APPENDIX A
ROM BIOS LISTINGS

CONTENTS

```
LOC OBJ                 LINE    SOURCE

                          1     $TITLE(ROM BIOS FOR IBM PERSONAL COMPUTER)
                          2     ;--------------------------
                          3     ; EQUATES
                          4     ;--------------------------
  0060                    5     PORT_A          EQU     60H                     ;8255 PORT A ADDR
  0061                    6     PORT_B          EQU     61H                     ;8255 PORT B ADDR
  0062                    7     PORT_C          EQU     62H                     ;8255 PORT C ADDR
  0063                    8     CMD_PORT EQU    63H
  0020                    9     INTA00          EQU     20H                     ;8259 PORT
  0021                   10     INTA01          EQU     21H                     ;8259 PORT
  0020                   11     EOI             EQU     20H
  0040                   12     TIMER           EQU     40H
  0043                   13     TIM_CTL EQU     43H                     ;8253 TIMER CONTROL PORT ADDR
  0040                   14     TIMER0  EQU     40H                     ;8253 TIMER/CNTER 0 PORT ADDR
  0001                   15     TMINT   EQU     01                      ;TIMER 0 INTR RECVD MASK
  0008                   16     DMA08   EQU     08                      ;DMA STATUS REG PORT ADDR
  0000                   17     DMA     EQU     00                      ;DMA CHANNEL 0 ADDRESS REG PORT ADDR
  0540                   18     MAX_PERIOD EQU  540H
  0410                   19     MIN_PERIOD EQU  410H
  0060                   20     KBD_IN  EQU     60H                             ;KEYBOARD DATA IN ADDR PORT
  0002                   21     KBDINT  EQU     02                              ;KEYBOARD INTR MASK
  0060                   22     KB_DATA         EQU     60H     ; KEYBOARD SCAN CODE PORT
  0061                   23     KB_CTL          EQU     61H     ; CONTROL BITS FOR KEYBOARD SENSE DATA
                         24     ;-----------------------------------------------
                         25     ; 8088 INTERRUPT LOCATIONS
                         26     ;-----------------------------------------------
0000                     27     ABS0    SEGMENT AT 0
0000                     28     STG_LOC0        LABEL   BYTE
0008                     29             ORG     2*4
0008                     30     NMI_PTR         LABEL    WORD
0014                     31             ORG     5*4
0014                     32     INT5_PTR        LABEL   WORD
0020                     33             ORG     8*4
0020                     34     INT_ADDR        LABEL   WORD
0020                     35     INT_PTR         LABEL   DWORD
0040                     36             ORG     10H*4
0040                     37     VIDEO_INT       LABEL   WORD
0074                     38             ORG     1DH*4
0074                     39     PARM_PTR        LABEL   DWORD           ; POINTER TO VIDEO PARMS
0078                     40             ORG     01EH*4                  ; INTERRUPT 1EH
0078                     41     DISK_POINTER    LABEL   DWORD
007C                     42             ORG     01FH*4                  ; LOCATION OF POINTER
007C                     43     EXT_PTR LABEL   DWORD           ; POINTER TO EXTENSION
7C00                     44             ORG     7C00H
7C00                     45     BOOT_LOCN       LABEL   FAR
                         46     ABS0    ENDS
                         47
                         48     ;----------------------
                         49     ; STACK -- USED DURING INITIALIZATION ONLY
                         50     ;----------------------
                         51     STACK   SEGMENT AT 30H
0000 (128 ????)          52             DW      128 DUP(?)
0100                     53     TOS     LABEL   WORD
                         54     STACK   ENDS
                         55
                         56     ;-----------------------------------------------
                         57     ; ROM BIOS DATA AREAS
                         58     ;-----------------------------------------------
0040                     59     DATA    SEGMENT AT 40H
0000 (4 ????)            60     RS232_BASE      DW      4 DUP(?)        ; ADDRESSES OF RS232 ADAPTERS
0008 (4 ????)            61     PRINTER_BASE    DW      4 DUP(?)        ; ADDRESSES OF PRINTERS
0010 ????                62     EQUIP_FLAG      DW      ?               ; INSTALLED HARDWARE
0012 ??                  63     MFG_TST         DB      ?               ; INITIALIZATION FLAG
0013 ????                64     MEMORY_SIZE     DW      ?               ; MEMORY SIZE IN K BYTES
0015 ????                65     IO_RAM_SIZE     DW      ?               ; MEMORY IN I/O CHANNEL
                         66     ;-----------------------------------------------
                         67     ; KEYBOARD DATA AREAS
                         68     ;-----------------------------------------------
0017 ??                  69     KB_FLAG         DB      ?
                         70
                         71     ;------ SHIFT FLAG EQUATES WITHIN KB_FLAG
                         72
  0080                   73     INS_STATE       EQU     80H     ; INSERT STATE IS ACTIVE
  0040                   74     CAPS_STATE      EQU     40H     ; CAPS LOCK STATE HAS BEEN TOGGLED
  0020                   75     NUM_STATE       EQU     20H     ; NUM LOCK STATE HAS BEEN TOGGLED
  0010                   76     SCROLL_STATE    EQU     10H     ; SCROLL LOCK STATE HAS BEEN TOGGLED
  0008                   77     ALT_SHIFT       EQU     08H     ; ALTERNATE SHIFT KEY DEPRESSED
```

```
LOC OBJ                 LINE    SOURCE

  0004                   78    CTL_SHIFT       EQU     04H     ; CONTROL SHIFT KEY DEPRESSED
  0002                   79    LEFT_SHIFT      EQU     02H     ; LEFT SHIFT KEY DEPRESSED
  0001                   80    RIGHT_SHIFT     EQU     01H     ; RIGHT SHIFT KEY DEPRESSED
                         81
0018 ??                  82    KB_FLAG_1       DB      ?       ; SECOND BYTE OF KEYBOARD STATUS
                         83
  0080                   84    INS_SHIFT       EQU     80H     ; INSERT KEY IS DEPRESSED
  0040                   85    CAPS_SHIFT      EQU     40H     ; CAPS LOCK KEY IS DEPRESSED
  0020                   86    NUM_SHIFT       EQU     20H     ; NUM LOCK KEY IS DEPRESSED
  0010                   87    SCROLL_SHIFT    EQU     10H     ; SCROLL LOCK KEY IS DEPRESSED
  0008                   88    HOLD_STATE      EQU     08H     ; SUSPEND KEY HAS BEEN TOGGLED
                         89
0019 ??                  90    ALT_INPUT       DB      ?       ; STORAGE FOR ALTERNATE KEYPAD ENTRY
001A ????                91    BUFFER_HEAD     DW      ?       ; POINTER TO HEAD OF KEYBOARD BUFFER
001C ????                92    BUFFER_TAIL     DW      ?       ; POINTER TO TAIL OF KEYBOARD BUFFER
001E (16 ????)           93    KB_BUFFER       DW      16 DUP(?) ; ROOM FOR 15 ENTRIES
003E                     94    KB_BUFFER_END   LABEL   WORD
                         95
                         96    ;------ HEAD = TAIL INDICATES THAT THE BUFFER IS EMPTY
                         97
  0045                   98    NUM_KEY         EQU     69      ; SCAN CODE FOR NUMBER LOCK
  0046                   99    SCROLL_KEY      EQU     70      ; SCROLL LOCK KEY
  0038                  100    ALT_KEY         EQU     56      ; ALTERNATE SHIFT KEY SCAN CODE
  001D                  101    CTL_KEY         EQU     29      ; SCAN CODE FOR CONTROL KEY
  003A                  102    CAPS_KEY        EQU     58      ; SCAN CODE FOR SHIFT LOCK
  002A                  103    LEFT_KEY        EQU     42      ; SCAN CODE FOR LEFT SHIFT
  0036                  104    RIGHT_KEY       EQU     54      ; SCAN CODE FOR RIGHT SHIFT
  0052                  105    INS_KEY         EQU     82      ; SCAN CODE FOR INSERT KEY
  0053                  106    DEL_KEY         EQU     83      ; SCAN CODE FOR DELETE KEY
                        107
                        108    ;----------------------------------------------
                        109    ; DISKETTE DATA AREAS
                        110    ;----------------------------------------------
003E ??                 111    SEEK_STATUS     DB      ?       ; DRIVE RECALIBRATION STATUS
                        112    ;                               BIT 3-0 = DRIVE 3-0 NEEDS RECAL BEFORE
                        113    ;                               NEXT SEEK IF BIT IS = 0
  0080                  114    INT_FLAG        EQU     080H    ; INTERRUPT OCCURRENCE FLAG
003F ??                 115    MOTOR_STATUS    DB      ?       ; MOTOR STATUS
                        116    ;                               BIT 3-0 = DRIVE 3-0 IS CURRENTLY RUNNING
                        117    ;                               BIT 7   = CURRENT OPERATION IS A WRITE, REQUIRES DELAY
0040 ??                 118    MOTOR_COUNT     DB      ?       ; TIME OUT COUNTER FOR DRIVE TURN OFF
  0025                  119    MOTOR_WAIT      EQU     37      ; TWO SECONDS OF COUNTS FOR MOTOR TURN OFF
                        120
                        121    ;
0041 ??                 122    DISKETTE_STATUS DB      ?       ; SINGLE BYTE OF RETURN CODE INFO FOR STATUS
  0080                  123    TIME_OUT        EQU     80H     ; ATTACHMENT FAILED TO RESPOND
  0040                  124    BAD_SEEK        EQU     40H     ; SEEK OPERATION FAILED
  0020                  125    BAD_NEC         EQU     20H     ; NEC CONTROLLER HAS FAILED
  0010                  126    BAD_CRC         EQU     10H     ; BAD CRC ON DISKETTE READ
  0009                  127    DMA_BOUNDARY    EQU     09H     ; ATTEMPT TO DMA ACROSS 64K BOUNDARY
  0008                  128    BAD_DMA         EQU     08H     ; DMA OVERRUN ON OPERATION
  0004                  129    RECORD_NOT_FND  EQU     04H     ; REQUESTED SECTOR NOT FOUND
  0003                  130    WRITE_PROTECT   EQU     03H     ; WRITE ATTEMPTED ON WRITE PROT DISK
  0002                  131    BAD_ADDR_MARK   EQU     02H     ; ADDRESS MARK NOT FOUND
  0001                  132    BAD_CMD         EQU     01H     ; BAD COMMAND PASSED TO DISKETTE I/O
                        133
0042 (7 ??)             134    NEC_STATUS      DB      7 DUP(?)        ; STATUS BYTES FROM NEC
                        135
                        136    ;----------------------------------------------
                        137    ; VIDEO DISPLAY DATA AREA
                        138    ;----------------------------------------------
0049 ??                 139    CRT_MODE        DB      ?       ; CURRENT CRT MODE
004A ????               140    CRT_COLS        DW      ?       ; NUMBER OF COLUMNS ON SCREEN
004C ????               141    CRT_LEN         DW      ?       ; LENGTH OF REGEN IN BYTES
004E ????               142    CRT_START       DW      ?       ; STARTING ADDRESS IN REGEN BUFFER
0050 (8 ????)           143    CURSOR_POSN     DW      8 DUP(?)        ; CURSOR FOR EACH OF UP TO 8 PAGES
0060 ????               144    CURSOR_MODE     DW      ?       ; CURRENT CURSOR MODE SETTING
0062 ??                 145    ACTIVE_PAGE     DB      ?       ; CURRENT PAGE BEING DISPLAYED
0063 ????               146    ADDR_6845       DW      ?       ; BASE ADDRESS FOR ACTIVE DISPLAY CARD
0065 ??                 147    CRT_MODE_SET    DB      ?       ; CURRENT SETTING OF THE 3X8 REGISTER
0066 ??                 148    CRT_PALLETTE    DB      ?       ; CURRENT PALLETTE SETTING COLOR CARD
                        149
                        150    ;----------------------------------------------
                        151    ; CASSETTE DATA AREA
                        152    ;----------------------------------------------
0067 ????               153    EDGE_CNT        DW      ?               ;TIME COUNT AT DATA EDGE
0069 ????               154    CRC_REG         DW      ?               ;CRC REGISTER
```

```
LOC OBJ                  LINE    SOURCE

006B ??                   155    LAST_VAL        DB      ?               ;LAST INPUT VALUE
                          156
                          157    ;--------------------------------------------
                          158    ; TIMER DATA AREA
                          159    ;--------------------------------------------
006C ????                 160    TIMER_LOW       DW      ?       ; LOW WORD OF TIMER COUNT
006E ????                 161    TIMER_HIGH      DW      ?       ; HIGH WORD OF TIMER COUNT
0070 ??                   162    TIMER_OFL       DB      ?       ; TIMER HAS ROLLED OVER SINCE LAST READ
                          163    ;COUNTS_SEC     EQU     18
                          164    ;COUNTS_MIN     EQU     1092
                          165    ;COUNTS_HOUR    EQU     65543
                          166    ;COUNTS_DAY     EQU     1573040 = 1800B0H
                          167
                          168    ;--------------------------------------------
                          169    ; SYSTEM DATA AREA
                          170    ;--------------------------------------------
0071 ??                   171    BIOS_BREAK      DB      ?       ; BIT 7 = 1 IF BREAK KEY HAS BEEN DEPRESSED
0072 ????                 172    RESET_FLAG      DW      ?       ; WORD = 1234H IF KEYBOARD RESET UNDERWAY
                          173    DATA    ENDS
                          174
                          175    ;--------------------------------------------
                          176    ; EXTRA DATA AREA
                          177    ;--------------------------------------------
0050                      178    XXDATA  SEGMENT AT 50H
0000 ??                   179    STATUS_BYTE     DB      ?
                          180    XXDATA  ENDS
                          181
                          182    ;--------------------------------------------
                          183    ; VIDEO DISPLAY BUFFER
                          184    ;--------------------------------------------
B800                      185    VIDEO_RAM       SEGMENT AT 0B800H
0000                      186    REGEN   LABEL   BYTE
0000                      187    REGENW  LABEL   WORD
0000 (16384 ??)           188            DB      16384 DUP(?)
                          189    VIDEO_RAM       ENDS
                          190    ;--------------------------------------------
                          191    ; ROM RESIDENT CODE
                          192    ;--------------------------------------------
F000                      193    CODE    SEGMENT AT 0F000H
0000 (57344 ??)           194            DB      57344 DUP(?)    ; FILL LOWEST 56K
                          195
E000 35373030303531       196            DB      '5700051 COPR. IBM 1981'        ; COPYRIGHT NOTICE
     20434F50522E20
     494240203139 38
     31
                          197
                          198    ;--------------------------------------------
                          199    ; INITIAL RELIABILITY TESTS -- PHASE 1
                          200    ;--------------------------------------------
                          201            ASSUME  CS:CODE,SS:CODE,ES:ABS0,DS:DATA
                          202    ;--------------------------------------------
                          203    ;       DATA DEFINITIONS
                          204    ;--------------------------------------------
E016 D8E0          R      205    C1      DW      C11             ; RETURN ADDRESS
E018 EDE1          R      206    C2      DW      C24             ; RETURN ADDRESS FOR DUMMY STACK
                          207    ;--------------------------------------------
                          208    ;       THIS SUBROUTINE PERFORMS A READ/WRITE STORAGE TEST ON A 16K BLOCK
                          209    ;       OF STORGAGE.
                          210    ;ENTRY REQUIREMENTS:
                          211    ;       ES = ADDRESS OF STORAGE SEGMENT BEING TESTED
                          212    ;       DS = ADDRESS OF STORAGE SEGMENT BEING TESTED
                          213    ;       WHEN ENTERING AT STGTST_CNT, CX MUST BE LOADED WITH THE BYTE COUNT.
                          214    ;EXIT PARAMETERS:
                          215    ;       ZERO FLAG = 0 IF STORAGE ERROR (DATA COMPARE OR PARITY CHECK.  AL=0
                          216    ;                       DENOTES A PARITY CHECK. ELSE AL=XOR'ED BIT PATTERN OF THE
                          217    ;                       EXPECTED DATA PATTERN VS THE ACTUAL DATA READ.
                          218    ;       AX,BX,CX,DX,DI, AND SI ARE ALL DESTROYED.
                          219    ;--------------------------------------------
E01A                      220    STGTST          PROC    NEAR
E01A B90040               221            MOV     CX,4000H                ;SETUP CNT TO TEST A 16K BLK
E01D                      222    STGTST_CNT:
E01D FC                   223            CLD                             ;SET DIR FLAG TO INCREMENT
E01E 8BD9                 224            MOV     BX,CX                   ;SAVE BYTE CNT (4K FOR VIDEO OR 16K)
E020 B8FFFF               225            MOV     AX,0FFFFH               ;GET DATA PATTERN TO WRITE
E023 BA55AA               226            MOV     DX,0AA55H               ;SETUP OTHER DATA PATTERNS TO USE
E026 2BFF                 227            SUB     DI,DI                   ;DI = OFFSET 0 RELATIVE TO ES REG
E028 F3                   228            REP     STOSB                   ;WRITE STORAGE LOCATIONS
E029 AA
```

```
LOC OBJ                 LINE    SOURCE


E02A                    229     C3:                                     ; STG01
E02A 4F                 230             DEC     DI                      ;POINT TO LAST BYTE JUST WRITTEN
E02B FD                 231             STD                             ;SET DIR FLAG TO GO BACKWARDS
E02C 8BF7               232     C4:     MOV     SI,DI
E02E 8BCB               233             MOV     CX,BX                   ;SETUP BYTE CNT
E030 AC                 234     C5:     LODSB                           ;READ CHAR FROM STORAGE
E031 32C4               235             XOR     AL,AH                   ;DATA READ AS EXPECTED?
E033 7525               236             JNE     C7                      ;NO - GO TO ERROR ROUTINE
E035 E462               237             IN      AL,PORT_C               ;DID A PARITY ERROR OCCUR?
E037 24C0               238             AND     AL,0C0H
E039 B000               239             MOV     AL,0                    ;AL=0 DATA COMPARE OK
E03B 751D               240             JNZ     C7
E03D 80FC00             241             CMP     AH,0                    ;READING ZERO PATTERN?
E040 7403               242             JE      C6                      ;CONTINUE READING TILL END
E042 8AC2               243             MOV     AL,DL                   ;GET NEXT DATA PATTERN TO WRITE
E044 AA                 244             STOSB                           ;WRITE IN BYTE LOC WE JUST READ
E045                    245     C6:                                     ; WRITE_NO_MORE
E045 E2E9               246             LOOP    C5                      ;CONTINUE TILL 16K/4K BLOCK TESTED
E047 80FC00             247             CMP     AH,0                    ;ZERO PATTERN WRITTEN TO STG
E04A 740E               248             JE      C7                      ;YES - RETURN TO CALLER
E04C 8AE0               249             MOV     AH,AL                   ;SETUP TO NEW VALUE TO COMPARE
E04E 86F2               250             XCHG    DH,DL                   ;MOVE ZERO DATA PATTERN TO DL
E050 FC                 251             CLD                             ;SET DIR FLAG TO GO FORWARD
E051 47                 252             INC     DI                      ;SET POINTER TO BEG LOCATION
E052 74D8               253             JZ      C4                      ;READ/WRITE FORWARD IN STG
E054 4F                 254             DEC     DI
E055 BA0100             255             MOV     DX,1                    ;SETUP 01 AND 00 PATTERNS
E058 EBD0               256             JMP     SHORT C3                ;READ/WRITE BACKWARD IN STG
E05A                    257     C7:
E05A C3                 258             RET
                        259     STGTST  ENDP
                        260     ;-----------------------------------------------
                        261     ;TEST.01
                        262     ;       8088 PROCESSOR TEST
                        263     ;DESCRIPTION
                        264     ;       VERIFY 8088 FLAGS, REGISTERS AND CONDITIONAL JUMPS
                        265     ;-----------------------------------------------
E05B                    266     RESET   LABEL   NEAR
E05B FA                 267     START:  CLI                             ;DISABLE INTERRUPTS
E05C B4D5               268             MOV     AH,0D5H                 ;SET SF, CF, ZF, AND AF FLAGS ON
E05E 9E                 269             SAHF
E05F 734E               270             JNC     ERR01                   ;GO TO ERR ROUTINE IF CF NOT SET
E061 754C               271             JNZ     ERR01                   ;GO TO ERR ROUTINE IF ZF NOT SET
E063 7B4A               272             JNP     ERR01                   ;GO TO ERR ROUTINE IF PF NOT SET
E065 7948               273             JNS     ERR01                   ;GO TO ERR ROUTINE IF SF NOT SET
E067 9F                 274             LAHF                            ;LOAD FLAG IMAGE TO AH
E068 B105               275             MOV     CL,5                    ;LOAD CNT REG WITH SHIFT CNT
E06A D2EC               276             SHR     AH,CL                   ;SHIFT AF INTO CARRY BIT POS
E06C 7341               277             JNC     ERR01                   ;GO TO ERR ROUTINE IF AF NOT SET
E06E B040               278             MOV     AL,40H                  ;SET THE OF FLAG ON
E070 D0E0               279             SHL     AL,1                    ;SETUP FOR TESTING
E072 713B               280             JNO     ERR01                   ;GO TO ERR ROUTINE IF OF NOT SET
E074 32E4               281             XOR     AH,AH                   ;SET AH = 0
E076 9E                 282             SAHF                            ;CLEAR SF, CF, ZF, AND PF
E077 7236               283             JC      ERR01                   ;GO TO ERR ROUTINE IF CF ON
E079 7434               284             JZ      ERR01                   ;GO TO ERR ROUTINE IF ZF ON
E07B 7832               285             JS      ERR01                   ;GO TO ERR ROUTINE IF SF ON
E07D 7A30               286             JP      ERR01                   ;GO TO ERR ROUTINE IF PF ON
E07F 9F                 287             LAHF                            ;LOAD FLAG IMAGE TO AH
E080 B105               288             MOV     CL,5                    ;LOAD CNT REG WITH SHIFT CNT
E082 D2EC               289             SHR     AH,CL                   ;SHIFT  AF' INTO CARRY BIT POS
E084 7229               290             JC      ERR01                   ;GO TO ERR ROUTINE IF ON
E086 D0E4               291             SHL     AH,1                    ;CHECK THAT  OF' IS CLEAR
E088 7025               292             JO      ERR01                   ;GO TO ERR ROUTINE IF ON
                        293
                        294     ;         READ/WRITE THE 8088 GENERAL AND SEGMENTATION REGISTERS
                        295     ;         WITH ALL ONE'S AND ZEROES'S.
                        296
E08A B8FFFF             297             MOV     AX,0FFFFH                       ;SETUP ONE'S PATTERN IN AX
E08D F9                 298             STC
E08E 8ED8               299     C8:     MOV     DS,AX                           ;WRITE PATTERN TO ALL REGS
E090 8CDB               300             MOV     BX,DS
E092 8EC3               301             MOV     ES,BX
E094 8CC1               302             MOV     CX,ES
E096 8ED1               303             MOV     SS,CX
E098 8CD2               304             MOV     DX,SS
E09A 8BE2               305             MOV     SP,DX
E09C 8BEC               306             MOV     BP,SP
```

```
LOC OBJ                 LINE    SOURCE

E09E 8BF5               307             MOV     SI,BP
E0A0 8BFE               308             MOV     DI,SI
E0A2 7307               309             JNC     C9                              ; TST1A
E0A4 33C7               310             XOR     AX,DI                           ;PATTERN MAKE IT THRU ALL REGS
E0A6 7507               311             JNZ     ERR01                           ;NO - GO TO ERR ROUTINE
E0A8 F8                 312             CLC
E0A9 73E3               313             JNC     C8
E0AB                    314     C9:                                             ; TST1A
E0AB 0BC7               315             OR      AX,DI                           ;ZERO PATTERN MAKE IT THRU?
E0AD 7401               316             JZ       C10                            ;YES - GO TO NEXT TEST
E0AF F4                 317     ERR01:  HLT                                     ;HALT SYSTEM
                        318     ;-----------------------------------------
                        319     ;TEST.02
                        320     ;       ROS CHECKSUM TEST I
                        321     ;DESCRIPTION
                        322     ;       A CHECKSUM IS DONE FOR THE 8K ROS MODULE CONTAINING POD AND BIOS.
                        323     ;-----------------------------------------
E0B0                    324     C10:
E0B0 B000               325             MOV     AL,0                            ;DISABLE NMI INTERRUPTS
E0B2 E6A0               326             OUT     0A0H,AL
E0B4 E683               327             OUT     83H,AL                          ;INITIALZE DMA PAGE REG
E0B6 B099               328             MOV     AL,99H                          ;SET 8255 A,C-INPUT,B-OUTPUT
E0B8 E663               329             OUT     CMD_PORT,AL                     ;WRITE 8255 CMD/MODE REG
E0BA B0FC               330             MOV     AL,0FCH                         ;DISABLE PARITY CHECKERS AND
E0BC E661               331             OUT     PORT_B,AL                       ; GATE SNS SW3,CASS MOTOR OFF
E0BE 2AC0               332             SUB     AL,AL                           ;
E0C0 BAD803             333             MOV     DX,3D8H
E0C3 EE                 334             OUT     DX,AL                           ;DISABLE COLOR VIDEO
E0C4 FEC0               335             INC     AL
E0C6 BAB803             336             MOV     DX,3B8H
E0C9 EE                 337             OUT     DX,AL                           ;DISABLE B/W VIDEO,EN HIGH RES
E0CA B800F0         R   338             MOV     AX,CODE                         ;SETUP SS SEG REG
E0CD 8ED0               339             MOV     SS,AX
E0CF BB00E0             340             MOV     BX,0E000H                       ;SETUP STARTING ROS ADDR
E0D2 BC16E0         R   341             MOV     SP,OFFSET C1                    ;SETUP RETURN ADDRESS
E0D5 E93301             342             JMP     ROS_CHECKSUM
E0D8 7505               343     C11:    JNE     ERR01                           ;HALT SYSTEM IF ERROR
                        344     ;-----------------------------------------
                        345     ;TEST.03
                        346     ;       8237 DMA INITIALIZATION CHANNEL REGISTER TEST
                        347     ;DESCRIPTION
                        348     ;       DISABLE THE 8237 DMA CONTROLLER.  VERIFY THAT TIMER 1 FUNCTIONS OK.
                        349     ;       WRITE/READ THE CURRENT ADDRESS AND WORD COUNT REGISTERS FOR ALL
                        350     ;       CHANNELS.  INITIALIZE AND START DMA FOR MEMORY REFRESH.
                        351     ;-----------------------------------------
                        352     ;       DISABLE DMA CONTROLLER
                        353
E0DA B004               354             MOV     AL,04                           ;DISABLE DMA CONTROLLER
E0DC E608               355             OUT     DMA08,AL
                        356
                        357     ;       VERIFY THAT TIMER 1 FUNCTIONS OK
                        358
E0DE B054               359             MOV     AL,54H                          ;SEL TIMER 1,LSB,MODE 2
E0E0 E643               360             OUT     TIMER+3,AL
E0E2 2BC9               361             SUB     CX,CX                           ;
E0E4 8AD9               362             MOV     BL,CL
E0E6 8AC1               363             MOV     AL,CL                           ;SET INITIAL TIMER CNT TO 0
E0E8 E641               364             OUT     TIMER+1,AL
E0EA                    365     C12:                                            ; TIMER1_BITS_ON
E0EA B040               366             MOV     AL,40H                          ;LATCH TIMER 1 COUNT
E0EC E643               367             OUT     TIMER+3,AL
E0EE E441               368             IN      AL,TIMER+1                      ;READ TIMER 1 COUNT
E0F0 0AD8               369             OR      BL,AL                           ;ALL BITS ON IN TIMER
E0F2 80FBFF             370             CMP     BL,0FFH                         ;YES - SEE IF ALL BITS GO OFF
E0F5 7404               371             JE      C13                             ; TIMER1_BITS_OFF
E0F7 E2F1               372             LOOP    C12                             ; TIMER1_BITS_ON
E0F9 EBB4               373             JMP     SHORT ERR01                     ;TIMER 1 FAILURE, HALT SYS
E0FB                    374     C13:                                            ; TIMER1_BITS_OFF
E0FB 8AC3               375             MOV     AL,BL                           ;SET TIMER 1 CNT
E0FD 2BC9               376             SUB     CX,CX
E0FF E641               377             OUT     TIMER+1,AL
E101                    378     C14:                                            ; TIMER_LOOP
E101 B040               379             MOV     AL,40H                          ;LATCH TIMER 1 COUNT
E103 E643               380             OUT     TIMER+3,AL
E105 E441               381             IN      AL,TIMER+1                      ;READ TIMER 1 COUNT
E107 22D8               382             AND     BL,AL
E109 7404               383             JZ      C15                             ; WRAP_DMA_REG
E10B E2F4               384             LOOP    C14                             ; TIMER_LOOP
```

```
LOC OBJ                 LINE    SOURCE

E10D EBA0               385             JMP     SHORT ERR01
                        386
                        387     ;       INITIALIZE TIMER 1 TO REFRESH MEMORY
                        388
E10F                    389     C15:                                    ; WRAP_DMA_REG
E10F B054               390             MOV     AL,54H                  ;SEL TIM 1, LSB, MODE 2
E111 E643               391             OUT     TIMER+3,AL              ;WRITE TIMER MODE REG
E113 B012               392             MOV     AL,18                   ;SETUP DIVISOR FOR REFRESH
E115 E641               393             OUT     TIMER+1,AL              ;WRITE TIMER 1 CNT REG
E117 E60D               394             OUT     DMA+0DH,AL              ;SEND MASTER CLEAR TO DMA
                        395
                        396     ;       WRAP DMA CHANNELS ADDRESS AND COUNT REGISTERS
                        397
E119 B0FF               398             MOV     AL,0FFH                 ;WRITE PATTERN FFH TO ALL REGS
E11B 8AD8               399     C16:    MOV     BL,AL                   ;SAVE PATTERN FOR COMPARE
E11D 8AF8               400             MOV     BH,AL
E11F B90800             401             MOV     CX,8                    ;SETUP LOOP CNT
E122 BA0000             402             MOV     DX,DMA                  ;SETUP I/O PORT ADDR OF REG
E125 EE                 403     C17:    OUT     DX,AL                   ;WRITE PATTERN TO REG, LSB
E126 EE                 404             OUT     DX,AL                   ;MSB OF 16 BIT REG
E127 B80101             405             MOV     AX,0101H                ;AX TO ANOTHER PAT BEFORE RD
E12A EC                 406             IN      AL,DX                   ;READ 16-BIT DMA CH REG, LSB
E12B 8AE0               407             MOV     AH,AL                   ;SAVE LSB OF 16-BIT REG
E12D EC                 408             IN      AL,DX                   ;READ MSB OF DMA CH REG
E12E 3BD8               409             CMP     BX,AX                   ;PATTERN READ AS WRITTEN?
E130 7403               410             JE      C18                     ;YES - CHECK NEXT REG
E132 E97AFF             411             JMP     ERR01                   ;NO - HALT THE SYSTEM
E135                    412     C18:                                    ; NXT_DMA_CH
E135 42                 413             INC     DX                      ;SET I/O PORT TO NEXT CH REG
E136 E2ED               414             LOOP    C17                     ;WRITE PATTERN TO NEXT REG
E138 F6D0               415             NOT     AL                      ;SET PATTERN TO ZERO
E13A 74DF               416             JZ      C16                     ;WRITE TO CHANNEL REGS
                        417
                        418     ;       INITIALIZE AND START DMA FOR MEMORY REFRESH.
                        419
E13C B0FF               420             MOV     AL,0FFH                 ;SET CNT OF 64K FOR RAM REFRESH
E13E E601               421             OUT     DMA+1,AL
E140 E601               422             OUT     DMA+1,AL
E142 B058               423             MOV     AL,058H                 ;SET DMA MODE,CH 0,READ,AUOTINT
E144 E60B               424             OUT     DMA+0BH,AL              ;WRITE DMA MODE REG
E146 B000               425             MOV     AL,0                    ;ENABLE DMA CONTROLLER
E148 E608               426             OUT     DMA+8,AL                ;SETUP DMA COMMAND REG
E14A E60A               427             OUT     DMA+10,AL               ;ENABLE CHANNEL 0 FOR REFRESH
E14C B041               428             MOV     AL,41H                  ;SET MODE FOR CHANNEL 1
E14E E60B               429             OUT     DMA+0BH,AL
E150 B042               430             MOV     AL,42H                  ;SET MODE FOR CHANNEL 2
E152 E60B               431             OUT     DMA+0BH,AL              ;
E154 B043               432             MOV     AL,43H                  ;SET MODE FOR CHANNEL 3
E156 E60B               433             OUT     DMA+0BH,AL
                        434     ;-----------------------------------------
                        435     ;TEST.04
                        436     ;       BASE 16K READ/WRITE STORAGE TEST
                        437     ;
                        438     ;DESCRIPTION
                        439     ;       WRITE/READ/VERIFY DATA PATTERNS FF,55,AA,01, AND 00 TO 1ST 16K OF
                        440     ;       STORAGE.  VERIFY STORAGE ADDRESSABILITY.
                        441     ;       INITIALIZE THE 8259 INTERRUPT CONTROLLER CHIP FOR CHECKING
                        442     ;       MANUFACTURING TEST 2 MODE.
                        443     ;-----------------------------------------
                        444     ;       DETERMINE MEMORY SIZE AND FILL MEMORY WITH DATA
                        445
E158 B84000         R   446             MOV     AX,DATA                 ;POINT DS TO DATA SEG
E15B 8ED8               447             MOV     DS,AX                   ;
E15D 8B1E7200       R   448             MOV     BX,RESET_FLAG           ;SAVE  RESET_FLAG' IN BX
E161 2BC0               449             SUB     AX,AX                   ;SET ES AND DS TO 0
E163 8EC0               450             MOV     ES,AX                   ;SETUP ES SEGMENT REG
E165 8ED8               451             MOV     DS,AX
E167 2BFF               452             SUB     DI,DI
E169 E460               453             IN      AL,PORT_A               ;DETERMINE BASE RAM SIZE
E16B 240C               454             AND     AL,0CH                  ;ISOLATE RAM SIZE SWS
E16D 0404               455             ADD     AL, 4                   ; CALCULATE MEMORY SIZE
E16F B10C               456             MOV     CL, 12
E171 D3E0               457             SHL     AX, CL
E173 8BC8               458             MOV     CX, AX
E175 8AE0               459             MOV     AH, AL
E177 FC                 460             CLD                             ;SET DIR FLAG TO INCR
E178 AA                 461     C19:    STOSB                           ;FILL BASE RAM WITH DATA
```

```
LOC OBJ                 LINE    SOURCE


E179 E2FD               462             LOOP    C19                             ; LOOP TIL ALL ZERO
                        463
                        464     ;       DETERMINE IO CHANNEL RAM SIZE
                        465
E17B E462               466             IN      AL,PORT_C
E17D 240F               467             AND     AL,0FH
E17F 7418               468             JZ      C21
E181 BA0010             469             MOV     DX,1000H                        ; SEGMENT FOR I/O RAM
E184 8AE0               470             MOV     AH,AL
E186 B000               471             MOV     AL,0
E188                    472     C20:                                            ; FILL_IO:
E188 8EC2               473             MOV     ES,DX
E18A B90080             474             MOV     CX,8000H                        ; FILL 32K BYTES
E18D 2BFF               475             SUB     DI,DI
E18F F3                 476             REP     STOSB
E190 AA
E191 81C20008           477             ADD     DX,800H                         ; NEXT SEGMENT VALUE
E195 FECC               478             DEC     AH
E197 75EF               479             JNZ     C20                             ; FILL_IO
                        480     ;-----------------------------------------
                        481     ;       INITIALIZE THE 8259 INTERRUPT CONTROLLER CHIP
                        482     ;-----------------------------------------
E199                    483     C21:
E199 B013               484             MOV     AL,13H                          ;ICW1 - EDGE, SNGL, ICW4
E19B E620               485             OUT     INTA00,AL
E19D B008               486             MOV     AL,8                            ;SETUP ICW2 - INT TYPE 8 (8-F)
E19F E621               487             OUT     INTA01,AL
E1A1 B009               488             MOV     AL,9                            ;SETUP ICW4 - BUFFRD,8086 MODE
E1A3 E621               489             OUT     INTA01,AL
E1A5 2BC0               490             SUB     AX,AX                           ;POINT DS AND ES TO BEGIN
E1A7 8EC0               491             MOV     ES,AX                           ;  OF R/W STORAGE
E1A9 BE4000         R   492             MOV     SI,DATA                         ;POINT DS TO DATA SEG
E1AC 8EDE               493             MOV     DS,SI                           ;
E1AE 891E7200       R   494             MOV     RESET_FLAG,BX                   ;RESTORE RESET_FLAG
E1B2 813E72003412   R   495             CMP     RESET_FLAG,1234H                ;RESET_FLAG SET?
E1B8 7438               496             JE      C25                             ;YES - SKIP STG TEST
E1BA 8ED8               497             MOV     DS,AX                           ;POINT DS TO 1ST 16K OF STG
                        498     ;-----------------------------------------
                        499     ;       CHECK FOR MANUFACTURING TEST 2 TO LOAD TEST PROGRAMS FROM KEYBOARD.
                        500     ;-----------------------------------------
E1BC BCF03F             501             MOV     SP, 3FF0H                       ; ESTABLISH TEMPORARY STACK
E1BF 8ED0               502             MOV     SS, AX
E1C1 8BF8               503             MOV     DI, AX
E1C3 BB2400             504             MOV     BX, 24H
E1C6 C707B6E2       R   505             MOV     WORD PTR [BX],OFFSET D11        ;SET UP KB INTERRUPT
E1CA 43                 506             INC     BX
E1CB 43                 507             INC     BX
E1CC 8C0F               508             MOV     [BX],CS
E1CE E8B704             509             CALL    KBD_RESET               ; READ IN KB RESET CODE TO BL
E1D1 80FB65             510             CMP     BL,065H                 ; IS THIS MANUFACTURING TEST 2?
E1D4 750E               511             JNZ     C23                     ; JUMP IF NOT MAN. TEST
E1D6 B2FF               512             MOV     DL,255                  ; READ IN TEST PROGRAM
E1D8 E8BA04             513     C22:    CALL    SP_TEST
E1DB 8AC3               514             MOV     AL,BL
E1DD AA                 515             STOSB
E1DE FECA               516             DEC     DL
E1E0 75F6               517             JNZ     C22                     ; JUMP IF NOT DONE YET
E1E2 CD3E               518             INT     3EH                     ;SET INTERUPT TYPE 62 ADDRESS F8H
E1E4                    519     C23:                                    ;CONTINUE IN NORMAL MODE
E1E4 0E                 520             PUSH    CS                      ; PUT SS BACK
E1E5 17                 521             POP     SS
E1E6 FA                 522             CLI                             ;
E1E7 BC18E0         R   523             MOV     SP,OFFSET C2                    ;SETUP RETURN ADDRESS
E1EA E92DFE             524             JMP     STGTST                          ;GO TO RD/WRT STG SUBROUTINE
E1ED 7403               525     C24:    JE      C25                             ;GO TO NEXT TEST IF OK
E1EF E9BDFE             526             JMP     ERR01
                        527
                        528     ;       SETUP STACK SEG AND SP
                        529
E1F2                    530     C25:
E1F2 B83000             531             MOV     AX,STACK                        ; GET STACK VALUE
E1F5 8ED0               532             MOV     SS,AX                           ; SET THE STACK UP
E1F7 BC0001             533             MOV     SP,OFFSET TOS                   ; STACK IS READY TO GO
                        534
                        535     ;       SETUP THE NMI INTERRUPT VECTOR POINTER
                        536
```

```
LOC OBJ                 LINE    SOURCE

E1FA 26C7060800C3E2 R    537            MOV     ES:NMI_PTR,OFFSET NMI_INT
E201 26C7060A0000F0 R    538            MOV     ES:NMI_PTR+2,CODE
E208 E92A00              539            JMP     TST6                            ;GO TO NEXT TEST
                         540
E20B                     541    ROS_CHECKSUM    PROC    NEAR                    ; NEXT_ROS_MODULE
E20B B90020              542            MOV     CX,8192                         ;NUMBER OF BYTES TO ADD
E20E 32C0                543            XOR     AL,AL
E210                     544    C26:
E210 2E0207              545            ADD     AL,CS:[BX]                      ;
E213 43                  546            INC     BX                              ;POINT TO NEXT BYTE
E214 E2FA                547            LOOP    C26                             ;ADD ALL BYTES IN ROS MODULE
E216 0AC0                548            OR      AL,AL                           ;SUM = 0?
E218 C3                  549            RET
                         550    ROS_CHECKSUM    ENDP
                         551    ;---------------------------------------
                         552    ; INITIAL RELIABILITY TEST -- PHASE 2
                         553    ;---------------------------------------
                         554            ASSUME  CS:CODE,ES:ABS0
                         555
E219 50415249545920      556    D1      DB      'PARITY CHECK 2'
     4348454434B2032
  000E                   557    D1L     EQU     $-D1
E227 50415249545920      558    D2      DB      'PARITY CHECK 1'
     4348454434B2031
  000E                   559    D2L     EQU     $-D2
                         560    ;---------------------------------------------
                         561    ;TEST.06
                         562    ;       8259 INTERRUPT CONTROLLER TEST
                         563    ;DESCRIPTION
                         564    ;       READ/WRITE THE INTERRUPT MASK REGISTER (IMR) WITH ALL ONES AND ZEROES.
                         565    ;       ENABLE SYSTEM INTERRUPTS.  MASK DEVICE INTERRUPTS OFF.  CHECK FOR
                         566    ;       HOT INTERRUPTS (UNEXPECTED).
                         567    ;---------------------------------------------
E235                     568    TST6:
E235 2BC0                569            SUB     AX,AX                           ;SET UP ES REG
E237 8EC0                570            MOV     ES,AX
                         571
                         572    ;------ SET UP THE INTERRUPT 5 POINTER TO A DUMMY
                         573
E239 26C706140054FF R    574            MOV     ES:INT5_PTR,OFFSET PRINT_SCREEN         ;PRINT SCREEN
E240 26C706160000F0 R    575            MOV     ES:INT5_PTR+2,CODE                      ;
                         576
                         577    ;       TEST THE IMR REGISTER
                         578
E247 FA                  579            CLI                                     ;DIABLE INTERRUPTS
E248 B000                580            MOV     AL,0                            ;SET IMR TO ZERO
E24A E621                581            OUT     INTA01,AL
E24C E421                582            IN      AL,INTA01                       ;READ IMR
E24E 0AC0                583            OR      AL,AL                           ;IMR = 0?
E250 752B                584            JNZ     D6                              ;GO TO ERR ROUTINE IF NOT 0
E252 B0FF                585            MOV     AL,0FFH                         ;DISABLE DEVICE INTERRUPTS
E254 E621                586            OUT     INTA01,AL                       ;WRITE TO IMR
E256 E421                587            IN      AL,INTA01                       ;READ IMR
E258 0401                588            ADD     AL,1                            ;ALL IMR BIT ON?
E25A 7521                589            JNZ     D6                              ;NO - GO TO ERR ROUTINE
                         590
                         591    ;       CHECK FOR HOT INTERRUPTS
                         592
E25C FC                  593            CLD                                     ;SET DIR FLAG TO GO FORWARD
E25D B90800              594            MOV     CX,8                            ;SETUP TEMP INT RTNE IN PRT TBL
E260 BF2000              595            MOV     DI,OFFSET INT_PTR               ;GET ADDRESS OF INT PROC TABLE
E263                     596    D3:                                             ; VECTBL0:
E263 B8B6E2          R   597            MOV     AX,OFFSET D11                   ;MOVE ADDR OF INTR PROC TO TBL
E266 AB                  598            STOSW
E267 B800F0          R   599            MOV     AX,CODE                         ;GET ADDR OF INTR PROC SEG
E26A AB                  600            STOSW
E26B 83C304              601            ADD     BX,4                            ;SET BX TO POINT TO NEXT VAL
E26E E2F3                602            LOOP    D3                              ; VECTBL0
                         603
                         604    ;       INTERRUPTS ARE MASKED OFF.  CHECK THAT NO INTERRUPTS OCCUR.
                         605
E270 32E4                606            XOR     AH,AH                           ;CLEAR AH REG
E272 FB                  607            STI                                     ;ENABLE EXTERNAL INTERRUPTS
E273 2BC9                608            SUB     CX,CX                           ;WAIT 1 SEC FOR ANY INTRS THAT
E275 E2FE                609    D4:     LOOP    D4                              ;MIGHT OCCUR
E277 E2FE                610    D5:     LOOP    D5
E279 0AE4                611            OR      AH,AH                           ;DID ANY INTERRUPTS OCCUR?
```

```
LOC OBJ                 LINE    SOURCE


E27B 7408               612             JZ      D7                              ;NO - GO TO NEXT TEST
E27D BA0101             613     D6:     MOV     DX,101H                         ;BEEP SPEAKER IF ERROR
E280 E8AD03             614             CALL    ERR_BEEP                        ;GO TO BEEP SUBROUTINE
E283 FA                 615             CLI
E284 F4                 616             HLT                                     ;HALT THE SYSTEM
                        617     ;-----------------------------------------
                        618     ;TEST.7
                        619     ;       8253 TIMER CHECKOUT
                        620     ;DESCRIPTION
                        621     ;       VERIFY THAT THE SYSTEM TIMER (0) DOESN'T COUNT TOO FAST NOR TOO
                        622     ;       SLOW.
                        623     ;-----------------------------------------
E285                    624     D7:
E285 B400               625             MOV     AH,0                    ;RESET TIMER INTR RECVD FLAG
E287 32ED               626             XOR     CH,CH                   ;CLEAR THE CH REG
E289 B0FE               627             MOV     AL,0FEH                 ;MASK ALL INTRS EXCEPT LVL 0
E28B E621               628             OUT     INTA01,AL               ;WRITE THE 8259 IMR
E28D B010               629             MOV     AL,00010000B            ;SEL TIM 0, LSB, MODE 0, BINARY
E28F E643               630             OUT     TIM_CTL,AL              ;WRITE TIMER CONTROL MODE REG
E291 B116               631             MOV     CL,16H                  ;SET PGM LOOP CNT
E293 8AC1               632             MOV     AL,CL                   ;SET TIMER 0 CNT REG
E295 E640               633             OUT     TIMER0,AL               ;WRITE TIMER 0 CNT REG
E297 F6C4FF             634     D8:     TEST    AH,0FFH                 ;DID TIMER 0 INTERRUPT OCCUR?
E29A 7504               635             JNZ     D9                      ; YES - CHECK TIMER OP FOR SLOW TIME
E29C E2F9               636             LOOP    D8                      ;WAIT FOR INTR FOR SPECIFIED TIME
E29E EBDD               637             JMP     D6                      ;TIMER 0 INTR DIDN T OCCUR - ERR
E2A0 B112               638     D9:     MOV     CL,18                   ;SET PGM LOOP CNT
E2A2 B0FF               639             MOV     AL,0FFH                 ;WRITE TIMER 0 CNT REG
E2A4 E640               640             OUT     TIMER0,AL
E2A6 B400               641             MOV     AH,0                    ;RESET INTR RECEIVED FLAG
E2A8 B0FE               642             MOV     AL,0FEH                 ;REENABLE TIMER 0 INTERRUTS
E2AA E621               643             OUT     INTA01,AL
E2AC F6C4FF             644     D10:    TEST    AH,0FFH                 ;DID TIMER 0 INTERRUPT OCCUR?
E2AF 75CC               645             JNZ     D6                      ;YES - TIMER CNTING TOO FAST, ERR
E2B1 E2F9               646             LOOP    D10                     ;WAIT FOR INTR FOR SPECIFIED TIME
E2B3 E93600             647             JMP     TST8                    ;GO TO NEXT TEST ROUTINE
                        648     ;-----------------------------------------
                        649     ;       TEMPORARY INTERRUPT SERVICE ROUTINE
                        650     ;-----------------------------------------
E2B6                    651     D11     PROC    NEAR
E2B6 B401               652             MOV     AH,1
E2B8 50                 653             PUSH    AX                              ;SAVE REG AX CONTENTS
E2B9 B0FF               654             MOV     AL,0FFH                         ;MASK ALL INTERRUPTS OFF
E2BB E621               655             OUT     INTA01,AL
E2BD B020               656             MOV     AL,EOI
E2BF E620               657             OUT     INTA00,AL
E2C1 58                 658             POP     AX                              ;RESTORE REG AX CONTENTS
E2C2 CF                 659             IRET
                        660     D11     ENDP
                        661
E2C3                    662     NMI_INT PROC    NEAR
E2C3 50                 663             PUSH    AX                              ;SAVE ORIG CONTENTS OF AX
E2C4 E462               664             IN      AL,PORT_C
E2C6 A840               665             TEST    AL,40H                          ;IO CH PARITY CHECK?
E2C8 7408               666             JZ      D12                             ;YES - FLAG IS SET TO 0
E2CA BE19E2         R   667             MOV     SI,OFFSET D1                    ;ADDR OF ERROR MSG
E2CD B90E00             668             MOV     CX,D1L                          ;MSG LENGTH
E2D0 EB0A               669             JMP     SHORT D13                       ;DISPLAY ERROR MSG
E2D2                    670     D12:
E2D2 A880               671             TEST    AL,80H                          ;PLANAR RAM P-CHECK?
E2D4 7410               672             JZ      D14                             ;NO - AUX INT
E2D6 BE27E2         R   673             MOV     SI,OFFSET D2                    ;ADDR OF ERROR MSG
E2D9 B90E00             674             MOV     CX,D2L                          ;MSG LENGTH
E2DC                    675     D13:
E2DC B80000             676             MOV     AX,0                            ;INIT AND SET MODE FOR VIDEO
E2DF CD10               677             INT     10H                             ;CALL VIDEO_IO PROCEDURE
E2E1 E8E603             678             CALL    P_MSG                           ;PRINT ERROR MSG
E2E4 FA                 679             CLI
E2E5 F4                 680             HLT                                     ;HALT SYSTEM
E2E6                    681     D14:
E2E6 58                 682             POP     AX                              ;RESTORE ORIG CONTENTS OF AX
E2E7 CF                 683             IRET
                        684     NMI_INT ENDP
```

```
LOC OBJ                 LINE    SOURCE

                        685     ;--------------------------------------
                        686     ; INITIAL RELIABILITY TEST -- PHASE 3
                        687     ;--------------------------------------
                        688             ASSUME  CS:CODE,DS:DATA
                        689
E2E8 20323031           690     E1      DB      ' 201'
  0004                  691     E1L     EQU     $-E1
                        692
                        693     ;       ESTABLISH BIOS SUBROUTINE CALL INTERRUPT VECTORS
                        694
E2EC                    695     TST8:
E2EC FC                 696             CLD                             ;SET DIR FLAG TO GO FORWARD
E2ED BF4000             697             MOV     DI,OFFSET VIDEO_INT     ;SETUP ADDR TO INTR AREA
E2F0 0E                 698             PUSH    CS
E2F1 1F                 699             POP     DS                      ;SETUP ADDR OF VECTOR TABLE
E2F2 BE13FF             700             MOV     SI,0FF13H               ; OFFSET VECTOR_TABLE+32
E2F5 B92000             701             MOV     CX,20H
E2F8 F3                 702             REP     MOVSW                   ;MOVE VECTOR TABLE TO RAM
E2F9 A5
                        703
                        704     ;       SETUP TIMER 0 TO MODE 3
                        705
E2FA B0FF               706             MOV     AL,0FFH                 ;DISABLE ALL DEVICE INTERRUPTS
E2FC E621               707             OUT     INTA01,AL
E2FE B036               708             MOV     AL,36H                  ;SEL TIM 0,LSB,MSB,MODE 3
E300 E643               709             OUT     TIMER+3,AL              ;WRITE TIMER MODE REG
E302 B000               710             MOV     AL,0
E304 E640               711             OUT     TIMER,AL                ;WRITE LSB TO TIMER 0 REG
E306 E640               712             OUT     TIMER,AL                ;WRITE MSB TO TIMER 0 REG
                        713
                        714     ;       SETUP TIMER 0 TO BLINK LED IF MANUFACTURING TEST MODE
                        715
                        716             ASSUME  DS:DATA
E308 B84000         R   717             MOV     AX,DATA                 ;POINT DS TO DATA SEG
E30B 8ED8               718             MOV     DS,AX
E30D E87803             719             CALL    KBD_RESET               ;SEND SOFTWARE RESET TO KEYBRD
E310 80FBAA             720             CMP     BL,0AAH                 ;SCAN CODE  AA' RETURNED?
E313 7426               721             JE      E3                      ;YES - CONTINUE (NON MFG MODE)
E315 B03C               722             MOV     AL,3CH                  ;EN KBD, SET KBD CLK LINE LOW
E317 E661               723             OUT     PORT_B,AL               ;WRITE 8255 PORT B
E319 90                 724             NOP
E31A 90                 725             NOP
E31B E460               726             IN      AL,PORT_A               ;WAS A BIT CLOCKED IN?
E31D 24FF               727             AND     AL,0FFH
E31F 7516               728             JNZ     E2                      ;YES - CONTINUE (NON MFG MODE)
E321 FE061200       R   729             INC     MFG_TST                 ;ELSE SET SW FOR MFG TEST MODE
E325 26C7062000B2E6 R   730             MOV     ES:INT_ADDR,OFFSET BLINK_INT    ;SETUP TIMER INTR TO BLINK LED
E32C 26C706220000F0 R   731             MOV     ES:INT_ADDR+2,CODE
E333 B0FE               732             MOV     AL,0FEH                 ;ENABLE TIMER INTERRUPT
E335 E621               733             OUT     INTA01,AL
E337                    734     E2:                                     ; JUMPER_NOT_IN:
E337 B0CC               735             MOV     AL,0CCH                 ;RESET THE KEYBOARD
E339 E661               736             OUT     PORT_B,AL
                        737     ;-----------------------------------------
                        738     ;TEST.05
                        739     ;       ROS CHECKSUM II
                        740     ;DESCRIPTION
                        741     ;       A CHECKSUM IS DONE FOR THE 4 ROS MODULES CONTAINING BASIC CODE
                        742     ;-----------------------------------------
E33B                    743     E3:
E33B B204               744             MOV     DL,4                    ;NO. OF ROS MODULES TO CHECK
E33D BB0060             745             MOV     BX,6000H                ;SETUP STARTING ROS ADDR
E340                    746     E4:                                     ; CHECK_ROS:
E340 E8C8FE             747             CALL    ROS_CHECKSUM
E343 7507               748             JNE     E5                      ;BEEP SPEAKER IF ERROR
E345 FECA               749             DEC     DL                      ;ANY MORE TO DO?
E347 75F7               750             JNZ     E4                      ;YES - CONTINUE
E349 EB0790             751             JMP     E6                      ;NO - GO TO NEXT TEST
E34C                    752     E5:                                     ; ROS_ERROR:
E34C BA0101             753             MOV     DX,101H
E34F E8DE02             754             CALL    ERR_BEEP                ;BEEP SPEAKER
```

```
LOC OBJ                  LINE   SOURCE

                         755    ;-----------------------------------------
                         756    ;TEST.08
                         757    ;       INITIALIZE AND START CRT CONTROLLER (6845)
                         758    ;       TEST VIDEO READ/WRITE STORAGE.
                         759    ;DESCRIPTION
                         760    ;       RESET THE VIDEO ENABLE SIGNAL.
                         761    ;       SELECT ALPHANUMERIC MODE, 40 * 25, B & W.
                         762    ;       READ/WRITE DATA PATTERNS TO STG. CHECK STG ADDRESSABILITY.
                         763    ;-----------------------------------------
E352                     764    E6:
E352 E460                765            IN      AL,PORT_A                       ;READ SENSE SWITCHES
E354 B400                766            MOV     AH,0
E356 A31000        R     767            MOV     EQUIP_FLAG,AX                   ;STORE SENSE SW INFO
E359 2430                768            AND     AL,30H                          ;ISOLATE VIDEO SWS
E35B 7503                769            JNZ     E7                              ;VIDEO SWS SET TO 0?
E35D E99800              770            JMP     E19                             ;SKIP VIDEO TESTS FOR BURN-IN
E360                     771    E7:                                             ; TEST_VIDEO:
E360 86E0                772            XCHG    AH,AL
E362 80FC30              773            CMP     AH,30H                          ;B/W CARD ATTACHED?
E365 7409                774            JE      E8                              ;YES - SET MODE FOR B/W CARD
E367 FEC0                775            INC     AL                              ;SET COLOR MODE FOR COLOR CD
E369 80FC20              776            CMP     AH,20H                          ;80X25 MODE SELECTED?
E36C 7502                777            JNE     E8                              ;NO - SET MODE FOR 40X25
E36E B003                778            MOV     AL,3                            ;SET MODE FOR 80X25
E370                     779    E8:                                             ; SET_MODE:
E370 50                  780            PUSH    AX                              ;SAVE VIDEO MODE ON STACK
E371 2AC4                781            SUB     AH,AH                           ;INITIALIZE TO ALPHANUMERIC MD
E373 CD10                782            INT     10H                             ;CALL VIDEO_IO
E375 58                  783            POP     AX                              ;RESTORE VIDEO SENSE SWS IN AH
E376 50                  784            PUSH    AX                              ; RESAVE VALUE
E377 BB00B0              785            MOV     BX,0B000H                       ;BEG VIDEO RAM ADDR B/W CD
E37A BAB803              786            MOV     DX,3B8H                         ;MODE REG FOR B/W
E37D B90010              787            MOV     CX,4096                         ;RAM BYTE CNT FOR B/W CD
E380 B001                788            MOV     AL,1                            ; SET MODE FOR BW CARD
E382 80FC30              789            CMP     AH,30H                          ;B/W VIDEO CARD ATTACHED?
E385 740B                790            JE      E9                              ;YES - GO TEST VIDEO STG
E387 BB00B8              791            MOV     BX,0B800H                       ;BEG VIDEO RAM ADDR COLOR CD
E38A BAD803              792            MOV     DX,3D8H                         ;MODE REG FOR COLOR CD
E38D B90040              793            MOV     CX,4000H                        ;RAM BYTE CNT FOR COLOR CD
E390 FEC8                794            DEC     AL                              ; SET MODE TO 0 FOR COLOR CD
E392                     795    E9:                                             ; TEST_VIDEO_STG:
E392 EE                  796            OUT     DX,AL                           ;DISABLE VIDEO FOR COLOR CD
E393 8EC3                797            MOV     ES,BX                           ;POINT ES TO VIDEO RAM STG
E395 B84000        R     798            MOV     AX,DATA                         ;POINT DS TO DATA SEGMENT
E398 8ED8                799            MOV     DS,AX
E39A 813E72003412  R     800            CMP     RESET_FLAG,1234H                ;POD INITIATED BY KBD RESET?
E3A0 740D                801            JE      E10                             ;YES - SKIP VIDEO RAM TEST
E3A2 8EDB                802            MOV     DS,BX                           ;POINT DS TO VIDEO RAM STG
E3A4 E876FC              803            CALL    STGTST_CNT                      ;GO TEST VIDEO R/W STG
E3A7 7406                804            JE      E10                             ;STG OK - CONTINUE TESTING
E3A9 BA0201              805            MOV     DX,102H                         ;SETUP # OF BEEPS
E3AC E88102              806            CALL    ERR_BEEP                        ;GO BEEP SPEAKER
                         807    ;-----------------------------------------
                         808    ;TEST.09
                         809    ;       SETUP VIDEO DATA ON SCREEN FOR VIDEO LINE TEST.
                         810    ;DESCRIPTION
                         811    ;       ENABLE VIDEO SIGNAL AND SET MODE.
                         812    ;       DISPLAY A HORIZONTAL BAR ON SCREEN.
                         813    ;-----------------------------------------
E3AF                     814    E10:
E3AF 58                  815            POP     AX                              ;GET VIDEO SENSE SWS (AH)
E3B0 50                  816            PUSH    AX                              ;SAVE IT
E3B1 B400                817            MOV     AH,0                            ;ENABLE VIDEO AND SET MODE
E3B3 CD10                818            INT     10H                             ; VIDEO
E3B5 B82070              819            MOV     AX,7020H                        ;WRT BLANKS IN REVERSE VIDEO
E3B8 2BFF                820            SUB     DI,DI                           ;SETUP STARTING LOC
E3BA B92800              821            MOV     CX,40                           ;NO. OF BLANKS TO DISPLAY
E3BD FC                  822            CLD                                     ;SET DIR FLAG TO INCREMENT
E3BE F3                  823            REP     STOSW                           ;WRITE VIDEO STORAGE
E3BF AB
```

```
LOC OBJ                 LINE    SOURCE

                        824     ;--------------------------------------------
                        825     ;TEST.10
                        826     ;       CRT INTERFACE LINES TEST
                        827     ;DESCRIPTION
                        828     ;       SENSE ON/OFF TRANSITION OF THE VIDEO ENABLE AND HORIZONTAL
                        829     ;       SYNC LINES.
                        830     ;--------------------------------------------
E3C0 58                 831             POP     AX                      ;GET VIDEO SENSE SW INFO
E3C1 50                 832             PUSH    AX                      ;SAVE IT
E3C2 80FC30             833             CMP     AH,30H                  ;B/W CARD ATTACHED?
E3C5 BABA03             834             MOV     DX,03BAH                ;SETUP ADDR OF BW STATUS PORT
E3C8 7403               835             JE      E11                     ;YES - GO TEST LINES
E3CA BADA03             836             MOV     DX,03DAH                ;COLOR CARD IS ATTACHED
E3CD                    837     E11:                                    ; LINE_TST:
E3CD B408               838             MOV     AH,8
E3CF                    839     E12:                                    ; OFLOOP_CNT:
E3CF 2BC9               840             SUB     CX,CX
E3D1 EC                 841     E13:    IN      AL,DX                   ;READ CRT STATUS PORT
E3D2 22C4               842             AND     AL,AH                   ;CHECK VIDEO/HORZ LINE
E3D4 7504               843             JNZ     E14                     ;ITS ON - CHECK IF IT GOES OFF
E3D6 E2F9               844             LOOP    E13                     ;LOOP TILL ON OR TIMEOUT
E3D8 EB13               845             JMP     SHORT E17               ;GO PRINT ERROR MSG
E3DA 2BC9               846     E14:    SUB     CX,CX
E3DC EC                 847     E15:    IN      AL,DX                   ;READ CRT STATUS PORT
E3DD 22C4               848             AND     AL,AH                   ;CHECK VIDEO/HORZ LINE
E3DF 7404               849             JZ      E16                     ;ITS ON - CHECK NEXT LINE
E3E1 E2F9               850             LOOP    E15                     ;LOOP IF OFF TILL IT GOES ON
E3E3 EB08               851             JMP     SHORT E17
E3E5                    852     E16:                                    ; NXT_LINE:
E3E5 B103               853             MOV     CL,3                    ;GET NEXT BIT TO CHECK
E3E7 D2EC               854             SHR     AH,CL                   ;
E3E9 75E4               855             JNZ     E12                     ;GO CHECK HORIZONTAL LINE
E3EB EB06               856             JMP     SHORT E18               ;DISPLAY CURSOR ON SCREEN
E3ED                    857     E17:                                    ; CRT_ERR:
E3ED BA0201             858             MOV     DX,102H
E3F0 E83D02             859             CALL    ERR_BEEP                ;GO BEEP SPEAKER
E3F3                    860     E18:                                    ; DISPLAY_CURSOR:
E3F3 58                 861             POP     AX                      ;GET VIDEO SENSE SWS (AH)
E3F4 B400               862             MOV     AH,0                    ;SET MODE AND DISPLAY CURSOR
E3F6 CD10               863             INT     10H                     ;CALL VIDEO I/O PROCEDURE
                        864     ;--------------------------------------------
                        865     ;TEST.11
                        866     ;       ADDITIONAL READ/WRITE STORAGE TEST
                        867     ;DESCRIPTION
                        868     ;       WRITE/READ DATA PATTERNS TO ANY READ/WRITE STORAGE AFTER THE BASIC
                        869     ;       16K.  STORAGE ADDRESSABILITY IS CHECKED.
                        870     ;--------------------------------------------
                        871             ASSUME  DS:DATA
E3F8                    872     E19:
E3F8 B84000         R   873             MOV     AX,DATA
E3FB 8ED8               874             MOV     DS,AX
                        875
                        876     ;       DETERMINE RAM SIZE ON PLANAR BOARD
                        877
E3FD 8A261000       R   878             MOV     AH,BYTE PTR EQUIP_FLAG  ;GET SENSE SWS INFO
E401 80E40C             879             AND     AH,0CH                  ;ISOLATE RAM SIZE SWS
E404 B004               880             MOV     AL,4
E406 F6E4               881             MUL     AH
E408 0410               882             ADD     AL,16                   ;ADD BASIC 16K
E40A 8BD0               883             MOV     DX,AX                   ;SAVE PLANAR RAM SIZE IN DX
E40C 8BD8               884             MOV     BX,AX                   ;  AND IN BX
                        885
                        886     ;       DETERMINE IO CHANNEL RAM SIZE
                        887
E40E E462               888             IN      AL,PORT_C               ;READ IO CH RAM SIZE SWS
E410 240F               889             AND     AL,0FH                  ;ISOLATE FROM OTHER BITS
E412 B420               890             MOV     AH,32
E414 F6E4               891             MUL     AH
E416 A31500         R   892             MOV     IO_RAM_SIZE,AX          ;SAVE IO CHANNEL RAM SIZE
E419 83FB40             893             CMP     BX,40H                  ;PLANAR RAM SIZE = 64K?
E41C 7402               894             JE      E20                     ;YES - ADD IO CHN RAM SIZE
E41E 2BC0               895             SUB     AX,AX                   ;NO - DON T ADD ANY IO RAM
E420                    896     E20:                                    ; ADD_IO_SIZE:
E420 03C3               897             ADD     AX,BX                   ;SUM TOTAL RAM SIZE
E422 A31300         R   898             MOV     MEMORY_SIZE,AX          ;SETUP MEMORY SIZE PARM
E425 813E72003412   R   899             CMP     RESET_FLAG,1234H        ;POD INITIATED BY KBD RESET?
```

```
LOC OBJ                 LINE    SOURCE

E42B 744D                900            JE      E22                             ;YES - SKIP MEMORY TEST
                         901
                         902    ;       TEST ANY OTHER READ/WRITE STORAGE AVAILABLE
                         903
E42D BB0004              904            MOV     BX,400H
E430 B91000              905            MOV     CX,16
E433                     906    E21:
E433 3BD1                907            CMP     DX,CX                           ;ANY MORE STG TO BE TESTED?
E435 7646                908            JBE     E23                             ;NO - GO TO NEXT TEST
E437 8EDB                909            MOV     DS,BX                           ;SETUP STG ADDR IN DS AND ES
E439 8EC3                910            MOV     ES,BX
E43B 83C110              911            ADD     CX,16                           ;INCREMENT STG BYTE COUNTER
E43E 81C30004            912            ADD     BX,400H                         ;SET POINTER TO NEXT 16K BLK
E442 51                  913            PUSH    CX                              ;SAVE REGS
E443 53                  914            PUSH    BX
E444 52                  915            PUSH    DX
E445 E8D2FB              916            CALL    STGTST                          ;GO TEST A 16K BLK OF STG
E448 5A                  917            POP     DX
E449 5B                  918            POP     BX                              ;RESTORE REGS
E44A 59                  919            POP     CX
E44B 74E6                920            JE      E21                             ;CHECK IF MORE STG TO TEST
                         921
                         922    ;       PRINT FAILING ADDRESS AND XOR'ED PATTERN IF DATA COMPARE ERROR
                         923
E44D 8CDA                924            MOV     DX,DS                           ;CONVERT FAILING HIGH-ORDER
E44F 8AE8                925            MOV     CH,AL                           ;SAVE FAILING BIT PATTERN
E451 8AC6                926            MOV     AL,DH                           ;GET FAILING ADDR (HIGH BYTE)
E453 B104                927            MOV     CL,4
E455 D2E8                928            SHR     AL,CL                           ;RIGHT-JUSTIFY HIGH BYTE
E457 E83E00              929            CALL    XLAT_PRINT_CODE                 ;CONVERT AND PRINT CODE
E45A 8AC6                930            MOV     AL,DH
E45C 240F                931            AND     AL,0FH
E45E E83700              932            CALL    XLAT_PRINT_CODE                 ;CONVERT AND PRINT CODE
E461 8AC5                933            MOV     AL,CH                           ;GET FAILING BIT PATTERN
E463 B104                934            MOV     CL,4                            ; AND ISOLATE LEFTMOST NIBBLE
E465 D2E8                935            SHR     AL,CL
E467 E82E00              936            CALL    XLAT_PRINT_CODE                 ;CONVERT AND PRINT CODE
E46A 8AC5                937            MOV     AL,CH                           ;GET FAILING BIT PATTERN AND
E46C 240F                938            AND     AL,0FH                          ;  ISOLATE RIGHTMOST NIBBLE
E46E E82700              939            CALL    XLAT_PRINT_CODE                 ;CONVERT AND PRINT CODE
E471 BEE8E2          R   940            MOV     SI,OFFSET E1                    ;SETUP ADDRESS OF ERROR MSG
E474 B90400              941            MOV     CX,E1L                          ;GET MSG BYTE COUNT
E477 E85002              942            CALL    P_MSG                           ;PRINT ERROR MSG
E47A                     943    E22:                                            ; GO_TST12:
E47A E94A00              944            JMP     TST12                           ;GO TO NEXT TEST
E47D                     945    E23:                                            ; STG_TEST_DONE:
E47D B84000          R   946            MOV     AX,DATA                         ;POINT DS TO DATA SEGMENT
E480 8ED8                947            MOV     DS,AX                           ; CHG MADE 3/27/81
E482 8B161500        R   948            MOV     DX,IO_RAM_SIZE                  ;GET IO CHANNEL RAM SIZE
E486 0BD2                949            OR      DX,DX                           ;SET FLAG RESULT
E488 74F0                950            JZ      E22                             ;NO IO RAM, GO TO NEXT TEST
E48A B90000              951            MOV     CX,0
E48D 81FB0010            952            CMP     BX,1000H                        ;HAS IO RAM BEEN TESTED
E491 77E7                953            JA      E22                             ;YES - GO TO NEXT TEST
E493 BB0010              954            MOV     BX,1000H                        ;SETUP BEG LOC FOR IO RAM
E496 EB9B                955            JMP     SHORT E21                       ;GO TEST IO CHANNEL RAM
                         956    ;-----------------------------------------
                         957    ;       CONVERT AND PRINT ASCII CODE
                         958    ;
                         959    ;       AL MUST CONTAIN NUMBER TO BE CONVERTED.
                         960    ;       AX AND BX DESTROYED.
                         961    ;-----------------------------------------
E498                     962    XLAT_PRINT_CODE  PROC    NEAR
E498 1E                  963            PUSH    DS                              ;SAVE DS VALUE
E499 0E                  964            PUSH    CS                              ;POINT DS TO CODE SEG
E49A 1F                  965            POP     DS
E49B BBB7E4              966            MOV     BX,0E4B7H                       ; OFFSET ASCII_TBL-XLAT TABLE
E49E D7                  967            XLATB
E49F B40E                968            MOV     AH,14
E4A1 B700                969            MOV     BH,0
E4A3 CD10                970            INT     10H                             ;CALL VIDEO_IO
E4A5 1F                  971            POP     DS                              ;RESTORE ORIG VALUE IN DS
E4A6 C3                  972            RET
                         973    XLAT_PRINT_CODE  ENDP
```

```
LOC OBJ                  LINE    SOURCE

                         974     ;--------------------------------------
                         975     ; INITIAL RELIABILITY TEST -- PHASE 4
                         976     ;--------------------------------------
                         977             ASSUME  CS:CODE,DS:DATA
E4A7 20333031            978     F1      DB      ' 301'
  0004                   979     F1L     EQU     $-F1            ; KEYBOARD MESSAGE
E4AB 313331              980     F2      DB      '131'
  0003                   981     F2L     EQU     $-F2            ; CASSETTE MESSAGE
E4AE 363031              982     F3      DB      '601'
  0003                   983     F3L     EQU     $-F3            ; DISKETTE MESSAGE
                         984
E4B1                     985     F4      LABEL   WORD            ; PRINTER SOURCE TABLE
E4B1 BC03                986             DW      3BCH
E4B3 7803                987             DW      378H
E4B5 7802                988             DW      278H
E4B7                     989     F4E     LABEL   WORD
E4B7 30313233343536      990     ASCII_TBL       DB      '0123456789ABCDEF'
     37383941424344
     4546
                         991     ;--------------------------------------------
                         992     ;TEST.12
                         993     ;       KEYBOARD TEST
                         994     ;DESCRIPTION
                         995     ;       PESET THE KEYBOARD AND CHECK THAT SCAN CODE  AA' IS RETURNED
                         996     ;       TO THE CPU.  CHECK FOR STUCK KEYS.
                         997     ;--------------------------------------------
E4C7                     998     TST12:
E4C7 B84000          R   999             MOV     AX,DATA                 ;POINT DS TO DATA SEG
E4CA 8ED8               1000             MOV     DS,AX
E4CC 803E120001      R  1001             CMP     MFG_TST,1               ;MANUFACTURING TEST MODE?
E4D1 7439               1002             JE      F7                      ;YES - SKIP KEYBOARD TEST
E4D3 E8B201             1003             CALL    KBD_RESET               ;ISSUE SOFTWARE RESET TO KEYBRD
E4D6 E32B               1004             JCXZ    F6                      ;PRINT ERR MSG IF NO INTERRUPT
E4D8 B04D               1005             MOV     AL,4DH                  ;ENABLE KEYBOARD
E4DA E661               1006             OUT     PORT_B,AL
E4DC 80FBAA             1007             CMP     BL,0AAH                 ;SCAN CODE AS EXPECTED?
E4DF 7522               1008             JNE     F6                      ;NO - DISPLAY ERROR MSG
                        1009
                        1010     ;       CHECK FOR STUCK KEYS
                        1011
E4E1 B0CC               1012             MOV     AL,0CCH                 ;CLR KBD, SET CLK LINE HIGH
E4E3 E661               1013             OUT     PORT_B,AL
E4E5 B04C               1014             MOV     AL,4CH                  ;ENABLE KBD,CLK IN NEXT BYTE
E4E7 E661               1015             OUT     PORT_B,AL
E4E9 2BC9               1016             SUB     CX,CX
E4EB                    1017     F5:                                     ; KBD_WAIT:
E4EB E2FE               1018             LOOP    F5                      ;DELAY FOR A WHILE
E4ED E460               1019             IN      AL,KBD_IN               ;CHECK FOR STUCK KEYS
E4EF 3C00               1020             CMP     AL,0                    ;SCAN CODE = 0?
E4F1 7419               1021             JE      F7                      ;YES - CONTINUE TESTING
E4F3 8AE8               1022             MOV     CH,AL                   ;SAVE SCAN CODE
E4F5 B104               1023             MOV     CL,4
E4F7 D2E8               1024             SHR     AL,CL                   ;RIGHT-JUSTIFY HIGH BYTE
E4F9 E89CFF             1025             CALL    XLAT_PRINT_CODE         ;CONVERT AND PRINT
E4FC 8AC5               1026             MOV     AL,CH                   ;RECOVER SCAN CODE
E4FE 240F               1027             AND     AL,0FH                  ;ISOLATE LOW ORDER BYTE
E500 E895FF             1028             CALL    XLAT_PRINT_CODE         ;CONVERT AND PRINT
E503 BEA7E4          R  1029     F6:     MOV     SI,OFFSET F1            ;GET MSG ADDR
E506 B90400             1030             MOV     CX,F1L                  ;GET MSG BYTE COUNT
E509 E8BE01             1031             CALL    P_MSG                   ;PRINT MSG ON SCREEN
                        1032
                        1033     ;       SETUP INTERRUPT VECTOR TABLE
                        1034
E50C                    1035     F7:                                     ; SETUP_INT_TABLE:
E50C 2BC0               1036             SUB     AX,AX
E50E 8EC0               1037             MOV     ES,AX
E510 B90300             1038             MOV     CX,24*2                 ;GET VECTOR CNT
E513 0E                 1039             PUSH    CS                      ;SETUP DS SEG REG
E514 1F                 1040             POP     DS
E515 BEF3FE             1041             MOV     SI,0FEF3H               ; OFFSET VECTOR_TABLE
E518 BF2000             1042             MOV     DI,OFFSET INT_PTR
E51B FC                 1043             CLD
E51C F3                 1044             REP     MOVSW
E51D A5
```

```
LOC OBJ                 LINE    SOURCE


                        1045    ;--------------------------------------------
                        1046    ;TEST.13
                        1047    ;       CASSETTE DATA WRAP TEST
                        1048    ;DESCRIPTION
                        1049    ;       TURN CASSETTE MOTOR OFF. WRITE A BIT OUT TO THE CASSETTE DATA BUS.
                        1050    ;       VERIFY THAT CASSETTE DATA READ IS WITHIN A VALID RANGE.
                        1051    ;--------------------------------------------
                        1052
                        1053    ;       TURN THE CASSETTE MOTOR OFF
                        1054
E51E B84000         R   1055            MOV     AX,DATA                 ;POINT DS REG TO DATA SEG
E521 8ED8               1056            MOV     DS,AX
E523 B04D               1057            MOV     AL,04DH                 ;SET TIMER 2 SPK OUT, AND CASST
E525 E661               1058            OUT     PORT_B,AL               ;OUT BITS ON, CASSETTE MOT OFF
                        1059
                        1060    ;       WRITE A BIT
                        1061
E527 B0FF               1062            MOV     AL,0FFH                 ;DISABLE TIMER INTERRUPTS
E529 E621               1063            OUT     INTA01,AL
E52B B0B6               1064            MOV     AL,0B6H                 ;SEL TIM 2, LSB, MSB, MD 3
E52D E643               1065            OUT     TIMER+3,AL              ;WRITE 8253 CMD/MODE REG
E52F B8D304             1066            MOV     AX,1235                 ;SET TIMER 2 CNT FOR 1000 USEC
E532 E642               1067            OUT     TIMER+2,AL              ;WRITE TIMER 2 COUNTER REG
E534 8AC4               1068            MOV     AL,AH                   ;WRITE MSB
E536 E642               1069            OUT     TIMER+2,AL
                        1070
                        1071    ;       READ CASSETTE INPUT
                        1072
E538 E462               1073            IN      AL,PORT_C               ;READ VALUE OF CASS IN BIT
E53A 2410               1074            AND     AL,10H                  ;ISOLATE FROM OTHER BITS
E53C A26B00         R   1075            MOV     LAST_VAL,AL
E53F E83E14             1076            CALL    READ_HALF_BIT
E542 E83B14             1077            CALL    READ_HALF_BIT
E545 E30C               1078            JCXZ    F8                      ; CAS_ERR
E547 81FB4005           1079            CMP     BX,MAX_PERIOD
E54B 7306               1080            JNC     F8                      ; CAS_ERR
E54D 81FB1004           1081            CMP     BX,MIN_PERIOD
E551 7309               1082            JNC     F9                      ;GO TO NEXT TEST IF OK
E553                    1083    F8:                                     ; CAS_ERR:
E553 BEABE4         R   1084            MOV     SI,OFFSET F2            ;CASSETTE WRAP FAILED
E556 B90300             1085            MOV     CX,F2L
E559 E86E01             1086            CALL    P_MSG                   ;GO PRINT ERROR MSG
                        1087    ;--------------------------------------------
                        1088    ;TEST.14
                        1089    ;       DISKETTE ATTACHMENT TEST
                        1090    ;DESCRIPTION
                        1091    ;       CHECK IF IPL DISKETTE DRIVE IS ATTACHED TO SYSTEM.  IF ATTACHED,
                        1092    ;       VERIFY STATUS OF NEC FDC AFTER A RESET.  ISSUE A RECAL AND SEEK
                        1093    ;       CMD TO FDC AND CHECK STATUS. COMPLETE SYSTEM INITIALIZATION THEN
                        1094    ;       PASS CONTROL TO THE BOOT LOADER PROGRAM.
                        1095    ;--------------------------------------------
E55C                    1096    F9:
E55C B0FC               1097            MOV     AL,0FCH                 ;ENABLE TIMER AND KBD INTS
E55E E621               1098            OUT     INTA01,AL
E560 A01000         R   1099            MOV     AL,BYTE PTR EQUIP_FLAG  ;GET SENSE SWS INFO
E563 A801               1100            TEST    AL,01H                  ;IPL DISKETTE DRIVE ATTCH?
E565 7503               1101            JNZ     F10                     ;YES - TEST DISKETTE CONTR
E567 E9B900             1102            JMP     F22                     ;NO - SKIP THIS TEST
E56A                    1103    F10:                                    ; DISK_TEST:
E56A B0BC               1104            MOV     AL,0BCH                 ;ENABLE DISKETTE, KEYBOARD,
E56C E621               1105            OUT     INTA01,AL               ; AND TIMER INTERRUPTS
E56E B400               1106            MOV     AH,0                    ;RESET NEC FDC
E570 CD13               1107            INT     13H                     ;VERIFY STATUS AFTER RESET
E572 F6C4FF             1108            TEST    AH,0FFH                 ;STATUS OK?
E575 7520               1109            JNZ     F13                     ;NO - FDC FAILED
                        1110
                        1111    ;       TURN DRIVE 0 MOTOR ON
                        1112
E577 BAF203             1113            MOV     DX,03F2H                ;GET ADDR OF FDC CARD
E57A B01C               1114            MOV     AL,1CH                  ;TURN MOTOR ON, EN DMA/INT
E57C EE                 1115            OUT     DX,AL                   ;WRITE FDC CONTROL REG
E57D 2BC9               1116            SUB     CX,CX
E57F                    1117    F11:                                    ; MOTOR_WAIT:
E57F E2FE               1118            LOOP    F11                     ;WAIT FOR 1 SECOND
E581                    1119    F12:                                    ; MOTOR_WAIT1:
E581 E2FE               1120            LOOP    F12
E583 33D2               1121            XOR     DX,DX                   ;SELECT DRIVE 0
```

```
LOC OBJ                  LINE    SOURCE

E585 B501                1122            MOV     CH,1                    ;SELECT TRACK 1
E587 88163E00      R     1123            MOV     SEEK_STATUS,DL
E58B E8F308              1124            CALL    SEEK                    ;RECALIBRATE DISKETTE
E58E 7207                1125            JC      F13                     ;GO TO ERR SUBROUTINE IF ERR
E590 B522                1126            MOV     CH,34                   ;SELECT TRACK 34
E592 E8EC08              1127            CALL    SEEK                    ;SEEK TO TRACK 34
E595 7309                1128            JNC     F14                     ;OK, TURN MOTOR OFF
E597                     1129    F13:                                    ; DSK_ERR:
E597 BEAEE4        R     1130            MOV     SI,OFFSET F3            ;GET ADDR OF MSG
E59A B90300              1131            MOV     CX,F3L                  ;GET MSG BYTE COUNT
E59D E82A01              1132            CALL    P_MSG                   ;GO PRINT ERROR MSG
                         1133
                         1134    ;       TURN DRIVE 0 MOTOR OFF
                         1135
E5A0                     1136    F14:                                    ; DRO_OFF:
E5A0 B00C                1137            MOV     AL,0CH                  ;TURN DRIVE 0 MOTOR OFF
E5A2 BAF203              1138            MOV     DX,03F2H                ; FDC CTL ADDRESS
E5A5 EE                  1139            OUT     DX,AL
                         1140
                         1141    ;       SETUP PRINTER AND RS232 BASE ADDRESSES IF DEVICE ATTACHED
                         1142
E5A6                     1143    F15:                                    ; JMP_BOOT:
E5A6 C7061A001E00  R     1144            MOV     BUFFER_HEAD,OFFSET KB_BUFFER    ;SETUP KEYBOARD PARAMETERS
E5AC C7061C001E00  R     1145            MOV     BUFFER_TAIL,OFFSET KB_BUFFER
E5B2 BDB1E4        R     1146            MOV     BP,OFFSET F4            ; PRT_SRC_TBL
E5B5 BE0000              1147            MOV     SI,0
E5B8                     1148    F16:                                    ; PRT_BASE:
E5B8 2E8B5600            1149            MOV     DX,CS:[BP]              ;GET PRINTER BASE ADDR
E5BC B0AA                1150            MOV     AL,0AAH                 ;WRITE DATA TO PORT A
E5BE EE                  1151            OUT     DX,AL
E5BF 2AC0                1152            SUB     AL,AL
E5C1 EC                  1153            IN      AL,DX                   ;READ PORT A
E5C2 3CAA                1154            CMP     AL,0AAH                 ;DATA PATTERN SAME
E5C4 7506                1155            JNE     F17                     ;NO - CHECK NEXT PRT CD
E5C6 89940800      R     1156            MOV     PRINTER_BASE[SI],DX     ;YES - STORE PRT BASE ADDR
E5CA 46                  1157            INC     SI                      ;INCREMENT TO NEXT WORD
E5CB 46                  1158            INC     SI
E5CC                     1159    F17:                                    ; NO_STORE:
E5CC 45                  1160            INC     BP                      ;POINT TO NEXT BASE ADDR
E5CD 45                  1161            INC     BP
E5CE 81FDB7E4      R     1162            CMP     BP,OFFSET F4E           ;ALL POSSIBLE ADDRS CHECKED?
E5D2 75E4                1163            JNE     F16                     ;PRT_BASE
E5D4 BB0000              1164            MOV     BX,0                    ;POINTER TO RS232 TABLE
E5D7 BAFA03              1165            MOV     DX,3FAH                 ;CHECK IF RS232 CD 1 ATTCH?
E5DA EC                  1166            IN      AL,DX                   ;READ INTR ID REG
E5DB A8F8                1167            TEST    AL,0F8H
E5DD 7508                1168            JNZ     F18
E5DF C7870000F803  R     1169            MOV     RS232_BASE[BX],3F8H     ;SETUP RS232 CD #1 ADDR
E5E5 43                  1170            INC     BX
E5E6 43                  1171            INC     BX
E5E7 BAFA02              1172    F18:    MOV     DX,2FAH                 ;CHECK IF RS232 CD 2 ATTCH
E5EA EC                  1173            IN      AL,DX                   ;READ INTERRUPT ID REG
E5EB A8F8                1174            TEST    AL,0F8H
E5ED 7508                1175            JNZ     F19                     ;BASE_END
E5EF C7870000F802  R     1176            MOV     RS232_BASE[BX],2F8H     ;SETUP RS232 CD #2
E5F5 43                  1177            INC     BX
E5F6 43                  1178            INC     BX
                         1179
                         1180    ;------ SET UP EQUIP FLAG TO INDICATE NUMBER OF PRINTERS AND RS232 CARDS
                         1181
E5F7                     1182    F19:                                    ; BASE_END:
E5F7 8BC6                1183            MOV     AX,SI                   ; SI HAS 2* NUMBER OF RS232
E5F9 B103                1184            MOV     CL,3                    ; SHIFT COUNT
E5FB D2C8                1185            ROR     AL,CL                   ; ROTATE RIGHT 3 POSITIONS
E5FD 0AC3                1186            OR      AL,BL                   ; OR IN THE PRINTER COUNT
E5FF A21100        R     1187            MOV     BYTE PTR EQUIP_FLAG+1,AL        ; STORE AS SECOND BYTE
E602 BA0102              1188            MOV     DX,201H
E605 EC                  1189            IN      AL,DX
E606 A80F                1190            TEST    AL,0FH
E608 7505                1191            JNZ     F20                     ; NO_GAME_CARD
E60A 800E110010    R     1192            OR      BYTE PTR EQUIP_FLAG+1,16
E60F                     1193    F20:                                    ; NO_GAME_CARD:
                         1194
                         1195    ;       ENABLF NMI INTERRUPTS
                         1196
E60F B080                1197            MOV     AL,80H                  ;ENABLE NMI INTERRUPTS
E611 E6A0                1198            OUT     0A0H,AL
```

```
LOC OBJ                 LINE    SOURCE

E613 803E120001    R    1199            CMP     MFG_TST,1                       ;MFG MODE?
E618 7406               1200            JE      F21                             ; LOAD_BOOT_STRAP
E61A BA0100             1201            MOV     DX,1                            ;
E61D E81000             1202            CALL    ERR_BEEP                        ;BEEP 1 SHORT TONE
E620                    1203    F21:                                            ; LOAD_BOOT_STRAP:
E620 E9CF00             1204            JMP     BOOT_STRAP                      ;GO TO THE BOOT LOADER
E623                    1205    F22:                                            ; LOOP_POD:
E623 803E120001    R    1206            CMP     MFG_TST,1                       ;MANUFACTURING TEST MODE?
E628 7503               1207            JNE     F23                             ;NO - GO TO BOOT LOADER
E62A E92EFA             1208            JMP     START                           ;YES - LOOP POWER-ON-DIAGS
E62D                    1209    F23:                                            ; GO_TO_BOOT:
E62D E976FF             1210            JMP     F15                             ; JMP_BOOT
                        1211    ;-----------------------------------------
                        1212    ; INITIAL RELIABILITY TEST -- SUBROUTINES
                        1213    ;-----------------------------------------
                        1214            ASSUME  CS:CODE,DS:DATA
                        1215    ;-----------------------------------------
                        1216    ;       SUBROUTINES FOR POWER ON DIAGNOSTICS
                        1217    ;-----------------------------------------
                        1218    ;       THIS PROCEDURE WILL ISSUE ONE LONG TONE (3 SECS) AND ONE OR
                        1219    ;       MORE SHORT TONES (1 SEC) TO INDICATE A FAILURE ON THE PLANAR
                        1220    ;       BOARD, A BAD RAM MODULE, OR A PROBLEM WITH THE CRT.
                        1221    ;ENTRY PARAMETERS:
                        1222    ;       DH = NUMBER OF LONG TONES TO BEEP
                        1223    ;       DL = NUMBER OF SHORT TONES TO BEEP.
                        1224    ;-----------------------------------------
E630                    1225    ERR_BEEP  PROC  NEAR
E630 9C                 1226            PUSHF                                   ;SAVE FLAGS
E631 FA                 1227            CLI                                     ;DISABLE SYSTEM INTERRUPTS
E632 1E                 1228            PUSH    DS                              ;SAVE DS REG CONTENTS
E633 B84000        R    1229            MOV     AX,DATA                         ;POINT DS TO DATA SEG
E636 8ED8               1230            MOV     DS,AX
E638 0AF6               1231            OR      DH,DH           ; ANY LONG ONES TO BEEP
E63A 7418               1232            JZ      G3              ; NO, DO THE SHORT ONES
E63C                    1233    G1:                             ; LONG_BEEP:
E63C B306               1234            MOV     BL,6            ; COUNTER FOR BEEPS
E63E E82500             1235            CALL    BEEP            ; DO THE BEEP
E641 E2FE               1236    G2:     LOOP    G2              ; DELAY BETWEEN BEEPS
E643 FECE               1237            DEC     DH              ; ANY MORE TO DO
E645 75F5               1238            JNZ     G1              ; DO IT
E647 803E120001    R    1239            CMP     MFG_TST,1       ; MFG TEST MODE?
E64C 7506               1240            JNE     G3              ; YES - CONTINUE BEEPING SPEAKER
E64E B0CD               1241            MOV     AL,0CDH         ; STOP BLINKING LED
E650 E661               1242            OUT     PORT_B,AL
E652 EBE8               1243            JMP     SHORT G1
E654                    1244    G3:                             ; SHORT_BEEP:
E654 B301               1245            MOV     BL,1            ; COUNTER FOR A SHORT BEEP
E656 E80D00             1246            CALL    BEEP            ; DO THE SOUND
E659 E2FE               1247    G4:     LOOP    G4              ; DELAY BETWEEN BEEPS
E65B FECA               1248            DEC     DL              ; DONE WITH SHORTS
E65D 75F5               1249            JNZ     G3              ; DO SOME MORE
E65F E2FE               1250    G5:     LOOP    G5              ; LONG DELAY BEFORE RETURN
E661 E2FE               1251    G6:     LOOP    G6
E663 1F                 1252            POP     DS              ;RESTORE ORIG CONTENTS OF DS
E664 9D                 1253            POPF                    ;RESTORE FLAGS TO ORIG SETTINGS
E665 C3                 1254            RET                     ; RETURN TO CALLER
                        1255    ERR_BEEP        ENDP
                        1256
                        1257    ;       ROUTINE TO SOUND BEEPER
                        1258
E666                    1259    BEEP    PROC    NEAR
E666 B0B6               1260            MOV     AL,10110110B                    ;SEL TIM 2,LSB,MSB,BINARY
E668 E643               1261            OUT     TIMER+3,AL                      ;WRITE THE TIMER MODE REG
E66A B83305             1262            MOV     AX,533H                         ;DIVISOR FOR 1000 HZ
E66D E642               1263            OUT     TIMER+2,AL                      ;WRITE TIMER 2 CNT - LSB
E66F 8AC4               1264            MOV     AL,AH
E671 E642               1265            OUT     TIMER+2,AL                      ;WRITE TIMER 2 CNT - MSB
E673 E461               1266            IN      AL,PORT_B                       ;GET CURRENT SETTING OF PORT
E675 8AE0               1267            MOV     AH,AL                           ; SAVE THAT SETTINGH
E677 0C03               1268            OR      AL,03                           ;TURN SPEAKER ON
E679 E661               1269            OUT     PORT_B,AL
E67B 2BC9               1270            SUB     CX,CX                           ;SET CNT TO WAIT 500 MS
E67D E2FE               1271    G7:     LOOP    G7                              ;DELAY BEFORE TURNING OFF
E67F FECB               1272            DEC     BL                              ;DELAY CNT EXPIRED?
E681 75FA               1273            JNZ     G7                              ;NO - CONTINUE BEEPING SPK
E683 8AC4               1274            MOV     AL,AH                           ; RECOVER VALUE OF PORT
```

```
LOC OBJ                 LINE    SOURCE

E685 E661               1275            OUT     PORT_B,AL
E687 C3                 1276            RET                                     ;RETURN TO CALLER
                        1277    BEEP    ENDP
                        1278    ;--------------------------------------------
                        1279    ;       THIS PROCEDURE WILL SEND A SOFTWARE RESET TO THE KEYBOARD.
                        1280    ;       SCAN CODE  AA' SHOULD BE RETURNED TO THE CPU.
                        1281    ;--------------------------------------------
E688                    1282    KBD_RESET       PROC    NEAR
E688 B00C               1283            MOV     AL,0CH                          ;SET KBD CLK LINE LOW
E68A E661               1284            OUT     PORT_B,AL                       ;WRITE 8255 PORT B
E68C B95629             1285            MOV     CX,10582                        ;HOLD KBD CLK LOW FOR 20 MS
E68F E2FE               1286    G8:     LOOP    G8                              ;LOOP FOR 20 MS
E691 B0CC               1287            MOV     AL,0CCH                         ;SET CLK, ENABLE LINES HIGH
E693 E661               1288            OUT     PORT_B,AL
E695                    1289    SP_TEST:                ; ENTRY FOR MANUFACTURING TEST 2
E695 B04C               1290            MOV     AL,4CH                          ;SET KBD CLK HIBH, ENABLE LOW
E697 E661               1291            OUT     PORT_B,AL
E699 B0FD               1292            MOV     AL,0FDH                         ;ENABLE KEYBOARD INTERRUPTS
E69B E621               1293            OUT     INTA01,AL                       ;WRITE 8259 IMR
E69D FB                 1294            STI                                     ;ENABLE SYSTEM INTERRUPTS
E69E B400               1295            MOV     AH,0                            ;RESET INTERRUPT INDICATOR
E6A0 2BC9               1296            SUB     CX,CX                           ;SETUP INTERRUPT TIMEOUT CNT
E6A2 F6C4FF             1297    G9:     TEST    AH,0FFH                         ;DID A KEYBOARD INTR OCCUR?
E6A5 7502               1298            JNZ     G10                             ;YES - READ SCAN CODE RETURNED
E6A7 E2F9               1299            LOOP    G9                              ;NO - LOOP TILL TIMEOUT
E6A9 E460               1300    G10:    IN      AL,PORT_A                       ;READ KEYBOARD SCAN CODE
E6AB 8AD8               1301            MOV     BL,AL                           ;SAVE SCAN CODE JUST READ
E6AD B0CC               1302            MOV     AL,0CCH                         ;CLEAR KEYBOARD
E6AF E661               1303            OUT     PORT_B,AL
E6B1 C3                 1304            RET                                     ;RETURN TO CALLER
                        1305    KBD_RESET       ENDP
                        1306    ;--------------------------------------------
                        1307    ;       BLINK LED PROCEDURE FOR MFG BURN-IN AND RUN-IN TESTS
                        1308    ;       (LED WILL BLINK APPROXIMATELY .25 SECOND)
                        1309    ;--------------------------------------------
E6B2                    1310    BLINK_INT       PROC    NEAR
E6B2 FB                 1311            STI
E6B3 51                 1312            PUSH    CX                              ;SAVE CX REG CONTENTS
E6B4 50                 1313            PUSH    AX                              ;SAVE AX REG CONTENTS
E6B5 E461               1314            IN      AL,PORT_B                       ;READ CURRENT VAL OF PORT B
E6B7 24BF               1315            AND     AL,0BFH
E6B9 E661               1316            OUT     PORT_B,AL                       ;BLINK LED
E6BB 2BC9               1317            SUB     CX,CX
E6BD E2FE               1318    G11:    LOOP    G11
E6BF 0C40               1319            OR      AL,40H                          ;STOP BLINKING LED
E6C1 E661               1320            OUT     PORT_B,AL
E6C3 B020               1321            MOV     AL,EOI
E6C5 E620               1322            OUT     INTA00,AL
E6C7 58                 1323            POP     AX                              ;RESTORE AX REG
E6C8 59                 1324            POP     CX                              ;RESTORE CX REG
E6C9 CF                 1325            IRET
                        1326    BLINK_INT       ENDP
                        1327    ;--------------------------------------------
                        1328    ;       THIS SUBROUTINE WILL PRINT A MESSAGE ON THE DISPLAY
                        1329    ;
                        1330    ;ENTRY REQUIREMENTS:
                        1331    ;       SI = OFFSET(ADDRESS) OF MESSAGE BUFFER
                        1332    ;       CX = MESSAGE BYTE COUNT
                        1333    ;       MAXIMUM MESSAGE LENGTH IS 36 CHARACTERS
                        1334    ;--------------------------------------------
E6CA                    1335    P_MSG   PROC    NEAR
E6CA B84000         R   1336            MOV     AX,DATA                         ;POINT DS TO DATA SEG
E6CD 8ED8               1337            MOV     DS,AX
E6CF 803E120001     R   1338            CMP     MFG_TST,1                       ;MFG TEST MODE?
E6D4 7505               1339            JNE     G12                             ;NO - DISPLAY ERROR MSG
E6D6 B601               1340            MOV     DH,1                            ;YES - SETUP TO BEEP SPEAKER
E6D8 E955FF             1341            JMP     ERR_BEEP                        ;YES - BEEP SPEAKER
E6DB                    1342    G12:                                            ; WRITE_MSG:
E6DB 2E8A04             1343            MOV     AL,CS:[SI]                      ;PUT CHAR IN AL
E6DE 46                 1344            INC     SI                              ;POINT TO NEXT CHAR
E6DF B700               1345            MOV     BH,0                            ;SET PAGE # TO ZERO
E6E1 B40E               1346            MOV     AH,14                           ;WRITE CHAR (TTY-INTERFACE)
E6E3 CD10               1347            INT     10H                             ;CALL VIDEO_IO
E6E5 E2F4               1348            LOOP    G12                             ;CONTINUE TILL MSG WRITTEN
E6E7 B80D0E             1349            MOV     AX,0E0DH                        ;POSITION CURSOR TO NEXT LINE
E6EA CD10               1350            INT     10H                             ;SEND CARRIAGE RETURN AND
```

```
LOC OBJ                 LINE    SOURCE


E6EC B80A0E             1351            MOV     AX,0E0AH                        ;LINE FEED CHARS
E6EF CD10               1352            INT     10H
E6F1 C3                 1353            RET
                        1354    P_MSG   ENDP
                        1355    ;--- INT 19 -----------------------------
                        1356    ;BOOT STRAP LOADER
                        1357    ;       IF A 5 1/4" DISKETTE DRIVE IS AVAILABLE
                        1358    ;       ON THE SYSTEM, TRACK 0, SECTOR 1 IS READ INTO THE
                        1359    ;       BOOT LOCATION (SEGMENT 0, OFFSET 7C00)
                        1360    ;       AND CONTROL IS TRANSFERRED THERE.
                        1361    ;
                        1362    ;       IF THERE IS NO DISKETTE DRIVE, OR IF THERE IS
                        1363    ;       IS A HARDWARE ERROR CONTROL IS TRANSFERRED
                        1364    ;       TO THE CASSETTE BASIC ENTRY POINT.
                        1365    ;
                        1366    ; IPL ASSUMPTIONS
                        1367    ;       8255 PORT 60H BIT 0
                        1368    ;       = 1 IF IPL FROM DISKETTE
                        1369    ;----------------------------------------
                        1370            ASSUME  CS:CODE,DS:DATA
E6F2                    1371    BOOT_STRAP      PROC    NEAR
                        1372
E6F2 FB                 1373            STI                     ; ENABLE INTERRUPTS
E6F3 B84000         R   1374            MOV     AX,DATA         ; ESTABLISH ADDRESSING
E6F6 8ED8               1375            MOV     DS,AX
E6F8 A11000         R   1376            MOV     AX,EQUIP_FLAG   ; GET THE EQUIPMENT SWITCHES
E6FB A801               1377            TEST    AL,1            ; ISOLATE IPL SENSE SWITCH
E6FD 7423               1378            JZ      H3              ; GO TO CASSETTE BASIC ENTRY POINT
                        1379
                        1380    ;------ MUST LOAD SYSTEM FROM DISKETTE -- CX HAS RETRY COUNT
                        1381
E6FF B90400             1382            MOV     CX,4            ; SET RETRY COUNT
E702                    1383    H1:                             ; IPL_SYSTEM
E702 51                 1384            PUSH    CX              ; SAVE RETRY COUNT
E703 B400               1385            MOV     AH,0            ; RESET THE DISKETTE SYSTEM
E705 CD13               1386            INT     13H             ; DISKETTE_IO
E707 7214               1387            JC      H2              ; IF ERROR, TRY AGAIN
E709 B402               1388            MOV     AH,2            ; READ IN THE SINGLE SECTOR
E70B BB0000             1389            MOV     BX,0            ; TO THE BOOT LOCATION
E70E 8EC3               1390            MOV     ES,BX
E710 BB007C             1391            MOV     BX,OFFSET BOOT_LOCN
E713 BA0000             1392            MOV     DX,0            ; DRIVE 0, HEAD 0
E716 B90100             1393            MOV     CX,1            ; SECTOR 1, TRACK 0
E719 B001               1394            MOV     AL,1            ; READ ONE SECTOR
E71B CD13               1395            INT     13H             ; DISKETTE_IO
E71D 59                 1396    H2:     POP     CX              ; RECOVER RETRY COUNT
E71E 7304               1397            JNC     H4              ; CF SET BY UNSUCCESSFUL READ
E720 E2E0               1398            LOOP    H1              ; DO IT FOR RETRY TIMES
                        1399
                        1400    ;------ UNABLE TO IPL FROM THE DISKETTE
                        1401
E722                    1402    H3:                             ; CASSETTE_JUMP:
E722 CD18               1403            INT     18H             ; USE INTERRUPT VECTOR TO GET TO BASIC
                        1404
                        1405    ;------ IPL WAS SUCCESSFUL
                        1406
E724                    1407    H4:
E724 EA007C0000         1408            JMP     BOOT_LOCN
                        1409    BOOT_STRAP      ENDP
                        1410    ;-----INT 14----------------------------------
                        1411    ;RS232_IO
                        1412    ;       THIS ROUTINE PROVIDES BYTE STREAM I/O TO THE COMMUNICATIONS
                        1413    ;       PORT ACCORDING TO THE PARAMETERS:
                        1414    ;       (AH)=0  INITIALIZE THE COMMUNICATIONS PORT
                        1415    ;               (AL) HAS PARMS FOR INITIALIZATION
                        1416    ;
                        1417    ;               7       6       5       4       3       2       1       0
                        1418    ;               ----- BAUD RATE --      -PARITY--     STOPBIT   --WORD LENGTH--
                        1419
                        1420    ;               000 - 110               X0 - NONE       0 - 1    10 - 7 BITS
                        1421    ;               001 - 150               01 - ODD        1 - 2    11 - 8 BITS
                        1422    ;               010 - 300               11 - EVEN
                        1423    ;               011 - 600
                        1424    ;               100 - 1200
                        1425    ;               101 - 2400
                        1426    ;               110 - 4800
                        1427    ;               111 - 9600
```

```
LOC OBJ                  LINE   SOURCE

                         1428   ;               ON RETURN, CONDITIONS SET AS IN CALL TO COMMO STATUS (AH=3)
                         1429   ;       (AH)=1  SEND THE CHARACTER IN (AL) OVER THE COMMO LINE
                         1430   ;               (AL) REGISTER IS PRESERVED
                         1431   ;               ON EXIT, BIT 7 OF AH IS SET IF THE ROUTINE WAS UNABLE TO
                         1432   ;                       TO TRANSMIT THE BYTE OF DATA OVER THE LINE.  THE
                         1433   ;                       REMAINDER OF AH IS SET AS IN A STATUS REQUEST,
                         1434   ;                       REFELECTING THE CURRENT STATUS OF THE LINE.
                         1435   ;       (AH)=2  RECEIVE A CHARACTER IN (AL) FROM COMMO LINE BEFORE
                         1436   ;                       RETURNING TO CALLER
                         1437   ;               ON EXIT, AH HAS THE CURRENT LINE STATUS, AS SET BY THE
                         1438   ;                       THE STATUS ROUTINE, EXCEPT THAT THE ONLY BITS
                         1439   ;                       LEFT ON ARE THE ERROR BITS (7,4,3,2,1)
                         1440   ;                       IN THIS CASE, THE TIME OUT BIT INDICATES DATA SET
                         1441   ;                       READY WAS NOT RECEIVED.
                         1442   ;                       THUS, AH IS NON ZERO ONLY WHEN AN ERROR OCCURRED.
                         1443   ;       (AH)=3  RETURN THE COMMO PORT STATUS IN (AX)
                         1444   ;               AH CONTAINS THE LINE CONTROL STATUS
                         1445   ;               BIT 7 = TIME OUT
                         1446   ;               BIT 6 = TRANS SHIFT REGISTER EMPTY
                         1447   ;               BIT 5 = TRAN HOLDING REGISTER EMPTY
                         1448   ;               BIT 4 = BREAK DETECT
                         1449   ;               BIT 3 = FRAMING ERROR
                         1450   ;               BIT 2 = PARITY ERROR
                         1451   ;               BIT 1 = OVERRUN ERROR
                         1452   ;               BIT 0 = DATA READY
                         1453   ;               AL CONTAINS THE MODEM STATUS
                         1454   ;               BIT 7 = RECEVED LINE SIGNAL DETECT
                         1455   ;               BIT 6 = RING INDICATOR
                         1456   ;               BIT 5 = DATA SET READY
                         1457   ;               BIT 4 = CLEAR TO SEND
                         1458   ;               BIT 3 = DELTA RECEIVE LINE SIGNAL DETECT
                         1459   ;               BIT 2 = TRAILING EDGE RING DETECTOR
                         1460   ;               BIT 1 = DELTA DATA SET READY
                         1461   ;               BIT 0 = DELTA CLEAR TO SEND
                         1462
                         1463   ;       (DX) = PARAMETER INDICATING WHICH RS232 CARD (0,1 ALLOWED)
                         1464   ; DATA AREA RS232_BASE CONTAINS THE BASE ADDRESS OF THE 8250 ON THE CARD
                         1465   ;       LOCATION 400H CONTAINS UP TO 4 RS232 ADDRESSES POSSIBLE
                         1466   ;OUTPUT
                         1467   ;               AX      MODIFIED ACCORDING TO PARMS OF CALL
                         1468   ;               ALL OTHERS UNCHANGED
                         1469   ;--------------------------------------------
                         1470           ASSUME  CS:CODE,DS:DATA
E729                     1471   A1      LABEL   WORD
E729 1704                1472                   DW      1047    ; 110 BAUD      ; TABLE OF INIT VALUE
E72B 0003                1473                   DW      768     ; 150
E72D 8001                1474                   DW      384     ; 300
E72F C000                1475                   DW      192     ; 600
E731 6000                1476                   DW      96      ; 1200
E733 3000                1477                   DW      48      ; 2400
E735 1800                1478                   DW      24      ; 4800
E737 0C00                1479                   DW      12      ; 9600
                         1480
E739                     1481   RS232_IO        PROC    FAR
                         1482
                         1483   ;------ VECTOR TO APPROPRIATE ROUTINE
                         1484
E739 FB                  1485           STI                     ; INTERRUPTS BACK ON
E73A 1E                  1486           PUSH    DS              ; SAVE SEGMENT
E73B 52                  1487           PUSH    DX
E73C 56                  1488           PUSH    SI
E73D 57                  1489           PUSH    DI
E73E 51                  1490           PUSH    CX
E73F 8BF2                1491           MOV     SI,DX           ; RS232 VALUE TO SI
E741 D1E6                1492           SHL     SI,1            ; WORD OFFSET
E743 BA4000        R     1493           MOV     DX,DATA
E746 8EDA                1494           MOV     DS,DX           ; SET UP OUR SEGMENT
E748 8B940000      R     1495           MOV     DX,RS232_BASE[SI]       ; GET BASE ADDRESS
E74C 0BD2                1496           OR      DX,DX           ; TEST FOR 0 BASE ADDRESS
E74E 7416                1497           JZ      A3              ; RETURN
E750 0AE4                1498           OR      AH,AH           ; TEST FOR (AH)=0
E752 7418                1499           JZ      A4              ; COMMUN INIT
E754 FECC                1500           DEC     AH              ; TEST FOR (AH)=1
E756 744E                1501           JZ      A5              ; SEND AL
E758 FECC                1502           DEC     AH              ; TEST FOR (AH)=2
E75A 7503                1503           JNZ     A2
E75C E98900              1504           JMP     A12             ; RECEIVE INTO AL
```

```
LOC OBJ                  LINE    SOURCE


E75F                     1505    A2:
E75F FECC                1506            DEC     AH              ; TEST FOR (AH)=3
E761 7503                1507            JNZ     A3
E763 E9B900              1508            JMP     A18             ; COMMUNICATION STATUS
E766                     1509    A3:                             ; RETURN FROM RS232
E766 59                  1510            POP     CX
E767 5F                  1511            POP     DI
E768 5E                  1512            POP     SI
E769 5A                  1513            POP     DX
E76A 1F                  1514            POP     DS
E76B CF                  1515            IRET                    ; RETURN TO CALLER, NO ACTION
                         1516
                         1517    ;------ INITIALIZE THE COMMUNICATIONS PORT
                         1518
E76C                     1519    A4:
E76C 8AE0                1520            MOV     AH,AL           ; SAVE INIT PARMS IN AH
E76E 83C203              1521            ADD     DX,3            ; POINT TO 8250 CONTROL REGISTER
E771 B080                1522            MOV     AL,80H
E773 EE                  1523            OUT     DX,AL           ; SET DLAB=1
                         1524
                         1525    ;------ DETERMINE BAUD RATE DIVISOR
                         1526
E774 8AD4                1527            MOV     DL,AH           ; GET PARMS TO DL
E776 D0C2                1528            ROL     DL,1
E778 D0C2                1529            ROL     DL,1            ; GET BAUD RATE TERM TO LOW BITS
E77A D0C2                1530            ROL     DL,1
E77C D0C2                1531            ROL     DL,1            ; *2 FOR WORD TABLE ACCESS
E77E 81E20E00            1532            AND     DX,0EH          ; ISOLATE THEM
E782 BF29E7        R     1533            MOV     DI,OFFSET A1    ; BASE OF TABLE
E785 03FA                1534            ADD     DI,DX           ; PUT INTO INDEX REGISTER
E787 8B940000      R     1535            MOV     DX,RS232_BASE[SI]       ; POINT TO HIGH ORDER OF DIVISOR
E78B 42                  1536            INC     DX
E78C 2E8A4501            1537            MOV     AL,CS:[DI]+1    ; GET HIGH ORDER OF DIVISOR
E790 EE                  1538            OUT     DX,AL           ; SET MS OF DIV TO 0
E791 4A                  1539            DEC     DX
E792 2E8A05              1540            MOV     AL,CS:[DI]      ; GET LOW ORDER OF DIVISOR
E795 EE                  1541            OUT     DX,AL           ; SET LOW OF DIVISOR
E796 83C203              1542            ADD     DX,3
E799 8AC4                1543            MOV     AL,AH           ; GET PARMS BACK
E79B 241F                1544            AND     AL,01FH         ; STRIP OFF THE BAUD BITS
E79D EE                  1545            OUT     DX,AL           ; LINE CONTROL TO 8 BITS
E79E 83EA02              1546            SUB     DX,2
E7A1 B000                1547            MOV     AL,0
E7A3 EE                  1548            OUT     DX,AL           ; INTERRUPT ENABLES ALL OFF
E7A4 EB79                1549            JMP     SHORT A18       ; COM_STATUS
                         1550
                         1551    ;------ SEND CHARACTER IN (AL) OVER COMMO LINE
                         1552
E7A6                     1553    A5:
E7A6 50                  1554            PUSH    AX              ; SAVE CHAR TO SEND
E7A7 83C204              1555            ADD     DX,4            ; MODEM CONTROL REGISTER
E7AA B003                1556            MOV     AL,3            ; DTR AND RTS
E7AC EE                  1557            OUT     DX,AL           ; DATA TERMINAL READY, REQUEST TO SEND
E7AD 33C9                1558            XOR     CX,CX           ; INITIALIZE TIME OUT COUNT
E7AF 83C202              1559            ADD     DX,2            ; MODEM STATUS REGISTER
E7B2                     1560    A6:                             ; WAIT_DATA_SET_READY
E7B2 EC                  1561            IN      AL,DX           ; GET MODEM STATUS
E7B3 A820                1562            TEST    AL,20H          ; DATA SET READY
E7B5 7508                1563            JNZ     A7              ; TEST_CLEAR_TO_SEND
E7B7 E2F9                1564            LOOP    A6              ; WAIT_DATA_SET_READY
E7B9 58                  1565            POP     AX
E7BA 80CC50              1566            OR      AH,80           ; INDICATE TIME OUT
E7BD EBA7                1567            JMP     A3              ; RETURN
E7BF                     1568    A7:                             ; TEST_CLEAR_TO_SEND
E7BF 2BC9                1569            SUB     CX,CX
E7C1                     1570    A8:                             ; WAIT_CLEAR_TO_SEND
E7C1 EC                  1571            IN      AL,DX           ; GET MODEM STATUS
E7C2 A810                1572            TEST    AL,10H          ; TEST CLEAR TO SEND
E7C4 7508                1573            JNZ     A9              ; CLEAR_TO_SEND
E7C6 E2F9                1574            LOOP    A8              ; WAIT_CLEAR_TO_SEND
E7C8 58                  1575            POP     AX              ; TIME OUT HAS OCCURRED
E7C9 80CC80              1576            OR      AH,80H
E7CC EB98                1577            JMP     A3              ; RETURN
E7CE                     1578    A9:                             ; CLEAR_TO_SEND
E7CE 4A                  1579            DEC     DX              ; LINE STATUS REGISTER
E7CF 2BC9                1580            SUB     CX,CX           ; INITIALIZE WAIT COUNT
E7D1                     1581    A10:                            ; WAIT_SEND
```

```
LOC OBJ                  LINE    SOURCE

E7D1 EC                  1582            IN      AL,DX           ; GET STATUS
E7D2 A820                1583            TEST    AL,20H          ; IS TRANSMITTER READY
E7D4 7508                1584            JNZ     A11             ; OUT_CHAR
E7D6 E2F9                1585            LOOP    A10             ; GO BACK FOR MORE, AND TEST FOR TIME OUT
E7D8 58                  1586            POP     AX              ; RECOVER ORIGINAL INPUT
E7D9 80CC80              1587            OR      AH,80H          ; SET THE TIME OUT BIT
E7DC EB88                1588            JMP     A3              ; RETURN
E7DE                     1589    A11:                            ; OUT_CHAR
E7DE 83EA05              1590            SUB     DX,5            ; DATA PORT
E7E1 59                  1591            POP     CX              ; RECOVER IN CX TEMPORARILY
E7E2 8AC1                1592            MOV     AL,CL           ; GET OUT CHAR TO AL FOR OUT, STATUS IN AH
E7E4 EE                  1593            OUT     DX,AL           ; OUTPUT CHARACTER
E7E5 E97EFF              1594            JMP     A3              ; RETURN
                         1595
                         1596    ;------ RECEIVE CHARACTER FROM COMMO LINE
                         1597
E7E8                     1598    A12:
E7E8 802671007F      R   1599            AND     BIOS_BREAK,07FH         ; TURN OFF BREAK BIT IN BYTE
E7ED 83C204              1600            ADD     DX,4            ; MODEM CONTROL REGISTER
E7F0 B001                1601            MOV     AL,1            ; DATA TERMINAL READY
E7F2 EE                  1602            OUT     DX,AL
E7F3 83C202              1603            ADD     DX,2            ; MODEM STATUS REGISTER
E7F6 2BC9                1604            SUB     CX,CX           ; ESTABLISH TIME OUT COUNT
E7F8                     1605    A13:                            ; WAIT_DSR
E7F8 EC                  1606            IN      AL,DX           ; MODEM STATUS
E7F9 A820                1607            TEST    AL,20H          ; DATA SET READY
E7FB 7507                1608            JNZ     A15             ; IS IT READY YET
E7FD E2F9                1609            LOOP    A13             ; WAIT UNTIL IT IS
E7FF                     1610    A14:                            ; TIME_OUT_ERR
E7FF B480                1611            MOV     AH,80H          ; SET TIME OUT ERROR
E801 E962FF              1612            JMP     A3              ; RETURN WITH ERROR
E804                     1613    A15:                            ; WAIT_DSR_END
E804 4A                  1614            DEC     DX              ; LINE STATUS REGISTER
E805                     1615    A16:                            ; WAIT_RECV
E805 EC                  1616            IN      AL,DX           ; GET STATUS
E806 A801                1617            TEST    AL,1            ; RECEIVE BUFFER FULL
E808 7509                1618            JNZ     A17             ; GET CHAR
E80A F606710080      R   1619            TEST    BIOS_BREAK,80H  ; TEST FOR BREAK KEY
E80F 74F4                1620            JZ      A16             ; LOOP IF NOT
E811 EBEC                1621            JMP     A14             ; SET TIME OUT ERROR
E813                     1622    A17:                            ; GET_CHAR
E813 241E                1623            AND     AL,00011110B    ; TEST FOR ERROR CONDITIONS ON RECV CHAR
E815 8AE0                1624            MOV     AH,AL           ; SAVE THIS PART OF STATUS FOR LATER OPERATION
E817 8B940000        R   1625            MOV     DX,RS232_BASE[SI]       ; DATA PORT
E81B EC                  1626            IN      AL,DX           ; GET CHARACTER FROM LINE
E81C E947FF              1627            JMP     A3              ; RETURN
                         1628
                         1629    ;------ COMMO PORT STATUS ROUTINE
                         1630
E81F                     1631    A18:
E81F 8B940000        R   1632            MOV     DX,RS232_BASE[SI]
E823 83C205              1633            ADD     DX,5            ; CONTROL PORT
E826 EC                  1634            IN      AL,DX           ; GET LINE CONTROL STATUS
E827 8AE0                1635            MOV     AH,AL           ; PUT IN AH FOR RETURN
E829 42                  1636            INC     DX              ; POINT TO MODEM STATUS REGISTER
E82A EC                  1637            IN      AL,DX           ; GET MODEM CONTROL STATUS
E82B E938FF              1638            JMP     A3              ; RETURN
                         1639    RS232_IO        ENDP
                         1640    ;---- INT 16 ---------------------------------
                         1641    ; KEYBOARD I/O
                         1642    ;       THESE ROUTINES PROVIDE KEYBOARD SUPPORT
                         1643    ; INPUT
                         1644    ;       (AH)=0  READ THE NEXT ASCII CHARACTER STRUCK FROM THE KEYBOARD
                         1645    ;               RETURN THE RESULT IN (AL), SCAN CODE IN (AH)
                         1646    ;       (AH)=1  SET THE Z FLAG TO INDICATE IF AN ASCII CHARACTER IS AVAILABLE
                         1647    ;               TO BE READ.
                         1648    ;               (ZF)=1 -- NO CODE AVAILABLE
                         1649    ;               (ZF)=0 -- CODE IS AVAILABLE
                         1650    ;               IF ZF = 0, THE NEXT CHARACTER IN THE BUFFER TO BE READ IS
                         1651    ;               IN AX, AND THE ENTRY REMAINS IN THE BUFFER
                         1652    ;       (AH)=2  RETURN THE CURRENT SHIFT STATUS IN AL REGISTER
                         1653    ;               THE BIT SETTINGS FOR THIS CODE ARE INDICATED IN THE
                         1654    ;               THE EQUATES FOR KB_FLAG
                         1655    ; OUTPUT
                         1656    ;       AS NOTED ABOVE, ONLY AX AND FLAGS CHANGED
                         1657    ;       ALL REGISTERS RETAINED
                         1658    ;----------------------------------------
```

```
LOC OBJ                  LINE    SOURCE


                         1659            ASSUME  CS:CODE,DS:DATA
E82E                     1660    KEYBOARD_IO     PROC    FAR
E82E FB                  1661            STI                     ; INTERRUPTS BACK ON
E82F 1E                  1662            PUSH    DS              ; SAVE CURRENT DS
E830 53                  1663            PUSH    BX              ; SAVE BX TEMPORARILY
E831 BB4000          R   1664            MOV     BX,DATA         ;
E834 8EDB                1665            MOV     DS,BX           ; ESTABLISH POINTER TO DATA REGION
E836 0AE4                1666            OR      AH,AH           ; AH=0
E838 740B                1667            JZ      K1              ; ASCII_READ
E83A FECC                1668            DEC     AH              ; AH=1
E83C 7420                1669            JZ      K2              ; ASCII_STATUS
E83E FECC                1670            DEC     AH              ; AH=2
E840 742D                1671            JZ      K3              ; SHIFT_STATUS
E842 5B                  1672            POP     BX              ; RECOVER REGISTER
E843 1F                  1673            POP     DS
E844 CF                  1674            IRET                    ; INVALID COMMAND
                         1675
                         1676    ;------ READ THE KEY TO FIGURE OUT WHAT TO DO
                         1677
E845                     1678    K1:                             ; ASCII READ
E845 FB                  1679            STI                     ; INTERRUPTS BACK ON DURING LOOP
E846 90                  1680            NOP                     ; ALLOW AN INTERRUPT TO OCCUR
E847 FA                  1681            CLI                     ; INTERRUPTS BACK OFF
E848 8B1E1A00        R   1682            MOV     BX,BUFFER_HEAD  ; GET POINTER TO HEAD OF BUFFER
E84C 3B1E1C00        R   1683            CMP     BX,BUFFER_TAIL  ; TEST END OF BUFFER
E850 74F3                1684            JZ      K1              ; LOOP UNTIL SOMETHING IN BUFFER
E852 8B07                1685            MOV     AX,[BX]         ; GET SCAN CODE AND ASCII CODE
E854 E81E00              1686            CALL    K4              ; MOVE POINTER TO NEXT POSITION
E857 891E1A00        R   1687            MOV     BUFFER_HEAD,BX  ; STORE VALUE IN VARIABLE
E85B 5B                  1688            POP     BX              ; RECOVER REGISTER
E85C 1F                  1689            POP     DS              ; RECOVER SEGMENT
E85D CF                  1690            IRET                    ; RETURN TO CALLER
                         1691
                         1692    ;------ ASCII STATUS
                         1693
E85E                     1694    K2:
E85E FA                  1695            CLI                     ; INTERRUPTS OFF
E85F 8B1E1A00        R   1696            MOV     BX,BUFFER_HEAD  ; GET HEAD POINTER
E863 3B1E1C00        R   1697            CMP     BX,BUFFER_TAIL  ; IF EQUAL (Z=1) THEN NOTHING THERE
E867 8B07                1698            MOV     AX,[BX]
E869 FB                  1699            STI                     ; INTERRUPTS BACK ON
E86A 5B                  1700            POP     BX              ; RECOVER REGISTER
E86B 1F                  1701            POP     DS              ; RECOVER SEGMENT
E86C CA0200              1702            RET     2               ; THROW AWAY FLAGS
                         1703
                         1704    ;------ SHIFT STATUS
                         1705
E86F                     1706    K3:
E86F A01700          R   1707            MOV     AL,KB_FLAG      ; GET THE SHIFT STATUS FLAGS
E872 5B                  1708            POP     BX              ; RECOVER REGISTER
E873 1F                  1709            POP     DS              ; RECOVER REGISTERS
E874 CF                  1710            IRET                    ; RETURN TO CALLER
                         1711    KEYBOARD_IO     ENDP
                         1712
                         1713    ;------ INCREMENT A BUFFER POINTER
                         1714
E875                     1715    K4      PROC    NEAR
E875 83C302              1716            ADD     BX,2                    ; MOVE TO NEXT WORD IN LIST
E878 81FB3E00        R   1717            CMP     BX,OFFSET KB_BUFFER_END ; AT END OF BUFFER?
E87C 7503                1718            JNE     K5                      ; NO, CONTINUE
E87E BB1E00          R   1719            MOV     BX,OFFSET KB_BUFFER     ; YES, RESET TO BUFFER BEGINNING
E881                     1720    K5:
E881 C3                  1721            RET
                         1722    K4      ENDP
                         1723
                         1724    ;------ TABLE OF SHIFT KEYS AND MASK VALUES
                         1725
E882                     1726    K6      LABEL   BYTE
E882 52                  1727            DB      INS_KEY                 ; INSERT KEY
E883 3A4546381D          1728            DB      CAPS_KEY,NUM_KEY,SCROLL_KEY,ALT_KEY,CTL_KEY
E888 2A36                1729            DB      LEFT_KEY,RIGHT_KEY
  0008                   1730    K6L     EQU     $-K6
                         1731
                         1732    ;------ SHIFT_MASK_TABLE
                         1733
E88A                     1734    K7      LABEL   BYTE
E88A 80                  1735            DB      INS_SHIFT               ; INSERT MODE SHIFT
```

```
LOC OBJ                  LINE    SOURCE

E88B 4020100804          1736            DB      CAPS_SHIFT,NUM_SHIFT,SCROLL_SHIFT,ALT_SHIFT,CTL_SHIFT
E890 0201                1737            DB      LEFT_SHIFT,RIGHT_SHIFT
                         1738
                         1739    ;------ SCAN CODE TABLES
                         1740
E892 1BFF00FFFFFF1EFF    1741    K8              DB      27,-1,0,-1,-1,-1,30,-1
E89A FFFFFF1FFF7FFF11    1742                    DB      -1,-1,-1,31,-1,127,-1,17
E8A2 1705121419150900F   1743                    DB      23,5,18,20,25,21,9,15
E8AA 101B1D0AFF0113      1744                    DB      16,27,29,10,-1,1,19
E8B1 040607080A0B0CFFFF  1745                    DB      4,6,7,8,10,11,12,-1,-1
E8BA FFFF1C1A18031602    1746                    DB      -1,-1,28,26,24,3,22,2
E8C2 0E0DFFFFFFFFFFFF    1747                    DB      14,13,-1,-1,-1,-1,-1,-1
E8CA 20FF                1748                    DB      ' ',-1
                         1749    ;-------- CTL TABLE SCAN
E8CC                     1750    K9      LABEL   BYTE
E8CC 5E5F606162636465    1751                    DB      94,95,96,97,98,99,100,101
E8D4 6667FFFF77FF84FF    1752                    DB      102,103,-1,-1,119,-1,132,-1
E8DC 73FF74FF75FF76FF    1753                    DB      115,-1,116,-1,117,-1,118,-1
E8E4 FF                  1754                    DB      -1
                         1755    ;------- LC TABLE
E8E5                     1756    K10     LABEL   BYTE
E8E5 1B                  1757                    DB      01BH,'1234567890-=',08H,09H
E8E6 31323334353637
     3839302D3D08090
E8F4 71776572747975      1758                    DB      'quertyuiop[]',0DH,-1,'asdfghjkl;',027H
     696F705B5D0DFF
E902 6173646667686A
     6B6C3B27
E90D 60FF5C              1759                    DB      60H,-1,5CH,'zxcvbnm,./',-1,'*',-1,' '
E910 7A7863766268E6D
     2C2E2FFF2AFF20
E91E FF                  1760                    DB      -1
                         1761
                         1762    ;------ UC TABLE
E91F                     1763    K11     LABEL   BYTE
E91F 1B                  1764                    DB      27,'!@#$',37,05EH,'&*()_+',08H,0
E920 21402324255E
E926 262A28295F2B0800
E92E 515745525455955     1765                    DB      'QWERTYUIOP  ',0DH,-1,'ASDFGHJKL:"'
     494F507B7D0DFF
E93C 4153444647484A
     4B4C3A22
E947 7EFF                1766                    DB      07EH,-1,' ZXCVBNM<>?',-1,0,-1,' ',-1
E949 7C5A5843564246E
     4D3C3E3FFF00FF20FF
                         1767    ;------ UC TABLE SCAN
E959                     1768    K12     LABEL   BYTE
E959 5455565758595A      1769                    DB      84,85,86,87,88,89,90
E960 5B5C5D              1770                    DB      91,92,93
                         1771    ;------ ALT TABLE SCAN
E963                     1772    K13     LABEL   BYTE
E963 68696A6B6C          1773                    DB      104,105,106,107,108
E968 6D6E6F7071          1774                    DB      109,110,111,112,113
                         1775
                         1776    ;------ NUM STATE TABLE
E96D                     1777    K14     LABEL   BYTE
E96D 3738392D343536      1778                    DB      '789-456+1230.'
     2B313233302E
                         1779    ;------ BASE CASE TABLE
E97A                     1780    K15     LABEL   BYTE
E97A 474849FF4BFF4D      1781                    DB      71,72,73,-1,75,-1,77
E981 FF4F50515253        1782                    DB      -1,79,80,81,82,83
                         1783
                         1784    ;------ KEYBOARD INTERRUPT ROUTINE
                         1785
E987                     1786    KB_INT  PROC    FAR
E987 FB                  1787            STI                     ; ALLOW FURTHER INTERRUPTS
E988 50                  1788            PUSH    AX
E989 53                  1789            PUSH    BX
E98A 51                  1790            PUSH    CX
E98B 52                  1791            PUSH    DX
E98C 56                  1792            PUSH    SI
E98D 57                  1793            PUSH    DI
E98E 1E                  1794            PUSH    DS
E98F 06                  1795            PUSH    ES
E990 FC                  1796            CLD                     ; FORWARD DIRECTION
E991 B84000        R     1797            MOV     AX,DATA
```

```
LOC OBJ                 LINE    SOURCE

E994 8ED8               1798            MOV     DS,AX           ; SET UP ADDRESSING
E996 E460               1799            IN      AL,KB_DATA      ; READ IN THE CHARACTER
E998 50                 1800            PUSH    AX              ; SAVE IT
E999 E461               1801            IN      AL,KB_CTL       ; GET THE CONTROL PORT
E99B 8AE0               1802            MOV     AH,AL           ; SAVE VALUE
E99D 0C80               1803            OR      AL,80H          ; RESET BIT FOR KEYBOARD
E99F E661               1804            OUT     KB_CTL,AL
E9A1 86E0               1805            XCHG    AH,AL           ; GET BACK ORIGINAL CONTROL
E9A3 E661               1806            OUT     KB_CTL,AL       ; KB HAS BEEN RESET
E9A5 58                 1807            POP     AX              ; RECOVER SCAN CODE
E9A6 8AE0               1808            MOV     AH,AL           ; SAVE SCAN CODE IN AH ALSO
                        1809
                        1810    ;------ TEST FOR OVERRUN SCAN CODE FROM KEYBOARD
                        1811
E9A8 3CFF               1812            CMP     AL,0FFH         ; IS THIS AN OVERRUN CHAR
E9AA 7503               1813            JNZ     K16             ; NO, TEST FOR SHIFT KEY
E9AC E97502             1814            JMP     K62             ; BUFFER_FULL_BEEP
                        1815
                        1816    ;------ TEST FOR SHIFT KEYS
                        1817
E9AF                    1818    K16:                            ; TEST_SHIFT
E9AF 247F               1819            AND     AL,07FH         ; TURN OFF THE BREAK BIT
E9B1 0E                 1820            PUSH    CS
E9B2 07                 1821            POP     ES              ; ESTABLISH ADDRESS OF SHIFT TABLE
E9B3 BF82E8         R   1822            MOV     DI,OFFSET K6    ; SHIFT KEY TABLE
E9B6 B90800             1823            MOV     CX,K6L          ; LENGTH
E9B9 F2                 1824            REPNE   SCASB           ; LOOK THROUGH THE TABLE FOR A MATCH
E9BA AE
E9BB 8AC4               1825            MOV     AL,AH           ; RECOVER SCAN CODE
E9BD 7403               1826            JE      K17             ; JUMP IF MATCH FOUND
E9BF E98800             1827            JMP     K25             ; IF NO MATCH, THEN SHIFT NOT FOUND
                        1828
                        1829    ;------ SHIFT KEY FOUND
                        1830
E9C2 81EF83E8       R   1831    K17:    SUB     DI,OFFSET K6+1  ; ADJUST PTR TO SCAN CODE MTCH
E9C6 2E8AA58AE8     R   1832            MOV     AH,CS:K7[DI]    ; GET MASK INTO AH
E9CB A880               1833            TEST    AL,80H          ; TEST FOR BREAK KEY
E9CD 7554               1834            JNZ     K23             ; BREAK_SHIFT_FOUND
                        1835
                        1836    ;------ SHIFT MAKE FOUND, DETERMINE SET OR TOGGLE
                        1837
E9CF 80FC10             1838            CMP     AH,SCROLL_SHIFT
E9D2 7307               1839            JAE     K18                     ; IF SCROLL SHIFT OR ABOVE, TOGGLE KEY
                        1840
                        1841    ;------ PLAIN SHIFT KEY, SET SHIFT ON
                        1842
E9D4 08261700       R   1843            OR      KB_FLAG,AH              ; TURN ON SHIFT BIT
E9D8 E98300             1844            JMP     K26                     ; INTERRUPT_RETURN
                        1845
                        1846    ;------ TOGGLED SHIFT KEY, TEST FOR 1ST MAKE OR NOT
                        1847
E9DB                    1848    K18:                                    ; SHIFT-TOGGLE
E9DB F606170004     R   1849            TEST    KB_FLAG, CTL_SHIFT      ; CHECK CTL SHIFT STATE
E9E0 7568               1850            JNZ     K25                     ; JUMP IF CTL STATE
E9E2 3C52               1851            CMP     AL, INS_KEY             ; CHECK FOR INSERT KEY
E9E4 7525               1852            JNZ     K22                     ; JUMP IF NOT INSERT KEY
E9E6 F606170008     R   1853            TEST    KB_FLAG, ALT_SHIFT      ; CHECK FOR ALTERNATE SHIFT
E9EB 7403               1854            JZ      K19                     ; JUMP IF NOT ALTERNATE SHIFT
E9ED EB5B90             1855            JMP     K25                     ; JUMP IF ALTERNATE SHIFT
E9F0 F606170020     R   1856    K19:    TEST    KB_FLAG, NUM_STATE      ; CHECK FOR BASE STATE
E9F5 750D               1857            JNZ     K21                     ; JUMP IF NUM LOCK IS ON
E9F7 F606170003     R   1858            TEST    KB_FLAG, LEFT_SHIFT+ RIGHT_SHIFT        ;
E9FC 740D               1859            JZ      K22                     ; JUMP IF BASE STATE
                        1860
E9FE                    1861    K20:                                    ; NUMERIC ZERO, NOT INSERT KEY
E9FE B83052             1862            MOV     AX, 5230H               ; PUT OUT AN ASCII ZERO
EA01 E9D801             1863            JMP     K57                     ; BUFFER_FILL
EA04                    1864    K21:                                    ; MIGHT BE NUMERIC
EA04 F606170003     R   1865            TEST    KB_FLAG, LEFT_SHIFT+ RIGHT_SHIFT        ;
EA09 74F3               1866            JZ      K20                     ; JUMP NUMERIC, NOT INSERT
                        1867
EA0B                    1868    K22:                                    ; SHIFT TOGGLE KEY HIT; PROCESS IT
EA0B 84261800       R   1869            TEST    AH,KB_FLAG_1            ; IS KEY ALREADY DEPRESSED
EA0F 754D               1870            JNZ     K26                     ; JUMP IF KEY ALREADY DEPRESSED
EA11 08261800       R   1871            OR      KB_FLAG_1,AH            ; INDICATE THAT THE KEY IS DEPRESSED
EA15 30261700       R   1872            XOR     KB_FLAG,AH              ; TOGGLE THE SHIFT STATE
EA19 3C52               1873            CMP     AL,INS_KEY              ; TEST FOR 1ST MAKE OF INSERT KEY
EA1B 7541               1874            JNE     K26                     ; JUMP IF NOT INSERT KEY
```

```
LOC OBJ                 LINE    SOURCE


EA1D B80052             1875            MOV     AX,INS_KEY*256          ; SET SCAN CODE INTO AH, 0 INTO AL
EA20 E9B901             1876            JMP     K57                     ; PUT INTO OUTPUT BUFFER
                        1877
                        1878    ;------ BREAK SHIFT FOUND
                        1879
EA23                    1880    K23:                                    ; BREAK-SHIFT-FOUND
EA23 80FC10             1881            CMP     AH,SCROLL_SHIFT         ; IS THIS A TOGGLE KEY
EA26 731A               1882            JAE     K24                     ; YES, HANDLE BREAK TOGGLE
EA28 F6D4               1883            NOT     AH                      ; INVERT MASK
EA2A 20261700       R   1884            AND     KB_FLAG,AH              ; TURN OFF SHIFT BIT
EA2E 3C88               1885            CMP     AL,ALT_KEY+80H          ; IS THIS ALTERNATE SHIFT RELEASE
EA30 752C               1886            JNE     K26                     ; INTERRUPT_RETURN
                        1887
                        1888    ;------ ALTERNATE SHIFT KEY RELEASED, GET THE VALUE INTO BUFFER
                        1889
EA32 A01900         R   1890            MOV     AL,ALT_INPUT
EA35 B400               1891            MOV     AH,0                    ; SCAN CODE OF 0
EA37 88261900       R   1892            MOV     ALT_INPUT,AH            ; ZERO OUT THE FIELD
EA3B 3C00               1893            CMP     AL,0                    ; WAS THE INPUT=0
EA3D 741F               1894            JE      K26                     ; INTERRUPT_RETURN
EA3F E9A301             1895            JMP     K58                     ; IT WASN'T, SO PUT IN BUFFER
                        1896
EA42                    1897    K24:                                    ; BREAK-TOGGLE
EA42 F6D4               1898            NOT     AH                      ; INVERT MASK
EA44 20261800       R   1899            AND     KB_FLAG_1,AH            ; INDICATE NO LONGER DEPRESSED
EA48 EB14               1900            JMP     SHORT K26               ; INTERRUPT_RETURN
                        1901
                        1902    ;------ TEST FOR HOLD STATE
                        1903
EA4A                    1904    K25:                                    ; NO-SHIFT-FOUND
EA4A 3C80               1905            CMP     AL,80H                  ; TEST FOR BREAK KEY
EA4C 7310               1906            JAE     K26                     ; NOTHING FOR BREAK CHARS FROM HERE ON
EA4E F606180008     R   1907            TEST    KB_FLAG_1,HOLD_STATE    ; ARE WE IN HOLD STATE
EA53 7417               1908            JZ      K28                     ; BRANCH AROUND TEST IF NOT
EA55 3C45               1909            CMP     AL,NUM_KEY
EA57 7405               1910            JE      K26                     ; CAN'T END HOLD ON NUM_LOCK
EA59 80261800F7     R   1911            AND     KB_FLAG_1,NOT HOLD_STATE        ; TURN OFF THE HOLD STATE BIT
                        1912
EA5E                    1913    K26:                                    ; INTERRUPT-RETURN
EA5E FA                 1914            CLI                     ; TURN OFF INTERRUPTS
EA5F B020               1915            MOV     AL,EOI          ; END OF INTERRUPT COMMAND
EA61 E620               1916            OUT     020H,AL         ; SEND COMMAND TO INTERRUPT CONTROL PORT
EA63                    1917    K27:                            ; INTERRUPT-RETURN-NO-EOI
EA63 07                 1918            POP     ES
EA64 1F                 1919            POP     DS
EA65 5F                 1920            POP     DI
EA66 5E                 1921            POP     SI
EA67 5A                 1922            POP     DX
EA68 59                 1923            POP     CX
EA69 5B                 1924            POP     BX
EA6A 58                 1925            POP     AX              ; RESTORE STATE
EA6B CF                 1926            IRET                    ; RETURN, INTERRUPTS BACK ON WITH FLAG CHANGE
                        1927
                        1928    ;------ NOT IN  HOLD STATE, TEST FOR SPECIAL CHARS
                        1929
EA6C                    1930    K28:                            ; NO-HOLD-STATE
EA6C F606170008     R   1931            TEST    KB_FLAG,ALT_SHIFT       ; ARE WE IN ALTERNATE SHIFT
EA71 7503               1932            JNZ     K29                     ; JUMP IF ALTERNATE SHIFT
EA73 E98F00             1933            JMP     K38                     ; JUMP IF NOT ALTERNATE
                        1934
                        1935    ;------ TEST FOR RESET KEY SEQUENCE (CTL ALT DEL)
                        1936
EA76                    1937    K29:                                    ; TEST-RESET
EA76 F606170004     R   1938            TEST    KB_FLAG,CTL_SHIFT       ; ARE WE IN CONTROL SHIFT ALSO
EA7B 7431               1939            JZ      K31                     ; NO_RESET
EA7D 3C53               1940            CMP     AL,DEL_KEY              ; SHIFT STATE IS THERE, TEST KEY
EA7F 752D               1941            JNE     K31                     ; NO_RESET
                        1942
                        1943    ;------ CTL-ALT-DEL HAS BEEN FOUND, DO I/O CLEANUP
                        1944
EA81 C706720034 12  R   1945            MOV     RESET_FLAG, 1234H       ; SET FLAG FOR RESET FUNCTION
EA87 E9D1F5             1946            JMP     RESET                   ; JUMP TO POWER ON DIAGNOSTICS
                        1947
                        1948    ;------ ALT-INPUT-TABLE
EA8A                    1949    K30     LABEL   BYTE
EA8A 524F50514B4C4D     1950            DB      82,79,80,81,75,76,77
```

```
LOC OBJ                  LINE    SOURCE

EA91 474849              1951            DB      71,72,73                ; 10 NUMBERS ON KEYPAD
                         1952    ;------ SUPER-SHIFT-TABLE
EA94 1011121314151617    1953            DB      16,17,18,19,20,21,22,23 ; A-Z TYPEWRITER CHARS
EA9C 18191E1F20212223    1954            DB      24,25,30,31,32,33,34,35
EAA4 2425262C2D2E2F30    1955            DB      36,37,38,44,45,46,47,48
EAAC 3132                1956            DB      49,50
                         1957
                         1958    ;------ IN ALTERNATE SHIFT, RESET NOT FOUND
                         1959
EAAE                     1960    K31:                                    ; NO-RESET
EAAE 3C39                1961            CMP     AL,57                   ; TEST FOR SPACE KEY
EAB0 7505                1962            JNE     K32                     ; NOT THERE
EAB2 B020                1963            MOV     AL,' '                  ; SET SPACE CHAR
EAB4 E92501              1964            JMP     K57                     ; BUFFER_FILL
                         1965
                         1966    ;------ LOOK FOR KEY PAD ENTRY
                         1967
EAB7                     1968    K32:                                    ; ALT-KEY-PAD
EAB7 BF8AEA          R   1969            MOV     DI,OFFSET K30           ; ALT-INPUT-TABLE
EABA B90A00              1970            MOV     CX,10                   ; LOOK FOR ENTRY USING KEYPAD
EABD F2                  1971            REPNE   SCASB                   ; LOOK FOR MATCH
EABE AE
EABF 7512                1972            JNE     K33                     ; NO_ALT_KEYPAD
EAC1 81EF8BEA        R   1973            SUB     DI,OFFSET K30+1         ; DI NOW HAS ENTRY VALUE
EAC5 A01900          R   1974            MOV     AL,ALT_INPUT            ; GET THE CURRENT BYTE
EAC8 B40A                1975            MOV     AH,10                   ; MULTIPLY BY 10
EACA F6E4                1976            MUL     AH
EACC 03C7                1977            ADD     AX,DI                   ; ADD IN THE LATEST ENTRY
EACE A21900          R   1978            MOV     ALT_INPUT,AL            ; STORE IT AWAY
EAD1 EB8B                1979            JMP     K26                     ; THROW AWAY THAT KEYSTROKE
                         1980
                         1981    ;------ LOOK FOR SUPERSHIFT ENTRY
                         1982
EAD3                     1983    K33:                                    ; NO-ALT-KEYPAD
EAD3 C606190000      R   1984            MOV     ALT_INPUT,0             ; ZERO ANY PREVIOUS ENTRY INTO INPUT
EAD8 B91A00              1985            MOV     CX,26                   ; DI,ES ALREADY POINTING
EADB F2                  1986            REPNE   SCASB                   ; LOOK FOR MATCH IN ALPHABET
EADC AE
EADD 7505                1987            JNE     K34                     ; NOT FOUND, FUNCTION KEY OR OTHER
EADF B000                1988            MOV     AL,0                    ; ASCII CODE OF ZERO
EAE1 E9F800              1989            JMP     K57                     ; PUT IT IN THE BUFFER
                         1990
                         1991    ;------ LOOK FOR TOP ROW OF ALTERNATE SHIFT
                         1992
EAE4                     1993    K34:                                    ; ALT-TOP-ROW
EAE4 3C02                1994            CMP     AL,2                    ; KEY WITH '1' ON IT
EAE6 720C                1995            JB      K35                     ; NOT ONE OF INTERESTING KEYS
EAE8 3C0E                1996            CMP     AL,14                   ; IS IT IN THE REGION
EAEA 7308                1997            JAE     K35                     ; ALT-FUNCTION
EAEC 80C476              1998            ADD     AH,118                  ; CONVERT PSUEDO SCAN CODE TO RANGE
EAEF B000                1999            MOV     AL,0                    ; INDICATE AS SUCH
EAF1 E9E800              2000            JMP     K57                     ; BUFFER_FILL
                         2001
                         2002    ;------ TRANSLATE ALTERNATE SHIFT PSEUDO SCAN CODES
                         2003
EAF4                     2004    K35:                                    ; ALT-FUNCTION
EAF4 3C3B                2005            CMP     AL,59                   ; TEST FOR IN TABLE
EAF6 7303                2006            JAE     K37                     ; ALT-CONTINUE
EAF8                     2007    K36:                                    ; CLOSE-RETURN
EAF8 E963FF              2008            JMP     K26                     ; IGNORE THE KEY
EAFB                     2009    K37:                                    ; ALT-CONTINUE
EAFB 3C47                2010            CMP     AL,71                   ; IN KEYPAD REGION
EAFD 73F9                2011            JAE     K36                     ; IF SO, IGNORE
EAFF BB63E9          R   2012            MOV     BX,OFFSET K13           ; ALT SHIFT PSEUDO SCAN TABLE
EB02 E92501              2013            JMP     K63                     ; TRANSLATE THAT
                         2014
                         2015    ;------ NOT IN ALTERNATE SHIFT
                         2016
EB05                     2017    K38:                                    ; NOT-ALT-SHIFT
EB05 F606170004      R   2018            TEST    KB_FLAG,CTL_SHIFT       ; ARE WE IN CONTROL SHIFT
EB0A 745B                2019            JZ      K44                     ; NOT-CTL-SHIFT
                         2020
                         2021    ;------ CONTROL SHIFT, TEST SPECIAL CHARACTERS
                         2022    ;------ TEST FOR BREAK AND PAUSE KEYS
                         2023
EB0C 3C46                2024            CMP     AL,SCROLL_KEY           ; TEST FOR BREAK
EB0E 7518                2025            JNE     K39                     ; NO-BREAK
```

```
LOC OBJ                 LINE    SOURCE

EB10 BB1E00       R     2026            MOV     BX,OFFSET KB_BUFFER     ; RESET BUFFER TO EMPTY
EB13 891E1A00     R     2027            MOV     BUFFER_HEAD,BX          ;
EB17 891E1C00     R     2028            MOV     BUFFER_TAIL,BX          ;
EB1B C6067100B0   R     2029            MOV     BIOS_BREAK,80H          ; TURN ON BIOS_BREAK BIT
EB20 CD1B               2030            INT     1BH                     ; BREAK INTERRUPT VECTOR
EB22 B80000             2031            MOV     AX, 0                   ; PUT OUT DUMMY CHARACTER
EB25 E9B400             2032            JMP     K57                     ; BUFFER_FILL
                        2033
EB28                    2034    K39:                                    ; NO-BREAK
EB28 3C45               2035            CMP     AL,NUM_KEY              ; LOOK FOR PAUSE KEY
EB2A 7521               2036            JNE     K41                     ; NO-PAUSE
EB2C 800E180008   R     2037            OR      KB_FLAG_1,HOLD_STATE    ; TURN ON THE HOLD FLAG
EB31 B020               2038            MOV     AL,EOI                  ; END OF INTERRUPT TO CONTROL PORT
EB33 E620               2039            OUT     020H,AL                 ; ALLOW FURTHER KEYSTROKE INTS
                        2040
                        2041    ;------ DURING PAUSE INTERVAL, TURN CRT BACK ON
                        2042
EB35 803E490007   R     2043            CMP     CRT_MODE,7              ; IS THIS BLACK AND WHITE CARD
EB3A 7407               2044            JE      K40                     ; YES, NOTHING TO DO
EB3C BAD803             2045            MOV     DX,03D8H                ; PORT FOR COLOR CARD
EB3F A06500       R     2046            MOV     AL,CRT_MODE_SET         ; GET THE VALUE OF THE CURRENT MODE
EB42 EE                 2047            OUT     DX,AL                   ; SET THE CRT MODE, SO THAT CRT IS ON
EB43                    2048    K40:                                    ; PAUSE-LOOP
EB43 F606180008   R     2049            TEST    KB_FLAG_1,HOLD_STATE
EB48 75F9               2050            JNZ     K40                     ; LOOP UNTIL FLAG TURNED OFF
EB4A E916FF             2051            JMP     K27                     ; INTERRUPT_RETURN_NO_EOI
EB4D                    2052    K41:                                    ; NO-PAUSE
                        2053
                        2054    ;------ TEST SPECIAL CASE KEY 55
                        2055
EB4D 3C37               2056            CMP     AL,55
EB4F 7506               2057            JNE     K42                     ; NOT-KEY-55
EB51 B80072             2058            MOV     AX,114*256              ; START/STOP PRINTING SWITCH
EB54 E98500             2059            JMP     K57                     ; BUFFER_FILL
                        2060
                        2061    ;------ SET UP TO TRANSLATE CONTROL SHIFT
                        2062
EB57                    2063    K42:                                    ; NOT-KEY-55
EB57 BB92E8       R     2064            MOV     BX,OFFSET K8            ; SET UP TO TRANSLATE CTL
EB5A 3C3B               2065            CMP     AL,59                   ; IS IT IN TABLE
EB5C 7303               2066            JAE     K43                     ; CTL-TABLE-TRANSLATE
EB5E EB7890             2067            JMP     K56                     ; YES, GO TRANSLATE CHAR
EB61                    2068    K43:                                    ; CTL-TABLE-TRANSLATE
EB61 BBCCE8       R     2069            MOV     BX,OFFSET K9            ; CTL TABLE SCAN
EB64 E9C300             2070            JMP     K63                     ; TRANSLATE_SCAN
                        2071
                        2072    ;------ NOT IN CONTROL SHIFT
                        2073
EB67                    2074    K44:                                    ; NOT-CTL-SHIFT
                        2075
EB67 3C47               2076            CMP     AL,71                   ; TEST FOR KEYPAD REGION
EB69 732D               2077            JAE     K48                     ; HANDLE KEYPAD REGION
EB6B F606170003   R     2078            TEST    KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT
EB70 745B               2079            JZ      K54                     ; TEST FOR SHIFT STATE
                        2080
                        2081    ;------ UPPER CASE, HANDLE SPECIAL CASES
                        2082
EB72 3C0F               2083            CMP     AL,15                   ; BACK TAB KEY
EB74 7506               2084            JNE     K45                     ; NOT-BACK-TAB
EB76 B8000F             2085            MOV     AX,15*256               ; SET PSEUDO SCAN CODE
EB79 EB6190             2086            JMP     K57                     ; BUFFER_FILL
                        2087
EB7C                    2088    K45:                                    ; NOT-BACK-TAB
EB7C 3C37               2089            CMP     AL,55                   ; PRINT SCREEN KEY
EB7E 7509               2090            JNE     K46                     ; NOT-PRINT-SCREEN
                        2091
                        2092    ;------ ISSUE INTERRUPT TO INDICATE PRINT SCREEN FUNCTION
                        2093
EB80 B020               2094            MOV     AL,EOI                  ; END OF CURRENT INTERRUPT
EB82 E620               2095            OUT     020H,AL                 ;  SO FURTHER THINGS CAN HAPPEN
EB84 CD05               2096            INT     5H                      ; ISSUE PRINT SCREEN INTERRUPT
EB86 E9DAFE             2097            JMP     K27                     ; GO BACK WITHOUT EOI OCCURRING
                        2098
EB89                    2099    K46:                                    ; NOT-PRINT-SCREEN
EB89 3C3B               2100            CMP     AL,59                   ; FUNCTION KEYS
EB8B 7206               2101            JB      K47                     ; NOT-UPPER-FUNCTION
EB8D BB59E9       R     2102            MOV     BX,OFFSET K12           ; UPPER CASE PSEUDO SCAN CODES
```

```
LOC OBJ                  LINE    SOURCE


EB90 E99700              2103            JMP     K63                     ; TRANSLATE_SCAN
                         2104
EB93                     2105    K47:                                    ; NOT-UPPER-FUNCTION
EB93 BB1FE9        R     2106            MOV     BX,OFFSET K11           ; POINT TO UPPER CASE TABLE
EB96 EB40                2107            JMP     SHORT K56               ; OK, TRANSLATE THE CHAR
                         2108
                         2109    ;------ KEYPAD KEYS, MUST TEST NUM LOCK FOR DETERMINATION
                         2110
EB98                     2111    K48:                                    ; KEYPAD-REGION
EB98 F606170020    R     2112            TEST    KB_FLAG,NUM_STATE       ; ARE WE IN NUM_LOCK
EB9D 7520                2113            JNZ     K52                     ; TEST FOR SURE
EB9F F606170003    R     2114            TEST    KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT  ; ARE WE IN SHIFT STATE
EBA4 7520                2115            JNZ     K53                     ; IF SHIFTED, REALLY NUM STATE
                         2116
                         2117    ;------ BASE CASE FOR KEYPAD
                         2118
EBA6                     2119    K49:                                    ; BASE-CASE
                         2120                                                    ;
EBA6 3C4A                2121            CMP     AL,74                   ; SPECIAL CASE FOR A COUPLE OF KEYS
EBA8 740B                2122            JE      K50                     ; MINUS
EBAA 3C4E                2123            CMP     AL,78
EBAC 740C                2124            JE      K51
EBAE 2C47                2125            SUB     AL,71                   ; CONVERT ORIGIN
EBB0 BB7AE9        R     2126            MOV     BX,OFFSET K15           ; BASE CASE TABLE
EBB3 EB77                2127            JMP     SHORT K64               ; CONVERT TO PSEUDO SCAN
                         2128
EBB5 B82D4A              2129    K50:    MOV     AX,74*256+'-'           ; MINUS
EBB8 EB22                2130            JMP     SHORT K57               ; BUFFER_FILL
                         2131
EBBA B82B4E              2132    K51:    MOV     AX,78*256+'+'           ; PLUS
EBBD EB1D                2133            JMP     SHORT K57               ; BUFFER_FILL
                         2134
                         2135    ;------ MIGHT BE NUM LOCK, TEST SHIFT STATUS
                         2136
EBBF                     2137    K52:                                    ; ALMOST-NUM-STATE
EBBF F606170003    R     2138            TEST    KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT
EBC4 75E0                2139            JNZ     K49                     ; SHIFTED TEMP OUT OF NUM STATE
                         2140
EBC6                     2141    K53:                                    ; REALLY_NUM_STATE
EBC6 2C46                2142            SUB     AL,70                   ; CONVERT ORIGIN
EBC8 BB6DE9        R     2143            MOV     BX,OFFSET K14           ; NUM STATE TABLE
EBCB EB0B                2144            JMP     SHORT K56               ; TRANSLATE_CHAR
                         2145
                         2146    ;------ PLAIN OLD LOWER CASE
                         2147
EBCD                     2148    K54:                                    ; NOT-SHIFT
EBCD 3C3B                2149            CMP     AL,59                   ; TEST FOR FUNCTION KEYS
EBCF 7204                2150            JB      K55                     ; NOT-LOWER-FUNCTION
EBD1 B000                2151            MOV     AL,0                    ; SCAN CODE IN AH ALREADY
EBD3 EB07                2152            JMP     SHORT K57               ; BUFFER_FILL
                         2153
EBD5                     2154    K55:                                    ; NOT-LOWER-FUNCTION
EBD5 BBE5E8        R     2155            MOV     BX,OFFSET K10           ; LC TABLE
                         2156
                         2157    ;------ TRANSLATE THE CHARACTER
                         2158
EBD8                     2159    K56:                                    ; TRANSLATE-CHAR
EBD8 FEC8                2160            DEC     AL                      ; CONVERT ORIGIN
EBDA 2ED7                2161            XLAT    CS:K11                  ; CONVERT THE SCAN CODE TO ASCII
                         2162
                         2163    ;------ PUT CHARACTER INTO BUFFER
                         2164
EBDC                     2165    K57:                                    ; BUFFER-FILL
EBDC 3CFF                2166            CMP     AL,-1                   ; IS THIS AN IGNORE CHAR
EBDE 741F                2167            JE      K59                     ; YES, DO NOTHING WITH IT
EBE0 80FCFF              2168            CMP     AH,-1                   ; LOOK FOR -1 PSEUDO SCAN
EBE3 741A                2169            JE      K59                     ; NEAR_INTERRUPT_RETURN
                         2170
                         2171    ;------ HANDLE THE CAPS LOCK PROBLEM
                         2172
EBE5                     2173    K58:                                    ; BUFFER-FILL-NOTEST
EBE5 F606170040    R     2174            TEST    KB_FLAG,CAPS_STATE      ; ARE WE IN CAPS LOCK STATE
EBEA 7420                2175            JZ      K61                     ; SKIP IF NOT
                         2176
                         2177    ;------ IN CAPS LOCK STATE
                         2178
EBEC F606170003    R     2179            TEST    KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT  ; TEST FOR SHIFT STATE
```

```
LOC OBJ                 LINE    SOURCE

EBF1 740F               2180            JZ      K60                     ; IF NOT SHIFT, CONVERT LOWER TO UPPER
                        2181
                        2182    ;------ CONVERT ANY UPPER CASE TO LOWER CASE
                        2183
EBF3 3C41               2184            CMP     AL,'A'                  ; FIND OUT IF ALPHABETIC
EBF5 7215               2185            JB      K61                     ; NOT_CAPS_STATE
EBF7 3C5A               2186            CMP     AL,'Z'
EBF9 7711               2187            JA      K61                     ; NOT_CAPS_STATE
EBFB 0420               2188            ADD     AL,'a'-'A'              ; CONVERT TO LOWER CASE
EBFD EB0D               2189            JMP     SHORT K61               ; NOT_CAPS_STATE
                        2190
EBFF                    2191    K59:                                    ; NEAR-INTERRUPT-RETURN
EBFF E95CFE             2192            JMP     K26                     ; INTERRUPT_RETURN
                        2193
                        2194    ;------ CONVERT ANY LOWER CASE TO UPPER CASE
                        2195
EC02                    2196    K60:                                    ; LOWER-TO-UPPER
EC02 3C61               2197            CMP     AL,'a'                  ; FIND OUT IF ALPHABETIC
EC04 7206               2198            JB      K61                     ; NOT_CAPS_STATE
EC06 3C7A               2199            CMP     AL,'z'
EC08 7702               2200            JA      K61                     ; NOT_CAPS_STATE
EC0A 2C20               2201            SUB     AL,'a'-'A'              ; CONVERT TO UPPER CASE
                        2202
EC0C                    2203    K61:                                    ; NOT-CAPS-STATE
EC0C 8B1E1C00       R   2204            MOV     BX,BUFFER_TAIL          ; GET THE END POINTER TO THE BUFFER
EC10 8BF3               2205            MOV     SI,BX                   ; SAVE THE VALUE
EC12 E860FC             2206            CALL    K4                      ; ADVANCE THE TAIL
EC15 3B1E1A00       R   2207            CMP     BX,BUFFER_HEAD          ; HAS THE BUFFER WRAPPED AROUND
EC19 7409               2208            JE      K62                     ; BUFFER_FULL_BEEP
EC1B 8904               2209            MOV     [SI],AX                 ; STORE THE VALUE
EC1D 891E1C00       R   2210            MOV     BUFFER_TAIL,BX          ; MOVE THE POINTER UP
EC21 E93AFE             2211            JMP     K26                     ; INTERRUPT_RETURN
                        2212
                        2213    ;------ BUFFER IS FULL, SOUND THE BEEPER
                        2214
EC24                    2215    K62:                                    ; BUFFER-FULL-BEEP
EC24 E80D00             2216            CALL    ERROR_BEEP
EC27 E934FE             2217            JMP     K26                     ; INTERRUPT_RETURN
                        2218
                        2219    ;------ TRANSLATE SCAN FOR PSEUDO SCAN CODES
                        2220
EC2A                    2221    K63:                                    ; TRANSLATE-SCAN
EC2A 2C3B               2222            SUB     AL,59                   ; CONVERT ORIGIN TO FUNCTION KEYS
EC2C                    2223    K64:                                    ; TRANSLATE-SCAN-ORGD
EC2C 2ED7               2224            XLAT    CS:K9                   ; CTL TABLE SCAN
EC2E 8AE0               2225            MOV     AH,AL                   ; PUT VALUE INTO AH
EC30 B000               2226            MOV     AL,0                    ; ZERO ASCII CODE
EC32 EBA8               2227            JMP     K57                     ; PUT IT INTO THE BUFFER
                        2228
                        2229    KB_INT  ENDP
EC34                    2230    ERROR_BEEP      PROC    NEAR
EC34 50                 2231            PUSH    AX              ; SAVE REGISTERS
EC35 53                 2232            PUSH    BX
EC36 51                 2233            PUSH    CX
EC37 BBC000             2234            MOV     BX,0C0H         ; NUMBER OF CYCLES FOR 1/8 SECOND TONE
EC3A E461               2235            IN      AL,KB_CTL       ; GET CONTROL INFORMATION
EC3C 50                 2236            PUSH    AX              ; SAVE
EC3D                    2237    K65:                            ; BEEP-CYCLE
EC3D 24FC               2238            AND     AL,0FCH         ; TURN OFF TIMER GATE AND SPEAKER DATA
EC3F E661               2239            OUT     KB_CTL,AL       ; OUTPUT TO CONTROL
EC41 B94800             2240            MOV     CX,48H          ; HALF CYCLE TIME FOR TONE
EC44 E2FE               2241    K66:    LOOP    K66             ; SPEAKER OFF
EC46 0C02               2242            OR      AL,2            ; TURN ON SPEAKER BIT
EC48 E661               2243            OUT     KB_CTL,AL       ; OUTPUT TO CONTROL
EC4A B94800             2244            MOV     CX,48H          ; SET UP COUNT
EC4D E2FE               2245    K67:    LOOP    K67             ; ANOTHER HALF CYCLE
EC4F 4B                 2246            DEC     BX              ; TOTAL TIME COUNT
EC50 75EB               2247            JNZ     K65             ; DO ANOTHER CYCLE
EC52 58                 2248            POP     AX              ; RECOVER CONTROL
EC53 E661               2249            OUT     KB_CTL,AL       ; OUTPUT THE CONTROL
EC55 59                 2250            POP     CX              ; RECOVER REGISTERS
EC56 5B                 2251            POP     BX
EC57 58                 2252            POP     AX
EC58 C3                 2253            RET
                        2254    ERROR_BEEP      ENDP
```

```
LOC OBJ                  LINE    SOURCE

                         2255    ;-- INT 13 ---------------------------------
                         2256    ;DISKETTE I/O
                         2257    ;       THIS INTERFACE PROVIDES ACCESS TO THE 5 1/4" DISKETTE DRIVES
                         2258    ;INPUT
                         2259    ;       (AH)=0  RESET DISKETTE SYSTEM
                         2260    ;               HARD RESET TO NEC, PREPARE COMMAND, RECAL REQD ON ALL DRIVES
                         2261    ;       (AH)=1  READ THE STATUS OF THE SYSTEM INTO (AL)
                         2262    ;               DISKETTE_STATUS FROM LAST OP'N IS USED
                         2263    ;       REGISTERS FOR READ/WRITE/VERIFY/FORMAT
                         2264    ;       (DL) - DRIVE NUMBER (0-3 ALLOWED, VALUE CHECKED)
                         2265    ;       (DH) - HEAD NUMBER (0-1 ALLOWED, NOT VALUE CHECKED)
                         2266    ;       (CH) - TRACK NUMBER (0-39, NOT VALUE CHECKED)
                         2267    ;       (CL) - SECTOR NUMBER (1-8, NOT VALUE CHECKED)
                         2268    ;       (AL) - NUMBER OF SECTORS ( MAX = 8, NOT VALUE CHECKED)
                         2269    ;
                         2270    ;       (ES:BX) - ADDRESS OF BUFFER ( NOT REQUIRED FOR VERIFY)
                         2271    ;
                         2272    ;       (AH)=2  READ THE DESIRED SECTORS INTO MEMORY
                         2273    ;       (AH)=3  WRITE THE DESIRED SECTORS FROM MEMORY
                         2274    ;       (AH)=4  VERIFY THE DESIRED SECTORS
                         2275    ;       (AH)=5  FORMAT THE DESIRED TRACK
                         2276    ;               FOR THE FORMAT OPERATION, THE BUFFER POINTER (ES,BX) MUST
                         2277    ;               POINT TO THE COLLECTION OF DESIRED ADDRESS FIELDS FOR THE
                         2278    ;               TRACK.  EACH FIELD IS COMPOSED OF 4 BYTES, (C,H,R,N), WHERE
                         2279    ;               C = TRACK NUMBER, H=HEAD NUMBER, R = SECTOR NUMBER, N= NUMBER
                         2280    ;               OF BYTES PER SECTOR (00=128, 01=256, 02=512, 03=1024,)
                         2281    ;               THERE MUST BE ONE ENTRY FOR EVERY SECTOR ON THE TRACK.
                         2282    ;               THIS INFORMATION IS USED TO FIND THE REQUESTED SECTOR DURING
                         2283    ;               READ/WRITE ACCESS.
                         2284    ; DATA VARIABLE -- DISK_POINTER
                         2285    ;       DOUBLE WORD POINTER TO THE CURRENT SET OF DISKETTE PARAMETERS
                         2286    ; OUTPUT
                         2287    ;       AH = STATUS OF OPERATION
                         2288    ;               STATUS BITS ARE DEFINED IN THE EQUATES FOR DISKETTE_STATUS
                         2289    ;               VARIABLE IN THE DATA SEGMENT OF THIS MODULE
                         2290    ;       CY = 0  SUCCESSFUL OPERATION (AH=0 ON RETURN)
                         2291    ;       CY = 1  FAILED OPERATION (AH HAS ERROR REASON)
                         2292    ;       FOR READ/WRITE/VERIFY
                         2293    ;               DS,BX,DX,CH,CL PRESERVED
                         2294    ;               AL = NUMBER OF SECTORS ACTUALLY READ
                         2295    ;               ***** AL MAY NOT BE CORRECT IF TIME OUT ERROR OCCURS
                         2296    ;       NOTE: IF AN ERROR IS REPORTED BY THE DISKETTE CODE, THE APPROPRIATE
                         2297    ;               ACTION IS TO RESET THE DISKETTE, THEN RETRY THE OPERATION.
                         2298    ;               ON READ ACCESSES, NO MOTOR START DELAY IS TAKEN, SO THAT
                         2299    ;               THREE RETRIES ARE REQUIRED ON READS TO ENSURE THAT THE
                         2300    ;               PROBLEM IS NOT DUE TO MOTOR START-UP.
                         2301    ;----------------------------------------------
                         2302            ASSUME  CS:CODE,DS:DATA,ES:DATA
EC59                     2303    DISKETTE_IO     PROC    FAR
EC59 FB                  2304            STI                     ; INTERRUPTS BACK ON
EC5A 53                  2305            PUSH    BX              ; SAVE ADDRESS
EC5B 51                  2306            PUSH    CX
EC5C 1E                  2307            PUSH    DS              ; SAVE SEGMENT REGISTER VALUE
EC5D 56                  2308            PUSH    SI              ; SAVE ALL REGISTERS DURING OPERATION
EC5E 57                  2309            PUSH    DI
EC5F 55                  2310            PUSH    BP
EC60 52                  2311            PUSH    DX
EC61 8BEC                2312            MOV     BP,SP           ; SET UP POINTER TO HEAD PARM
EC63 BE4000        R     2313            MOV     SI,DATA
EC66 8ED6                2314            MOV     DS,SI           ; SET DATA REGION
EC68 E81C00              2315            CALL    J1              ; CALL THE REST TO ENSURE DS RESTORED
EC6B BB0400              2316            MOV     BX,4            ; GET THE MOTOR WAIT PARAMETER
EC6E E8FF01              2317            CALL    GET_PARM
EC71 88264000      R     2318            MOV     MOTOR_COUNT,AH  ; SET THE TIMER COUNT FOR THE MOTOR
EC75 8A264100      R     2319            MOV     AH,DISKETTE_STATUS      ; GET STATUS OF OPERATION
EC79 80FC01              2320            CMP     AH,1            ; SET THE CARRY FLAG TO INDICATE
EC7C F5                  2321            CMC                     ;  SUCCESS OR FAILURE
EC7D 5A                  2322            POP     DX              ; RESTORE ALL REGISTERS
EC7E 5D                  2323            POP     BP
EC7F 5F                  2324            POP     DI
EC80 5E                  2325            POP     SI
EC81 1F                  2326            POP     DS
EC82 59                  2327            POP     CX
EC83 5B                  2328            POP     BX              ; RECOVER ADDRESS
EC84 CA0200              2329            RET     2               ; THROW AWAY SAVED FLAGS
                         2330    DISKETTE_IO     ENDP
EC87                     2331    J1      PROC    NEAR
```

```
LOC OBJ                 LINE    SOURCE

EC87 8AF0               2332            MOV     DH,AL           ; SAVE # SECTORS IN DH
EC89 80263F007F     R   2333            AND     MOTOR_STATUS,07FH       ; INDICATE A READ OPERATION
EC8E 0AE4               2334            OR      AH,AH           ; AH=0
EC90 7427               2335            JZ      DISK_RESET
EC92 FECC               2336            DEC     AH              ; AH=1
EC94 7474               2337            JZ      DISK_STATUS
EC96 C606410000     R   2338            MOV     DISKETTE_STATUS,0       ; RESET THE STATUS INDICATOR
EC9B 80FA04             2339            CMP     DL,4            ; TEST FOR DRIVE IN 0-3 RANGE
EC9E 7313               2340            JAE     J3              ; ERROR IF ABOVE
ECA0 FECC               2341            DEC     AH              ; AH=2
ECA2 746A               2342            JZ      DISK_READ
ECA4 FECC               2343            DEC     AH              ; AH=3
ECA6 7503               2344            JNZ     J2              ; TEST_DISK_VERF

ECA8 E99600             2345            JMP     DISK_WRITE
ECAB                    2346    J2:                             ; TEST_DISK_VERF

ECAB FECC               2347            DEC     AH              ; AH=4
ECAD 7468               2348            JZ      DISK_VERF
ECAF FECC               2349            DEC     AH              ; AH=5
ECB1 7468               2350            JZ      DISK_FORMAT
ECB3                    2351    J3:                             ; BAD_COMMAND
ECB3 C606410001     R   2352            MOV     DISKETTE_STATUS,BAD_CMD ; ERROR CODE, NO SECTORS TRANSFERRED
ECB8 C3                 2353            RET                     ; UNDEFINED OPERATION
                        2354    J1      ENDP
                        2355
                        2356    ;------ RESET THE DISKETTE SYSTEM
                        2357
ECB9                    2358    DISK_RESET      PROC    NEAR
ECB9 BAF203             2359            MOV     DX,03F2H        ; ADAPTER CONTROL PORT
ECBC FA                 2360            CLI                     ; NO INTERRUPTS
ECBD A03F00         R   2361            MOV     AL,MOTOR_STATUS ; WHICH MOTOR IS ON
ECC0 B104               2362            MOV     CL,4            ; SHIFT COUNT
ECC2 D2E0               2363            SAL     AL,CL           ; MOVE MOTOR VALUE TO HIGH NYBBLE
ECC4 A820               2364            TEST    AL, 20H         ; SELECT CORRESPONDING DRIVE
ECC6 750C               2365            JNZ     J5              ; JUMP IF MOTOR ONE IS ON
ECC8 A840               2366            TEST    AL, 40H
ECCA 7506               2367            JNZ     J4              ; JUMP IF MOTOR TWO IS ON
ECCC A880               2368            TEST    AL, 80H
ECCE 7406               2369            JZ      J6              ; JUMP IF MOTOR ZERO IS ON
ECD0 FEC0               2370            INC     AL
ECD2 FEC0               2371    J4:     INC     AL
ECD4 FEC0               2372    J5:     INC     AL
ECD6 0C08               2373    J6:     OR      AL,8            ; TURN ON INTERRUPT ENABLE
ECD8 EE                 2374            OUT     DX,AL           ; RESET THE ADAPTER
ECD9 C6063E0000     R   2375            MOV     SEEK_STATUS,0   ; SET RECAL REQUIRED ON ALL DRIVES
ECDE C606410000     R   2376            MOV     DISKETTE_STATUS,0       ; SET OK STATUS FOR DISKETTE
ECE3 0C04               2377            OR      AL,4            ; TURN OFF RESET
ECE5 EE                 2378            OUT     DX,AL           ; TURN OFF THE RESET
ECE6 FB                 2379            STI                     ; REENABLE THE INTERRUPTS
ECE7 E82802             2380            CALL    CHK_STAT_2      ; DO SENSE INTERRUPT STATUS FOLLOWING RESET
ECEA A04200         R   2381            MOV     AL,NEC_STATUS   ; IGNORE ERROR RETURN AND DO OWN TEST
ECED 3CC0               2382            CMP     AL,0C0H         ; TEST FOR DRIVE READY TRANSITION
ECEF 7407               2383            JZ      J7              ; EVERYTHING OK
ECF1 800E410020     R   2384            OR      DISKETTE_STATUS,BAD_NEC ; SET ERROR CODE
ECF6 EB11               2385            JMP     SHORT J8                ; RESET_RET
                        2386
                        2387    ;------ SEND SPECIFY COMMAND TO NEC
                        2388
ECF8                    2389    J7:                             ; DRIVE_READY
ECF8 B403               2390            MOV     AH,03H          ; SPECIFY COMMAND
ECFA E84701             2391            CALL    NEC_OUTPUT      ; OUTPUT THE COMMAND
ECFD BB0100             2392            MOV     BX,1            ; FIRST BYTE PARM IN BLOCK
ED00 E86D01             2393            CALL    GET_PARM        ;  TO THE NEC CONTROLLER
ED03 BB0300             2394            MOV     BX,3            ; SECOND BYTE PARM IN BLOCK
ED06 E86701             2395            CALL    GET_PARM        ;  TO THE NEC CONTROLLER
ED09                    2396    J8:                             ; RESET_RET
ED09 C3                 2397            RET                     ; RETURN TO CALLER
                        2398    DISK_RESET      ENDP
                        2399
                        2400    ;------ DISKETTE STATUS ROUTINE
                        2401
ED0A                    2402    DISK_STATUS     PROC    NEAR
ED0A A04100 R       R   2403            MOV     AL,DISKETTE_STATUS
ED0D C3                 2404            RET
                        2405    DISK_STATUS     ENDP
                        2406
```

```
LOC OBJ                 LINE    SOURCE


                        2407    ;------ DISKETTE READ
                        2408
ED0E                    2409    DISK_READ       PROC    NEAR
ED0E B046               2410            MOV     AL,046H         ; READ COMMAND FOR DMA
ED10                    2411    J9:                             ; DISK_READ_CONT
ED10 E8B901             2412            CALL    DMA_SETUP       ; SET UP THE DMA
ED13 B466               2413            MOV     AH,066H         ; SET UP READ COMMAND FOR NEC CONTROLLER
ED15 EB36               2414            JMP     SHORT RW_OPN    ; GO DO THE OPERATION
                        2415    DISK_READ       ENDP
                        2416
                        2417    ;------ DISKETTE VERIFY
                        2418
ED17                    2419    DISK_VERF       PROC    NEAR
ED17 B042               2420            MOV     AL,042H                 ; VERIFY COMMAND FOR DMA
ED19 EBF5               2421            JMP     J9                      ; DO AS IF DISK READ
                        2422    DISK_VERF       ENDP
                        2423
                        2424    ;------ DISKETTE FORMAT
                        2425
ED1B                    2426    DISK_FORMAT     PROC    NEAR
ED1B 800E3F0080     R   2427            OR      MOTOR_STATUS,80H        ; INDICATE WRITE OPERATION
ED20 B04A               2428            MOV     AL,04AH                 ; WILL WRITE TO THE DISKETTE
ED22 E8A701             2429            CALL    DMA_SETUP               ; SET UP THE DMA
ED25 B44D               2430            MOV     AH,04DH                 ; ESTABLISH THE FORMAT COMMAND
ED27 EB24               2431            JMP     SHORT RW_OPN            ; DO THE OPERATION
ED29                    2432    J10:                                    ; CONTINUATION OF RW_OPN FOR FMT
ED29 BB0700             2433            MOV     BX,7                    ; GET THE
ED2C E84101             2434            CALL    GET_PARM                ;  BYTES/SECTOR VALUE TO NEC
ED2F BB0900             2435            MOV     BX,9                    ; GET THE
ED32 E83B01             2436            CALL    GET_PARM                ;  SECTORS/TRACK VALUE TO NEC
ED35 BB0F00             2437            MOV     BX,15                   ; GET THE
ED38 E83501             2438            CALL    GET_PARM                ;  GAP LENGTH VALUE TO NEC
ED3B BB1100             2439            MOV     BX,17                   ; GET THE FILLER BYTE
ED3E E9AB00             2440            JMP     J16                     ;  TO THE CONTROLLER
                        2441    DISK_FORMAT     ENDP
                        2442
                        2443    ;------ DISKETTE WRITE ROUTINE
                        2444
ED41                    2445    DISK_WRITE      PROC    NEAR
ED41 800E3F0080     R   2446            OR      MOTOR_STATUS,80H        ; INDICATE WRITE OPERATION
ED46 B04A               2447            MOV     AL,04AH                 ; DMA WRITE COMMAND
ED48 E88101             2448            CALL    DMA_SETUP
ED4B B445               2449            MOV     AH,045H                 ; NEC COMMAND TO WRITE TO DISKETTE
                        2450    DISK_WRITE      ENDP
                        2451    ;----- ALLOW WRITE ROUTINE TO FALL INTO RW_OPN
                        2452    ;--------------------------------------
                        2453    ; RW_OPN
                        2454    ;       THIS ROUTINE PERFORMS THE READ/WRITE/VERIFY OPERATION
                        2455    ;--------------------------------------
ED4D                    2456    RW_OPN  PROC    NEAR
ED4D 7308               2457            JNC     J11                     ; TEST FOR DMA ERROR
ED4F C606410009     R   2458            MOV     DISKETTE_STATUS,DMA_BOUNDARY    ; SET ERROR
ED54 B000               2459            MOV     AL,0                    ; NO SECTORS TRANSFERRED
ED56 C3                 2460            RET                             ; RETURN TO MAIN ROUTINE
ED57                    2461    J11:                                    ; DO_RW_OPN
ED57 50                 2462            PUSH    AX              ; SAVE THE COMMAND
                        2463
                        2464    ;------ TURN ON THE MOTOR AND SELECT THE DRIVE
                        2465
ED58 51                 2466            PUSH    CX              ; SAVE THE T/S PARMS
ED59 8ACA               2467            MOV     CL,DL           ; GET DRIVE NUMBER AS SHIFT COUNT
ED5B B001               2468            MOV     AL,1            ; MASK FOR DETERMINING MOTOR BIT
ED5D D2E0               2469            SAL     AL,CL           ; SHIFT THE MASK BIT
ED5F FA                 2470            CLI                     ; NO INTERRUPTS WHILE DETERMINING MOTOR STATUS
ED60 C6064000FF     R   2471            MOV     MOTOR_COUNT,0FFH        ; SET LARGE COUNT DURING OPERATION
ED65 84063F00       R   2472            TEST    AL,MOTOR_STATUS ; TEST THAT MOTOR FOR OPERATING
ED69 7531               2473            JNZ     J14             ; IF RUNNING, SKIP THE WAIT
ED6B 80263F00F0     R   2474            AND     MOTOR_STATUS,0F0H       ; TURN OFF ALL MOTOR BITS
ED70 08063F00       R   2475            OR      MOTOR_STATUS,AL         ; TURN ON THE CURRENT MOTOR
ED74 FB                 2476            STI                     ; INTERRUPTS BACK ON
ED75 B010               2477            MOV     AL,10H          ; MASK BIT
ED77 D2E0               2478            SAL     AL,CL           ; DEVELOP BIT MASK FOR MOTOR ENABLE
ED79 0AC2               2479            OR      AL,DL           ; GET DRIVE SELECT BITS IN
ED7B 0C0C               2480            OR      AL,0CH          ; NO RESET, ENABLE DMA/INT
ED7D 52                 2481            PUSH    DX              ; SAVE REG
ED7E BAF203             2482            MOV     DX,03F2H        ; CONTROL PORT ADDRESS
ED81 EE                 2483            OUT     DX,AL
```

```
LOC OBJ                 LINE    SOURCE

ED82 5A                 2484            POP     DX              ; RECOVER REGISTERS
                        2485
                        2486    ;------ WAIT FOR MOTOR IF WRITE OPERATION
                        2487
ED83 F6063F0080     R   2488            TEST    MOTOR_STATUS,80H        ; IS THIS A WRITE
ED88 7412               2489            JZ      J14                     ; NO, CONTINUE WITHOUT WAIT
ED8A BB1400             2490            MOV     BX,20                   ; GET THE MOTOR WAIT
ED8D E8E000             2491            CALL    GET_PARM                ;  PARAMETER
ED90 0AE4               2492            OR      AH,AH                   ; TEST FOR NO WAIT
ED92                    2493    J12:                                    ; TEST_WAIT_TIME
ED92 7408               2494            JZ      J14                     ; EXIT WITH TIME EXPIRED
ED94 2BC9               2495            SUB     CX,CX                   ; SET UP 1/8 SECOND LOOP TIME
ED96 E2FE               2496    J13:    LOOP    J13                     ; WAIT FOR THE REQUIRED TIME
ED98 FECC               2497            DEC     AH                      ; DECREMENT TIME VALUE
ED9A EBF6               2498            JMP     J12                     ; ARE WE DONE YET
                        2499
ED9C                    2500    J14:                                    ; MOTOR_RUNNING
ED9C FB                 2501            STI                     ; INTERRUPTS BACK ON FOR BYPASS WAIT
ED9D 59                 2502            POP     CX
                        2503
                        2504    ;------ DO THE SEEK OPERATION
                        2505
ED9E E8E000             2506            CALL    SEEK            ; MOVE TO CORRECT TRACK
EDA1 58                 2507            POP     AX              ; RECOVER COMMAND
EDA2 8AFC               2508            MOV     BH,AH           ; SAVE COMMAND IN BH
EDA4 B600               2509            MOV     DH,0            ; SET NO SECTORS READ IN CASE OF ERROR
EDA6 724B               2510            JC      J17             ; IF ERROR, THEN EXIT AFTER MOTOR OFF
EDA8 BEF3ED90       R   2511            MOV     SI,OFFSET J17   ; DUMMY RETURN ON STACK FOR NEC_OUTPUT
EDAC 56                 2512            PUSH    SI              ;  SO THAT IT WILL RETURN TO MOTOR OFF LOCATION
                        2513
                        2514    ;------ SEND OUT THE PARAMETERS TO THE CONTROLLER
                        2515
EDAD E89400             2516            CALL    NEC_OUTPUT      ; OUTPUT THE OPERATION COMMAND
EDB0 8A6601             2517            MOV     AH,[BP+1]       ; GET THE CURRENT HEAD NUMBER
EDB3 D0E4               2518            SAL     AH,1            ; MOVE IT TO BIT 2
EDB5 D0E4               2519            SAL     AH,1
EDB7 80E404             2520            AND     AH,4            ; ISOLATE THAT BIT
EDBA 0AE2               2521            OR      AH,DL           ; OR IN THE DRIVE NUMBER
EDBC E88500             2522            CALL    NEC_OUTPUT
                        2523
                        2524    ;------ TEST FOR FORMAT COMMAND
                        2525
EDBF 80FF4D             2526            CMP     BH,04DH         ; IS THIS A FORMAT OPERATION
EDC2 7503               2527            JNE     J15             ; NO. CONTINUE WITH R/W/V
EDC4 E962FF             2528            JMP     J10             ; IF SO, HANDLE SPECIAL
                        2529
EDC7 8AE5               2530    J15:    MOV     AH,CH           ; CYLINDER NUMBER
EDC9 E87800             2531            CALL    NEC_OUTPUT
EDCC 8A6601             2532            MOV     AH,[BP+1]       ; HEAD NUMBER FROM STACK
EDCF E87200             2533            CALL    NEC_OUTPUT
EDD2 8AE1               2534            MOV     AH,CL           ; SECTOR NUMBER
EDD4 E86D00             2535            CALL    NEC_OUTPUT
EDD7 BB0700             2536            MOV     BX,7            ; BYTES/SECTOR PARM FROM BLOCK
EDDA E89300             2537            CALL    GET_PARM        ;  TO THE NEC
EDDD BB0900             2538            MOV     BX,9            ; EOT PARM FROM BLOCK
EDE0 E88D00             2539            CALL    GET_PARM        ;  TO THE NEC
EDE3 BB0B00             2540            MOV     BX,11           ; GAP LENGTH PARM FROM BLOCK
EDE6 E88700             2541            CALL    GET_PARM        ;  TO THE NEC
EDE9 BB0D00             2542            MOV     BX,13           ; DTL PARM FROM BLOCK
EDEC                    2543    J16:                            ; RW_OPN_FINISH
EDEC E88100             2544            CALL    GET_PARM        ;  TO THE NEC
EDEF 5E                 2545            POP     SI              ; CAN NOW DISCARD THAT DUMMY RETURN ADDRESS
                        2546
                        2547    ;------ LET THE OPERATION HAPPEN
                        2548
EDF0 E84001             2549            CALL    WAIT_INT        ; WAIT FOR THE INTERRUPT
EDF3                    2550    J17:                            ; MOTOR_OFF
EDF3 7245               2551            JC      J21             ; LOOK FOR ERROR
EDF5 E87301             2552            CALL    RESULTS         ; GET THE NEC STATUS
EDF8 723F               2553            JC      J20             ; LOOK FOR ERROR
                        2554
                        2555    ;------ CHECK THE RESULTS RETURNED BY THE CONTROLLER
                        2556
EDFA FC                 2557            CLD                     ; SET THE CORRECT DIRECTION
EDFB BE4200         R   2558            MOV     SI,OFFSET NEC_STATUS    ; POINT TO STATUS FIELD
EDFE AC                 2559            LODS    NEC_STATUS              ; GET ST0
EDFF 24C0               2560            AND     AL,0C0H                 ; TEST FOR NORMAL TERMINATION
```

```
LOC OBJ                  LINE    SOURCE

EE01 743B                2561            JZ      J22                     ; OPN_OK
EE03 3C40                2562            CMP     AL,040H                 ; TEST FOR ABNORMAL TERMINATION
EE05 7529                2563            JNZ     J18                     ; NOT ABNORMAL, BAD NEC
                         2564
                         2565    ;------ ABNORMAL TERMINATION, FIND OUT WHY
                         2566
EE07 AC                  2567            LODS    NEC_STATUS              ; GET ST1
EE08 D0E0                2568            SAL     AL,1                    ; TEST FOR EOT FOUND
EE0A B404                2569            MOV     AH,RECORD_NOT_FND
EE0C 7224                2570            JC      J19                     ; RW_FAIL
EE0E D0E0                2571            SAL     AL,1
EE10 D0E0                2572            SAL     AL,1                    ; TEST FOR CRC ERROR
EE12 B410                2573            MOV     AH,BAD_CRC
EE14 721C                2574            JC      J19                     ; RW_FAIL
EE16 D0E0                2575            SAL     AL,1                    ; TEST FOR DMA OVERRUN
EE18 B408                2576            MOV     AH,BAD_DMA
EE1A 7216                2577            JC      J19                     ; RW_FAIL
EE1C D0E0                2578            SAL     AL,1
EE1E D0E0                2579            SAL     AL,1                    ; TEST FOR RECORD NOT FOUND
EE20 B404                2580            MOV     AH,RECORD_NOT_FND
EE22 720E                2581            JC      J19                     ; RW_FAIL
EE24 D0E0                2582            SAL     AL,1
EE26 B403                2583            MOV     AH,WRITE_PROTECT        ; TEST FOR WRITE_PROTECT
EE28 7208                2584            JC      J19                     ; RW_FAIL
EE2A D0E0                2585            SAL     AL,1                    ; TEST MISSING ADDRESS MARK
EE2C B402                2586            MOV     AH,BAD_ADDR_MARK
EE2E 7202                2587            JC      J19                     ; RW_FAIL
                         2588
                         2589    ;------ NEC MUST HAVE FAILED
                         2590
EE30                     2591    J18:                                    ; RW-NEC-FAIL
EE30 B420                2592            MOV     AH,BAD_NEC
EE32                     2593    J19:                                    ; RW-FAIL
EE32 08264100      R     2594            OR      DISKETTE_STATUS,AH
EE36 E87701              2595            CALL    NUM_TRANS               ; HOW MANY WERE REALLY TRANSFERRED
EE39                     2596    J20:                                    ; RW_ERR
EE39 C3                  2597            RET                     ; RETURN TO CALLER
                         2598
EE3A                     2599    J21:                            ; RW_ERR_RES
EE3A E82E01              2600            CALL    RESULTS         ; FLUSH THE RESULTS BUFFER
EE3D C3                  2601            RET
                         2602
                         2603    ;------ OPERATION WAS SUCCESSFUL
                         2604
EE3E                     2605    J22:                                    ; OPN_OK
EE3E E86F01              2606            CALL    NUM_TRANS               ; HOW MANY GOT MOVED
EE41 32E4                2607            XOR     AH,AH                   ; NO ERRORS
EE43 C3                  2608            RET
                         2609    RW_OPN  ENDP
                         2610    ;--------------------------------------------
                         2611    ; NEC_OUTPUT
                         2612    ;       THIS ROUTINE SENDS A BYTE TO THE NEC CONTROLLER
                         2613    ;       AFTER TESTING FOR CORRECT DIRECTION AND CONTROLLER READY
                         2614    ;       THIS ROUTINE WILL TIME OUT IF THE BYTE IS NOT ACCEPTED
                         2615    ;       WITHIN A REASONABLE AMOUNT OF TIME, SETTING THE DISKETTE STATUS
                         2616    ;       ON COMPLETION
                         2617    ; INPUT
                         2618    ;       (AH)    BYTE TO BE OUTPUT
                         2619    ; OUTPUT
                         2620    ;       CY = 0  SUCCESS
                         2621    ;       CY = 1  FAILURE -- DISKETTE STATUS UPDATED
                         2622    ;               IF A FAILURE HAS OCCURRED, THE RETURN IS MADE ONE LEVEL
                         2623    ;               HIGHER THAN THE CALLER OF NEC_OUTPUT
                         2624    ;               THIS REMOVES THE REQUIREMENT OF TESTING AFTER EVERY CALL
                         2625    ;               OF NEC_OUTPUT
                         2626    ;       (AL) DESTROYED
                         2627    ;--------------------------------------------
EE44                     2628    NEC_OUTPUT      PROC    NEAR
EE44 52                  2629            PUSH    DX              ; SAVE REGISTERS
EE45 51                  2630            PUSH    CX
EE46 BAF403              2631            MOV     DX,03F4H        ; STATUS PORT
EE49 33C9                2632            XOR     CX,CX           ; COUNT FOR TIME OUT
EE4B                     2633    J23:
EE4B EC                  2634            IN      AL,DX           ; GET STATUS
EE4C A840                2635            TEST    AL,040H         ; TEST DIRECTION BIT
EE4E 740C                2636            JZ      J25             ; DIRECTION OK
EE50 E2F9                2637            LOOP    J23
EE52                     2638    J24:                            ; TIME_ERROR
```

```
LOC OBJ                 LINE    SOURCE

EE52 800E410080   R     2639            OR      DISKETTE_STATUS,TIME_OUT
EE57 59                 2640            POP     CX
EE58 5A                 2641            POP     DX              ; SET ERROR CODE AND RESTORE REGS
EE59 58                 2642            POP     AX              ; DISCARD THE RETURN ADDRESS
EE5A F9                 2643            STC                     ; INDICATE ERROR TO CALLER
EE5B C3                 2644            RET
                        2645
EE5C                    2646    J25:
EE5C 33C9               2647            XOR     CX,CX           ; RESET THE COUNT
EE5E                    2648    J26:
EE5E EC                 2649            IN      AL,DX           ; GET THE STATUS
EE5F A880               2650            TEST    AL,080H         ; IS IT READY
EE61 7504               2651            JNZ     J27             ; YES, GO OUTPUT
EE63 E2F9               2652            LOOP    J26             ; COUNT DOWN AND TRY AGAIN
EE65 EBEB               2653            JMP     J24             ; ERROR CONDITION
EE67                    2654    J27:                            ; OUTPUT
EE67 8AC4               2655            MOV     AL,AH           ; GET BYTE TO OUTPUT
EE69 BAF503             2656            MOV     DX,03F5H        ; DATA PORT
EE6C EE                 2657            OUT     DX,AL           ; OUTPUT THE BYTE
EE6D 59                 2658            POP     CX              ; RECOVER REGISTERS
EE6E 5A                 2659            POP     DX
EE6F C3                 2660            RET                     ; CY = 0 FROM TEST INSTRUCTION
                        2661    NEC_OUTPUT      ENDP
                        2662    ;--------------------------------------------
                        2663    ; GET_PARM
                        2664    ;  THIS ROUTINE FETCHES THE INDEXED POINTER FROM
                        2665    ;  THE DISK_BASE BLOCK POINTED AT BY THE DATA
                        2666    ;  VARIABLE DISK_POINTER
                        2667    ; A BYTE FROM THAT TABLE IS THEN MOVED INTO AH,
                        2668    ;  THE INDEX OF THAT BYTE BEING THE PARM IN BX
                        2669    ; ENTRY --
                        2670    ;   BX = INDEX OF BYTE TO BE FETCHED * 2
                        2671    ;        IF THE LOW BIT OF BX IS ON, THE BYTE IS IMMEDIATELY
                        2672    ;        OUTPUT TO THE NEC CONTROLLER
                        2673    ; EXIT --
                        2674    ;   AH = THAT BYTE FROM BLOCK
                        2675    ;--------------------------------------------
EE70                    2676    GET_PARM        PROC    NEAR
EE70 1E                 2677            PUSH    DS              ; SAVE SEGMENT
EE71 2BC0               2678            SUB     AX,AX           ; ZERO TO AX
EE73 8ED8               2679            MOV     DS,AX
                        2680            ASSUME  DS:ABS0
EE75 C5367800           2681            LDS     SI,DISK_POINTER ; POINT TO BLOCK
EE79 D1EB               2682            SHR     BX,1            ; DIVIDE BX BY 2, AND SET FLAG FOR EXIT
EE7B 8A20               2683            MOV     AH,[SI+BX]      ; GET THE WORD
EE7D 1F                 2684            POP     DS              ; RESTORE SEGMENT
                        2685            ASSUME  DS:DATA
EE7E 72C4               2686            JC      NEC_OUTPUT      ; IF FLAG SET, OUTPUT TO CONTROLLER
EE80 C3                 2687            RET                     ; RETURN TO CALLER
                        2688    GET_PARM        ENDP
                        2689    ;--------------------------------------------
                        2690    ; SEEK
                        2691    ;       THIS ROUTINE WILL MOVE THE HEAD ON THE NAMED DRIVE
                        2692    ;       TO THE NAMED TRACK.  IF THE DRIVE HAS NOT BEEN ACCESSED
                        2693    ;       SINCE THE DRIVE RESET COMMAND WAS ISSUED, THE DRIVE WILL BE
                        2694    ;       RECALIBRATED.
                        2695    ; INPUT
                        2696    ;       (DL) = DRIVE TO SEEK ON
                        2697    ;       (CH) = TRACK TO SEEK TO
                        2698    ; OUTPUT
                        2699    ;       CY = 0 SUCCESS
                        2700    ;       CY = 1 FAILURE -- DISKETTE_STATUS SET ACCORDINGLY
                        2701    ;       (AX) DESTROYED
                        2702    ;--------------------------------------------
EE81                    2703    SEEK    PROC    NEAR
EE81 B001               2704            MOV     AL,1            ; ESTABLISH MASK FOR RECAL TEST
EE83 51                 2705            PUSH    CX              ; SAVE INPUT VALUES
EE84 8ACA               2706            MOV     CL,DL           ; GET DRIVE VALUE INTO CL
EE86 D2C0               2707            ROL     AL,CL           ; SHIFT IT BY THE DRIVE VALUE
EE88 59                 2708            POP     CX              ; RECOVER TRACK VALUE
EE89 84063E00     R     2709            TEST    AL,SEEK_STATUS  ; TEST FOR RECAL REQUIRED
EE8D 7513               2710            JNZ     J28             ; NO_RECAL
EE8F 08063E00     R     2711            OR      SEEK_STATUS,AL  ; TURN ON THE NO RECAL BIT IN FLAG
EE93 B407               2712            MOV     AH,07H          ; RECALIBRATE COMMAND
EE95 E8ACFF             2713            CALL    NEC_OUTPUT
EE98 8AE2               2714            MOV     AH,DL
EE9A E8A7FF             2715            CALL    NEC_OUTPUT      ; OUTPUT THE DRIVE NUMBER
```

```
LOC OBJ                 LINE    SOURCE


EE9D E87200             2716            CALL    CHK_STAT_2      ; GET THE INTERUPT AND SENSE INT STATUS
EEA0 7229               2717            JC      J32             ; SEEK_ERROR
                        2718
                        2719    ;----- DRIVE IS IN SYNCH WITH CONTROLLER, SEEK TO TRACK
                        2720
EEA2                    2721    J28:
EEA2 B40F               2722            MOV     AH,0FH          ; SEEK COMMAND TO NEC
EEA4 E89DFF             2723            CALL    NEC_OUTPUT
EEA7 8AE2               2724            MOV     AH,DL           ; DRIVE NUMBER
EEA9 E898FF             2725            CALL    NEC_OUTPUT
EEAC 8AE5               2726            MOV     AH,CH           ; TRACK NUMBER
EEAE E893FF             2727            CALL    NEC_OUTPUT
EEB1 E85E00             2728            CALL    CHK_STAT_2      ; GET ENDING INTERRUPT AND SENSE STATUS
                        2729
                        2730    ;----- WAIT FOR HEAD SETTLE
                        2731
EEB4 9C                 2732            PUSHF                   ; SAVE STATUS FLAGS
EEB5 BB1200             2733            MOV     BX,18           ; GET HEAD SETTLE PARAMETER
EEB8 E8B5FF             2734            CALL    GET_PARM
EEBB 51                 2735            PUSH    CX              ; SAVE REGISTER
EEBC                    2736    J29:                            ; HEAD_SETTLE
EEBC B92602             2737            MOV     CX,550          ; 1 MS LOOP
EEBF 0AE4               2738            OR      AH,AH           ; TEST FOR TIME EXPIRED
EEC1 7406               2739            JZ      J31
EEC3 E2FE               2740    J30:    LOOP    J30             ; DELAY FOR 1 MS
EEC5 FECC               2741            DEC     AH              ; DECREMENT THE COUNT
EEC7 EBF3               2742            JMP     J29             ; DO IT SOME MORE
EEC9                    2743    J31:
EEC9 59                 2744            POP     CX              ; RECOVER STATE
EECA 9D                 2745            POPF
EECB                    2746    J32:                            ; SEEK_ERROR
EECB C3                 2747            RET                     ; RETURN TO CALLER
                        2748    SEEK    ENDP
                        2749    ;--------------------------------------------
                        2750    ; DMA_SETUP
                        2751    ;       THIS ROUTINE SETS UP THE DMA FOR READ/WRITE/VERIFY
                        2752    ;       OPERATIONS.
                        2753    ; INPUT
                        2754    ;       (AL) = MODE BYTE FOR THE DMA
                        2755    ;       (ES:BX) - ADDRESS TO READ/WRITE THE DATA
                        2756    ; OUTPUT
                        2757    ;       (AX) DESTROYED
                        2758    ;--------------------------------------------
EECC                    2759    DMA_SETUP       PROC    NEAR
EECC 51                 2760            PUSH    CX              ; SAVE THE REGISTER
EECD E60C               2761            OUT     DMA+12,AL       ; SET THE FIRST/LAST F/F
EECF E60B               2762            OUT     DMA+11,AL       ; OUTPUT THE MODE BYTE
EED1 8CC0               2763            MOV     AX,ES           ; GET THE ES VALUE
EED3 B104               2764            MOV     CL,4            ; SHIFT COUNT
EED5 D3C0               2765            ROL     AX,CL           ; ROTATE LEFT
EED7 8AE8               2766            MOV     CH,AL           ; GET HIGHEST NYBLE OF ES TO CH
EED9 24F0               2767            AND     AL,0F0H         ; ZERO THE LOW NYBBLE FROM SEGMENT
EEDB 03C3               2768            ADD     AX,BX           ; TEST FOR CARRY FROM ADDITION
EEDD 7302               2769            JNC     J33
EEDF FEC5               2770            INC     CH              ; CARRY MEANS HIGH 4 BITS MUST BE INC
EEE1                    2771    J33:
EEE1 50                 2772            PUSH    AX              ; SAVE START ADDRESS
EEE2 E604               2773            OUT     DMA+4,AL        ; OUTPUT LOW ADDRESS
EEE4 8AC4               2774            MOV     AL,AH
EEE6 E604               2775            OUT     DMA+4,AL        ; OUTPUT HIGH ADDRESS
EEE8 8AC5               2776            MOV     AL,CH           ; GET HIGH 4 BITS
EEEA 240F               2777            AND     AL,0FH
EEEC E681               2778            OUT     081H,AL         ; OUTPUT THE HIGH 4 BITS TO PAGE REGISTER
                        2779
                        2780    ;------ DETERMINE COUNT
                        2781
EEEE 8AE6               2782            MOV     AH,DH           ; NUMBER OF SECTORS
EEF0 2AC0               2783            SUB     AL,AL           ;  TIMES 256 INTO AX
EEF2 D1E8               2784            SHR     AX,1            ; SECTORS * 128 INTO AX
EEF4 50                 2785            PUSH    AX
EEF5 BB0600             2786            MOV     BX,6            ; GET THE BYTES/SECTOR PARM
EEF8 E875FF             2787            CALL    GET_PARM
EEFB 8ACC               2788            MOV     CL,AH           ; USE AS SHIFT COUNT (0=128, 1=256 ETC)
EEFD 58                 2789            POP     AX
EEFE D3E0               2790            SHL     AX,CL           ; MULTIPLY BY CORRECT AMOUNT
EF00 48                 2791            DEC     AX              ; -1 FOR DMA VALUE
EF01 50                 2792            PUSH    AX              ; SAVE COUNT VALUE
```

```
LOC OBJ                 LINE    SOURCE

EF02 E605               2793            OUT     DMA+5,AL        ; LOW BYTE OF COUNT
EF04 8AC4               2794            MOV     AL,AH
EF06 E605               2795            OUT     DMA+5,AL        ; HIGH BYTE OF COUNT
EF08 59                 2796            POP     CX              ; RECOVER COUNT VALUE
EF09 58                 2797            POP     AX              ; RECOVER ADDRESS VALUE
EF0A 03C1               2798            ADD     AX,CX           ; ADD, TEST FOR 64K OVERFLOW
EF0C 59                 2799            POP     CX              ; RECOVER REGISTER
EF0D B002               2800            MOV     AL,2            ; MODE FOR 8237
EF0F E60A               2801            OUT     DMA+10,AL       ; INITIALIZE THE DISKETTE CHANNEL
EF11 C3                 2802            RET                     ; RETURN TO CALLER, CFL SET BY ABOVE IF ERROR
                        2803    DMA_SETUP       ENDP
                        2804    ;--------------------------------------------
                        2805    ; CHK_STAT_2
                        2806    ;       THIS ROUTINE HANDLES THE INTERRUPT RECEIVED AFTER
                        2807    ;       A RECALIBRATE, SEEK, OR RESET TO THE ADAPTER.
                        2808    ;       THE INTERRUPT IS WAITED FOR, THE INTERRUPT STATUS SENSED,
                        2809    ;       AND THE RESULT RETURNED TO THE CALLER.
                        2810    ; INPUT
                        2811    ;       NONE
                        2812    ; OUTPUT
                        2813    ;       CY = 0 SUCCESS
                        2814    ;       CY = 1 FAILURE -- ERROR IS IN DISKETTE_STATUS
                        2815    ;       (AX) DESTROYED
                        2816    ;--------------------------------------------
EF12                    2817    CHK_STAT_2      PROC    NEAR
EF12 E81E00             2818            CALL    WAIT_INT                ; WAIT FOR THE INTERRUPT
EF15 7214               2819            JC      J34                     ; IF ERROR, RETURN IT
EF17 B408               2820            MOV     AH,08H                  ; SENSE INTERRUPT STATUS COMMAND
EF19 E828FF             2821            CALL    NEC_OUTPUT
EF1C E84C00             2822            CALL    RESULTS                 ; READ IN THE RESULTS
EF1F 720A               2823            JC      J34                     ; CHK2_RETURN
EF21 A04200         R   2824            MOV     AL,NEC_STATUS           ; GET THE FIRST STATUS BYTE
EF24 2460               2825            AND     AL,060H                 ; ISOLATE THE BITS
EF26 3C60               2826            CMP     AL,060H                 ; TEST FOR CORRECT VALUE
EF28 7402               2827            JZ      J35                     ; IF ERROR, GO MARK IT
EF2A F8                 2828            CLC                             ; GOOD RETURN
EF2B                    2829    J34:
EF2B C3                 2830            RET                             ; RETURN TO CALLER
EF2C                    2831    J35:                                    ; CHK2_ERROR
EF2C 800E410040     R   2832            OR      DISKETTE_STATUS,BAD_SEEK
EF31 F9                 2833            STC                             ; ERROR RETURN CODE
EF32 C3                 2834            RET
                        2835    CHK_STAT_2      ENDP
                        2836    ;--------------------------------------------
                        2837    ; WAIT_INT
                        2838    ;       THIS ROUTINE WAITS FOR AN INTERRUPT TO OCCUR
                        2839    ;       A TIME OUT ROUTINE TAKES PLACE DURING THE WAIT, SO
                        2840    ;       THAT AN ERROR MAY BE RETURNED IF THE DRIVE IS NOT READY
                        2841    ; INPUT
                        2842    ;       NONE
                        2843    ; OUTPUT
                        2844    ;       CY = 0 SUCCESS
                        2845    ;       CY = 1 FAILURE -- DISKETTE_STATUS IS SET ACCORDINGLY
                        2846    ;       (AX) DESTROYED
                        2847    ;--------------------------------------------
EF33                    2848    WAIT_INT        PROC    NEAR
EF33 FB                 2849            STI             ; TURN ON INTERRUPTS, JUST IN CASE
EF34 53                 2850            PUSH    BX
EF35 51                 2851            PUSH    CX                      ; SAVE REGISTERS
EF36 B302               2852            MOV     BL,2                    ; CLEAR THE COUNTERS
EF38 33C9               2853            XOR     CX,CX                   ; FOR 2 SECOND WAIT
EF3A                    2854    J36:
EF3A F6063E0080     R   2855            TEST    SEEK_STATUS,INT_FLAG    ; TEST FOR INTERRUPT OCCURRING
EF3F 750C               2856            JNZ     J37
EF41 E2F7               2857            LOOP    J36                     ; COUNT DOWN WHILE WAITING
EF43 FECB               2858            DEC     BL                      ; SECOND LEVEL COUNTER
EF45 75F3               2859            JNZ     J36
EF47 800E410080     R   2860            OR      DISKETTE_STATUS,TIME_OUT        ; NOTHING HAPPENED
EF4C F9                 2861            STC                             ; ERROR RETURN
EF4D                    2862    J37:
EF4D 9C                 2863            PUSHF                           ; SAVE CURRENT CARRY
EF4E 80263E007F     R   2864            AND     SEEK_STATUS,NOT INT_FLAG        ; TURN OFF INTERRUPT FLAG
EF53 9D                 2865            POPF                            ; RECOVER CARRY
EF54 59                 2866            POP     CX
EF55 5B                 2867            POP     BX              ; RECOVER REGISTERS
EF56 C3                 2868            RET                     ; GOOD RETURN CODE COMES FROM TEST INST
                        2869    WAIT_INT        ENDP
```

```
LOC OBJ                  LINE    SOURCE


                         2870    ;--------------------------------------------
                         2871    ; DISK_INT
                         2872    ;       THIS ROUTINE HANDLES THE DISKETTE INTERRUPT
                         2873    ;INPUT
                         2874    ;       NONE
                         2875    ; OUTPUT
                         2876    ;       THE INTERRUPT FLAG IS SET IS SEEK_STATUS
                         2877    ;--------------------------------------------
EF57                     2878    DISK_INT        PROC    FAR
EF57 FB                  2879            STI                     ; RE ENABLE INTERRUPTS
EF58 1E                  2880            PUSH    DS
EF59 50                  2881            PUSH    AX
EF5A B84000        R     2882            MOV     AX,DATA
EF5D 8ED8                2883            MOV     DS,AX
EF5F 800E3E0080    R     2884            OR      SEEK_STATUS,INT_FLAG
EF64 B020                2885            MOV     AL,20H                  ; END OF INTERRUPT MARKER
EF66 E620                2886            OUT     20H,AL                  ; INTERRUPT CONTROL PORT
EF68 58                  2887            POP     AX
EF69 1F                  2888            POP     DS                      ; RECOVER SYSTEM
EF6A CF                  2889            IRET                            ; RETURN FROM INTERRUPT
                         2890    DISK_INT        ENDP
                         2891    ;--------------------------------------------
                         2892    ; RESULTS
                         2893    ;       THIS ROUTINE WILL READ ANYTHING THAT THE NEC CONTROLLER
                         2894    ;       HAS TO SAY FOLLOWING AN INTERRUPT.
                         2895    ; INPUT
                         2896    ;       NONE
                         2897    ; OUTPUT
                         2898    ;       CY = 0  SUCCESSFUL TRANSFER
                         2899    ;       CY = 1  FAILURE -- TIME OUT IN WAITING FOR STATUS
                         2900    ;       NEC_STATUS AREA HAS STATUS BYTE LOADED INTO IT
                         2901    ;       (AH) DESTROYED
                         2902    ;--------------------------------------------
EF6B                     2903    RESULTS PROC    NEAR
EF6B FC                  2904            CLD
EF6C BF4200        R     2905            MOV     DI,OFFSET NEC_STATUS    ; POINTER TO DATA AREA
EF6F 51                  2906            PUSH    CX                      ; SAVE COUNTER
EF70 52                  2907            PUSH    DX
EF71 53                  2908            PUSH    BX
EF72 B307                2909            MOV     BL,7                    ; MAX STATUS BYTES
                         2910
                         2911    ;------ WAIT FOR REQUEST FOR MASTER
                         2912
EF74                     2913    J38:                                    ; INPUT_LOOP
EF74 33C9                2914            XOR     CX,CX                   ; COUNTER
EF76 BAF403              2915            MOV     DX,03F4H                ; STATUS PORT
EF79                     2916    J39:                                    ; WAIT FOR MASTER
EF79 EC                  2917            IN      AL,DX                   ; GET STATUS
EF7A A880                2918            TEST    AL,080H                 ; MASTER READY
EF7C 750C                2919            JNZ     J40A                    ; TEST_DIR
EF7E E2F9                2920            LOOP    J39                     ; WAIT_MASTER
EF80 800E410080    R     2921            OR      DISKETTE_STATUS,TIME_OUT
EF85                     2922    J40:                                    ; RESULTS_ERROR
EF85 F9                  2923            STC                             ; SET ERROR RETURN
EF86 5B                  2924            POP     BX
EF87 5A                  2925            POP     DX
EF88 59                  2926            POP     CX
EF89 C3                  2927            RET
                         2928
                         2929    ;------ TEST THE DIRECTION BIT
                         2930
EF8A EC                  2931    J40A:   IN      AL,DX           ; GET STATUS REG AGAIN
EF8B A840                2932            TEST    AL,040H         ; TEST DIRECTION BIT
EF8D 7507                2933            JNZ     J42             ; OK TO READ STATUS
EF8F                     2934    J41:                            ; NEC_FAIL
EF8F 800E410020    R     2935            OR      DISKETTE_STATUS,BAD_NEC
EF94 EBEF                2936            JMP     J40             ; RESULTS_ERROR
                         2937
                         2938    ;------ READ IN THE STATUS
                         2939
EF96                     2940    J42:                            ; INPUT_STAT
EF96 42                  2941            INC     DX              ; POINT AT DATA PORT
EF97 EC                  2942            IN      AL,DX           ; GET THE DATA
EF98 8805                2943            MOV     [DI],AL         ; STORE THE BYTE
EF9A 47                  2944            INC     DI              ; INCREMENT THE POINTER
EF9B B90A00              2945            MOV     CX,10           ; LOOP TO KILL TIME FOR NEC
```

```
LOC OBJ                 LINE    SOURCE


EF9E E2FE               2946    J43:    LOOP    J43
EFA0 4A                 2947            DEC     DX              ; POINT AT STATUS PORT
EFA1 EC                 2948            IN      AL,DX           ; GET STATUS
EFA2 A810               2949            TEST    AL,010H         ; TEST FOR NEC STILL BUSY
EFA4 7406               2950            JZ      J44             ; RESULTS DONE
EFA6 FECB               2951            DEC     BL              ; DECREMENT THE STATUS COUNTER
EFA8 75CA               2952            JNZ     J38             ; GO BACK FOR MORE
EFAA EBE3               2953            JMP     J41             ; CHIP HAS FAILED
                        2954
                        2955    ;------ RESULT OPERATION IS DONE
                        2956
EFAC                    2957    J44:
EFAC 5B                 2958            POP     BX
EFAD 5A                 2959            POP     DX
EFAE 59                 2960            POP     CX              ; RECOVER REGISTERS
EFAF C3                 2961            RET                     ; GOOD RETURN CODE FROM TEST INST
                        2962    ;----------------------------------------------
                        2963    ; NUM_TRANS
                        2964    ;       THIS ROUTINE CALCULATES THE NUMBER OF SECTORS THAT
                        2965    ;       WERE ACTUALLY TRANSFERRED TO/FROM THE DISKETTE
                        2966    ; INPUT
                        2967    ;       (CH) = CYLINDER OF OPERATION
                        2968    ;       (CL) = START SECTOR OF OPERATION
                        2969    ; OUTPUT
                        2970    ;       (AL) = NUMBER ACTUALLY TRANSFERRED
                        2971    ;       NO OTHER REGISTERS MODIFIED
                        2972    ;----------------------------------------------
EFB0                    2973    NUM_TRANS       PROC    NEAR
EFB0 A04500         R   2974            MOV     AL,NEC_STATUS+3         ; GET CYLINDER ENDED UP ON
EFB3 3AC5               2975            CMP     AL,CH                   ; SAME AS WE STARTED
EFB5 A04700         R   2976            MOV     AL,NEC_STATUS+5         ; GET ENDING SECTOR
EFB8 740A               2977            JZ      J45                     ; IF ON SAME CYL, THEN NO ADJUST
EFBA BB0800             2978            MOV     BX,8
EFBD E8B0FE             2979            CALL    GET_PARM                ; GET EOT VALUE
EFC0 8AC4               2980            MOV     AL,AH                   ;  INTO AL
EFC2 FEC0               2981            INC     AL                      ; USE EOT+1 FOR CALCULATION
EFC4 2AC1               2982    J45:    SUB     AL,CL                   ; SUBTRACT START FROM END
EFC6 C3                 2983            RET
                        2984    NUM_TRANS       ENDP
                        2985    RESULTS ENDP
                        2986
                        2987    ;----------------------------------------------
                        2988    ; DISK_BASE
                        2989    ;  THIS IS THE SET OF PARAMETERS REQUIRED FOR
                        2990    ;  DISKETTE OPERATION.  THEY ARE POINTED AT BY THE
                        2991    ;  DATA VARIABLE DISK_POINTER.  TO MODIFY THE PARAMETERS,
                        2992    ;  BUILD ANOTHER PARAMETER BLOCK AND POINT AT IT
                        2993    ;----------------------------------------------
                        2994
EFC7                    2995    DISK_BASE       LABEL   BYTE
EFC7 CF                 2996            DB      11001111B       ; SRT=C, HD UNLOAD=0F - 1ST SPECIFY BYTE
EFC8 02                 2997            DB      2               ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
EFC9 25                 2998            DB      MOTOR_WAIT      ; WAIT AFTER OPN TIL MOTOR OFF
EFCA 02                 2999            DB      2               ; 512 BYTES/SECTOR
EFCB 08                 3000            DB      8               ; EOT ( LAST SECTOR ON TRACK)
EFCC 2A                 3001            DB      02AH            ; GAP LENGTH
EFCD FF                 3002            DB      0FFH            ; DTL
EFCE 50                 3003            DB      050H            ; GAP LENGTH FOR FORMAT
EFCF F6                 3004            DB      0F6H            ; FILL BYTE FOR FORMAT
EFD0 19                 3005            DB      25              ; HEAD SETTLE TIME (MILLISECONDS)
EFD1 04                 3006            DB      4               ; MOTOR START TIME (1/8 SECONDS)
```

```
LOC OBJ                 LINE    SOURCE


                        3007    ;--- INT 17 ---------------------------------
                        3008    ;PRINTER_IO
                        3009    ;       THIS ROUTINE PROVIDES COMMUNICATION WITH THE PRINTER
                        3010    ;       (AH)=0  PRINT THE CHARACTER IN (AL)
                        3011    ;               ON RETURN, AH=1 IF CHARACTER COULD NOT BE PRINTED (TIME OUT)
                        3012    ;               OTHER BITS SET AS ON NORMAL STATUS CALL
                        3013    ;       (AH)=1  INITIALIZE THE PRINTER PORT
                        3014    ;               RETURNS WITH (AH) SET WITH PRINTER STATUS
                        3015    ;       (AH)=2  READ THE PRINTER STATUS INTO (AH)
                        3016    ;               7       6       5       4       3       2-1     0
                        3017    ;               |       |       |       |       |       |       |_ TIME OUT
                        3018    ;               |       |       |       |       |       |_ UNUSED
                        3019    ;               |       |       |       |       |_ 1 = I/O ERROR
                        3020    ;               |       |       |       |_ 1 = SELECTED
                        3021    ;               |       |       |_ 1 = OUT OF PAPER
                        3022    ;               |       |_ 1 = ACKNOWLEDGE
                        3023    ;               |_ 1 = BUSY
                        3024    ;
                        3025    ;       (DX) = PRINTER TO BE USED (0,1,2) CORRESPONDING TO ACTUAL VALUES
                        3026    ;               IN PRINTER_BASE AREA
                        3027    ; DATA AREA PRINTER_BASE CONTAINS THE BASE ADDRESS OF THE PRINTER CARD(S)
                        3028    ;   AVAILABLE (LOCATED AT BEGINNING OF DATA SEGMENT, 408H ABSOLUTE, 3 WORDS)
                        3029    ;REGISTERS      AH IS MODIFIED
                        3030    ;               ALL OTHERS UNCHANGED
                        3031    ;-----------------------------------------------
                        3032            ASSUME  CS:CODE,DS:DATA
EFD2                    3033    PRINTER_IO      PROC    FAR
EFD2 FB                 3034            STI                     ; INTERRUPTS BACK ON
EFD3 1E                 3035            PUSH    DS              ; SAVE SEGMENT
EFD4 52                 3036            PUSH    DX
EFD5 56                 3037            PUSH    SI
EFD6 51                 3038            PUSH    CX
EFD7 53                 3039            PUSH    BX
EFD8 BE4000         R   3040            MOV     SI,DATA
EFDB 8EDE               3041            MOV     DS,SI           ; ESTABLISH PRINTER SEGMENT
EFDD 8BF2               3042            MOV     SI,DX           ; GET PRINTER PARM
EFDF D1E6               3043            SHL     SI,1            ; WORD OFFSET INTO TABLE
EFE1 8B940800       R   3044            MOV     DX,PRINTER_BASE[SI]     ; GET BASE ADDRESS FOR PRINTER CARD
EFE5 0BD2               3045            OR      DX,DX           ; TEST DX FOR ZERO, INDICATING NO PRINTER
EFE7 740C               3046            JZ      B1              ; RETURN
EFE9 0AE4               3047            OR      AH,AH           ; TEST FOR (AH)=0
EFEB 740E               3048            JZ      B2              ; PRINT_AL
EFED FECC               3049            DEC     AH              ; TEST FOR (AH)=1
EFEF 7442               3050            JZ      B8              ; INIT_PRT
EFF1 FECC               3051            DEC     AH              ; TEST FOR (AH)=2
EFF3 742A               3052            JZ      B5              ; PRINTER STATUS
EFF5                    3053    B1:                             ; RETURN
EFF5 5B                 3054            POP     BX
EFF6 59                 3055            POP     CX
EFF7 5E                 3056            POP     SI              ; RECOVER REGISTERS
EFF8 5A                 3057            POP     DX              ; RECOVER REGISTERS
EFF9 1F                 3058            POP     DS
EFFA CF                 3059            IRET
                        3060
                        3061    ;------ PRINT THE CHARACTER IN (AL)
                        3062
EFFB                    3063    B2:
EFFB 50                 3064            PUSH    AX              ; SAVE VALUE TO PRINT
EFFC B30A               3065            MOV     BL,10           ; TIME OUT VALUE
EFFE 33C9               3066            XOR     CX,CX           ; ESTABLISH SHIFT COUNT
F000 EE                 3067            OUT     DX,AL           ; OUTPUT CHAR TO PORT
F001 42                 3068            INC     DX              ; POINT TO STATUS PORT
F002                    3069    B3:                             ; WAIT_BUSY
F002 EC                 3070            IN      AL,DX           ; GET STATUS
F003 8AE0               3071            MOV     AH,AL           ; STATUS TO AH ALSO
F005 A880               3072            TEST    AL,80H          ; IS THE PRINTER CURRENTLY BUSY
F007 750E               3073            JNZ     B4              ; OUT_STROBE
F009 E2F7               3074            LOOP    B3              ; DECREMENT COUNT ON TIME OUT
F00B FECB               3075            DEC     BL
F00D 75F3               3076            JNZ     B3              ; WAIT FOR NOT BUSY
F00F 80CC01             3077            OR      AH,1            ; SET ERROR FLAG
F012 80E4F9             3078            AND     AH,0F9H         ; TURN OFF THE OTHER BITS
F015 EB14               3079            JMP     SHORT B7        ; RETURN WITH ERROR FLAG SET
F017                    3080    B4:                             ; OUT_STROBE
F017 B00D               3081            MOV     AL,0DH          ; SET THE STROBE HIGH
F019 42                 3082            INC     DX              ; STROBE IS BIT 0 OF PORT C OF 8255
```

```
LOC OBJ                  LINE    SOURCE


F01A EE                  3083            OUT     DX,AL
F01B B00C                3084            MOV     AL,0CH          ; SET THE STROBE LOW
F01D EE                  3085            OUT     DX,AL
F01E 58                  3086            POP     AX              ; RECOVER THE OUTPUT CHAR
                         3087
                         3088    ;------ PRINTER STATUS
                         3089
F01F                     3090    B5:
F01F 50                  3091            PUSH    AX              ; SAVE AL REG
F020                     3092    B6:
F020 8B940800      R     3093            MOV     DX,PRINTER_BASE[SI]
F024 42                  3094            INC     DX
F025 EC                  3095            IN      AL,DX           ; GET PRINTER STATUS
F026 8AE0                3096            MOV     AH,AL
F028 80E4F8              3097            AND     AH,0F8H         ; TURN OFF UNUSED BITS
F02B                     3098    B7:                             ; STATUS_SET
F02B 5A                  3099            POP     DX              ; RECOVER AL REG
F02C 8AC2                3100            MOV     AL,DL           ; GET CHARACTER INTO AL
F02E 80F448              3101            XOR     AH,48H          ; FLIP A COUPLE OF BITS
F031 EBC2                3102            JMP     B1              ; RETURN FROM ROUTINE
                         3103
                         3104    ;------ INITIALIZE THE PRINTER PORT
                         3105
F033                     3106    B8:
F033 50                  3107            PUSH    AX              ; SAVE AL
F034 83C202              3108            ADD     DX,2            ; POINT TO OUTPUT PORT
F037 B008                3109            MOV     AL,8            ; SET INIT LINE LOW
F039 EE                  3110            OUT     DX,AL
F03A B8E803              3111            MOV     AX,1000
F03D                     3112    B9:                             ; INIT_LOOP
F03D 48                  3113            DEC     AX              ; LOOP FOR RESET TO TAKE
F03E 75FD                3114            JNZ     B9              ; INIT_LOOP
F040 B00C                3115            MOV     AL,0CH          ; NO INTERRUPTS, NON AUTO LF, INIT HIGH
F042 EE                  3116            OUT     DX,AL
F043 EBDB                3117            JMP     B6              ; PRT_STATUS_1
                         3118    PRINTER_IO      ENDP
                         3119    ;--- INT 10 --------------------------------
                         3120    ; VIDEO_IO
                         3121    ;       THESE ROUTINES PROVIDE THE CRT INTERFACE
                         3122    ;       THE FOLLOWING FUNCTIONS ARE PROVIDED:
                         3123    ;       (AH)=0  SET MODE (AL) CONTAINS MODE VALUE
                         3124    ;               (AL)=0 40X25 BW (POWER ON DEFAULT)
                         3125    ;               (AL)=1 40X25 COLOR
                         3126    ;               (AL)=2  80X25 BW
                         3127    ;               (AL)=3  80X25 COLOR
                         3128    ;               GRAPHICS MODES
                         3129    ;               (AL)=4  320X200 COLOR
                         3130    ;               (AL)=5  320X200 BW
                         3131    ;               (AL)=6  640X200 BW
                         3132    ;               CRT MODE = 7 80X25 B&W CARD (USED INTERNAL TO VIDEO ONLY)
                         3133    ;               *** NOTE BW MODES OPERATE SAME AS COLOR MODES, BUT COLOR
                         3134    ;                       BURST IS NOT ENABLED
                         3135    ;       (AH)=1  SET CURSOR TYPE
                         3136    ;               (CH) =  BITS 4-0 = START LINE FOR CURSOR
                         3137    ;                       ** HARDWARE WILL ALWAYS CAUSE BLINK
                         3138    ;                       ** SETTING BIT 5 OR 6 WILL CAUSE ERRATIC BLINKING
                         3139    ;                          OR NO CURSOR AT ALL
                         3140    ;               (CL) =  BITS 4-0 = END LINE FOR CURSOR
                         3141    ;       (AH)=2  SET CURSOR POSITION
                         3142    ;               (DH,DL) = ROW,COLUMN  (0,0) IS UPPER LEFT
                         3143    ;               (BH) = PAGE NUMBER (MUST BE 0 FOR GRAPHICS MODES)
                         3144    ;       (AH)=3  READ CURSOR POSITION
                         3145    ;               (BH) = PAGE NUMBER (MUST BE 0 FOR GRAPHICS MODES)
                         3146    ;               ON EXIT (DH,DL) = ROW,COLUMN OF CURRENT CURSOR
                         3147    ;                       (CH,CL) = CURSOR MODE CURRENTLY SET
                         3148    ;       (AH)=4  READ LIGHT PEN POSITION
                         3149    ;               ON EXIT:
                         3150    ;               (AH) = 0 -- LIGHT PEN SWITCH NOT DOWN/NOT TRIGGERED
                         3151    ;               (AH) = 1 -- VALID LIGHT PEN VALUE IN REGISTERS
                         3152    ;                       (DH,DL) = ROW,COLUMN OF CHARACTER LP POSN
                         3153    ;                       (CH) = RASTER LINE (0-199)
                         3154    ;                       (BX) = PIXEL COLUMN (0-319,639)
                         3155    ;       (AH)=5  SELECT ACTIVE DISPLAY PAGE (VALID ONLY FOR ALPHA MODES)
                         3156    ;               (AL)=NEW PAGE VALUE (0-7 FOR MODES 0&1, 0-3 FOR MODES 2&3)
```

```
LOC OBJ                 LINE    SOURCE

                        3157    ;       (AH)=6  SCROLL ACTIVE PAGE UP
                        3158    ;               (AL) = NUMBER OF LINES, INPUT LINES BLANKED AT BOTTOM OF WINDOW
                        3159    ;                       AL = 0 MEANS BLANK ENTIRE WINDOW
                        3160    ;               (CH,CL) = ROW,COLUMN OF UPPER LEFT CORNER OF SCROLL
                        3161    ;               (DH,DL) = ROW,COLUMN OF LOWER RIGHT CORNER OF SCROLL
                        3162    ;               (BH) = ATTRIBUTE TO BE USED ON BLANK LINE
                        3163    ;       (AH)=7  SCROLL ACTIVE PAGE DOWN
                        3164    ;               (AL) = NUMBER OF LINES, INPUT LINES BLANKED AT TOP OF WINDOW
                        3165    ;                       AL = 0 MEANS BLANK ENTIRE WINDOW
                        3166    ;               (CH,CL) = ROW,COLUMN OF UPPER LEFT CORNER OF SCROLL
                        3167    ;               (DH,DL) = ROW,COLUMN OF LOWER RIGHT CORNER OF SCROLL
                        3168    ;               (BH) = ATTRIBUTE TO BE USED ON BLANK LINE
                        3169    ;
                        3170    ;       CHARACTER HANDLING ROUTINES
                        3171    ;
                        3172    ;       (AH) = 8 READ ATTRIBUTE/CHARACTER AT CURRENT CURSOR POSITION
                        3173    ;               (BH) = DISPLAY PAGE (VALID FOR ALPHA MODES ONLY)
                        3174    ;               ON EXIT:
                        3175    ;               (AL) = CHAR READ
                        3176    ;               (AH) = ATTRIBUTE OF CHARACTER READ (ALPHA MODES ONLY)
                        3177    ;       (AH) = 9 WRITE ATTRIBUTE/CHARACTER AT CURRENT CURSOR POSITION
                        3178    ;               (BH) = DISPLAY PAGE (VALID FOR ALPHA MODES ONLY)
                        3179    ;               (CX) = COUNT OF CHARACTERS TO WRITE
                        3180    ;               (AL) = CHAR TO WRITE
                        3181    ;               (BL) = ATTRIBUTE OF CHARACTER (ALPHA)/COLOR OF CHAR (GRAPHICS)
                        3182    ;                       SEE NOTE ON WRITE DOT FOR BIT 7 OF BL = 1.
                        3183    ;       (AH) = 10 WRITE CHARACTER ONLY AT CURRENT CURSOR POSITION
                        3184    ;               (BH) = DISPLAY PAGE (VALID FOR ALPHA MODES ONLY)
                        3185    ;               (CX) = COUNT OF CHARACTERS TO WRITE
                        3186    ;               (AL) = CHAR TO WRITE
                        3187    ;       FOR READ/WRITE CHARACTER INTERFACE WHILE IN GRAPHICS MODE, THE
                        3188    ;               CHARACTERS ARE FORMED FROM A CHARACTER GENERATOR IMAGE
                        3189    ;               MAINTAINED IN THE SYSTEM ROM.  ONLY THE 1ST 128 CHARS
                        3190    ;               ARE CONTAINED THERE.  TO READ/WRITE THE SECOND 128 CHARS,
                        3191    ;               THE USER MUST INITIALIZE THE POINTER AT INTERRUPT 1FH
                        3192    ;               (LOCATION 0007CH) TO POINT TO THE 1K BYTE TABLE CONTAINING
                        3193    ;               THE CODE POINTS FOR THE SECOND 128 CHARS (128-255).
                        3194    ;       FOR WRITE CHARACTER INTERFACE IN GRAPHICS MODE, THE REPLICATION FACTOR
                        3195    ;               CONTAINED IN (CX) ON ENTRY WILL PRODUCE VALID RESULTS ONLY
                        3196    ;               FOR CHARACTERS CONTAINED ON THE SAME ROW.  CONTINUATION TO
                        3197    ;               SUCCEEDING LINES WILL NOT PRODUCE CORRECTLY.
                        3198    ;
                        3199    ;       GRAPHICS INTERFACE
                        3200    ;       (AH) = 11 SET COLOR PALETTE
                        3201    ;               (BH) = PALLETTE COLOR ID BEING SET (0-127)
                        3202    ;               (BL) = COLOR VALUE TO BE USED WITH THAT COLOR ID
                        3203    ;                  NOTE: FOR THE CURRENT COLOR CARD, THIS ENTRY POINT HAS
                        3204    ;                                MEANING ONLY FOR 320X200 GRAPHICS.
                        3205    ;                        COLOR ID = 0 SELECTS THE BACKGROUND COLOR (0-15)
                        3206    ;                        COLOR ID = 1 SELECTS THE PALLETTE TO BE USED:
                        3207    ;                                0 = GREEN(1)/RED(2)/YELLOW(3)
                        3208    ;                                1 = CYAN(1)/MAGENTA(2)/WHITE(3)
                        3209    ;                        IN 40X25 OR 80X25 ALPHA MODES, THE VALUE SET FOR
                        3210    ;                                PALLETTE COLOR 0 INDICATES THE BORDER COLOR
                        3211    ;                                TO BE USED (VALUES 0-31, WHERE 16-31 SELECT THE
                        3212    ;                                HIGH INTENSITY BACKGROUND SET.
                        3213    ;       (AH) = 12 WRITE DOT
                        3214    ;               (DX) = ROW NUMBER
                        3215    ;               (CX) = COLUMN NUMBER
                        3216    ;               (AL) = COLOR VALUE
                        3217    ;                       IF BIT 7 OF AL = 1, THEN THE COLOR VALUE IS EXCLUSIVE
                        3218    ;                       OR'D WITH THE CURRENT CONTENTS OF THE DOT
                        3219    ;       (AH) = 13 READ DOT
                        3220    ;               (DX) = ROW NUMBER
                        3221    ;               (CX) = COLUMN NUMBER
                        3222    ;               (AL) RETURNS THE DOT READ
                        3223    ;
                        3224    ; ASCII TELETYPE ROUTINE FOR OUTPUT
                        3225    ;
                        3226    ;       (AH) = 14 WRITE TELETYPE
                        3227    ;               (AL) = CHAR TO WRITE
                        3228    ;               (BL) = FOREGROUND COLOR IN GRAPHICS MODE
                        3229    ;               (BH) = DISPLAY PAGE IN ALPHA MODE
                        3230    ;               NOTE -- SCREEN WIDTH IS CONTROLLED BY PREVIOUS MODE SET
                        3231    ;
```

```
LOC OBJ                 LINE    SOURCE

                        3232    ;       (AH) = 15 CURRENT VIDEO STATE
                        3233    ;               RETURNS THE CURRENT VIDEO STATE
                        3234    ;               (AL) = MODE CURRENTLY SET ( SEE AH=0 FOR EXPLANATION)
                        3235    ;               (AH) = NUMBER OF CHARACTER COLUMNS ON SCREEN
                        3236    ;               (BH) = CURRENT ACTIVE DISPLAY PAGE
                        3237    ;
                        3238    ;       CS,SS,DS,ES,BX,CX,DX PRESERVED DURING CALL
                        3239    ;       ALL OTHERS DESTROYED
                        3240    ;--------------------------------------------
                        3241            ASSUME  CS:CODE,DS:DATA,ES:VIDEO_RAM
                        3242
F045                    3243    M1      LABEL   WORD    ; TABLE OF ROUTINES WITHIN VIDEO I/O
F045 FCF0          R    3244            DW      OFFSET  SET_MODE
F047 CFF1          R    3245            DW      OFFSET  SET_CTYPE
F049 F0F1          R    3246            DW      OFFSET  SET_CPOS
F04B 1AF2          R    3247            DW      OFFSET  READ_CURSOR
F04D A9F7          R    3248            DW      OFFSET  READ_LPEN
F04F 30F2          R    3249            DW      OFFSET  ACT_DISP_PAGE
F051 9CF2          R    3250            DW      OFFSET  SCROLL_UP
F053 41F3          R    3251            DW      OFFSET  SCROLL_DOWN
F055 7DF3          R    3252            DW      OFFSET  READ_AC_CURRENT
F057 C3F3          R    3253            DW      OFFSET  WRITE_AC_CURRENT
F059 F6F3          R    3254            DW      OFFSET  WRITE_C_CURRENT
F05B 54F2          R    3255            DW      OFFSET  SET_COLOR
F05D 38F4          R    3256            DW      OFFSET  WRITE_DOT
F05F 27F4          R    3257            DW      OFFSET  READ_DOT
F061 22F7          R    3258            DW      OFFSET  WRITE_TTY
F063 7AF2          R    3259            DW      OFFSET  VIDEO_STATE
  0020                  3260    M1L     EQU     $-M1
                        3261
F065                    3262    VIDEO_IO        PROC    NEAR
F065 FB                 3263            STI             ; INTERRUPTS BACK ON
F066 FC                 3264            CLD             ; SET DIRECTION FORWARD
F067 06                 3265            PUSH    ES
F068 1E                 3266            PUSH    DS              ; SAVE SEGMENT REGISTERS
F069 52                 3267            PUSH    DX
F06A 51                 3268            PUSH    CX
F06B 53                 3269            PUSH    BX
F06C 56                 3270            PUSH    SI
F06D 57                 3271            PUSH    DI
F06E 50                 3272            PUSH    AX              ; SAVE AX VALUE
F06F 8AC4               3273            MOV     AL,AH           ; GET INTO LOW BYTE
F071 32E4               3274            XOR     AH,AH           ; ZERO TO HIGH BYTE
F073 D1E0               3275            SAL     AX,1            ; *2 FOR TABLE LOOKUP
F075 8BF0               3276            MOV     SI,AX           ; PUT INTO SI FOR BRANCH
F077 3D2000             3277            CMP     AX,M1L          ; TEST FOR WITHIN RANGE
F07A 7204               3278            JB      M2              ; BRANCH AROUND BRANCH
F07C 58                 3279            POP     AX              ; THROW AWAY THE PARAMETER
F07D E94701             3280            JMP     VIDEO_RETURN            ; DO NOTHING IF NOT IN RANGE
F080 B84000        R    3281    M2:     MOV     AX,DATA
F083 8ED8               3282            MOV     DS,AX
F085 B800B8             3283            MOV     AX,0B800H       ; SEGMENT FOR COLOR CARD
F088 8B3E1000      R    3284            MOV     DI,EQUIP_FLAG   ; GET EQUIPMENT SETTING
F08C 81E73000           3285            AND     DI,30H          ; ISOLATE CRT SWITCHES
F090 83FF30             3286            CMP     DI,30H          ; IS SETTING FOR BW CARD?
F093 7503               3287            JNE     M3
F095 B800B0             3288            MOV     AX,0B000H       ; SEGMENT FOR BW CARD
F098 8EC0               3289    M3:     MOV     ES,AX           ; SET UP TO POINT AT VIDEO RAM AREAS
F09A 58                 3290            POP     AX              ; RECOVER VALUE
F09B 8A26490D      R    3291            MOV     AH,CRT_MODE     ; GET CURRENT MODE INTO AH
F09F 2EFFA445F0    R    3292            JMP     WORD PTR CS:[SI+OFFSET M1]
                        3293    VIDEO_IO        ENDP
                        3294    ;--------------------------------------------
                        3295    ; SET_MODE
                        3296    ;       THIS ROUTINE INITIALIZES THE ATTACHMENT TO
                        3297    ;       THE SELECTED MODE.  THE SCREEN IS BLANKED.
                        3298    ; INPUT
                        3299    ;       (AL) = MODE SELECTED (RANGE 0-9)
                        3300    ; OUTPUT
                        3301    ;       NONE
                        3302    ;--------------------------------------------
                        3303
                        3304    ;------ TABLES FOR USE IN SETTING OF MODE
                        3305
F0A4                    3306    VIDEO_PARMS     LABEL   BYTE
                        3307    ;------ INIT_TABLE
```

```
LOC OBJ                  LINE    SOURCE


F0A4 38282D0A1F0619      3308            DB      38H,28H,2DH,0AH,1FH,6,19H       ; SET UP FOR 40X25
F0AB 1C02070607          3309            DB      1CH,2,7,6,7
F0B0 00000000            3310            DB      0,0,0,0
  0010                   3311    M4      EQU     $-VIDEO_PARMS
                         3312
F0B4 71505A0A1F0619      3313            DB      71H,50H,5AH,0AH,1FH,6,19H       ; SET UP FOR 80X25
F0BB 1C02070607          3314            DB      1CH,2,7,6,7
F0C0 00000000            3315            DB      0,0,0,0
                         3316
F0C4 38282D0A7F0664      3317            DB      38H,28H,2DH,0AH,7FH,6,64H       ; SET UP FOR GRAPHICS
F0CB 7002010607          3318            DB      70H,2,1,6,7
F0D0 00000000            3319            DB      0,0,0,0
                         3320
F0D4 6150520F190619      3321            DB      61H,50H,52H,0FH,19H,6,19H       ; SET UP FOR 80X25 B&W CARD
F0DB 19020D0B0C          3322            DB      19H,2,0DH,0BH,0CH
F0E0 00000000            3323            DB      0,0,0,0
                         3324
F0E4                     3325    M5      LABEL   WORD                    ; TABLE OF REGEN LENGTHS
F0E4 0008                3326            DW      2048            ; 40X25
F0E6 0010                3327            DW      4096            ; 80X25
F0E8 0040                3328            DW      16384           ; GRAPHICS
F0EA 0040                3329            DW      16384           ;
                         3330
                         3331    ;------ COLUMNS
F0EC                     3332    M6      LABEL   BYTE
F0EC 2828505028285050    3333            DB      40,40,80,80,40,40,80,80
                         3334
                         3335    ;------ C_REG_TAB
F0F4                     3336    M7      LABEL   BYTE                    ; TABLE OF MODE SETS
F0F4 2C282D292A2E1E29    3337            DB      2CH,28H,2DH,29H,2AH,2EH,1EH,29H ;
                         3338
F0FC                     3339    SET_MODE        PROC    NEAR
F0FC BAD403              3340            MOV     DX,03D4H                ; ADDRESS OF COLOR CARD
F0FF B300                3341            MOV     BL,0                    ; MODE SET FOR COLOR CARD
F101 83FF30              3342            CMP     DI,30H                  ; IS BW CARD INSTALLED
F104 7507                3343            JNE     M8                      ; OK WITH COLOR
F106 B007                3344            MOV     AL,7                    ; INDICATE BW CARD MODE
F108 BAB403              3345            MOV     DX,03B4H                ; ADDRESS OF BW CARD
F10B FEC3                3346            INC     BL                      ; MODE SET FOR BW CARD
F10D 8AE0                3347    M8:     MOV     AH,AL           ; SAVE MODE IN AH
F10F A24900          R   3348            MOV     CRT_MODE,AL     ; SAVE IN GLOBAL VARIABLE
F112 89166300        R   3349            MOV     ADDR_6845,DX    ; SAVE ADDRESS OF BASE
F116 1E                  3350            PUSH    DS              ; SAVE POINTER TO DATA SEGMENT
F117 50                  3351            PUSH    AX              ; SAVE MODE
F118 52                  3352            PUSH    DX              ; SAVE OUTPUT PORT VALUE
F119 83C204              3353            ADD     DX,4            ; POINT TO CONTROL REGISTER
F11C 8AC3                3354            MOV     AL,BL           ; GET MODE SET FOR CARD
F11E EE                  3355            OUT     DX,AL           ; RESET VIDEO
F11F 5A                  3356            POP     DX              ; BACK TO BASE REGISTER
F120 2BC0                3357            SUB     AX,AX           ; SET UP FOR ABS0 SEGMENT
F122 8ED8                3358            MOV     DS,AX           ; ESTABLISH VECTOR TABLE ADDRESSING
                         3359            ASSUME  DS:ABS0
F124 C51E7400            3360            LDS     BX,PARM_PTR     ; GET POINTER TO VIDEO PARMS
F128 58                  3361            POP     AX              ; RECOVER PARMS
                         3362            ASSUME  DS:CODE
F129 B91000              3363            MOV     CX,M4           ; LENGTH OF EACH ROW OF TABLE
F12C 80FC02              3364            CMP     AH,2            ; DETERMINE WHICH ONE TO USE
F12F 7210                3365            JC      M9              ; MODE IS 0 OR 1
F131 03D9                3366            ADD     BX,CX           ; MOVE TO NEXT ROW OF INIT TABLE
F133 80FC04              3367            CMP     AH,4
F136 7209                3368            JC      M9              ; MODE IS 2 OR 3
F138 03D9                3369            ADD     BX,CX           ; MOVE TO GRAPHICS ROW OF INIT_TABLE
F13A 80FC07              3370            CMP     AH,7
F13D 7202                3371            JC      M9              ; MODE IS 4,5, OR 6
F13F 03D9                3372            ADD     BX,CX           ; MOVE TO BW CARD ROW OF INIT_TABLE
                         3373
                         3374    ;------ BX POINTS TO CORRECT ROW OF INITIALIZATION TABLE
                         3375
F141                     3376    M9:                             ; OUT_INIT
F141 50                  3377            PUSH    AX              ; SAVE MODE IN AH
F142 32E4                3378            XOR     AH,AH           ; AH WILL SERVE AS REGISTER NUMBER DURING LOOP
                         3379
                         3380    ;------ LOOP THROUGH TABLE, OUTPUTTTING REG ADDRESS, THEN VALUE FROM TABLE
                         3381
F144                     3382    M10:                            ; INIT LOOP
F144 8AC4                3383            MOV     AL,AH           ; GET 6845 REGISTER NUMBER
```

```
LOC OBJ                  LINE   SOURCE

F146 EE                  3384           OUT     DX,AL
F147 42                  3385           INC     DX              ; POINT TO DATA PORT
F148 FEC4                3386           INC     AH              ; NEXT REGISTER VALUE
F14A 8A07                3387           MOV     AL,[BX]         ; GET TABLE VALUE
F14C EE                  3388           OUT     DX,AL           ; OUT TO CHIP
F14D 43                  3389           INC     BX              ; NEXT IN TABLE
F14E 4A                  3390           DEC     DX              ; BACK TO POINTER REGISTER
F14F E2F3                3391           LOOP    M10             ; DO THE WHOLE TABLE
F151 58                  3392           POP     AX              ; GET MODE BACK
F152 1F                  3393           POP     DS              ; RECOVER SEGMENT VALUE
                         3394           ASSUME  DS:DATA
                         3395
                         3396   ;------ FILL REGEN AREA WITH BLANK
                         3397
F153 33FF                3398           XOR     DI,DI           ; SET UP POINTER FOR REGEN
F155 893E4E00        R   3399           MOV     CRT_START,DI    ; START ADDRESS SAVED IN GLOBAL
F159 C606620000      R   3400           MOV     ACTIVE_PAGE,0   ; SET PAGE VALUE
F15E B90020              3401           MOV     CX,8192         ; NUMBER OF WORDS IN COLOR CARD
F161 80FC04              3402           CMP     AH,4            ; TEST FOR GRAPHICS
F164 720C                3403           JC      M12             ; NO_GRAPHICS_INIT
F166 80FC07              3404           CMP     AH,7            ; TEST FOR BW CARD
F169 7404                3405           JE      M11             ; BW_CARD_INIT
F16B 33C0                3406           XOR     AX,AX           ; FILL FOR GRAPHICS MODE
F16D EB06                3407           JMP     SHORT M13       ; CLEAR_BUFFER
F16F                     3408   M11:                            ; BW_CARD_INIT
F16F B90008              3409           MOV     CX,2048         ; BUFFER SIZE ON BW CARD
F172                     3410   M12:                            ; NO_GRAPHICS_INIT
F172 B82007              3411           MOV     AX,' '+7*256    ; FILL CHAR FOR ALPHA
F175                     3412   M13:                            ; CLEAR_BUFFER
F175 F3                  3413           REP     STOSW           ; FILL THE REGEN BUFFER WITH BLANKS
F176 AB
                         3414
                         3415   ;------ ENABLE VIDEO AND CORRECT PORT SETTING
                         3416
F177 C70660006700    R   3417           MOV     CURSOR_MODE,67H ; SET CURRENT CURSOR MODE
F17D A04900          R   3418           MOV     AL,CRT_MODE     ; GET THE MODE
F180 32E4                3419           XOR     AH,AH           ;  INTO AX REGISTER
F182 8BF0                3420           MOV     SI,AX           ; TABLE POINTER, INDEXED BY MODE
F184 8B166300        R   3421           MOV     DX,ADDR_6845    ; PREPARE TO OUTPUT TO VIDEO ENABLE PORT
F188 83C204              3422           ADD     DX,4
F18B 2E8A84F4F0      R   3423           MOV     AL,CS:[SI+OFFSET M7]
F190 EE                  3424           OUT     DX,AL           ; SET VIDEO ENABLE PORT
F191 A26500          R   3425           MOV     CRT_MODE_SET,AL ; SAVE THAT VALUE
                         3426
                         3427   ;------ DETERMINE NUMBER OF COLUMNS, BOTH FOR ENTIRE DISPLAY
                         3428   ;------ AND THE NUMBER TO BE USED FOR TTY INTERFACE
                         3429
F194 2E8A84ECF0      R   3430           MOV     AL,CS:[SI + OFFSET M6]
F199 32E4                3431           XOR     AH,AH
F19B A34A00          R   3432           MOV     CRT_COLS,AX     ; NUMBER OF COLUMNS IN THIS SCREEN
                         3433
                         3434   ;------ SET CURSOR POSITIONS
                         3435
F19E 81E60E00            3436           AND     SI,0EH          ; WORD OFFSET INTO CLEAR LENGTH TABLE
F1A2 2E8B8CE4F0      R   3437           MOV     CX,CS:[SI + OFFSET M5]  ; LENGTH TO CLEAR
F1A7 890E4C00        R   3438           MOV     CRT_LEN,CX      ; SAVE LENGTH OF CRT -- NOT USED FOR BW
F1AB B90800              3439           MOV     CX,8            ; CLEAR ALL CURSOR POSITIONS
F1AE BF5000          R   3440           MOV     DI,OFFSET CURSOR_POSN
F1B1 1E                  3441           PUSH    DS              ; ESTABLISH SEGMENT
F1B2 07                  3442           POP     ES              ;   ADDRESSING
F1B3 33C0                3443           XOR     AX,AX
F1B5 F3                  3444           REP     STOSW           ; FILL WITH ZEROES
F1B6 AB
                         3445
                         3446   ;------ SET UP OVERSCAN REGISTER
                         3447
F1B7 42                  3448           INC     DX              ; SET OVERSCAN PORT TO A DEFAULT
F1B8 B030                3449           MOV     AL,30H          ; VALUE OF 30H FOR ALL MODES EXCEPT 640X200
F1BA 803E490006      R   3450           CMP     CRT_MODE,6      ; SEE IF THE MODE IS 640X200 BW
F1BF 7502                3451           JNZ     M14             ; IF IT ISNT 640X200, THEN GOTO REGULAR
F1C1 B03F                3452           MOV     AL,3FH          ; IF IT IS 640X200, THEN PUT IN 3FH
F1C3 EE                  3453   M14:    OUT     DX,AL           ; OUTPUT THE CORRECT VALUE TO 3D9 PORT
F1C4 A26600          R   3454           MOV     CRT_PALLETTE,AL ; SAVE THE VALUE FOR FUTURE USE
                         3455
                         3456   ;------ NORMAL RETURN FROM ALL VIDEO RETURNS
                         3457
```

```
LOC OBJ                  LINE    SOURCE

F1C7                     3458    VIDEO_RETURN:
F1C7 5F                  3459            POP     DI
F1C8 5E                  3460            POP     SI
F1C9 5B                  3461            POP     BX
F1CA                     3462    M15:                            ; VIDEO_RETURN_C
F1CA 59                  3463            POP     CX
F1CB 5A                  3464            POP     DX
F1CC 1F                  3465            POP     DS
F1CD 07                  3466            POP     ES              ; RECOVER SEGMENTS
F1CE CF                  3467            IRET                    ; ALL DONE
                         3468    SET_MODE        ENDP
                         3469    ;--------------------------------------------
                         3470    ; SET_CTYPE
                         3471    ;       THIS ROUTINE SETS THE CURSOR VALUE
                         3472    ; INPUT
                         3473    ;       (CX) HAS CURSOR VALUE CH-START LINE, CL-STOP LINE
                         3474    ; OUTPUT
                         3475    ;       NONE
                         3476    ;--------------------------------------------
F1CF                     3477    SET_CTYPE       PROC    NEAR
F1CF B40A                3478            MOV     AH,10           ; 6845 REGISTER FOR CURSOR SET
F1D1 890E6000      R     3479            MOV     CURSOR_MODE,CX  ; SAVE IN DATA AREA
F1D5 E80200              3480            CALL    M16             ; OUTPUT CX REG
F1D8 EBED                3481            JMP     VIDEO_RETURN
                         3482
                         3483    ;------ THIS ROUTINE OUTPUTS THE CX REGISTER TO THE 6845 REGS NAMED IN AH
                         3484
F1DA                     3485    M16:
F1DA 8B166300      R     3486            MOV     DX,ADDR_6845    ; ADDRESS REGISTER
F1DE 8AC4                3487            MOV     AL,AH           ; GET VALUE
F1E0 EE                  3488            OUT     DX,AL           ; REGISTER SET
F1E1 42                  3489            INC     DX              ; DATA REGISTER
F1E2 8AC5                3490            MOV     AL,CH           ; DATA
F1E4 EE                  3491            OUT     DX,AL
F1E5 4A                  3492            DEC     DX
F1E6 8AC4                3493            MOV     AL,AH
F1E8 FEC0                3494            INC     AL              ; POINT TO OTHER DATA REGISTER
F1EA EE                  3495            OUT     DX,AL           ; SET FOR SECOND REGISTER
F1EB 42                  3496            INC     DX
F1EC 8AC1                3497            MOV     AL,CL           ; SECOND DATA VALUE
F1EE EE                  3498            OUT     DX,AL
F1EF C3                  3499            RET                     ; ALL DONE
                         3500    SET_CTYPE       ENDP
                         3501    ;--------------------------------------------
                         3502    ; SET_CPOS
                         3503    ;       THIS ROUTINE SETS THE CURRENT CURSOR POSITION TO THE
                         3504    ;       NEW X-Y VALUES PASSED
                         3505    ; INPUT
                         3506    ;       DX - ROW,COLUMN OF NEW CURSOR
                         3507    ;       BH - DISPLAY PAGE OF CURSOR
                         3508    ; OUTPUT
                         3509    ;       CURSOR IS SET AT 6845 IF DISPLAY PAGE IS CURRENT DISPLAY
                         3510    ;--------------------------------------------
F1F0                     3511    SET_CPOS        PROC    NEAR
F1F0 8ACF                3512            MOV     CL,BH
F1F2 32ED                3513            XOR     CH,CH           ; ESTABLISH LOOP COUNT
F1F4 D1E1                3514            SAL     CX,1            ; WORD OFFSET
F1F6 8BF1                3515            MOV     SI,CX           ; USE INDEX REGISTER
F1F8 89945000      R     3516            MOV     [SI+OFFSET CURSOR_POSN],DX      ; SAVE THE POINTER
F1FC 383E6200      R     3517            CMP     ACTIVE_PAGE,BH
F200 7505                3518            JNZ     M17             ; SET_CPOS_RETURN
F202 8BC2                3519            MOV     AX,DX           ; GET ROW/COLUMN TO AX
F204 E80200              3520            CALL    M18             ; CURSOR_SET
F207                     3521    M17:                            ; SET_CPOS_RETURN
F207 EBBE                3522            JMP     VIDEO_RETURN
                         3523    SET_CPOS        ENDP
                         3524
                         3525    ;------ SET CURSOR POSITION, AX HAS ROW/COLUMN FOR CURSOR
                         3526
F209                     3527    M18     PROC    NEAR
F209 E87F00              3528            CALL    POSITION        ; DETERMINE LOCATION IN REGEN BUFFER
F20C 8BC8                3529            MOV     CX,AX
F20E 030E4E00      R     3530            ADD     CX,CRT_START    ; ADD IN THE START ADDRESS FOR THIS PAGE
F212 D1F9                3531            SAR     CX,1            ; DIVIDE BY 2 FOR CHAR ONLY COUNT
F214 B40E                3532            MOV     AH,14           ; REGISTER NUMBER FOR CURSOR
```

```
LOC OBJ                 LINE    SOURCE

F216 E8C1FF             3533            CALL    M16             ; OUTPUT THE VALUE TO THE 6845
F219 C3                 3534            RET
                        3535    M18     ENDP
                        3536    ;--------------------------------------------
                        3537    ; READ_CURSOR
                        3538    ;       THIS ROUTINE READS THE CURRENT CURSOR VALUE FROM THE
                        3539    ;       6845, FORMATS IT, AND SENDS IT BACK TO THE CALLER
                        3540    ; INPUT
                        3541    ;       BH - PAGE OF CURSOR
                        3542    ; OUTPUT
                        3543    ;       DX - ROW, COLUMN OF THE CURRENT CURSOR POSITION
                        3544    ;       CX - CURRENT CURSOR MODE
                        3545    ;--------------------------------------------
F21A                    3546    READ_CURSOR     PROC    NEAR
F21A 8ADF               3547            MOV     BL,BH
F21C 32FF               3548            XOR     BH,BH
F21E D1E3               3549            SAL     BX,1            ; WORD OFFSET
F220 8B975000       R   3550            MOV     DX,[BX+OFFSET CURSOR_POSN]
F224 8B0E6000       R   3551            MOV     CX,CURSOR_MODE
F228 5F                 3552            POP     DI
F229 5E                 3553            POP     SI
F22A 5B                 3554            POP     BX
F22B 58                 3555            POP     AX      ; DISCARD SAVED CX AND DX
F22C 58                 3556            POP     AX
F22D 1F                 3557            POP     DS
F22E 07                 3558            POP     ES
F22F CF                 3559            IRET
                        3560    READ_CURSOR     ENDP
                        3561    ;--------------------------------------------
                        3562    ; ACT_DISP_PAGE
                        3563    ;       THIS ROUTINE SETS THE ACTIVE DISPLAY PAGE, ALLOWING
                        3564    ;       THE FULL USE OF THE RAM SET ASIDE FOR THE VIDEO ATTACHMENT
                        3565    ; INPUT
                        3566    ;       AL HAS THE NEW ACTIVE DISPLAY PAGE
                        3567    ; OUTPUT
                        3568    ;       THE 6845 IS RESET TO DISPLAY THAT PAGE
                        3569    ;--------------------------------------------
F230                    3570    ACT_DISP_PAGE   PROC    NEAR
F230 A26200         R   3571            MOV     ACTIVE_PAGE,AL  ; SAVE ACTIVE PAGE VALUE
F233 8B0E4C00       R   3572            MOV     CX,CRT_LEN      ; GET SAVED LENGTH OF REGEN BUFFER
F237 98                 3573            CBW                     ; CONVERT AL TO WORD
F238 50                 3574            PUSH    AX              ; SAVE PAGE VALUE
F239 F7E1               3575            MUL     CX              ; DISPLAY PAGE TIMES REGEN LENGTH
F23B A34E00         R   3576            MOV     CRT_START,AX    ; SAVE START ADDRESS FOR LATER REQUIREMENTS
F23E 8BC8               3577            MOV     CX,AX           ; START ADDRESS TO CX
F240 D1F9               3578            SAR     CX,1            ; DIVIDE BY 2 FOR 6845 HANDLING
F242 B40C               3579            MOV     AH,12           ; 6845 REGISTER FOR START ADDRESS
F244 E893FF             3580            CALL    M16
F247 5B                 3581            POP     BX              ; RECOVER PAGE VALUE
F248 D1E3               3582            SAL     BX,1            ; *2 FOR WORD OFFSET
F24A 8B875000       R   3583            MOV     AX,[BX + OFFSET CURSOR_POSN]    ; GET CURSOR FOR THIS PAGE
F24E E8B8FF             3584            CALL    M18             ; SET THE CURSOR POSITION
F251 E973FF             3585            JMP     VIDEO_RETURN
                        3586    ACT_DISP_PAGE   ENDP
                        3587    ;--------------------------------------------
                        3588    ; SET COLOR
                        3589    ;       THIS ROUTINE WILL ESTABLISH THE BACKGROUND COLOR, THE OVERSCAN COLOR,
                        3590    ;       AND THE FOREGROUND COLOR SET FOR MEDIUM RESOLUTION GRAPHICS
                        3591    ; INPUT
                        3592    ;       (BH) HAS COLOR ID
                        3593    ;               IF BH=0, THE BACKGROUND COLOR VALUE IS SET
                        3594    ;                       FROM THE LOW BITS OF BL (0-31)
                        3595    ;               IF BH=1, THE PALLETTE SELECTION IS MADE
                        3596    ;                       BASED ON THE LOW BIT OF BL:
                        3597    ;                               0 = GREEN, RED, YELLOW FOR COLORS 1,2,3
                        3598    ;                               1 = BLUE, CYAN, MAGENTA FOR COLORS 1,2,3
                        3599    ;       (BL) HAS THE COLOR VALUE TO BE USED
                        3600    ; OUTPUT
                        3601    ;       THE COLOR SELECTION IS UPDATED
                        3602    ;--------------------------------------------
F254                    3603    SET_COLOR       PROC    NEAR
F254 8B166300       R   3604            MOV     DX,ADDR_6845    ; I/O PORT FOR PALETTE
F258 83C205             3605            ADD     DX,5            ; OVERSCAN PORT
F25B A06600         R   3606            MOV     AL,CRT_PALLETTE ; GET THE CURRENT PALLETTE VALUE
F25E 0AFF               3607            OR      BH,BH           ; IS THIS COLOR 0?
```

```
LOC OBJ                 LINE    SOURCE


F260 750E               3608            JNZ     M20             ; OUTPUT COLOR 1
                        3609
                        3610    ;------ HANDLE COLOR 0 BY SETTING THE BACKGROUND COLOR
                        3611
F262 24E0               3612            AND     AL,0E0H         ; TURN OFF LOW 5 BITS OF CURRENT
F264 80E31F             3613            AND     BL,01FH         ; TURN OFF HIGH 3 BITS OF INPUT VALUE
F267 0AC3               3614            OR      AL,BL           ; PUT VALUE INTO REGISTER
F269                    3615    M19:                            ; OUTPUT THE PALLETTE
F269 EE                 3616            OUT     DX,AL           ;output color selection to 3d9 port
F26A A26600         R   3617            MOV     CRT_PALLETTE,AL ; SAVE THE COLOR VALUE
F26D E957FF             3618            JMP     VIDEO_RETURN
                        3619
                        3620    ;------ HANDLE COLOR 1 BY SELECTING THE PALLETTE TO BE USED
                        3621
F270                    3622    M20:
F270 24DF               3623            AND     AL,0DFH         ; TURN OFF PALLETTE SELECT BIT
F272 D0EB               3624            SHR     BL,1            ; TEST THE LOW ORDER BIT OF BL
F274 73F3               3625            JNC     M19             ; ALREADY DONE
F276 0C20               3626            OR      AL,20H          ; TURN ON PALLETTE SELECT BIT
F278 EBEF               3627            JMP     M19             ; GO DO IT
                        3628    SET_COLOR       ENDP
                        3629    ;----------------------------------------------
                        3630    ;VIDEO STATE
                        3631    ;  RETURNS THE CURRENT VIDEO STATE IN AX
                        3632    ;  AH = NUMBER OF COLUMNS ON THE SCREEN
                        3633    ;  AL = CURRENT VIDEO MODE
                        3634    ;  BH = CURRENT ACTIVE PAGE
                        3635    ;----------------------------------------------
F27A                    3636    VIDEO_STATE     PROC    NEAR
F27A 8A264A00       R   3637            MOV     AH,BYTE PTR CRT_COLS    ; GET NUMBER OF COLUMNS
F27E A04900         R   3638            MOV     AL,CRT_MODE             ; CURRENT MODE
F281 8A3E6200       R   3639            MOV     BH,ACTIVE_PAGE          ; GET CURRENT ACTIVE PAGE
F285 5F                 3640            POP     DI                      ; RECOVER REGISTERS
F286 5E                 3641            POP     SI                      ;
F287 59                 3642            POP     CX                      ; DISCARD SAVED BX
F288 E93FFF             3643            JMP     M15                     ; RETURN TO CALLER
                        3644    VIDEO_STATE     ENDP
                        3645    ;-----------------------------------
                        3646    ; POSITION
                        3647    ;       THIS SERVICE ROUTINE CALCULATES THE REGEN BUFFER ADDRESS
                        3648    ;       OF A CHARACTER IN THE ALPHA MODE
                        3649    ;INPUT
                        3650    ;       AX = ROW, COLUMN POSITION
                        3651    ; OUTPUT
                        3652    ;       AX = OFFSET OF CHAR POSITION IN REGEN BUFFER
                        3653    ;-----------------------------------
F28B                    3654    POSITION        PROC    NEAR
F28B 53                 3655            PUSH    BX              ; SAVE REGISTER
F28C 8BD8               3656            MOV     BX,AX
F28E 8AC4               3657            MOV     AL,AH           ; ROWS TO AL
F290 F6264A00       R   3658            MUL     BYTE PTR CRT_COLS       ; DETERMINE BYTES TO ROW
F294 32FF               3659            XOR     BH,BH
F296 03C3               3660            ADD     AX,BX           ; ADD IN COLUMN VALUE
F298 D1E0               3661            SAL     AX,1            ; * 2 FOR ATTRIBUTE BYTES
F29A 5B                 3662            POP     BX
F29B C3                 3663            RET
                        3664    POSITION        ENDP
                        3665    ;----------------------------------------
                        3666    ; SCROLL UP
                        3667    ;       THIS ROUTINE MOVES A BLOCK OF CHARACTERS UP
                        3668    ;       ON THE SCREEN
                        3669    ;INPUT
                        3670    ;       (AH) = CURRENT CRT MODE
                        3671    ;       (AL) = NUMBER OF ROWS TO SCROLL
                        3672    ;       (CX) = ROW/COLUMN OF UPPER LEFT CORNER
                        3673    ;       (DX) = ROW/COLUMN OF LOWER RIGHT CORNER
                        3674    ;       (BH) = ATTRIBUTE TO BE USED ON BLANKED LINE
                        3675    ;       (DS) = DATA SEGMENT
                        3676    ;       (ES) = REGEN BUFFER SEGMENT
                        3677    ;OUTPUT
                        3678    ;       NONE -- THE REGEN BUFFER IS MODIFIED
                        3679    ;----------------------------------------------
                        3680            ASSUME  CS:CODE,DS:DATA,ES:DATA
F29C                    3681    SCROLL_UP       PROC    NEAR
F29C 8AD8               3682            MOV     BL,AL           ; SAVE LINE COUNT IN BL
F29E 80FC04             3683            CMP     AH,4            ; TEST FOR GRAPHICS MODE
```

```
LOC OBJ                 LINE    SOURCE

F2A1 7208               3684            JC      N1              ; HANDLE SEPARATELY
F2A3 80FC07             3685            CMP     AH,7            ; TEST FOR BW CARD
F2A6 7403               3686            JE      N1
F2A8 E9F301             3687            JMP     GRAPHICS_UP
F2AB                    3688    N1:                             ; UP_CONTINUE
F2AB 53                 3689            PUSH    BX              ; SAVE FILL ATTRIBUTE IN BH
F2AC 8BC1               3690            MOV     AX,CX           ; UPPER LEFT POSITION
F2AE E83900             3691            CALL    SCROLL_POSITION ; DO SETUP FOR SCROLL
F2B1 7433               3692            JZ      N7              ; BLANK_FIELD
F2B3 03F0               3693            ADD     SI,AX           ; FROM ADDRESS
F2B5 8AE6               3694            MOV     AH,DH           ; # ROWS IN BLOCK
F2B7 2AE3               3695            SUB     AH,BL           ; # ROWS TO BE MOVED
F2B9                    3696    N2:                             ; ROW_LOOP
F2B9 E87500             3697            CALL    N10             ; MOVE ONE ROW
F2BC 03F5               3698            ADD     SI,BP
F2BE 03FD               3699            ADD     DI,BP           ; POINT TO NEXT LINE IN BLOCK
F2C0 FECC               3700            DEC     AH              ; COUNT OF LINES TO MOVE
F2C2 75F5               3701            JNZ     N2              ; ROW_LOOP
F2C4                    3702    N3:                             ; CLEAR_ENTRY
F2C4 58                 3703            POP     AX              ; RECOVER ATTRIBUTE IN AH
F2C5 B020               3704            MOV     AL,' '          ; FILL WITH BLANKS
F2C7                    3705    N4:                             ; CLEAR_LOOP
F2C7 E87000             3706            CALL    N11             ; CLEAR THE ROW
F2CA 03FD               3707            ADD     DI,BP           ; POINT TO NEXT LINE
F2CC FECB               3708            DEC     BL              ; COUNTER OF LINES TO SCROLL
F2CE 75F7               3709            JNZ     N4              ; CLEAR_LOOP
F2D0                    3710    N5:                             ; SCROLL_END
F2D0 B84000         R   3711            MOV     AX,DATA         ; GET LOCATION
F2D3 8ED8               3712            MOV     DS,AX
F2D5 803E490007     R   3713            CMP     CRT_MODE,7      ; IS THIS THE BLACK AND WHITE CARD
F2DA 7407               3714            JE      N6              ; IF SO, SKIP THE MODE RESET
F2DC A06500         R   3715            MOV     AL,CRT_MODE_SET ; GET THE VALUE OF THE MODE SET
F2DF BAD803             3716            MOV     DX,03D8H        ; ALWAYS SET COLOR CARD PORT
F2E2 EE                 3717            OUT     DX,AL
F2E3                    3718    N6:                             ; VIDEO_RET_HERE
F2E3 E9E1FE             3719            JMP     VIDEO_RETURN
F2E6                    3720    N7:                             ; BLANK_FIELD
F2E6 8ADE               3721            MOV     BL,DH           ; GET ROW COUNT
F2E8 EBDA               3722            JMP     N3              ; GO CLEAR THAT AREA
                        3723    SCROLL_UP       ENDP
                        3724
                        3725    ;----- HANDLE COMMON SCROLL SET UP HERE
                        3726
F2EA                    3727    SCROLL_POSITION PROC    NEAR
F2EA 803E490002     R   3728            CMP     CRT_MODE,2      ; TEST FOR SPECIAL CASE HERE
F2EF 7219               3729            JB      N9              ; HAVE TO HANDLE 80X25 SEPARATELY
F2F1 803E490003     R   3730            CMP     CRT_MODE,3
F2F6 7712               3731            JA      N9
                        3732
                        3733    ;------ 80X25 COLOR CARD SCROLL
                        3734
F2F8 52                 3735            PUSH    DX
F2F9 BADA03             3736            MOV     DX,3DAH         ; GUARANTEED TO BE COLOR CARD HERE
F2FC 50                 3737            PUSH    AX
F2FD                    3738    N8:                             ; WAIT_DISP_ENABLE
F2FD EC                 3739            IN      AL,DX           ; GET PORT
F2FE A808               3740            TEST    AL,8            ; WAIT FOR VERTICAL RETRACE
F300 74FB               3741            JZ      N8              ; WAIT_DISP_ENABLE
F302 B025               3742            MOV     AL,25H
F304 BAD803             3743            MOV     DX,03D8H
F307 EE                 3744            OUT     DX,AL           ; TURN OFF VIDEO
F308 58                 3745            POP     AX              ; DURING VERTICAL RETRACE
F309 5A                 3746            POP     DX
F30A E87EFF             3747    N9:     CALL    POSITION        ; CONVERT TO REGEN POINTER
F30D 03064E00       R   3748            ADD     AX,CRT_START    ; OFFSET OF ACTIVE PAGE
F311 8BF8               3749            MOV     DI,AX           ; TO ADDRESS FOR SCROLL
F313 8BF0               3750            MOV     SI,AX           ; FROM ADDRESS FOR SCROLL
F315 2BD1               3751            SUB     DX,CX           ; DX = #ROWS, #COLS IN BLOCK
F317 FEC6               3752            INC     DH
F319 FEC2               3753            INC     DL              ; INCREMENT FOR 0 ORIGIN
F31B 32ED               3754            XOR     CH,CH           ; SET HIGH BYTE OF COUNT TO ZERO
F31D 8B2E4A00       R   3755            MOV     BP,CRT_COLS     ; GET NUMBER OF COLUMNS IN DISPLAY
F321 03ED               3756            ADD     BP,BP           ; TIMES 2 FOR ATTRIBUTE BYTE
F323 8AC3               3757            MOV     AL,BL           ; GET LINE COUNT
F325 F6264A00       R   3758            MUL     BYTE PTR CRT_COLS       ; DETERMINE OFFSET TO FROM ADDRESS
F329 03C0               3759            ADD     AX,AX           ; *2 FOR ATTRIBUTE BYTE
```

```
LOC OBJ                 LINE    SOURCE


F32B 06                 3760            PUSH    ES                      ; ESTABLISH ADDRESSING TO REGEN BUFFER
F32C 1F                 3761            POP     DS                      ;  FOR BOTH POINTERS
F32D 80FB00             3762            CMP     BL,0                    ; 0 SCROLL MEANS BLANK FIELD
F330 C3                 3763            RET                             ; RETURN WITH FLAGS SET
                        3764    SCROLL_POSITION ENDP
                        3765
                        3766    ;------ MOVE_ROW
F331                    3767    N10     PROC    NEAR
F331 8ACA               3768            MOV     CL,DL                   ; GET # OF COLS TO MOVE
F333 56                 3769            PUSH    SI
F334 57                 3770            PUSH    DI                      ; SAVE START ADDRESS
F335 F3                 3771            REP     MOVSW                   ; MOVE THAT LINE ON SCREEN
F336 A5
F337 5F                 3772            POP     DI
F338 5E                 3773            POP     SI                      ; RECOVER ADDRESSES
F339 C3                 3774            RET
                        3775    N10     ENDP
                        3776
                        3777    ;------ CLEAR_ROW
F33A                    3778    N11     PROC    NEAR
F33A 8ACA               3779            MOV     CL,DL                   ; GET # COLUMNS TO CLEAR
F33C 57                 3780            PUSH    DI
F33D F3                 3781            REP     STOSW                   ; STORE THE FILL CHARACTER
F33E AB
F33F 5F                 3782            POP     DI
F340 C3                 3783            RET
                        3784    N11     ENDP
                        3785    ;--------------------------------
                        3786    ; SCROLL_DOWN
                        3787    ;       THIS ROUTINE MOVES THE CHARACTERS WITHIN A DEFINED
                        3788    ;       BLOCK DOWN ON THE SCREEN, FILLING THE TOP LINES
                        3789    ;       WITH A DEFINED CHARACTER
                        3790    ;INPUT
                        3791    ;       (AH) = CURRENT CRT MODE
                        3792    ;       (AL) = NUMBER OF LINES TO SCROLL
                        3793    ;       (CX) = UPPER LEFT CORNER OF REGION
                        3794    ;       (DX) = LOWER RIGHT CORNER OF REGION
                        3795    ;       (BH) = FILL CHARACTER
                        3796    ;       (DS) = DATA SEGMENT
                        3797    ;       (ES) = REGEN SEGMENT
                        3798    ;OUPUT
                        3799    ;       NONE -- SCREEN IS SCROLLED
                        3800    ;--------------------------------------
F341                    3801    SCROLL_DOWN     PROC    NEAR
F341 FD                 3802            STD                             ; DIRECTION FOR SCROLL DOWN
F342 8AD8               3803            MOV     BL,AL                   ; LINE COUNT TO BL
F344 80FC04             3804            CMP     AH,4                    ; TEST FOR GRAPHICS
F347 7208               3805            JC      N12
F349 80FC07             3806            CMP     AH,7                    ; TEST FOR BW CARD
F34C 7403               3807            JE      N12
F34E E9A601             3808            JMP     GRAPHICS_DOWN
F351                    3809    N12:                                    ; CONTINUE_DOWN
F351 53                 3810            PUSH    BX                      ; SAVE ATTRIBUTE IN BH
F352 8BC2               3811            MOV     AX,DX                   ; LOWER RIGHT CORNER
F354 E893FF             3812            CALL    SCROLL_POSITION         ; GET REGEN LOCATION
F357 7420               3813            JZ      N16
F359 2BF0               3814            SUB     SI,AX                   ; SI IS FROM ADDRESS
F35B 8AE6               3815            MOV     AH,DH                   ; GET TOTAL # ROWS
F35D 2AE3               3816            SUB     AH,BL                   ; COUNT TO MOVE IN SCROLL
F35F                    3817    N13:
F35F E8CFFF             3818            CALL    N10                     ; MOVE ONE ROW
F362 2BF5               3819            SUB     SI,BP
F364 2BFD               3820            SUB     DI,BP
F366 FECC               3821            DEC     AH
F368 75F5               3822            JNZ     N13
F36A                    3823    N14:
F36A 58                 3824            POP     AX                      ; RECOVER ATTRIBUTE IN AH
F36B B020               3825            MOV     AL,' '
F36D                    3826    N15:
F36D E8CAFF             3827            CALL    N11                     ; CLEAR ONE ROW
F370 2BFD               3828            SUB     DI,BP                   ; GO TO NEXT ROW
F372 FECB               3829            DEC     BL
F374 75F7               3830            JNZ     N15
F376 E957FF             3831            JMP     N5                              ; SCROLL_END
F379                    3832    N16:
F379 8ADE               3833            MOV     BL,DH
```

```
LOC OBJ                 LINE    SOURCE

F37B EBED               3834            JMP     N14
                        3835    SCROLL_DOWN     ENDP
                        3836    ;----------------------------------------
                        3837    ; READ_AC_CURRENT
                        3838    ;       THIS ROUTINE READS THE ATTRIBUTE AND CHARACTER AT THE CURRENT
                        3839    ;       CURSOR POSITION AND RETURNS THEM TO THE CALLER
                        3840    ;INPUT
                        3841    ;       (AH) = CURRENT CRT MODE
                        3842    ;       (BH) = DISPLAY PAGE ( ALPHA MODES ONLY )
                        3843    ;       (DS) = DATA SEGMENT
                        3844    ;       (ES) = REGEN SEGMENT
                        3845    ;OUTPUT
                        3846    ;       (AL) = CHAR READ
                        3847    ;       (AH) = ATTRIBUTE READ
                        3848    ;----------------------------------------
                        3849            ASSUME  CS:CODE,DS:DATA,ES:DATA
F37D                    3850    READ_AC_CURRENT PROC    NEAR
F37D 80FC04             3851            CMP     AH,4                    ; IS THIS GRAPHICS
F380 7208               3852            JC      P1
F382 80FC07             3853            CMP     AH,7                    ; IS THIS BW CARD
F385 7403               3854            JE      P1
F387 E9A902             3855            JMP     GRAPHICS_READ
F38A                    3856    P1:                                     ; READ_AC_CONTINUE
F38A E81A00             3857            CALL    FIND_POSITION
F38D 8BF3               3858            MOV     SI,BX                   ; ESTABLISH ADDRESSING IN SI
                        3859
                        3860    ;------ WAIT FOR HORIZONTAL RETRACE
                        3861
F38F 8B166300       R   3862            MOV     DX,ADDR_6845            ; GET BASE ADDRESS
F393 83C206             3863            ADD     DX,6                    ; POINT AT STATUS PORT
F396 06                 3864            PUSH    ES                      ;
F397 1F                 3865            POP     DS                      ; GET SEGMENT FOR QUICK ACCESS
F398                    3866    P2:                                     ; WAIT FOR RETRACE LOW
F398 EC                 3867            IN      AL,DX                   ; GET STATUS
F399 A801               3868            TEST    AL,1                    ; IS HORZ RETRACE LOW
F39B 75FB               3869            JNZ     P2                      ; WAIT UNTIL IT IS
F39D FA                 3870            CLI                             ; NO MORE INTERRUPTS
F39E                    3871    P3:                                     ; WAIT FOR RETRACE HIGH
F39E EC                 3872            IN      AL,DX                   ; GET STATUS
F39F A801               3873            TEST    AL,1                    ; IS IT HIGH
F3A1 74FB               3874            JZ      P3                      ; WAIT UNTIL IT IS
F3A3 AD                 3875            LODSW                           ; GET THE CHAR/ATTR
F3A4 E920FE             3876            JMP     VIDEO_RETURN
                        3877    READ_AC_CURRENT ENDP
                        3878
F3A7                    3879    FIND_POSITION   PROC    NEAR
F3A7 8ACF               3880            MOV     CL,BH                   ; DISPLAY PAGE TO CX
F3A9 32ED               3881            XOR     CH,CH
F3AB 8BF1               3882            MOV     SI,CX                   ; MOVE TO SI FOR INDEX
F3AD D1E6               3883            SAL     SI,1                    ; * 2 FOR WORD OFFSET
F3AF 8B845000       R   3884            MOV     AX,[SI+ OFFSET CURSOR_POSN]     ; GET ROW/COLUMN OF THAT PAGE
F3B3 33DB               3885            XOR     BX,BX                   ; SET START ADDRESS TO ZERO
F3B5 E306               3886            JCXZ    P5                      ; NO_PAGE
F3B7                    3887    P4:                                     ; PAGE_LOOP
F3B7 031E4C00       R   3888            ADD     BX,CRT_LEN              ; LENGTH OF BUFFER
F3BB E2FA               3889            LOOP    P4
F3BD                    3890    P5:                                     ; NO_PAGE
F3BD E8CBFE             3891            CALL    POSITION                ; DETERMINE LOCATION IN REGEN
F3C0 03D8               3892            ADD     BX,AX                   ; ADD TO START OF REGEN
F3C2 C3                 3893            RET
                        3894    FIND_POSITION   ENDP
                        3895    ;----------------------------------------
                        3896    ;WRITE_AC_CURRENT
                        3897    ;       THIS ROUTINE WRITES THE ATTRIBUTE AND CHARACTER AT
                        3898    ;       THE CURRENT CURSOR POSITION
                        3899    ;INPUT
                        3900    ;       (AH) = CURRENT CRT MODE
                        3901    ;       (BH) = DISPLAY PAGE
                        3902    ;       (CX) = COUNT OF CHARACTERS TO WRITE
                        3903    ;       (AL) = CHAR TO WRITE
                        3904    ;       (BL) = ATTRIBUTE OF CHAR TO WRITE
                        3905    ;       (DS) = DATA SEGMENT
                        3906    ;       (ES) = REGEN SEGMENT
                        3907    ;OUTPUT
                        3908    ;       NONE
                        3909    ;----------------------------------------------
```

```
LOC OBJ                 LINE    SOURCE


F3C3                    3910    WRITE_AC_CURRENT        PROC    NEAR
F3C3 80FC04             3911            CMP     AH,4                    ; IS THIS GRAPHICS
F3C6 7208               3912            JC      P6
F3C8 80FC07             3913            CMP     AH,7                    ; IS THIS BW CARD
F3CB 7403               3914            JE      P6
F3CD E9B101             3915            JMP     GRAPHICS_WRITE
F3D0                    3916    P6:                                     ; WRITE_AC_CONTINUE
F3D0 8AE3               3917            MOV     AH,BL                   ; GET ATTRIBUTE TO AH
F3D2 50                 3918            PUSH    AX                      ; SAVE ON STACK
F3D3 51                 3919            PUSH    CX                      ; SAVE WRITE COUNT
F3D4 E8D0FF             3920            CALL    FIND_POSITION
F3D7 8BFB               3921            MOV     DI,BX                   ; ADDRESS TO DI REGISTER
F3D9 59                 3922            POP     CX                      ; WRITE COUNT
F3DA 5B                 3923            POP     BX                      ; CHARACTER IN BX REG
F3DB                    3924    P7:                                     ; WRITE_LOOP
                        3925
                        3926    ;------ WAIT FOR HORIZONTAL RETRACE
                        3927
F3DB 8B166300       R   3928            MOV     DX,ADDR_6845            ; GET BASE ADDRESS
F3DF 83C206             3929            ADD     DX,6                    ; POINT AT STATUS PORT
F3E2                    3930    P8:
F3E2 EC                 3931            IN      AL,DX                   ; GET STATUS
F3E3 A801               3932            TEST    AL,1                    ; IS IT LOW
F3E5 75FB               3933            JNZ     P8                      ; WAIT UNTIL IT IS
F3E7 FA                 3934            CLI                             ; NO MORE INTERRUPTS
F3E8                    3935    P9:
F3E8 EC                 3936            IN      AL,DX                   ; GET STATUS
F3E9 A801               3937            TEST    AL,1                    ; IS IT HIGH
F3EB 74FB               3938            JZ      P9                      ; WAIT UNTIL IT IS
F3ED 8BC3               3939            MOV     AX,BX                   ; RECOVER THE CHAR/ATTR
F3EF AB                 3940            STOSW                           ; PUT THE CHAR/ATTR
F3F0 FB                 3941            STI                             ; INTERRUPTS BACK ON
F3F1 E2E8               3942            LOOP    P7                      ;  AS MANY TIMES AS REQUESTED
F3F3 E9D1FD             3943            JMP     VIDEO_RETURN
                        3944    WRITE_AC_CURRENT        ENDP
                        3945    ;----------------------------------------
                        3946    ;WRITE_C_CURRENT
                        3947    ;       THIS ROUTINE WRITES THE CHARACTER AT
                        3948    ;       THE CURRENT CURSOR POSITION, ATTRIBUTE UNCHANGED
                        3949    ;INPUT
                        3950    ;       (AH) = CURRENT CRT MODE
                        3951    ;       (BH) = DISPLAY PAGE
                        3952    ;       (CX) = COUNT OF CHARACTERS TO WRITE
                        3953    ;       (AL) = CHAR TO WRITE
                        3954    ;       (DS) = DATA SEGMENT
                        3955    ;       (ES) = REGEN SEGMENT
                        3956    ;OUTPUT
                        3957    ;       NONE
                        3958    ;----------------------------------------------
F3F6                    3959    WRITE_C_CURRENT PROC    NEAR
F3F6 80FC04             3960            CMP     AH,4                    ; IS THIS GRAPHICS
F3F9 7208               3961            JC      P10
F3FB 80FC07             3962            CMP     AH,7                    ; IS THIS BW CARD
F3FE 7403               3963            JE      P10
F400 E97E01             3964            JMP     GRAPHICS_WRITE
F403                    3965    P10:
F403 50                 3966            PUSH    AX                      ; SAVE ON STACK
F404 51                 3967            PUSH    CX                      ; SAVE WRITE COUNT
F405 E89FFF             3968            CALL    FIND_POSITION
F408 8BFB               3969            MOV     DI,BX                   ; ADDRESS TO DI
F40A 59                 3970            POP     CX                      ; WRITE COUNT
F40B 5B                 3971            POP     BX                      ; BL HAS CHAR TO WRITE
F40C                    3972    P11:                                    ; WRITE_LOOP
                        3973
                        3974    ;------ WAIT FOR HORIZONTAL RETRACE
                        3975
F40C 8B166300       R   3976            MOV     DX,ADDR_6845            ; GET BASE ADDRESS
F410 83C206             3977            ADD     DX,6                    ; POINT AT STATUS PORT
F413                    3978    P12:
F413 EC                 3979            IN      AL,DX                   ; GET STATUS
F414 A801               3980            TEST    AL,1                    ; IS IT LOW
F416 75FB               3981            JNZ     P12                     ; WAIT UNTIL IT IS
F418 FA                 3982            CLI                             ; NO MORE INTERRUPTS
F419                    3983    P13:
F419 EC                 3984            IN      AL,DX                   ; GET STATUS
F41A A801               3985            TEST    AL,1                    ; IS IT HIGH
```

```
LOC OBJ                 LINE    SOURCE

F41C 74FB               3986            JZ      P13                     ; WAIT UNTIL IT IS
F41E 8AC3               3987            MOV     AL,BL                   ; RECOVER CHAR
F420 AA                 3988            STOSB                           ; PUT THE CHAR/ATTR
F421 47                 3989            INC     DI                      ; BUMP POINTER PAST ATTRIBUTE
F422 E2E8               3990            LOOP    P11                     ;  AS MANY TIMES AS REQUESTED
F424 E9A0FD             3991            JMP     VIDEO_RETURN
                        3992    WRITE_C_CURRENT ENDP
                        3993    ;-----------------------------------------------
                        3994    ; READ DOT  -- WRITE DOT
                        3995    ; THESE ROUTINES WILL WRITE A DOT, OR READ THE
                        3996    ;  DOT AT THE INDICATED LOCATION
                        3997    ; ENTRY --
                        3998    ;   DX = ROW (0-199)    (THE ACTUAL VALUE DEPENDS ON THE MODE)
                        3999    ;   CX = COLUMN ( 0-639) ( THE VALUES ARE NOT RANGE CHECKED )
                        4000    ;   AL = DOT VALUE TO WRITE (1,2 OR 4 BITS DEPENDING ON MODE,
                        4001    ;       REQ'D FOR WRITE DOT ONLY, RIGHT JUSTIFIED)
                        4002    ;       BIT 7 OF AL = 1 INDICATES XOR THE VALUE INTO THE LOCATION
                        4003    ;   DS = DATA SEGMENT
                        4004    ;   ES = REGEN SEGMENT
                        4005    ;
                        4006    ; EXIT
                        4007    ;       AL = DOT VALUE READ, RIGHT JUSTIFIED, READ ONLY
                        4008    ;-----------------------------------------------
                        4009            ASSUME  CS:CODE,DS:DATA,ES:DATA
F427                    4010    READ_DOT        PROC    NEAR
F427 E83100             4011            CALL    R3              ; DETERMINE BYTE POSITION OF DOT
F42A 268A04             4012            MOV     AL,ES:[SI]              ; GET THE BYTE
F42D 22C4               4013            AND     AL,AH           ; MASK OFF THE OTHER BITS IN THE BYTE
F42F D2E0               4014            SHL     AL,CL           ; LEFT JUSTIFY THE VALUE
F431 8ACE               4015            MOV     CL,DH           ; GET NUMBER OF BITS IN RESULT
F433 D2C0               4016            ROL     AL,CL           ; RIGHT JUSTIFY THE RESULT
F435 E98FFD             4017            JMP     VIDEO_RETURN    ; RETURN FROM VIDEO IO
                        4018    READ_DOT        ENDP
                        4019
F438                    4020    WRITE_DOT       PROC    NEAR
F438 50                 4021            PUSH    AX              ; SAVE DOT VALUE
F439 50                 4022            PUSH    AX              ;  TWICE
F43A E81E00             4023            CALL    R3              ; DETERMINE BYTE POSITION OF THE DOT
F43D D2E8               4024            SHR     AL,CL           ; SHIFT TO SET UP THE BITS FOR OUTPUT
F43F 22C4               4025            AND     AL,AH           ; STRIP OFF THE OTHER BITS
F441 268A0C             4026            MOV     CL,ES:[SI]      ; GET THE CURRENT BYTE
F444 5B                 4027            POP     BX              ; RECOVER XOR FLAG
F445 F6C380             4028            TEST    BL,80H          ; IS IT ON
F448 750D               4029            JNZ     R2              ; YES, XOR THE DOT
F44A F6D4               4030            NOT     AH              ; SET THE MASK TO REMOVE THE INDICATED BITS
F44C 22CC               4031            AND     CL,AH
F44E 0AC1               4032            OR      AL,CL           ; OR IN THE NEW VALUE OF THOSE BITS
F450                    4033    R1:                             ; FINISH_DOT
F450 268804             4034            MOV     ES:[SI],AL      ; RESTORE THE BYTE IN MEMORY
F453 58                 4035            POP     AX
F454 E970FD             4036            JMP     VIDEO_RETURN    ; RETURN FROM VIDEO IO
F457                    4037    R2:                             ; XOR_DOT
F457 32C1               4038            XOR     AL,CL           ; EXCLUSIVE OR THE DOTS
F459 EBF5               4039            JMP     R1              ; FINISH UP THE WRITING
                        4040    WRITE_DOT       ENDP
                        4041    ;-----------------------------------------------
                        4042    ; THIS SUBROUTINE DETERMINES THE REGEN BYTE LOCATION OF THE
                        4043    ; INDICATED ROW COLUMN VALUE IN GRAPHICS MODE.
                        4044    ; ENTRY --
                        4045    ;  DX = ROW VALUE (0-199)
                        4046    ;  CX = COLUMN VALUE (0-639)
                        4047    ; EXIT --
                        4048    ;  SI = OFFSET INTO REGEN BUFFER FOR BYTE OF INTEREST
                        4049    ;  AH = MASK TO STRIP OFF THE BITS OF INTEREST
                        4050    ;  CL = BITS TO SHIFT TO RIGHT JUSTIFY THE MASK IN AH
                        4051    ;  DH = # BITS IN RESULT
                        4052    ;-----------------------------------------------
F45B                    4053    R3      PROC    NEAR
F45B 53                 4054            PUSH    BX              ; SAVE BX DURING OPERATION
F45C 50                 4055            PUSH    AX              ; WILL SAVE AL DURING OPERATION
                        4056
                        4057    ;------ DETERMINE 1ST BYTE IN IDICATED ROW BY MULTIPLYING ROW VALUE BY 40
                        4058    ;------ ( LOW BIT OF ROW DETERMINES EVEN/ODD, 80 BYTES/ROW
                        4059
F45D B028               4060            MOV     AL,40
F45F 52                 4061            PUSH    DX              ; SAVE ROW VALUE
```

```
LOC OBJ                  LINE    SOURCE

F460 80E2FE              4062            AND     DL,0FEH         ; STRIP OFF ODD/EVEN BIT
F463 F6E2                4063            MUL     DL              ; AX HAS ADDRESS OF 1ST BYTE OF INDICATED ROW
F465 5A                  4064            POP     DX              ; RECOVER IT
F466 F6C201              4065            TEST    DL,1            ; TEST FOR EVEN/ODD
F469 7403                4066            JZ      R4              ; JUMP IF EVEN ROW
F46B 050020              4067            ADD     AX,2000H        ; OFFSET TO LOCATION OF ODD ROWS
F46E                     4068    R4:                             ; EVEN_ROW
F46E 8BF0                4069            MOV     SI,AX           ; MOVE POINTER TO SI
F470 58                  4070            POP     AX              ; RECOVER AL VALUE
F471 8BD1                4071            MOV     DX,CX           ; COLUMN VALUE TO DX
                         4072
                         4073    ;------ DETERMINE GRAPHICS MODE CURRENTLY IN EFFECT
                         4074
                         4075    ; SET UP THE REGISTERS ACCORDING TO THE MODE
                         4076    ;  CH = MASK FOR LOW OF COLUMN ADDRESS ( 7/3 FOR HIGH/MED RES)
                         4077    ;  CL = # OF ADDRESS BITS IN COLUMN VALUE ( 3/2 FOR H/M)
                         4078    ;  BL = MASK TO SELECT BITS FROM POINTED BYTE (80H/C0H FOR H/M)
                         4079    ;  BH = NUMBER OF VALID BITS IN POINTED BYTE ( 1/2 FOR H/M)
                         4080
F473 BBC002              4081            MOV     BX,2C0H
F476 B90203              4082            MOV     CX,302H         ; SET PARMS FOR MED RES
F479 803E490006    R     4083            CMP     CRT_MODE,6
F47E 7206                4084            JC      R5              ; HANDLE IF MED ARES
F480 BB8001              4085            MOV     BX,180H
F483 B90307              4086            MOV     CX,703H         ; SET PARMS FOR HIGH RES
                         4087
                         4088    ;------ DETERMINE BIT OFFSET IN BYTE FROM COLUMN MASK
F486                     4089    R5:
F486 22EA                4090            AND     CH,DL           ; ADDRESS OF PEL WITHIN BYTE TO CH
                         4091
                         4092    ;------ DETERMINE BYTE OFFSET FOR THIS LOCATION IN COLUMN
                         4093
F488 D3EA                4094            SHR     DX,CL           ; SHIFT BY CORRECT AMOUNT
F48A 03F2                4095            ADD     SI,DX           ; INCREMENT THE POINTER
F48C 8AF7                4096            MOV     DH,BH           ; GET THE # OF BITS IN RESULT TO DH
                         4097
                         4098    ;------ MULTIPLY BH (VALID BITS IN BYTE) BY CH (BIT OFFSET)
                         4099
F48E 2AC9                4100            SUB     CL,CL           ; ZERO INTO STORAGE LOCATION
F490                     4101    R6:
F490 D0C8                4102            ROR     AL,1            ; LEFT JUSTIFY THE VALUE IN AL (FOR WRITE)
F492 02CD                4103            ADD     CL,CH           ; ADD IN THE BIT OFFSET VALUE
F494 FECF                4104            DEC     BH              ; LOOP CONTROL
F496 75F8                4105            JNZ     R6              ; ON EXIT, CL HAS SHIFT COUNT TO RESTORE BITS
F498 8AE3                4106            MOV     AH,BL           ; GET MASK TO AH
F49A D2EC                4107            SHR     AH,CL           ; MOVE THE MASK TO CORRECT LOCATION
F49C 5B                  4108            POP     BX              ; RECOVER REG
F49D C3                  4109            RET                     ; RETURN WITH EVERYTHING SET UP
                         4110    R3      ENDP
                         4111    ;-----------------------------------------------
                         4112    ; SCROLL UP
                         4113    ;  THIS ROUTINE SCROLLS UP THE INFORMATION ON THE CRT
                         4114    ; ENTRY --
                         4115    ;  CH,CL = UPPER LEFT CORNER OF REGION TO SCROLL
                         4116    ;  DH,DL = LOWER RIGHT CORNER OF REGION TO SCROLL
                         4117    ;   BOTH OF THE ABOVE ARE IN CHARACTER POSITIONS
                         4118    ;  BH = FILL VALUE FOR BLANKED LINES
                         4119    ;  AL = # LINES TO SCROLL (AL=0 MEANS BLANK THE ENTIRE FIELD)
                         4120    ;  DS = DATA SEGMENT
                         4121    ;  ES = REGEN SEGMENT
                         4122    ; EXIT --
                         4123    ;  NOTHING, THE SCREEN IS SCROLLED
                         4124    ;-----------------------------------------------
F49E                     4125    GRAPHICS_UP     PROC    NEAR
F49E 8AD8                4126            MOV     BL,AL   ; SAVE LINE COUNT IN BL
F4A0 8BC1                4127            MOV     AX,CX   ; GET UPPER LEFT POSITION INTO AX REG
                         4128
                         4129    ;------ USE CHARACTER SUBROUTINE FOR POSITIONING
                         4130    ;------ ADDRESS RETURNED IS MULTIPLIED BY 2 FROM CORRECT VALUE
                         4131
F4A2 E86A02              4132            CALL    GRAPH_POSN
F4A5 8BF8                4133            MOV     DI,AX           ; SAVE RESULT AS DESTINATION ADDRESS
                         4134
                         4135    ;------ DETERMINE SIZE OF WINDOW
                         4136
```

```
LOC OBJ                 LINE    SOURCE

F4A7 2BD1               4137            SUB     DX,CX
F4A9 81C20101           4138            ADD     DX,101H         ; ADJUST VALUES
F4AD D0E6               4139            SAL     DH,1            ; MULTIPLY # ROWS BY 4 SINCE 8 VERT DOTS/CHAR
F4AF D0E6               4140            SAL     DH,1            ;  AND EVEN/ODD ROWS
                        4141
                        4142    ;------ DETERMINE CRT MODE
                        4143
F4B1 803E490006     R   4144            CMP     CRT_MODE,6      ; TEST FOR MEDIUM RES
F4B6 7304               4145            JNC     R7              ; FIND_SOURCE
                        4146
                        4147    ;------ MEDIUM RES UP
F4B8 D0E2               4148            SAL     DL,1            ; # COLUMNS * 2, SINCE 2 BYTES/CHAR
F4BA D1E7               4149            SAL     DI,1            ; OFFSET *2 SINCE 2 BYTES/CHAR
                        4150
                        4151    ;------ DETERMINE THE SOURCE ADDRESS IN THE BUFFER
F4BC                    4152    R7:                             ; FIND_SOURCE
F4BC 06                 4153            PUSH    ES              ; GET SEGMENTS BOTH POINTING TO REGEN
F4BD 1F                 4154            POP     DS
F4BE 2AED               4155            SUB     CH,CH           ; ZERO TO HIGH OF COUNT REG
F4C0 D0E3               4156            SAL     BL,1            ; MULTIPLY NUMBER OF LINES BY 4
F4C2 D0E3               4157            SAL     BL,1
F4C4 742D               4158            JZ      R11             ; IF ZERO, THEN BLANK ENTIRE FIELD
F4C6 8AC3               4159            MOV     AL,BL           ; GET NUMBER OF LINES IN AL
F4C8 B450               4160            MOV     AH,80           ; 80 BYTES/ROW
F4CA F6E4               4161            MUL     AH              ; DETERMINE OFFSET TO SOURCE
F4CC 8BF7               4162            MOV     SI,DI           ; SET UP SOURCE
F4CE 03F0               4163            ADD     SI,AX           ;  ADD IN OFFSET TO IT
F4D0 8AE6               4164            MOV     AH,DH           ; NUMBER OF ROWS IN FIELD
F4D2 2AE3               4165            SUB     AH,BL           ; DETERMINE NUMBER TO MOVE
                        4166
                        4167    ;------ LOOP THROUGH, MOVING ONE ROW AT A TIME, BOTH EVEN AND ODD FIELDS
F4D4                    4168    R8:                             ; ROW_LOOP
F4D4 E88000             4169            CALL    R17             ; MOVE ONE ROW
F4D7 81EEB01F           4170            SUB     SI,2000H-80     ; MOVE TO NEXT ROW
F4DB 81EFB01F           4171            SUB     DI,2000H-80
F4DF FECC               4172            DEC     AH              ; NUMBER OF ROWS TO MOVE
F4E1 75F1               4173            JNZ     R8              ; CONTINUE TILL ALL MOVED
                        4174
                        4175    ;------ FILL IN THE VACATED LINE(S)
F4E3                    4176    R9:                             ; CLEAR_ENTRY
F4E3 8AC7               4177            MOV     AL,BH           ; ATTRIBUTE TO FILL WITH
F4E5                    4178    R10:
F4E5 E88800             4179            CALL    R18             ; CLEAR THAT ROW
F4E8 81EFB01F           4180            SUB     DI,2000H-80     ; POINT TO NEXT LINE
F4EC FECB               4181            DEC     BL              ; NUMBER OF LINES TO FILL
F4EE 75F5               4182            JNZ     R10             ; CLEAR_LOOP
F4F0 E9D4FC             4183            JMP     VIDEO_RETURN    ; EVERYTHING DONE
                        4184
F4F3                    4185    R11:                            ; BLANK_FIELD
F4F3 8ADE               4186            MOV     BL,DH           ; SET BLANK COUNT TO EVERYTHING IN FIELD
F4F5 EBEC               4187            JMP     R9              ; CLEAR THE FIELD
                        4188    GRAPHICS_UP     ENDP
                        4189    ;--------------------------------------------
                        4190    ; SCROLL DOWN
                        4191    ;  THIS ROUTINE SCROLLS DOWN THE INFORMATION ON THE CRT
                        4192    ; ENTRY --
                        4193    ;  CH,CL = UPPER LEFT CORNER OF REGION TO SCROLL
                        4194    ;  DH,DL = LOWER RIGHT CORNER OF REGION TO SCROLL
                        4195    ;   BOTH OF THE ABOVE ARE IN CHARACTER POSITIONS
                        4196    ;  BH = FILL VALUE FOR BLANKED LINES
                        4197    ;  AL = # LINES TO SCROLL (AL=0 MEANS BLANK THE ENTIRE FIELD)
                        4198    ;  DS = DATA SEGMENT
                        4199    ;  ES = REGEN SEGMENT
                        4200    ; EXIT --
                        4201    ;  NOTHING, THE SCREEN IS SCROLLED
                        4202    ;--------------------------------------------
                        4203
F4F7                    4204    GRAPHICS_DOWN   PROC    NEAR
F4F7 FD                 4205            STD             ; SET DIRECTION
F4F8 8AD8               4206            MOV     BL,AL   ; SAVE LINE COUNT IN BL
F4FA 8BC2               4207            MOV     AX,DX   ; GET LOWER RIGHT POSITION INTO AX REG
                        4208
                        4209    ;------ USE CHARACTER SUBROUTINE FOR POSITIONING
                        4210    ;------ ADDRESS RETURNED IS MULTIPLIED BY 2 FROM CORRECT VALUE
                        4211
F4FC E81002             4212            CALL    GRAPH_POSN
```

```
LOC OBJ                  LINE    SOURCE

F4FF 8BF8                4213            MOV     DI,AX           ; SAVE RESULT AS DESTINATION ADDRESS
                         4214
                         4215    ;------ DETERMINE SIZE OF WINDOW
                         4216
F501 2BD1                4217            SUB     DX,CX
F503 81C20101            4218            ADD     DX,101H         ; ADJUST VALUES
F507 D0E6                4219            SAL     DH,1            ; MULTIPLY # ROWS BY 4 SINCE 8 VERT DOTS/CHAR
F509 D0E6                4220            SAL     DH,1            ;  AND EVEN/ODD ROWS
                         4221
                         4222    ;------ DETERMINE CRT MODE
                         4223
F50B 803E490006      R   4224            CMP     CRT_MODE,6      ; TEST FOR MEDIUM RES
F510 7305                4225            JNC     R12             ; FIND_SOURCE_DOWN
                         4226
                         4227    ;------ MEDIUM RES DOWN
F512 D0E2                4228            SAL     DL,1            ; # COLUMNS * 2, SINCE 2 BYTES/CHAR (OFFSET OK)
F514 D1E7                4229            SAL     DI,1            ; OFFSET *2 SINCE 2 BYTES/CHAR
F516 47                  4230            INC     DI              ; POINT TO LAST BYTE
                         4231
                         4232    ;------ DETERMINE THE SOURCE ADDRESS IN THE BUFFER
F517                     4233    R12:                            ; FIND_SOURCE_DOWN
F517 06                  4234            PUSH    ES              ; BOTH SEGMENTS TO REGEN
F518 1F                  4235            POP     DS
F519 2AED                4236            SUB     CH,CH           ; ZERO TO HIGH OF COUNT REG
F51B 81C7F000            4237            ADD     DI,240          ; POINT TO LAST ROW OF PIXELS
F51F D0E3                4238            SAL     BL,1            ; MULTIPLY NUMBER OF LINES BY 4
F521 D0E3                4239            SAL     BL,1
F523 742E                4240            JZ      R16             ; IF ZERO, THEN BLANK ENTIRE FIELD
F525 8AC3                4241            MOV     AL,BL           ; GET NUMBER OF LINES IN AL
F527 B450                4242            MOV     AH,80           ; 80 BYTES/ROW
F529 F6E4                4243            MUL     AH              ; DETERMINE OFFSET TO SOURCE
F52B 8BF7                4244            MOV     SI,DI           ; SET UP SOURCE
F52D 2BF0                4245            SUB     SI,AX           ;  SUBTRACT THE OFFSET
F52F 8AE6                4246            MOV     AH,DH           ; NUMBER OF ROWS IN FIELD
F531 2AE3                4247            SUB     AH,BL           ; DETERMINE NUMBER TO MOVE
                         4248
                         4249    ;------ LOOP THROUGH, MOVING ONE ROW AT A TIME, BOTH EVEN AND ODD FIELDS
F533                     4250    R13:                            ; ROW_LOOP_DOWN
F533 E82100              4251            CALL    R17             ; MOVE ONE ROW
F536 81EE5020            4252            SUB     SI,2000H+80     ; MOVE TO NEXT ROW
F53A 81EF5020            4253            SUB     DI,2000H+80
F53E FECC                4254            DEC     AH              ; NUMBER OF ROWS TO MOVE
F540 75F1                4255            JNZ     R13             ; CONTINUE TILL ALL MOVED
                         4256
                         4257    ;------ FILL IN THE VACATED LINE(S)
F542                     4258    R14:                            ; CLEAR_ENTRY_DOWN
F542 8AC7                4259            MOV     AL,BH           ; ATTRIBUTE TO FILL WITH
F544                     4260    R15:                            ; CLEAR_LOOP_DOWN
F544 E82900              4261            CALL    R18             ; CLEAR A ROW
F547 81EF5020            4262            SUB     DI,2000H+80     ; POINT TO NEXT LINE
F54B FECB                4263            DEC     BL              ; NUMBER OF LINES TO FILL
F54D 75F5                4264            JNZ     R15             ; CLEAR_LOOP_DOWN
F54F FC                  4265            CLD                     ; RESET THE DIRECTION FLAG
F550 E974FC              4266            JMP     VIDEO_RETURN    ; EVERYTHING DONE
                         4267
F553                     4268    R16:                            ; BLANK_FIELD_DOWN
F553 8ADE                4269            MOV     BL,DH           ; SET BLANK COUNT TO EVERYTHING IN FIELD
F555 EBEB                4270            JMP     R14             ; CLEAR THE FIELD
                         4271    GRAPHICS_DOWN   ENDP
                         4272
                         4273    ;------ ROUTINE TO MOVE ONE ROW OF INFORMATION
                         4274
F557                     4275    R17     PROC    NEAR
F557 8ACA                4276            MOV     CL,DL           ; NUMBER OF BYTES IN THE ROW
F559 56                  4277            PUSH    SI
F55A 57                  4278            PUSH    DI              ; SAVE POINTERS
F55B F3                  4279            REP     MOVSB           ; MOVE THE EVEN FIELD
F55C A4
F55D 5F                  4280            POP     DI
F55E 5E                  4281            POP     SI
F55F 81C60020            4282            ADD     SI,2000H
F563 81C70020            4283            ADD     DI,2000H                ; POINT TO THE ODD FIELD
F567 56                  4284            PUSH    SI
F568 57                  4285            PUSH    DI              ; SAVE THE POINTERS
F569 8ACA                4286            MOV     CL,DL           ; COUNT BACK
```

```
LOC OBJ                 LINE    SOURCE

F56B F3                 4287            REP     MOVSB           ; MOVE THE ODD FIELD
F56C A4
F56D 5F                 4288            POP     DI
F56E 5E                 4289            POP     SI              ; POINTERS BACK
F56F C3                 4290            RET                     ; RETURN TO CALLER
                        4291    R17     ENDP
                        4292
                        4293    ;------ CLEAR A SINGLE ROW
                        4294
F570                    4295    R18     PROC    NEAR
F570 8ACA               4296            MOV     CL,DL           ; NUMBER OF BYTES IN FIELD
F572 57                 4297            PUSH    DI              ; SAVE POINTER
F573 F3                 4298            REP     STOSB           ; STORE THE NEW VALUE
F574 AA
F575 5F                 4299            POP     DI              ; POINTER BACK
F576 81C70020           4300            ADD     DI,2000H                ; POINT TO ODD FIELD
F57A 57                 4301            PUSH    DI
F57B 8ACA               4302            MOV     CL,DL
F57D F3                 4303            REP     STOSB           ; FILL THE ODD FILELD
F57E AA
F57F 5F                 4304            POP     DI
F580 C3                 4305            RET                     ; RETURN TO CALLER
                        4306    R18     ENDP
                        4307    ;-----------------------------------------
                        4308    ; GRAPHICS WRITE
                        4309    ;  THIS ROUTINE WRITES THE ASCII CHARACTER TO THE CURRENT
                        4310    ;  POSITION ON THE SCREEN.
                        4311    ; ENTRY --
                        4312    ;  AL = CHARACTER TO WRITE
                        4313    ;  BL = COLOR ATTRIBUTE TO BE USED FOR FOREGROUND COLOR
                        4314    ;       IF BIT 7 IS SET, THE CHAR IS XOR'D INTO THE REGEN BUFFER
                        4315    ;       (0 IS USED FOR THE BACKGROUND COLOR)
                        4316    ;  CX = NUMBER OF CHARS TO WRITE
                        4317    ;  DS = DATA SEGMENT
                        4318    ;  ES = REGEN SEGMENT
                        4319    ; EXIT --
                        4320    ;  NOTHING IS RETURNED
                        4321    ;
                        4322    ; GRAPHICS READ
                        4323    ;   THIS ROUTINE READS THE ASCII CHARACTER AT THE CURRENT CURSOR
                        4324    ;   POSITION ON THE SCREEN BY MATCHING THE DOTS ON THE SCREEN TO THE
                        4325    ;   CHARACTER GENERATOR CODE POINTS
                        4326    ; ENTRY --
                        4327    ;  NONE  ( 0 IS ASSUMED AS THE BACKGROUND COLOR
                        4328    ; EXIT --
                        4329    ;  AL = CHARACTER READ AT THAT POSITION (0 RETURNED IF NONE FOUND)
                        4330    ;
                        4331    ; FOR BOTH ROUTINES, THE IMAGES USED TO FORM CHARS ARE CONTAINED IN ROM
                        4332    ;  FOR THE 1ST 128 CHARS.  TO ACCESS CHARS IN THE SECOND HALF, THE USER
                        4333    ;  MUST INITIALIZE THE VECTOR AT INTERRUPT 1FH (LOCATION 0007CH) TO
                        4334    ;  POINT TO THE USER SUPPLIED TABLE OF GRAPHIC IMAGES (8X8 BOXES).
                        4335    ;  FAILURE TO DO SO WILL CAUSE IN STRANGE RESULTS
                        4336    ;-----------------------------------------
                        4337            ASSUME  CS:CODE,DS:DATA,ES:DATA
F581                    4338    GRAPHICS_WRITE  PROC    NEAR
F581 B400               4339            MOV     AH,0            ; ZERO TO HIGH OF CODE POINT
F583 50                 4340            PUSH    AX              ; SAVE CODE POINT VALUE
                        4341
                        4342    ;------ DETERMINE POSITION IN REGEN BUFFER TO PUT CODE POINTS
                        4343
F584 E88501             4344            CALL    S26             ; FIND LOCATION IN REGEN BUFFER
F587 8BF8               4345            MOV     DI,AX           ; REGEN POINTER IN DI
                        4346
                        4347    ;------ DETERMINE REGION TO GET CODE POINTS FROM
                        4348
F589 58                 4349            POP     AX              ; RECOVER CODE POINT
F58A 3C80               4350            CMP     AL,80H          ; IS IT IN SECOND HALF
F58C 7306               4351            JAE     S1              ; YES
                        4352
                        4353    ;------ IMAGE IS IN FIRST HALF, CONTAINED IN ROM
                        4354
F58E BE6EFA             4355            MOV     SI,0FA6EH       ; OFFSET CRT_CHAR_GEN-OFFSET OF IMAGES
F591 0E                 4356            PUSH    CS                      ; SAVE SEGMENT ON STACK
F592 EB0F               4357            JMP     SHORT S2                ; DETERMINE_MODE
                        4358
```

```
LOC OBJ                  LINE   SOURCE


                         4359   ;------ IMAGE IS IN SECOND HALF, IN USER RAM
                         4360
F594                     4361   S1:                                     ; EXTEND_CHAR
F594 2C80                4362           SUB     AL,80H                  ; ZERO ORIGIN FOR SECOND HALF
F596 1E                  4363           PUSH    DS                      ; SAVE DATA POINTER
F597 2BF6                4364           SUB     SI,SI
F599 8EDE                4365           MOV     DS,SI                   ; ESTABLISH VECTOR ADDRESSING
                         4366           ASSUME  DS:ABS0
F59B C5367C00            4367           LDS     SI,EXT_PTR              ; GET THE OFFSET OF THE TABLE
F59F 8CDA                4368           MOV     DX,DS                   ; GET THE SEGMENT OF THE TABLE
                         4369           ASSUME  DS:DATA
F5A1 1F                  4370           POP     DS                      ; RECOVER DATA SEGMENT
F5A2 52                  4371           PUSH    DX                      ; SAVE TABLE SEGMENT ON STACK
                         4372
                         4373   ;------ DETERMINE GRAPHICS MODE IN OPERATION
                         4374
F5A3                     4375   S2:                                     ; DETERMINE_MODE
F5A3 D1E0                4376           SAL     AX,1                    ; MULTIPLY CODE POINT
F5A5 D1E0                4377           SAL     AX,1                    ;  VALUE BY 8
F5A7 D1E0                4378           SAL     AX,1
F5A9 03F0                4379           ADD     SI,AX                   ; SI HAS OFFSET OF DESIRED CODES
F5AB 803E490006      R   4380           CMP     CRT_MODE,6
F5B0 1F                  4381           POP     DS              ; RECOVER TABLE POINTER SEGMENT
F5B1 722C                4382           JC      S7              ; TEST FOR MEDIUM RESOLUTION MODE
                         4383
                         4384   ;------ HIGH RESOLUTION MODE
F5B3                     4385   S3:                             ; HIGH_CHAR
F5B3 57                  4386           PUSH    DI              ; SAVE REGEN POINTER
F5B4 56                  4387           PUSH    SI              ; SAVE CODE POINTER
F5B5 B604                4388           MOV     DH,4            ; NUMBER OF TIMES THROUGH LOOP
F5B7                     4389   S4:
F5B7 AC                  4390           LODSB                   ; GET BYTE FROM CODE POINTS
F5B8 F6C380              4391           TEST    BL,80H          ; SHOULD WE USE THE FUNCTION
F5BB 7516                4392           JNZ     S6              ;  TO PUT CHAR IN
F5BD AA                  4393           STOSB                   ; STORE IN REGEN BUFFER
F5BE AC                  4394           LODSB
F5BF                     4395   S5:                             ;
F5BF 268885FF1F          4396           MOV     ES:[DI+2000H-1],AL      ; STORE IN SECOND HALF
F5C4 83C74F              4397           ADD     DI,79           ; MOVE TO NEXT ROW IN REGEN
F5C7 FECE                4398           DEC     DH              ; DONE WITH LOOP
F5C9 75EC                4399           JNZ     S4
F5CB 5E                  4400           POP     SI
F5CC 5F                  4401           POP     DI              ; RECOVER REGEN POINTER
F5CD 47                  4402           INC     DI              ; POINT TO NEXT CHAR POSITION
F5CE E2E3                4403           LOOP    S3              ; MORE CHARS TO WRITE
F5D0 E9F4FB              4404           JMP     VIDEO_RETURN
                         4405
F5D3                     4406   S6:
F5D3 263205              4407           XOR     AL,ES:[DI]      ; EXCLUSIVE OR WITH CURRENT
F5D6 AA                  4408           STOSB                   ; STORE THE CODE POINT
F5D7 AC                  4409           LODSB                   ; AGAIN FOR ODD FIELD
F5D8 263285FF1F          4410           XOR     AL,ES:[DI+2000H-1]      ;
F5DD EBE0                4411           JMP     S5              ; BACK TO MAINSTREAM
                         4412
                         4413   ;------ MEDIUM RESOLUTION WRITE
F5DF                     4414   S7:                             ; MED_RES_WRITE
F5DF 8AD3                4415           MOV     DL,BL           ; SAVE HIGH COLOR BIT
F5E1 D1E7                4416           SAL     DI,1            ; OFFSET*2 SINCE 2 BYTES/CHAR
F5E3 E8D100              4417           CALL    S19             ; EXPAND BL TO FULL WORD OF COLOR
F5E6                     4418   S8:                             ; MED_CHAR
F5E6 57                  4419           PUSH    DI              ; SAVE REGEN POINTER
F5E7 56                  4420           PUSH    SI                      ; SAVE THE CODE POINTER
F5E8 B604                4421           MOV     DH,4            ; NUMBER OF LOOPS
F5EA                     4422   S9:
F5EA AC                  4423           LODSB                   ; GET CODE POINT
F5EB E8DE00              4424           CALL    S21             ; DOUBLE UP ALL THE BITS
F5EE 23C3                4425           AND     AX,BX           ; CONVERT THEM TO FOREGROUND COLOR ( 0 BACK )
F5F0 F6C280              4426           TEST    DL,80H          ; IS THIS XOR FUNCTION
F5F3 7407                4427           JZ      S10             ; NO, STORE IT IN AS IT IS
F5F5 263225              4428           XOR     AH,ES:[DI]      ; DO FUNCTION WITH HALF
F5F8 26324501            4429           XOR     AL,ES:[DI+1]    ;  AND WITH OTHER HALF
F5FC                     4430   S10:                            ;
F5FC 268825              4431           MOV     ES:[DI],AH      ; STORE FIRST BYTE
F5FF 26884501            4432           MOV     ES:[DI+1],AL    ; STORE SECOND BYTE
F603 AC                  4433           LODSB                   ; GET CODE POINT
F604 E8C500              4434           CALL    S21
```

```
LOC OBJ                 LINE    SOURCE


F607 23C3               4435            AND     AX,BX           ; CONVERT TO COLOR
F609 F6C280             4436            TEST    DL,80H          ; AGAIN, IS THIS XOR FUNCTION
F60C 740A               4437            JZ      S11             ; NO, JUST STORE THE VALUES
F60E 2632A50020         4438            XOR     AH,ES:[DI+2000H]        ; FUNCTION WITH FIRST HALF
F613 2632850120         4439            XOR     AL,ES:[DI+2001H]        ; AND WITH SECOND HALF
F618                    4440    S11:                    ;
F618 2688A50020         4441            MOV     ES:[DI+2000H],AH
F61D 2688850120         4442            MOV     ES:[DI+2000H+1],AL      ; STORE IN SECOND PORTION OF BUFFER
F622 83C750             4443            ADD     DI,80           ; POINT TO NEXT LOCATION
F625 FECE               4444            DEC     DH
F627 75C1               4445            JNZ     S9              ; KEEP GOING
F629 5E                 4446            POP     SI              ; RECOVER CODE PONTER
F62A 5F                 4447            POP     DI              ; RECOVER REGEN POINTER
F62B 83C702             4448            ADD     DI,2            ; POINT TO NEXT CHAR POSITION
F62E E2B6               4449            LOOP    S8              ; MORE TO WRITE
F630 E994FB             4450            JMP     VIDEO_RETURN
                        4451    GRAPHICS_WRITE  ENDP
                        4452    ;--------------------------------------
                        4453    ; GRAPHICS READ
                        4454    ;--------------------------------------
F633                    4455    GRAPHICS_READ   PROC    NEAR
F633 E8D600             4456            CALL    S26             ; CONVERTED TO OFFSET IN REGEN
F636 8BF0               4457            MOV     SI,AX           ; SAVE IN SI
F638 83EC08             4458            SUB     SP,8            ; ALLOCATE SPACE TO SAVE THE READ CODE POINT
F63B 8BEC               4459            MOV     BP,SP           ; POINTER TO SAVE AREA
                        4460
                        4461    ;------ DETERMINE GRAPHICS MODES
                        4462
F63D 803E490006     R   4463            CMP     CRT_MODE,6
F642 06                 4464            PUSH    ES
F643 1F                 4465            POP     DS              ; POINT TO REGEN SEGMENT
F644 721A               4466            JC      S13             ; MEDIUM RESOLUTION
                        4467
                        4468    ;------ HIGH RESOLUTION READ
                        4469
                        4470    ;------ GET VALUES FROM REGEN BUFFER AND CONVERT TO CODE POINT
F646 B604               4471            MOV     DH,4            ; NUMBER OF PASSES
F648                    4472    S12:
F648 8A04               4473            MOV     AL,[SI]         ; GET FIRST BYTE
F64A 8846 00            4474            MOV     [BP],AL         ; SAVE IN STORAGE AREA
F64D 45                 4475            INC     BP              ; NEXT LOCATION
F64E 8A840020           4476            MOV     AL,[SI+2000H]   ; GET LOWER REGION BYTE
F652 8846 00            4477            MOV     [BP],AL         ; ADJUST AND STORE
F655 45                 4478            INC     BP
F656 83C650             4479            ADD     SI,80           ; POINTER INTO REGEN
F659 FECE               4480            DEC     DH              ; LOOP CONTROL
F65B 75EB               4481            JNZ     S12             ; DO IT SOME MORE
F65D EB1790             4482            JMP     S15             ; GO MATCH THE SAVED CODE POINTS
                        4483
                        4484    ;------ MEDIUM RESOLUTION READ
F660                    4485    S13:                            ; MED_RES_READ
F660 D1E6               4486            SAL     SI,1            ; OFFSET*2 SINCE 2 BYTES/CHAR
F662 B604               4487            MOV     DH,4            ; NUMBER OF PASSES
F664                    4488    S14:
F664 E88800             4489            CALL    S23             ; GET PAIR BYTES FROM REGEN INTO SINGLE SAVE
F667 81C60020           4490            ADD     SI,2000H        ; GO TO LOWER REGION
F66B E88100             4491            CALL    S23             ; GET THIS PAIR INTO SAVE
F66E 81EEB01F           4492            SUB     SI,2000H-80     ; ADJUST POINTER BACK INTO UPPER
F672 FECE               4493            DEC     DH
F674 75EE               4494            JNZ     S14             ; KEEP GOING UNTIL ALL 8 DONE
                        4495
                        4496    ;-------- SAVE AREA HAS CHARACTER IN IT, MATCH IT
F676                    4497    S15:                            ; FIND_CHAR
F676 BF6EFA             4498            MOV     DI,0FA6EH       ; OFFSET CRT_CHAR_GEN-ESTABLISH ADDRESSING
F679 0E                 4499            PUSH    CS
F67A 07                 4500            POP     ES                      ; CODE POINTS IN CS
F67B 83ED08             4501            SUB     BP,8            ; ADJUST POINTER TO BEGINNING OF SAVE AREA
F67E 8BF5               4502            MOV     SI,BP
F680 FC                 4503            CLD                     ; ENSURE DIRECTION
F681 B000               4504            MOV     AL,0            ; CURRENT CODE POINT BEING MATCHED
F683                    4505    S16:
F683 16                 4506            PUSH    SS              ; ESTABLISH ADDRESSING TO STACK
F684 1F                 4507            POP     DS              ; FOR THE STRING COMPARE
F685 BA8000             4508            MOV     DX,128          ; NUMBER TO TEST AGAINST
F688                    4509    S17:
F688 56                 4510            PUSH    SI              ; SAVE SAVE AREA POINTER
```

```
LOC OBJ                 LINE    SOURCE


F689 57                 4511            PUSH    DI              ; SAVE CODE POINTER
F68A B90800             4512            MOV     CX,8            ; NUMBER OF BYTES TO MATCH
F68D F3                 4513            REPE    CMPSB           ; COMPARE THE 8 BYTES
F68E A6
F68F 5F                 4514            POP     DI              ; RECOVER THE POINTERS
F690 5E                 4515            POP     SI
F691 741E               4516            JZ      S18             ; IF ZERO FLAG SET, THEN MATCH OCCURRED
F693 FEC0               4517            INC     AL              ; NO MATCH, MOVE ON TO NEXT
F695 83C708             4518            ADD     DI,8            ; NEXT CODE POINT
F698 4A                 4519            DEC     DX              ; LOOP CONTROL
F699 75ED               4520            JNZ     S17             ; DO ALL OF THEM
                        4521
                        4522    ;------ CHAR NOT MATCHED, MIGHT BE IN USER SUPPLIED SECOND HALF
                        4523
F69B 3C00               4524            CMP     AL,0            ; AL<> 0 IF ONLY 1ST HALF SCANNED
F69D 7412               4525            JE      S18             ; IF = 0, THEN ALL HAS BEEN SCANNED
F69F 2BC0               4526            SUB     AX,AX
F6A1 8ED8               4527            MOV     DS,AX           ; ESTABLISH ADDRESSING TO VECTOR
                        4528            ASSUME  DS:ABS0
F6A3 C43E7C00           4529            LES     DI,EXT_PTR      ; GET POINTER
F6A7 8CC0               4530            MOV     AX,ES           ; SEE IF THE POINTER REALLY EXISTS
F6A9 0BC7               4531            OR      AX,DI           ; IF ALL 0, THEN DOESN'T EXIST
F6AB 7404               4532            JZ      S18             ; NO SENSE LOOKING
F6AD B080               4533            MOV     AL,128          ; ORIGIN FOR SECOND HALF
F6AF EBD2               4534            JMP     S16             ; GO BACK AND TRY FOR IT
                        4535            ASSUME  DS:DATA
                        4536
                        4537    ;------ CHARACTER IS FOUND ( AL=0 IF NOT FOUND )
F6B1                    4538    S18:
F6B1 83C408             4539            ADD     SP,8            ; READJUST THE STACK, THROW AWAY SAVE
F6B4 E910FB             4540            JMP     VIDEO_RETURN    ; ALL DONE
                        4541    GRAPHICS_READ   ENDP
                        4542    ;--------------------------------------------
                        4543    ; EXPAND_MED_COLOR
                        4544    ;  THIS ROUTINE EXPANDS THE LOW 2 BITS IN BL TO
                        4545    ;  FILL THE ENTIRE BX REGISTER
                        4546    ; ENTRY --
                        4547    ;  BL = COLOR TO BE USED ( LOW 2 BITS )
                        4548    ; EXIT --
                        4549    ;  BX = COLOR TO BE USED ( 8 REPLICATIONS OF THE 2 COLOR BITS )
                        4550    ;--------------------------------------------
F6B7                    4551    S19     PROC    NEAR
F6B7 80E303             4552            AND     BL,3            ; ISOLATE THE COLOR BITS
F6BA 8AC3               4553            MOV     AL,BL           ; COPY TO AL
F6BC 51                 4554            PUSH    CX              ; SAVE REGISTER
F6BD B90300             4555            MOV     CX,3            ; NUMBER OF TIMES TO DO THIS
F6C0                    4556    S20:
F6C0 D0E0               4557            SAL     AL,1
F6C2 D0E0               4558            SAL     AL,1            ; LEFT SHIFT BY 2
F6C4 0AD8               4559            OR      BL,AL           ; ANOTHER COLOR VERSION INTO BL
F6C6 E2F8               4560            LOOP    S20             ; FILL ALL OF BL
F6C8 8AFB               4561            MOV     BH,BL           ; FILL UPPER PORTION
F6CA 59                 4562            POP     CX              ; REGISTER BACK
F6CB C3                 4563            RET                     ; ALL DONE
                        4564    S19     ENDP
                        4565    ;--------------------------------------------
                        4566    ; EXPAND_BYTE
                        4567    ;  THIS ROUTINE TAKES THE BYTE IN AL AND DOUBLES ALL
                        4568    ;  OF THE BITS, TURNING THE 8 BITS INTO 16 BITS.
                        4569    ;  THE RESULT IS LEFT IN AX
                        4570    ;--------------------------------------------
F6CC                    4571    S21     PROC    NEAR
F6CC 52                 4572            PUSH    DX              ; SAVE REGISTERS
F6CD 51                 4573            PUSH    CX
F6CE 53                 4574            PUSH    BX
F6CF BA0000             4575            MOV     DX,0            ; RESULT REGISTER
F6D2 B90100             4576            MOV     CX,1            ; MASK REGISTER
F6D5                    4577    S22:
F6D5 8BD8               4578            MOV     BX,AX           ; BASE INTO TEMP
F6D7 23D9               4579            AND     BX,CX           ; USE MASK TO EXTRACT A BIT
F6D9 0BD3               4580            OR      DX,BX           ; PUT INTO RESULT REGISTER
F6DB D1E0               4581            SHL     AX,1
F6DD D1E1               4582            SHL     CX,1            ; SHIFT BASE AND MASK BY 1
F6DF 8BD8               4583            MOV     BX,AX           ; BASE TO TEMP
F6E1 23D9               4584            AND     BX,CX           ; EXTRACT THE SAME BIT
F6E3 0BD3               4585            OR      DX,BX           ; PUT INTO RESULT
```

```
LOC OBJ                 LINE    SOURCE

F6E5 D1E1               4586            SHL     CX,1            ; SHIFT ONLY MASK NOW, MOVING TO NEXT BASE
F6E7 73EC               4587            JNC     S22             ; USE MASK BIT COMING OUT TO TERMINATE
F6E9 8BC2               4588            MOV     AX,DX           ; RESULT TO PARM REGISTER
F6EB 5B                 4589            POP     BX
F6EC 59                 4590            POP     CX              ; RECOVER REGISTERS
F6ED 5A                 4591            POP     DX
F6EE C3                 4592            RET                     ; ALL DONE
                        4593    S21     ENDP
                        4594    ;--------------------------------------------
                        4595    ; MED_READ_BYTE
                        4596    ; THIS ROUTINE WILL TAKE 2 BYTES FROM THE REGEN BUFFER,
                        4597    ;  COMPARE AGAINST THE CURRENT FOREGROUND COLOR, AND PLACE
                        4598    ;  THE CORRESPONDING ON/OFF BIT PATTERN INTO THE CURRENT
                        4599    ;  POSITION IN THE SAVE AREA
                        4600    ; ENTRY --
                        4601    ;  SI,DS = POINTER TO REGEN AREA OF INTEREST
                        4602    ;  BX = EXPANDED FOREGROUND COLOR
                        4603    ;  BP = POINTER TO SAVE AREA
                        4604    ; EXIT --
                        4605    ;  BP IS INCREMENT AFTER SAVE
                        4606    ;--------------------------------------------
F6EF                    4607    S23     PROC    NEAR
F6EF 8A24               4608            MOV     AH,[SI]         ; GET FIRST BYTE
F6F1 8A4401             4609            MOV     AL,[SI+1]       ; GET SECOND BYTE
F6F4 B900C0             4610            MOV     CX,0C000H       ; 2 BIT MASK TO TEST THE ENTRIES
F6F7 B200               4611            MOV     DL,0            ; RESULT REGISTER
F6F9                    4612    S24:
F6F9 85C1               4613            TEST    AX,CX           ; IS THIS SECTION BACKGROUND?
F6FB F8                 4614            CLC                     ; CLEAR CARRY IN HOPES THAT IT IS
F6FC 7401               4615            JZ      S25             ; IF ZERO, IT IS BACKGROUND
F6FE F9                 4616            STC                     ; WASN'T, SO SET CARRY
F6FF D0D2               4617    S25:    RCL     DL,1            ; MOVE THAT BIT INTO THE RESULT
F701 D1E9               4618            SHR     CX,1
F703 D1E9               4619            SHR     CX,1            ; MOVE THE MASK TO THE RIGHT BY 2 BITS
F705 73F2               4620            JNC     S24             ; DO IT AGAIN IF MASK DIDN'T FALL OUT
F707 885600             4621            MOV     [BP],DL         ; STORE RESULT IN SAVE AREA
F70A 45                 4622            INC     BP              ; ADJUST POINTER
F70B C3                 4623            RET                     ; ALL DONE
                        4624    S23     ENDP
                        4625    ;--------------------------------------------
                        4626    ; V4_POSITION
                        4627    ;  THIS ROUTINE TAKES THE CURSOR POSITION CONTAINED IN
                        4628    ;  THE MEMORY LOCATION, AND CONVERTS IT INTO AN OFFSET
                        4629    ;  INTO THE REGEN BUFFER, ASSUMING ONE BYTE/CHAR.
                        4630    ;  FOR MEDIUM RESOLUTION GRAPHICS, THE NUMBER MUST
                        4631    ;  BE DOUBLED.
                        4632    ; ENTRY -- NO REGISTERS,MEMORY LOCATION CURSOR_POSN IS USED
                        4633    ; EXIT--
                        4634    ;  AX CONTAINS OFFSET INTO REGEN BUFFER
                        4635    ;--------------------------------------------
F70C                    4636    S26     PROC    NEAR
F70C A15000         R   4637            MOV     AX,CURSOR_POSN          ; GET CURRENT CURSOR
F70F                    4638    GRAPH_POSN      LABEL   NEAR
F70F 53                 4639            PUSH    BX                      ; SAVE REGISTER
F710 8BD8               4640            MOV     BX,AX                   ; SAVE A COPY OF CURRENT CURSOR
F712 8AC4               4641            MOV     AL,AH                   ; GET ROWS TO AL
F714 F6264A00       R   4642            MUL     BYTE PTR CRT_COLS       ; MULTIPLY BY BYTES/COLUMN
F718 D1E0               4643            SHL     AX,1                    ; MULTIPLY * 4 SINCE 4 ROWS/BYTE
F71A D1E0               4644            SHL     AX,1
F71C 2AFF               4645            SUB     BH,BH                   ; ISOLATE COLUMN VALUE
F71E 03C3               4646            ADD     AX,BX                   ; DETERMINE OFFSET
F720 5B                 4647            POP     BX                      ; RECOVER POINTER
F721 C3                 4648            RET                             ; ALL DONE
                        4649    S26     ENDP
                        4650    ;--------------------------------------------
                        4651    ; WRITE_TTY
                        4652    ; THIS INTERFACE PROVIDES A TELETYPE LIKE INTERFACE TO THE
                        4653    ;   VIDEO CARD.  THE INPUT CHARACTER IS WRITTEN TO THE CURRENT
                        4654    ;   CURSOR POSITION, AND THE CURSOR IS MOVED TO THE NEXT POSITION.
                        4655    ;   IF THE CURSOR LEAVES THE LAST COLUMN OF THE FIELD, THE COLUMN
                        4656    ;   IS SET TO ZERO, AND THE ROW VALUE IS INCREMENTED.  IF THE ROW
                        4657    ;   ROW VALUE LEAVES THE FIELD, THE CURSOR IS PLACED ON THE LAST ROW,
                        4658    ;   FIRST COLUMN, AND THE ENTIRE SCREEN IS SCROLLED UP ONE LINE.
                        4659    ;   WHEN THE SCREEN IS SCROLLED UP, THE ATTRIBUTE FOR FILLING THE
                        4660    ;   NEWLY BLANKED LINE IS READ FROM THE CURSOR POSITION ON THE PREVIOUS
                        4661    ;   LINE BEFORE THE SCROLL, IN CHARACTER MODE.  IN GRAPHICS MODE,
```

```
LOC OBJ                 LINE    SOURCE

                        4662    ;   THE 0 COLOR IS USED.
                        4663    ; ENTRY --
                        4664    ;   (AH) = CURRENT CRT MODE
                        4665    ;   (AL) = CHARACTER TO BE WRITTEN
                        4666    ;           NOTE THAT BACK SPACE, CAR RET, BELL AND LINE FEED ARE HANDLED
                        4667    ;           AS COMMANDS RATHER THAN AS DISPLAYABLE GRAPHICS
                        4668    ;   (BL) = FOREGROUND COLOR FOR CHAR WRITE IF CURRENTLY IN A GRAPHICS MODE
                        4669    ; EXIT --
                        4670    ;    ALL REGISTERS SAVED
                        4671    ;--------------------------------------------
                        4672            ASSUME  CS:CODE,DS:DATA
F722                    4673    WRITE_TTY       PROC    NEAR
F722 50                 4674            PUSH    AX              ; SAVE REGISTERS
F723 50                 4675            PUSH    AX              ; SAVE CHAR TO WRITE
F724 B403               4676            MOV     AH,3
F726 CD10               4677            INT     10H             ; READ THE CURSOR POSITION
F728 58                 4678            POP     AX              ; RECOVER CHAR
                        4679
                        4680    ;------ DX NOW HAS THE CURRENT CURSOR POSITION
                        4681
F729 3C08               4682            CMP     AL,8            ; IS IT A BACKSPACE
F72B 7459               4683            JE      U8              ; BACK_SPACE
F72D 3C0D               4684            CMP     AL,0DH          ; IS IT CARRIAGE RETURN
F72F 745E               4685            JE      U9              ; CAR_RET
F731 3C0A               4686            CMP     AL,0AH          ; IS IT A LINE FEED
F733 745E               4687            JE      U10             ; LINE_FEED
F735 3C07               4688            CMP     AL,07H          ; IS IT A BELL
F737 7461               4689            JE      U11             ; BELL
                        4690
                        4691    ;------ WRITE THE CHAR TO THE SCREEN
                        4692
F739 8A3E6200       R   4693            MOV     BH,ACTIVE_PAGE  ; GET THE CURRENT ACTIVE PAGE
F73D B40A               4694            MOV     AH,10           ; WRITE CHAR ONLY
F73F B90100             4695            MOV     CX,1            ; ONLY ONE CHAR
F742 CD10               4696            INT     10H             ; WRITE THE CHAR
                        4697
                        4698    ;------ POSITION THE CURSOR FOR NEXT CHAR
                        4699
F744 FEC2               4700            INC     DL
F746 3A164A00       R   4701            CMP     DL,BYTE PTR CRT_COLS    ; TEST FOR COLUMN OVERFLOW
F74A 7536               4702            JNZ     U7              ; SET_CURSOR
F74C B200               4703            MOV     DL,0            ; COLUMN FOR CURSOR
F74E 80FE18             4704            CMP     DH,24
F751 752D               4705            JNZ     U6              ; SET_CURSOR_INC
                        4706
                        4707    ;------ SCROLL REQUIRED
F753                    4708    U1:
                        4709
F753 B402               4710            MOV     AH,2
F755 B700               4711            MOV     BH,0
F757 CD10               4712            INT     10H             ; SET THE CURSOR
                        4713
                        4714    ;------ DETERMINE VALUE TO FILL WITH DURING SCROLL
                        4715
F759 A04900         R   4716            MOV     AL,CRT_MODE     ; GET THE CURRENT MODE
F75C 3C04               4717            CMP     AL,4
F75E 7206               4718            JC      U2              ; READ-CURSOR
F760 3C07               4719            CMP     AL,7
F762 B700               4720            MOV     BH,0            ; FILL WITH BACKGROUND
F764 7506               4721            JNE     U3              ; SCROLL-UP
                        4722
F766                    4723    U2:                             ; READ-CURSOR
F766 B408               4724            MOV     AH,8
F768 CD10               4725            INT     10H             ; READ CHAR/ATTR AT CURRENT CURSOR
F76A 8AFC               4726            MOV     BH,AH           ; STORE IN BH
                        4727
F76C                    4728    U3:                             ; SCROLL-UP
F76C B80106             4729            MOV     AX,601H         ; SCROLL ONE LINE
F76F B90000             4730            MOV     CX,0            ; UPPER LEFT CORNER
F772 B618               4731            MOV     DH,24           ; LOWER RIGHT ROW
F774 8A164A00       R   4732            MOV     DL,BYTE PTR CRT_COLS    ; LOWER RIGHT COLUMN
F778 FECA               4733            DEC     DL
F77A                    4734    U4:                             ; VIDEO-CALL-RETURN
F77A CD10               4735            INT     10H             ; SCROLL UP THE SCREEN
F77C                    4736    U5:                             ; TTY-RETURN
F77C 58                 4737            POP     AX              ; RESTORE THE CHARACTER
```

```
LOC OBJ                  LINE    SOURCE

F77D E947FA              4738            JMP     VIDEO_RETURN    ; RETURN TO CALLER
                         4739
F780                     4740    U6:                             ; SET-CURSOR-INC
F780 FEC6                4741            INC     DH              ; NEXT ROW
F782                     4742    U7:                             ; SET-CURSOR
F782 B402                4743            MOV     AH,2
F784 EBF4                4744            JMP     U4              ; ESTABLISH THE NEW CURSOR
                         4745
                         4746    ;------ BACK SPACE FOUND
                         4747
F786                     4748    U8:
F786 80FA00              4749            CMP     DL,0            ; ALREADY AT END OF LINE
F789 74F7                4750            JE      U7              ; SET_CURSOR
F78B FECA                4751            DEC     DL              ; NO -- JUST MOVE IT BACK
F78D EBF3                4752            JMP     U7              ; SET_CURSOR
                         4753
                         4754    ;------ CARRIAGE RETURN FOUND
                         4755
F78F                     4756    U9:
F78F B200                4757            MOV     DL,0            ; MOVE TO FIRST COLUMN
F791 EBEF                4758            JMP     U7              ; SET_CURSOR
                         4759
                         4760    ;------ LINE FEED FOUND
                         4761
F793                     4762    U10:
F793 80FE18              4763            CMP     DH,24           ; BOTTOM OF SCREEN
F796 75E8                4764            JNE     U6              ; YES, SCROLL THE SCREEN
F798 EBB9                4765            JMP     U1              ; NO, JUST SET THE CURSOR
                         4766
                         4767    ;------ BELL FOUND
                         4768
F79A                     4769    U11:
F79A B302                4770            MOV     BL,2            ; SET UP COUNT FOR BEEP
F79C E8C7EE              4771            CALL    BEEP            ; SOUND THE POD BELL
F79F EBDB                4772            JMP     U5              ; TTY_RETURN
                         4773    WRITE_TTY       ENDP
                         4774    ;--------------------------------------------
                         4775    ; LIGHT PEN
                         4776    ;       THIS ROUTINE TESTS THE LIGHT PEN SWITCH AND THE LIGHT
                         4777    ;         PEN TRIGGER.  IF BOTH ARE SET, THE LOCATION OF THE LIGHT
                         4778    ;         PEN IS DETERMINED.  OTHERWISE, A RETURN WITH NO INFORMATION
                         4779    ;         IS MADE.
                         4780    ;       ON EXIT:
                         4781    ;       (AH) = 0 IF NO LIGHT PEN INFORMATION IS AVAILABLE
                         4782    ;               BX,CX,DX ARE DESTROYED
                         4783    ;       (AH) = 1 IF LIGHT PEN IS AVAILABLE
                         4784    ;               (DH,DL) = ROW,COLUMN OF CURRENT LIGHT PEN POSITION
                         4785    ;               (CH) = RASTER POSITION
                         4786    ;               (BX) = BEST GUESS AT PIXEL HORIZONTAL POSITION
                         4787    ;--------------------------------------------
                         4788            ASSUME  CS:CODE,DS:DATA
                         4789    ;------ SUBTRACT_TABLE
F7A1                     4790    V1      LABEL   BYTE
F7A1 0303050503030304    4791            DB      3,3,5,5,3,3,3,4 ;
F7A9                     4792    READ_LPEN       PROC    NEAR
                         4793
                         4794    ;------ WAIT FOR LIGHT PEN TO BE DEPRESSED
                         4795
F7A9 B400                4796            MOV     AH,0            ; SET NO LIGHT PEN RETURN CODE
F7AB 8B166300      R     4797            MOV     DX,ADDR_6845            ; GET BASE ADDRESS OF 6845
F7AF 83C206              4798            ADD     DX,6                    ; POINT TO STATUS REGISTER
F7B2 EC                  4799            IN      AL,DX                   ; GET STATUS REGISTER
F7B3 A804                4800            TEST    AL,4                    ; TEST LIGHT PEN SWITCH
F7B5 7578                4801            JNZ     V6                      ; NOT SET, RETURN
                         4802
                         4803    ;------ NOW TEST FOR LIGHT PEN TRIGGER
                         4804
F7B7 A802                4805            TEST    AL,2                    ; TEST LIGHT PEN TRIGGER
F7B9 747E                4806            JZ      V7                      ; RETURN WITHOUT RESETTING TRIGGER
                         4807
                         4808    ;------ TRIGGER HAS BEEN SET, READ THE VALUE IN
                         4809
F7BB B410                4810            MOV     AH,16           ; LIGHT PEN REGISTERS ON 6845
                         4811
                         4812    ;------ INPUT REGS POINTED TO BY AH, AND CONVERT TO ROW COLUMN IN DX
                         4813
F7BD 8B166300      R     4814            MOV     DX,ADDR_6845    ; ADDRESS REGISTER FOR 6845
```

```
LOC OBJ                  LINE    SOURCE

F7C1 8AC4                4815            MOV     AL,AH           ; REGISTER TO READ
F7C3 EE                  4816            OUT     DX,AL           ; SET IT UP
F7C4 42                  4817            INC     DX              ; DATA REGISTER
F7C5 EC                  4818            IN      AL,DX           ; GET THE VALUE
F7C6 8AE8                4819            MOV     CH,AL           ; SAVE IN CX
F7C8 4A                  4820            DEC     DX              ; ADDRESS REGISTER
F7C9 FEC4                4821            INC     AH
F7CB 8AC4                4822            MOV     AL,AH           ; SECOND DATA REGISTER
F7CD EE                  4823            OUT     DX,AL
F7CE 42                  4824            INC     DX              ; POINT TO DATA REGISTER
F7CF EC                  4825            IN      AL,DX           ; GET SECOND DATA VALUE
F7D0 8AE5                4826            MOV     AH,CH           ; AX HAS INPUT VALUE
                         4827
                         4828    ;------ AX HAS THE VALUE READ IN FROM THE 6845
                         4829
F7D2 8A1E4900      R     4830            MOV     BL,CRT_MODE
F7D6 2AFF                4831            SUB     BH,BH           ; MODE VALUE TO BX
F7D8 2E8A9FA1F7    R     4832            MOV     BL,CS:V1[BX]    ; DETERMINE AMOUNT TO SUBTRACT
F7DD 2BC3                4833            SUB     AX,BX           ; TAKE IT AWAY
F7DF 2B064E00      R     4834            SUB     AX,CRT_START    ; CONVERT TO CORRECT PAGE ORIGIN
F7E3 7903                4835            JNS     V2              ; IF POSITIVE, DETERMINE MODE
F7E5 B80000              4836            MOV     AX,0            ; <0 PLAYS AS 0
                         4837
                         4838    ;------ DETERMINE MODE OF OPERATION
                         4839
F7E8                     4840    V2:                             ; DETERMINE_MODE
F7E8 B103                4841            MOV     CL,3            ; SET *8 SHIFT COUNT
F7EA 803E490004    R     4842            CMP     CRT_MODE,4      ; DETERMINE IF GRAPHICS OR ALPHA
F7EF 722A                4843            JB      V4              ; ALPHA_PEN
F7F1 803E490007    R     4844            CMP     CRT_MODE,7
F7F6 7423                4845            JE      V4              ; ALPHA_PEN
                         4846
                         4847    ;------ GRAPHICS MODE
                         4848
F7F8 B228                4849            MOV     DL,40           ; DIVISOR FOR GRAPHICS
F7FA F6F2                4850            DIV     DL              ; DETERMINE ROW(AL) AND COLUMN(AH)
                         4851                                    ;  AL RANGE 0-99, AH RANGE 0-39
                         4852    ;------ DETERMINE GRAPHIC ROW POSITION
                         4853
F7FC 8AE8                4854            MOV     CH,AL           ; SAVE ROW VALUE IN CH
F7FE 02ED                4855            ADD     CH,CH           ; *2 FOR EVEN/ODD FIELD
F800 8ADC                4856            MOV     BL,AH           ; COLUMN VALUE TO BX
F802 2AFF                4857            SUB     BH,BH           ; MULTIPLY BY 8 FOR MEDIUM RES
F804 803E490006    R     4858            CMP     CRT_MODE,6      ; DETERMINE MEDIUM OR HIGH RES
F809 7504                4859            JNE     V3              ; NOT_HIGH_RES
F80B B104                4860            MOV     CL,4            ; SHIFT VALUE FOR HIGH RES
F80D D0E4                4861            SAL     AH,1            ; COLUMN VALUE TIMES 2 FOR HIGH RES
F80F                     4862    V3:                             ; NOT_HIGH_RES
F80F D3E3                4863            SHL     BX,CL           ; MULTIPLY *16 FOR HIGH RES
                         4864
                         4865    ;------ DETERMINE ALPHA CHAR POSITION
                         4866
F811 8AD4                4867            MOV     DL,AH           ; COLUMN VALUE FOR RETURN
F813 8AF0                4868            MOV     DH,AL           ; ROW VALUE
F815 D0EE                4869            SHR     DH,1            ; DIVIDE BY 4
F817 D0EE                4870            SHR     DH,1            ;  FOR VALUE IN 0-24 RANGE
F819 EB12                4871            JMP     SHORT V5        ; LIGHT_PEN_RETURN_SET
                         4872
                         4873    ;------ ALPHA MODE ON LIGHT PEN
                         4874
F81B                     4875    V4:                             ; ALPHA_PEN
F81B F6364A00      R     4876            DIV     BYTE PTR CRT_COLS       ; DETERMINE ROW,COLUMN VALUE
F81F 8AF0                4877            MOV     DH,AL           ; ROWS TO DH
F821 8AD4                4878            MOV     DL,AH           ; COLS TO DL
F823 D2E0                4879            SAL     AL,CL           ; MULTIPLY ROWS * 8
F825 8AE8                4880            MOV     CH,AL           ; GET RASTER VALUE TO RETURN REG
F827 8ADC                4881            MOV     BL,AH           ; COLUMN VALUE
F829 32FF                4882            XOR     BH,BH           ;  TO BX
F82B D3E3                4883            SAL     BX,CL
F82D                     4884    V5:                             ; LIGHT_PEN_RETURN_SET
F82D B401                4885            MOV     AH,1            ; INDICATE EVERTHING SET
F82F                     4886    V6:                             ; LIGHT_PEN_RETURN
F82F 52                  4887            PUSH    DX              ; SAVE RETURN VALUE (IN CASE)
F830 8B166300      R     4888            MOV     DX,ADDR_6845    ; GET BASE ADDRESS
F834 83C207              4889            ADD     DX,7            ; POINT TO RESET PARM
F837 EE                  4890            OUT     DX,AL           ; ADDRESS, NOT DATA, IS IMPORTANT
```

```
LOC OBJ                 LINE    SOURCE

F838 5A                 4891            POP     DX              ; RECOVER VALUE
F839                    4892    V7:                             ; RETURN_NO_RESET
F839 5F                 4893            POP     DI
F83A 5E                 4894            POP     SI
F83B 1F                 4895            POP     DS                      ; DISCARD SAVED BX,CX,DX
F83C 1F                 4896            POP     DS
F83D 1F                 4897            POP     DS
F83E 1F                 4898            POP     DS
F83F 07                 4899            POP     ES
F840 CF                 4900            IRET
                        4901    READ_LPEN       ENDP
                        4902    ;--- INT 12 -----------------------------------
                        4903    ; MEMORY_SIZE_DETERMINE
                        4904    ;       THIS ROUTINE DETERMINES THE AMOUNT OF MEMORY IN THE SYSTEM
                        4905    ;       AS REPRESENTED BY THE SWITCHES ON THE PLANAR.  NOTE THAT
                        4906    ;       THE SYSTEM MAY NOT BE ABLE TO USE I/O MEMORY UNLESS THERE
                        4907    ;       IS A FULL COMPLEMENT OF 64K BYTES ON THE PLANAR.
                        4908    ; INPUT
                        4909    ;       NO REGISTERS
                        4910    ;       THE MEMORY_SIZE VARIABLE IS SET DURING POWER ON DIAGNOSTICS
                        4911    ;        ACCORDING TO THE FOLLOWING HARDWARE ASSUMPTIONS:
                        4912    ;       PORT 60 BITS 3,2 = 00 - 16K BASE RAM
                        4913    ;                          01 - 32K BASE RAM
                        4914    ;                          10 - 48K BASE RAM
                        4915    ;                          11 - 64K BASE RAM
                        4916    ;       PORT 62 BITS 3-0 INDICATE AMOUNT OF I/O RAM IN 32K INCREMENTS
                        4917    ;               E.G., 0000 - NO RAM IN I/O CHANNEL
                        4918    ;                     0010 - 64K RAM IN I/O CHANNEL, ETC.
                        4919    ; OUTPUT
                        4920    ;       (AX) = NUMBER OF CONTIGUOUS 1K BLOCKS OF MEMORY
                        4921    ;-----------------------------------------------
                        4922            ASSUME  CS:CODE,DS:DATA
F841                    4923    MEMORY_SIZE_DETERMINE   PROC    FAR
F841 FB                 4924            STI                     ; INTERRUPTS BACK ON
F842 1E                 4925            PUSH    DS              ; SAVE SEGMENT
F843 B84000         R   4926            MOV     AX,DATA         ; ESTABLISH ADDRESSING
F846 8ED8               4927            MOV     DS,AX
F848 A11300         R   4928            MOV     AX,MEMORY_SIZE  ; GET VALUE
F84B 1F                 4929            POP     DS              ; RECOVER SEGMENT
F84C CF                 4930            IRET                    ; RETURN TO CALLER
                        4931    MEMORY_SIZE_DETERMINE   ENDP
                        4932    ;--- INT 11 -----------------------------------
                        4933    ; EQUIPMENT DETERMINATION
                        4934    ;       THIS ROUTINE ATTEMPTS TO DETERMINE WHAT OPTIONAL
                        4935    ;       DEVICES ARE ATTACHED TO THE SYSTEM.
                        4936    ; INPUT
                        4937    ;       NO REGISTERS
                        4938    ;       THE EQUIP_FLAG VARIABLE IS SET DURING THE POWER ON DIAGNOSTICS
                        4939    ;        USING THE FOLLOWING HARDWARE ASSUMPTIONS:
                        4940    ;       PORT 60  = LOW ORDER BYTE OF EQUPMENT
                        4941    ;       PORT 3FA = INTERRUPT ID REGISTER OF 8250
                        4942    ;                BITS 7-3 ARE ALWAYS 0
                        4943    ;       PORT 378 = OUTPUT PORT OF PRINTER -- 8255 PORT THAT
                        4944    ;                  CAN BE READ AS WELL AS WRITTEN
                        4945    ; OUTPUT
                        4946    ;       (AX) IS SET, BIT SIGNIFICANT, TO INDICATE ATTACHED I/O
                        4947    ;       BIT 15,14 = NUMBER OF PRINTERS ATTACHED
                        4948    ;       BIT 13 NOT USED
                        4949    ;       BIT 12 = GAME I/O ATTACHED
                        4950    ;       BIT 11,10,9 = NUMBER OF RS232 CARDS ATTACHED
                        4951    ;       BIT 8 UNUSED
                        4952    ;       BIT 7,6 = NUMBER OF DISKETTE DRIVES
                        4953    ;                 00=1, 01=2, 10=3, 11=4 ONLY IF BIT 0 = 1
                        4954    ;       BIT 5,4 = INITIAL VIDEO MODE
                        4955    ;                         00 - UNUSED
                        4956    ;                         01 - 40X25 BW USING COLOR CARD
                        4957    ;                         10 - 80X25 BW USING COLOR CARD
                        4958    ;                         11 - 80X25 BW USING BW CARD
                        4959    ;       BIT 3,2 = PLANAR RAM SIZE (00=16K,01=32K,10=48K,11=64K)
                        4960    ;       BIT 1 NOT USED
                        4961    ;       BIT 0 = IPL FROM DISKETTE -- THIS BIT INDICATES THAT THERE ARE DISKETTE
                        4962    ;                DRIVES ON THE SYSTEM
                        4963    ;
                        4964    ;       NO OTHER REGISTERS AFFECTED
                        4965    ;-----------------------------------------------
                        4966            ASSUME  CS:CODE,DS:DATA
```

```
LOC OBJ                  LINE    SOURCE


F84D                     4967    EQUIPMENT       PROC    FAR
F84D FB                  4968            STI                     ; INTERRUPTS BACK ON
F84E 1E                  4969            PUSH    DS              ; SAVE SEGMENT REGISTER
F84F B84000          R   4970            MOV     AX,DATA         ; ESTABLISH ADDRESSING
F852 8ED8                4971            MOV     DS,AX
F854 A11000          R   4972            MOV     AX,EQUIP_FLAG   ; GET THE CURRENT SETTINGS
F857 1F                  4973            POP     DS              ; RECOVER SEGMENT
F858 CF                  4974            IRET                    ; RETURN TO CALLER
                         4975    EQUIPMENT       ENDP
                         4976    ;--- INT 15 ---------------------------------
                         4977    ; CASSETTE I/O
                         4978    ;       (AH) = 0  TURN CASSETTE MOTOR ON
                         4979    ;       (AH) = 1  TURN CASSETTE MOTOR OFF
                         4980    ;       (AH) = 2  READ 1 OR MORE 256 BYTE BLOCKS FROM CASSETTE
                         4981    ;               (ES,BX) = POINTER TO DATA BUFFER
                         4982    ;               (CX) = COUNT OF BYTES TO READ
                         4983    ;               ON EXIT:
                         4984    ;               (ES,BX) = POINTER TO LAST BYTE READ + 1
                         4985    ;               (DX) = COUNT OF BYTES ACTUALLY READ
                         4986    ;               (CY) = 0 IF NO ERROR OCCURRED
                         4987    ;                    = 1 IF ERROR OCCURRED
                         4988    ;               (AH) = ERROR RETURN IF (CY)= 1
                         4989    ;                      = 01 IF CRC ERROR WAS DETECTED
                         4990    ;                      = 02 IF DATA TRANSITIONS ARE LOST
                         4991    ;                      = 04 IF NO DATA WAS FOUND
                         4992    ;       (AH) = 3 WRITE 1 OR MORE 256 BYTE BLOCKS TO CASSETTE
                         4993    ;               (ES,BX) = POINTER TO DATA BUFFER
                         4994    ;               (CX) = COUNT OF BYTES TO WRITE
                         4995    ;               ON EXIT:
                         4996    ;               (EX,BX) = POINTER TO LAST BYTE WRITTEN + 1
                         4997    ;               (CX) = 0
                         4998    ;       (AH) = ANY OTHER THAN ABOVE VALUES CAUSES (CY)= 1
                         4999    ;               AND (AH)= 80 TO BE RETURNED (INVALID COMMAND).
                         5000    ;-----------------------------------------------
                         5001            ASSUME  DS:DATA, ES:NOTHING,SS:NOTHING,CS:CODE
F859                     5002    CASSETTE_IO     PROC    FAR
F859 FB                  5003            STI                             ; INTERRUPTS BACK ON
F85A 1E                  5004            PUSH    DS                      ; ESTABLISH ADDRESSING TO DATA
F85B 50                  5005            PUSH    AX
F85C B84000          R   5006            MOV     AX, DATA
F85F 8ED8                5007            MOV     DS, AX
F861 802671007F      R   5008            AND     BIOS_BREAK, 7FH         ; MAKE SURE BREAK FLAG IS OFF
F866 58                  5009            POP     AX
F867 E80400              5010            CALL    W1                      ; CASSETTE_IO_CONT
F86A 1F                  5011            POP     DS
F86B CA0200              5012            RET     2               ; INTERRUPT RETURN
                         5013    CASSETTE_IO     ENDP
F86E                     5014    W1      PROC    NEAR
                         5015    ;-----------------------------------------------
                         5016    ; PURPOSE:
                         5017    ;  TO CALL APPROPRIATE ROUTINE DEPENDING ON REG AH
                         5018    ;
                         5019    ;  AH           ROUTINE
                         5020    ;_____________________________________
                         5021    ;  0            MOTOR ON
                         5022    ;  1            MOTOR OFF
                         5023    ;  2            READ CASSETTE BLOCK
                         5024    ;  3            WRITE CASSETTE BLOCK
                         5025    ;_____________________________________
                         5026
F86E 0AE4                5027            OR      AH,AH           ;TURN ON MOTOR?
F870 7413                5028            JZ      MOTOR_ON        ;YES, DO IT
F872 FECC                5029            DEC     AH              ;TURN OFF MOTOR?
F874 7418                5030            JZ      MOTOR_OFF       ;YES, DO IT
F876 FECC                5031            DEC     AH              ;READ CASSETTE BLOCK?
F878 741A                5032            JZ      READ_BLOCK      ;YES, DO IT
F87A FECC                5033            DEC     AH              ;WRITE CASSETTE BLOCK?
F87C 7503                5034            JNZ     W2              ; NOT_DEFINED
F87E E92701              5035            JMP     WRITE_BLOCK     ;YES, DO IT
                         5036
F881                     5037    W2:                             ;COMMAND NOT DEFINED
F881 B480                5038            MOV     AH,080H         ;ERROR, UNDEFINED OPERATION
F883 F9                  5039            STC                     ;ERROR FLAG
F884 C3                  5040            RET
                         5041    W1      ENDP
                         5042
```

```
LOC OBJ                 LINE    SOURCE


F885                    5043    MOTOR_ON        PROC    NEAR
                        5044    ;--------------------------------
                        5045    ; PURPOSE:
                        5046    ;  TO TURN ON CASSETTE MOTOR
                        5047    ;--------------------------------
F885 E461               5048            IN      AL,PORT_B               ;READ CASSETTE OUTPUT
F887 24F7               5049            AND     AL,NOT 08H              ; CLEAR BIT TO TURN ON MOTOR
F889 E661               5050    W3:     OUT     PORT_B,AL               ;WRITE IT OUT
F88B 2AE4               5051            SUB     AH,AH                   ;CLEAR AH
F88D C3                 5052            RET
                        5053    MOTOR_ON        ENDP
                        5054
F88E                    5055    MOTOR_OFF       PROC    NEAR
                        5056    ;--------------------------------
                        5057    ; PURPOSE:
                        5058    ;  TO TURN CASSETTE MOTOR OFF
                        5059    ;--------------------------------
F88E E461               5060            IN      AL,PORT_B               ;READ CASSETTE OUTPUT
F890 0C08               5061            OR      AL,08H                  ; SET BIT TO TURN OFF
F892 EBF5               5062            JMP     W3                      ;WRITE IT, CLEAR ERROR, RETURN
                        5063    MOTOR_OFF       ENDP
F894                    5064    READ_BLOCK      PROC    NEAR
                        5065    ;-----------------------------------------
                        5066    ; PURPOSE:
                        5067    ;  TO READ 1 OR MORE 256 BYTE BLOCKS FROM CASSETTE
                        5068    ;
                        5069    ; ON ENTRY:
                        5070    ;  ES IS SEGMENT FOR MEMORY BUFFER (FOR COMPACT CODE)
                        5071    ;  BX POINTS TO START OF MEMORY BUFFER
                        5072    ;  CX CONTAINS NUMBER OF BYTES TO READ
                        5073    ; ON EXIT:
                        5074    ;  BX POINTS 1 BYTE PAST LAST BYTE PUT IN MEM
                        5075    ;  CX CONTAINS DECREMENTED BYTE COUNT
                        5076    ;  DX CONTAINS NUMBER OF BYTES ACTUALLY READ
                        5077    ;
                        5078    ;  CARRY FLAG IS CLEAR IF NO ERROR DETECTED
                        5079    ;  CARRY FLAG IS SET IF CRC ERROR DETECTED
                        5080    ;-----------------------------------------
F894 53                 5081            PUSH    BX                      ;SAVE BX
F895 51                 5082            PUSH    CX                      ;SAVE CX
F896 56                 5083            PUSH    SI                      ; SAVE SI
F897 BE0700             5084            MOV     SI, 7                   ; SET UP RETRY COUNT FOR LEADER
F89A E8C201             5085            CALL    BEGIN_OP                ; BEGIN BY STARTING MOTOR
F89D                    5086    W4:                                     ; SEARCH FOR LEADER
F89D E462               5087            IN      AL,PORT_C               ;GET INTIAL VALUE
F89F 2410               5088            AND     AL,010H                 ;MASK OFF EXTRANEOUS BITS
F8A1 A26B00         R   5089            MOV     LAST_VAL,AL             ;SAVE IN LOC LAST_VAL
F8A4 BA7A3F             5090            MOV     DX,16250                ; # OF TRANSITIONS TO LOOK FOR
                        5091
F8A7                    5092    W5:                                     ; WAIT_FOR_EDGE
F8A7 F606710080     R   5093            TEST    BIOS_BREAK, 80H         ; CHECK FOR BREAK KEY
F8AC 7403               5094            JZ      W6                      ; JUMP IF NO BREAK KEY
F8AE E98A00             5095            JMP     W17                     ; JUMP IF BREAK KEY HIT
                        5096
F8B1 4A                 5097    W6:     DEC     DX
F8B2 7503               5098            JNZ     W7                      ; JUMP IF BEGINNING OF LEADER
F8B4 E98400             5099            JMP     W17                     ; JUMP IF NO LEADER FOUND
                        5100
F8B7 E8C600             5101    W7:     CALL    READ_HALF_BIT           ;IGNORE FIRST EDGE
F8BA E3EB               5102            JCXZ    W5                      ; JUMP IF NO EDGE DETECTED
F8BC BA7803             5103            MOV     DX,0378H                ; CHECK FOR HALF BITS
F8BF B90002             5104            MOV     CX,200H                 ;MUST HAVE AT LEAST THIS MANY ONE SIZE
                        5105                                            ;PULSES BEFORE CHECKNG FOR SYNC BIT (0)
F8C2 E421               5106            IN      AL, 021H                ; INTERRUPT MASK REGISTER
F8C4 0C01               5107            OR      AL,1                    ; DISABLE TIMER INTERRUPTS
F8C6 E621               5108            OUT     021H, AL
F8C8                    5109    W8:                                     ; SEARCH-LDR
F8C8 F606710080     R   5110            TEST    BIOS_BREAK, 80H         ; CHECK FOR BREAK KEY
F8CD 756C               5111            JNZ     W17                     ; JUMP IF BREAK KEY HIT
F8CF 51                 5112            PUSH    CX                      ;SAVE REG CX
F8D0 E8AD00             5113            CALL    READ_HALF_BIT           ;GET PULSE WIDTH
F8D3 0BC9               5114            OR      CX, CX                  ; CHECK FOR TRANSITION
F8D5 59                 5115            POP     CX                      ;RESTORE ONE BIT COUNTER
F8D6 74C5               5116            JZ      W4                      ; JUMP IF NO TRANSITION
F8D8 3BD3               5117            CMP     DX,BX                   ;CHECK PULSE WIDTH
```

```
LOC OBJ                 LINE    SOURCE


F8DA E304               5118            JCXZ    W9                      ;IF CX=0 THEN WE CAN LOOK
                        5119                                            ;FOR SYNC BIT (0)
F8DC 73BF               5120            JNC     W4                      ; JUMP IF ZERO BIT (NOT GOOD LEADER)
F8DE E2E8               5121            LOOP    W8                      ;DEC CX AND READ ANOTHER HALF ONE BIT
F8E0                    5122    W9:                                     ; FIND-SYNC
F8E0 72E6               5123            JC      W8                      ; JUMP IF ONE BIT (STILL LEADER)
                        5124
                        5125    ; A SYNCH BIT HAS BEEN FOUND.  READ SYN CHARACTER:
                        5126
F8E2 E89B00             5127            CALL    READ_HALF_BIT           ;SKIP OTHER HALF OF SYNC BIT (0)
F8E5 E86A00             5128            CALL    READ_BYTE               ; READ SYN BYTE
F8E8 3C16               5129            CMP     AL, 16H                 ; SYNCHRONIZATION CHARACTER
F8EA 7549               5130            JNE     W16                     ; JUMP IF BAD LEADER FOUND.
                        5131
                        5132    ;------ GOOD CRC SO READ DATA BLOCK(S)
F8EC 5E                 5133            POP     SI                      ; RESTORE REGS
F8ED 59                 5134            POP     CX
F8EE 5B                 5135            POP     BX
                        5136    ;---------------------------------------------
                        5137    ;  READ 1 OR MORE 256 BYTE BLOCKS FROM CASSETTE
                        5138    ;
                        5139    ; ON ENTRY:
                        5140    ;  ES IS SEGMENT FOR MEMORY BUFFER (FOR COMPACT CODE)
                        5141    ;  BX POINTS TO START OF MEMORY BUFFER
                        5142    ;  CX CONTAINS NUMBER OF BYTES TO READ
                        5143    ; ON EXIT:
                        5144    ;  BX POINTS 1 BYTE PAST LAST BYTE PUT IN MEM
                        5145    ;  CX CONTAINS DECREMENTED BYTE COUNT
                        5146    ;  DX CONTAINS NUMBER OF BYTES ACTUALLY READ
                        5147    ;---------------------------------------------
F8EF 51                 5148            PUSH    CX                      ;SAVE BYTE COUNT
F8F0                    5149    W10:                                    ;COME HERE BEFORE EACH
                        5150                                            ;256 BYTE BLOCK IS READ
F8F0 C7066900FFFF   R   5151            MOV     CRC_REG,0FFFFH          ;INIT CRC REG
F8F6 BA0001             5152            MOV     DX,256                  ;SET DX TO DATA BLOCK SIZE
F8F9                    5153    W11:                                    ; RD_BLK
F8F9 F606710080     R   5154            TEST    BIOS_BREAK, 80H         ; CHECK FOR BREAK KEY
F8FE 7523               5155            JNZ     W13                     ; JUMP IF BREAK KEY HIT
F900 E84F00             5156            CALL    READ_BYTE               ;READ BYTE FROM CASSETTE
F903 721E               5157            JC      W13                     ;CY SET INDICATES NO DATA TRANSITIONS
F905 E305               5158            JCXZ    W12                     ;IF WE'VE ALREADY REACHED
                        5159                                            ;END OF MEMORY BUFFER
                        5160                                            ;SKIP REST OF BLOCK
F907 268807             5161            MOV     ES:[BX],AL              ;STORE DATA BYTE AT BYTE PTR
F90A 43                 5162            INC     BX                      ;INC BUFFER PTR
F90B 49                 5163            DEC     CX                      ;DEC BYTE COUNTER
F90C                    5164    W12:                    ; LOOP UNTIL DATA BLOCK HAS BEEN READ FROM CASSETTE.
F90C 4A                 5165            DEC     DX                      ;DEC BLOCK CNT
F90D 7FEA               5166            JG      W11                     ; RD_BLK
F90F E84000             5167            CALL    READ_BYTE               ;NOW READ TWO CRC BYTES
F912 E83D00             5168            CALL    READ_BYTE
F915 2AE4               5169            SUB     AH,AH                   ;CLEAR AH
F917 813E69000F1D   R   5170            CMP     CRC_REG,1D0FH           ;IS THE CRC CORRECT
F91D 7506               5171            JNE     W14                     ;IF NOT EQUAL CRC IS BAD
F91F E306               5172            JCXZ    W15                     ;IF BYTE COUNT IS ZERO
                        5173                                            ;THEN WE HAVE READ ENOUGH
                        5174                                            ;SO WE WILL EXIT
F921 EBCD               5175            JMP     W10                     ;STILL MORE, SO READ ANOTHER BLOCK
F923                    5176    W13:                                    ;MISSING-DATA
                        5177                                            ;NO DATA TRANSITIONS SO
F923 B401               5178            MOV     AH,01H                  ;SET AH=02 TO INDICATE
                        5179                                            ;DATA TIMEOUT
F925                    5180    W14:                                    ; BAD-CRC
F925 FEC4               5181            INC     AH                      ;EXIT EARLY ON ERROR
                        5182                                            ;SET AH=01 TO INDICATE CRC ERROR
F927                    5183    W15:                                    ; RD-BLK-EX
F927 5A                 5184            POP     DX                      ;CALCULATE COUNT OF
F928 2BD1               5185            SUB     DX,CX                   ;DATA BYTES ACTUALLY READ
                        5186                                            ;RETURN COUNT IN REG DX
F92A 50                 5187            PUSH    AX                      ;SAVE AX (RET CODE)
F92B F6C403             5188            TEST    AH, 03H                 ; CHECK FOR ERRORS
F92E 7513               5189            JNZ     W18                     ; JUMP IF ERROR DETECTED
F930 E81F00             5190            CALL    READ_BYTE               ;READ TRAILER
F933 EB0E               5191            JMP     SHORT W18               ;SKIP TO TURN OFF MOTOR
F935                    5192    W16:                                    ; BAD-LEADER
F935 4E                 5193            DEC     SI                      ; CHECK RETRIES
```

```
LOC OBJ                 LINE    SOURCE

F936 7403               5194            JZ      W17                     ; JUMP IF TOO MANY RETRIES
F938 E962FF             5195            JMP     W4                      ; JUMP IF NOT TOO MANY RETRIES
F93B                    5196    W17:                                    ; NO VALID DATA FOUND
                        5197    ;------ NO DATA FROM CASSETTE ERROR, I.E. TIMEOUT
                        5198
F93B 5E                 5199            POP     SI                      ; RESTORE REGS
F93C 59                 5200            POP     CX                      ;RESTORE REGS
F93D 5B                 5201            POP     BX
F93E 2BD2               5202            SUB     DX,DX                   ;ZERO NUMBER OF BYTES READ
F940 B404               5203            MOV     AH,04H                  ;TIME OUT ERROR (NO LEADER)
F942 50                 5204            PUSH    AX
F943                    5205    W18:                                    ; MOT-OFF
F943 E421               5206            IN      AL, 021H                ; RE_ENABLE INTERRUPTS
F945 24FE               5207            AND     AL, 0FFH- 1
F947 E621               5208            OUT     021H, AL
F949 E842FF             5209            CALL    MOTOR_OFF               ;TURN OFF MOTOR
F94C 58                 5210            POP     AX                      ;RESTORE RETURN CODE
F94D 80FC01             5211            CMP     AH,01H                  ;SET CARRY IF ERROR (AH>0)
F950 F5                 5212            CMC
F951 C3                 5213            RET                             ;FINISHED
                        5214    READ_BLOCK      ENDP
                        5215    ;--------------------------------------------
F952                    5216    READ_BYTE       PROC    NEAR
                        5217    ; PURPOSE:
                        5218    ;  TO READ A BYTE FROM CASSETTE
                        5219    ;
                        5220    ; ON EXIT REG AL CONTAINS READ DATA BYTE
                        5221    ;--------------------------------------------
F952 53                 5222            PUSH    BX                      ;SAVE REGS BX,CX
F953 51                 5223            PUSH    CX
F954 B108               5224            MOV     CL,8H                   ; SET BIT COUNTER FOR 8 BITS
F956                    5225    W19:                                    ; BYTE-ASM
F956 51                 5226            PUSH    CX                      ;SAVE CX
                        5227    ;--------------------------------------
                        5228    ;  READ DATA BIT FROM CASSETTE
                        5229    ;--------------------------------------
F957 E82600             5230            CALL    READ_HALF_BIT           ;READ ONE PULSE
F95A E320               5231            JCXZ    W21                     ;IF CX=0 THEN TIMEOUT
                        5232                                            ;BECAUSE OF NO DATA TRANSITIONS
F95C 53                 5233            PUSH    BX                      ;SAVE 1ST HALF BIT'S
                        5234                                            ;PULSE WIDTH (IN BX)
F95D E82000             5235            CALL    READ_HALF_BIT           ;READ COMPLEMENTARY PULSE
F960 58                 5236            POP     AX                      ;COMPUTE DATA BIT
F961 E319               5237            JCXZ    W21                     ;IF CX=0 THEN TIMEOUT DUE TO
                        5238                                            ;NO DATA TRANSITIONS
F963 03D8               5239            ADD     BX,AX                   ;PERIOD
F965 81FBF006           5240            CMP     BX, 06F0H               ; CHECK FOR ZERO BIT
F969 F5                 5241            CMC                             ; CARRY IS SET IF ONE BIT
F96A 9F                 5242            LAHF                            ;SAVE CARRY IN AH
F96B 59                 5243            POP     CX                      ;RESTORE CX
                        5244                                            ;NOTE:
                        5245                                            ; MS BIT OF BYTE IS READ FIRST.
                        5246                                            ; REG CH IS SHIFTED LEFT WITH
                        5247                                            ; CARRY BEING INSERTED INTO LS
                        5248                                            ;   BIT OF CH.
                        5249                                            ; AFTER ALL 8 BITS HAVE BEEN
                        5250                                            ;   READ, THE MS BIT OF THE DATA BYTE
                        5251                                            ;   WILL BE IN THE MS BIT OF REG CH
F96C D0D5               5252            RCL     CH,1                    ;ROTATE REG CH LEFT WITH CARRY TO
                        5253                                            ;   LS BIT OF REG CH
F96E 9E                 5254            SAHF                            ;RESTORE CARRY FOR CRC ROUTINE
F96F E8D900             5255            CALL    CRC_GEN                 ;GENERATE CRC FOR BIT
F972 FEC9               5256            DEC     CL                      ;LOOP TILL ALL 8 BITS OF DATA
                        5257                                            ;ASSEMBLED IN REG CH
F974 75E0               5258            JNZ     W19                     ; BYTE_ASM
F976 8AC5               5259            MOV     AL,CH                   ;RETURN DATA BYTE IN REG AL
F978 F8                 5260            CLC
F979                    5261    W20:                                    ; RD-BYT-EX
F979 59                 5262            POP     CX                      ;RESTORE REGS CX,BX
F97A 5B                 5263            POP     BX
F97B C3                 5264            RET                             ;FINISHED
F97C                    5265    W21:                                    ; NO-DATA
F97C 59                 5266            POP     CX                      ;RESTORE CX
F97D F9                 5267            STC                             ;INDICATE ERROR
```

```
LOC OBJ                 LINE    SOURCE


F97E EBF9               5268            JMP     W20                     ; RD_BYT_EX
                        5269    READ_BYTE       ENDP
                        5270    ;-----------------------------------------
F980                    5271    READ_HALF_BIT   PROC    NEAR
                        5272    ; PURPOSE:
                        5273    ;  TO COMPUTE TIME TILL NEXT DATA
                        5274    ;  TRANSITION (EDGE)
                        5275    ;
                        5276    ; ON ENTRY:
                        5277    ;  EDGE_CNT CONTAINS LAST EDGE COUNT
                        5278    ;
                        5279    ; ON EXIT:
                        5280    ;  AX CONTAINS OLD LAST EDGE COUNT
                        5281    ;  BX CONTAINS PULSE WIDTH (HALF BIT)
                        5282    ;---------------------------------------
F980 B96400             5283            MOV     CX, 100                 ; SET TIME TO WAIT FOR BIT
F983 8A266B00     R     5284            MOV     AH,LAST_VAL             ;GET PRESENT INPUT VALUE
F987                    5285    W22:                                    ; RD-H-BIT
F987 E462               5286            IN      AL,PORT_C               ;INPUT DATA BIT
F989 2410               5287            AND     AL,010H                 ;MASK OFF EXTRANEOUS BITS
F98B 3AC4               5288            CMP     AL,AH                   ;SAME AS BEFORE?
F98D E1F8               5289            LOOPE   W22                     ;LOOP TILL IT CHANGES
F98F A26B00       R     5290            MOV     LAST_VAL,AL             ;UPDATE LAST_VAL WITH NEW VALUE
F992 B000               5291            MOV     AL,0                    ;READ TIMER'S COUNTER COMMAND
F994 E643               5292            OUT     TIM_CTL,AL              ;LATCH COUNTER
F996 E440               5293            IN      AL,TIMER0               ;GET LS BYTE
F998 8AE0               5294            MOV     AH,AL                   ;SAVE IN AH
F99A E440               5295            IN      AL,TIMER0               ;GET MS BYTE
F99C 86C4               5296            XCHG    AL,AH                   ;XCHG AL,AH
F99E 8B1E6700     R     5297            MOV     BX,EDGE_CNT             ;BX GETS LAST EDGE COUNT
F9A2 2BD8               5298            SUB     BX,AX                   ;SET BX EQUAL TO HALF BIT PERIOD
F9A4 A36700       R     5299            MOV     EDGE_CNT,AX             ;UPDATE EDGE COUNT;
F9A7 C3                 5300            RET
                        5301    READ_HALF_BIT   ENDP
                        5302    ;-----------------------------------------
F9A8                    5303    WRITE_BLOCK     PROC    NEAR
                        5304    ;
                        5305    ; WRITE 1 OR MORE 256 BYTE BLOCKS TO CASSETTE.
                        5306    ;       THE DATA IS PADDED TO FILL OUT THE LAST 256 BYTE BLOCK.
                        5307    ;
                        5308    ;  ON ENTRY:
                        5309    ;   BX POINTS TO MEMORY BUFFER ADDRESS
                        5310    ;   CX CONTAINS NUMBER OF BYTES TO WRITE
                        5311    ;
                        5312    ;  ON EXIT:
                        5313    ;   BX POINTS 1 BYTE PAST LAST BYTE WRITTEN TO CASSETTE
                        5314    ;   CX IS ZERO
                        5315    ;-----------------------------------------------
F9A8 53                 5316            PUSH    BX
F9A9 51                 5317            PUSH    CX
F9AA E461               5318            IN      AL,PORT_B               ;DISABLE SPEAKER
F9AC 24FD               5319            AND     AL,NOT 02H
F9AE 0C01               5320            OR      AL, 01H                 ; ENABLE TIMER
F9B0 E661               5321            OUT     PORT_B,AL
F9B2 B0B6               5322            MOV     AL,0B6H                 ; SET UP TIMER -- MODE 3 SQUARE WAVE
F9B4 E643               5323            OUT     TIM_CTL,AL
F9B6 E8A600             5324            CALL    BEGIN_OP                ; START MOTOR AND DELAY
F9B9 B8A004             5325            MOV     AX,1184                 ; SET NORMAL BIT SIZE
F9BC E88500             5326            CALL    W31                     ; SET_TIMER
F9BF B90008             5327            MOV     CX,0800H                ;SET CX FOR LEADER BYTE COUNT
F9C2                    5328    W23:                                    ; WRITE LEADER
F9C2 F9                 5329            STC                             ; WRITE ONE BITS
F9C3 E86800             5330            CALL    WRITE_BIT               ;
F9C6 E2FA               5331            LOOP    W23                     ; LOOP 'TIL LEADER IS WRITTEN
F9C8 F8                 5332            CLC                             ;WRITE SYNC BIT (0)
F9C9 E86200             5333            CALL    WRITE_BIT
F9CC 59                 5334            POP     CX                      ;RESTORE REGS CX,BX
F9CD 5B                 5335            POP     BX
F9CE B016               5336            MOV     AL, 16H                 ; WRITE SYN CHARACTER
F9D0 E84400             5337            CALL    WRITE_BYTE              ;
```

```
LOC OBJ                 LINE    SOURCE

                        5338    ;-----------------------------------------
                        5339    ;  WRITE 1 OR MORE 256 BYTE BLOCKS TO CASSETTE
                        5340    ;
                        5341    ;  ON ENTRY:
                        5342    ;    BX POINTS TO MEMORY BUFFER ADDRESS
                        5343    ;    CX CONTAINS NUMBER OF BYTES TO WRITE
                        5344    ;
                        5345    ;  ON EXIT:
                        5346    ;    BX POINTS 1 BYTE PAST LAST BYTE WRITTEN TO CASSETTE
                        5347    ;    CX IS ZERO
                        5348    ;-----------------------------------------
F9D3                    5349    WR_BLOCK:
F9D3 C7066900FFFF   R   5350            MOV     CRC_REG,0FFFFH          ;INIT CRC
F9D9 BA0001             5351            MOV     DX,256                  ;FOR 256 BYTES
F9DC                    5352    W24:                                    ; WR-BLK
F9DC 268A07             5353            MOV     AL,ES:[BX]              ;READ BYTE FROM MEM
F9DF E83500             5354            CALL    WRITE_BYTE              ;WRITE IT TO CASSETTE
F9E2 E302               5355            JCXZ    W25                     ;UNLESS CX=0, ADVANCE PTRS & DEC COUNT
F9E4 43                 5356            INC     BX                      ;INC BUFFER POINTER
F9E5 49                 5357            DEC     CX                      ;DEC BYTE COUNTER
F9E6                    5358    W25:                                    ; SKIP-ADV
F9E6 4A                 5359            DEC     DX                      ;DEC BLOCK CNT
F9E7 7FF3               5360            JG      W24                     ;LOOP TILL 256 BYTE BLOCK
                        5361                                            ;  IS WRITTEN TO TAPE
                        5362    ;------------------ WRITE CRC --------------
                        5363    ;  WRITE 1'S COMPLEMENT OF CRC REG TO CASSETTE
                        5364    ;  WHICH IS CHECKED FOR CORRECTNESS WHEN THE BLOCK IS READ
                        5365    ;
                        5366    ; REG AX IS MODIFIED
                        5367    ;-----------------------------------------
F9E9 A16900         R   5368            MOV     AX,CRC_REG              ;WRITE THE ONE'S COMPLEMENT OF THE
                        5369                                            ;  TWO BYTE CRC TO TAPE
F9EC F7D0               5370            NOT     AX                      ;FOR 1'S COMPLEMENT
F9EE 50                 5371            PUSH    AX                      ;SAVE IT
F9EF 86E0               5372            XCHG    AH,AL                   ;WRITE MS BYTE FIRST
F9F1 E82300             5373            CALL    WRITE_BYTE              ;WRITE IT
F9F4 58                 5374            POP     AX                      ;GET IT BACK
F9F5 E81F00             5375            CALL    WRITE_BYTE              ;NOW WRITE LS BYTE
F9F8 0BC9               5376            OR      CX,CX                   ;IS BYTE COUNT EXHAUSTED?
F9FA 75D7               5377            JNZ     WR_BLOCK                ;JUMP IF NOT DONE YET
F9FC 51                 5378            PUSH    CX                      ;SAVE REG CX
F9FD B92000             5379            MOV     CX, 32                  ;WRITE OUT TRAILER BITS
FA00                    5380    W26:                                    ; TRAIL-LOOP
FA00 F9                 5381            STC
FA01 E82A00             5382            CALL    WRITE_BIT
FA04 E2FA               5383            LOOP    W26                     ; WRITE UNTIL TRAILER WRITTEN
FA06 59                 5384            POP     CX                      ;RESTORE REG CX
FA07 B0B0               5385            MOV     AL, 0B0H                ; TURN TIMER2 OFF
FA09 E643               5386            OUT     TIM_CTL, AL
FA0B B80100             5387            MOV     AX, 1
FA0E E83300             5388            CALL    W31                     ; SET_TIMER
FA11 E87AFE             5389            CALL    MOTOR_OFF               ;TURN MOTOR OFF
FA14 2BC0               5390            SUB     AX,AX                   ;NO ERRORS REPORTED ON WRITE OP
FA16 C3                 5391            RET                             ;FINISHED
                        5392    WRITE_BLOCK     ENDP
                        5393    ;-----------------------------------------
FA17                    5394    WRITE_BYTE      PROC    NEAR
                        5395    ;  WRITE A BYTE TO CASSETTE.
                        5396    ;  BYTE TO WRITE IS IN REG AL.
                        5397    ;-----------------------------------------
FA17 51                 5398            PUSH    CX                      ;SAVE REGS CX,AX
FA18 50                 5399            PUSH    AX
FA19 8AE8               5400            MOV     CH,AL                   ;AL=BYTE TO WRITE.
                        5401                                            ;   (MS BIT WRITTEN FIRST)
FA1B B108               5402            MOV     CL,8                    ;FOR 8 DATA BITS IN BYTE.
                        5403                                            ;   NOTE: TWO EDGES PER BIT
FA1D                    5404    W27:                                    ; DISASSEMBLE THE DATA BIT
FA1D D0D5               5405            RCL     CH,1                    ;ROTATE MS BIT INTO CARRY
FA1F 9C                 5406            PUSHF                           ;SAVE FLAGS.
                        5407                                            ;   NOTE: DATA BIT IS IN CARRY
FA20 E80B00             5408            CALL    WRITE_BIT               ;WRITE DATA BIT
FA23 9D                 5409            POPF                            ;RESTORE CARRY FOR CRC CALC
FA24 E82400             5410            CALL    CRC_GEN                 ;COMPUTE CRC ON DATA BIT
FA27 FEC9               5411            DEC     CL                      ;LOOP TILL ALL 8 BITS DONE
FA29 75F2               5412            JNZ     W27                     ; JUMP IF NOT DONE YET
FA2B 58                 5413            POP     AX                      ;RESTORE REGS AX,CX
FA2C 59                 5414            POP     CX
```

```
LOC OBJ                 LINE    SOURCE

FA2D C3                 5415            RET                             ;WE ARE FINISHED
                        5416    WRITE_BYTE      ENDP
                        5417    ;-----------------------------------------
FA2E                    5418    WRITE_BIT       PROC    NEAR
                        5419    ; PURPOSE:
                        5420    ;
                        5421    ;  TO WRITE A DATA BIT TO CASSETTE
                        5422    ;  CARRY FLAG CONTAINS DATA BIT
                        5423    ;  I.E. IF SET   DATA BIT IS A ONE
                        5424    ;       IF CLEAR DATA BIT IS A ZERO
                        5425    ;
                        5426    ;  NOTE: TWO EDGES ARE WRITTEN PER BIT
                        5427    ;        ONE BIT  HAS 500 USEC BETWEEN EDGES
                        5428    ;            FOR A 1000 USEC PERIOD (1 MILLISEC)
                        5429    ;
                        5430    ;        ZERO BIT HAS 250 USEC BETWEEN EDGES
                        5431    ;            FOR A  500 USEC PERIOD (.5 MILLISEC)
                        5432    ; CARRY FLAG IS DATA BIT
                        5433    ;-----------------------------------------
                        5434                                            ;ASSUME IT'S A '1'
FA2E B8A004             5435            MOV     AX,1184                 ; SET AX TO NOMINAL ONE SIZE
FA31 7203               5436            JC      W28                     ; JUMP IF ONE BIT
FA33 B85002             5437            MOV     AX,592                  ; NO, SET TO NOMINAL ZERO SIZE
FA36                    5438    W28:                                    ; WRITE-BIT-AX
FA36 50                 5439            PUSH    AX                      ;WRITE BIT WITH PERIOD EQ TO VALUE AX
FA37                    5440    W29:
FA37 E462               5441            IN      AL,PORT_C               ;INPUT TIMER_0 OUTPUT
FA39 2420               5442            AND     AL,020H
FA3B 74FA               5443            JZ      W29                     ;LOOP TILL HIGH
FA3D                    5444    W30:
FA3D E462               5445            IN      AL,PORT_C               ;NOW WAIT TILL TIMER'S OUTPUT IS LOW
FA3F 2420               5446            AND     AL,020H
FA41 75FA               5447            JNZ     W30
                        5448                                            ;RELOAD TIMER WITH PERIOD
                        5449                                            ;FOR NEXT DATA BIT
FA43 58                 5450            POP     AX                      ;RESTORE PERIOD COUNT
FA44                    5451    W31:                                    ; SET TIMER
FA44 E642               5452            OUT     042H, AL                ; SET LOW BYTE OF TIMER 2
FA46 8AC4               5453            MOV     AL, AH
FA48 E642               5454            OUT     042H, AL                ; SET HIGH BYTE OF TIMER 2
FA4A C3                 5455            RET
                        5456    WRITE_BIT       ENDP
                        5457    ;-----------------------------------------
FA4B                    5458    CRC_GEN         PROC    NEAR
                        5459    ;  UPDATE CRC REGISTER WITH NEXT DATA BIT
                        5460    ;
                        5461    ;  CRC IS USED TO DETECT READ ERRORS
                        5462    ;
                        5463    ;  ASSUMES DATA BIT IS IN CARRY
                        5464    ;
                        5465    ;  REG AX IS MODIFIED
                        5466    ;  FLAGS ARE MODIFIED
                        5467    ;-----------------------------------------
FA4B A16900         R   5468            MOV     AX,CRC_REG
                        5469                                            ;THE FOLLOWING INSTUCTIONS
                        5470                                            ;WILL SET THE OVERFLOW FLAG
                        5471                                            ;IF CARRY AND MS BIT OF CRC
                        5472                                            ;ARE UNEQUAL
FA4E D1D8               5473            RCR     AX,1
FA50 D1D0               5474            RCL     AX,1
FA52 F8                 5475            CLC                             ;CLEAR CARRY
FA53 7104               5476            JNO     W32                     ;SKIP IF NO OVERFLOW
                        5477                                            ;IF DATA BIT XORED WITH
                        5478                                            ; CRC REG BIT 15 IS ONE
FA55 351008             5479            XOR     AX,0810H                ;THEN XOR CRC REG WITH
                        5480                                            ; 0810H
FA58 F9                 5481            STC                             ;SET CARRY
FA59                    5482    W32:
FA59 D1D0               5483            RCL     AX,1                    ;ROTATE CARRY (DATA BIT)
                        5484                                            ;INTO CRC REG
FA5B A36900         R   5485            MOV     CRC_REG,AX              ;UPDATE CRC_REG
FA5E C3                 5486            RET                             ;FINISHED
                        5487    CRC_GEN         ENDP
                        5488    ;-----------------------------------------
FA5F                    5489    BEGIN_OP        PROC    NEAR            ; START TAPE AND DELAY
                        5490    ;
                        5491    ;-----------------------------------------
```

```
LOC OBJ                  LINE    SOURCE

FA5F E823FE              5492            CALL    MOTOR_ON                ;TURN ON MOTOR
FA62 B342                5493            MOV     BL,42H                  ;DELAY FOR TAPE DRIVE
                         5494                                            ;TO GET UP TO SPEED  (1/2 SEC)
FA64                     5495    W33:
FA64 B90007              5496            MOV     CX,700H                 ;INNER LOOP= APPROX. 10 MILLISEC
FA67 E2FE                5497    W34:    LOOP    W34
FA69 FECB                5498            DEC     BL
FA6B 75F7                5499            JNZ     W33
FA6D C3                  5500            RET
                         5501    BEGIN_OP        ENDP
                         5502    ;----------------------------------------
                         5503    ; CHARACTER GENERATOR GRAPHICS FOR 320X200 AND 640X200 GRAPHICS
                         5504    ;----------------------------------------
FA6E                     5505    CRT_CHAR_GEN    LABEL   BYTE
FA6E 0000000000000000    5506            DB      000H,000H,000H,000H,000H,000H,000H,000H ; D_00
FA76 7E81A581BD99817E    5507            DB      07EH,081H,0A5H,081H,0BDH,099H,081H,07EH ; D_01
FA7E 7EFFDBFFC3E7FF7E    5508            DB      07EH,0FFH,0DBH,0FFH,0C3H,0E7H,0FFH,07EH ; D_02
FA86 6CFEFEFE7C381000    5509            DB      06CH,0FEH,0FEH,0FEH,07CH,038H,010H,000H ; D_03
FA8E 10387CFE7C381008    5510            DB      010H,038H,07CH,0FEH,07CH,038H,010H,008H ; D_04
FA96 387C38FEFE7C387C    5511            DB      038H,07CH,038H,0FEH,0FEH,07CH,038H,07CH ; D_05
FA9E 1010387CFE7C387C    5512            DB      010H,010H,038H,07CH,0FEH,07CH,038H,07CH ; D_06
FAA6 0000183C3C180000    5513            DB      000H,000H,018H,03CH,03CH,018H,000H,000H ; D_07
FAAE FFFFE7C3C3E7FFFF    5514            DB      0FFH,0FFH,0E7H,0C3H,0C3H,0E7H,0FFH,0FFH ; D_08
FAB6 003C664242663C00    5515            DB      000H,03CH,066H,042H,042H,066H,03CH,000H ; D_09
FABE FFC399BDBD99C3FF    5516            DB      0FFH,0C3H,099H,0BDH,0BDH,099H,0C3H,0FFH ; D_0A
FAC6 0F070F7DCCCCCC78    5517            DB      00FH,007H,00FH,07DH,0CCH,0CCH,0CCH,078H ; D_0B
FACE 3C6666663C187E18    5518            DB      03CH,066H,066H,066H,03CH,018H,07EH,018H ; D_0C
FAD6 3F333F303070F0E0    5519            DB      03FH,033H,03FH,030H,030H,070H,0F0H,0E0H ; D_0D
FADE 7F637F636367E6C0    5520            DB      07FH,063H,07FH,063H,063H,067H,0E6H,0C0H ; D_0E
FAE6 995A3CE7E73C5A99    5521            DB      099H,05AH,03CH,0E7H,0E7H,03CH,05AH,099H ; D_0F
                         5522
FAEE 80E0F8FEF8E08000    5523            DB      080H,0E0H,0F8H,0FEH,0F8H,0E0H,080H,000H ; D_10
FAF6 020E3EFE3E0E0200    5524            DB      002H,00EH,03EH,0FEH,03EH,00EH,002H,000H ; D_11
FAFE 183C7E18187E3C18    5525            DB      018H,03CH,07EH,018H,018H,07EH,03CH,018H ; D_12
FB06 6666666666006600    5526            DB      066H,066H,066H,066H,066H,000H,066H,000H ; D_13
FB0E 7FDBDB7B1B1B1B00    5527            DB      07FH,0DBH,0DBH,07BH,01BH,01BH,01BH,000H ; D_14
FB16 3E63386C6C38CC78    5528            DB      03EH,063H,038H,06CH,06CH,038H,0CCH,078H ; D_15
FB1E 000000007E7E7E00    5529            DB      000H,000H,000H,000H,07EH,07EH,07EH,000H ; D_16
FB26 183C7E187E3C18FF    5530            DB      018H,03CH,07EH,018H,07EH,03CH,018H,0FFH ; D_17
FB2E 183C7E1818181800    5531            DB      018H,03CH,07EH,018H,018H,018H,018H,000H ; D_18
FB36 181818187E3C1800    5532            DB      018H,018H,018H,018H,07EH,03CH,018H,000H ; D_19
FB3E 00180CFE0C180000    5533            DB      000H,018H,00CH,0FEH,00CH,018H,000H,000H ; D_1A
FB46 003060FE60300000    5534            DB      000H,030H,060H,0FEH,060H,030H,000H,000H ; D_1B
FB4E 0000C0C0C0FE0000    5535            DB      000H,000H,0C0H,0C0H,0C0H,0FEH,000H,000H ; D_1C
FB56 002466FF66240000    5536            DB      000H,024H,066H,0FFH,066H,024H,000H,000H ; D_1D
FB5E 00183C7EFFFF0000    5537            DB      000H,018H,03CH,07EH,0FFH,0FFH,000H,000H ; D_1E
FB66 00FFFF7E3C180000    5538            DB      000H,0FFH,0FFH,07EH,03CH,018H,000H,000H ; D_1F
                         5539
FB6E 0000000000000000    5540            DB      000H,000H,000H,000H,000H,000H,000H,000H ; SP D_20
FB76 3078783030003000    5541            DB      030H,078H,078H,030H,030H,000H,030H,000H ; ! D_21
FB7E 6C6C6C0000000000    5542            DB      06CH,06CH,06CH,000H,000H,000H,000H,000H ; " D_22
FB86 6C6CFE6CFE6C6C00    5543            DB      06CH,06CH,0FEH,06CH,0FEH,06CH,06CH,000H ; # D_23
FB8E 307CC0780CF83000    5544            DB      030H,07CH,0C0H,078H,00CH,0F8H,030H,000H ; $ D_24
FB96 00C6CC183066C600    5545            DB      000H,0C6H,0CCH,018H,030H,066H,0C6H,000H ; PER CENT D_25
FB9E 386C3876DCCC7600    5546            DB      038H,06CH,038H,076H,0DCH,0CCH,076H,000H ; & D_26
FBA6 6060C00000000000    5547            DB      060H,060H,0C0H,000H,000H,000H,000H,000H ; ' D_27
FBAE 1830606060301800    5548            DB      018H,030H,060H,060H,060H,030H,018H,000H ; ( D_28
FBB6 6030181818306000    5549            DB      060H,030H,018H,018H,018H,030H,060H,000H ; ) D_29
FBBE 00663CFF3C660000    5550            DB      000H,066H,03CH,0FFH,03CH,066H,000H,000H ; * D_2A
FBC6 003030FC30300000    5551            DB      000H,030H,030H,0FCH,030H,030H,000H,000H ; + D_2B
FBCE 0000000000303060    5552            DB      000H,000H,000H,000H,000H,030H,030H,060H ; , D_2C
FBD6 000000FC00000000    5553            DB      000H,000H,000H,0FCH,000H,000H,000H,000H ; - D_2D
FBDE 0000000000303000    5554            DB      000H,000H,000H,000H,000H,030H,030H,000H ; . D_2E
FBE6 060C183060C08000    5555            DB      006H,00CH,018H,030H,060H,0C0H,080H,000H ; / D_2F
                         5556
FBEE 7CC6CEDEF6E67C00    5557            DB      07CH,0C6H,0CEH,0DEH,0F6H,0E6H,07CH,000H ; 0 D_30
FBF6 307030303030FC00    5558            DB      030H,070H,030H,030H,030H,030H,0FCH,000H ; 1 D_31
FBFE 78CC0C3860CCFC00    5559            DB      078H,0CCH,00CH,038H,060H,0CCH,0FCH,000H ; 2 D_32
FC06 78CC0C380CCC7800    5560            DB      078H,0CCH,00CH,038H,00CH,0CCH,078H,000H ; 3 D_33
FC0E 1C3C6CCCFE0C1E00    5561            DB      01CH,03CH,06CH,0CCH,0FEH,00CH,01EH,000H ; 4 D_34
FC16 FCC0F80C0CCC7800    5562            DB      0FCH,0C0H,0F8H,00CH,00CH,0CCH,078H,000H ; 5 D_35
FC1E 3860C0F8CCCC7800    5563            DB      038H,060H,0C0H,0F8H,0CCH,0CCH,078H,000H ; 6 D_36
FC26 FCCC0C1830303000    5564            DB      0FCH,0CCH,00CH,018H,030H,030H,030H,000H ; 7 D_37
FC2E 78CCCC78CCCC7800    5565            DB      078H,0CCH,0CCH,078H,0CCH,0CCH,078H,000H ; 8 D_38
FC36 78CCCC7C0C187000    5566            DB      078H,0CCH,0CCH,07CH,00CH,018H,070H,000H ; 9 D_39
FC3E 0030300000303000    5567            DB      000H,030H,030H,000H,000H,030H,030H,000H ; : D_3A
```

```
LOC OBJ                  LINE    SOURCE

FC46 0030300000303060    5568            DB      000H,030H,030H,000H,000H,030H,030H,060H ; ; D_3B
FC4E 183060C060301800    5569            DB      018H,030H,060H,0C0H,060H,030H,018H,000H ; < D_3C
FC56 0000FC0000FC0000    5570            DB      000H,000H,0FCH,000H,000H,0FCH,000H,000H ; = D_3D
FC5E 6030180C18306000    5571            DB      060H,030H,018H,00CH,018H,030H,060H,000H ; > D_3E
FC66 78CC0C1830003000    5572            DB      078H,0CCH,00CH,018H,030H,000H,030H,000H ; ? D_3F
                         5573
FC6E 7CC6DEDEDEC07800    5574            DB      07CH,0C6H,0DEH,0DEH,0DEH,0C0H,078H,000H ; @ D_40
FC76 3078CCCCFCCCCC00    5575            DB      030H,078H,0CCH,0CCH,0FCH,0CCH,0CCH,000H ; A D_41
FC7E FC66667C6666FC00    5576            DB      0FCH,066H,066H,07CH,066H,066H,0FCH,000H ; B D_42
FC86 3C66C0C0C0663C00    5577            DB      03CH,066H,0C0H,0C0H,0C0H,066H,03CH,000H ; C D_43
FC8E F86C6666666CF800    5578            DB      0F8H,06CH,066H,066H,066H,06CH,0F8H,000H ; D D_44
FC96 FE6268786862FE00    5579            DB      0FEH,062H,068H,078H,068H,062H,0FEH,000H ; E D_45
FC9E FE6268786860F000    5580            DB      0FEH,062H,068H,078H,068H,060H,0F0H,000H ; F D_46
FCA6 3C66C0C0CE663E00    5581            DB      03CH,066H,0C0H,0C0H,0CEH,066H,03EH,000H ; G D_47
FCAE CCCCCCFCCCCCCC00    5582            DB      0CCH,0CCH,0CCH,0FCH,0CCH,0CCH,0CCH,000H ; H D_48
FCB6 7830303030307800    5583            DB      078H,030H,030H,030H,030H,030H,078H,000H ; I D_49
FCBE 1E0C0C0CCCCC7800    5584            DB      01EH,00CH,00CH,00CH,0CCH,0CCH,078H,000H ; J D_4A
FCC6 E6666C786C66E600    5585            DB      0E6H,066H,06CH,078H,06CH,066H,0E6H,000H ; K D_4B
FCCE F06060606266FE00    5586            DB      0F0H,060H,060H,060H,062H,066H,0FEH,000H ; L D_4C
FCD6 C6EEFEFED6C6C600    5587            DB      0C6H,0EEH,0FEH,0FEH,0D6H,0C6H,0C6H,000H ; M D_4D
FCDE C6E6F6DECEC6C600    5588            DB      0C6H,0E6H,0F6H,0DEH,0CEH,0C6H,0C6H,000H ; N D_4E
FCE6 386CC6C6C66C3800    5589            DB      038H,06CH,0C6H,0C6H,0C6H,06CH,038H,000H ; O D_4F
                         5590
FCEE FC66667C6060F000    5591            DB      0FCH,066H,066H,07CH,060H,060H,0F0H,000H ; P D_50
FCF6 78CCCCCCDC781C00    5592            DB      078H,0CCH,0CCH,0CCH,0DCH,078H,01CH,000H ; Q D_51
FCFE FC66667C6C66E600    5593            DB      0FCH,066H,066H,07CH,06CH,066H,0E6H,000H ; R D_52
FD06 78CCE0701CCC7800    5594            DB      078H,0CCH,0E0H,070H,01CH,0CCH,078H,000H ; S D_53
FD0E FCB4303030307800    5595            DB      0FCH,0B4H,030H,030H,030H,030H,078H,000H ; T D_54
FD16 CCCCCCCCCCCCFC00    5596            DB      0CCH,0CCH,0CCH,0CCH,0CCH,0CCH,0FCH,000H ; U D_55
FD1E CCCCCCCCCC783000    5597            DB      0CCH,0CCH,0CCH,0CCH,0CCH,078H,030H,000H ; V D_56
FD26 C6C6C6D6FEEEC600    5598            DB      0C6H,0C6H,0C6H,0D6H,0FEH,0EEH,0C6H,000H ; W D_57
FD2E C6C66C38386CC600    5599            DB      0C6H,0C6H,06CH,038H,038H,06CH,0C6H,000H ; X D_58
FD36 CCCCCC7830307800    5600            DB      0CCH,0CCH,0CCH,078H,030H,030H,078H,000H ; Y D_59
FD3E FEC68C183266FE00    5601            DB      0FEH,0C6H,08CH,018H,032H,066H,0FEH,000H ; Z D_5A
FD46 7860606060607800    5602            DB      078H,060H,060H,060H,060H,060H,078H,000H ; [ D_5B
FD4E C06030180C060200    5603            DB      0C0H,060H,030H,018H,00CH,006H,002H,000H ; BACKSLASH D_5C
FD56 7818181818187800    5604            DB      078H,018H,018H,018H,018H,018H,078H,000H ; ] D_5D
FD5E 10386CC600000000    5605            DB      010H,038H,06CH,0C6H,000H,000H,000H,000H ; CIRCUMFLEX D_5E
FD66 00000000000000FF    5606            DB      000H,000H,000H,000H,000H,000H,000H,0FFH ; _ D_5F
                         5607
FD6E 3030180000000000    5608            DB      030H,030H,018H,000H,000H,000H,000H,000H ;   D_60
FD76 0000780C7CCC7600    5609            DB      000H,000H,078H,00CH,07CH,0CCH,076H,000H ; LOWER CASE A D_61
FD7E E060607C6666DC00    5610            DB      0E0H,060H,060H,07CH,066H,066H,0DCH,000H ; L.C. B D_62
FD86 000078CCC0CC7800    5611            DB      000H,000H,078H,0CCH,0C0H,0CCH,078H,000H ; L.C. C D_63
FD8E 1C0C0C7CCCCC7600    5612            DB      01CH,00CH,00CH,07CH,0CCH,0CCH,076H,000H ; L.C. D D_64
FD96 000078CCFCC07800    5613            DB      000H,000H,078H,0CCH,0FCH,0C0H,078H,000H ; L.C. E D_65
FD9E 386C60F06060F000    5614            DB      038H,06CH,060H,0F0H,060H,060H,0F0H,000H ; L.C. F D_66
FDA6 000076CCCC7C0CF8    5615            DB      000H,000H,076H,0CCH,0CCH,07CH,00CH,0F8H ; L.C. G D_67
FDAE E0606C766666E600    5616            DB      0E0H,060H,06CH,076H,066H,066H,0E6H,000H ; L.C. H D_68
FDB6 3000703030307800    5617            DB      030H,000H,070H,030H,030H,030H,078H,000H ; L.C. I D_69
FDBE 0C000C0C0CCCCC78    5618            DB      00CH,000H,00CH,00CH,00CH,0CCH,0CCH,078H ; L.C. J D_6A
FDC6 E060666C786CE600    5619            DB      0E0H,060H,066H,06CH,078H,06CH,0E6H,000H ; L.C. K D_6B
FDCE 7030303030307800    5620            DB      070H,030H,030H,030H,030H,030H,078H,000H ; L.C. L D_6C
FDD6 0000CCFEFED6C600    5621            DB      000H,000H,0CCH,0FEH,0FEH,0D6H,0C6H,000H ; L.C. M D_6D
FDDE 0000F8CCCCCCCC00    5622            DB      000H,000H,0F8H,0CCH,0CCH,0CCH,0CCH,000H ; L.C. N D_6E
FDE6 000078CCCCCC7800    5623            DB      000H,000H,078H,0CCH,0CCH,0CCH,078H,000H ; L.C. O D_6F
                         5624
FDEE 0000DC66667C60F0    5625            DB      000H,000H,0DCH,066H,066H,07CH,060H,0F0H ; L.C. P D_70
FDF6 000076CCCC7C0C1E    5626            DB      000H,000H,076H,0CCH,0CCH,07CH,00CH,01EH ; L.C. Q D_71
FDFE 0000DC766660F000    5627            DB      000H,000H,0DCH,076H,066H,060H,0F0H,000H ; L.C. R D_72
FE06 00007CC0780CF800    5628            DB      000H,000H,07CH,0C0H,078H,00CH,0F8H,000H ; L.C. S D_73
FE0E 10307C3030341800    5629            DB      010H,030H,07CH,030H,030H,034H,018H,000H ; L.C. T D_74
FE16 0000CCCCCCCC7600    5630            DB      000H,000H,0CCH,0CCH,0CCH,0CCH,076H,000H ; L.C. U D_75
FE1E 0000CCCCCC783000    5631            DB      000H,000H,0CCH,0CCH,0CCH,078H,030H,000H ; L.C. V D_76
FE26 0000C6D6FEFE6C00    5632            DB      000H,000H,0C6H,0D6H,0FEH,0FEH,06CH,000H ; L.C. W D_77
FE2E 0000C66C386CC600    5633            DB      000H,000H,0C6H,06CH,038H,06CH,0C6H,000H ; L.C. X D_78
FE36 0000CCCCCC7C0CF8    5634            DB      000H,000H,0CCH,0CCH,0CCH,07CH,00CH,0F8H ; L.C. Y D_79
FE3E 0000FC983064FC00    5635            DB      000H,000H,0FCH,098H,030H,064H,0FCH,000H ; L.C. Z D_7A
FE46 1C3030E030301C00    5636            DB      01CH,030H,030H,0E0H,030H,030H,01CH,000H ;   D_7B
FE4E 1818180018181800    5637            DB      018H,018H,018H,000H,018H,018H,018H,000H ;   D_7C
FE56 E030301C3030E000    5638            DB      0E0H,030H,030H,01CH,030H,030H,0E0H,000H ;   D_7D
FE5E 76DC000000000000    5639            DB      076H,0DCH,000H,000H,000H,000H,000H,000H ;   D_7E
FE66 0010386CC6C6FE00    5640            DB      000H,010H,038H,06CH,0C6H,0C6H,0FEH,000H ; DELTA D_7F
```

```
LOC OBJ                 LINE    SOURCE

                        5641    ;--- INT 1A --------------------------------
                        5642    ; TIME_OF_DAY
                        5643    ;  THIS ROUTINE ALLOWS THE CLOCK TO BE SET/READ
                        5644    ;
                        5645    ; INPUT
                        5646    ;   (AH) = 0    READ THE CURRENT CLOCK SETTING
                        5647    ;               RETURNS CX = HIGH PORTION OF COUNT
                        5648    ;                       DX = LOW PORTION OF COUNT
                        5649    ;                       AL = 0 IF TIMER HAS NOT PASSED 24 HOURS SINCE LAST READ
                        5650    ;                          <>0 IF ON ANOTHER DAY
                        5651    ;   (AH) = 1    SET THE CURRENT CLOCK
                        5652    ;       CX = HIGH PORTION OF COUNT
                        5653    ;       DX = LOW PORTION OF COUNT
                        5654    ; NOTE: COUNTS OCCUR AT THE RATE OF 1193180/65536 COUNTS/SEC
                        5655    ;       (OR ABOUT 18.2 PER SECOND -- SEE EQUATES BELOW)
                        5656    ;--------------------------------------------
                        5657            ASSUME  CS:CODE,DS:DATA
FE6E                    5658    TIME_OF_DAY     PROC    FAR
FE6E FB                 5659            STI                     ; INTERRUPTS BACK ON
FE6F 1E                 5660            PUSH    DS              ; SAVE SEGMENT
FE70 50                 5661            PUSH    AX              ; SAVE PARM
FE71 B84000         R   5662            MOV     AX,DATA
FE74 8ED8               5663            MOV     DS,AX           ; ESTABLISH ADDRESSING TO VALUES
FE76 58                 5664            POP     AX              ; GET BACK INPUT PARM
FE77 0AE4               5665            OR      AH,AH           ; AH=0
FE79 7407               5666            JZ      T2              ; READ_TIME
FE7B FECC               5667            DEC     AH              ; AH=1
FE7D 7416               5668            JZ      T3              ; SET_TIME
FE7F                    5669    T1:                             ; TOD_RETURN
FE7F FB                 5670            STI                     ; INTERRUPTS BACK ON
FE80 1F                 5671            POP     DS              ; RECOVER SEGMENT
FE81 CF                 5672            IRET                    ; RETURN TO CALLER
                        5673
FE82                    5674    T2:                             ; READ_TIME
FE82 FA                 5675            CLI                     ; NO TIMER INTERRUPTS WHILE READING
FE83 A07000         R   5676            MOV     AL,TIMER_OFL
FE86 C606700000     R   5677            MOV     TIMER_OFL,0     ; GET OVERFLOW, AND RESET THE FLAG
FE8B 8B0E6E00       R   5678            MOV     CX,TIMER_HIGH
FE8F 8B166C00       R   5679            MOV     DX,TIMER_LOW
FE93 EBEA               5680            JMP     T1                      ; TOD_RETURN
                        5681
FE95                    5682    T3:                             ; SET_TIME
FE95 FA                 5683            CLI                     ; NO INTERRUPTS WHILE WRITING
FE96 89166C00       R   5684            MOV     TIMER_LOW,DX
FE9A 890E6E00       R   5685            MOV     TIMER_HIGH,CX   ; SET THE TIME
FE9E C606700000     R   5686            MOV     TIMER_OFL,0     ; RESET OVERFLOW
FEA3 EBDA               5687            JMP     T1              ; TOD_RETURN
                        5688    TIME_OF_DAY     ENDP
                        5689    ;--------------------------------------------
                        5690    ; THIS ROUTINE HANDLES THE TIMER INTERRUPT FROM
                        5691    ; CHANNEL 0 OF THE 8253 TIMER.  INPUT FREQUENCY IS 1.19318 MHZ
                        5692    ; AND THE DIVISOR IS 65536, RESULTING IN APPROX. 18.2 INTERRUPTS
                        5693    ; EVERY SECOND.
                        5694    ;
                        5695    ; THE INTERRUPT HANDLER MAINTAINS A COUNT OF INTERRUPTS SINCE POWER
                        5696    ;  ON TIME, WHICH MAY BE USED TO ESTABLISH TIME OF DAY.
                        5697    ; THE INTERRUPT HANDLER ALSO DECREMENTS THE MOTOR CONTROL COUNT
                        5698    ;  OF THE DISKETTE, AND WHEN IT EXPIRES, WILL TURN OFF THE DISKETTE
                        5699    ;  MOTOR, AND RESET THE MOTOR RUNNING FLAGS
                        5700    ; THE INTERRUPT HANDLER WILL ALSO INVOKE A USER ROUTINE THROUGH INTERRUPT
                        5701    ;  1CH AT EVERY TIME TICK.  THE USER MUST CODE A ROUTINE AND PLACE THE
                        5702    ;  CORRECT ADDRESS IN THE VECTOR TABLE.
                        5703    ;--------------------------------------------
FEA5                    5704    TIMER_INT       PROC    FAR
FEA5 FB                 5705            STI                     ; INTERRUPTS BACK ON
FEA6 1E                 5706            PUSH    DS
FEA7 50                 5707            PUSH    AX
FEA8 52                 5708            PUSH    DX              ; SAVE MACHINE STATE
FEA9 B84000         R   5709            MOV     AX,DATA
FEAC 8ED8               5710            MOV     DS,AX           ; ESTABLISH ADDRESSABILITY
FEAE FF066C00       R   5711            INC     TIMER_LOW       ; INCREMENT TIME
FEB2 7504               5712            JNZ     T4              ; TEST_DAY
FEB4 FF066E00       R   5713            INC     TIMER_HIGH      ; INCREMENT HIGH WORD OF TIME
FEB8                    5714    T4:                             ; TEST_DAY
```

```
LOC OBJ                  LINE    SOURCE


FEB8 833E6E0018     R    5715            CMP     TIMER_HIGH,018H ; TEST FOR COUNT EQUALLING 24 HOURS
FEBD 7519                5716            JNZ     T5              ; DISKETTE_CTL
FEBF 813E6C00B000   R    5717            CMP     TIMER_LOW,0B0H
FEC5 7511                5718            JNZ     T5              ; DISKETTE_CTL
                         5719
                         5720    ;------ TIMER HAS GONE 24 HOURS
                         5721
FEC7 C7066E000000   R    5722            MOV     TIMER_HIGH,0
FECD C7066C000000   R    5723            MOV     TIMER_LOW,0
FED3 C606700001     R    5724            MOV     TIMER_OFL,1
                         5725
                         5726    ;------ TEST FOR DISKETTE TIME OUT
                         5727
FED8                     5728    T5:                             ; DISKETTE_CTL
FED8 FE0E4000       R    5729            DEC     MOTOR_COUNT
FEDC 750B                5730            JNZ     T6              ; RETURN IF COUNT NOT OUT
FEDE 80263F00F0     R    5731            AND     MOTOR_STATUS,0F0H       ; TURN OFF MOTOR RUNNING BITS
FEE3 B00C                5732            MOV     AL,0CH
FEE5 BAF203              5733            MOV     DX,03F2H                ; FDC CTL PORT
FEE8 EE                  5734            OUT     DX,AL           ; TURN OFF THE MOTOR
                         5735
FEE9                     5736    T6:                             ; TIMER_RET:
FEE9 CD1C                5737            INT     1CH             ; TRANSFER CONTROL TO A USER ROUTINE
FEEB B020                5738            MOV     AL,EOI
FEED E620                5739            OUT     020H,AL         ; END OF INTERRUPT TO 8259
FEEF 5A                  5740            POP     DX
FEF0 58                  5741            POP     AX
FEF1 1F                  5742            POP     DS              ; RESET MACHINE STATE
FEF2 CF                  5743            IRET                    ; RETURN FROM INTERRUPT
                         5744    TIMER_INT       ENDP
                         5745    ;--------------------------------------------
                         5746    ; THESE ARE THE VECTORS WHICH ARE MOVED INTO
                         5747    ;  THE 8086 INTERRUPT AREA DURING POWER ON
                         5748    ;--------------------------------------------
FEF3                     5749    VECTOR_TABLE    LABEL   WORD            ; VECTOR TABLE FOR MOVE TO INTERRUPTS
                         5750
FEF3 A5FE           R    5751            DW      OFFSET TIMER_INT        ; INTERRUPT 8
FEF5 00F0           R    5752            DW      CODE
                         5753
FEF7 87E9           R    5754            DW      OFFSET  KB_INT          ; INTERRUPT 9
FEF9 00F0           R    5755            DW      CODE
                         5756
FEFB 00000000            5757            DD      0                       ; INTERRUPT A
FEFF 00000000            5758            DD      0                       ; INTERRUPT B
FF03 00000000            5759            DD      0                       ; INTERRUPT C
FF07 00000000            5760            DD      0                       ; INTERRUPT D
                         5761
FF0B 57EF           R    5762            DW      OFFSET DISK_INT         ; INTERRUPT E
FF0D 00F0           R    5763            DW      CODE
                         5764
FF0F 00000000            5765            DD      0                       ; INTERRUPT F
                         5766
FF13 65F0           R    5767            DW      OFFSET VIDEO_IO         ; INTERRUPT 10H
FF15 00F0           R    5768            DW      CODE
                         5769
FF17 4DF8           R    5770            DW      OFFSET EQUIPMENT        ; INTERRUPT 11H
FF19 00F0           R    5771            DW      CODE
                         5772
FF1B 41F8           R    5773            DW      OFFSET MEMORY_SIZE_DETERMINE    ; INT 12H
FF1D 00F0           R    5774            DW      CODE
                         5775
FF1F 59EC           R    5776            DW      OFFSET DISKETTE_IO      ; INTERRUPT 13H
FF21 00F0           R    5777            DW      CODE
                         5778
FF23 39E7           R    5779            DW      OFFSET RS232_IO         ; INTERRUPT 14H
FF25 00F0           R    5780            DW      CODE
                         5781
FF27 59F8           R    5782            DW      OFFSET CASSETTE_IO      ; INTERRUPT 15H
FF29 00F0           R    5783            DW      CODE
                         5784
FF2B 2EE8           R    5785            DW      OFFSET KEYBOARD_IO      ; INTERRUPT 16H
FF2D 00F0           R    5786            DW      CODE
                         5787
FF2F D2EF           R    5788            DW      OFFSET PRINTER_IO       ; INTERRUPT 17H
FF31 00F0           R    5789            DW      CODE
                         5790
```

```
LOC OBJ                 LINE    SOURCE

FF33 0000               5791            DW      00000H                  ; INTERRUPT 18H
FF35 00F6               5792            DW      0F600H                  ;  ROM BASIC ENTRY POINT
                        5793
FF37 F2E6       R       5794            DW      OFFSET BOOT_STRAP       ; INTERRUPT 19H
FF39 00F0       R       5795            DW      CODE
                        5796
FF3B 6EFE       R       5797            DW      TIME_OF_DAY             ; INTERRUPT 1AH -- TIME OF DAY
FF3D 00F0       R       5798            DW      CODE
                        5799
FF3F 53FF       R       5800            DW      DUMMY_RETURN            ; INTERRUPT 1BH -- KEYBOARD BREAK ADDR
FF41 00F0       R       5801            DW      CODE
                        5802
FF43 53FF       R       5803            DW      DUMMY_RETURN            ; INTERRUPT 1C -- TIMER BREAK ADDR
FF45 00F0       R       5804            DW      CODE
                        5805
FF47 A4F0       R       5806            DW      VIDEO_PARMS             ; INTERRUPT 1D -- VIDEO PARAMETERS
FF49 00F0       R       5807            DW      CODE
                        5808
FF4B C7EF       R       5809            DW      OFFSET DISK_BASE        ; INTERRUPT 1E -- DISK PARMS
FF4D 00F0       R       5810            DW      CODE
                        5811
FF4F 00000000           5812            DD      0                       ; INTERRUPT 1F -- POINTER TO VIDEO EXT
                        5813
FF53                    5814    DUMMY_RETURN:
FF53 CF                 5815            IRET                            ; DUMMY RETURN FOR BREAK FROM KEYBOARD
                        5816    ;-- INT 5 ---------------------------------
                        5817    ;       THIS LOGIC WILL BE INVOKED BY INTERRUPT 05H TO PRINT
                        5818    ;       THE SCREEN. THE CURSOR POSITION AT THE TIME THIS ROUTINE
                        5819    ;       IS INVOKED WILL BE SAVED AND RESTORED UPON COMPLETION. THE
                        5820    ;       ROUTINE IS INTENDED TO RUN WITH INTERRUPTS ENABLED.
                        5821    ;       IF A SUBSEQUENT 'PRINT SCREEN KEY IS DEPRESSED DURING THE
                        5822    ;       TIME THIS ROUTINE IS PRINTING IT WILL BE IGNORED.
                        5823    ;       ADDRESS 50:0 CONTAINS THE STATUS OF THE PRINT SCREEN:
                        5824    ;
                        5825    ;       50:0    =0      EITHER PRINT SCREEN HAS NOT BEEN CALLED
                        5826    ;                       OR UPON RETURN FROM A CALL THIS INDICATES
                        5827    ;                       A SUCCESSFUL OPERATION.
                        5828    ;
                        5829    ;               =1      PRINT SCREEN IS IN PROGRESS
                        5830    ;
                        5831    ;               =377    ERROR ENCOUNTERED DURING PRINTING
                        5832    ;----------------------------------------------
                        5833            ASSUME  CS:CODE,DS:XXDATA
                        5834
FF54                    5835    PRINT_SCREEN    PROC    FAR
FF54 FB                 5836            STI                             ;MUST RUN WITH INTERRUPTS ENABLED
FF55 1E                 5837            PUSH    DS                      ;MUST USE 50:0 FOR DATA AREA STORAGE
FF56 50                 5838            PUSH    AX
FF57 53                 5839            PUSH    BX
FF58 51                 5840            PUSH    CX                      ;WILL USE THIS LATER FOR CURSOR LIMITS
FF59 52                 5841            PUSH    DX                      ;WILL HOLD CURRENT CURSOR POSITION
FF5A B85000             5842            MOV     AX,XXDATA               ;HEX 50
FF5D 8ED8               5843            MOV     DS,AX
FF5F 803E000001         5844            CMP     STATUS_BYTE,1   ;SEE IF PRINT ALREADY IN PROGRESS
FF64 745F               5845            JZ      EXIT            ; JUMP IF PRINT ALREADY IN PROGRESS
FF66 C606000001         5846            MOV     STATUS_BYTE,1   ;INDICATE PRINT NOW IN PROGRESS
FF6B B40F               5847            MOV     AH,15           ;WILL REQUEST THE CURRENT SCREEN MODE
FF6D CD10               5848            INT     10H             ;       [AL]=MODE
                        5849                                    ;       [AH]=NUMBER COLUMNS/LINE
                        5850                                    ;       [BH]=VISUAL PAGE
                        5851            ;************************************
                        5852            ;       AT THIS POINT WE KNOW THE COLUMNS/LINE ARE IN
                        5853            ;       [AX] AND THE PAGE IF APPLICABLE IS IN [BH]. THE STACK
                        5854            ;       HAS DS,AX,BX,CX,DX PUSHED. [AL] HAS VIDEO MODE
                        5855            ;
                        5856            ;************************************
FF6F 8ACC               5857            MOV     CL,AH           ;WILL MAKE USE OF [CX] REGISTER TO
FF71 B519               5858            MOV     CH,25           ;CONTROL ROW & COLUMNS
FF73 E85500             5859            CALL    CRLF            ;CARRIAGE RETURN LINE FEED ROUTINE
FF76 51                 5860            PUSH    CX              ;SAVE SCREEN BOUNDS
FF77 B403               5861            MOV     AH,3            ;WILL NOW READ THE CURSOR.
FF79 CD10               5862            INT     10H             ;AND PRESERVE THE POSITION
FF7B 59                 5863            POP     CX              ;RECALL SCREEN BOUNDS
FF7C 52                 5864            PUSH    DX              ;RECALL [BH]=VISUAL PAGE
FF7D 33D2               5865            XOR     DX,DX           ;WILL SET CURSOR POSITION TO [0,0]
```

```
LOC OBJ                  LINE    SOURCE

                         5866    ;*****************************************
                         5867    ;       THE LOOP FROM PRI10 TO THE INSTRUCTION PRIOR TO PRI20
                         5868    ;       IS THE LOOP TO READ EACH CURSOR POSITION FROM THE SCREEN
                         5869    ;       AND PRINT.
                         5870    ;*****************************************
=F7F B402                5871    PRI10:  MOV     AH,2            ;TO INDICATE CURSOR SET REQUEST
=F81 CD10                5872            INT     10H             ;NEW CURSOR POSITION ESTABLISHED
=F83 B408                5873            MOV     AH,8            ;TO INDICATE READ CHARACTER
=F85 CD10                5874            INT     10H             ;CHARACTER NOW IN [AL]
FF87 0AC0                5875            OR      AL,AL           ;SEE IF VALID CHAR
FF89 7502                5876            JNZ     PRI15           ;JUMP IF VALID CHAR
FF8B B020                5877            MOV     AL,' '          ;MAKE A BLANK
FF8D                     5878    PRI15:
FF8D 52                  5879            PUSH    DX              ;SAVE CURSOR POSITION
FF8E 33D2                5880            XOR     DX,DX           ;INDICATE PRINTER 1
FF90 32E4                5881            XOR     AH,AH           ;TO INDICATE PRINT CHAR IN [AL]
FF92 CD17                5882            INT     17H             ;PRINT THE CHARACTER
FF94 5A                  5883            POP     DX              ;RECALL CURSOR POSITION
FF95 F6C425              5884            TEST    AH, 25H         ; TEST FOR PRINTER ERROR
FF98 7521                5885            JNZ     ERR10           ; JUMP IF ERROR DETECTED
FF9A FEC2                5886            INC     DL              ;ADVANCE TO NEXT COLUMN
FF9C 3ACA                5887            CMP     CL,DL           ;SEE IF AT END OF LINE
FF9E 75DF                5888            JNZ     PRI10           ;IF NOT PROCEED
FFA0 32D2                5889            XOR     DL,DL           ;BACK TO COLUMN 0
FFA2 8AE2                5890            MOV     AH,DL           ;[AH]=0
FFA4 52                  5891            PUSH    DX              ;SAVE NEW CURSOR POSITION
FFA5 E82300              5892            CALL    CRLF            ; LINE FEED CARRIAGE RETURN
FFA8 5A                  5893            POP     DX              ;RECALL CURSOR POSITION
FFA9 FEC6                5894            INC     DH              ;ADVANCE TO NEXT LINE
FFAB 3AEE                5895            CMP     CH,DH           ;FINISHED?
FFAD 75D0                5896            JNZ     PRI10           ;IF NOT CONTINUE
FFAF 5A                  5897    PRI20:  POP     DX              ;RECALL CURSOR POSITION
FFB0 B402                5898            MOV     AH,2            ;TO INDICATE CURSOR SET REQUEST
FFB2 CD10                5899            INT     10H             ;CURSOR POSITION RESTORED
FFB4 C606000000          5900            MOV     STATUS_BYTE,0   ;INDICATE FINISHED
FFB9 EB0A                5901            JMP     SHORT EXIT      ;EXIT THE ROUTINE
FFBB 5A                  5902    ERR10:  POP     DX              ;GET CURSOR POSITION
FFBC B402                5903            MOV     AH,2            ;TO REQUEST CURSOR SET
FFBE CD10                5904            INT     10H             ;CURSOR POSITION RESTORED
FFC0 C6060000FF          5905    ERR20:  MOV     STATUS_BYTE,0FFH        ;INDICATE ERROR
                         5906
FFC5 5A                  5907    EXIT:   POP     DX              ;RESTORE ALL THE REGISTERS USED
FFC6 59                  5908            POP     CX
FFC7 5B                  5909            POP     BX
FFC8 58                  5910            POP     AX
FFC9 1F                  5911            POP     DS
FFCA CF                  5912            IRET
                         5913    PRINT_SCREEN    ENDP
                         5914
                         5915    ;------ CARRIAGE RETURN, LINE FEED SUBROUTINE
                         5916
FFCB                     5917    CRLF    PROC    NEAR
FFCB 33D2                5918            XOR     DX,DX           ;PRINTER 0
FFCD 32E4                5919            XOR     AH,AH           ;WILL NOW SEND INITIAL LF,CR TO PRINTER
FFCF B00A                5920            MOV     AL,12Q          ;LF
FFD1 CD17                5921            INT     17H             ;SEND THE LINE FEED
FFD3 32E4                5922            XOR     AH,AH           ;NOW FOR THE CR
FFD5 B00D                5923            MOV     AL,15Q          ;CR
FFD7 CD17                5924            INT     17H             ;SEND THE CARRIAGE RETURN
FFD9 C3                  5925            RET
                         5926    CRLF    ENDP
                         5927    CODE    ENDS
                         5928
                         5929    ;---------------------------------
                         5930    ; POWER ON RESET VECTOR
                         5931    ;---------------------------------
FFFF                     5932    VECTOR  SEGMENT AT 0FFFFH
                         5933
                         5934    ;------ POWER ON RESET
                         5935
0000 EA5B0000F0     R    5936            JMP     RESET
                         5937
0005 30342F32342F38      5938            DB      '04/24/81'              ; RELEASE MARKER
     31
                         5939    VECTOR  ENDS
                         5940            END
```