHG    AL,AH                   ; SAVE IT
EB3B  E4 40                             IN      AL,TIMER0               ; READ MOST SIGNIFICANT BYTE
EB3D  86 C4                             XCHG    AL,AH                   ; REARRANGE FOR PROPER ORDER
EB3F  3D 012C                           CMP     AX,THRESHOLD            ; IS TIMER0 CLOSE TO WRAPPING?
EB42  72 ED                             JC      CLOCK_WAIT              ; JUMP IF CLOCK IS WITHIN THRESHOLD
EB44  C3                                RET                             ; OK TO READ TIMER1
EB45                            CLOCK_WAIT      ENDP
                                ;--------------------------------------------------------------------------
                                ;GET_DRIVE
                                ;       THIS ROUTINE WILL CALCULATE A BIT MASK FOR THE DRIVE WHICH
                                ;       IS SELECTED BY THE CURRENT INT 13 CALL.  THE DRIVE SELECTED
                                ;       CORRESPONDS TO THE BIT IN THE MASK,  I.E. DRIVE ZERO
                                ;       CORRESPONDS TO BIT ZERO AND A 01H IS RETURNED. THE BIT IS
                                ;       CALCULATED BY ACCESSING THE PARAMETERS PASSED TO INT 13
                                ;       WHICH WERE SAVED ON THE STACK.
                                ;INPUT
                                ;       BYTE PTR[BP] MUST POINT TO DRIVE FOR SELECTION.
                                ;OUPTUT
                                ;       AL CONTAINS THE BIT MASK.  ALL OTHER REGISTERS ARE INTACT
                                ;--------------------------------------------------------------------------
EB45                            GET_DRIVE       PROC    NEAR
EB45  51                                PUSH    CX                      ; SAVE REGISTER.
EB46  8A 4E 00                          MOV     CL,BYTE PTR[BP]         ; GET DRIVE NUMBER
EB49  B0 01                             MOV     AL,1                    ; INITIALIZE AL WITH VALUE FOR
                                                                        ; SHIFTING
EB4B  D2 E0                             SHL     AL,CL                   ; SHIFT BIT POSITION BY DRIVE
                                                                        ; NUMBER (DRIVE IN RANGE 0-2)
EB4D  24 07                             AND     AL,07H                  ; ONLY THREE DRIVES ARE SUPPORTED.
                                                                        ;  RANGE CHECK
EB4F  59                                POP     CX                      ; RESTORE REGISTERS
EB50  C3                                RET
EB51                            GET_DRIVE       ENDP
                                ;--------------------------------------------------
                                ; THIS ROUTINE  CHECKS OPTIONAL ROM MODULES (CHECKSUM
                                ; FOR MODULES FROM C0000->D0000, CRC CHECK FOR CARTRIDGES
                                ; (D0000->F0000)
                                ; IF CHECK IS OK, CALLS INIT/TEST CODE IN MODULE
                                ;    MFG ERROR CODE= 25XX (XX=MSB OF SEGMENT IN ERROR)
                                ;--------------------------------------------------
EB51                            ROM_CHECK       PROC    NEAR
EB51  2B F6                             SUB     SI,SI                   ; SET SI TO POINT TO BEGINNING
                                                                        ; (REL. TO DS)
EB53  2A C0                             SUB     AL,AL                   ; ZERO OUT AL
EB55  8A 67 02                          MOV     AH,[BX+2]               ; GET LENGTH INDICATOR
EB58  D1 E0                             SHL     AX,1                    ; FORM COUNT
EB5A  50                                PUSH    AX                      ; SAVE COUNT
EB5B  81 FA D000                        CMP     DX,0D000H               ; SEE IF POINTER IS BELOW D000
EB5F  9C                                PUSHF                           ; SAVE RESULTS
EB60  B1 04                             MOV     CL,4                    ; ADJUST
EB62  D3 E8                             SHR     AX,CL                   ;
EB64  03 D0                             ADD     DX,AX                   ; SET POINTER TO NEXT MODULE
EB66  9D                                POPF                            ; RECOVER FLAGS FROM POINTER RANGE
                                                                        ; CHECK
EB67  59                                POP     CX                      ; RECOVER COUNT IN CX REGISTER
EB68  52                                PUSH    DX                      ; SAVE POINTER
EB69  7C 07                             JL      ROM_1                   ; DO ARITHMETIC CHECKSUM IF BELOW
                                                                        ; D0000
EB6B  E8 FE71 R                         CALL    CRC_CHECK               ; DO CRC CHECK
EB6E  74 2B                             JZ      ROM_CHECK_1             ; PROCEED IF OK
EB70  EB 05                             JMP     SHORT ROM_2             ; ELSE POST ERROR
EB72  E8 FEEB R                 ROM_1:  CALL    ROS_CHECKSUM            ; DO ARITHMETIC CHECKSUM
EB75  74 24                             JZ      ROM_CHECK_1             ; PROCEED IF OK
EB77  BA 1626                   ROM_2:  MOV     DX,1626H                ; POSITION CURSOR, ROW 22, COL 38
EB7A  B4 02                             MOV     AH,2
EB7C  B7 07                             MOV     BH,7
EB7E  CD 10                             INT     10H
EB80  8C DA                             MOV     DX,DS                   ; RECOVER DATA SEG
EB82  8A C6                             MOV     AL,DH                   ;
EB84  E8 18A9 R                         CALL    XPC_BYTE                ; DISPLAY MSB OF DATA SEG
EB87  8A DE                             MOV     BL,DH                   ; FORM XX VALUE OF ERROR CODE
EB89  B7 25                             MOV     BH,25H                  ; FORM 25 PORTION
EB8B  80 FE D0                          CMP     DH,0D0H                 ; IN CARTRIDGE SPACE?
EB8E  BE 003B R                         MOV     SI,OFFSET CART_ERR
EB91  7D 03                             JGE     ROM_CHECK_0     ;
EB93  BE 003A R                         MOV     SI,OFFSET ROM_ERR
EB96                            ROM_CHECK_0:
EB96  E8 09BC R                         CALL    E_MSG                   ; GO ERROR ROUTINE
EB99  EB 16                             JMP     SHORT ROM_CHECK_END     ; AND EXIT
EB9B                            ROM_CHECK_1:
EB9B  B8  ---- R                        MOV     AX,XXDATA               ; SET ES TO POINT TO XXDATA AREA
EB9E  8E C0                             MOV     ES,AX           ;
EBA0  26: C7 06 0014 R 0003             MOV     ES:IO_ROM_INIT,0003H    ; LOAD OFFSET
EBA7  26: 8C 1E 0016 R                  MOV     ES:IO_ROM_SEG,DS        ; LOAD SEGMENT
EBAC  26: FF 1E 0014 R                  CALL    DWORD PTR ES:IO_ROM_INIT ; CALL INIT./TEST ROUTINE
```

```
EBB1                            ROM_CHECK_END:
EBB1  5A                                POP     DX                      ; RECOVER POINTER
EBB2  C3                                RET                             ; RETURN TO CALLER
EBB3                            ROM_CHECK       ENDP
                                ;-- INT 13 ------------------------------------------------------------
                                ; DISKETTE I/O
                                ;       THIS INTERFACE PROVIDES ACCESS TO THE 5 1/4" DISKETTE DRIVES
                                ; INPUT
                                ;       (AH)=0  RESET DISKETTE SYSTEM
                                ;               HARD RESET TO NEC, PREPARE COMMAND, RECAL REQD ON
                                ;               ALL DRIVES
                                ;       (AH)=1  READ THE STATUS OF THE SYSTEM INTO (AL)
                                ;               DISKETTE_STATUS FROM LAST OP'N IS USED
                                ;       REGISTERS FOR READ/WRITE/VERIFY/FORMAT
                                ;       (DL) - DRIVE NUMBER (0-3 ALLOWED, VALUE CHECKED)
                                ;       (DH) - HEAD NUMBER (0-1 ALLOWED, NOT VALUE CHECKED)
                                ;       (CH) - TRACK NUMBER (0-39, NOT VALUE CHECKED)
                                ;       (CL) - SECTOR NUMBER (1-8, NOT VALUE CHECKED, NOT USED FOR
                                ;               FORMAT)
                                ;       (AL) - NUMBER OF SECTORS ( MAX = 8, NOT VALUE CHECKED, NOT
                                ;               USED FOR FORMAT, HOWEVER, CANNOT BE ZERO!!!)
                                ;       (ES:BX) - ADDRESS OF BUFFER ( NOT REQUIRED FOR VERIFY)
                                ;
                                ;       (AH)=2  READ THE DESIRED SECTORS INTO MEMORY
                                ;       (AH)=3  WRITE THE DESIRED SECTORS FROM MEMORY
                                ;       (AH)=4  VERIFY THE DESIRED SECTORS
                                ;       (AH)=5  FORMAT THE DESIRED TRACK
                                ;               FOR THE FORMAT OPERATION, THE BUFFER POINTER
                                ;               (ES,BX) MUST POINT TO THE COLLECTION OF DESIRED
                                ;               ADDRESS FIELDS FOR THE TRACK.  EACH FIELD IS
                                ;               COMPOSED OF 4 BYTES, (C,H,R,N), WHERE
                                ;               C = TRACK NUMBER, H=HEAD NUMBER, R = SECTOR NUMBER,
                                ;               N= NUMBER OF BYTES PER SECTOR (00=128, 01=256,
                                ;               02=512, 03=1024,).  THERE MUST BE ONE ENTRY FOR
                                ;               EVERY SECTOR ON THE TRACK. THIS INFORMATION IS USED
                                ;               TO FIND THE REQUESTED SECTOR DURING READ/WRITE
                                ;               ACCESS.
                                ; DATA VARIABLE -- DISK_POINTER
                                ;       DOUBLE WORD POINTER TO THE CURRENT SET OF DISKETTE PARAMETERS
                                ; OUTPUT
                                ;       AH = STATUS OF OPERATION
                                ;               STATUS BITS ARE DEFINED IN THE EQUATES FOR
                                ;               DISKETTE_STATUS VARIABLE IN THE DATA SEGMENT OF
                                ;               THIS MODULE
                                ;       CY = 0  SUCCESSFUL OPERATION (AH=0 ON RETURN)
                                ;       CY = 1  FAILED OPERATION (AH HAS ERROR REASON)
                                ;       FOR READ/WRITE/VERIFY
                                ;               DS,BX,DX,CH,CL PRESERVED
                                ;               AL = NUMBER OF SECTORS ACTUALLY READ
                                ;               **** AL MAY NOT BE CORRECT IF TIME OUT ERROR OCCURS
                                ;       NOTE: IF AN ERROR IS REPORTED BY THE DISKETTE CODE, THE
                                ;               APPROPRIATE ACTION IS TO RESET THE DISKETTE, THEN
                                ;               RETRY THE OPERATION. ON READ ACCESSES, NO MOTOR
                                ;               START DELAY IS TAKEN, SO THAT THREE RETRIES ARE
                                ;               REQUIRED ON READS TO ENSURE THAT THE PROBLEM IS NOT
                                ;               DUE TO MOTOR START-UP.
                                ;--------------------------------------------------------------
                                        ASSUME  CS:CODE,DS:DATA,ES:DATA
EC59                                    ORG     0EC59H
EC59                            DISKETTE_IO     PROC    FAR
EC59  FB                                STI                             ; INTERRUPTS BACK ON
EC5A  06                                PUSH    ES                      ; SAVE ES
EC5B  50                                PUSH    AX                      ; ALLOCATE ONE WORD OF STORAGE FOR
                                                                        ; TIMER1 INITIAL VALUE
EC5C  50                                PUSH    AX                      ; ALLOCATE ONE WORD ON STACK FOR
                                                                        ; USE IN PROCS ENABLE AND DISABLE.
                                                                        ;  WILL HOLD 8259 MASK.
EC5D  50                                PUSH    AX                      ; SAVE COMMAND AND N_SECTORS
EC5E  53                                PUSH    BX                      ; SAVE ADDRESS
EC5F  51                                PUSH    CX
EC60  1E                                PUSH    DS                      ; SAVE SEGMENT REGISTER VALUE
EC61  56                                PUSH    SI                      ; SAVE ALL REGISTERS DURING
                                                                        ; OPERATION
EC62  57                                PUSH    DI
EC63  55                                PUSH    BP
EC64  52                                PUSH    DX
EC65  8B EC                             MOV     BP,SP                   ; SET UP POINTER TO HEAD PARM
EC67  E8 138B R                         CALL    DDS                     ; SET DS=DATA
EC6A  E8 EC90 R                         CALL    J1                      ; CALL THE REST TO ENSURE DS
                                                                        ; RESTORED
EC6D  B3 04                             MOV     BL,4                    ; GET THE MOTOR WAIT PARAMETER
EC6F  E8 E9B4 R                         CALL    GET_PARM
EC72  88 26 0040 R                      MOV     MOTOR_COUNT,AH  ; SET THE TIMER COUNT FOR THE MOTOR
EC76  8A 26 0041 R                      MOV     AH,DISKETTE_STATUS      ; GET STATUS OF OPERATION
EC7A  88 66 0F                          MOV     [BP+15],AH              ; RETURN STATUS IN AL
EC7D  5A                                POP     DX                      ; RESTORE ALL REGISTERS
EC7E  5D                                POP     BP
EC7F  5F                                POP     DI
EC80  5E                                POP     SI
EC81  1F                                POP     DS
EC82  59                                POP     CX
EC83  5B                                POP     BX                      ; RECOVER OFFSET
EC84  58                                POP     AX
EC85  83 C4 04                          ADD     SP,4                    ; DISCARD DUMMY SPACE FOR 8259 MASK
EC88  07                                POP     ES                      ; RECOVER SEGMENT
EC89  80 FC 01                          CMP     AH,1                    ; SET THE CARRY FLAG TO INDICATE
EC8C  F5                                CMC                             ;  SUCCESS OR FAILURE
EC8D  CA 0002                           RET     2                       ; THROW AWAY SAVED FLAGS
```

```
EC90                            DISKETTE_IO     ENDP
EC90                            J1      PROC    NEAR
EC90  8A F0                             MOV     DH,AL           ; SAVE # SECTORS IN DH
EC92  80 26 003F R 7F                   AND     MOTOR_STATUS,07FH       ; INDICATE A READ OPERATION
EC97  0A E4                             OR      AH,AH           ; AH=0
EC99  74 27                             JZ      DISK_RESET
EC9B  FE CC                             DEC     AH              ; AH=1
EC9D  74 74                             JZ      DISK_STATUS
EC9F  C6 06 0041 R 00                   MOV     DISKETTE_STATUS,0 ; RESET THE STATUS INDICATOR
ECA4  80 FA 02                          CMP     DL,2            ; TEST FOR DRIVE IN 0-2 RANGE
ECA7  77 13                             JA      J3              ; ERROR IF ABOVE
ECA9  FE CC                             DEC     AH              ; AH=2
ECAB  74 6D                             JZ      DISK_READ
ECAD  FE CC                             DEC     AH              ; AH=3
ECAF  75 03                             JNZ     J2              ; TEST_DISK_VERF
ECB1  E9 ED3D R                         JMP     DISK_WRITE
ECB4                            J2:                             ; TEST_DISK_VERF
ECB4  FE CC                             DEC     AH              ; AH=4
ECB6  74 62                             JZ      DISK_VERF
ECB8  FE CC                             DEC     AH              ; AH=5
ECBA  74 62                             JZ      DISK_FORMAT
ECBC                            J3:                             ; BAD_COMMAND
ECBC  C6 06 0041 R 01                   MOV     DISKETTE_STATUS,BAD_CMD ; ERROR CODE, NO SECTORS
                                                                ; TRANSFERRED
ECC1  C3                                RET                     ; UNDEFINED OPERATION
ECC2                            J1      ENDP
                                ;------ RESET THE DISKETTE SYSTEM
ECC2                            DISK_RESET      PROC    NEAR
ECC2  BA 00F2                           MOV     DX,NEC_CTL      ; ADAPTER CONTROL PORT
ECC5  FA                                CLI                     ; NO INTERRUPTS
ECC6  A0 003F R                         MOV     AL,MOTOR_STATUS ; FIND OUT IF MOTOR IS RUNNING
ECC9  24 07                             AND     AL,07H          ; DRIVE BITS
ECCB  EE                                OUT     DX,AL           ; RESET THE ADAPTER
ECCC  C6 06 003E R 00                   MOV     SEEK_STATUS,0   ; SET RECAL REQUIRED ON ALL DRIVES
ECD1  C6 06 0041 R 00                   MOV     DISKETTE_STATUS,0 ; SET OK STATUS FOR DISKETTE
ECD6  0C 80                             OR      AL,FDC_RESET    ; TURN OFF RESET
ECD8  EE                                OUT     DX,AL           ; TURN OFF THE RESET
ECD9  FB                                STI                     ; REENABLE THE INTERRUPTS
ECDA  BE ECFA R                         MOV     SI,OFFSET J4_2  ; DUMMY RETURN FOR
ECDD  56                                PUSH    SI              ; PUSH RETURN IF ERROR
                                                                ; IN NEC_OUTPUT
ECDE  B9 0010                           MOV     CX,10H          ; NUMBER OF SENSE INTERRUPTS TO
                                                                ; ISSUE
ECE1  B4 08                     J4_0:   MOV     AH,08H          ; COMMAND FOR SENSE INTERRUPT
                                                                ; STATUS
ECE3  E8 E98A R                         CALL    NEC_OUTPUT      ; OUTPUT  THE SENSE INTERRUPT
                                                                ; STATUS
ECE6  E8 EAA0 R                         CALL    RESULTS         ; GET STATUS FOLLOWING COMPLETION
                                                                ; OF RESET
ECE9  A0 0042 R                         MOV     AL,NEC_STATUS   ; IGNORE ERROR RETURN AND DO OWN
                                                                ; TEST
ECEC  3C C0                             CMP     AL,0C0H         ; TEST FOR DRIVE READY TRANSITION
ECEE  74 12                             JZ      J7              ; EVERYTHING OK
ECF0  E2 EF                             LOOP    J4_0            ; RETRY THE COMMAND
ECF2  80 0E 0041 R 20           J4_1:   OR      DISKETTE_STATUS,BAD_NEC ; SET ERROR CODE
ECF7  5E                                POP     SI
ECF8  EB 18                             JMP     SHORT J8
ECFA  BE ECFA R                 J4_2:   MOV     SI,OFFSET J4_2  ; NEC_OUTPUT FAILED, RETRY THE
                                                                ; SENSE INTERRUPT
ECFD  56                                PUSH    SI              ; OFFSET OF BAD RETURN IN
                                                                ; NEC_OUTPUT
ECFE  E2 E1                             LOOP    J4_0            ; RETRY
ED00  EB F0                             JMP     SHORT J4_1
                                ;------ SEND SPECIFY COMMAND TO NEC
ED02  5E                        J7:     POP     SI              ; GET RID OF DUMMY ARGUMENT
ED03  B4 03                             MOV     AH,03H          ; SPECIFY COMMAND
ED05  E8 E98A R                         CALL    NEC_OUTPUT      ; OUTPUT THE COMMAND
ED08  B3 01                             MOV     BL,1            ; STEP RATE TIME AND HEAD UNLOAD
ED0A  E8 E9B4 R                         CALL    GET_PARM        ; OUTPUT TO THE NEC CONTROLLER
ED0D  B3 03                             MOV     BL,3            ; PARM1 HEAD LOAD AND NO DMA
ED0F  E8 E9B4 R                         CALL    GET_PARM        ;  TO THE NEC CONTROLLER
ED12                            J8:                             ; RESET_RET
ED12  C3                                RET                     ; RETURN TO CALLER
ED13                            DISK_RESET      ENDP
                                ;------ DISKETTE STATUS ROUTINE
ED13                            DISK_STATUS     PROC    NEAR
ED13  A0 0041 R                         MOV     AL,DISKETTE_STATUS
ED16  88 46 0E                          MOV     BYTE PTR[BP+14],AL ; PUT STATUS ON STACK, IT WILL
                                                                ; POP IN AL
ED19  C3                                RET
ED1A                            DISK_STATUS     ENDP
                                ;------ DISKETTE VERIFY
ED1A                            DISK_VERF       LABEL   NEAR
                                ;------ DISKETTE READ
ED1A                            DISK_READ       PROC    NEAR
ED1A                            J9:                             ; DISK_READ_CONT
ED1A  B4 46                             MOV     AH,046H         ; SET UP READ COMMAND FOR NEC
                                                                ; CONTROLLER
ED1C  EB 26                             JMP     SHORT RW_OPN    ; GO DO THE OPERATION
ED1E                            DISK_READ       ENDP
                                ;------ DISKETTE FORMAT
ED1E                            DISK_FORMAT     PROC    NEAR
ED1E  80 0E 003F R 80                   OR      MOTOR_STATUS,80H ; INDICATE A WRITE OPERATION
ED23  B4 4D                             MOV     AH,04DH         ; ESTABLISH THE FORMAT COMMAND
ED25  EB 1D                             JMP     SHORT RW_OPN    ; DO THE OPERATION
```

```
ED27                            J10:                                    ; CONTINUATION OF RW_OPN FOR FMT
ED27  B3 07                             MOV     BL,7                    ; GET THE
ED29  E8 E9B4 R                         CALL    GET_PARM                ;  BYTES/SECTOR VALUE TO NEC
ED2C  B3 09                             MOV     BL,9                    ; GET THE
ED2E  E8 E9B4 R                         CALL    GET_PARM                ;  SECTORS/TRACK VALUE TO NEC
ED31  B3 0F                             MOV     BL,15                   ; GET THE
ED33  E8 E9B4 R                         CALL    GET_PARM                ;  GAP LENGTH VALUE TO NEC
ED36  BB 0011                           MOV     BX,17                   ; GET THE FILLER BYTE
ED39  53                                PUSH    BX                      ; SAVE PARAMETER INDEX ON STACK
ED3A  E9 EDCD R                         JMP     J16                     ;  TO THE CONTROLLER
ED3D                            DISK_FORMAT     ENDP
                                ;------ DISKETTE WRITE ROUTINE
ED3D                            DISK_WRITE      PROC    NEAR
ED3D  80 0E 003F R 80                   OR      MOTOR_STATUS,80H ; INDICATE A WRITE OPERATION
ED42  B4 45                             MOV     AH,045H                 ; NEC COMMAND TO WRITE TO DISKETTE
ED44                            DISK_WRITE      ENDP
                                ;----- ALLOW WRITE ROUTINE TO FALL INTO RW_OPN
                                ;--------------------------------------
                                ; RW_OPN
                                ;       THIS ROUTINE PERFORMS THE READ/WRITE/VERIFY OPERATION
                                ;--------------------------------------
ED44                            RW_OPN  PROC    NEAR
ED44  50                                PUSH    AX                      ; SAVE THE COMMAND
                                ;------ TURN ON THE MOTOR AND SELECT THE DRIVE
ED45  51                                PUSH    CX                      ; SAVE THE T/S PARMS
ED46  FA                                CLI                             ; NO INTERRUPTS WHILE DETERMINING
                                                                        ; MOTOR STATUS
ED47  C6 06 0040 R FF                   MOV     MOTOR_COUNT,0FFH ; SET LARGE COUNT DURING OPERATION
ED4C  E8 EB45 R                         CALL    GET_DRIVE               ; GET THE DRIVE PARAMETER FROM THE
                                                                        ; STACK
ED4F  84 06 003F R                      TEST    MOTOR_STATUS,AL ; TEST MOTOR FOR OPERATING
ED53  75 1F                             JNZ     J14                     ; IF RUNNING, SKIP THE WAIT
ED55  80 26 003F R F0                   AND     MOTOR_STATUS,0F0H ; TURN OFF RUNNING DRIVE
ED5A  08 06 003F R                      OR      MOTOR_STATUS,AL         ; TURN ON THE CURRENT MOTOR
ED5E  FB                                STI                             ; INTERRUPTS BACK ON
ED5F  0C 80                             OR      AL,FDC_RESET    ; NO RESET.  TURN ON MOTOR
ED61  E6 F2                             OUT     NEC_CTL,AL
                                ;------ WAIT FOR MOTOR BOTH READ AND WRITE
ED63  B3 14                             MOV     BL,20                   ; GET MOTOR START TIME
ED65  E8 E9B4 R                         CALL    GET_PARM
ED68  0A E4                             OR      AH,AH                   ; TEST FOR NO WAIT
ED6A                            J12:                                    ; TEST_WAIT_TIME
ED6A  74 08                             JZ      J14                     ; EXIT WITH TIME EXPIRED
ED6C  2B C9                             SUB     CX,CX                   ; SET UP 1/8 SECOND LOOP TIME
ED6E  E2 FE                     J13:    LOOP    J13                     ; WAIT FOR THE REQUIRED TIME
ED70  FE CC                             DEC     AH                      ; DECREMENT TIME VALUE
ED72  EB F6                             JMP     J12                     ; ARE WE DONE YET
ED74                            J14:                                    ; MOTOR_RUNNING
ED74  FB                                STI                             ; INTERRUPTS BACK ON FOR BYPASS
                                                                        ; WAIT
ED75  59                                POP     CX
                                ;------ DO THE SEEK OPERATION
ED76  E8 E9FB R                         CALL    SEEK                    ; MOVE TO CORRECT TRACK
ED79  58                                POP     AX                      ; RECOVER COMMAND
ED7A  8A FC                             MOV     BH,AH                   ; SAVE COMMAND IN BH
ED7C  B6 00                             MOV     DH,0                    ; SET NO SECTORS READ IN CASE OF
                                                                        ; ERROR
ED7E  73 03                             JNC     J14_1                   ; IF NO ERROR CONTINUE, JUMP AROUND
                                                                        ; JMP
ED80  E9 EED7 R                         JMP     J17                     ; CARRY SET JUMP TO MOTOR WAIT
ED83  BE EED7 R                 J14_1:  MOV     SI,OFFSET J17   ; DUMMY RETURN ON STACK FOR
                                                                        ; NEC_OUTPUT
ED86  56                                PUSH    SI                      ;  SO THAT IT WILL RETURN TO MOTOR
                                                                        ; OFF LOCATION
                                ;------ SEND OUT THE PARAMETERS TO THE CONTROLLER
ED87  E8 E98A R                         CALL    NEC_OUTPUT      ; OUTPUT THE OPERATION COMMAND
ED8A  8A 66 01                          MOV     AH,[BP+1]       ; GET THE CURRENT HEAD NUMBER
ED8D  D0 E4                             SAL     AH,1            ; MOVE IT TO BIT 2
ED8F  D0 E4                             SAL     AH,1
ED91  80 E4 04                          AND     AH,4            ; ISOLATE THAT BIT
ED94  0A E2                             OR      AH,DL           ; OR IN THE DRIVE NUMBER
ED96  E8 E98A R                         CALL    NEC_OUTPUT
                                ;------ TEST FOR FORMAT COMMAND
ED99  80 FF 4D                          CMP     BH,04DH         ; IS THIS A FORMAT OPERATION?
ED9C  75 02                             JNE     J15             ; NO. CONTINUE WITH R/W/V
ED9E  EB 87                             JMP     J10             ; IF SO, HANDLE SPECIAL
EDA0  8A E5                     J15:    MOV     AH,CH           ; CYLINDER NUMBER
EDA2  E8 E98A R                         CALL    NEC_OUTPUT
EDA5  8A 66 01                          MOV     AH,[BP+1]       ; HEAD NUMBER FROM STACK
EDA8  E8 E98A R                         CALL    NEC_OUTPUT
EDAB  8A E1                             MOV     AH,CL           ; SECTOR NUMBER
EDAD  E8 E98A R                         CALL    NEC_OUTPUT
EDB0  B3 07                             MOV     BL,7            ; BYTES/SECTOR PARM FROM BLOCK
EDB2  E8 E9B4 R                         CALL    GET_PARM        ;  TO THE NEC
EDB5  B3 08                             MOV     BL,8            ; EOT PARM FROM BLOCK
EDB7  E8 E9B4 R                         CALL    GET_PARM        ;  RETURNED IN AH
EDBA  02 4E 0E                          ADD     CL,[BP+14]      ; ADD CURRENT SECTOR TO NUMBER IN
                                                                ; TRANSFER
EDBD  FE C9                             DEC     CL              ; CURRENT_SECTOR + N_SECTORS - 1
EDBF  8A E1                             MOV     AH,CL           ; EOT PARAMETER IS THE CALCULATED
                                                                ; ONE
EDC1  E8 E98A R                         CALL    NEC_OUTPUT
EDC4  B3 0B                             MOV     BL,11           ; GAP LENGTH PARM FROM BLOCK
EDC6  E8 E9B4 R                         CALL    GET_PARM        ;  TO THE NEC
EDC9  BB 000D                           MOV     BX,13           ; DTL PARM FROM BLOCK
EDCC  53                                PUSH    BX              ; SAVE INDEX TO DISK PARAMETER ON
                                                                ; STACK
```

```
EDCD  FC                      J16:    CLD                     ; FORWARD DIRECTION
                              ;------ START TIMER1 WITH INITIAL VALUE OF FFFF
EDCE  B0 70                           MOV     AL,01110000B    ; SELECT TIMER1,LSB-MSB, MODE 0,
                                                              ; BINARY COUNTER
EDD0  E6 43                           OUT     TIM_CTL,AL      ; INITIALIZE THE COUNTER
EDD2  50                              PUSH    AX
EDD3  58                              POP     AX              ; ALLOW ENOUGH TIME FOR THE 8253 TO
                                                              ; INITIALIZE ITSELF
EDD4  B0 FF                           MOV     AL,0FFH         ; INITIAL COUNT VALUE FOR THE 8253
EDD6  E6 41                           OUT     TIMER+1,AL      ; OUTPUT LEAST SIGNIFICANT BYTE
EDD8  50                              PUSH    AX
EDD9  58                              POP     AX              ; WAIT
EDDA  E6 41                           OUT     TIMER+1,AL      ; OUTPUT MOST SIGNIFACNT BYTE
                              ;------INITIALIZE CX FOR JUMP AFTER LAST PARAMETER IS PASSED TO NEC
EDDC  8A 46 0F                        MOV     AL,[BP+15]      ; RETRIEVE COMMAND PARAMETER
EDDF  A8 01                           TEST    AL,01H          ; IS THIS AN ODD NUMBERED FUNCTION?
EDE1  74 05                           JZ      J16_1           ; JUMP IF NOT ODD NUMBERED
EDE3  B9 EE4E R                       MOV     CX,OFFSET WRITE_LOOP
EDE6  EB 0C                           JMP     SHORT J16_3
EDE8  3C 02                   J16_1:  CMP     AL,2            ; IS THIS A READ?
EDEA  75 05                           JNZ     J16_2           ; JUMP IF VERIFY
EDEC  B9 EE3A R                       MOV     CX,OFFSET READ_LOOP
EDEF  EB 03                           JMP     SHORT J16_3
EDF1  B9 EE20 R               J16_2:  MOV     CX,OFFSET VERIFY_LOOP
                              ;------FINISH INITIALIZATION
EDF4                          J16_3:
                              ;--------------------------------------------------------------------
                              ;***NOTE***
                              ;ALL INTERRUPTS ARE ABOUT TO BE DISABLED.  THERE IS A POTENTIAL
                              ;       THAT THIS TIME PERIOD WILL BE LONG ENOUGH TO MISS TIME OF
                              ;       DAY INTERRUPTS.  FOR THIS REASON, TIMER1 WILL BE USED TO
                              ;       KEEP TRACK OF THE NUMBER OF TIME OF DAY INTERRUPTS WHICH
                              ;       WILL BE MISSED.  THIS INFORMATION IS USED AFTER THE DISKETTE
                              ;       OPERATION TO UPDATE THE TIME OF DAY.
                              ;--------------------------------------------------------------------
EDF4  B0 10                           MOV     AL,10H          ; DISABLE NMI
EDF6  E6 A0                           OUT     NMI_PORT,AL     ; NO KEYBOARD INTERRUPT
EDF8  E8 EB31 R                       CALL    CLOCK_WAIT      ; WAIT IF TIMER0 IS ABOUT TO
                                                              ; INTERRUPT
                              ;------ ENABLE WATCHDOG TIMER
                              ;--------------------------------------------------------------------
                              ;***NOTE***
                              ;       GIVEN THE CURRENT SYSTEM CONFIGURATION A  METHOD IS NEEDED
                              ;       TO PULL THE NEC OUT OF "FATAL ERROR" SITUATIONS.  A TIMER
                              ;       ON THE ADAPTER CARD IS PROVIDED WHICH WILL PERFORM THIS
                              ;       FUNCTION.  THE WATCHDOG TIMER ON THE ADAPTER CARD IS ENABLED
                              ;       AND STROBED BEFORE THE 8259 INTERRUPT 6 LINE IS ENABLED.
                              ;       THIS IS BECAUSE OF A GLITCH ON THE LINE LARGE ENOUGH TO
                              ;       TRIGGER AN INTERRUPT.
                              ;--------------------------------------------------------------------
EDFB  E8 EB45 R                       CALL    GET_DRIVE       ; GET BIT MASK FOR DRIVE
EDFE  BA 00F2                         MOV     DX,NEC_CTL      ; CONTROL PORT TO NEC
EE01  0C E0                           OR      AL,FDC_RESET+WD_ENABLE+WD_STROBE
EE03  EE                              OUT     DX,AL           ; OUTPUT CONTROL INFO FOR
                                                              ; WATCHDOG(WD) ENABLE
EE04  24 A7                           AND     AL,FDC_RESET+WD_ENABLE+7H
EE06  EE                              OUT     DX,AL           ; OUTPUT CONTROL INFO TO STROBE
                                                              ; WATCHDOG
EE07  BA 00F4                         MOV     DX,NEC_STAT     ; PORT TO NEC STATUS
EE0A  B0 20                           MOV     AL,20H          ; SELECT TIMER1 INPUT FROM TIMER0
                                                              ; OUTPUT
EE0C  E6 A0                           OUT     NMI_PORT,AL
                              ;------ READ TIMER1 NOW AND SAVE THE INITIAL VALUE
EE0E  E8 E81A R                       CALL    READ_TIME       ; GET TIMER1 VALUE
EE11  89 46 12                        MOV     [BP+18],AX      ; SAVE INITIAL VALUE FOR CLOCK
                                                              ; UPDATE IN TEMPORAY STORAGE
EE14  E8 EAFC R                       CALL    DISABLE         ; DISABLE ALL INTERRUPTS
                              ;------ NEC BEGINS OPERATION WHEN NEC RECEIVES LAST PARAMETER
EE17  5B                              POP     BX              ; GET PARAMTER FROM STACK
EE18  E8 E9B4 R                       CALL    GET_PARM        ; OUTPUT LAST PARAMETER TO THE NEC
EE1B  58                              POP     AX              ; CAN NOW DISCARD THAT DUMMY RETURN
                                                              ; ADDRESS
EE1C  06                              PUSH    ES
EE1D  1F                              POP     DS              ; INITIALIZE DS FOR WRITE
EE1E  FF E1                           JMP     CX              ; JUMP TO APPROPRIATE R/W/V LOOP
                              ;--------------------------------------------------------------------
                              ;***NOTE***
                              ;       DATA IS TRANSFERRED USING POLLING ALGORITHMS.  THESE LOOPS
                              ;       TRANSFER A DATA BYTE AT A TIME WHILE POLLING THE NEC FOR
                              ;       NEXT DATA BYTE AND COMPLETION STATUS.
                              ;--------------------------------------------------------------------
                              ;-----VERIFY OPERATION
EE20                          VERIFY_LOOP:
EE20  EC                              IN      AL,DX           ; READ STATUS
EE21  A8 20                           TEST    AL,BUSY_BIT     ; HAS NEC ENTERED EXECUTION PHASE
                                                              ; YET?
EE23  74 FB                           JZ      VERIFY_LOOP     ; NO, CONTINUE SAMPLING
EE25                          J22_2:
EE25  A8 80                           TEST    AL,RQM          ; IS DATA READY?
EE27  75 07                           JNZ     J22_4           ; JUMP IF DATA TRANSFER IS READY
EE29  EC                              IN      AL,DX           ; READ STATUS PORT
EE2A  A8 20                           TEST    AL,BUSY_BIT     ; ARE WE DONE?
EE2C  75 F7                           JNZ     J22_2           ; JUMP IF MORE TRANSFERS
EE2E  EB 35                           JMP     SHORT OP_END    ; TRANSFER DONE
EE30  42                      J22_4:  INC     DX              ; POINT AT NEC DATA REGISTER
EE31  EC                              IN      AL,DX           ; READ DATA
EE32  4A                              DEC     DX              ; POINT AT NEC STATUS REGISTER
EE33  EC                              IN      AL,DX           ; READ STATUS PORT
EE34  A8 20                           TEST    AL,BUSY_BIT     ; ARE WE DONE?
EE36  75 ED                           JNZ     J22_2           ; CONTINUE
EE38  EB 2B                           JMP     SHORT OP_END    ; WE ARE DONE
```

```
                                ;-----READ OPERATION
EE3A                            READ_LOOP:
EE3A  EC                                IN      AL,DX           ; READ STATUS REGISTER
EE3B  A8 20                             TEST    AL,BUSY_BIT     ; HAS NEC STARTED THE EXECUTION
                                                                ; PHASE?
EE3D  74 FB                             JZ      READ_LOOP       ; HAS NOT STATRED YET
EE3F  EC                        J22_5:  IN      AL,DX           ; READ STATUS PORT
EE40  A8 20                             TEST    AL,BUSY_BIT     ; HAS NEC COMPLETED EXECUTION
                                                                ; PHASE?
EE42  74 21                             JZ      OP_END          ; JUMP IF EXECUTION PHASE IS OVER
EE44  A8 80                             TEST    AL,RQM          ; IS DATA READY?
EE46  74 F7                             JZ      J22_5           ; READ THE DATA
EE48  42                                INC     DX              ; POINT AT NEC_DATA
EE49  EC                                IN      AL,DX           ; READ DATA
EE4A  AA                                STOSB                   ; TRANSFER DATA
EE4B  4A                                DEC     DX              ; POINT AT NEC_STATUS
EE4C  EB F1                             JMP     J22_5           ; CONTINUE WITH READ OPERATION
                                ;-----WRITE AND FORMAT OPERATION
EE4E                            WRITE_LOOP:
EE4E  EC                                IN      AL,DX           ; READ NEC STATUS PORT
EE4F  A8 20                             TEST    AL,BUSY_BIT     ; HAS THE NEC ENTERED EXECUTION
                                                                ; PHASE YET?
EE51  74 FB                             JZ      WRITE_LOOP      ; NO, CONTINUE LOOPING
EE53  B9 2080                           MOV     CX,BUSY_BIT*256+RQM
EE56                            J22_7:
EE56  EC                                IN      AL,DX           ; READ STATUS PORT
EE57  84 C5                             TEST    AL,CH           ; IS THE FEC STILL IN THE EXECUTION
                                                                ; PHASE?
EE59  74 0A                             JZ      OP_END          ; JUMP IF EXECUTION PHASE IS DONE.
EE5B  84 C1                             TEST    AL,CL           ; IS THE DATA PORT READY FOR THE
                                                                ; TRANSFER?
EE5D  74 F7                             JZ      J22_7           ; JUMP TO WRITE DATA
EE5F  42                                INC     DX              ; POINT AT DATA REGISTER
EE60  AC                                LODSB                   ; TRANSFER BYTE
EE61  EE                                OUT     DX,AL           ; WRITE THE BYTE ON THE DISKETTE
EE62  4A                                DEC     DX              ; POINT AT THE STATUS REGISTER
EE63  EB F1                             JMP     J22_7           ; CONTINUE WITH WRITE OR FORMAT
                                ;------TRANSFER PROCESS IS OVER
EE65  9C                        OP_END: PUSHF                   ; SAVE THE CARRY BIT SET IN
                                                                ; DISK_INT
EE66  E8 EB45 R                         CALL    GET_DRIVE       ; GET BIT MASK FOR DRIVE SELECTION
EE69  0C 80                             OR      AL,FDC_RESET    ; NO RESET, KEEP DRIVE SPINNING
EE6B  BA 00F2                           MOV     DX,NEC_CTL      ;
EE6E  EE                                OUT     DX,AL           ; DISABLE WATCHDOG
                                ;------ UPDATE TIME OF DAY
EE6F  E8 138B R                         CALL    DDS             ; POINT DS AT BIOS DATA SEGMENT
EE72  E8 EB31 R                         CALL    CLOCK_WAIT      ; WAIT IF TIMER0 IS CLOSE TO
                                                                ; WRAPPING
EE75  E8 E81A R                         CALL    READ_TIME
EE78  8B 5E 12                          MOV     BX,[BP+18]      ; GET THE INITIAL VALUE OF TIMER1
EE7B  2B C3                             SUB     AX,BX           ; UPDATE NUMBER OF INTERRUPTS
                                                                ; MISSED
EE7D  F7 D8                             NEG     AX              ; PUT IT IN AX
EE7F  50                                PUSH    AX              ; SAVE IT FOR REUSE IN ISSUING USER
                                                                ; TIMER INTERRUPTS
EE80  01 06 006C R                      ADD     TIMER_LOW,AX    ; ADD NUMBER OF TIMER INTERRUPTS TO
                                                                ; TIME
EE84  73 04                             JNC     J16_4           ; JUMP IF TIMER_LOW DID NOT SPILL
                                                                ; OVER TO TIMER_HI
EE86  FF 06 006E R                      INC     TIMER_HIGH
EE8A  83 3E 006E R 18           J16_4:  CMP     TIMER_HIGH,018H ; TEST FOR COUNT TOTALING 24 HOURS
EE8F  75 19                             JNZ     J16_5           ; JUMP IF NOT 24 HOURS
EE91  81 3E 006C R 00B0                 CMP     TIMER_LOW,0B0H  ; LOW VALUE = 24 HOUR VALUE?
EE97  7C 11                             JL      J16_5           ; NOT 24 HOUR VALUE?
                                ;------ TIMER HAS GONE 24 HOURS
EE99  C7 06 006E R 0000                 MOV     TIMER_HIGH,0    ; ZERO OUT TIMER_HIGH VALUE
EE9F  81 2E 006C R 00B0                 SUB     TIMER_LOW,0B0H  ; VALUE REFLECTS CORRECT TICKS PAST
                                                                ; 00B0H
EEA5  C6 06 0070 R 01                   MOV     TIMER_OFL,1     ; INDICATES 24 HOUR THRESHOLD
EEAA  E8 EB0B R                 J16_5:  CALL    ENABLE          ; ENABLE ALL INTERRUPTS
EEAD  59                                POP     CX              ; CX:=AX, COUNT FOR NUMBER OF USER
                                                                ; TIME INTERRUPTS
EEAE  E3 26                             JCXZ    J16_7           ; IF ZERO DO NOT ISSUE ANY
                                                                ; INTERRUPTS
EEB0  1E                                PUSH    DS              ; SAVE ALL REGISTERS SAVED PRIOR TO
                                                                ; INT 1C CALL FROM TIMERINT
EEB1  50                                PUSH    AX              ; THIS PROVIDES A COMPATIBLE
                                                                ; INTERFACE TO 1C
EEB2  52                                PUSH    DX              ;
EEB3                            J16_6:
EEB3  CD 1C                             INT     1CH             ; TRANSFER CONTROL TO USER
                                                                ; INTERRUPT
EEB5  E2 FC                             LOOP    J16_6           ; DO ALL USER TIMER INTERRUPTS
EEB7  5A                                POP     DX
EEB8  58                                POP     AX
EEB9  1F                                POP     DS              ; RESTORE REGISTERS
                                ;------ CLOCK IS UPDATED AND USER INTERRUPTS 1C HAVE BEEN ISSUED.
                                ;       CHECK IF KEYSTROKE OCCURED
EEBA  0A C0                             OR      AL,AL           ; AL WAS SET DURING CALL TO ENABLE
EEBC  74 18                             JZ      J16_7           ; NO KEY WAS PRESSED WHILE SYSTEM
                                                                ; WAS MASKED
EEBE  BB 0080                           MOV     BX,080H         ; DURATION OF TONE
EEC1  B9 0048                           MOV     CX,048H         ; FREQUNCY OF TONE
EEC4  E8 E035 R                         CALL    KB_NOISE        ; NOTIFY USER OF MISSED KEYBORAD
                                                                ; INPUT
```

```
                        ;-------CLEAR SHIFT STATES DONT LEAVE POSSIBILTY OF DANGLING STATES
                        ;       OF MISSED BREAKS
EEC7  80 26 0017 R F0           AND     KB_FLAG,0F0H    ; CLEAR ALT,CLRL,LEFT AND RIGHT
                                                        ; SHIFTS
EECC  80 26 0018 R 0F           AND     KB_FLAG_1,0FH   ; CLEAR POTENTIAL BREAK OF INS,CAPS
                                                        ; NUM AND SCROLL SHIFT
EED1  80 26 0088 R 1F           AND     KB_FLAG_2,1FH   ; CLEAR FUNCTION STATES
EED6  9D                J16_7: POPF                    ; GET THE FLAGS
EED7                    J17:
EED7  72 40                     JC      J20
EED9  E8 EAA0 R                 CALL    RESULTS         ; GET THE NEC STATUS
EEDC  72 3B                     JC      J20             ; LOOK FOR ERROR
                        ;------ CHECK THE RESULTS RETURNED BY THE CONTROLLER
EEDE  FC                        CLD                     ; SET THE CORRECT DIRECTION
EEDF  BE 0042 R                 MOV     SI,OFFSET NEC_STATUS ; POINT TO STATUS FIELD
EEE2  AC                        LODS    NEC_STATUS      ; GET STO
EEE3  24 C0                     AND     AL,0C0H         ; TEST FOR NORMAL TERMINATION
EEE5  74 58                     JZ      J22             ; OPN_OK
EEE7  3C 40                     CMP     AL,040H         ; TEST FOR ABNORMAL TERMINATION
EEE9  75 25                     JNZ     J18             ; NOT ABNORMAL, BAD NEC
                        ;--------------------------------------------------------------
                        ;***NOTE***
                        ;       THE CURRENT SYSTEM CONFIGURATION HAS NO DMA.  IN ORDER TO
                        ;       STOP THE NEC AN EOT MUST BE PASSED TO FORCE THE NEC TO HALT
                        ;       THEREFORE, THE STATUS RETURNED BY THE NEC WILL ALWAYS SHOW
                        ;       AN EOT ERROR.  IF THIS IS THE ONLY ERROR RETURNED AND THE
                        ;       NUMBER OF SECTORS TRANSFERRED EQUALS THE NUMBER SECTORS
                        ;       REQUESTED IN THIS INTERRUPT CALL THEN THE OPERATION HAS
                        ;       COMPLETED SUCCESSFULLY.  IF AN EOT ERROR IS RETURNED AND THE
                        ;       REQUESTED NUMBER OF SECTORS IS NOT THE NUMBER OF SECTORS
                        ;       TRANSFERRED THEN THE ERROR IS LEGITIMATE.  WHEN THE EOT
                        ;       ERROR IS INVALID THE STATUS BYTES RETURNED ARE UPDATED TO
                        ;       REFLECT THE STATUS OF THE OPERATION IF DMA HAD BEEN PRESENT
                        ;--------------------------------------------------------------
EEEB  AC                        LODS    NEC_STATUS      ; GET ST1
EEEC  3C 80                     CMP     AL,80H          ; IS THIS THE ONLY ERROR?
EEEE  74 2A                     JE      J21_1           ; NORMAL TERMINATION, NO ERROR
EEF0  D0 E0                     SAL     AL,1            ; NOT EOT ERROR, BYPASS ERROR BITS
EEF2  D0 E0                     SAL     AL,1
EEF4  D0 E0                     SAL     AL,1            ; TEST FOR CRC ERROR
EEF6  B4 10                     MOV     AH,BAD_CRC
EEF8  72 18                     JC      J19             ; RW_FAIL
EEFA  D0 E0                     SAL     AL,1            ; TEST FOR DMA OVERRUN
EEFC  B4 08                     MOV     AH,BAD_DMA
EEFE  72 12                     JC      J19             ; RW_FAIL
EF00  D0 E0                     SAL     AL,1
EF02  D0 E0                     SAL     AL,1            ; TEST FOR RECORD NOT FOUND
EF04  B4 04                     MOV     AH,RECORD_NOT_FND
EF06  72 0A                     JC      J19             ; RW_FAIL
EF08  D0 E0                     SAL     AL,1
EF0A  D0 E0                     SAL     AL,1            ; TEST MISSING ADDRESS MARK
EF0C  B4 02                     MOV     AH,BAD_ADDR_MARK
EF0E  72 02                     JC      J19             ; RW_FAIL
                        ;------ NEC MUST HAVE FAILED
EF10                    J18:                            ; RW-NEC-FAIL
EF10  B4 20                     MOV     AH,BAD_NEC
EF12                    J19:                            ; RW-FAIL
EF12  08 26 0041 R              OR      DISKETTE_STATUS,AH
EF16  E8 EAE1 R                 CALL    NUM_TRANS       ; HOW MANY WERE REALLY TRANSFERRED
EF19                    J20:                            ; RW_ERR
EF19  C3                        RET                     ; RETURN TO CALLER
                        ;------ OPERATION WAS SUCCESSFUL
EF1A                    J21_1:
EF1A  8A 5E 0E                  MOV     BL,[BP+14]      ; GET NUMBER OF SECTORS PASSED
                                                        ; FROM STACK
EF1D  E8 EAE1 R                 CALL    NUM_TRANS       ; HOW MANY GOT MOVED, AL CONTAINS
                                                        ; NUM OF SECTORS
EF20  3A D8                     CMP     BL,AL           ; NUMBER REQUESTED=NUMBER ACTUALLY
                                                        ; TRANSFERRED?
EF22  74 0C                     JE      J21_2           ; TRANSFER SUCCESSFUL
                        ;------ OPERATION ATTEMPTED TO ACCESS DATA PAST REAL EOT.  THIS IS
                        ;       A REAL ERROR
EF24  80 0E 0041 R 04           OR      DISKETTE_STATUS ,RECORD_NOT_FND
EF29  C6 06 0043 R 80           MOV     NEC_STATUS+1,80H ; ST1 GETS CORRECT VALUE
EF2E  F9                        STC
EF2F  C3                        RET
EF30  33 C0             J21_2:  XOR     AX,AX           ; CLEAR AX FOR NEC_STATUS UPDATE
EF32  33 F6                     XOR     SI,SI           ; INDEX TO NEC_STATUS ARRAY
EF34  88 84 0042 R              MOV     NEC_STATUS[SI],AL ; ZERO OUT BYTE, STO
EF38  46                        INC     SI              ; POINT INDEX AT SECOND BYTE
EF39  88 84 0042 R              MOV     NEC_STATUS[SI],AL ; ZERO OUT BUYE, ST1
EF3D  EB 03                     JMP     SHORT J21_3     ; OPN_OK
EF3F  E8 EAE1 R         J22:    CALL    NUM_TRANS
EF42  32 E4             J21_3:  XOR     AH,AH                   ; NO ERRORS
EF44  C3                        RET
EF45                    RW_OPN  ENDP
                        ;--------------------------------------------
                        ; DISK_INT
                        ;       THIS ROUTINE HANDLES THE DISKETTE INTERRUPT. AN INTERRUPT
                        ;       WILL OCCUR ONLY WHEN THE ONE-SHOT TIMER IS FIRED.  THIS
                        ;       OCCURS IN AN ERROR SITUATION.  THIS ROUTINE SETS ERRORS IN
                        ;       THE DISKETTE STATUS BYTE AND DISABLES THE ONE-SHOT TIMER.
                        ;       THEN THE RETURN ADDRESS ON THE STACK IS CHANGED TO RETURN
                        ;       TO THE OP_END LABEL.
                        ; INPUT
                        ;       NONE.
                        ; OUTPUT
                        ;       NONE.  DS POINTS AT BIOS DATA AREA.  CARRY FLAG IS SET SO
                        ;       THAT ERROR WILL BE CAUGHT IN THE ENVIRONMEMT RETURNED TO.
                        ;--------------------------------------------
```

```
EF57                            ORG     0EF57H
EF57                    DISK_INT        PROC    FAR
EF57  1E                        PUSH    DS
EF58  50                        PUSH    AX
EF59  52                        PUSH    DX                      ; SAVE REGISTER
EF5A  55                        PUSH    BP                      ; SAVE THE BP REGISTER
EF5B  E8 138B R                 CALL    DDS                     ; SETUP DS TO POINT AT BIOS DATA
                                ;------ CHECK IF INTERRUPT OCCURED IN INT13 OR WHETHER IT IS A
                                SPURIOUS INTERRUPT
EF5E  8B EC                     MOV     BP,SP                   ; POINT BP AT STACK
EF60  0E                        PUSH    CS                      ; WAS IT IN THE BIOS AREA
EF61  58                        POP     AX
EF62  3B 46 0A                  CMP     AX,WORD PTR[BP+10] ; GET INTERRUPTED SEGMENT
EF65  75 48                     JNE     D13                     ; NOT IN BIOS, ERROR CONDITION
EF67  8B 46 08                  MOV     AX,WORD PTR[BP+8] ; GET IP ON THE STACK
EF6A  3D EE20 R                 CMP     AX,OFFSET VERIFY_LOOP ; RANGE CHECK IP FOR DISK
                                                        ; TRANSFER
EF6D  7C 40                     JL      D13                     ; BELOW TRANSFER CODE
EF6F  3D EE66 R                 CMP     AX,OFFSET OP_END+1 ; UPPER RANGE OF TRANSFER CODE
EF72  7D 3B                     JGE     D13                     ; ABOVE RANGE OF WATCHDOG TERRAIN
                                ;-------VALID DISKETTE INTERRUPT CHANGE RETURN ADDRESS ON STACK TO
                                PULL OUT OF LOOP
EF74  C7 46 08 EE65 R           MOV     WORD PTR[BP+8],OFFSET OP_END
EF79  81 4E 0C 0001             OR      WORD PTR[BP+12],1 ; TURN ON CARRY FLAG IN FLAGS ON
                                                        ; STACK
                                ;--------------------------------------------------------------------
                                ;***NOTE***
                                ; A WRITE PROTECTED DISKETTE WILL ALWAYS GET STUCK IN WRITE LOOP
                                ; WAITING FOR BEGINNING OF EXECUTION PHASE.  WHEN THE WATCHDOG
                                ; FIRES AND THE STATUS IN PORT NEC_STAT = DXH (X MEANS DON'T CARE)
                                ; STATUS FROM THE RESULT PHASE IS AVAILABLE.  THE STATUS IS READ
                                ; AND WRITE PROTECT IS  CHECKED FOR.
                                ;--------------------------------------------------------------------
EF7E  BA 00F4                   MOV     DX,NEC_STAT
EF81  EC                        IN      AL,DX                   ; GET NEC STATUS BYTE
EF82  24 F0                     AND     AL,0F0H                 ; MASK HIGH NIBBLE
EF84  3C D0                     CMP     AL,0D0H                 ; IS EXECUTION PHASE DONE
EF86  75 14                     JNE     D11                     ; STUCK IN LOOP
EF88  E8 EAA0 R                 CALL    RESULTS                 ; GET STATUS OF OPERATION
EF8B  BE 0042 R                 MOV     SI,OFFSET NEC_STATUS ; ADDRESS OF BYTES RETURNED BY
                                                        ; NEC
EF8E  8A 44 01                  MOV     AL,[SI+1]               ; GET ST1
EF91  A8 02                     TEST    AL,02H                  ; WRITE PROTECT SIGNAL ACTIVE?
EF93  74 07                     JZ      D11                     ; TIME OUT ERROR
EF95  80 0E 0041 R 03           OR      DISKETTE_STATUS,WRITE_PROTECT
EF9A  EB 13                     JMP     SHORT D13
                                ;------TIME OUT ERROR
EF9C  80 0E 0041 R 80           D11:    OR      DISKETTE_STATUS,TIME_OUT
EFA1  C6 06 003E R 00           MOV     SEEK_STATUS,0   ; SET RECAL ON DRIVES
                                ;------ RESET THE NEC AND DISABLE WATCHDOG
EFA6  8A 00F2                   D12:    MOV     DX,NEC_CTL      ; ADDRESS TO NEC CONTROL PORT
EFA9  5D                        POP     BP                      ; POINT BP AT BASE OF STACKED
                                                        ; PARAMETERS
EFAA  E8 EB45 R                 CALL    GET_DRIVE       ; RESET ADAPTER AND DISABLE WD
EFAD  55                        PUSH    BP              ; RESTORE FOR RETURNED CALL
EFAE  EE                        OUT     DX,AL
EFAF  B0 20                     D13:    MOV     AL,EOI          ; GIVE EOI TO 8259
EFB1  E6 20                     OUT     INTA00,AL
EFB3  5D                        POP     BP
EFB4  5A                        POP     DX
EFB5  58                        POP     AX
EFB6  1F                        POP     DS
EFB7  CF                        IRET                    ; RETURN FROM INTERRUPT
EFB8                    DISK_INT        ENDP
                                ;--------------------------------------------------
                                ; DISK_BASE
                                ;  THIS IS THE SET OF PARAMETERS REQUIRED FOR
                                ;  DISKETTE OPERATION.  THEY ARE POINTED AT BY THE
                                ;  DATA VARIABLE DISK_POINTER.  TO MODIFY THE PARAMETERS,
                                ;  BUILD ANOTHER PARAMETER BLOCK AND POINT AT IT
                                ;--------------------------------------------------
EFC7                            ORG     0EFC7H
EFC7                    DISK_BASE       LABEL   BYTE
EFC7  CF                        DB      11001111B       ; SRT=C, HD UNLOAD=0F - 1ST SPECIFY
                                                        ; BYTE
EFC8  03                        DB      3               ; HD LOAD=1, MODE=NO DMA - 2ND
                                                        ; SPECIFY BYTE
EFC9  25                        DB      MOTOR_WAIT      ; WAIT AFTER OPN TIL MOTOR OFF
EFCA  02                        DB      2               ; 512 BYTES/SECTOR
EFCB  08                        DB      8               ; EOT ( LAST SECTOR ON TRACK)
EFCC  2A                        DB      02AH            ; GAP LENGTH
EFCD  FF                        DB      0FFH            ; DTL
EFCE  50                        DB      050H            ; GAP LENGTH FOR FORMAT
EFCF  F6                        DB      0F6H            ; FILL BYTE FOR FORMAT
EFD0  19                        DB      25              ; HEAD SETTLE TIME (MILLISECONDS)
EFD1  04                        DB      4               ; MOTOR START TIME (1/8 SECONDS)
```

```
                        ;--- INT 17 ------------------------------------------------------
                        ; PRINTER_IO
                        ;       THIS ROUTINE PROVIDES COMMUNICATION WITH THE PRINTER
                        ;       (AH)=0  PRINT THE CHARACTER IN (AL)
                        ;               ON RETURN, AH=1 IF CHARACTER COULD NOT BE PRINTED
                        ;               (TIME OUT), OTHER BITS SET AS ON NORMAL STATUS CALL
                        ;       (AH)=1  INITIALIZE THE PRINTER PORT
                        ;               RETURNS WITH (AH) SET WITH PRINTER STATUS
                        ;       (AH)=2  READ THE PRINTER STATUS INTO (AH)
                        ;       7       6       5       4       3       2-1     0
                        ;       :       :       :       :       :       :       :_ TIME OUT
                        ;       :       :       :       :       :       :_ UNUSED
                        ;       :       :       :       :       :_ 1 = I/O ERROR
                        ;       :       :       :       :_ 1 = SELECTED
                        ;       :       :       :_ 1 = OUT OF PAPER
                        ;       :       :_ 1 = ACKNOWLEDGE
                        ;       :_ 1 = NOT BUSY
                        ;
                        ;       (DX) = PRINTER TO BE USED (0,1,2) CORRESPONDING TO ACTUAL
                        ;               VALUES IN PRINTER_BASE AREA
                        ; DATA AREA PRINTER_BASE CONTAINS THE BASE ADDRESS OF THE PRINTER
                        ; CARD(S) AVAILABLE (LOCATED AT BEGINNING OF DATA SEGMENT, 408H
                        ; ABSOLUTE, 3 WORDS), UNLESS THERE IS ONLY A SERIAL PRINTER
                        ; ATTACHED, IN WHICH CASE THE WORD AT 40:8 WILL CONTAIN A 02F8H.
                        ; REGISTERS     AH IS MODIFIED
                        ;               ALL OTHERS UNCHANGED
                        ;----------------------------------------------------------------
                                ASSUME  CS:CODE,DS:DATA
EFD2                            ORG     0EFD2H
EFD2                    PRINTER_IO      PROC    FAR
EFD2  FB                                STI                     ; INTERRUPTS BACK ON
EFD3  1E                                PUSH    DS              ; SAVE SEGMENT
EFD4  52                                PUSH    DX
EFD5  56                                PUSH    SI
EFD6  51                                PUSH    CX
EFD7  53                                PUSH    BX
EFD8  E8 138B R                         CALL    DDS
                        ;REDIRECT TO SERIAL ONLY IF:
                        ; 1> SERIAL PRINTER IS ATTACHED,AND...
                        ; 2> WORD AT PRINTER BASE = 02F8H.
                        ; POWER ONS WILL ONLY PUT A 02F8H IN THE PRINTER BASE IF THERE'S
                        ; NO PARALLEL PRINTER ATTACHED.
EFDB  8B 0E 0010 R                      MOV     CX,EQUIP_FLAG   ;GET FLAG IN CX
EFDF  F6 C5 20                          TEST    CH,00100000B    ;SERIAL ATTACHED?
EFE2  74 0D                             JZ      B0              ;NO -HANDLE NORMALLY
EFE4  8B 1E 0008 R                      MOV     BX,PRINTER_BASE ;SEE IF THERE'S AN RS232
EFE8  81 FB 02F8                        CMP     BX,02F8H        ;BASE IN THE PRINTER BASE.
EFEC  75 03                             JNE     B0
EFEE  E9 18C3 R                 B00:    JMP     B1_A            ;IF THERE IS REDIRECT
                                                                ;ELSE... HANDLE AS PARALLEL
                        ;CONTROL IS PASSED TO THIS POINT IF THERE IS A PARALLEL OR
                        ;THERE'S NO SERIAL PRINTER ATTACHED.
EFF1  8B F2                     B0:     MOV     SI,DX           ; GET PRINTER PARM
EFF3  8A 9C 0078 R                      MOV     BL,PRINT_TIM_OUT[SI] ; LOAD TIMEOUT VALUE
EFF7  D1 E6                             SHL     SI,1            ; WORD OFFSET INTO TABLE
EFF9  8B 94 0008 R                      MOV     DX,PRINTER_BASE[SI] ; GET BASE ADDRESS FOR PRINTER
                                                                ; CARD
EFFD  0B D2                             OR      DX,DX           ; TEST DX FOR ZERO, INDICATING NO
                                                                ; PRINTER
EFFF  74 0C                             JZ      B1              ; IF NO PARALLEL, RETURN
F001  0A E4                             OR      AH,AH           ; TEST FOR (AH)=0
F003  74 0E                             JZ      B2              ; PRINT_AL
F005  FE CC                             DEC     AH              ; TEST FOR (AH)=1
F007  74 40                             JZ      B8              ; INIT_PRT
F009  FE CC                             DEC     AH              ; TEST FOR (AH)=2
F00B  74 28                             JZ      B5              ; PRINTER STATUS
F00D                            B1:                             ; RETURN
F00D  5B                                POP     BX
F00E  59                                POP     CX
F00F  5E                                POP     SI              ; RECOVER REGISTERS
F010  5A                                POP     DX              ; RECOVER REGISTERS
F011  1F                                POP     DS
F012  CF                                IRET
                        ;------ PRINT THE CHARACTER IN (AL)
F013  50                        B2:     PUSH    AX              ; SAVE VALUE TO PRINT
F014  EE                                OUT     DX,AL           ; OUTPUT CHAR TO PORT
F015  42                                INC     DX              ; POINT TO STATUS PORT
                                ;
                                ;------WAIT BUSY
                                ;
F016  2B C9                     B3:     SUB     CX,CX           ; INNER LOOP (64K)
F018  EC                        B3_1:   IN      AL,DX           ; GET STATUS
F019  8A E0                             MOV     AH,AL           ; STATUS TO AH ALSO
F01B  A8 80                             TEST    AL,80H          ; IS THE PRINTER CURRENTLY BUSY
F01D  75 0E                             JNZ     B4              ; OUT_STROBE
F01F  E2 F7                             LOOP    B3_1            ; LOOP IF NOT
F021  FE CB                             DEC     BL              ; DROP OUTER LOOP COUNT
F023  75 F1                             JNZ     B3              ; MAKE ANOTHER PASS IF NOT ZERO
F025  80 CC 01                          OR      AH,1            ; SET ERROR FLAG
F028  80 E4 F9                          AND     AH,0F9H         ; TURN OFF THE UNUSED BITS
F02B  EB 14                             JMP     SHORT B7        ; RETURN WITH ERROR FLAG SET
F02D                            B4:                             ; OUT_STROBE
F02D  B0 0D                             MOV     AL,0DH          ; SET THE STROBE HIGH
F02F  42                                INC     DX
F030  EE                                OUT     DX,AL
F031  B0 0C                             MOV     AL,0CH          ; SET THE STROBE LOW
F033  EE                                OUT     DX,AL
F034  58                                POP     AX              ; RECOVER THE OUTPUT CHAR
```

```
                        ;------ PRINTER STATUS
F035  50                B5:     PUSH    AX              ; SAVE AL REG
F036  8B 94 0008 R      B6:     MOV     DX,PRINTER_BASE[SI]
F03A  42                        INC     DX
F03B  EC                        IN      AL,DX           ; GET PRINTER STATUS
F03C  8A E0                     MOV     AH,AL
F03E  80 E4 F8                  AND     AH,0F8H         ; TURN OFF UNUSED BITS
F041                    B7:                             ; STATUS_SET
F041  5A                        POP     DX              ; RECOVER AL REG
F042  8A C2                     MOV     AL,DL           ; GET CHARACTER INTO AL
F044  80 F4 48                  XOR     AH,48H          ; FLIP A COUPLE OF BITS
F047  EB C4                     JMP     B1              ; RETURN FROM ROUTINE
                        ;------ INITIALIZE THE PRINTER PORT
F049  50                B8:     PUSH    AX              ; SAVE AL
F04A  42                        INC     DX              ; POINT TO OUTPUT PORT
F04B  42                        INC     DX
F04C  B0 08                     MOV     AL,8            ; SET INIT LINE LOW
F04E  EE                        OUT     DX,AL
F04F  B8 03E8                   MOV     AX,1000
F052                    B9:                             ; INIT_LOOP
F052  48                        DEC     AX              ; LOOP FOR RESET TO TAKE
F053  75 FD                     JNZ     B9              ; INIT_LOOP
F055  B0 0C                     MOV     AL,0CH          ; NO INTERRUPTS, NON AUTO LF, INIT
                                                        ; HIGH
F057  EE                        OUT     DX,AL
F058  EB DC                     JMP     B6              ; PRT_STATUS_1
F05A                    PRINTER_IO      ENDP
F065                            ORG     0F065H
F065  E9 0D0B R                 JMP     NEAR PTR VIDEO_IO
                        ;--------------------------------------------
                        ;    SUBROUTINE TO SAVE  ANY SCAN CODE RECEIVED ;
                        ;    BY THE NMI ROUTINE (PASSED IN AL)          ;
                        ;    DURING POST IN THE KEYBOARD BUFFER         ;
                        ;    CALLED THROUGH INT. 48H                    ;
                        ;--------------------------------------------
F068                    KEY_SCAN_SAVE   PROC    FAR
                                ASSUME  DS:DATA
F068  E8 138B R                 CALL    DDS             ; POINT DS TO DATA AREA
F06B  BE 001E R                 MOV     SI,OFFSET KB_BUFFER ; POINT TO FIRST LOC. IN BUFFER
F06E  88 04                     MOV     [SI],AL         ; SAVE SCAN CODE
F070  8B C4                     MOV     AX,SP           ; CHECK FOR STACK UNDERFLOW
F072  80 E4 E0                  AND     AH,11100000B    ; (THESE BITS WILL BE 111 IF
                                                        ; UNDERFLOW HAPPEND)
F075  74 0D                     JZ      KS_1
F077  32 C0                     XOR     AL,AL
F079  E6 A0                     OUT     0A0H,AL         ; SHUT OFF NMI
F07B  BB 2000                   MOV     BX,2000H        ; ERROR CODE 2000H
F07E  BE 0036 R                 MOV     SI,OFFSET KEY_ERR ; POST MESSAGE
F081  E8 09BC R                 CALL    E_MSG           ; AND HALT SYSTEM
F084  CF                KS_1:   IRET                    ; RETURN TO CALLER
F085                    KEY_SCAN_SAVE   ENDP
                        ;--------------------------------------------------------------------
                        ;   SUBROUTINE TO SET AN INS8250 CHIP'S BAUD RATE TO 9600 BPS AND
                        ;   DEFINE IT'S DATA WORD AS HAVING 8 BITS/WORD, 2 STOP BITS, AND
                        ;   ODD PARITY.
                        ;
                        ;   EXPECTS TO BE PASSED:
                        ;       (DX) = LINE CONTROL REGISTER
                        ;
                        ;   UPON RETURN:
                        ;       (DX) = TRANSMIT/RECEIVE BUFFER ADDRESS
                        ;
                        ;   ALSO, ALTERS REGISTER AL.  ALL OTHERS REMAIN INTACT.
                        ;--------------------------------------------------------------------
F085                    S8250   PROC    NEAR
F085  B0 80                     MOV     AL,80H          ; SET DLAB = 1
F087  EE                        OUT     DX,AL
F088  EB 00                     JMP     $+2             ; I/O DELAY
F08A  83 EA 03                  SUB     DX,3            ; LSB OF DIVISOR LATCH
F08D  B0 0C                     MOV     AL,12           ; DIVISOR = 12 PRODUCES 9600 BPS
F08F  EE                        OUT     DX,AL           ; SET LSB
F090  EB 00                     JMP     $+2             ; I/O DELAY
F092  42                        INC     DX              ; MSB OF DIVISOR LATCH
F093  B0 00                     MOV     AL,0            ; HIGH ORDER OF DIVISORS
F095  EE                        OUT     DX,AL           ; SET MSB
F096  EB 00                     JMP     $+2             ; I/O DELAY
F098  42                        INC     DX
F099  42                        INC     DX              ; LINE CONTROL REGISTER
F09A  B0 0F                     MOV     AL,00001111B    ; 8 BITS/WORD, 2 STOP BITS, ODD
                                                        ; PARITY
F09C  EE                        OUT     DX,AL
F09D  EB 00                     JMP     $+2             ; I/O DELAY
F09F  83 EA 03                  SUB     DX,3            ; RECEIVER BUFFER
F0A2  EC                        IN      AL,DX           ; IN CASE WRITING TO PORT LCR
                                                        ; CAUSED DATA READY TO GO HIGH!
F0A3  C3                        RET
F0A4                    S8250   ENDP
                        ;------ TABLES FOR USE IN SETTING OF CRT MODE
F0A4                            ORG     0F0A4H
F0A4                    VIDEO_PARMS     LABEL   BYTE
                        ;------ INIT_TABLE
F0A4  38 28 2C 06 1F 06         DB      38H,28H,2CH,06H,1FH,6,19H ; SETUP FOR 40X25
      19
F0AB  1C 02 07 06 07            DB      1CH,2,7,6,7
F0B0  00 00 00 00               DB      0,0,0,0
```

```
= 0010                          M0040   EQU     $-VIDEO_PARMS

F0B4  71 50 5A 0C 1F 06                 DB      71H,50H,5AH,0CH,1FH,6,19H ; SETUP FOR 80X25
      19
F0BB  1C 02 07 06 07                    DB      1CH,2,7,6,7
F0C0  00 00 00 00                       DB      0,0,0,0

F0C4  38 28 2B 06 7F 06                 DB      38H,28H,2BH,06H,7FH,6,64H ; SET UP FOR GRAPHICS
      64
F0CB  70 02 01 26 07                    DB      70H,2,1,26H,7
F0D0  00 00 00 00                       DB      0,0,0,0

F0D4  71 50 56 0C 3F 06                 DB      71H,50H,56H,0CH,3FH,6,32H ; SET UP FOR GRAPHICS
      32
F0DB  38 02 03 26 07                    DB      38H,2,3,26H,7             ; USING 32K OF MEMORY
F0E0  00 00 00 00                       DB      0,0,0,0                   ; (MODES 9 & A)
                                ;--------------------------------------------
                                ; READ_AC_CURRENT
                                ;       THIS ROUTINE READS THE ATTRIBUTE AND CHARACTER AT THE
                                ;       CURRENT CURSOR POSITION AND RETURNS THEM TO THE CALLER
                                ; INPUT
                                ;       (AH) = CURRENT CRT MODE
                                ;       (BH) = DISPLAY PAGE ( ALPHA MODES ONLY )
                                ;       (DS) = DATA SEGMENT
                                ;       (ES) = REGEN SEGMENT
                                ; OUTPUT
                                ;       (AL) = CHAR READ
                                ;       (AH) = ATTRIBUTE READ
                                ;--------------------------------------------
                                        ASSUME  CS:CODE,DS:DATA,ES:DATA
F0E4                            READ_AC_CURRENT PROC    NEAR
F0E4  80 FC 04                          CMP     AH,4                    ; IS THIS GRAPHICS?
F0E7  72 03                             JC      C60
F0E9  E9 F531 R                         JMP     GRAPHICS_READ
F0EC                            C60:                                    ; READ_AC_CONTINUE
F0EC  E8 F0F7 R                         CALL    FIND_POSITION
F0EF  8B F3                             MOV     SI,BX                   ; ESTABLISH ADDRESSING IN SI
F0F1  06                                PUSH    ES                      ;
F0F2  1F                                POP     DS                      ; GET SEGMENT FOR QUICK ACCESS
F0F3  AD                                LODSW                           ; GET THE CHAR/ATTR
F0F4  E9 0F70 R                         JMP     VIDEO_RETURN
F0F7                            READ_AC_CURRENT ENDP
F0F7                            FIND_POSITION   PROC    NEAR
F0F7  8A CF                             MOV     CL,BH                   ; DISPLAY PAGE TO CX
F0F9  32 ED                             XOR     CH,CH
F0FB  8B F1                             MOV     SI,CX                   ; MOVE TO SI FOR INDEX
F0FD  D1 E6                             SAL     SI,1                    ; * 2 FOR WORD OFFSET
F0FF  8B 84 0050 R                      MOV     AX,[SI+ OFFSET CURSOR_POSN] ; GET ROW/COLUMN OF
                                                                        ; THAT PAGE
F103  33 DB                             XOR     BX,BX                   ; SET START ADDRESS TO ZERO
F105  E3 06                             JCXZ    C62                     ; NO_PAGE
F107                            C61:                                    ; PAGE_LOOP
F107  03 1E 004C R                      ADD     BX,CRT_LEN              ; LENGTH OF BUFFER
F10B  E2 FA                             LOOP    C61
F10D                            C62:                                    ; NO_PAGE
F10D  E8 E5C2 R                         CALL    POSITION                ; DETERMINE LOCATION IN REGEN
F110  03 D8                             ADD     BX,AX                   ; ADD TO START OF REGEN
F112  C3                                RET
F113                            FIND_POSITION   ENDP
                                ;--------------------------------------------
                                ; WRITE_AC_CURRENT
                                ;       THIS ROUTINE WRITES THE ATTRIBUTE AND CHARACTER AT
                                ;       THE CURRENT CURSOR POSITION
                                ; INPUT
                                ;       (AH) = CURRENT CRT MODE
                                ;       (BH) = DISPLAY PAGE
                                ;       (CX) = COUNT OF CHARACTERS TO WRITE
                                ;       (AL) = CHAR TO WRITE
                                ;       (BL) = ATTRIBUTE OF CHAR TO WRITE
                                ;       (DS) = DATA SEGMENT
                                ;       (ES) = REGEN SEGMENT
                                ; OUTPUT
                                ;       NONE
                                ;--------------------------------------------
F113                            WRITE_AC_CURRENT        PROC    NEAR
F113  80 FC 04                          CMP     AH,4                    ; IS THIS GRAPHICS?
F116  72 03                             JC      C63
F118  E9 F3F1 R                         JMP     GRAPHICS_WRITE
F11B                            C63:                                    ; WRITE_AC_CONTINUE
F11B  8A E3                             MOV     AH,BL                   ; GET ATTRIBUTE TO AH
F11D  50                                PUSH    AX                      ; SAVE ON STACK
F11E  51                                PUSH    CX                      ; SAVE WRITE COUNT
F11F  E8 F0F7 R                         CALL    FIND_POSITION
F122  8B FB                             MOV     DI,BX                   ; ADDRESS TO DI REGISTER
F124  59                                POP     CX                      ; WRITE COUNT
F125  58                                POP     AX                      ; CHARACTER IN AX REG
F126                            C64:                                    ; WRITE_LOOP
F126  AB                                STOSW                           ; PUT THE CHAR/ATTR
F127  E2 FD                             LOOP    C64                     ;  AS MANY TIMES AS REQUESTED
F129  E9 0F70 R                         JMP     VIDEO_RETURN
F12C                            WRITE_AC_CURRENT        ENDP
```

```
                        ;--------------------------------------------
                        ; WRITE_C_CURRENT
                        ;       THIS ROUTINE WRITES THE CHARACTER AT
                        ;       THE CURRENT CURSOR POSITION, ATTRIBUTE UNCHANGED
                        ; INPUT
                        ;       (AH) = CURRENT CRT MODE
                        ;       (BH) = DISPLAY PAGE
                        ;       (CX) = COUNT OF CHARACTERS TO WRITE
                        ;       (AL) = CHAR TO WRITE
                        ;       (DS) = DATA SEGMENT
                        ;       (ES) = REGEN SEGMENT
                        ; OUTPUT
                        ;       NONE
                        ;--------------------------------------------
F12C                    WRITE_C_CURRENT PROC    NEAR
F12C  80 FC 04                  CMP     AH,4                    ; IS THIS GRAPHICS?
F12F  72 03                     JC      C65
F131  E9 F3F1 R                 JMP     GRAPHICS_WRITE
F134  50                C65:    PUSH    AX                      ; SAVE ON STACK
F135  51                        PUSH    CX                      ; SAVE WRITE COUNT
F136  E8 F0F7 R                 CALL    FIND_POSITION
F139  8B FB                     MOV     DI,BX                   ; ADDRESS TO DI
F13B  59                        POP     CX                      ; WRITE COUNT
F13C  5B                        POP     BX                      ; BL HAS CHAR TO WRITE
F13D                    C66:                                    ; WRITE_LOOP
F13D  8A C3                     MOV     AL,BL                   ; RECOVER CHAR
F13F  AA                        STOSB                           ; PUT THE CHAR/ATTR
F140  47                        INC     DI                      ; BUMP POINTER PAST ATTRIBUTE
F141  E2 FA                     LOOP    C66                     ;  AS MANY TIMES AS REQUESTED
F143  E9 0F70 R                 JMP     VIDEO_RETURN
F146                    WRITE_C_CURRENT ENDP
                        ;--------------------------------------------
                        ; READ DOT  -- WRITE DOT
                        ; THESE ROUTINES WILL WRITE A DOT, OR READ THE
                        ;  DOT AT THE INDICATED LOCATION
                        ; ENTRY --
                        ;   DX = ROW (0-199)    (THE ACTUAL VALUE DEPENDS ON THE MODE)
                        ;   CX = COLUMN ( 0-639) ( THE VALUES ARE NOT RANGE CHECKED )
                        ;   AL = DOT VALUE TO WRITE (1,2 OR 4 BITS DEPENDING ON MODE,
                        ;        REQ'D FOR WRITE DOT ONLY, RIGHT JUSTIFIED)
                        ;        BIT 7 OF AL = 1 INDICATES XOR THE VALUE INTO THE LOCATION
                        ;   DS = DATA SEGMENT
                        ;   ES = REGEN SEGMENT
                        ;
                        ; EXIT
                        ;       AL = DOT VALUE READ, RIGHT JUSTIFIED, READ ONLY
                        ;--------------------------------------------
                                ASSUME  CS:CODE,DS:DATA,ES:DATA
F146                    READ_DOT        PROC    NEAR
F146  80 3E 0049 R 0A           CMP     CRT_MODE,0AH            ; 640X200 4 COLOR?
F14B  74 11                     JE      READ_ODD                ; YES, HANDLE SEPARATELY
F14D  E8 F1D9 R                 CALL    C72                     ; DETERMINE BYTE POSITION OF DOT
F150  26: 8A 04                 MOV     AL,ES:[SI]              ; GET THE BYTE
F153  22 C4                     AND     AL,AH                   ; MASK OFF THE OTHER BITS IN THE
                                                                ; BYTE
F155  D2 E0                     SHL     AL,CL                   ; LEFT JUSTIFY THE VALUE
F157  8A CE                     MOV     CL,DH                   ; GET NUMBER OF BITS IN RESULT
F159  D2 C0                     ROL     AL,CL                   ; RIGHT JUSTIFY THE RESULT
F15B  E9 0F70 R                 JMP     VIDEO_RETURN            ; RETURN FROM VIDEO IO
                        ; IN 640X200 4 COLOR MODE, THE 2 COLOR BITS (C1,C0) ARE DIFFERENT
                        ; THAN OTHER MODES.  C0 IS IN THE EVEN BYTE, C1 IS IN THE FOLLOWING
                        ; ODD BYTE - BOTH AT THE SAME BIT POSITION WITHIN THEIR RESPECTIVE
                        ; BYTES.
F15E                    READ_ODD:
F15E  E8 F1D9 R                 CALL    C72                     ; DETERMINE POSITION OF DOT
F161  52                        PUSH    DX                      ; SAVE INFO
F162  51                        PUSH    CX
F163  50                        PUSH    AX
F164  26: 8A 44 01              MOV     AL,ES:[SI+1]            ; GET C1 COLOR BIT FROM ODD BYTE
F168  22 C4                     AND     AL,AH                   ; MASK OFF OTHER BITS
F16A  D2 E0                     SHL     AL,CL                   ; LEFT JUSTIFY THE VALUE
F16C  8A CE                     MOV     CL,DH                   ; GET NUMBER OF BITS IN RESULT
F16E  FE C1                     INC     CL
F170  D2 C0                     ROL     AL,CL                   ; RIGHT JUSTIFY THE RESULT
F172  8B D8                     MOV     BX,AX                   ; SAVE IN BX REG
F174  58                        POP     AX                      ; RESTORE POSITION INFO
F175  59                        POP     CX                      ;
F176  5A                        POP     DX
F177  26: 8A 04                 MOV     AL,ES:[SI]              ; GET C0 COLOR BIT FROM EVEN BYTE
F17A  22 C4                     AND     AL,AH                   ; MASK OFF OTHER BITS
F17C  D2 E0                     SHL     AL,CL                   ; LEFT JUSTIFY THE VALUE
F17E  8A CE                     MOV     CL,DH                   ; GET NUMBER OF BITS IN RESULT
F180  D2 C0                     ROL     AL,CL                   ; RIGHT JUSTIFY THE RESULT
F182  0A C3                     OR      AL,BL                   ; COMBINE C1 & C0
F184  E9 0F70 R                 JMP     VIDEO_RETURN
```

```
F187                            READ_DOT        ENDP
F187                            WRITE_DOT       PROC    NEAR
F187  51                                PUSH    CX              ; SAVE COL
F188  52                                PUSH    DX              ; SAVE ROW
F189  50                                PUSH    AX              ; SAVE DOT VALUE
F18A  50                                PUSH    AX              ;  TWICE
F18B  E8 F1D9 R                         CALL    C72             ; DETERMINE BYTE POSITION OF THE
                                                                ; DOT
F18E  D2 E8                             SHR     AL,CL           ; SHIFT TO SET UP THE BITS FOR
                                                                ; OUTPUT
F190  22 C4                             AND     AL,AH           ; STRIP OFF THE OTHER BITS
F192  26: 8A 0C                         MOV     CL,ES:[SI]      ; GET THE CURRENT BYTE
F195  5B                                POP     BX              ; RECOVER XOR FLAG
F196  F6 C3 80                          TEST    BL,80H          ; IS IT ON
F199  75 36                             JNZ     C70             ; YES, XOR THE DOT
F19B  F6 D4                             NOT     AH              ; SET THE MASK TO REMOVE THE
                                                                ; INDICATED BITS
F19D  22 CC                             AND     CL,AH
F19F  0A C1                             OR      AL,CL           ; OR IN THE NEW VALUE OF THOSE BITS
F1A1                            C67:                            ; FINISH_DOT
F1A1  26: 88 04                         MOV     ES:[SI],AL      ; RESTORE THE BYTE IN MEMORY
F1A4  58                                POP     AX
F1A5  5A                                POP     DX              ; RECOVER ROW
F1A6  59                                POP     CX              ; RECOVER COL
F1A7  80 3E 0049 R 0A                   CMP     CRT_MODE,0AH    ; 640X200 4 COLOR?
F1AC  75 20                             JNE     C69             ; NO, JUMP
F1AE  50                                PUSH    AX              ; SAVE DOT VALUE
F1AF  50                                PUSH    AX              ;  TWICE
F1B0  D0 E8                             SHR     AL,1            ; SHIFT c1 BIT INTO c0 POSITION
F1B2  E8 F1D9 R                         CALL    C72             ; DETERMINE BYTE POSITION OF THE
                                                                ; DOT
F1B5  D2 E8                             SHR     AL,CL           ; SHIFT TO SET UP THE BITS FOR
                                                                ; OUTPUT
F1B7  22 C4                             AND     AL,AH           ; STRIP OFF THE OTHER BITS
F1B9  26: 8A 4C 01                      MOV     CL,ES:[SI+1]    ; GET THE CURRENT BYTE
F1BD  5B                                POP     BX              ; RECOVER XOR FLAG
F1BE  F6 C3 80                          TEST    BL,80H          ; IS IT ON
F1C1  75 12                             JNZ     C71             ; YES, XOR THE DOT
F1C3  F6 D4                             NOT     AH              ; SET THE MASK TO REMOVE THE
                                                                ; INDICATED BITS
F1C5  22 CC                             AND     CL,AH
F1C7  0A C1                             OR      AL,CL           ; OR IN THE NEW VALUE OF THOSE BITS
F1C9                            C68:                            ; FINISH_DOT
F1C9  26: 88 44 01                      MOV     ES:[SI+1],AL    ; RESTORE THE BYTE IN MEMORY
F1CD  58                                POP     AX
F1CE  E9 0F70 R                 C69:    JMP     VIDEO_RETURN    ; RETURN FROM VIDEO IO
F1D1                            C70:                            ; XOR_DOT
F1D1  32 C1                             XOR     AL,CL           ; EXCLUSIVE OR THE DOTS
F1D3  EB CC                             JMP     C67             ; FINISH UP THE WRITING
F1D5                            C71:                            ; XOR_DOT
F1D5  32 C1                             XOR     AL,CL           ; EXCLUSIVE OR THE DOTS
F1D7  EB F0                             JMP     C68             ; FINISH UP THE WRITING
F1D9                            WRITE_DOT       ENDP
                                ;--------------------------------------------
                                ; THIS SUBROUTINE DETERMINES THE REGEN BYTE LOCATION OF THE
                                ; INDICATED ROW COLUMN VALUE IN GRAPHICS MODE.
                                ; ENTRY --
                                ;   DX = ROW VALUE (0-199)
                                ;   CX = COLUMN VALUE (0-639)
                                ; EXIT --
                                ;   SI = OFFSET INTO REGEN BUFFER FOR BYTE OF INTEREST
                                ;   AH = MASK TO STRIP OFF THE BITS OF INTEREST
                                ;   CL = BITS TO SHIFT TO RIGHT JUSTIFY THE MASK IN AH
                                ;   DH = # BITS IN RESULT
                                ;--------------------------------------------
F1D9                            C72     PROC    NEAR
F1D9  53                                PUSH    BX              ; SAVE BX DURING OPERATION
F1DA  50                                PUSH    AX              ; WILL SAVE AL DURING OPERATION
                                ;------ DETERMINE 1ST BYTE IN IDICATED ROW BY MULTIPLYING ROW VALUE
                                ;       BY 40( LOW BIT OF ROW DETERMINES EVEN/ODD, 80 BYTES/ROW
F1DB  B0 28                             MOV     AL,40
F1DD  52                                PUSH    DX              ; SAVE ROW VALUE
F1DE  80 E2 FE                          AND     DL,0FEH         ; STRIP OFF ODD/EVEN BIT
F1E1  80 3E 0049 R 09                   CMP     CRT_MODE,09H    ; MODE USING 32K REGEN?
F1E6  72 03                             JC      C73             ; NO,  JUMP
F1E8  80 E2 FC                          AND     DL,0FCH         ; STRIP OFF LOW 2 BITS
F1EB  F6 E2                     C73:    MUL     DL              ; AX HAS ADDRESS OF 1ST BYTE OF
                                                                ; INDICATED ROW
F1ED  5A                                POP     DX              ; RECOVER IT
F1EE  F6 C2 01                          TEST    DL,1            ; TEST FOR EVEN/ODD
F1F1  74 03                             JZ      C74             ; JUMP IF EVEN ROW
F1F3  05 2000                           ADD     AX,2000H        ; OFFSET TO LOCATION OF ODD ROWS
F1F6                            C74:                            ; EVEN_ROW
F1F6  80 3E 0049 R 09                   CMP     CRT_MODE,09H    ; MODE USING 32K REGEN?
F1FB  72 08                             JC      C75             ; NO,  JUMP
F1FD  F6 C2 02                          TEST    DL,2            ; TEST FOR ROW 2 OR ROW 3
F200  74 03                             JZ      C75             ; JUMP IF ROW 0 OR 1
F202  05 4000                           ADD     AX,4000H        ; OFFSET TO LOCATION OF ROW 2 OR 3
F205  8B F0                     C75:    MOV     SI,AX           ; MOVE POINTER TO SI
F207  58                                POP     AX              ; RECOVER AL VALUE
F208  8B D1                             MOV     DX,CX           ; COLUMN VALUE TO DX
```

```
                        ;------ DETERMINE GRAPHICS MODE CURRENTLY IN EFFECT
                        ;SET UP THE REGISTERS ACCORDING TO THE MODE
                        ;CH = MASK FOR LOW OF COLUMN ADDRESS ( 7/3/1 FOR HIGH/MED/LOW RES)
                        ;CL = # OF ADDRESS BITS IN COLUMN VALUE ( 3/2/1 FOR H/M/L)
                        ;BL = MASK TO SELECT BITS FROM POINTED BYTE (80H/C0H/F0H FOR H/M/L)
                        ;BH = NUMBER OF VALID BITS IN POINTED BYTE ( 1/2/4 FOR H/M/L)
F20A  BB 02C0                   MOV     BX,2C0H
F20D  B9 0302                   MOV     CX,302H         ; SET PARMS FOR MED RES
F210  80 3E 0049 R 04           CMP     CRT_MODE,4
F215  74 21                     JE      C77             ; HANDLE IF MED RES
F217  80 3E 0049 R 05           CMP     CRT_MODE,5
F21C  74 1A                     JE      C77             ; HANDLE IF MED RES
F21E  BB 04F0                   MOV     BX,4F0H
F221  B9 0101                   MOV     CX,101H         ;SET PARMS FOR LOW RES
F224  80 3E 0049 R 0A           CMP     CRT_MODE,0AH
F229  74 07                     JE      C76             ; HANDLE MODE A AS HIGH RES
F22B  80 3E 0049 R 06           CMP     CRT_MODE,6
F230  75 06                     JNE     C77             ; HANDLE IF LOW RES
F232  BB 0180           C76:    MOV     BX,180H
F235  B9 0703                   MOV     CX,703H         ; SET PARMS FOR HIGH RES
                        ;------ DETERMINE BIT OFFSET IN BYTE FROM COLUMN MASK
F238  22 EA             C77:    AND     CH,DL           ; ADDRESS OF PEL WITHIN BYTE TO CH
                        ;------ DETERMINE BYTE OFFSET FOR THIS LOCATION IN COLUMN
F23A  D3 EA                     SHR     DX,CL           ; SHIFT BY CORRECT AMOUNT
F23C  03 F2                     ADD     SI,DX           ; INCREMENT THE POINTER
F23E  80 3E 0049 R 0A           CMP     CRT_MODE,0AH    ; 640X200 4 COLOR?
F243  75 02                     JNE     C78             ; NO, JUMP
F245  03 F2                     ADD     SI,DX           ; INCREMENT THE POINTER
F247  8A F7             C78:    MOV     DH,BH           ; GET THE # OF BITS IN RESULT TO DH
                        ;------ MULTIPLY BH (VALID BITS IN BYTE) BY CH (BIT OFFSET)
F249  2A C9                     SUB     CL,CL           ; ZERO INTO STORAGE LOCATION
F24B  D0 C8             C79:    ROR     AL,1            ; LEFT JUSTIFY THE VALUE IN AL
                                                        ; (FOR WRITE)
F24D  02 CD                     ADD     CL,CH           ; ADD IN THE BIT OFFSET VALUE
F24F  FE CF                     DEC     BH              ; LOOP CONTROL
F251  75 F8                     JNZ     C79             ; ON EXIT, CL HAS SHIFT COUNT TO
                                                        ; RESTORE BITS
F253  8A E3                     MOV     AH,BL           ;  GET MASK TO AH
F255  D2 EC                     SHR     AH,CL           ;  MOVE THE MASK TO CORRECT
                                                        ; LOCATION
F257  5B                        POP     BX              ;  RECOVER REG
F258  C3                        RET                     ;  RETURN WITH EVERYTHING SET UP
F259                    C72      ENDP
                        ;--------------------------------------------------------
                        ;  SCROLL UP
                        ;   THIS ROUTINE SCROLLS UP THE INFORMATION ON THE CRT
                        ; ENTRY --
                        ;  CH,CL = UPPER LEFT CORNER OF REGION TO SCROLL
                        ;  DH,DL = LOWER RIGHT CORNER OF REGION TO SCROLL
                        ;   BOTH OF THE ABOVE ARE IN CHARACTER POSITIONS
                        ;  BH = FILL VALUE FOR BLANKED LINES
                        ;  AL = # LINES TO SCROLL (AL=0 MEANS BLANK THE ENTIRE FIELD)
                        ;  DS = DATA SEGMENT
                        ;  ES = REGEN SEGMENT
                        ; EXIT --
                        ;  NOTHING, THE SCREEN IS SCROLLED
                        ;--------------------------------------------------------
F259                    GRAPHICS_UP     PROC    NEAR
F259  8A D8                     MOV     BL,AL   ; SAVE LINE COUNT IN BL
F25B  8B C1                     MOV     AX,CX   ; GET UPPER LEFT POSITION INTO AX REG
                        ;------ USE CHARACTER SUBROUTINE FOR POSITIONING
                        ;------ ADDRESS RETURNED IS MULTIPLIED BY 2 FROM CORRECT VALUE
F25D  E8 F72C R                 CALL    GRAPH_POSN
F260  8B F8                     MOV     DI,AX           ; SAVE RESULT AS DESTINATION
                                                        ; ADDRESS
                        ;------ DETERMINE SIZE OF WINDOW
F262  2B D1                     SUB     DX,CX
F264  81 C2 0101                ADD     DX,101H         ; ADJUST VALUES
F268  D0 E6                     SAL     DH,1            ; MULTIPLY # ROWS BY 4 SINCE 8 VERT
                                                        ; DOTS/CHAR
F26A  D0 E6                     SAL     DH,1            ;  AND EVEN/ODD ROWS
                        ;------ DETERMINE CRT MODE
F26C  80 3E 0049 R 06           CMP     CRT_MODE,6      ; TEST FOR HIGH RES
F271  74 1D                     JE      C80             ; FIND_SOURCE
                        ;------ MEDIUM RES UP
F273  D0 E2                     SAL     DL,1            ; # COLUMNS * 2, SINCE 2 BYTES/CHAR
F275  D1 E7                     SAL     DI,1            ; OFFSET *2 SINCE 2 BYTES/CHAR
F277  80 3E 0049 R 04           CMP     CRT_MODE,4      ; TEST FOR MEDIUM RES
F27C  74 12                     JE      C80
F27E  80 3E 0049 R 05           CMP     CRT_MODE,5      ; TEST FOR MEDIUM RES
F283  74 0B                     JE      C80
F285  80 3E 0049 R 0A           CMP     CRT_MODE,0AH    ; TEST FOR MEDIUM RES
F28A  74 04                     JE      C80
                        ;------ LOW RES UP
F28C  D0 E2                     SAL     DL,1            ; # COLUMNS * 2 AGAIN, SINCE 4
                                                        ; BYTES/CHAR
F28E  D1 E7                     SAL     DI,1            ; OFFSET *2 AGAIN, SINCE 4
                                                        ; BYTES/CHAR
```

```
                        ;------ DETERMINE THE SOURCE ADDRESS IN THE BUFFER
F290                    C80:                            ; FIND_SOURCE
F290  06                        PUSH    ES              ; GET SEGMENTS BOTH POINTING TO
                                                        ; REGEN
F291  1F                        POP     DS
F292  2A ED                     SUB     CH,CH           ; ZERO TO HIGH OF COUNT REG
F294  D0 E3                     SAL     BL,1            ; MULTIPLY NUMBER OF LINES BY 4
F296  D0 E3                     SAL     BL,1
F298  74 67                     JZ      C86             ; IF ZERO, THEN BLANK ENTIRE FIELD
F29A  8A C3                     MOV     AL,BL           ; GET NUMBER OF LINES IN AL
F29C  B4 50                     MOV     AH,80           ; 80 BYTES/ROW
F29E  F6 E4                     MUL     AH              ; DETERMINE OFFSET TO SOURCE
F2A0  8B F7                     MOV     SI,DI           ; SET UP SOURCE
F2A2  03 F0                     ADD     SI,AX           ;  ADD IN OFFSET TO IT
F2A4  8A E6                     MOV     AH,DH           ; NUMBER OF ROWS IN FIELD
F2A6  2A E3                     SUB     AH,BL           ; DETERMINE NUMBER TO MOVE
                        ;------ LOOP THROUGH, MOVING ONE ROW AT A TIME, BOTH EVEN AND ODD
                                                        ; FIELDS
F2A8                    C81:                            ; ROW_LOOP
F2A8  E8 F3C7 R                 CALL    C95             ; MOVE ONE ROW
F2AB  1E                        PUSH    DS              ; SAVE DATA SEG
F2AC  E8 138B R                 CALL    DDS             ; POINT TO BIOS DATA AREA
F2AF  80 3E 0049 R 09           CMP     CRT_MODE,9      ; MODE USES 32K REGEN?
F2B4  1F                        POP     DS              ; RESTORE DATA SEG
F2B5  72 15                     JC      C82             ; NO, JUMP
F2B7  81 C6 2000                ADD     SI,2000H        ; ADJUST POINTERS
F2BB  81 C7 2000                ADD     DI,2000H
F2BF  E8 F3C7 R                 CALL    C95             ; MOVE 2 MORE ROWS
F2C2  81 EE 3FB0                SUB     SI,4000H-80     ; BACK UP POINTERS
F2C6  81 EF 3FB0                SUB     DI,4000H-80     ;
F2CA  FE CC                     DEC     AH              ; ADJUST COUNT
F2CC  81 EE 1FB0        C82:    SUB     SI,2000H-80     ; MOVE TO NEXT ROW
F2D0  81 EF 1FB0                SUB     DI,2000H-80
F2D4  FE CC                     DEC     AH              ; NUMBER OF ROWS TO MOVE
F2D6  75 D0                     JNZ     C81             ; CONTINUE TILL ALL MOVED
                        ;------ FILL IN THE VACATED LINE(S)
F2D8                    C83:                            ; CLEAR_ENTRY
F2D8  8A C7                     MOV     AL,BH           ; ATTRIBUTE TO FILL WITH
F2DA  E8 F3E0 R         C84:    CALL    C96             ; CLEAR THAT ROW
F2DD  1E                        PUSH    DS              ; SAVE DATA SEG
F2DE  E8 138B R                 CALL    DDS             ; POINT TO BIOS DATA AREA
F2E1  80 3E 0049 R 09           CMP     CRT_MODE,9      ; MODE USES 32K REGEN?
F2E6  1F                        POP     DS              ; RESTORE DATA SEG
F2E7  72 0D                     JC      C85             ; NO, JUMP
F2E9  81 C7 2000                ADD     DI,2000H
F2ED  E8 F3E0 R                 CALL    C96             ; CLEAR 2 MORE ROWS
F2F0  81 EF 3FB0                SUB     DI,4000H-80     ; BACK UP POINTERS
F2F4  FE CB                     DEC     BL              ; ADJUST COUNT
F2F6  81 EF 1FB0        C85:    SUB     DI,2000H-80     ; POINT TO NEXT LINE
F2FA  FE CB                     DEC     BL              ; NUMBER OF LINES TO FILL
F2FC  75 DC                     JNZ     C84             ; CLEAR_LOOP
F2FE  E9 0F70 R                 JMP     VIDEO_RETURN    ; EVERYTHING DONE
F301                    C86:                            ; BLANK_FIELD
F301  8A DE                     MOV     BL,DH           ; SET BLANK COUNT TO EVERYTHING IN
                                                        ; FIELD
F303  EB D3                     JMP     C83             ; CLEAR THE FIELD
F305                    GRAPHICS_UP     ENDP
                        ;--------------------------------------------------------
                        ; SCROLL DOWN
                        ;  THIS ROUTINE SCROLLS DOWN THE INFORMATION ON THE CRT
                        ; ENTRY --
                        ;  CH,CL = UPPER LEFT CORNER OF REGION TO SCROLL
                        ;  DH,DL = LOWER RIGHT CORNER OF REGION TO SCROLL
                        ;   BOTH OF THE ABOVE ARE IN CHARACTER POSITIONS
                        ;  BH = FILL VALUE FOR BLANKED LINES
                        ;  AL = # LINES TO SCROLL (AL=0 MEANS BLANK THE ENTIRE FIELD)
                        ;  DS = DATA SEGMENT
                        ;  ES = REGEN SEGMENT
                        ; EXIT --
                        ;  NOTHING, THE SCREEN IS SCROLLED
                        ;--------------------------------------------------------
F305                    GRAPHICS_DOWN   PROC    NEAR
F305  FD                        STD             ; SET DIRECTION
F306  8A D8                     MOV     BL,AL   ; SAVE LINE COUNT IN BL
F308  8B C2                     MOV     AX,DX   ; GET LOWER RIGHT POSITION INTO AX REG
                        ;------ USE CHARACTER SUBROUTINE FOR POSITIONING
                        ;------ ADDRESS RETURNED IS MULTIPLIED BY 2 FROM CORRECT VALUE
F30A  E8 F72C R                 CALL    GRAPH_POSN
F30D  8B F8                     MOV     DI,AX           ; SAVE RESULT AS DESTINATION
                                                        ; ADDRESS
                        ;------ DETERMINE SIZE OF WINDOW
F30F  2B D1                     SUB     DX,CX
F311  81 C2 0101                ADD     DX,101H         ; ADJUST VALUES
F315  D0 E6                     SAL     DH,1            ; MULTIPLY # ROWS BY 4 SINCE 8 VERT
                                                        ; DOTS/CHAR
F317  D0 E6                     SAL     DH,1            ;  AND EVEN/ODD ROWS
                        ;------ DETERMINE CRT MODE
F319  80 3E 0049 R 06           CMP     CRT_MODE,6      ; TEST FOR HIGH RES
F31E  74 22                     JZ      C87             ; FIND_SOURCE_DOWN
```

```
                        ;------ MEDIUM RES DOWN
F320  D0 E2                     SAL     DL,1            ; # COLUMNS * 2, SINCE 2 BYTES/CHAR
                                                        ; (OFFSET OK)
F322  D1 E7                     SAL     DI,1            ; OFFSET #2 SINCE 2 BYTES/CHAR
F324  47                        INC     DI              ; POINT TO LAST BYTE
F325  80 3E 0049 R 04           CMP     CRT_MODE,4      ; TEST FOR MEDIUM RES
F32A  74 16                     JZ      C87             ; FIND_SOURCE_DOWN
F32C  80 3E 0049 R 05           CMP     CRT_MODE,5      ; TEST FOR MEDIUM RES
F331  74 0F                     JZ      C87             ; FIND_SOURCE_DOWN
F333  80 3E 0049 R 0A           CMP     CRT_MODE,0AH    ; TEST FOR MEDIUM RES
F338  74 08                     JZ      C87             ; FIND_SOURCE_DOWN
F33A  4F                        DEC     DI
F33B  D0 E2                     SAL     DL,1            ; # COLUMNS * 2 AGAIN, SINCE 4
                                                        ; BYTES/CHAR (OFFSET OK)
F33D  D1 E7                     SAL     DI,1            ; OFFSET #2 AGAIN, SINCE 4
                                                        ; BYTES/CHAR
F33F  83 C7 03                  ADD     DI,3            ; POINT TO LAST BYTE
                        ;------ DETERMINE THE SOURCE ADDRESS IN THE BUFFER
F342                    C87:                            ; FIND_SOURCE_DOWN
F342  2A ED                     SUB     CH,CH           ; ZERO TO HIGH OF COUNT REG
F344  B8 00F0                   MOV     AX,240          ; OFFSET TO LAST ROW OF PIXELS IF
                                                        ; 16K REGEN
F347  80 3E 0049 R 09           CMP     CRT_MODE,9      ; USING 32K REGEN?
F34C  72 03                     JC      C88             ; NO, JUMP
F34E  B8 00A0                   MOV     AX,160          ; OFFSET TO LAST ROW OF PIXELS IF
                                                        ; 32K REGEN
F351  03 F8                     ADD     DI,AX           ; POINT TO LAST ROW OF PIXELS
F353  D0 E3                     SAL     BL,1            ; MULTIPLY NUMBER OF LINES BY 4
F355  D0 E3                     SAL     BL,1
F357  74 6A                     JZ      C94             ; IF ZERO, THEN BLANK ENTIRE FIELD
F359  8A C3                     MOV     AL,BL           ; GET NUMBER OF LINES IN AL
F35B  B4 50                     MOV     AH,80           ; 80 BYTES/ROW
F35D  F6 E4                     MUL     AH              ; DETERMINE OFFSET TO SOURCE
F35F  8B F7                     MOV     SI,DI           ; SET UP SOURCE
F361  2B F0                     SUB     SI,AX           ;  SUBTRACT THE OFFSET
F363  8A E6                     MOV     AH,DH           ; NUMBER OF ROWS IN FIELD
F365  2A E3                     SUB     AH,BL           ; DETERMINE NUMBER TO MOVE
F367  06                        PUSH    ES              ; BOTH SEGMENTS TO REGEN
F368  1F                        POP     DS
                        ;------ LOOP THROUGH, MOVING ONE ROW AT A TIME, BOTH EVEN AND ODD
                        ;       FIELDS
F369                    C89:                            ; ROW_LOOP_DOWN
F369  E8 F3C7 R                 CALL    C95             ; MOVE ONE ROW
F36C  1E                        PUSH    DS              ; SAVE DATA SEG
F36D  E8 138B R                 CALL    DDS             ; POINT TO BIOS DATA AREA
F370  80 3E 0049 R 09           CMP     CRT_MODE,9      ; MODE USES 32K REGEN?
F375  1F                        POP     DS              ; RESTORE DATA SEG
F376  72 15                     JC      C90             ; NO, JUMP
F378  81 C6 2000                ADD     SI,2000H        ; ADJUST POINTERS
F37C  81 C7 2000                ADD     DI,2000H
F380  E8 F3C7 R                 CALL    C95             ; MOVE 2 MORE ROWS
F383  81 EE 4050                SUB     SI,4000H+80     ; BACK UP POINTERS
F387  81 EF 4050                SUB     DI,4000H+80     ;
F38B  FE CC                     DEC     AH              ; ADJUST COUNT
F38D  81 EE 2050        C90:    SUB     SI,2000H+80     ; MOVE TO NEXT ROW
F391  81 EF 2050                SUB     DI,2000H+80
F395  FE CC                     DEC     AH              ; NUMBER OF ROWS TO MOVE
F397  75 D0                     JNZ     C89             ; CONTINUE TILL ALL MOVED
                        ;------ FILL IN THE VACATED LINE(S)
F399                    C91:                            ; CLEAR_ENTRY_DOWN
F399  8A C7                     MOV     AL,BH           ; ATTRIBUTE TO FILL WITH
F39B                    C92:                            ; CLEAR_LOOP_DOWN
F39B  E8 F3E0 R                 CALL    C96             ; CLEAR A ROW
F39E  1E                        PUSH    DS              ; SAVE DATA SEG
F39F  E8 138B R                 CALL    DDS             ; POINT TO BIOS DATA AREA
F3A2  80 3E 0049 R 09           CMP     CRT_MODE,9      ; MODE USES 32K REGEN?
F3A7  1F                        POP     DS              ; RESTORE DATA SEG
F3A8  72 0D                     JC      C93             ; NO, JUMP
F3AA  81 C7 2000                ADD     DI,2000H
F3AE  E8 F3E0 R                 CALL    C96             ; CLEAR 2 MORE ROWS
F3B1  81 EF 4050                SUB     DI,4000H+80     ; BACK UP POINTERS
F3B5  FE CB                     DEC     BL              ; ADJUST COUNT
F3B7  81 EF 2050        C93:    SUB     DI,2000H+80     ; POINT TO NEXT LINE
F3BB  FE CB                     DEC     BL              ; NUMBER OF LINES TO FILL
F3BD  75 DC                     JNZ     C92             ; CLEAR_LOOP_DOWN
F3BF  FC                        CLD                     ; RESET THE DIRECTION FLAG
F3C0  E9 0F70 R                 JMP     VIDEO_RETURN    ; EVERYTHING DONE
F3C3                    C94:                            ; BLANK_FIELD_DOWN
F3C3  8A DE                     MOV     BL,DH           ; SET BLANK COUNT TO EVERYTHING IN
                                                        ; FIELD
F3C5  EB D2                     JMP     C91             ; CLEAR THE FIELD
F3C7                    GRAPHICS_DOWN   ENDP
                        ;------ ROUTINE TO MOVE ONE ROW OF INFORMATION
F3C7                    C95     PROC    NEAR
F3C7  8A CA                     MOV     CL,DL           ; NUMBER OF BYTES IN THE ROW
F3C9  56                        PUSH    SI
F3CA  57                        PUSH    DI              ; SAVE POINTERS
F3CB  F3/ A4                    REP     MOVSB           ; MOVE THE EVEN FIELD
F3CD  5F                        POP     DI
F3CE  5E                        POP     SI
F3CF  81 C6 2000                ADD     SI,2000H
F3D3  81 C7 2000                ADD     DI,2000H        ; POINT TO THE ODD FIELD
F3D7  56                        PUSH    SI
F3D8  57                        PUSH    DI              ; SAVE THE POINTERS
F3D9  8A CA                     MOV     CL,DL           ; COUNT BACK
F3DB  F3/ A4                    REP     MOVSB           ; MOVE THE ODD FIELD
F3DD  5F                        POP     DI
F3DE  5E                        POP     SI              ; POINTERS BACK
F3DF  C3                        RET                     ; RETURN TO CALLER
F3E0                    C95     ENDP
```

```
                        ;------ CLEAR A SINGLE ROW
F3E0                    C96     PROC    NEAR
F3E0  8A CA                     MOV     CL,DL           ; NUMBER OF BYTES IN FIELD
F3E2  57                        PUSH    DI              ; SAVE POINTER
F3E3  F3/ AA                    REP     STOSB           ; STORE THE NEW VALUE
F3E5  5F                        POP     DI              ; POINTER BACK
F3E6  81 C7 2000                ADD     DI,2000H        ; POINT TO ODD FIELD
F3EA  57                        PUSH    DI
F3EB  8A CA                     MOV     CL,DL
F3ED  F3/ AA                    REP     STOSB           ; FILL THE ODD FILELD
F3EF  5F                        POP     DI
F3F0  C3                        RET                     ; RETURN TO CALLER
F3F1                    C96     ENDP
                        ;--------------------------------------------------
                        ; GRAPHICS WRITE
                        ;  THIS ROUTINE WRITES THE ASCII CHARACTER TO THE CURRENT
                        ;  POSITION ON THE SCREEN.
                        ; ENTRY --
                        ;  AL = CHARACTER TO WRITE
                        ;  BL = COLOR ATTRIBUTE TO BE USED FOR FOREGROUND COLOR
                        ;       IF BIT 7 IS SET, THE CHAR IS XOR'D INTO THE REGEN BUFFER
                        ;       (0 IS USED FOR THE BACKGROUND COLOR)
                        ;  CX = NUMBER OF CHARS TO WRITE
                        ;  DS = DATA SEGMENT
                        ;  ES = REGEN SEGMENT
                        ; EXIT --
                        ;  NOTHING IS RETURNED
                        ;
                        ; GRAPHICS READ
                        ;   THIS ROUTINE READS THE ASCII CHARACTER AT THE CURRENT CURSOR
                        ;   POSITION ON THE SCREEN BY MATCHING THE DOTS ON THE SCREEN TO
                        ;   THE CHARACTER GENERATOR CODE POINTS
                        ; ENTRY --
                        ;  NONE  (0 IS ASSUMED AS THE BACKGROUND COLOR)
                        ; EXIT --
                        ;  AL = CHARACTER READ AT THAT POSITION (0 RETURNED IF NONE FOUND)
                        ;
                        ; FOR BOTH ROUTINES, THE IMAGES USED TO FORM CHARS ARE CONTAINED IN
                        ; ROM.  INTERRUPT 44H IS USED TO POINT TO THE TABLE FOR THE FIRST
                        ; 128 CHARS.  INTERRUPT 17H IS USED TO POINT TO THE TABLE FOR THE
                        ; SECOND 128 CHARS.
                        ;--------------------------------------------------
                                ASSUME  CS:CODE,DS:DATA,ES:DATA
F3F1                    GRAPHICS_WRITE  PROC    NEAR
F3F1  32 E4                     XOR     AH,AH           ; ZERO TO HIGH OF CODE POINT
F3F3  50                        PUSH    AX              ; SAVE CODE POINT VALUE
                        ;------ DETERMINE POSITION IN REGEN BUFFER TO PUT CODE POINTS
F3F4  E8 F729 R                 CALL    R59             ; FIND LOCATION IN REGEN BUFFER
F3F7  8B F8                     MOV     DI,AX           ; REGEN POINTER IN DI
                        ;------ DETERMINE REGION TO GET CODE POINTS FROM
F3F9  58                        POP     AX              ; RECOVER CODE POINT
F3FA  BE 0110 R                 MOV     SI,OFFSET CSET_PTR ; ASSUME FIRST HALF
F3FD  3C 80                     CMP     AL,80H          ; IS IT IN FIRST HALF?
F3FF  72 05                     JB      R1              ; JUMP IF IT IS
F401  BE 007C R                 MOV     SI,OFFSET EXT_PTR ; SET POINTER FOR SECOND HALF
F404  2C 80                     SUB     AL,80H          ; ZERO ORIGIN FOR SECOND HALF
F406                    R1:                             ; EXTEND_CHAR
F406  1E                        PUSH    DS              ; SAVE DATA POINTER
F407  33 D2                     XOR     DX,DX
F409  8E DA                     MOV     DS,DX           ; ESTABLISH VECTOR ADDRESSING
                                ASSUME  DS:ABS0
F40B  C5 34                     LDS     SI,DWORD PTR [SI] ; GET THE OFFSET OF THE TABLE
F40D  8C DA                     MOV     DX,DS           ; GET THE SEGMENT OF THE TABLE
                                ASSUME  DS:DATA
F40F  1F                        POP     DS              ; RECOVER DATA SEGMENT
F410  52                        PUSH    DX              ; SAVE TABLE SEGMENT ON STACK
                        ;------ DETERMINE GRAPHICS MODE IN OPERATION
F411  D1 E0                     SAL     AX,1            ; MULTIPLY CODE POINT
F413  D1 E0                     SAL     AX,1            ;  VALUE BY 8
F415  D1 E0                     SAL     AX,1
F417  03 F0                     ADD     SI,AX           ; SI HAS OFFSET OF DESIRED CODES
F419  80 3E 0049 R 04           CMP     CRT_MODE,4
F41E  74 45                     JE      R9              ; TEST FOR MEDIUM RESOLUTION MODE
F420  80 3E 0049 R 05           CMP     CRT_MODE,5
F425  74 3E                     JE      R9              ; TEST FOR MEDIUM RESOLUTION MODE
F427  80 3E 0049 R 0A           CMP     CRT_MODE,0AH
F42C  75 03                     JNE     R3              ; TEST FOR MEDIUM RESOLUTION MODE
F42E  E9 F4D4 R                 JMP     R16
F431  80 3E 0049 R 06   R3:     CMP     CRT_MODE,6      ; TEST FOR HIGH RESOLUTION MODE
F436  75 53                     JNE     R12             ; GOTO LOW RESOLUTION IF NOT
                        ;------ HIGH RESOLUTION MODE
F438  1F                        POP     DS              ; RECOVER TABLE POINTER SEGMENT
F439  57                R5:     PUSH    DI              ; SAVE REGEN POINTER
F43A  56                        PUSH    SI              ; SAVE CODE POINTER
F43B  B6 04                     MOV     DH,4            ; NUMBER OF TIMES THROUGH LOOP
F43D  AC                R6:     LODSB                   ; GET BYTE FROM CODE POINTS
F43E  F6 C3 80                  TEST    BL,80H          ; SHOULD WE USE THE FUNCTION
F441  75 16                     JNZ     R8              ;  TO PUT CHAR IN?
F443  AA                        STOSB                   ; STORE IN REGEN BUFFER
F444  AC                        LODSB
F445  26: 88 85 1FFF    R7:     MOV     ES:[DI+2000H-1],AL ; STORE IN SECOND HALF
F44A  83 C7 4F                  ADD     DI,79           ; MOVE TO NEXT ROW IN REGEN
F44D  FE CE                     DEC     DH              ; DONE WITH LOOP
F44F  75 EC                     JNZ     R6
F451  5E                        POP     SI
F452  5F                        POP     DI              ; RECOVER REGEN POINTER
F453  47                        INC     DI              ; POINT TO NEXT CHAR POSITION
F454  E2 E3                     LOOP    R5              ; MORE CHARS TO WRITE
```

```
F456  E9 0F70 R                   R705:   JMP     VIDEO_RETURN
F459  26: 32 05                   R8:     XOR     AL,ES:[DI]          ; EXCLUSIVE OR WITH CURRENT DATA
F45C  AA                                  STOSB                       ; STORE THE CODE POINT
F45D  AC                                  LODSB                       ; AGAIN FOR ODD FIELD
F45E  26: 32 85 1FFF                      XOR     AL,ES:[DI+2000H-1]
F463  EB E0                               JMP     R7                  ; BACK TO MAINSTREAM
                                  ;------ MEDIUM RESOLUTION WRITE
F465                              R9:                                 ; MED_RES_WRITE
F465  1F                                  POP     DS                  ; RECOVER TABLE POINTER SEGMENT
F466  8A D3                               MOV     DL,BL               ; SAVE HIGH COLOR BIT
F468  D1 E7                               SAL     DI,1                ; OFFSET*2 SINCE 2 BYTES/CHAR
F46A  E8 F659 R                           CALL    R40                 ; EXPAND BL TO FULL WORD OF COLOR
F46D                              R10:                                ; MED_CHAR
F46D  57                                  PUSH    DI                  ; SAVE REGEN POINTER
F46E  56                                  PUSH    SI                  ; SAVE THE CODE POINTER
F46F  B6 04                               MOV     DH,4                ; NUMBER OF LOOPS
F471  E8 F626 R                   R11:    CALL    R35                 ; DO FIRST 2 BYTES
F474  81 C7 2000                          ADD     DI,2000H            ; NEXT SPOT IN REGEN
F478  E8 F626 R                           CALL    R35                 ; DO NEXT 2 BYTES
F47B  81 EF 1FB0                          SUB     DI,2000H-80
F47F  FE CE                               DEC     DH
F481  75 EE                               JNZ     R11                 ; KEEP GOING
F483  5E                                  POP     SI                  ; RECOVER CODE PONTER
F484  5F                                  POP     DI                  ; RECOVER REGEN POINTER
F485  47                                  INC     DI                  ; POINT TO NEXT CHAR POSITION
F486  47                                  INC     DI
F487  E2 E4                               LOOP    R10                 ; MORE TO WRITE
F489  EB CB                               JMP     R705
                                  ;------ LOW RESOLUTION WRITE
F48B                              R12:                                ; LOW_RES_WRITE
F48B  1F                                  POP     DS                  ; RECOVER TABLE POINTER SEGMENT
F48C  8A D3                               MOV     DL,BL               ; SAVE HIGH COLOR BIT
F48E  D1 E7                               SAL     DI,1                ; OFFSET*4 SINCE 4 BYTES/CHAR
F490  D1 E7                               SAL     DI,1                ;
F492  E8 F66E R                           CALL    R42                 ; EXPAND BL TO FULL WORD OF COLOR
F495                              R13:                                ; MED_CHAR
F495  57                                  PUSH    DI                  ; SAVE REGEN POINTER
F496  56                                  PUSH    SI                  ; SAVE THE CODE POINTER
F497  B6 04                               MOV     DH,4                ; NUMBER OF LOOPS
F499  E8 F645 R                   R14:    CALL    R39                 ; EXPAND DOT ROW IN REGEN
F49C  81 C7 2000                          ADD     DI,2000H            ; POINT TO NEXT REGEN ROW
F4A0  E8 F645 R                           CALL    R39                 ; EXPAND DOT ROW IN REGEN
F4A3  1E                                  PUSH    DS                  ; SAVE DS
F4A4  E8 138B R                           CALL    DDS                 ; POINT TO BIOS DATA AREA
F4A7  80 3E 0049 R 09                     CMP     CRT_MODE,09H        ; USING 32K REGEN AREA?
F4AC  1F                                  POP     DS                  ; RECOVER DS
F4AD  75 14                               JNE     R15                 ; JUMP IF 16K REGEN
F4AF  81 C7 2000                          ADD     DI,2000H            ; POINT TO NEXT REGEN ROW
F4B3  E8 F645 R                           CALL    R39                 ; EXPAND DOT ROW IN REGEN
F4B6  81 C7 2000                          ADD     DI,2000H            ; POINT TO NEXT REGEN ROW
F4BA  E8 F645 R                           CALL    R39                 ; EXPAND DOT ROW IN REGEN
F4BD  81 EF 3FB0                          SUB     DI,4000H-80         ; ADJUST REGEN POINTER
F4C1  FE CE                               DEC     DH
F4C3  81 EF 1FB0                  R15:    SUB     DI,2000H-80         ; ADJUST REGEN POINTER TO NEXT ROW
F4C7  FE CE                               DEC     DH
F4C9  75 CE                               JNZ     R14                 ; KEEP GOING
F4CB  5E                                  POP     SI                  ; RECOVER CODE PONTER
F4CC  5F                                  POP     DI                  ; RECOVER REGEN POINTER
F4CD  83 C7 04                            ADD     DI,4                ; POINT TO NEXT CHAR POSITION
F4D0  E2 C3                               LOOP    R13                 ; MORE TO WRITE
F4D2  EB 82                               JMP     R705
                                  ;----- 640X200 4 COLOR GRAPHICS WRITE
F4D4  1F                          R16:    POP     DS                  ; RECOVER TABLE SEGMENT POINTER
F4D5  8A D3                               MOV     DL,BL               ; SAVE HIGH COLOR BIT
F4D7  D1 E7                               SAL     DI,1                ; OFFSET*2 SINCE 2 BYTES/CHAR
                                  ; EXPAND LOW 2 COLOR BITS IN BL (c1c0)
                                  ; INTO BX (c0c0c0c0c0c0c0c0c1c1c1c1c1c1c1c1)
F4D9  33 C0                               XOR     AX,AX
F4DB  F6 C3 01                            TEST    BL,1                ; c0 COLOR BIT ON?
F4DE  74 02                               JZ      R17                 ; NO, JUMP
F4E0  B4 FF                               MOV     AH,0FFH             ; YES, SET ALL c0 BITS ON
F4E2  F6 C3 02                    R17:    TEST    BL,2                ; c1 COLOR BIT ON?
F4E5  74 02                               JZ      R18                 ; NO, JUMP
F4E7  B0 FF                               MOV     AL,0FFH             ; YES, SET ALL c1 BITS ON
F4E9  8B D8                       R18:    MOV     BX,AX               ; COLOR MASK IN BX
F4EB                              R19:
F4EB  57                                  PUSH    DI                  ; SAVE REGEN POINTER
F4EC  56                                  PUSH    SI                  ; SAVE CODE POINT POINTER
F4ED  B6 02                               MOV     DH,2                ; SET LOOP COUNTER
F4EF  E8 F518 R                   R20:    CALL    R21                 ; DO FIRST DOT ROW
F4F2  81 C7 2000                          ADD     DI,2000H            ; ADJUST REGEN POINTER
F4F6  E8 F518 R                           CALL    R21                 ; DO NEXT DOT ROW
F4F9  81 C7 2000                          ADD     DI,2000H            ; ADJUST REGEN POINTER
F4FD  E8 F518 R                           CALL    R21                 ; DO NEXT DOT ROW
F500  81 C7 2000                          ADD     DI,2000H            ; ADJUST REGEN POINTER
F504  E8 F518 R                           CALL    R21                 ; DO NEXT DOT ROW
F507  81 EF 5F60                          SUB     DI,6000H-160        ; ADJUST REGEN POINTER TO NEXT ROW
F50B  FE CE                               DEC     DH
F50D  75 E0                               JNZ     R20                 ; KEEP GOING
F50F  5E                                  POP     SI                  ; RECOVER CODE POINT POINTER
F510  5F                                  POP     DI                  ; RECOVER REGEN POINTER
F511  47                                  INC     DI                  ; POINT TO NEXT CHARACTER
F512  47                                  INC     DI
F513  E2 D6                               LOOP    R19                 ; MORE TO WRITE
F515  E9 0F70 R                           JMP     VIDEO_RETURN
```

```
F518                            R21     PROC    NEAR
F518  AC                                LODSB                   ; GET CODE POINT
F519  8A E0                             MOV     AH,AL           ; COPY INTO AH
F51B  23 C3                             AND     AX,BX           ; SET COLOR
F51D  F6 C2 80                          TEST    DL,80H          ; XOR FUNCTION?
F520  74 07                             JZ      R22             ; NO, JUMP
F522  26: 32 25                         XOR     AH,ES:[DI]      ; EXCLUSIVE OR WITH CURRENT DATA
F525  26: 32 45 01                      XOR     AL,ES:[DI+1]
F529  26: 88 25                 R22:    MOV     ES:[DI],AH      ; STORE IN REGEN BUFFER
F52C  26: 88 45 01                      MOV     ES:[DI+1],AL
F530  C3                                RET
F531                            R21     ENDP
F531                            GRAPHICS_WRITE  ENDP
                                ;--------------------------------------
                                ; GRAPHICS READ
                                ;--------------------------------------
F531                            GRAPHICS_READ   PROC    NEAR
F531  E8 F729 R                         CALL    R59             ; CONVERTED TO OFFSET IN REGEN
F534  8B F0                             MOV     SI,AX           ; SAVE IN SI
F536  83 EC 08                          SUB     SP,8            ; ALLOCATE SPACE TO SAVE THE READ
                                                                ; CODE POINT
F539  8B EC                             MOV     BP,SP           ; POINTER TO SAVE AREA
                                ;------ DETERMINE GRAPHICS MODES
F53B  06                                PUSH    ES
F53C  B6 04                             MOV     DH,4            ; NUMBER OF PASSES
F53E  80 3E 0049 R 06                   CMP     CRT_MODE,6
F543  74 17                             JZ      R23             ; HIGH RESOLUTION
F545  80 3E 0049 R 04                   CMP     CRT_MODE,4
F54A  74 61                             JZ      R28             ; MEDIUM RESOLUTION
F54C  80 3E 0049 R 05                   CMP     CRT_MODE,5
F551  74 5A                             JZ      R28             ; MEDIUM RESOLUTION
F553  80 3E 0049 R 0A                   CMP     CRT_MODE,0AH
F558  74 53                             JZ      R28             ; MEDIUM RESOLUTION
F55A  EB 18                             JMP     SHORT R25       ; LOW RESOLUTION
                                ;------ HIGH RESOLUTION READ
                                ;------ GET VALUES FROM REGEN BUFFER AND CONVERT TO CODE POINT
F55C  1F                        R23:    POP     DS              ; POINT TO REGEN SEGMENT
F55D  8A 04                     R24:    MOV     AL,[SI]         ; GET FIRST BYTE
F55F  88 46 00                          MOV     [BP],AL         ; SAVE IN STORAGE AREA
F562  45                                INC     BP              ; NEXT LOCATION
F563  8A 84 2000                        MOV     AL,[SI+2000H]   ; GET LOWER REGION BYTE
F567  88 46 00                          MOV     [BP],AL         ; ADJUST AND STORE
F56A  45                                INC     BP
F56B  83 C6 50                          ADD     SI,80           ; POINTER INTO REGEN
F56E  FE CE                             DEC     DH              ; LOOP CONTROL
F570  75 EB                             JNZ     R24             ; DO IT SOME MORE
F572  EB 6E                             JMP     SHORT R31       ; GO MATCH THE SAVED CODE POINTS
                                ;------ LOW RESOLUTION READ
F574  1F                        R25:    POP     DS              ; POINT TO REGEN SEGMENT
F575  D1 E6                             SAL     SI,1            ; OFFSET*4 SINCE 4 BYTES/CHAR
F577  D1 E6                             SAL     SI,1
F579  E8 F6FC R                 R26:    CALL    R55             ; GET 4 BYTES FROM REGEN INTO
                                                                ; SINGLE SAVE
F57C  81 C6 2000                        ADD     SI,2000H        ; GOTO LOWER REGION
F580  E8 F6FC R                         CALL    R55             ; GET 4 BYTES FROM REGEN INTO
                                                                ; SINGLE SAVE
F583  1E                                PUSH    DS              ; SAVE DS
F584  E8 138B R                         CALL    DDS             ; POINT TO BIOS DATA AREA
F587  80 3E 0049 R 09                   CMP     CRT_MODE,9      ; DO WE HAVE A 32K REGEN AREA?
F58C  1F                                POP     DS
F58D  75 14                             JNE     R27             ; NO, JUMP
F58F  81 C6 2000                        ADD     SI,2000H        ; GOTO LOWER REGION
F593  E8 F6FC R                         CALL    R55             ; GET 4 BYTES FROM REGEN INTO
                                                                ; SINGLE SAVE
F596  81 C6 2000                        ADD     SI,2000H        ; GOTO LOWER REGION
F59A  E8 F6FC R                         CALL    R55             ; GET 4 BYTES FROM REGEN INTO
                                                                ; SINGLE SAVE
F59D  81 EE 3FB0                        SUB     SI,4000H-80     ; ADJUST POINTER
F5A1  FE CE                             DEC     DH
F5A3  81 EE 1FB0                R27:    SUB     SI,2000H-80     ; ADJUST POINTER BACK TO UPPER
F5A7  FE CE                             DEC DH
F5A9  75 CE                             JNZ     R26             ;DO IT SOME MORE
F5AB  EB 35                             JMP     SHORT R31       ; GO MATCH THE SAVED CODE POINTS
                                ;------ MEDIUM RESOLUTION READ
F5AD                            R28:                            ; MED_RES_READ
F5AD  1F                                POP     DS              ; POINT TO REGEN SEGMENT
F5AE  D1 E6                             SAL     SI,1            ; OFFSET*2 SINCE 2 BYTES/CHAR
F5B0  E8 F6C3 R                 R29:    CALL    R50             ; GET PAIR BYTES FROM REGEN INTO
                                                                ; SINGLE SAVE
F5B3  81 C6 2000                        ADD     SI,2000H        ; GO TO LOWER REGION
F5B7  E8 F6C3 R                         CALL    R50             ; GET THIS PAIR INTO SAVE
F5BA  1E                                PUSH    DS              ; SAVE DS
F5BB  E8 138B R                         CALL    DDS             ; POINT TO BIOS DATA AREA
F5BE  80 3E 0049 R 0A                   CMP     CRT_MODE,0AH    ; DO WE HAVE A 32K REGEN AREA?
F5C3  1F                                POP     DS
F5C4  75 14                             JNE     R30             ; NO, JUMP
F5C6  81 C6 2000                        ADD     SI,2000H        ; GOTO LOWER REGION
F5CA  E8 F6C3 R                         CALL    R50             ; GET PAIR BYTES FROM REGEN INTO
                                                                ; SINGLE SAVE
F5CD  81 C6 2000                        ADD     SI,2000H        ; GOTO LOWER REGION
F5D1  E8 F6C3 R                         CALL    R50             ; GET PAIR BYTES FROM REGEN INTO
                                                                ; SINGLE SAVE
F5D4  81 EE 3FB0                        SUB     SI,4000H-80     ; ADJUST POINTER
F5D8  FE CE                             DEC     DH
F5DA                            R30:
F5DA  81 EE 1FB0                        SUB     SI,2000H-80     ; ADJUST POINTER BACK INTO UPPER
F5DE  FE CE                             DEC     DH
F5E0  75 CE                             JNZ     R29             ; KEEP GOING UNTIL ALL 8 DONE
```

```
                                ;-------- SAVE AREA HAS CHARACTER IN IT, MATCH IT
F5E2                            R31:                            ; FIND_CHAR
F5E2  33 C0                             XOR     AX,AX
F5E4  8E D8                             MOV     DS,AX           ; ESTABLISH ADDRESSING TO VECTOR
                                        ASSUME  DS:ABS0
F5E6  C4 3E 0110 R                      LES     DI,CSET_PTR     ; GET POINTER TO FIRST HALF
F5EA  83 ED 08                          SUB     BP,8            ; ADJUST POINTER TO BEGINNING OF
                                                                ; SAVE AREA
F5ED  8B F5                             MOV     SI,BP
F5EF  FC                                CLD                     ; ENSURE DIRECTION
F5F0  32 C0                             XOR     AL,AL           ; CURRENT CODE POINT BEING MATCHED
F5F2  16                        R32:    PUSH    SS              ; ESTABLISH ADDRESSING TO STACK
F5F3  1F                                POP     DS              ; FOR THE STRING COMPARE
F5F4  BA 0080                           MOV     DX,128          ; NUMBER TO TEST AGAINST
F5F7  56                        R33:    PUSH    SI              ; SAVE AREA POINTER
F5F8  57                                PUSH    DI              ; SAVE CODE POINTER
F5F9  B9 0008                           MOV     CX,8            ; NUMBER OF BYTES TO MATCH
F5FC  F3/ A6                            REPE    CMPSB           ; COMPARE THE 8 BYTES
F5FE  5F                                POP     DI              ; RECOVER THE POINTERS
F5FF  5E                                POP     SI
F600  74 1E                             JZ      R34             ; IF ZERO FLAG SET, THEN MATCH
                                                                ; OCCURRED
F602  FE C0                             INC     AL              ; NO MATCH, MOVE ON TO NEXT
F604  83 C7 08                          ADD     DI,8            ; NEXT CODE POINT
F607  4A                                DEC     DX              ; LOOP CONTROL
F608  75 ED                             JNZ     R33             ; DO ALL OF THEM
                                ;------ CHAR NOT MATCHED, MIGHT BE IN SECOND HALF
F60A  0A C0                             OR      AL,AL           ; AL<> 0 IF ONLY 1ST HALF SCANNED
F60C  74 12                             JE      R34             ; IF = 0, THEN ALL HAS BEEN SCANNED
F60E  2B C0                             SUB     AX,AX
F610  8E D8                             MOV     DS,AX           ; ESTABLISH ADDRESSING TO VECTOR
                                        ASSUME  DS:ABS0
F612  C4 3E 007C R                      LES     DI,EXT_PTR      ; GET POINTER
F616  8C C0                             MOV     AX,ES           ; SEE IF THE POINTER REALLY EXISTS
F618  0B C7                             OR      AX,DI           ; IF ALL 0, THEN DOESN'T EXIST
F61A  74 04                             JZ      R34             ; NO SENSE LOOKING
F61C  B0 80                             MOV     AL,128          ; ORIGIN FOR SECOND HALF
F61E  EB D2                             JMP     R32             ; GO BACK AND TRY FOR IT
                                        ASSUME  DS:DATA
                                ;------ CHARACTER IS FOUND ( AL=0 IF NOT FOUND )
F620  83 C4 08                  R34:    ADD     SP,8            ; READJUST THE STACK, THROW AWAY
                                                                ; WORK AREA
F623  E9 0F70 R                         JMP     VIDEO_RETURN    ; ALL DONE
F626                            GRAPHICS_READ   ENDP
                                ;--------
F626                            R35     PROC    NEAR
F626  AC                                LODSB                   ; GET CODE POINT
F627  E8 F67E R                         CALL    R43             ; DOUBLE UP ALL THE BITS
F62A  23 C3                     R36:    AND     AX,BX           ; CONVERT THEM TO FOREGROUND COLOR
                                                                ; ( 0 BACK )
F62C  F6 C2 80                          TEST    DL,80H          ; IS THIS XOR FUNCTION?
F62F  74 07                             JZ      R37             ; NO, STORE IT IN AS IT IS
F631  26: 32 25                         XOR     AH,ES:[DI]      ; DO FUNCTION WITH HALF
F634  26: 32 45 01                      XOR     AL,ES:[DI+1]    ;  AND WITH OTHER HALF
F638  26: 88 25                 R37:    MOV     ES:[DI],AH      ; STORE FIRST BYTE
F63B  26: 88 45 01                      MOV     ES:[DI+1],AL    ; STORE SECOND BYTE
F63F  C3                                RET
F640                            R35     ENDP
                                ;--------
F640                            R38     PROC    NEAR
F640  E8 F6A0 R                         CALL    R45             ; QUAD UP THE LOW NIBBLE
F643  EB E5                             JMP     R36
F645                            R38     ENDP
                                ;-----------
                                ; EXPAND 1 DOT ROW OF A CHAR INTO 4 BYTES IN THE REGEN BUFFER
                                ;-------------
F645                            R39     PROC    NEAR
F645  AC                                LODSB                   ; GET CODE POINT
F646  50                                PUSH    AX              ; SAVE
F647  51                                PUSH    CX
F648  B1 04                             MOV     CL,4            ; MOV HIGH NIBBLE TO LOW
F64A  D2 E8                             SHR     AL,CL
F64C  59                                POP     CX
F64D  E8 F640 R                         CALL    R38             ; EXPAND TO 2 BYTES & PUT IN REGEN
F650  58                                POP     AX              ; RECOVER CODE POINT
F651  47                                INC     DI              ; ADJUST REGEN POINTER
F652  47                                INC     DI
F653  E8 F640 R                         CALL    R38             ; EXPAND LOW NIBBLE & PUT IN REGEN
F656  4F                                DEC     DI              ; RESTORE REGEN POINTER
F657  4F                                DEC     DI
F658  C3                                RET
F659                            R39     ENDP
                                ;--------------------------------------------
                                ; EXPAND_MED_COLOR
                                ;  THIS ROUTINE EXPANDS THE LOW 2 BITS IN BL TO
                                ;  FILL THE ENTIRE BX REGISTER
                                ; ENTRY --
                                ;  BL = COLOR TO BE USED ( LOW 2 BITS )
                                ; EXIT --
                                ;  BX = COLOR TO BE USED ( 8 REPLICATIONS OF THE 2 COLOR BITS )
                                ;--------------------------------------------
```

```
F659                            ;
F659                    R40     PROC    NEAR
F659  80 E3 03                  AND     BL,3                    ; ISOLATE THE COLOR BITS
F65C  8A C3                     MOV     AL,BL                   ; COPY TO AL
F65E  51                        PUSH    CX                      ; SAVE REGISTER
F65F  B9 0003                   MOV     CX,3                    ; NUMBER OF TIMES TO DO THIS
F662  D0 E0                     SAL     AL,1
F664  D0 E0                     SAL     AL,1                    ; LEFT SHIFT BY 2
F666  0A D8                     OR      BL,AL                   ; ANOTHER COLOR VERSION INTO BL
F668  E2 F8                     LOOP    R41                     ; FILL ALL OF BL
F66A  8A FB                     MOV     BH,BL                   ; FILL UPPER PORTION
F66C  59                        POP     CX                      ; REGISTER BACK
F66D  C3                        RET                             ; ALL DONE
F66E                            ENDP
                                ;--------------------------------------------
                                ; EXPAND_LOW_COLOR
                                ;  THIS ROUTINE EXPANDS THE LOW 4 BITS IN BL TO
                                ;  FILL THE ENTIRE BX REGISTER
                                ; ENTRY --
                                ;  BL = COLOR TO BE USED ( LOW 4 BITS )
                                ; EXIT --
                                ;  BX = COLOR TO BE USED ( 4 REPLICATIONS OF THE 4 COLOR BITS )
                                ;--------------------------------------------
F66E                    R42     PROC    NEAR
F66E  51                        PUSH    CX
F66F  80 E3 0F                  AND     BL,0FH                  ; ISOLATE THE COLOR BITS
F672  8A FB                     MOV     BH,BL                   ; COPY TO BH
F674  B1 04                     MOV     CL,4                    ; MOVE TO HIGH NIBBLE
F676  D2 E7                     SHL     BH,CL                   ;
F678  0A FB                     OR      BH,BL                   ; MAKE BYTE FROM HIGH AND LOW
                                                                ; NIBBLES
F67A  8A DF                     MOV     BL,BH                   ;
F67C  59                        POP     CX
F67D  C3                        RET                             ; ALL DONE
F67E                            ENDP
                                ;--------------------------------------------
                                ; EXPAND_BYTE
                                ;  THIS ROUTINE TAKES THE BYTE IN AL AND DOUBLES ALL
                                ;  OF THE BITS, TURNING THE 8 BITS INTO 16 BITS.
                                ;  THE RESULT IS LEFT IN AX
                                ;--------------------------------------------
F67E                    R43     PROC    NEAR
F67E  52                        PUSH    DX                      ; SAVE REGISTERS
F67F  51                        PUSH    CX
F680  53                        PUSH    BX
F681  2B D2                     SUB     DX,DX                   ; RESULT REGISTER
F683  B9 0001                   MOV     CX,1                    ; MASK REGISTER
F686  8B D8                     MOV     BX,AX                   ; BASE INTO TEMP
F688  23 D9                     AND     BX,CX                   ; USE MASK TO EXTRACT A BIT
F68A  0B D3                     OR      DX,BX                   ; PUT INTO RESULT REGISTER
F68C  D1 E0                     SHL     AX,1
F68E  D1 E1                     SHL     CX,1                    ; SHIFT BASE AND MASK BY 1
F690  8B D8                     MOV     BX,AX                   ; BASE TO TEMP
F692  23 D9                     AND     BX,CX                   ; EXTRACT THE SAME BIT
F694  0B D3                     OR      DX,BX                   ; PUT INTO RESULT
F696  D1 E1                     SHL     CX,1                    ; SHIFT ONLY MASK NOW, MOVING TO
                                                                ; NEXT BASE
F698  73 EC                     JNC     R44                     ; USE MASK BIT COMING OUT TO
                                                                ; TERMINATE
F69A  8B C2                     MOV     AX,DX                   ; RESULT TO PARM REGISTER
F69C  5B                        POP     BX
F69D  59                        POP     CX                      ; RECOVER REGISTERS
F69E  5A                        POP     DX
F69F  C3                        RET                             ; ALL DONE
F6A0                            ENDP
                                ;--------------------------------------------
                                ; EXPAND_NIBBLE
                                ;  THIS ROUTINE TAKES THE LOW NIBBLE IN AL AND QUADS ALL
                                ;  OF THE BITS, TURNING THE 4 BITS INTO 16 BITS.
                                ;  THE RESULT IS LEFT IN AX
                                ;--------------------------------------------
F6A0                    R45     PROC    NEAR
F6A0  52                        PUSH    DX                      ; SAVE REGISTERS
F6A1  33 D2                     XOR     DX,DX                   ; RESULT REGISTER
F6A3  A8 08                     TEST    AL,8
F6A5  74 03                     JZ      R46
F6A7  80 CE F0                  OR      DH,0F0H
F6AA  A8 04                     TEST    AL,4
F6AC  74 03                     JZ      R47
F6AE  80 CE 0F                  OR      DH,0FH
F6B1  A8 02                     TEST    AL,2
F6B3  74 03                     JZ      R48
F6B5  80 CA F0                  OR      DL,0F0H
F6B8  A8 01                     TEST    AL,1
F6BA  74 03                     JZ      R49
F6BC  80 CA 0F                  OR      DL,0FH
F6BF  8B C2                     MOV     AX,DX                   ; RESULT TO PARM REGISTER
F6C1  5A                        POP     DX                      ; RECOVER REGISTERS
F6C2  C3                        RET                             ; ALL DONE
F6C3                            ENDP
```

```
                        ;--------------------------------------------------
                        ; MED_READ_BYTE
                        ; THIS ROUTINE WILL TAKE 2 BYTES FROM THE REGEN BUFFER,
                        ;  COMPARE AGAINST THE CURRENT FOREGROUND COLOR, AND PLACE
                        ;  THE CORRESPONDING ON/OFF BIT PATTERN INTO THE CURRENT
                        ;  POSITION IN THE SAVE AREA
                        ; ENTRY --
                        ;  SI,DS = POINTER TO REGEN AREA OF INTEREST
                        ;  BX = EXPANDED FOREGROUND COLOR
                        ;  BP = POINTER TO SAVE AREA
                        ; EXIT --
                        ;  BP IS INCREMENT AFTER SAVE
                        ;--------------------------------------------------
F6C3                    R50     PROC    NEAR
F6C3  8A 24                     MOV     AH,[SI]         ; GET FIRST BYTE
F6C5  8A 44 01                  MOV     AL,[SI+1]       ; GET SECOND BYTE
F6C8  1E                        PUSH    DS              ; SAVE DS
F6C9  E8 1388 R                 CALL    DDS             ; POINT TO BIOS DATA AREA
F6CC  80 3E 0049 R 0A           CMP     CRT_MODE,0AH    ; IN 640X200 4 COLOR MODE?
F6D1  1F                        POP     DS              ; RESTORE REGEN SEG
F6D2  75 11                     JNE     R52             ; NO,  JUMP
                        ; IN 640X200 4 COLOR MODE, ALL THE c0 BITS ARE IN ONE BYTE, AND ALL
                        ; THE c1 BITS ARE IN THE NEXT BYTE. HERE WE CHANGE THEM BACK TO
                        ; NORMAL c1c0 ADJACENT PAIRS.
F6D4  53                        PUSH    BX              ; SAVE REG
F6D5  B9 0008                   MOV     CX,8            ; SET LOOP COUNTER
F6D8  D0 FC             R51:    SAR     AH,1            ; c0 BIT INTO CARRY
F6DA  D1 DB                     RCR     BX,1            ;  AND INTO BX
F6DC  D0 F8                     SAR     AL,1            ; c1 BIT INTO CARRY
F6DE  D1 DB                     RCR     BX,1            ;  AND INTO BX
F6E0  E2 F6                     LOOP    R51             ; REPEAT
F6E2  8B C3                     MOV     AX,BX           ; RESULT INTO AX
F6E4  5B                        POP     BX              ; RESTORE BX
F6E5  B9 C000           R52:    MOV     CX,0C000H       ; 2 BIT MASK TO TEST THE ENTRIES
F6E8  32 D2                     XOR     DL,DL           ; RESULT REGISTER
F6EA  85 C1             R53:    TEST    AX,CX           ; IS THIS SECTION BACKGROUND?
F6EC  74 01                     JZ      R54             ; IF ZERO,  IT IS BACKGROUND
F6EE  F9                        STC                     ; WASN'T,  SO SET CARRY
F6EF  D0 D2             R54:    RCL     DL,1            ; MOVE THAT BIT INTO THE RESULT
F6F1  D1 E9                     SHR     CX,1
F6F3  D1 E9                     SHR     CX,1            ; MOVE THE MASK TO THE RIGHT BY 2
                                                        ; BITS
F6F5  73 F3                     JNC     R53             ; DO IT AGAIN IF MASK DIDN'T FALL
                                                        ; OUT
F6F7  88 56 00                  MOV     [BP],DL         ; STORE RESULT IN SAVE AREA
F6FA  45                        INC     BP              ; ADJUST POINTER
F6FB  C3                        RET                     ; ALL DONE
F6FC                    R50     ENDP
                        ;--------------------------------------------------
                        ; LOW_READ_BYTE
                        ; THIS ROUTINE WILL TAKE 4 BYTES FROM THE REGEN BUFFER,
                        ;  COMPARE FOR BACKGROUND COLOR, AND PLACE
                        ;  THE CORRESPONDING ON/OFF BIT PATTERN INTO THE CURRENT
                        ;  POSITION IN THE SAVE AREA
                        ; ENTRY --
                        ;  SI,DS = POINTER TO REGEN AREA OF INTEREST
                        ;  BP = POINTER TO SAVE AREA
                        ; EXIT --
                        ;  BP IS INCREMENT AFTER SAVE
                        ;--------------------------------------------------
F6FC                    R55     PROC    NEAR
F6FC  8A 24                     MOV     AH,[SI]         ;GET FIRST 2 BYTES
F6FE  8A 44 01                  MOV     AL,[SI+1]
F701  32 D2                     XOR     DL,DL
F703  E8 F714 R                 CALL    R56             ;BUILD HIGH NIBBLE
F706  8A 64 02                  MOV     AH,[SI+2]       ;GET SECOND 2 BYTES
F709  8A 44 03                  MOV     AL,[SI+3]
F70C  E8 F714 R                 CALL    R56             ;BUILD LOW NIBBLE
F70F  88 56 00                  MOV     [BP],DL         ;STORE RESULT IN SAVE AREA
F712  45                        INC     BP              ;ADJUST POINTER
F713  C3                        RET
F714                    R55     ENDP
F714                    R56     PROC    NEAR
F714  B9 F000                   MOV     CX,0F000H       ; 4 BIT MASK TO TEST THE ENTRIES
F717  85 C1             R57:    TEST    AX,CX           ; IS THIS SECTION BACKGROUND?
F719  74 01                     JZ      R58             ; IF ZERO,  IT IS BACKGROUND
F71B  F9                        STC                     ; WASN'T,  SO SET CARRY
F71C  D0 D2             R58:    RCL     DL,1            ;MOVE THAT BIT INTO RESULT
F71E  D1 E9                     SHR     CX,1            ;MOVE MASK RIGH 4 BITS
F720  D1 E9                     SHR     CX,1
F722  D1 E9                     SHR     CX,1
F724  D1 E9                     SHR     CX,1
F726  73 EF                     JNC     R57             ;DO IT AGAIN IF MASK DID'T FALL OUT
F728  C3                        RET
F729                    R56     ENDP
```

```
                        ;--------------------------------------
                        ; V4_POSITION
                        ;  THIS ROUTINE TAKES THE CURSOR POSITION CONTAINED IN
                        ;  THE MEMORY LOCATION, AND CONVERTS IT INTO AN OFFSET
                        ;  INTO THE REGEN BUFFER, ASSUMING ONE BYTE/CHAR.
                        ;  FOR MEDIUM RESOLUTION GRAPHICS, THE NUMBER MUST
                        ;  BE DOUBLED.
                        ; ENTRY -- NO REGISTERS,MEMORY LOCATION CURSOR_POSN IS USED
                        ; EXIT--
                        ;  AX CONTAINS OFFSET INTO REGEN BUFFER
                        ;--------------------------------------
F729                    R59     PROC    NEAR
F729  A1 0050 R                 MOV     AX,CURSOR_POSN  ; GET CURRENT CURSOR
F72C                    GRAPH_POSN      LABEL   NEAR
F72C  53                        PUSH    BX              ; SAVE REGISTER
F72D  8B D8                     MOV     BX,AX           ; SAVE A COPY OF CURRENT CURSOR
F72F  8A C4                     MOV     AL,AH           ; GET ROWS TO AL
F731  F6 26 004A R              MUL     BYTE PTR CRT_COLS ; MULTIPLY BY BYTES/COLUMN
F735  80 3E 0049 R 09           CMP     CRT_MODE,9      ; MODE USING 32K REGEN?
F73A  73 02                     JNC     R60             ; YES, JUMP
F73C  D1 E0                     SHL     AX,1            ; MULTIPLY * 4 SINCE 4 ROWS/BYTE
F73E  D1 E0                     SHL     AX,1
F740  2A FF             R60:    SUB     BH,BH           ; ISOLATE COLUMN VALUE
F742  03 C3                     ADD     AX,BX           ; DETERMINE OFFSET
F744  5B                        POP     BX              ; RECOVER POINTER
F745  C3                        RET                     ; ALL DONE
F746                    R59     ENDP
                        ;--------------------------------------------------------------------
                        ; LIGHT PEN
                        ;       THIS ROUTINE TESTS THE LIGHT PEN SWITCH AND THE LIGHT
                        ;       PEN TRIGGER.  IF BOTH ARE SET, THE LOCATION OF THE LIGHT
                        ;       PEN IS DETERMINED.  OTHERWISE, A RETURN WITH NO INFORMATION
                        ;       IS MADE.
                        ; ON EXIT:
                        ;       (AH) = 0 IF NO LIGHT PEN INFORMATION IS AVAILABLE
                        ;               BX,CX,DX ARE DESTROYED
                        ;       (AH) = 1 IF LIGHT PEN IS AVAILABLE
                        ;               (DH,DL) = ROW,COLUMN OF CURRENT LIGHT PEN POSITION
                        ;               (CH) = RASTER POSITION
                        ;               (BX) = BEST GUESS AT PIXEL HORIZONTAL POSITION
                        ;--------------------------------------------------------------------
                                ASSUME  CS:CODE,DS:DATA
                        ;----- SUBTRACT_TABLE
F746                    V1      LABEL   BYTE
F746  03 03 05 05 03 03         DB      3,3,5,5,3,3,3,0,2,3,4
      03 00 02 03 04
F751                    READ_LPEN       PROC    NEAR
                        ;----- WAIT FOR LIGHT PEN TO BE DEPRESSED
F751  32 E4                     XOR     AH,AH           ; SET NO LIGHT PEN RETURN CODE
F753  BA 03DA                   MOV     DX,VGA_CTL      ;GET ADDRESS OF VGA CONTROL REG
F756  EC                        IN      AL,DX           ; GET STATUS REGISTER
F757  A8 04                     TEST    AL,4            ; TEST LIGHT PEN SWITCH
F759  74 03                     JZ      V7B
F75B  E9 F803 R                 JMP     V6              ; NOT SET, RETURN
                        ;----- NOW TEST FOR LIGHT PEN TRIGGER
F75E  A8 02             V7B:    TEST    AL,2            ; TEST LIGHT PEN TRIGGER
F760  75 03                     JNZ     V7A             ; RETURN WITHOUT RESETTING TRIGGER
F762  E9 F80D R                 JMP     V7
                        ;----- TRIGGER HAS BEEN SET, READ THE VALUE IN
F765  B4 10             V7A:    MOV     AH,16           ; LIGHT PEN REGISTERS ON 6845
                        ;----- INPUT REGS POINTED TO BY AH, AND CONVERT TO ROW COLUMN IN DX
F767  8B 16 0063 R              MOV     DX,ADDR_6845    ; ADDRESS REGISTER FOR 6845
F76B  8A C4                     MOV     AL,AH           ; REGISTER TO READ
F76D  EE                        OUT     DX,AL           ; SET IT UP
F76E  42                        INC     DX              ; DATA REGISTER
F76F  EC                        IN      AL,DX           ; GET THE VALUE
F770  8A E8                     MOV     CH,AL           ; SAVE IN CX
F772  4A                        DEC     DX              ; ADDRESS REGISTER
F773  FE C4                     INC     AH
F775  8A C4                     MOV     AL,AH           ; SECOND DATA REGISTER
F777  EE                        OUT     DX,AL
F778  42                        INC     DX              ; POINT TO DATA REGISTER
F779  EC                        IN      AL,DX           ; GET SECOND DATA VALUE
F77A  8A E5                     MOV     AH,CH           ; AX HAS INPUT VALUE
                        ;----- AX HAS THE VALUE READ IN FROM THE 6845
F77C  8A 1E 0049 R              MOV     BL,CRT_MODE
F780  2A FF                     SUB     BH,BH           ; MODE VALUE TO BX
F782  2E: 8A 9F F746 R          MOV     BL,CS:V1[BX]    ; DETERMINE AMOUNT TO SUBTRACT
F787  2B C3                     SUB     AX,BX           ; TAKE IT AWAY
F789  3D 0FA0                   CMP     AX,4000         ; IN TOP OR BOTTOM BORDER?
F78C  72 02                     JB      V15             ; NO, OKAY
F78E  33 C0                     XOR     AX,AX           ; YES, SET TO ZERO
F790  8B 1E 004E R      V15:    MOV     BX,CRT_START
F794  D1 EB                     SHR     BX,1
F796  2B C3                     SUB     AX,BX           ; CONVERT TO CORRECT PAGE ORIGIN
F798  79 02                     JNS     V2              ; IF POSITIVE, DETERMINE MODE
F79A  2B C0                     SUB     AX,AX           ; <0 PLAYS AS 0
                        ;----- DETERMINE MODE OF OPERATION
F79C                    V2:                             ; DETERMINE_MODE
F79C  B1 03                     MOV     CL,3            ; SET #8 SHIFT COUNT
F79E  80 3E 0049 R 04           CMP     CRT_MODE,4      ; DETERMINE IF GRAPHICS OR ALPHA
F7A3  72 4A                     JB      V4              ; ALPHA_PEN
                        ;----- GRAPHICS MODE
F7A5  B2 28                     MOV     DL,40           ; DIVISOR FOR GRAPHICS
F7A7  80 3E 0049 R 09           CMP     CRT_MODE,9      ; USING 32K REGEN?
F7AC  72 02                     JB      V20             ; NO, JUMP
F7AE  B2 50                     MOV     DL,80           ; YES, SET RIGHT DIVSOR
F7B0  F6 F2             V20:    DIV     DL              ; DETERMINE ROW(AL) AND COLUMN(AH)
                                                        ;  AL RANGE 0-99, AH RANGE 0-39
```

```
                        ;----- DETERMINE GRAPHIC ROW POSITION
F7B2  8A E8                     MOV     CH,AL           ; SAVE ROW VALUE IN CH
F7B4  02 ED                     ADD     CH,CH           ; *2 FOR EVEN/ODD FIELD
F7B6  80 3E 0049 R 09           CMP     CRT_MODE,9      ; USING 32K REGEN?
F7BB  72 06                     JB      V21             ; NO, JUMP
F7BD  D0 EC                     SHR     AH,1            ; ADJUST ROW & COLUMN
F7BF  D0 E0                     SHL     AL,1
F7C1  02 ED                     ADD     CH,CH           ; *4 FOR 4 SCAN LINES
F7C3  8A DC             V21:    MOV     BL,AH           ; COLUMN VALUE TO BX
F7C5  2A FF                     SUB     BH,BH           ; MULTIPLY BY 8 FOR MEDIUM RES
F7C7  80 3E 0049 R 06           CMP     CRT_MODE,6      ; DETERMINE MEDIUM OR HIGH RES
F7CC  72 15                     JB      V3              ; MODE 4 OR 5
F7CE  77 06                     JA      V23             ; MODE 8, 9, OR A
F7D0  B1 04             V22:    MOV     CL,4            ; SHIFT VALUE FOR HIGH RES
F7D2  D0 E4                     SAL     AH,1            ; COLUMN VALUE TIMES 2 FOR HIGH RES
F7D4  EB 0D                     JMP     SHORT V3
F7D6  80 3E 0049 R 09   V23:    CMP     CRT_MODE,9      ; CHECK MODE
F7DB  77 F3                     JA      V22             ; MODE A
F7DD  74 04                     JE      V3              ; MODE 9
F7DF  B1 02                     MOV     CL,2            ; MODE 8 SHIFT VALUE
F7E1  D0 EC                     SHR     AH,1
F7E3                    V3:                             ; NOT_HIGH_RES
F7E3  D3 E3                     SHL     BX,CL           ; MULTIPLY *16 FOR HIGH RES
                        ;----- DETERMINE ALPHA CHAR POSITION
F7E5  8A D4                     MOV     DL,AH           ; COLUMN VALUE FOR RETURN
F7E7  8A F0                     MOV     DH,AL           ; ROW VALUE
F7E9  D0 EE                     SHR     DH,1            ; DIVIDE BY 4
F7EB  D0 EE                     SHR     DH,1            ;  FOR VALUE IN 0-24 RANGE
F7ED  EB 12                     JMP     SHORT V5        ; LIGHT_PEN_RETURN_SET
                        ;----- ALPHA MODE ON LIGHT PEN
F7EF                    V4:                             ; ALPHA_PEN
F7EF  F6 36 004A R              DIV     BYTE PTR CRT_COLS ; DETERMINE ROW,COLUMN VALUE
F7F3  8A F0                     MOV     DH,AL           ; ROWS TO DH
F7F5  8A D4                     MOV     DL,AH           ; COLS TO DL
F7F7  D2 E0                     SAL     AL,CL           ; MULTIPLY ROWS * 8
F7F9  8A E8                     MOV     CH,AL           ; GET RASTER VALUE TO RETURN REG
F7FB  8A DC                     MOV     BL,AH           ; COLUMN VALUE
F7FD  32 FF                     XOR     BH,BH           ;  TO BX
F7FF  D3 E3                     SAL     BX,CL
F801                    V5:                             ; LIGHT_PEN_RETURN_SET
F801  B4 01                     MOV     AH,1            ; INDICATE EVERYTHING SET
F803                    V6:                             ; LIGHT_PEN_RETURN
F803  52                        PUSH    DX              ; SAVE RETURN VALUE (IN CASE)
F804  8B 16 0063 R              MOV     DX,ADDR_6845    ; GET BASE ADDRESS
F808  83 C2 07                  ADD     DX,7            ; POINT TO RESET PARM
F80B  EE                        OUT     DX,AL           ; ADDRESS, NOT DATA, IS IMPORTANT
F80C  5A                        POP     DX              ; RECOVER VALUE
F80D                    V7:                             ; RETURN_NO_RESET
F80D  5F                        POP     DI
F80E  5E                        POP     SI
F80F  1F                        POP     DS              ; DISCARD SAVED BX,CX,DX
F810  1F                        POP     DS
F811  1F                        POP     DS
F812  1F                        POP     DS
F813  07                        POP     ES
F814  CF                        IRET
F815                    READ_LPEN       ENDP
                        ;--------------------------------------------------------
                        ; TEMPORARY INTERRUPT SERVICE ROUTINE                   :
                        ;       1. THIS ROUTINE IS ALSO LEFT IN PLACE AFTER THE :
                        ;       POWER ON DIAGNOSTICS TO SERVICE UNUSED          :
                        ;       INTERRUPT VECTORS. LOCATION 'INTR_FLAG' WILL    :
                        ;       CONTAIN EITHER: 1. LEVEL OF HARDWARE INT. THAT  :
                        ;       CAUSED CODE TO BE EXEC.                         :
                        ;       2. 'FF' FOR NON-HARDWARE INTERRUPTS THAT WERE   :
                        ;       EXECUTED ACCIDENTLY.                            :
                        ;--------------------------------------------------------
F815                    D11     PROC    NEAR
                                ASSUME  DS:DATA
F815  1E                        PUSH    DS
F816  50                        PUSH    AX              ; SAVE REG AX CONTENTS
F817  E8 138B R                 CALL    DDS
F81A  B0 0B                     MOV     AL,0BH          ; READ IN-SERVICE REG
F81C  E6 20                     OUT     INTA00,AL       ; (FIND OUT WHAT LEVEL BEING
F81E  90                        NOP                     ; SERVICED)
F81F  E4 20                     IN      AL,INTA00       ; GET LEVEL
F821  8A E0                     MOV     AH,AL           ; SAVE IT
F823  0A C4                     OR      AL,AH           ; 00? (NO HARDWARE ISR ACTIVE)
F825  75 04                     JNZ     HW_INT
F827  B4 FF                     MOV     AH,0FFH
F829  EB 0A                     JMP     SHORT SET_INTR_FLAG ; SET FLAG TO FF IF NON-HDWARE
F82B  E4 21             HW_INT: IN      AL,INTA01       ; GET MASK VALUE
F82D  0A C4                     OR      AL,AH           ; MASK OFF LVL BEING SERVICED
F82F  E6 21                     OUT     INTA01,AL
F831  B0 20                     MOV     AL,EOI
F833  E6 20                     OUT     INTA00,AL
F835                    SET_INTR_FLAG:
F835  88 26 0084 R              MOV     INTR_FLAG,AH    ; SET FLAG
F839  58                        POP     AX              ; RESTORE REG AX CONTENTS
F83A  1F                        POP     DS
F83B  FB                        STI                     ; INTERRUPTS BACK ON
F83C                    DUMMY_RETURN:                   ; NEED IRET FOR VECTOR TABLE
F83C  CF                        IRET
F83D                    D11     ENDP
```

```
                        ;--- INT 12 ---------------------------------
                        ; MEMORY_SIZE_DETERMINE
                        ; INPUT
                        ;       NO REGISTERS
                        ;       THE MEMORY_SIZE VARIABLE IS SET DURING POWER ON DIAGNOSTICS
                        ; OUTPUT
                        ;       (AX) = NUMBER OF CONTIGUOUS 1K BLOCKS OF MEMORY
                        ;-----------------------------------------------
                                ASSUME  CS:CODE,DS:DATA
F841                            ORG     0F841H
F841                    MEMORY_SIZE_DETERMINE   PROC    FAR
F841  FB                        STI                     ; INTERRUPTS BACK ON
F842  1E                        PUSH    DS              ; SAVE SEGMENT
F843  B8  ---- R                MOV     AX,DATA         ; ESTABLISH ADDRESSING
F846  8E D8                     MOV     DS,AX
F848  A1 0013 R                 MOV     AX,MEMORY_SIZE  ; GET VALUE
F84B  1F                        POP     DS              ; RECOVER SEGMENT
F84C  CF                        IRET                    ; RETURN TO CALLER
F84D                    MEMORY_SIZE_DETERMINE   ENDP
                        ;--- INT 11 ---------------------------------
                        ; EQUIPMENT DETERMINATION
                        ;       THIS ROUTINE ATTEMPTS TO DETERMINE WHAT OPTIONAL
                        ;       DEVICES ARE ATTACHED TO THE SYSTEM.
                        ; INPUT
                        ;       NO REGISTERS
                        ;       THE EQUIP_FLAG VARIABLE IS SET DURING THE POWER ON
                        ;        DIAGNOSTICS USING THE FOLLOWING HARDWARE ASSUMPTIONS:
                        ;       PORT 62 (0->3) = LOW ORDER BYTE OF EQUIPMENT
                        ;       PORT 3FA = INTERRUPT ID REGISTER OF 8250
                        ;                  BITS 7-3 ARE ALWAYS 0
                        ;       PORT 378 = OUTPUT PORT OF PRINTER -- 8255 PORT THAT
                        ;                  CAN BE READ AS WELL AS WRITTEN
                        ;  OUTPUT
                        ;       (AX) IS SET, BIT SIGNIFICANT, TO INDICATE ATTACHED I/O
                        ;       BIT 15,14 = NUMBER OF PRINTERS ATTACHED
                        ;       BIT 13 = 1 = SERIAL PRINTER ATTACHED
                        ;       BIT 12 = GAME I/O ATTACHED
                        ;       BIT 11,10,9 = NUMBER OF RS232 CARDS ATTACHED
                        ;       BIT 8  0 = DMA CHIP PRESENT ON SYSTEM, 1 = NO DMA ON SYSTEM
                        ;       BIT 7,6 = NUMBER OF DISKETTE DRIVES
                        ;                 00=1, 01=2, 10=3, 11=4 ONLY IF BIT 0 = 1
                        ;       BIT 5,4 = INITIAL VIDEO MODE
                        ;                         00 - UNUSED
                        ;                         01 - 40X25 BW USING COLOR CARD
                        ;                         10 - 80X25 BW USING COLOR CARD
                        ;                         11 - 80X25 BW USING BW CARD
                        ;       BIT 3,2 = PLANAR RAM SIZE (10=48K,11=64K)
                        ;       BIT 1 NOT USED
                        ;       BIT 0 = 1 (IPL DISKETTE INSTALLED)
                        ;       NO OTHER REGISTERS AFFECTED
                        ;-----------------------------------------------
                                ASSUME  CS:CODE,DS:DATA
F84D                            ORG      0F84DH
F84D                    EQUIPMENT       PROC    FAR
F84D  FB                        STI                     ; INTERRUPTS BACK ON
F84E  1E                        PUSH    DS              ; SAVE SEGMENT REGISTER
F84F  B8  ---- R                MOV     AX,DATA         ; ESTABLISH ADDRESSING
F852  8E D8                     MOV     DS,AX
F854  A1 0010 R                 MOV     AX,EQUIP_FLAG   ; GET THE CURRENT SETTINGS
F857  1F                        POP     DS              ; RECOVER SEGMENT
F858  CF                        IRET                    ; RETURN TO CALLER
F859                    EQUIPMENT       ENDP
                        ;--- INT 15 ---------------------------------------------------------
                        ; CASSETTE I/O
                        ;       (AH) = 0  TURN CASSETTE MOTOR ON
                        ;       (AH) = 1  TURN CASSETTE MOTOR OFF
                        ;       (AH) = 2  READ 1 OR MORE 256 BYTE BLOCKS FROM CASSETTE
                        ;                 (ES,BX) = POINTER TO DATA BUFFER
                        ;                 (CX) = COUNT OF BYTES TO READ
                        ; ON EXIT
                        ;       (ES,BX) = POINTER TO LAST BYTE READ + 1
                        ;       (DX) = COUNT OF BYTES ACTUALLY READ
                        ;       (CY) = 0 IF NO ERROR OCCURRED
                        ;            = 1 IF ERROR OCCURRED
                        ;       (AH) = ERROR RETURN IF (CY)= 1
                        ;                 = 01 IF CRC ERROR WAS DETECTED
                        ;                 = 02 IF DATA TRANSITIONS ARE LOST
                        ;                 = 04 IF NO DATA WAS FOUND
                        ;       (AH) = 3 WRITE 1 OR MORE 256 BYTE BLOCKS TO CASSETTE
                        ;                 (ES,BX) = POINTER TO DATA BUFFER
                        ;                 (CX) = COUNT OF BYTES TO WRITE
                        ; ON EXIT
                        ;       (EX,BX) = POINTER TO LAST BYTE WRITTEN + 1
                        ;       (CX) = 0
                        ;       (AH) = ANY OTHER THAN ABOVE VALUES CAUSES (CY)= 1
                        ;                 AND (AH)= 80 TO BE RETURNED (INVALID COMMAND).
                        ;---------------------------------------------------------------------
                                ASSUME  DS:DATA, ES:NOTHING,SS:NOTHING,CS:CODE
F859                            ORG     0F859H
F859                    CASSETTE_IO     PROC    FAR
F859  FB                        STI                     ; INTERRUPTS BACK ON
F85A  1E                        PUSH    DS              ; ESTABLISH ADDRESSING TO DATA
F85B  E8 138B R                 CALL    DDS
F85E  80 26 0071 R 7F           AND     BIOS_BREAK, 7FH ; MAKE SURE BREAK FLAG IS OFF
F863  E8 F86A R                 CALL    W1              ; CASSETTE_IO_CONT
F866  1F                        POP     DS
F867  CA 0002                   RET     2               ; INTERRUPT RETURN
F86A                    CASSETTE_IO     ENDP
F86A                    W1      PROC    NEAR
```

```
                        ;--------------------------------------------------------
                        ; PURPOSE:
                        ;  TO CALL APPROPRIATE ROUTINE DEPENDING ON REG AH
                        ;  AH            ROUTINE
                        ;--------------------------------------------------------
                        ;  0             MOTOR ON
                        ;  1             MOTOR OFF
                        ;  2             READ CASSETTE BLOCK
                        ;  3             WRITE CASSETTE BLOCK
                        ;--------------------------------------------------------
F86A  0A E4                     OR      AH,AH           ; TURN ON MOTOR?
F86C  74 13                     JZ      MOTOR_ON        ; YES, DO IT
F86E  FE CC                     DEC     AH              ; TURN OFF MOTOR?
F870  74 18                     JZ      MOTOR_OFF       ; YES, DO IT
F872  FE CC                     DEC     AH              ; READ CASSETTE BLOCK?
F874  74 1A                     JZ      READ_BLOCK      ; YES, DO IT
F876  FE CC                     DEC     AH              ; WRITE CASSETTE BLOCK?
F878  75 03                     JNZ     W2              ;  NOT_DEFINED
F87A  E9 F997 R                 JMP     WRITE_BLOCK     ; YES, DO IT
F87D                    W2:                             ; COMMAND NOT DEFINED
F87D  B4 80                     MOV     AH,080H         ; ERROR, UNDEFINED OPERATION
F87F  F9                        STC                     ; ERROR FLAG
F880  C3                        RET
F881                    W1      ENDP
F881                    MOTOR_ON        PROC    NEAR
                        ;--------------------------------------
                        ; PURPOSE:
                        ;       TO TURN ON CASSETTE MOTOR
                        ;--------------------------------------
F881  E4 61                     IN      AL,PORT_B       ; READ CASSETTE OUTPUT
F883  24 F7                     AND     AL,NOT 08H      ; CLEAR BIT TO TURN ON MOTOR
F885  E6 61             W3:     OUT     PORT_B,AL       ; WRITE IT OUT
F887  2A E4                     SUB     AH,AH           ; CLEAR AH
F889  C3                        RET
F88A                    MOTOR_ON        ENDP
F88A                    MOTOR_OFF       PROC    NEAR
                        ;--------------------------------------
                        ; PURPOSE:                            :
                        ;       TO TURN CASSETTE MOTOR OFF      :
                        ;--------------------------------------
F88A  E4 61                     IN      AL,PORT_B       ; READ CASSETTE OUTPUT
F88C  0C 08                     OR      AL,08H          ; SET BIT TO TURN OFF
F88E  EB F5                     JMP     W3              ; WRITE IT, CLEAR ERROR, RETURN
F890                    MOTOR_OFF       ENDP
F890                    READ_BLOCK      PROC    NEAR
                        ;--------------------------------------------------------------
                        ; PURPOSE:
                        ;       TO READ 1 OR MORE 256 BYTE BLOCKS FROM CASSETTE
                        ;
                        ; ON ENTRY:
                        ;       ES IS SEGMENT FOR MEMORY BUFFER (FOR COMPACT CODE)
                        ;       BX POINTS TO START OF MEMORY BUFFER
                        ;       CX CONTAINS NUMBER OF BYTES TO READ
                        ; ON EXIT:
                        ;       BX POINTS 1 BYTE PAST LAST BYTE PUT IN MEM
                        ;       CX CONTAINS DECREMENTED BYTE COUNT
                        ;       DX CONTAINS NUMBER OF BYTES ACTUALLY READ
                        ;
                        ;       CARRY FLAG IS CLEAR IF NO ERROR DETECTED
                        ;       CARRY FLAG IS SET IF CRC ERROR DETECTED
                        ;--------------------------------------------------------------
F890  53                        PUSH    BX              ; SAVE BX
F891  51                        PUSH    CX              ; SAVE CX
F892  56                        PUSH    SI              ; SAVE SI
F893  BE 0007                   MOV     SI, 7           ; SET UP RETRY COUNT FOR LEADER
F896  E8 FA50 R                 CALL    BEGIN_OP        ; BEGIN BY STARTING MOTOR
F899                    W4:                             ; SEARCH FOR LEADER
F899  E4 62                     IN      AL,PORT_C       ; GET INITIAL VALUE
F89B  24 10                     AND     AL,010H         ; MASK OFF EXTRANEOUS BITS
F89D  A2 006B R                 MOV     LAST_VAL,AL     ; SAVE IN LOC LAST_VAL
F8A0  BA 3F7A                   MOV     DX,16250        ; # OF TRANSITIONS TO LOOK FOR
F8A3                    W5:                             ; WAIT_FOR_EDGE
F8A3  F6 06 0071 R 80           TEST    BIOS_BREAK, 80H ; CHECK FOR BREAK KEY
F8A8  75 03                     JNZ     W6A             ; JUMP IF NO BREAK KEY
                                                        ; JUMP IF BREAK KEY HIT
F8AA  4A                        DEC     DX
F8AB  75 03                     JNZ     W7              ; JUMP IF BEGINNING OF LEADER
F8AD  E9 F92F R         W6A:    JMP     W17             ; JUMP IF NO LEADER FOUND
F8B0  E8 F96F R         W7:     CALL    READ_HALF_BIT   ; IGNORE FIRST EDGE
F8B3  E3 EE                     JCXZ    W5              ; JUMP IF NO EDGE DETECTED
F8B5  BA 0378                   MOV     DX,0378H        ; CHECK FOR HALF BITS
F8B8  B9 0200                   MOV     CX,200H         ; MUST HAVE AT LEAST THIS MANY ONE
                                                        ; SIZE PULSES BEFORE CHCKNG FOR
                                                        ; SYNC BIT (0)
F8BB  FA                        CLI                     ; DISABLE INTERRUPTS
F8BC                    W8:                             ; SEARCH-LDR
F8BC  F6 06 0071 R 80           TEST    BIOS_BREAK, 80H ; CHECK FOR BREAK KEY
F8C1  75 6C                     JNZ     W17             ; JUMP IF BREAK KEY HIT
F8C3  51                        PUSH    CX              ; SAVE REG CX
F8C4  E8 F96F R                 CALL    READ_HALF_BIT   ; GET PULSE WIDTH
F8C7  0B C9                     OR      CX, CX          ; CHECK FOR TRANSITION
F8C9  59                        POP     CX              ; RESTORE ONE BIT COUNTER
F8CA  74 CD                     JZ      W4              ; JUMP IF NO TRANSITION
F8CC  3B D3                     CMP     DX,BX           ; CHECK PULSE WIDTH
F8CE  E3 04                     JCXZ    W9              ; IF CX=0 THEN WE CAN LOOK
                                                        ; FOR SYNC BIT (0)
F8D0  73 C7                     JNC     W4              ; JUMP IF ZERO BIT (NOT GOOD
                                                        ; LEADER)
F8D2  E2 E8                     LOOP    W8              ; DEC CX AND READ ANOTHER HALF ONE
                                                        ; BIT
F8D4                    W9:                             ; FIND-SYNC
F8D4  72 E6                     JC      W8              ; JUMP IF ONE BIT (STILL LEADER)
```

```
                                ;----- A SYNCH BIT HAS BEEN FOUND.  READ SYN CHARACTER:
F8D6  E8 F96F R                         CALL    READ_HALF_BIT   ; SKIP OTHER HALF OF SYNC BIT (0)
F8D9  E8 F941 R                         CALL    READ_BYTE       ; READ SYNC BYTE
F8DC  3C 16                             CMP     AL, 16H         ; SYNCHRONIZATION CHARACTER
F8DE  75 49                             JNE     W16             ; JUMP IF BAD LEADER FOUND.
                                ;----- GOOD CRC SO READ DATA BLOCK(S)
F8E0  5E                                POP     SI              ; RESTORE REGS
F8E1  59                                POP     CX
F8E2  5B                                POP     BX
                                ;--------------------------------------------------------------
                                ; READ 1 OR MORE 256 BYTE BLOCKS FROM CASSETTE
                                ; ON ENTRY:
                                ;       ES IS SEGMENT FOR MEMORY BUFFER (FOR COMPACT CODE)
                                ;       BX POINTS TO START OF MEMORY BUFFER
                                ;       CX CONTAINS NUMBER OF BYTES TO READ
                                ; ON EXIT:
                                ;       BX POINTS 1 BYTE PAST LAST BYTE PUT IN MEM
                                ;       CX CONTAINS DECREMENTED BYTE COUNT
                                ;       DX CONTAINS NUMBER OF BYTES ACTUALLY READ
                                ;--------------------------------------------------------------
F8E3  51                                PUSH    CX              ; SAVE BYTE COUNT
F8E4                            W10:                            ; COME HERE BEFORE EACH
                                                                ; 256 BYTE BLOCK IS READ
F8E4  C7 06 0069 R FFFF                 MOV     CRC_REG,0FFFFH  ; INIT CRC REG
F8EA  BA 0100                           MOV     DX,256          ; SET DX TO DATA BLOCK SIZE
F8ED                            W11:                            ; RD_BLK
F8ED  F6 06 0071 R 80                   TEST    BIOS_BREAK, 80H ; CHECK FOR BREAK KEY
F8F2  75 23                             JNZ     W13             ; JUMP IF BREAK KEY HIT
F8F4  E8 F941 R                         CALL    READ_BYTE       ; READ BYTE FROM CASSETTE
F8F7  72 1E                             JC      W13             ; CY SET INDICATES NO DATA
                                                                ; TRANSITIONS
F8F9  E3 05                             JCXZ    W12             ; IF WE'VE ALREADY REACHED
                                                                ; END OF MEMORY BUFFER
                                                                ; SKIP REST OF BLOCK
F8FB  26: 88 07                         MOV     ES:[BX],AL      ; STORE DATA BYTE AT BYTE PTR
F8FE  43                                INC     BX              ; INC BUFFER PTR
F8FF  49                                DEC     CX              ; DEC BYTE COUNTER
F900                            W12:                            ; LOOP UNTIL DATA BLOCK HAS BEEN READ FROM CASSETTE
F900  4A                                DEC     DX              ; DEC BLOCK CNT
F901  7F EA                             JG      W11             ; RD_BLK
F903  E8 F941 R                         CALL    READ_BYTE       ; NOW READ TWO CRC BYTES
F906  E8 F941 R                         CALL    READ_BYTE
F909  2A E4                             SUB     AH,AH           ; CLEAR AH
F90B  81 3E 0069 R 1D0F                 CMP     CRC_REG,1D0FH   ; IS THE CRC CORRECT?
F911  75 06                             JNE     W14             ; IF NOT EQUAL CRC IS BAD
F913  E3 06                             JCXZ    W15             ; IF BYTE COUNT IS ZERO
                                                                ; THEN WE HAVE READ ENOUGH
                                                                ; SO WE WILL EXIT
F915  EB CD                             JMP     W10             ; STILL MORE, SO READ ANOTHER BLOCK
F917                            W13:                            ; MISSING-DATA
                                                                ; NO DATA TRANSITIONS SO
F917  B4 01                             MOV     AH,01H          ; SET AH=02 TO INDICATE
                                                                ; DATA TIMEOUT
F919                            W14:                            ; BAD-CRC
F919  FE C4                             INC     AH              ; EXIT EARLY ON ERROR
                                                                ; SET AH=01 TO INDICATE CRC ERROR
F91B                            W15:                            ; RD-BLK-EX
F91B  5A                                POP     DX              ; CALCULATE COUNT OF
F91C  2B D1                             SUB     DX,CX           ; DATA BYTES ACTUALLY READ
                                                                ; RETURN COUNT IN REG DX
F91E  50                                PUSH    AX              ; SAVE AX (RET CODE)
F91F  F6 C4 90                          TEST    AH, 90H         ; CHECK FOR ERRORS
F922  75 13                             JNZ     W18             ; JUMP IF ERROR DETECTED
F924  E8 F941 R                         CALL    READ_BYTE       ; READ TRAILER
F927  EB 0E                             JMP     SHORT W18       ; SKIP TO TURN OFF MOTOR
F929                            W16:                            ; BAD-LEADER
F929  4E                                DEC     SI              ; CHECK RETRIES
F92A  74 03                             JZ      W17             ; JUMP IF TOO MANY RETRIES
F92C  E9 F899 R                         JMP     W4              ; JUMP IF NOT TOO MANY RETRIES
F92F                            W17:                            ; NO VALID DATA FOUND
                                ;----- NO DATA FROM CASSETTE ERROR, I.E. TIMEOUT
F92F  5E                                POP     SI              ; RESTORE REGS
F930  59                                POP     CX              ; RESTORE REGS
F931  5B                                POP     BX
F932  2B D2                             SUB     DX,DX           ; ZERO NUMBER OF BYTES READ
F934  B4 04                             MOV     AH,04H          ; TIME OUT ERROR (NO LEADER)
F936  50                                PUSH    AX
F937                            W18:                            ; MOT-OFF
F937  FB                                STI                     ; REENABLE INTERRUPTS
F938  E8 F88A R                         CALL    MOTOR_OFF       ; TURN OFF MOTOR
F93B  58                                POP     AX              ; RESTORE RETURN CODE
F93C  80 FC 01                          CMP     AH,01H          ; SET CARRY IF ERROR (AH>0)
F93F  F5                                CMC
F940  C3                                RET                     ; FINISHED
F941                            READ_BLOCK      ENDP
```

```
                                ;----------------------------------------
                                ; PURPOSE:
                                ;       TO READ A BYTE FROM CASSETTE
                                ; ON EXIT
                                ;       REG AL CONTAINS READ DATA BYTE
                                ;----------------------------------------
F941                            READ_BYTE       PROC    NEAR
F941  53                                PUSH    BX              ; SAVE REGS BX,CX
F942  51                                PUSH    CX
F943  B1 08                             MOV     CL,8H           ; SET BIT COUNTER FOR 8 BITS
F945                            W19:                            ; BYTE-ASM
F945  51                                PUSH    CX              ; SAVE CX
                                ;------------------------------
                                ; READ DATA BIT FROM CASSETTE
                                ;------------------------------
F946  E8 F96F R                         CALL    READ_HALF_BIT   ; READ ONE PULSE
F949  E3 20                             JCXZ    W21             ; IF CX=0 THEN TIMEOUT
                                                                ; BECAUSE OF NO DATA TRANSITIONS
F94B  53                                PUSH    BX              ; SAVE 1ST HALF BIT'S
                                                                ; PULSE WIDTH (IN BX)
F94C  E8 F96F R                         CALL    READ_HALF_BIT   ; READ COMPLEMENTARY PULSE
F94F  58                                POP     AX              ; COMPUTE DATA BIT
F950  E3 19                             JCXZ    W21             ; IF CX=0 THEN TIMEOUT DUE TO
                                                                ; NO DATA TRANSITIONS
F952  03 D8                             ADD     BX,AX           ; PERIOD
F954  81 FB 06F0                        CMP     BX, 06F0H       ; CHECK FOR ZERO BIT
F958  F5                                CMC                     ; CARRY IS SET IF ONE BIT
F959  9F                                LAHF                    ; SAVE CARRY IN AH
F95A  59                                POP     CX              ; RESTORE CX
                                                                ; NOTE:
                                                                ; MS BIT OF BYTE IS READ FIRST.
                                                                ; REG CH IS SHIFTED LEFT WITH
                                                                ; CARRY BEING INSERTED INTO LS
                                                                ; BIT OF CH.
                                                                ; AFTER ALL 8 BITS HAVE BEEN
                                                                ; READ, THE MS BIT OF THE DATA
                                                                ; BYTE WILL BE IN THE MS BIT OF
                                                                ; REG CH
F95B  D0 D5                             RCL     CH,1            ; ROTATE REG CH LEFT WITH CARRY TO
                                                                ;   LS BIT OF REG CH
F95D  9E                                SAHF                    ; RESTORE CARRY FOR CRC ROUTINE
F95E  E8 FA3C R                         CALL    CRC_GEN         ; GENERATE CRC FOR BIT
F961  FE C9                             DEC     CL              ; LOOP TILL ALL 8 BITS OF DATA
                                                                ; ASSEMBLED IN REG CH
F963  75 E0                             JNZ     W19             ; BYTE_ASM
F965  8A C5                             MOV     AL,CH           ; RETURN DATA BYTE IN REG AL
F967  F8                                CLC
F968                            W20:                            ; RD-BYT-EX
F968  59                                POP     CX              ; RESTORE REGS CX,BX
F969  5B                                POP     BX
F96A  C3                                RET                     ; FINISHED
F96B                            W21:                            ; NO-DATA
F96B  59                                POP     CX              ; RESTORE CX
F96C  F9                                STC                     ; INDICATE ERROR
F96D  EB F9                             JMP     W20             ; RD_BYT_EX
F96F                            READ_BYTE       ENDP
                                ;--------------------------------------------
                                ; PURPOSE:
                                ;       TO COMPUTE TIME TILL NEXT DATA
                                ;       TRANSITION (EDGE)
                                ; ON ENTRY:
                                ;       EDGE_CNT CONTAINS LAST EDGE COUNT
                                ; ON EXIT:
                                ;       AX CONTAINS OLD LAST EDGE COUNT
                                ;       BX CONTAINS PULSE WIDTH (HALF BIT)
                                ;--------------------------------------------
F96F                            READ_HALF_BIT   PROC    NEAR
F96F  B9 0064                           MOV     CX, 100         ; SET TIME TO WAIT FOR BIT
F972  8A 26 006B R                      MOV     AH,LAST_VAL     ; GET PRESENT INPUT VALUE
F976                            W22:                            ; RD-H-BIT
F976  E4 62                             IN      AL,PORT_C       ; INPUT DATA BIT
F978  24 10                             AND     AL,010H         ; MASK OFF EXTRANEOUS BITS
F97A  3A C4                             CMP     AL,AH           ; SAME AS BEFORE?
F97C  E1 F8                             LOOPE   W22             ; LOOP TILL IT CHANGES
F97E  A2 006B R                         MOV     LAST_VAL,AL     ; UPDATE LAST_VAL WITH NEW VALUE
F981  B0 40                             MOV     AL,40H          ; READ TIMER'S COUNTER COMMAND
F983  E6 43                             OUT     TIM_CTL,AL      ; LATCH COUNTER
F985  8B 1E 0067 R                      MOV     BX,EDGE_CNT     ; BX GETS LAST EDGE COUNT
F989  E4 41                             IN      AL,TIMER+1      ; GET LS BYTE
F98B  8A E0                             MOV     AH,AL           ; SAVE IN AH
F98D  E4 41                             IN      AL,TIMER+1      ; GET MS BYTE
F98F  86 C4                             XCHG    AL,AH           ; XCHG AL,AH
F991  2B D8                             SUB     BX,AX           ; SET BX EQUAL TO HALF BIT PERIOD
F993  A3 0067 R                         MOV     EDGE_CNT,AX     ; UPDATE EDGE COUNT;
F996  C3                                RET
F997                            READ_HALF_BIT   ENDP
```

```
                        ;--------------------------------------------------------------
                        ; PURPOSE
                        ;       WRITE 1 OR MORE 256 BYTE BLOCKS TO CASSETTE.
                        ;       THE DATA IS PADDED TO FILL OUT THE LAST 256 BYTE BLOCK.
                        ; ON ENTRY:
                        ;       BX POINTS TO MEMORY BUFFER ADDRESS
                        ;       CX CONTAINS NUMBER OF BYTES TO WRITE
                        ; ON EXIT:
                        ;       BX POINTS 1 BYTE PAST LAST BYTE WRITTEN TO CASSETTE
                        ;       CX IS ZERO
                        ;--------------------------------------------------------------
F997                    WRITE_BLOCK     PROC    NEAR
F997  53                        PUSH    BX
F998  51                        PUSH    CX
F999  E4 61                     IN      AL,PORT_B       ; DISABLE SPEAKER
F99B  24 FD                     AND     AL,NOT 02H
F99D  0C 01                     OR      AL, 01H         ; ENABLE TIMER
F99F  E6 61                     OUT     PORT_B,AL
F9A1  B0 B6                     MOV     AL,0B6H         ; SET UP TIMER - MODE 3 SQUARE WAVE
F9A3  E6 43                     OUT     TIM_CTL,AL
F9A5  E8 FA50 R                 CALL    BEGIN_OP        ; START MOTOR AND DELAY
F9A8  B8 04A0                   MOV     AX,1184         ; SET NORMAL BIT SIZE
F9AB  E8 FA35 R                 CALL    W31             ; SET_TIMER
F9AE  B9 0800                   MOV     CX,0800H        ; SET CX FOR LEADER BYTE COUNT
F9B1                    W23:                            ; WRITE LEADER
F9B1  F9                        STC                     ; WRITE ONE BITS
F9B2  E8 FA1F R                 CALL    WRITE_BIT
F9B5  E2 FA                     LOOP    W23             ; LOOP 'TIL LEADER IS WRITTEN
F9B7  FA                        CLI                     ; DISABLE INTS.
F9B8  F8                        CLC                     ; WRITE SYNC BIT (0)
F9B9  E8 FA1F R                 CALL    WRITE_BIT
F9BC  59                        POP     CX              ; RESTORE REGS CX,BX
F9BD  5B                        POP     BX
F9BE  B0 16                     MOV     AL, 16H         ; WRITE SYNC CHARACTER
F9C0  E8 FA08 R                 CALL    WRITE_BYTE
                        ;--------------------------------------------------------------
                        ; PURPOSE
                        ;       WRITE 1 OR MORE 256 BYTE BLOCKS TO CASSETTE
                        ; ON ENTRY:
                        ;       BX POINTS TO MEMORY BUFFER ADDRESS
                        ;       CONTAINS NUMBER OF BYTES TO WRITE
                        ; ON EXIT:
                        ;       BX POINTS 1 BYTE PAST LAST BYTE WRITTEN TO CASSETTE
                        ;       CX IS ZERO
                        ;--------------------------------------------------------------
F9C3                    WR_BLOCK:
F9C3  C7 06 0069 R FFFF         MOV     CRC_REG,0FFFFH  ; INIT CRC
F9C9  BA 0100                   MOV     DX,256          ; FOR 256 BYTES
F9CC                    W24:                            ; WR-BLK
F9CC  26: 8A 07                 MOV     AL,ES:[BX]      ; READ BYTE FROM MEM
F9CF  E8 FA08 R                 CALL    WRITE_BYTE      ; WRITE IT TO CASSETTE
F9D2  E3 02                     JCXZ    W25             ; UNLESS CX=0, ADVANCE PTRS & DEC
                                                        ;  COUNT
F9D4  43                        INC     BX              ; INC BUFFER POINTER
F9D5  49                        DEC     CX              ; DEC BYTE COUNTER
F9D6                    W25:                            ; SKIP-ADV
F9D6  4A                        DEC     DX              ; DEC BLOCK CNT
F9D7  7F F3                     JG      W24             ; LOOP TILL 256 BYTE BLOCK
                                                        ;  IS WRITTEN TO TAPE
                        ;--------------------------------------------------------------
                        ; WRITE CRC
                        ;       WRITE 1'S COMPLEMENT OF CRC REG TO CASSETTE
                        ;       WHICH IS CHECKED FOR CORRECTNESS WHEN THE BLOCK IS READ
                        ; REG AX IS MODIFIED
                        ;--------------------------------------------------------------
F9D9  A1 0069 R                 MOV     AX,CRC_REG      ; WRITE THE ONE'S COMPLEMENT OF THE
                                                        ;  TWO BYTE CRC TO TAPE
F9DC  F7 D0                     NOT     AX              ; FOR 1'S COMPLEMENT
F9DE  50                        PUSH    AX              ; SAVE IT
F9DF  86 E0                     XCHG    AH,AL           ; WRITE MS BYTE FIRST
F9E1  E8 FA08 R                 CALL    WRITE_BYTE      ; WRITE IT
F9E4  58                        POP     AX              ; GET IT BACK
F9E5  E8 FA08 R                 CALL    WRITE_BYTE      ; NOW WRITE LS BYTE
F9E8  0B C9                     OR      CX,CX           ; IS BYTE COUNT EXHAUSTED?
F9EA  75 D7                     JNZ     WR_BLOCK        ; JUMP IF NOT DONE YET
F9EC  51                        PUSH    CX              ; SAVE REG CX
F9ED  FB                        STI                     ; RE-ENABLE INTERUPTS
F9EE  B9 0020                   MOV     CX, 32          ; WRITE OUT TRAILER BITS
F9F1                    W26:                            ; TRAIL-LOOP
F9F1  F9                        STC
F9F2  E8 FA1F R                 CALL    WRITE_BIT
F9F5  E2 FA                     LOOP    W26             ; WRITE UNTIL TRAILER WRITTEN
F9F7  59                        POP     CX              ; RESTORE REG CX
F9F8  B0 B0                     MOV     AL, 0B0H        ; TURN TIMER2 OFF
F9FA  E6 43                     OUT     TIM_CTL, AL
F9FC  B8 0001                   MOV     AX, 1
F9FF  E8 FA35 R                 CALL    W31             ; SET_TIMER
FA02  E8 F88A R                 CALL    MOTOR_OFF       ; TURN MOTOR OFF
FA05  2B C0                     SUB     AX,AX           ; NO ERRORS REPORTED ON WRITE OP
FA07  C3                        RET                     ; FINISHED
FA08                    WRITE_BLOCK     ENDP
```

```
                        ;--------------------------------
                        ; WRITE A BYTE TO CASSETTE.
                        ; BYTE TO WRITE IS IN REG AL.
                        ;--------------------------------
FA08                    WRITE_BYTE      PROC    NEAR
FA08  51                       PUSH    CX              ; SAVE REGS CX,AX
FA09  50                       PUSH    AX
FA0A  8A E8                    MOV     CH,AL           ; AL=BYTE TO WRITE.
                                                       ;   (MS BIT WRITTEN FIRST)
FA0C  B1 08                    MOV     CL,8            ; FOR 8 DATA BITS IN BYTE.
                                                       ;   NOTE: TWO EDGES PER BIT
FA0E                    W27:                           ; DISASSEMBLE THE DATA BIT
FA0E  D0 D5                    RCL     CH,1            ; ROTATE MS BIT INTO CARRY
FA10  9C                      PUSHF                    ; SAVE FLAGS.
                                                       ;   NOTE: DATA BIT IS IN CARRY
FA11  E8 FA1F R                CALL    WRITE_BIT       ; WRITE DATA BIT
FA14  9D                       POPF                    ; RESTORE CARRY FOR CRC CALC
FA15  E8 FA3C R                CALL    CRC_GEN         ; COMPUTE CRC ON DATA BIT
FA18  FE C9                    DEC     CL              ; LOOP TILL ALL 8 BITS DONE
FA1A  75 F2                    JNZ     W27             ; JUMP IF NOT DONE YET
FA1C  58                       POP     AX              ; RESTORE REGS AX,CX
FA1D  59                       POP     CX
FA1E  C3                       RET                     ; WE ARE FINISHED
FA1F                    WRITE_BYTE      ENDP
                        ;----------------------------------------
FA1F                    WRITE_BIT       PROC    NEAR
                        ; PURPOSE:
                        ;
                        ;   TO WRITE A DATA BIT TO CASSETTE
                        ;   CARRY FLAG CONTAINS DATA BIT
                        ;   I.E. IF SET   DATA BIT IS A ONE
                        ;        IF CLEAR DATA BIT IS A ZERO
                        ;
                        ;   NOTE: TWO EDGES ARE WRITTEN PER BIT
                        ;         ONE BIT  HAS 500 USEC BETWEEN EDGES
                        ;             FOR A 1000 USEC PERIOD (1 MILLISEC)
                        ;
                        ;          ZERO BIT HAS 250 USEC BETWEEN EDGES
                        ;              FOR A  500 USEC PERIOD (.5 MILLISEC)
                        ; CARRY FLAG IS DATA BIT
                        ;----------------------------------------
                                                       ;ASSUME IT'S A '1'
FA1F  B8 04A0                  MOV     AX,1184         ; SET AX TO NOMINAL ONE SIZE
FA22  72 03                    JC      W28             ; JUMP IF ONE BIT
FA24  B8 0250                  MOV     AX,592          ; NO, SET TO NOMINAL ZERO SIZE
FA27                    W28:                           ; WRITE-BIT-AX
FA27  50                       PUSH    AX              ;WRITE BIT WITH PERIOD EQ TO VALUE
                                                       ; AX
FA28  E4 62             W29:   IN      AL,PORT_C       ;INPUT TIMER_0 OUTPUT
FA2A  24 20                    AND     AL,020H
FA2C  74 FA                    JZ      W29             ;LOOP TILL HIGH
FA2E  E4 62             W30:   IN      AL,PORT_C       ;NOW WAIT TILL TIMER'S OUTPUT IS
                                                       ; LOW
FA30  24 20                    AND     AL,020H
FA32  75 FA                    JNZ     W30
                                                       ;RELOAD TIMER WITH PERIOD
                                                       ;FOR NEXT DATA BIT
FA34  58                       POP     AX              ;RESTORE PERIOD COUNT
FA35                    W31:                           ; SET TIMER
FA35  E6 42                    OUT     042H, AL        ; SET LOW BYTE OF TIMER 2
FA37  8A C4                    MOV     AL, AH
FA39  E6 42                    OUT     042H, AL        ; SET HIGH BYTE OF TIMER 2
FA3B  C3                       RET
FA3C                    WRITE_BIT       ENDP
                        ;----------------------------------------
FA3C                    CRC_GEN         PROC    NEAR
                        ;  UPDATE CRC REGISTER WITH NEXT DATA BIT
                        ;  CRC IS USED TO DETECT READ ERRORS
                        ;  ASSUMES DATA BIT IS IN CARRY
                        ;  REG AX IS MODIFIED
                        ;  FLAGS ARE MODIFIED
                        ;----------------------------------------
FA3C  A1 0069 R                MOV     AX,CRC_REG
                                                       ;THE FOLLOWING INSTUCTIONS
                                                       ;WILL SET THE OVERFLOW FLAG
                                                       ;IF CARRY AND MS BIT OF CRC
                                                       ;ARE UNEQUAL
FA3F  D1 D8                    RCR     AX,1
FA41  D1 D0                    RCL     AX,1
FA43  F8                       CLC                     ;CLEAR CARRY
FA44  71 04                    JNO     W32             ;SKIP IF NO OVERFLOW
                                                       ;IF DATA BIT XORED WITH
                                                       ; CRC REG BIT 15 IS ONE
FA46  35 0810                  XOR     AX,0810H        ;THEN XOR CRC REG WITH
                                                       ; 0810H
FA49  F9                       STC                     ;SET CARRY
FA4A  D1 D0             W32:   RCL     AX,1            ;ROTATE CARRY (DATA BIT)
                                                       ;INTO CRC REG
FA4C  A3 0069 R                MOV     CRC_REG,AX      ;UPDATE CRC_REG
FA4F  C3                       RET                     ;FINISHED
FA50                    CRC_GEN         ENDP
```

```
                                ;--------------------------------------------------------------------------
FA50                            BEGIN_OP        PROC    NEAR    ; START TAPE AND DELAY
FA50  E8 F881 R                         CALL    MOTOR_ON        ;TURN ON MOTOR
FA53  B3 42                             MOV     BL,42H          ;DELAY FOR TAPE DRIVE
                                                                ;TO GET UP TO SPEED  (1/2 SEC)
FA55  B9 0700                   W33:    MOV     CX,700H         ;INNER LOOP= APPROX.  10 MILLISEC
FA58  E2 FE                     W34:    LOOP    W34
FA5A  FE CB                             DEC     BL
FA5C  75 F7                             JNZ     W33
FA5E  C3                                RET
FA5F                            BEGIN_OP        ENDP
                                ;------ CARRIAGE RETURN, LINE FEED SUBROUTINE
FA5F                            CRLF    PROC    NEAR
FA5F  33 D2                             XOR     DX,DX           ; PRINTER 0
FA61  32 E4                             XOR     AH,AH           ; WILL NOW SEND INITIAL LF,CR TO
                                                                ; PRINTER
FA63  B0 0D                             MOV     AL,0DH          ; CR
FA65  CD 17                             INT     17H             ; SEND THE LINE FEED
FA67  32 E4                             XOR     AH,AH           ; NOW FOR THE CR
FA69  B0 0A                             MOV     AL,0AH          ; LF
FA6B  CD 17                             INT     17H             ; SEND THE CARRIAGE RETURN
FA6D  C3                                RET
FA6E                            CRLF    ENDP
                                ;--------------------------------------------------------------------------
                                ;       CHARACTER GENERATOR GRAPHICS FOR 320X200 AND 640X200
                                ;       GRAPHICS FOR CHARACTERS 00H THRU 7FH
                                ;--------------------------------------------------------------------------
FA6E                                    ORG     0FA6EH
FA6E                            CRT_CHAR_GEN    LABEL   BYTE
FA6E  00 00 00 00 00 00                 DB      000H,000H,000H,000H,000H,000H,000H,000H ; D_00
      00 00
FA76  7E 81 A5 81 BD 99                 DB      07EH,081H,0A5H,081H,0BDH,099H,081H,07EH ; D_01
      81 7E
FA7E  7E FF DB FF C3 E7                 DB      07EH,0FFH,0DBH,0FFH,0C3H,0E7H,0FFH,07EH ; D_02
      FF 7E
FA86  6C FE FE FE 7C 38                 DB      06CH,0FEH,0FEH,0FEH,07CH,038H,010H,000H ; D_03
      10 00
FA8E  10 38 7C FE 7C 38                 DB      010H,038H,07CH,0FEH,07CH,038H,010H,000H ; D_04
      10 00
FA96  38 7C 38 FE FE 7C                 DB      038H,07CH,038H,0FEH,0FEH,07CH,038H,07CH ; D_05
      38 7C
FA9E  10 10 38 7C FE 7C                 DB      010H,010H,038H,07CH,0FEH,07CH,038H,07CH ; D_06
      38 7C
FAA6  00 00 18 3C 3C 18                 DB      000H,000H,018H,03CH,03CH,018H,000H,000H ; D_07
      00 00
FAAE  FF FF E7 C3 C3 E7                 DB      0FFH,0FFH,0E7H,0C3H,0C3H,0E7H,0FFH,0FFH ; D_08
      FF FF
FAB6  00 3C 66 42 42 66                 DB      000H,03CH,066H,042H,042H,066H,03CH,000H ; D_09
      3C 00
FABE  FF C3 99 BD BD 99                 DB      0FFH,0C3H,099H,0BDH,0BDH,099H,0C3H,0FFH ; D_0A
      C3 FF
FAC6  0F 07 0F 7D CC CC                 DB      00FH,007H,00FH,07DH,0CCH,0CCH,0CCH,078H ; D_0B
      CC 78
FACE  3C 66 66 66 3C 18                 DB      03CH,066H,066H,066H,03CH,018H,07EH,018H ; D_0C
      7E 18
FAD6  3F 33 3F 30 30 70                 DB      03FH,033H,03FH,030H,030H,070H,0F0H,0E0H ; D_0D
      F0 E0
FADE  7F 63 7F 63 63 67                 DB      07FH,063H,07FH,063H,063H,067H,0E6H,0C0H ; D_0E
      E6 C0
FAE6  99 5A 3C E7 E7 3C                 DB      099H,05AH,03CH,0E7H,0E7H,03CH,05AH,099H ; D_0F
      5A 99

FAEE  80 E0 F8 FE F8 E0                 DB      080H,0E0H,0F8H,0FEH,0F8H,0E0H,080H,000H ; D_10
      80 00
FAF6  02 0E 3E FE 3E 0E                 DB      002H,00EH,03EH,0FEH,03EH,00EH,002H,000H ; D_11
      02 00
FAFE  18 3C 7E 18 18 7E                 DB      018H,03CH,07EH,018H,018H,07EH,03CH,018H ; D_12
      3C 18
FB06  66 66 66 66 66 00                 DB      066H,066H,066H,066H,066H,000H,066H,000H ; D_13
      66 00
FB0E  7F DB DB 7B 1B 1B                 DB      07FH,0DBH,0DBH,07BH,01BH,01BH,01BH,000H ; D_14
      1B 00
FB16  3E 63 38 6C 6C 38                 DB      03EH,063H,038H,06CH,06CH,038H,0CCH,078H ; D_15
      CC 78
FB1E  00 00 00 00 7E 7E                 DB      000H,000H,000H,000H,07EH,07EH,07EH,000H ; D_16
      7E 00
FB26  18 3C 7E 18 7E 3C                 DB      018H,03CH,07EH,018H,07EH,03CH,018H,0FFH ; D_17
      18 FF
FB2E  18 3C 7E 18 18 18                 DB      018H,03CH,07EH,018H,018H,018H,018H,000H ; D_18
      18 00
FB36  18 18 18 18 7E 3C                 DB      018H,018H,018H,018H,07EH,03CH,018H,000H ; D_19
      18 00
FB3E  00 18 0C FE 0C 18                 DB      000H,018H,00CH,0FEH,00CH,018H,000H,000H ; D_1A
      00 00
FB46  00 30 60 FE 60 30                 DB      000H,030H,060H,0FEH,060H,030H,000H,000H ; D_1B
      00 00
FB4E  00 00 C0 C0 C0 FE                 DB      000H,000H,0C0H,0C0H,0C0H,0FEH,000H,000H ; D_1C
      00 00
FB56  00 24 66 FF 66 24                 DB      000H,024H,066H,0FFH,066H,024H,000H,000H ; D_1D
      00 00
FB5E  00 18 3C 7E FF FF                 DB      000H,018H,03CH,07EH,0FFH,0FFH,000H,000H ; D_1E
      00 00
FB66  00 FF FF 7E 3C 18                 DB      000H,0FFH,0FFH,07EH,03CH,018H,000H,000H ; D_1F
      00 00
```

```
FB6E  00 00 00 00 00 00 00 00          DB      000H,000H,000H,000H,000H,000H,000H,000H ; SP D_20
FB76  30 78 78 30 30 00 30 00          DB      030H,078H,078H,030H,030H,000H,030H,000H ; ! D_21
FB7E  6C 6C 6C 00 00 00 00 00          DB      06CH,06CH,06CH,000H,000H,000H,000H,000H ; " D_22
FB86  6C 6C FE 6C FE 6C 6C 00          DB      06CH,06CH,0FEH,06CH,0FEH,06CH,06CH,000H ; # D_23
FB8E  30 7C C0 78 0C F8 30 00          DB      030H,07CH,0C0H,078H,00CH,0F8H,030H,000H ; $ D_24
FB96  00 C6 CC 18 30 66 C6 00          DB      000H,0C6H,0CCH,018H,030H,066H,0C6H,000H ;
                                                       ; PER CENT D_25
FB9E  38 6C 38 76 DC CC 76 00          DB      038H,06CH,038H,076H,0DCH,0CCH,076H,000H ; & D_26
FBA6  60 60 C0 00 00 00 00 00          DB      060H,060H,0C0H,000H,000H,000H,000H,000H ; ' D_27
FBAE  18 30 60 60 60 30 18 00          DB      018H,030H,060H,060H,060H,030H,018H,000H ; ( D_28
FBB6  60 30 18 18 18 30 60 00          DB      060H,030H,018H,018H,018H,030H,060H,000H ; ) D_29
FBBE  00 66 3C FF 3C 66 00 00          DB      000H,066H,03CH,0FFH,03CH,066H,000H,000H ; * D_2A
FBC6  00 30 30 FC 30 30 00 00          DB      000H,030H,030H,0FCH,030H,030H,000H,000H ; + D_2B
FBCE  00 00 00 00 00 30 30 60          DB      000H,000H,000H,000H,000H,030H,030H,060H ; , D_2C
FBD6  00 00 00 FC 00 00 00 00          DB      000H,000H,000H,0FCH,000H,000H,000H,000H ;- D_2D
FBDE  00 00 00 00 00 30 30 00          DB      000H,000H,000H,000H,000H,030H,030H,000H ;   D_2E
FBE6  06 0C 18 30 60 C0 80 00          DB      006H,00CH,018H,030H,060H,0C0H,080H,000H ; / D_2F

FBEE  7C C6 CE DE F6 E6 7C 00          DB      07CH,0C6H,0CEH,0DEH,0F6H,0E6H,07CH,000H ; 0 D_30
FBF6  30 70 30 30 30 30 FC 00          DB      030H,070H,030H,030H,030H,030H,0FCH,000H ; 1 D_31
FBFE  78 CC 0C 38 60 CC FC 00          DB      078H,0CCH,00CH,038H,060H,0CCH,0FCH,000H ; 2 D_32
FC06  78 CC 0C 38 0C CC 78 00          DB      078H,0CCH,00CH,038H,00CH,0CCH,078H,000H ; 3 D_33
FC0E  1C 3C 6C CC FE 0C 1E 00          DB      01CH,03CH,06CH,0CCH,0FEH,00CH,01EH,000H ; 4 D_34
FC16  FC C0 F8 0C 0C CC 78 00          DB      0FCH,0C0H,0F8H,00CH,00CH,0CCH,078H,000H ; 5 D_35
FC1E  38 60 C0 F8 CC CC 78 00          DB      038H,060H,0C0H,0F8H,0CCH,0CCH,078H,000H ; 6 D_36
FC26  FC CC 0C 18 30 30 30 00          DB      0FCH,0CCH,00CH,018H,030H,030H,030H,000H ; 7 D_37
FC2E  78 CC CC 78 CC CC 78 00          DB      078H,0CCH,0CCH,078H,0CCH,0CCH,078H,000H ; 8 D_38
FC36  78 CC CC 7C 0C 18 70 00          DB      078H,0CCH,0CCH,07CH,00CH,018H,070H,000H ; 9 D_39
FC3E  00 30 30 00 00 30 30 00          DB      000H,030H,030H,000H,000H,030H,030H,000H ; : D_3A
FC46  00 30 30 00 00 30 30 60          DB      000H,030H,030H,000H,000H,030H,030H,060H ; ; D_3B
FC4E  18 30 60 C0 60 30 18 00          DB      018H,030H,060H,0C0H,060H,030H,018H,000H ; < D_3C
FC56  00 00 FC 00 00 FC 00 00          DB      000H,000H,0FCH,000H,000H,0FCH,000H,000H ; = D_3D
FC5E  60 30 18 0C 18 30 60 00          DB      060H,030H,018H,00CH,018H,030H,060H,000H ; > D_3E
FC66  78 CC 0C 18 30 00 30 00          DB      078H,0CCH,00CH,018H,030H,000H,030H,000H ; ? D_3F

FC6E  7C C6 DE DE DE C0 78 00          DB      07CH,0C6H,0DEH,0DEH,0DEH,0C0H,078H,000H ; @ D_40
FC76  30 78 CC CC FC CC CC 00          DB      030H,078H,0CCH,0CCH,0FCH,0CCH,0CCH,000H ; A D_41
FC7E  FC 66 66 7C 66 66 FC 00          DB      0FCH,066H,066H,07CH,066H,066H,0FCH,000H ; B D_42
FC86  3C 66 C0 C0 C0 66 3C 00          DB      03CH,066H,0C0H,0C0H,0C0H,066H,03CH,000H ; C D_43
FC8E  F8 6C 66 66 66 6C F8 00          DB      0F8H,06CH,066H,066H,066H,06CH,0F8H,000H ; D D_44
FC96  FE 62 68 78 68 62 FE 00          DB      0FEH,062H,068H,078H,068H,062H,0FEH,000H ; E D_45
FC9E  FE 62 68 78 68 60 F0 00          DB      0FEH,062H,068H,078H,068H,060H,0F0H,000H ; F D_46
FCA6  3C 66 C0 C0 CE 66 3E 00          DB      03CH,066H,0C0H,0C0H,0CEH,066H,03EH,000H ; G D_47
FCAE  CC CC CC FC CC CC CC 00          DB      0CCH,0CCH,0CCH,0FCH,0CCH,0CCH,0CCH,000H ; H D_48
FCB6  78 30 30 30 30 30 78 00          DB      078H,030H,030H,030H,030H,030H,078H,000H ; I D_49
FCBE  1E 0C 0C 0C CC CC 78 00          DB      01EH,00CH,00CH,00CH,0CCH,0CCH,078H,000H ; J D_4A
FCC6  E6 66 6C 78 6C 66 E6 00          DB      0E6H,066H,06CH,078H,06CH,066H,0E6H,000H ; K D_4B
FCCE  F0 60 60 60 62 66 FE 00          DB      0F0H,060H,060H,060H,062H,066H,0FEH,000H ; L D_4C
FCD6  C6 EE FE FE D6 C6 C6 00          DB      0C6H,0EEH,0FEH,0FEH,0D6H,0C6H,0C6H,000H ; M D_4D
FCDE  C6 E6 F6 DE CE C6 C6 00          DB      0C6H,0E6H,0F6H,0DEH,0CEH,0C6H,0C6H,000H ; N D_4E
FCE6  38 6C C6 C6 C6 6C 38 00          DB      038H,06CH,0C6H,0C6H,0C6H,06CH,038H,000H ; O D_4F
```

```
FCEE  FC 66 66 7C 60 60          DB      0FCH,066H,066H,07CH,060H,060H,0F0H,000H ; P D_50
      F0 00
FCF6  78 CC CC CC DC 78          DB      078H,0CCH,0CCH,0CCH,0DCH,078H,01CH,000H ; Q D_51
      1C 00
FCFE  FC 66 66 7C 6C 66          DB      0FCH,066H,066H,07CH,06CH,066H,0E6H,000H ; R D_52
      E6 00
FD06  78 CC E0 70 1C CC          DB      078H,0CCH,0E0H,070H,01CH,0CCH,078H,000H ; S D_53
      78 00
FD0E  FC B4 30 30 30 30          DB      0FCH,0B4H,030H,030H,030H,030H,078H,000H ; T D_54
      78 00
FD16  CC CC CC CC CC CC          DB      0CCH,0CCH,0CCH,0CCH,0CCH,0CCH,0FCH,000H ; U D_55
      FC 00
FD1E  CC CC CC CC CC 78          DB      0CCH,0CCH,0CCH,0CCH,0CCH,078H,030H,000H ; V D_56
      30 00
FD26  C6 C6 C6 D6 FE EE          DB      0C6H,0C6H,0C6H,0D6H,0FEH,0EEH,0C6H,000H ; W D_57
      C6 00
FD2E  C6 C6 6C 38 38 6C          DB      0C6H,0C6H,06CH,038H,038H,06CH,0C6H,000H ; X D_58
      C6 00
FD36  CC CC CC 78 30 30          DB      0CCH,0CCH,0CCH,078H,030H,030H,078H,000H ; Y D_59
      78 00
FD3E  FE C6 8C 18 32 66          DB      0FEH,0C6H,08CH,018H,032H,066H,0FEH,000H ; Z D_5A
      FE 00
FD46  78 60 60 60 60 60          DB      078H,060H,060H,060H,060H,060H,078H,000H ; [ D_5B
      78 00
FD4E  C0 60 30 18 0C 06          DB      0C0H,060H,030H,018H,00CH,006H,002H,000H ;
      02 00
                                                ; BACKSLASH D_5C
FD56  78 18 18 18 18 18          DB      078H,018H,018H,018H,018H,018H,078H,000H ; ] D_5D
      78 00
FD5E  10 38 6C C6 00 00          DB      010H,038H,06CH,0C6H,000H,000H,000H,000H ;
      00 00
                                                ; CIRCUMFLEX D_5E
FD66  00 00 00 00 00 00          DB      000H,000H,000H,000H,000H,000H,000H,0FFH ; _ D_5F
      00 FF

FD6E  30 30 18 00 00 00          DB      030H,030H,018H,000H,000H,000H,000H,000H ; ` D_60
      00 00
FD76  00 00 78 0C 7C CC          DB      000H,000H,078H,00CH,07CH,0CCH,076H,000H ;
      76 00
                                                ; LOWER CASE A D_61
FD7E  E0 60 60 7C 66 66          DB      0E0H,060H,060H,07CH,066H,066H,0DCH,000H ; LC B D_62
      DC 00
FD86  00 00 78 CC C0 CC          DB      000H,000H,078H,0CCH,0C0H,0CCH,078H,000H ; LC C D_63
      78 00
FD8E  1C 0C 0C 7C CC CC          DB      01CH,00CH,00CH,07CH,0CCH,0CCH,076H,000H ; LC D D_64
      76 00
FD96  00 00 78 CC FC C0          DB      000H,000H,078H,0CCH,0FCH,0C0H,078H,000H ; LC E D_65
      78 00
FD9E  38 6C 60 F0 60 60          DB      038H,06CH,060H,0F0H,060H,060H,0F0H,000H ; LC F D_66
      F0 00
FDA6  00 00 76 CC CC 7C          DB      000H,000H,076H,0CCH,0CCH,07CH,00CH,0F8H ; LC G D_67
      0C F8
FDAE  E0 60 6C 76 66 66          DB      0E0H,060H,06CH,076H,066H,066H,0E6H,000H ; LC H D_68
      E6 00
FDB6  30 00 70 30 30 30          DB      030H,000H,070H,030H,030H,030H,078H,000H ; LC I D_69
      78 00
FDBE  0C 00 0C 0C 0C CC          DB      00CH,000H,00CH,00CH,00CH,0CCH,0CCH,078H ; LC J D_6A
      CC 78
FDC6  E0 60 66 6C 78 6C          DB      0E0H,060H,066H,06CH,078H,06CH,0E6H,000H ; LC K D_6B
      E6 00
FDCE  70 30 30 30 30 30          DB      070H,030H,030H,030H,030H,030H,078H,000H ; LC L D_6C
      78 00
FDD6  00 00 CC FE FE D6          DB      000H,000H,0CCH,0FEH,0FEH,0D6H,0C6H,000H ; LC M D_6D
      C6 00
FDDE  00 00 F8 CC CC CC          DB      000H,000H,0F8H,0CCH,0CCH,0CCH,0CCH,000H ; LC N D_6E
      CC 00
FDE6  00 00 78 CC CC CC          DB      000H,000H,078H,0CCH,0CCH,0CCH,078H,000H ; LC O D_6F
      78 00

FDEE  00 00 DC 66 66 7C          DB      000H,000H,0DCH,066H,066H,07CH,060H,0F0H ; LC P D_70
      60 F0
FDF6  00 00 76 CC CC 7C          DB      000H,000H,076H,0CCH,0CCH,07CH,00CH,01EH ; LC Q D_71
      0C 1E
FDFE  00 00 DC 76 66 60          DB      000H,000H,0DCH,076H,066H,060H,0F0H,000H ; LC R D_72
      F0 00
FE06  00 00 7C C0 78 0C          DB      000H,000H,07CH,0C0H,078H,00CH,0F8H,000H ; LC S D_73
      F8 00
FE0E  10 30 7C 30 30 34          DB      010H,030H,07CH,030H,030H,034H,018H,000H ; LC T D_74
      18 00
FE16  00 00 CC CC CC CC          DB      000H,000H,0CCH,0CCH,0CCH,0CCH,076H,000H ; LC U D_75
      76 00
FE1E  00 00 CC CC CC 78          DB      000H,000H,0CCH,0CCH,0CCH,078H,030H,000H ; LC V D_76
      30 00
FE26  00 00 C6 D6 FE FE          DB      000H,000H,0C6H,0D6H,0FEH,0FEH,06CH,000H ; LC W D_77
      6C 00
FE2E  00 00 C6 6C 38 6C          DB      000H,000H,0C6H,06CH,038H,06CH,0C6H,000H ; LC X D_78
      C6 00
FE36  00 00 CC CC CC 7C          DB      000H,000H,0CCH,0CCH,0CCH,07CH,00CH,0F8H ; LC Y D_79
      0C F8
FE3E  00 00 FC 98 30 64          DB      000H,000H,0FCH,098H,030H,064H,0FCH,000H ; LC Z D_7A
      FC 00
FE46  1C 30 30 E0 30 30          DB      01CH,030H,030H,0E0H,030H,030H,01CH,000H ; { D_7B
      1C 00
FE4E  18 18 18 00 18 18          DB      018H,018H,018H,000H,018H,018H,018H,000H ; : D_7C
      18 00
FE56  E0 30 30 1C 30 30          DB      0E0H,030H,030H,01CH,030H,030H,0E0H,000H ; } D_7D
      E0 00
FE5E  76 DC 00 00 00 00          DB      076H,0DCH,000H,000H,000H,000H,000H,000H ; ~ D_7E
      00 00
FE66  00 10 38 6C C6 C6          DB      000H,010H,038H,06CH,0C6H,0C6H,0FEH,000H ;
      FE 00
                                                ; DELTA D_7F
```

```
FE6E                                    ORG     OFE6EH
FE6E  E9 1393 R                         JMP     NEAR PTR TIME_OF_DAY
                                ;--------------------------------------------------------
                                ;               CRC CHECK/GENERATION ROUTINE
                                ;       ROUTINE TO CHECK A ROM MODULE USING THE POLYNOMIMAL:
                                ;                 X16 + X12 + X5 + 1
                                ;       CALLING PARAMETERS:
                                ;       DS = DATA SEGMENT OF ROM SPACE TO BE CHECKED
                                ;       SI = INDEX OFFSET INTO DS POINTING TO 1ST BYTE
                                ;       CX = LENGTH OF SPACE TO BE CHECKED (INCLUDING CRC BYTES)
                                ;       ON EXIT:
                                ;       ZERO FLAG = SET = CRC CHECKED OK
                                ;       AH = 00
                                ;       AL = ??
                                ;       BX = 0000
                                ;       CL = 04
                                ;       DX = 0000 IF CRC CHECKED OK, ELSE, ACCUMULATED CRC
                                ;       SI = (SI(ENTRY)+BX(ENTRY)
                                ;       NOTE: ROUTINE WILL RETURN IMMEDIATLY IF "RESET_FLAG
                                ;             IS EQUAL TO "1234H" (WARM START)
                                ;--------------------------------------------------------
                                ;
FE71                            CRC_CHECK       PROC    NEAR
                                        ASSUME  DS:NOTHING
FE71  8B D9                             MOV     BX,CX                   ; SAVE COUNT
FE73  BA FFFF                           MOV     DX,OFFFFH               ; INIT. ENCODE REGISTER
FE76  FC                                CLD                             ; SET DIR FLAG TO INCREMENT
FE77  32 E4                             XOR     AH,AH                   ; INIT. WORK REG HIGH
FE79  B1 04                             MOV     CL,4                    ; SET ROTATE COUNT
FE7B  AC                        CRC_1:  LODSB                           ; GET A BYTE
FE7C  32 F0                             XOR     DH,AL                   ; FORM Aj + Cj + 1
FE7E  8A C6                             MOV     AL,DH
FE80  D3 C0                             ROL     AX,CL                   ; SHIFT WORK REG BACK 4
FE82  33 D0                             XOR     DX,AX                   ; ADD INTO RESULT REG
FE84  D1 C0                             ROL     AX,1                    ; SHIFT WORK REG BACK 1
FE86  86 F2                             XCHG    DH,DL                   ; SWAP PARTIAL SUM INTO RESULT REG
FE88  33 D0                             XOR     DX,AX                   ; ADD WORK REG INTO RESULTS
FE8A  D3 C8                             ROR     AX,CL                   ; SHIFT WORK REG OVER 4
FE8C  24 E0                             AND     AL,11100000B            ; CLEAR OFF (EFGH)
FE8E  33 D0                             XOR     DX,AX                   ; ADD (ABCD) INTO RESULTS
FE90  D1 C8                             ROR     AX,1                    ; SHIFT WORK REG ON OVER (AH=0 FOR
                                                                        ; NEXT PASS)
FE92  32 F0                             XOR     DH,AL                   ; ADD (ABCD INTO RESULTS LOW)
FE94  4B                                DEC     BX                      ; DECREMENT COUNT
FE95  75 E4                             JNZ     CRC_1                   ; LOOP TILL COUNT = 0000
FE97  0B D2                             OR      DX,DX                   ; DX S/B = 0000 IF O.K.
FE99  C3                                RET                             ; RETURN TO CALLER
FE9A                            CRC_CHECK       ENDP
                                ;--------------------------------------------------------
                                ;   SUBROUTINE TO READ AN 8250 REGISTER.  MAY ALSO BUMP ERROR
                                ;       REPORTER (BL) AND/OR REG DX (PORT ADDRESS) DEPENDING ON
                                ;       WHICH ENTRY POINT IS CHOSEN.
                                ;   THIS SUBROUTINE WAS WRITTEN TO AVOID MULTIPLE USE OF I/O TIME
                                ;       DELAYS FOR THE 8250.  IT WAS THE MOST EFFICIENT WAY TO
                                ;       INCLUDE THE DELAYS.
                                ;   IN EVERY CASE, UPON RETURN, REG AL WILL CONTAIN THE CONTENTS OF
                                ;       PORT(DX)
                                ;--------------------------------------------------------
FE9A                            RR1     PROC    NEAR
FE9A  32 C0                             XOR     AL,AL
FE9C  EE                                OUT     DX,AL                   ; DISABLE ALL INTERRUPTS
FE9D  FE C3                             INC     BL                      ; BUMP ERROR REPORTER
FE9F  42                        RR2:    INC     DX                      ; INCR PORT ADDR
FEA0  EC                        RR3:    IN      AL,DX                   ; READ REGISTER
FEA1  C3                                RET
FEA2                            RR1     ENDP
                                ;-----------------------------------------------
                                ; THIS ROUTINE HANDLES THE TIMER INTERRUPT FROM
                                ; CHANNEL 0 OF THE 8253 TIMER.  INPUT FREQUENCY IS 1.19318 MHZ
                                ; AND THE DIVISOR IS 65536, RESULTING IN APPROX. 18.2 INTERRUPTS
                                ; EVERY SECOND.
                                ;
                                ; THE INTERRUPT HANDLER MAINTAINS A COUNT OF INTERRUPTS SINCE POWER
                                ;  ON TIME, WHICH MAY BE USED TO ESTABLISH TIME OF DAY.
                                ;  INTERRUPTS MISSED WHILE INTS. WERE DISABLED ARE TAKEN CARE OF
                                ;  BY THE USE OF TIMER 1 AS A OVERFLOW COUNTER
                                ; THE INTERRUPT HANDLER ALSO DECREMENTS THE MOTOR CONTROL COUNT
                                ;  OF THE DISKETTE, AND WHEN IT EXPIRES, WILL TURN OFF THE DISKETTE
                                ;  MOTOR, AND RESET THE MOTOR RUNNING FLAGS
                                ; THE INTERRUPT HANDLER WILL ALSO INVOKE A USER ROUTINE THROUGH
                                ;  INTERRUPT 1CH AT EVERY TIME TICK.  THE USER MUST CODE A ROUTINE
                                ; AND PLACE THE CORRECT ADDRESS IN THE VECTOR TABLE.
                                ;-----------------------------------------------
FEA5                                    ORG     OFEA5H
                                        ASSUME  DS:DATA
FEA5                            TIMER_INT       PROC    FAR
FEA5  FB                                STI                             ; INTERRUPTS BACK ON
FEA6  1E                                PUSH    DS
FEA7  50                                PUSH    AX
FEA8  52                                PUSH    DX                      ; SAVE MACHINE STATE
FEA9  E8 1388 R                         CALL    DDS
FEAC  FF 06 006C R                      INC     TIMER_LOW               ; INCREMENT TIME
FEB0  75 04                             JNZ     T4                      ; TEST_DAY
FEB2  FF 06 006E R                      INC     TIMER_HIGH              ; INCREMENT HIGH WORD OF TIME
FEB6                            T4:                                     ; TEST_DAY
FEB6  83 3E 006E R 18                   CMP     TIMER_HIGH,018H ; TEST FOR COUNT EQUALLING 24 HOURS
FEBB  75 15                             JNZ     T5                      ; DISKETTE_CTL
FEBD  81 3E 006C R 00B0                 CMP     TIMER_LOW,0B0H
FEC3  75 0D                             JNZ     T5                      ; DISKETTE_CTL
```

```
                        ;------ TIMER HAS GONE 24 HOURS
FEC5  2B C0                     SUB     AX,AX
FEC7  A3 006E R                 MOV     TIMER_HIGH,AX
FECA  A3 006C R                 MOV     TIMER_LOW,AX
FECD  C6 06 0070 R 01           MOV     TIMER_OFL,1
                        ;------ TEST FOR DISKETTE TIME OUT
FED2                    T5:                             ; LOOP TILL ALL OVERFLOWS TAKEN
                                                        ; CARE OF
FED2  FE 0E 0040 R              DEC     MOTOR_COUNT
FED6  75 09                     JNZ     T6              ; RETURN IF COUNT NOT OUT
FED8  80 26 003F R F0           AND     MOTOR_STATUS,0F0H ; TURN OFF MOTOR RUNNING BITS
FEDD  B0 80                     MOV     AL,FDC_RESET    ; TURN OFF MOTOR, DO NOT RESET FDC
FEDF  E6 F2                     OUT     NEC_CTL,AL      ; TURN OFF THE MOTOR
FEE1  CD 1C             T6:     INT     1CH             ; TRANSFER CONTROL TO A USER
                                                        ; ROUTINE
FEE3  B0 20                     MOV     AL,EOI
FEE5  E6 20                     OUT     020H,AL         ; END OF INTERRUPT TO 8259
FEE7  5A                        POP     DX
FEE8  58                        POP     AX
FEE9  1F                        POP     DS              ; RESET MACHINE STATE
FEEA  CF                        IRET                    ; RETURN FROM INTERRUPT
FEEB                    TIMER_INT       ENDP
                        ;--------------------------------------------------------------
                        ;       ARITHMETIC CHECKSUM  ROUTINE
                        ;       ENTRY:
                        ;       DS = DATA SEGMENT OF ROM SPACE TO BE CHECKED
                        ;       SI = INDEX OFFSET INTO DS POINTING TO 1ST BYTE
                        ;       CX = LENGTH OF SPACE TO BE CHECKED
                        ;       EXIT: ZERO FLAG OFF=ERROR, ON= SPACE CHECKED OK
                        ;--------------------------------------------------------------
FEEB                    ROS_CHECKSUM    PROC    NEAR
FEEB  02 04             RC_0:   ADD     AL,DS:[SI]
FEED  46                        INC     SI
FEEE  E2 FB                     LOOP    RC_0
FEF0  0A C0                     OR      AL,AL
FEF2  C3                        RET
FEF3                    ROS_CHECKSUM    ENDP
                        ;-----------------------------------------------
                        ; THESE ARE THE VECTORS WHICH ARE MOVED INTO    :
                        ; THE 8086 INTERRUPT AREA DURING POWER ON.      :
                        ; ONLY THE OFFSETS ARE DISPLAYED HERE, CODE     :
                        ; SEGMENT WILL BE ADDED FOR ALL OF THEM, EXCEPT :
                        ; WHERE NOTED.                                  :
                        ;-----------------------------------------------
                                ASSUME  CS:CODE
FEF3                            ORG     0FEF3H
FEF3                    VECTOR_TABLE    LABEL   WORD ; VECTOR TABLE FOR MOVE TO INTERRUPTS
FEF3  FEA5 R                    DW      OFFSET TIMER_INT ; INTERRUPT 8
FEF5  1561 R                    DW      OFFSET  KB_INT  ; INTERRUPT 9
FEF7  F815 R                    DW      OFFSET  D11     ; INTERRUPT A
FEF9  F815 R                    DW      OFFSET  D11     ; INTERRUPT B
FEFB  F815 R                    DW      OFFSET  D11     ; INTERRUPT C
FEFD  F815 R                    DW      OFFSET  D11     ; INTERRUPT D
FEFF  EF57 R                    DW      OFFSET DISK_INT ; INTERRUPT E
FF01  F815 R                    DW      OFFSET  D11     ; INTERRUPT F
FF03  0D0B R                    DW      OFFSET VIDEO_IO ; INTERRUPT 10H
FF05  F84D R                    DW      OFFSET EQUIPMENT ; INTERRUPT 11H
FF07  F841 R                    DW      OFFSET MEMORY_SIZE_DETERMINE ; INTERRUPT 12H
FF09  EC59 R                    DW      OFFSET DISKETTE_IO ; INTERRUPT 13H
FF0B  E739 R                    DW      OFFSET RS232_IO ; INTERRUPT 14H
FF0D  F859 R                    DW      CASSETTE_IO     ; INTERRUPT 15H
FF0F  13DD R                    DW      OFFSET KEYBOARD_IO ; INTERRUPT 16H
FF11  EFD2 R                    DW      OFFSET PRINTER_IO ; INTERRUPT 17H
FF13  0000                      DW      00000H          ; INTERRUPT 18H
                                DW      0F600H          ; MUST BE INSERTED INTO TABLE LATER
FF15  0B1B R                    DW      OFFSET BOOT_STRAP ; INTERRUPT 19H
FF17  1393 R                    DW      TIME_OF_DAY     ; INTERRUPT 1AH -- TIME OF DAY
FF19  F83C R                    DW      DUMMY_RETURN    ; INTERRUPT 1BH -- KEYBD BREAK ADDR
FF1B  F83C R                    DW      DUMMY_RETURN    ; INTERRUPT 1C -- TIMER BREAK ADDR
FF1D  F0A4 R                    DW      VIDEO_PARMS     ; INTERRUPT 1D -- VIDEO PARAMETERS
FF1F  EFC7 R                    DW      OFFSET DISK_BASE ; INTERRUPT 1E -- DISK PARMS
FF21  E05E R                    DW      CRT_CHARH       ; INTERRUPT 1F --  VIDEO EXT
FF23                    P_MSG   PROC    NEAR
FF23  2E: 8A 04         G12A:   MOV     AL,CS:[SI]      ; PUT CHAR IN AL
FF26  46                        INC     SI              ; POINT TO NEXT CHAR
FF27  50                        PUSH    AX              ; SAVE PRINT CHAR
FF28  E8 18BA R                 CALL    PRT_HEX         ; CALL VIDEO_IO
FF2B  58                        POP     AX              ; RECOVER PRINT CHAR
FF2C  3C 0D                     CMP     AL,13           ; WAS IT CARRAGE RETURN?
FF2E  75 F3                     JNE     G12A            ; NO,KEEP PRINTING STRING
FF30  C3                        RET
FF31                    P_MSG   ENDP
                        ;       ROUTINE TO SOUND BEEPER
FF31                    BEEP    PROC    NEAR
FF31  B0 B6                     MOV     AL,10110110B    ; SEL TIM 2,LSB,MSB,BINARY
FF33  E6 43                     OUT     TIMER+3,AL      ; WRITE THE TIMER MODE REG
FF35  B8 0533                   MOV     AX,533H         ; DIVISOR FOR 1000 HZ
FF38  E6 42                     OUT     TIMER+2,AL      ; WRITE TIMER 2 CNT - LSB
FF3A  8A C4                     MOV     AL,AH
FF3C  E6 42                     OUT     TIMER+2,AL      ; WRITE TIMER 2 CNT - MSB
FF3E  E4 61                     IN      AL,PORT_B       ; GET CURRENT SETTING OF PORT
FF40  8A E0                     MOV     AH,AL           ; SAVE THAT SETTING
FF42  0C 03                     OR      AL,03           ; TURN SPEAKER ON
FF44  E6 61                     OUT     PORT_B,AL
FF46  2B C9                     SUB     CX,CX           ; SET CNT TO WAIT 500 MS
FF48  E2 FE             G7:     LOOP    G7              ; DELAY BEFORE TURNING OFF
FF4A  FE CB                     DEC     BL              ; DELAY CNT EXPIRED?
FF4C  75 FA                     JNZ     G7              ; NO - CONTINUE BEEPING SPK
FF4E  8A C4                     MOV     AL,AH           ; RECOVER VALUE OF PORT
FF50  E6 61                     OUT     PORT_B,AL
FF52  C3                        RET                     ; RETURN TO CALLER
FF53                    BEEP    ENDP
```

```
                        ;--------------------------------
                        ; DUMMY RETURN FOR ADDRESS COMPATIBILITY
                        ;--------------------------------
FF53                            ORG     0FF53H
FF53  CF                        IRET
                        ;-- INT 5 ------------------------------------------------------------
                        ;       THIS LOGIC WILL BE INVOKED BY INTERRUPT 05H TO PRINT
                        ;       THE SCREEN. THE CURSOR POSITION AT THE TIME THIS ROUTINE
                        ;       IS INVOKED WILL BE SAVED AND RESTORED UPON COMPLETION. THE
                        ;       ROUTINE IS INTENDED TO RUN WITH INTERRUPTS ENABLED.
                        ;       IF A SUBSEQUENT 'PRINT SCREEN KEY IS DEPRESSED DURING THE
                        ;       TIME THIS ROUTINE IS PRINTING IT WILL BE IGNORED.
                        ;       ADDRESS 50:0 CONTAINS THE STATUS OF THE PRINT SCREEN:
                        ;
                        ;       50:0    =0      EITHER PRINT SCREEN HAS NOT BEEN CALLED
                        ;                       OR UPON RETURN FROM A CALL THIS INDICATES
                        ;                       A SUCCESSFUL OPERATION.
                        ;
                        ;               =1      PRINT SCREEN IS IN PROGRESS
                        ;
                        ;               =0FFH   ERROR ENCOUNTERED DURING PRINTING
                        ;-------------------------------------------------------------------
                                ASSUME  CS:CODE,DS:XXDATA
FF54                            ORG     0FF54H
FF54                    PRINT_SCREEN    PROC    FAR
FF54  FB                        STI                     ; MUST RUN WITH INTERRUPTS ENABLED
FF55  1E                        PUSH    DS              ; MUST USE 50:0 FOR DATA AREA
                                                        ; STORAGE
FF56  50                        PUSH    AX
FF57  53                        PUSH    BX
FF58  51                        PUSH    CX              ; WILL USE THIS LATER FOR CURSOR
                                                        ; LIMITS
FF59  52                        PUSH    DX              ; WILL HOLD CURRENT CURSOR POSITION
FF5A  B8  ---- R                MOV     AX,XXDATA       ; HEX 50
FF5D  8E D8                     MOV     DS,AX
FF5F  80 3E 0000 R 01           CMP     STATUS_BYTE,1   ; SEE IF PRINT ALREADY IN PROGRESS
FF64  74 5F                     JZ      EXIT            ; JUMP IF PRINT ALREADY IN PROGRESS
FF66  C6 06 0000 R 01           MOV     STATUS_BYTE,1   ; INDICATE PRINT NOW IN PROGRESS
FF6B  B4 0F                     MOV     AH,15           ; WILL REQUEST THE CURRENT SCREEN
                                                        ; MODE
FF6D  CD 10                     INT     10H             ;       [AL]=MODE
                                                        ;       [AH]=NUMBER COLUMNS/LINE
                                                        ;       [BH]=VISUAL PAGE
                        ; ##########################################################
                        ;       AT THIS POINT WE KNOW THE COLUMNS/LINE ARE IN
                        ;       [AX] AND THE PAGE IF APPLICABLE IS IN [BH]. THE STACK
                        ;       HAS DS,AX,BX,CX,DX PUSHED. [AL] HAS VIDEO MODE
                        ; ##########################################################
FF6F  8A CC                     MOV     CL,AH           ; WILL MAKE USE OF [CX] REGISTER TO
FF71  B5 19                     MOV     CH,25           ; CONTROL ROW & COLUMNS
FF73  E8 FA5F R                 CALL    CRLF            ; CARRIAGE RETURN LINE FEED ROUTINE
FF76  51                        PUSH    CX              ; SAVE SCREEN BOUNDS
FF77  B4 03                     MOV     AH,3            ; WILL NOW READ THE CURSOR.
FF79  CD 10                     INT     10H             ; AND PRESERVE THE POSITION
FF7B  59                        POP     CX              ; RECALL SCREEN BOUNDS
FF7C  52                        PUSH    DX              ; RECALL [BH]=VISUAL PAGE
FF7D  33 D2                     XOR     DX,DX           ; WILL SET CURSOR POSITION TO [0,0]
                        ; ##########################################################
                        ;       THE LOOP FROM PRI10 TO THE INSTRUCTION PRIOR TO PRI20
                        ;       IS THE LOOP TO READ EACH CURSOR POSITION FROM THE SCREEN
                        ;       AND PRINT.
                        ; ##########################################################
FF7F  B4 02             PRI10:  MOV     AH,2            ; TO INDICATE CURSOR SET REQUEST
FF81  CD 10                     INT     10H             ; NEW CURSOR POSITION ESTABLISHED
FF83  B4 08                     MOV     AH,8            ; TO INDICATE READ CHARACTER
FF85  CD 10                     INT     10H             ; CHARACTER NOW IN [AL]
FF87  0A C0                     OR      AL,AL           ; SEE IF VALID CHAR
FF89  75 02                     JNZ     PRI15           ; JUMP IF VALID CHAR
FF8B  B0 20                     MOV     AL,' '          ; MAKE A BLANK
FF8D  52                PRI15:  PUSH    DX              ; SAVE CURSOR POSITION
FF8E  33 D2                     XOR     DX,DX           ; INDICATE PRINTER 1
FF90  32 E4                     XOR     AH,AH           ; TO INDICATE PRINT CHAR IN [AL]
FF92  CD 17                     INT     17H             ; PRINT THE CHARACTER
FF94  5A                        POP     DX              ; RECALL CURSOR POSITION
FF95  F6 C4 29                  TEST    AH,029H         ; TEST FOR PRINTER ERROR
FF98  75 21                     JNZ     ERR10           ; JUMP IF ERROR DETECTED
FF9A  FE C2                     INC     DL              ; ADVANCE TO NEXT COLUMN
FF9C  3A CA                     CMP     CL,DL           ; SEE IF AT END OF LINE
FF9E  75 DF                     JNZ     PRI10           ; IF NOT PROCEED
FFA0  32 D2                     XOR     DL,DL           ; BACK TO COLUMN 0
FFA2  8A E2                     MOV     AH,DL           ; [AH]=0
FFA4  52                        PUSH    DX              ; SAVE NEW CURSOR POSITION
FFA5  E8 FA5F R                 CALL    CRLF            ; LINE FEED CARRIAGE RETURN
FFA8  5A                        POP     DX              ; RECALL CURSOR POSITION
FFA9  FE C6                     INC     DH              ; ADVANCE TO NEXT LINE
FFAB  3A EE                     CMP     CH,DH           ; FINISHED?
FFAD  75 D0                     JNZ     PRI10           ; IF NOT CONTINUE
FFAF  5A                        POP     DX              ; RECALL CURSOR POSITION
FFB0  B4 02                     MOV     AH,2            ; TO INDICATE CURSOR SET REQUEST
FFB2  CD 10                     INT     10H             ; CURSOR POSITION RESTORED
FFB4  C6 06 0000 R 00           MOV     STATUS_BYTE,0   ; INDICATE FINISHED
FFB9  EB 0A                     JMP     SHORT EXIT      ; EXIT THE ROUTINE
FFBB  5A                ERR10:  POP     DX              ; GET CURSOR POSITION
FFBC  B4 02                     MOV     AH,2            ; TO REQUEST CURSOR SET
FFBE  CD 10                     INT     10H             ; CURSOR POSITION RESTORED
FFC0  C6 06 0000 R FF           MOV     STATUS_BYTE,0FFH        ; INDICATE ERROR
FFC5  5A                EXIT:   POP     DX              ; RESTORE ALL THE REGISTERS USED
FFC6  59                        POP     CX
FFC7  5B                        POP     BX
FFC8  58                        POP     AX
FFC9  1F                        POP     DS
FFCA  CF                        IRET
FFCB                    PRINT_SCREEN    ENDP
```

```
                        ;--------------------------------------------------;
                        ;  EASE OF USE REVECTOR ROUTINE - CALLED THROUGH   ;
                        ;  INT 18H WHEN CASSETTE BASIC IS INVOKED (NO DISKETTE
                        ;  NO CARTRIDGES)                                  ;
                        ;  KEYBOARD VECTOR IS RESET TO POINT TO "NEW_INT_9" ;
                        ;  BASIC VECTOR IS SET TO POINT TO F600:0          ;
                        ;--------------------------------------------------
FFC8                    BAS_ENT PROC    FAR
                                ASSUME  DS:ABS0
FFC8  2B C0                     SUB     AX,AX
FFCA  8E D8                     MOV     DS,AX           ;SET ADDRESSING
FFCC  C7 06 0024 R 1937 R       MOV     WORD PTR INT_PTR+4,OFFSET NEW_INT_9
FFD2  A3 0060 R                 MOV     BASIC_PTR,AX    ; SET INT 18=F600:0
FFD5  C7 06 0062 R F600         MOV     BASIC_PTR+2,0F600H
FFDB  CD 18                     INT     18H             ; GO TO BASIC
FFDD                    BAS_ENT ENDP
                        ;--------------------------------------------------------------------------
                        ;       INITIALIZE TIMER SUBROUTINE - ASSUMES BOTH THE LSB AND MSB
                        ;       OF THE TIMER WILL BE USED.
                        ;   CALLING PARAMETERS:
                        ;       (AH) = TIMER #
                        ;       (AL) = BIT PATTERN OF INITIALIZATION WORD
                        ;       (BX) = INITIAL COUNT
                        ;              (BH) = MSB COUNT
                        ;              (BL) = LSB COUNT
                        ;   ALTERS REGISTERS DX AND AL.
                        ;--------------------------------------------------------------------------
FFE0                    INIT_TIMER      PROC  NEAR
FFE0  E6 43                     OUT     TIM_CTL,AL      ; OUTPUT INITIAL CONTROL WORD
FFE2  BA 0040                   MOV     DX,TIMER        ; BASE PORT ADDR FOR TIMERS
FFE5  02 D4                     ADD     DL,AH           ; ADD IN THE TIMER #
FFE7  8A C3                     MOV     AL,BL           ; LOAD LSB
FFE9  EE                        OUT     DX,AL
FFEA  52                        PUSH    DX              ; PAUSE
FFEB  5A                        POP     DX
FFEC  8A C7                     MOV     AL,BH           ; LOAD MSB
FFEE  EE                        OUT     DX,AL
FFEF  C3                        RET
FFF0                    INIT_TIMER      ENDP

                        ;--------------------------------
                        ;       POWER ON RESET VECTOR    :
                        ;--------------------------------
FFF0                            ORG     0FFF0H
                        ;
                        ;----- POWER ON RESET
                        :
FFF0  EA                        DB      0EAH                    ; JUMP FAR
FFF1  0043 R                    DW      OFFSET RESET
FFF3  F000                      DW      0F000H

FFF5  30 36 2F 30 31 2F         DB      '06/01/83'              ; RELEASE MARKER
      38 33

FFFD  FF                        DB      0FFH                    ; FILLER

FFFE  FD                        DB      0FDH                    ; SYSTEM IDENTIFIER

                        ;       DB      0FFH                    ; CHECKSUM
FFFF                    CODE    ENDS
                                END
```