```
                        ;--- 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
F028  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 DB                             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 F8                             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
```

```
F290                                ;------ 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
```

```
F3E0                    ;------ 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
F459  26: 32 05
F45C  AA
F45D  AC
F45E  26: 32 85 1FFF
F463  EB E0

F465
F465  1F
F466  8A D3
F468  D1 E7
F46A  E8 F659 R
F46D
F46D  57
F46E  56
F46F  B6 04
F471  E8 F626 R
F474  81 C7 2000
F478  E8 F626 R
F47B  81 EF 1FB0
F47F  FE CE
F481  75 EE
F483  5E
F484  5F
F485  47
F486  47
F487  E2 E4
F489  EB CB

F48B
F48B  1F
F48C  8A D3
F48E  D1 E7
F490  D1 E7
F492  E8 F66E R
F495
F495  57
F496  56
F497  B6 04
F499  E8 F645 R
F49C  81 C7 2000
F4A0  E8 F645 R
F4A3  1E
F4A4  E8 1388 R
F4A7  80 3E 0049 R 09
F4AC  1F
F4AD  75 14
F4AF  81 C7 2000
F4B3  E8 F645 R
F4B6  81 C7 2000
F4BA  E8 F645 R
F4BD  81 EF 3FB0
F4C1  FE CE
F4C3  81 EF 1FB0
F4C7  FE CE
F4C9  75 CE
F4CB  5E
F4CC  5F
F4CD  83 C7 04
F4D0  E2 C3
F4D2  EB B2

F4D4  1F
F4D5  8A D3
F4D7  D1 E7

F4D9  33 C0
F4DB  F6 C3 01
F4DE  74 02
F4E0  B4 FF
F4E2  F6 C3 02
F4E5  74 02
F4E7  B0 FF
F4E9  8B D8
F4EB
F4EB  57
F4EC  56
F4ED  B6 02
F4EF  E8 F518 R
F4F2  81 C7 2000
F4F6  E8 F518 R
F4F9  81 C7 2000
F4FD  E8 F518 R
F500  81 C7 2000
F504  E8 F518 R
F507  81 EF 5F60
F50B  FE CE
F500  75 E0
F50F  5E
F510  5F
F511  47
F512  47
F513  E2 D6
F515  E9 0F70 R
```

```
R705:   JMP     VIDEO_RETURN
R8:     XOR     AL,ES:[DI]          ; EXCLUSIVE OR WITH CURRENT DATA
        STOSB                       ; STORE THE CODE POINT
        LODSB                       ; AGAIN FOR ODD FIELD
        XOR     AL,ES:[DI+2000H-1]
        JMP     R7                  ; BACK TO MAINSTREAM
;------ MEDIUM RESOLUTION WRITE
R9:                                 ; MED_RES_WRITE
        POP     DS                  ; RECOVER TABLE POINTER SEGMENT
        MOV     DL,BL               ; SAVE HIGH COLOR BIT
        SAL     DI,1                ; OFFSET*2 SINCE 2 BYTES/CHAR
        CALL    R40                 ; EXPAND BL TO FULL WORD OF COLOR
R10:                                ; MED_CHAR
        PUSH    DI                  ; SAVE REGEN POINTER
        PUSH    SI                  ; SAVE THE CODE POINTER
        MOV     DH,4                ; NUMBER OF LOOPS
R11:    CALL    R35                 ; DO FIRST 2 BYTES
        ADD     DI,2000H            ; NEXT SPOT IN REGEN
        CALL    R35                 ; DO NEXT 2 BYTES
        SUB     DI,2000H-80
        DEC     DH
        JNZ     R11                 ; KEEP GOING
        POP     SI                  ; RECOVER CODE PONTER
        POP     DI                  ; RECOVER REGEN POINTER
        INC     DI                  ; POINT TO NEXT CHAR POSITION
        INC     DI
        LOOP    R10                 ; MORE TO WRITE
        JMP     R705
;------ LOW RESOLUTION WRITE
R12:                                ; LOW_RES_WRITE
        POP     DS                  ; RECOVER TABLE POINTER SEGMENT
        MOV     DL,BL               ; SAVE HIGH COLOR BIT
        SAL     DI,1                ; OFFSET*4 SINCE 4 BYTES/CHAR
        SAL     DI,1                ;
        CALL    R42                 ; EXPAND BL TO FULL WORD OF COLOR
R13:                                ; MED_CHAR
        PUSH    DI                  ; SAVE REGEN POINTER
        PUSH    SI                  ; SAVE THE CODE POINTER
        MOV     DH,4                ; NUMBER OF LOOPS
R14:    CALL    R39                 ; EXPAND DOT ROW IN REGEN
        ADD     DI,2000H            ; POINT TO NEXT REGEN ROW
        CALL    R39                 ; EXPAND DOT ROW IN REGEN
        PUSH    DS                  ; SAVE DS
        CALL    DDS                 ; POINT TO BIOS DATA AREA
        CMP     CRT_MODE,09H        ; USING 32K REGEN AREA?
        POP     DS                  ; RECOVER DS
        JNE     R15                 ; JUMP IF 16K REGEN
        ADD     DI,2000H            ; POINT TO NEXT REGEN ROW
        CALL    R39                 ; EXPAND DOT ROW IN REGEN
        ADD     DI,2000H            ; POINT TO NEXT REGEN ROW
        CALL    R39                 ; EXPAND DOT ROW IN REGEN
        SUB     DI,4000H-80         ; ADJUST REGEN POINTER
        DEC     DH
R15:    SUB     DI,2000H-80         ; ADJUST REGEN POINTER TO NEXT ROW
        DEC     DH
        JNZ     R14                 ; KEEP GOING
        POP     SI                  ; RECOVER CODE PONTER
        POP     DI                  ; RECOVER REGEN POINTER
        ADD     DI,4                ; POINT TO NEXT CHAR POSITION
        LOOP    R13                 ; MORE TO WRITE
        JMP     R705
;----- 640X200 4 COLOR GRAPHICS WRITE
R16:    POP     DS                  ; RECOVER TABLE SEGMENT POINTER
        MOV     DL,BL               ; SAVE HIGH COLOR BIT
        SAL     DI,1                ; OFFSET*2 SINCE 2 BYTES/CHAR
; EXPAND LOW 2 COLOR BITS IN BL (c1c0)
; INTO BX (c0c0c0c0c0c0c0c0c1c1c1c1c1c1c1c1)
        XOR     AX,AX
        TEST    BL,1                ; c0 COLOR BIT ON?
        JZ      R17                 ; NO, JUMP
        MOV     AH,0FFH             ; YES, SET ALL c0 BITS ON
R17:    TEST    BL,2                ; c1 COLOR BIT ON?
        JZ      R18                 ; NO, JUMP
        MOV     AL,0FFH             ; YES, SET ALL c1 BITS ON
R18:    MOV     BX,AX               ; COLOR MASK IN BX
R19:
        PUSH    DI                  ; SAVE REGEN POINTER
        PUSH    SI                  ; SAVE CODE POINT POINTER
        MOV     DH,2                ; SET LOOP COUNTER
R20:    CALL    R21                 ; DO FIRST DOT ROW
        ADD     DI,2000H            ; ADJUST REGEN POINTER
        CALL    R21                 ; DO NEXT DOT ROW
        ADD     DI,2000H            ; ADJUST REGEN POINTER
        CALL    R21                 ; DO NEXT DOT ROW
        ADD     DI,2000H            ; ADJUST REGEN POINTER
        CALL    R21                 ; DO NEXT DOT ROW
        SUB     DI,6000H-160        ; ADJUST REGEN POINTER TO NEXT ROW
        DEC     DH
        JNZ     R20                 ; KEEP GOING
        POP     SI                  ; RECOVER CODE POINT POINTER
        POP     DI                  ; RECOVER REGEN POINTER
        INC     DI                  ; POINT TO NEXT CHARACTER
        INC     DI
        LOOP    R19                 ; MORE TO WRITE
        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                    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             R41:    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                    R40     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                    R42     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             R44:    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                    R43     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             R46:    TEST    AL,4
F6AC  74 03                     JZ      R47
F6AE  80 CE 0F                  OR      DH,0FH
F6B1  A8 02             R47:    TEST    AL,2
F6B3  74 03                     JZ      R48
F6B5  80 CA F0                  OR      DL,0F0H
F6B8  A8 01             R48:    TEST    AL,1
F6BA  74 03                     JZ      R49
F6BC  80 CA 0F                  OR      DL,0FH
F6BF  8B C2             R49:    MOV     AX,DX           ; RESULT TO PARM REGISTER
F6C1  5A                        POP     DX              ; RECOVER REGISTERS
F6C2  C3                        RET                     ; ALL DONE
F6C3                    R45     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 138B 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  8A 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
F782  8A E8                     MOV     CH,AL           ; SAVE ROW VALUE IN CH
F784  02 ED                     ADD     CH,CH           ; *2 FOR EVEN/ODD FIELD
F786  80 3E 0049 R 09           CMP     CRT_MODE,9      ; USING 32K REGEN?
F78B  72 06                     JB      V21             ; NO, JUMP
F78D  D0 EC                     SHR     AH,1            ; ADJUST ROW & COLUMN
F78F  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                                ;--------------------------------------------------------
FA50  E8 F881 R                     BEGIN_OP        PROC    NEAR    ; START TAPE AND DELAY
FA53  B3 42                                 CALL    MOTOR_ON        ;TURN ON MOTOR
                                            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             
      00 00                                 DB      000H,000H,000H,000H,000H,000H,000H,000H ; D_00
FA76  7E 81 A5 81 BD 99             
      81 7E                                 DB      07EH,081H,0A5H,081H,0BDH,099H,081H,07EH ; D_01
FA7E  7E FF DB FF C3 E7             
      FF 7E                                 DB      07EH,0FFH,0DBH,0FFH,0C3H,0E7H,0FFH,07EH ; D_02
FA86  6C FE FE FE 7C 38             
      10 00                                 DB      06CH,0FEH,0FEH,0FEH,07CH,038H,010H,000H ; D_03
FA8E  10 38 7C FE 7C 38             
      10 00                                 DB      010H,038H,07CH,0FEH,07CH,038H,010H,000H ; D_04
FA96  38 7C 38 FE FE 7C             
      38 7C                                 DB      038H,07CH,038H,0FEH,0FEH,07CH,038H,07CH ; D_05
FA9E  10 10 38 7C FE 7C             
      38 7C                                 DB      010H,010H,038H,07CH,0FEH,07CH,038H,07CH ; D_06
FAA6  00 00 18 3C 3C 18             
      00 00                                 DB      000H,000H,018H,03CH,03CH,018H,000H,000H ; D_07
FAAE  FF FF E7 C3 C3 E7             
      FF FF                                 DB      0FFH,0FFH,0E7H,0C3H,0C3H,0E7H,0FFH,0FFH ; D_08
FAB6  00 3C 66 42 42 66             
      3C 00                                 DB      000H,03CH,066H,042H,042H,066H,03CH,000H ; D_09
FABE  FF C3 99 BD BD 99             
      C3 FF                                 DB      0FFH,0C3H,099H,0BDH,0BDH,099H,0C3H,0FFH ; D_0A
FAC6  0F 07 0F 7D CC CC             
      CC 78                                 DB      00FH,007H,00FH,07DH,0CCH,0CCH,0CCH,078H ; D_0B
FACE  3C 66 66 66 3C 18             
      7E 18                                 DB      03CH,066H,066H,066H,03CH,018H,07EH,018H ; D_0C
FAD6  3F 33 3F 30 30 70             
      F0 E0                                 DB      03FH,033H,03FH,030H,030H,070H,0F0H,0E0H ; D_0D
FADE  7F 63 7F 63 63 67             
      E6 C0                                 DB      07FH,063H,07FH,063H,063H,067H,0E6H,0C0H ; D_0E
FAE6  99 5A 3C E7 E7 3C             
      5A 99                                 DB      099H,05AH,03CH,0E7H,0E7H,03CH,05AH,099H ; D_0F

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

```
FB6E  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 F0 00          DB      0FCH,066H,066H,07CH,060H,060H,0F0H,000H ; P D_50
FCF6  78 CC CC CC DC 78 1C 00          DB      078H,0CCH,0CCH,0CCH,0DCH,078H,01CH,000H ; Q D_51
FCFE  FC 66 66 7C 6C 66 E6 00          DB      0FCH,066H,066H,07CH,06CH,066H,0E6H,000H ; R D_52
FD06  78 CC E0 70 1C CC 78 00          DB      078H,0CCH,0E0H,070H,01CH,0CCH,078H,000H ; S D_53
FD0E  FC B4 30 30 30 30 78 00          DB      0FCH,0B4H,030H,030H,030H,030H,078H,000H ; T D_54
FD16  CC CC CC CC CC CC FC 00          DB      0CCH,0CCH,0CCH,0CCH,0CCH,0CCH,0FCH,000H ; U D_55
FD1E  CC CC CC CC CC 78 30 00          DB      0CCH,0CCH,0CCH,0CCH,0CCH,078H,030H,000H ; V D_56
FD26  C6 C6 C6 D6 FE EE C6 00          DB      0C6H,0C6H,0C6H,0D6H,0FEH,0EEH,0C6H,000H ; W D_57
FD2E  C6 C6 6C 38 38 6C C6 00          DB      0C6H,0C6H,06CH,038H,038H,06CH,0C6H,000H ; X D_58
FD36  CC CC CC 78 30 30 78 00          DB      0CCH,0CCH,0CCH,078H,030H,030H,078H,000H ; Y D_59
FD3E  FE C6 8C 18 32 66 FE 00          DB      0FEH,0C6H,08CH,018H,032H,066H,0FEH,000H ; Z D_5A
FD46  78 60 60 60 60 60 78 00          DB      078H,060H,060H,060H,060H,060H,078H,000H ; [ D_5B
FD4E  C0 60 30 18 0C 06 02 00          DB      0C0H,060H,030H,018H,00CH,006H,002H,000H ;
                                                       ; BACKSLASH D_5C
FD56  78 18 18 18 18 18 78 00          DB      078H,018H,018H,018H,018H,018H,078H,000H ; ] D_5D
FD5E  10 38 6C C6 00 00 00 00          DB      010H,038H,06CH,0C6H,000H,000H,000H,000H ;
                                                       ; CIRCUMFLEX D_5E
FD66  00 00 00 00 00 00 00 FF          DB      000H,000H,000H,000H,000H,000H,000H,0FFH ; _ D_5F
FD6E  30 30 18 00 00 00 00 00          DB      030H,030H,018H,000H,000H,000H,000H,000H ; ` D_60
FD76  00 00 78 0C 7C CC 76 00          DB      000H,000H,078H,00CH,07CH,0CCH,076H,000H ;
                                                       ; LOWER CASE A D_61
FD7E  E0 60 60 7C 66 66 DC 00          DB      0E0H,060H,060H,07CH,066H,066H,0DCH,000H ; LC B D_62
FD86  00 00 78 CC C0 CC 78 00          DB      000H,000H,078H,0CCH,0C0H,0CCH,078H,000H ; LC C D_63
FD8E  1C 0C 0C 7C CC CC 76 00          DB      01CH,00CH,00CH,07CH,0CCH,0CCH,076H,000H ; LC D D_64
FD96  00 00 78 CC FC C0 78 00          DB      000H,000H,078H,0CCH,0FCH,0C0H,078H,000H ; LC E D_65
FD9E  38 6C 60 F0 60 60 F0 00          DB      038H,06CH,060H,0F0H,060H,060H,0F0H,000H ; LC F D_66
FDA6  00 00 76 CC CC 7C 0C F8          DB      000H,000H,076H,0CCH,0CCH,07CH,00CH,0F8H ; LC G D_67
FDAE  E0 60 6C 76 66 66 E6 00          DB      0E0H,060H,06CH,076H,066H,066H,0E6H,000H ; LC H D_68
FDB6  30 00 70 30 30 30 78 00          DB      030H,000H,070H,030H,030H,030H,078H,000H ; LC I D_69
FDBE  0C 00 0C 0C 0C CC CC 78          DB      00CH,000H,00CH,00CH,00CH,0CCH,0CCH,078H ; LC J D_6A
FDC6  E0 60 66 6C 78 6C E6 00          DB      0E0H,060H,066H,06CH,078H,06CH,0E6H,000H ; LC K D_6B
FDCE  70 30 30 30 30 30 78 00          DB      070H,030H,030H,030H,030H,030H,078H,000H ; LC L D_6C
FDD6  00 00 CC FE FE D6 C6 00          DB      000H,000H,0CCH,0FEH,0FEH,0D6H,0C6H,000H ; LC M D_6D
FDDE  00 00 F8 CC CC CC CC 00          DB      000H,000H,0F8H,0CCH,0CCH,0CCH,0CCH,000H ; LC N D_6E
FDE6  00 00 78 CC CC CC 78 00          DB      000H,000H,078H,0CCH,0CCH,0CCH,078H,000H ; LC O D_6F

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

```
FE6E                            ORG     0FE6EH
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,0FFFFH       ; 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     0FEA5H
                                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 138B 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  EF02 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 0F                     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
                        ;-----------------------------------------------
FFCB                    BAS_ENT PROC    FAR
                                ASSUME  DS:ABS0
FFCB  2B C0                     SUB     AX,AX
FFCD  8E D8                     MOV     DS,AX           ;SET ADDRESSING
FFCF  C7 06 0024 R 1937 R       MOV     WORD PTR INT_PTR+4,OFFSET NEW_INT_9
FFD5  A3 0060 R                 MOV     BASIC_PTR,AX    ; SET INT 18=F600:0
FFD8  C7 06 0062 R F600         MOV     BASIC_PTR+2,0F600H
FFDE  CD 18                     INT     18H             ; GO TO BASIC
FFE0                    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
```

# Notes:

# Appendix B. LOGIC DIAGRAMS



## Contents

System Board ..... B-3
Program Cartridge ..... B-20
Power Supply Board ..... B-23
64KB Memory and Display Expansion ..... B-25
Color Display ..... B-29
Diskette Drive Adapter ..... B-30
Internal Modem ..... B-36
Parallel Printer Attachment ..... B-37
Infra-Red Receiver Board ..... B-42
Graphics Printer ..... B-43
Compact Printer ..... B-47

# Notes: