Personal Computer
Hardware Reference
Library

Technical Reference

First Edition (August 1986)

The following statement applies to all IBM Personal Computer products unless otherwise indicated by the information referring to that product.

FEDERAL COMMUNICATIONS COMMISSION RADIO FREQUENCY INTERFERENCE STATEMENT

Warning: This equipment has been certified to comply with the limits for a Class B computing device, pursuant to Subpart J of Part 15 of FCC rules. Only peripherals (computer input/output devices, terminals, printers, etc.) certified to comply with the Class B limits may be attached to this computer when this computer is operated in a residential environment. Operation with noncertified peripherals is likely to result in interference to radio and TV reception.

CAUTION

This product is equipped with a 3-wire power cord and plug for the user's safety. Use this power cord in conjunction with a properly grounded electrical outlet to avoid electrical shock.

Notes:

Preface

This manual describes the various units of the IBM Personal Computer XT Model 286 and how they interact. It also has information about the basic input/output system (BIOS) and about programming support.

The information in this publication is for reference, and is intended for hardware and program designers, programmers, engineers, and anyone else who needs to understand the design and operation of the IBM Personal Computer XT Model 286.

This manual consists of eight sections:

- The first three sections describe the IBM Personal Computer XT Model 286 including hardware, charts, and register information
- Section 4 describes keyboard operation, the commands to and from the system, and the various keyboard layouts.
- Section 5 contains information about the usage of BIOS and a system BIOS listing.
- Section 6 contains instruction sets for the 80286 microprocessor and the 80287 math coprocessor.
- Section 7 provides information about characters, keystrokes, and colors.
- Section 8 contains information about the compatibility of the IBM Personal Computer XT Model 286 and the rest of the IBM Personal Computer family.

A glossary, bibliography, and index are included.

Prerequisite Publications

Guide to Operations for the IBM Personal Computer XT Model 286

Suggested Reading

- *BASIC* for the IBM Personal Computer
- *Disk Operating System (DOS)*
- *Macro Assembler* for the IBM Personal Computer

Additional Information

The *Technical Directory* lists all the service and technical information that is available for the IBM Personal Computer family of products. To receive a free copy of the *Technical Directory*, call toll free **1-800-IBM-PCTB**, Monday through Friday, 8:00 a.m. to 8:00 p.m. Eastern Time.

Contents

Notes:

INDEX TAB LISTING

SECTION 1

SECTION 2

SECTION 3

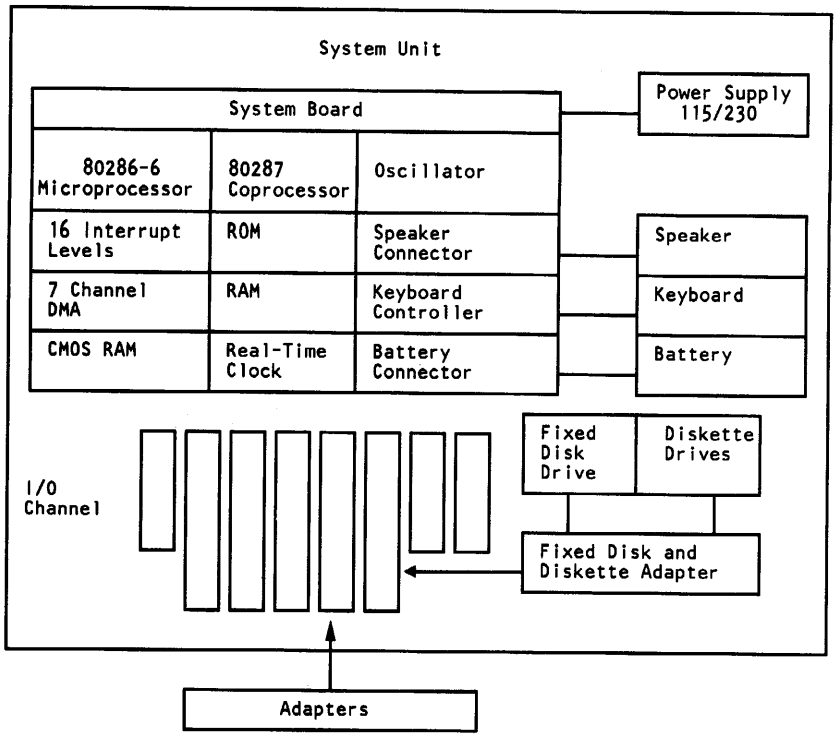
SECTION 4

SECTION 5

SECTION 6

Notes:

System Block Diagram



SECTION 1. SYSTEM BOARD

The system board is approximately 22 by 33.8 centimeters (8.5 by 13.2 inches). It uses very large scale integration (VLSI) technology and has the following components:

- Intel 80286 Microprocessor
- System support function:
  - Seven-Channel Direct Memory Access (DMA)
  - Sixteen-level interrupt
  - Three programmable timers
  - System clock
- 64K read-only memory (ROM) subsystem, expandable to 128K.
- A 640K (may be set to 512K) random-access memory (RAM) subsystem
- Eight input/output (I/O) slots:
  - Five with a 98-pin card-edge socket
  - Three with a 62-pin card-edge socket
- Speaker attachment
- Keyboard attachment
- Complementary metal oxide semiconductor (CMOS) memory RAM to maintain system configuration
- Real-Time Clock
- Battery backup for CMOS configuration table and Real-Time Clock

Memory

The system board consists of two 256K-by-9 random access memory module packages, plus two banks of 64K-by-4 random access memory (RAM) modules. Total memory size is 640K, with parity checking.

Microprocessor

The Intel 80286 microprocessor has a 24-bit address, 16-bit memory interface[1], an extensive instruction set, DMA and interrupt support capabilities, a hardware fixed-point multiply and divide, integrated memory management, four-level memory protection, 1G (1,073,741,824 bytes) of virtual address space for each task, and two operating modes: the 8086-compatible real address mode and the protected virtual address mode. More detailed descriptions of the microprocessor may be found in the publications listed in the Bibliography of this manual.

Real Address Mode

In the real address mode, the microprocessor's physical memory is a contiguous array of up to one megabyte. The microprocessor addresses memory by generating 20-bit physical addresses.

The selector portion of the pointer is interpreted as the upper 16 bits of a 20-bit segment address. The lower 4 bits of the 20-bit segment address are always zero. Therefore, segment addresses begin on multiples of 16 bytes.

All segments in the real address mode are 64K in size and may be read, written, or executed. An exception or interrupt can occur if data operands or instructions attempt to wrap around the end of a segment. An example of this is a word with its low-order byte at offset FFFF and its high-order byte at 0000. If, in the real

[1] In this manual, the term interface refers to a device that carries signals between functional units.

address mode, the information contained in the segment does not use the full 64K, the unused end of the segment may be overlayed by another segment to reduce physical memory requirements.

Protected (Virtual Address) Mode

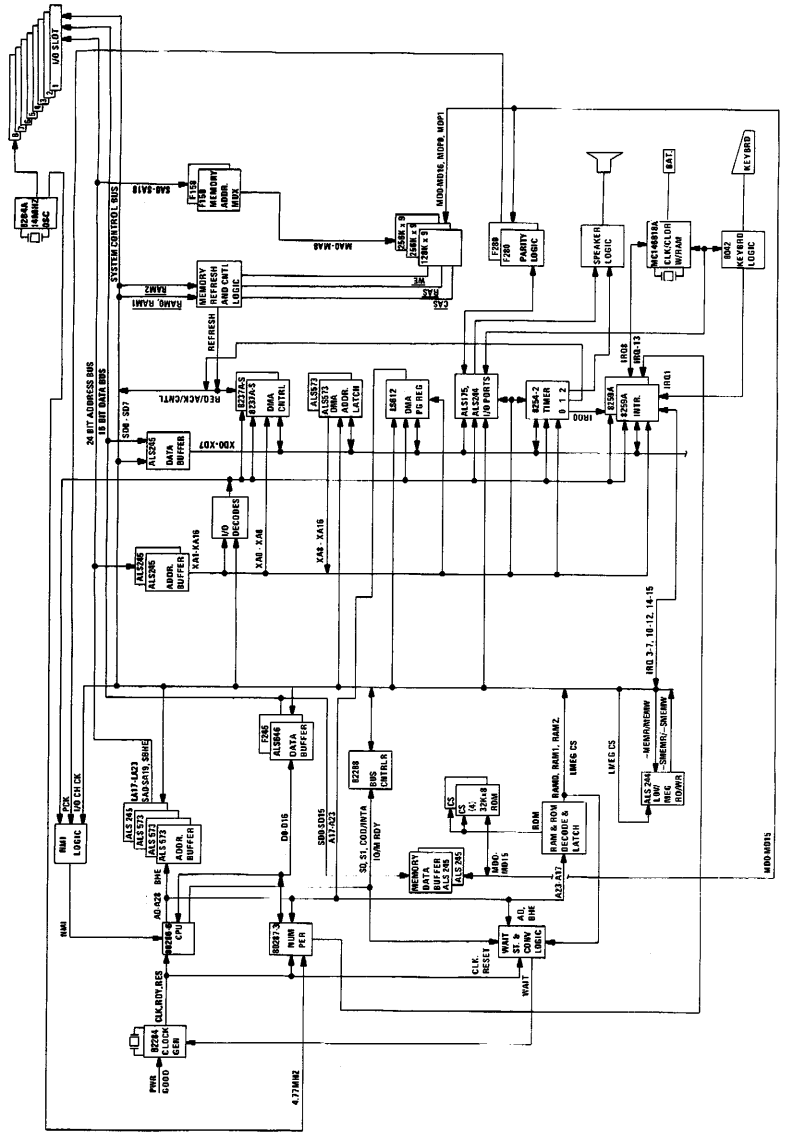
The protected mode offers extended physical and virtual memory address space, memory protection mechanisms, and new operations to support operating systems and virtual memory.

> **Note:** See "BIOS Programming Hints" in Section 5 for special cautions while operating in the protected mode.

The protected mode provides a 1G virtual address space for each task mapped into a 16M physical address space. The virtual address space may be larger than the physical address space, because any use of an address that does not map to a physical memory location will cause a restartable exception.

As in the real address mode, the protected mode uses 32-bit pointers, consisting of 16-bit selector and offset components. The selector, however, specifies an index into a memory resident table rather than the upper 16 bits of a real memory address. The 24-bit base address of the desired segment is obtained from the tables in memory. The 16-bit offset is added to the segment base address to form the physical address. The microprocessor automatically refers to the tables whenever a segment register is loaded with a selector. All instructions that load a segment register will refer to the memory-based tables without additional program support. The memory-based tables contain 8-byte values called *descriptors*.

The following is a block diagram of the system board.



System Performance

The 80286 microprocessor operates at 6 MHz, resulting in a clock cycle time of 167 nanoseconds.

A bus cycle requires two clock cycles, making a 334-nanosecond 16-bit, microprocessor cycle time. Eight-bit bus operations to 8-bit devices take six clock cycles (which include four wait states), resulting in a 1-microsecond microprocessor cycle. Sixteen-bit bus operations to 8-bit devices take 12 clock cycles (which include 10 wait states) resulting in a 2-microsecond microprocessor cycle.

The refresh controller steps one refresh address every 15 microseconds. Each refresh cycle requires eight clock cycles to refresh all of the system's dynamic memory; 256 refresh cycles are required every 4 milliseconds, but the system hardware refreshes every 3.84ms. The following formula determines the percentage of bandwidth used for refresh for the 6 MHz clock.

$$\%\ \text{Bandwidth used for Refresh} = \frac{8 \text{ cycles} \times 256}{3.84\text{ms}/167\text{ns}} = \frac{2048}{22994} = 9\%$$

The DMA controller operates at 3 MHz, which results in a clock cycle time of 334 nanoseconds. All DMA data-transfer bus cycles are five clock cycles or 1.66 microseconds. Cycles spent in the transfer of bus control are not included.

System Memory Mapping

The following shows the mapping of the system memory.

Address	Name	Function
000000 to 07FFFF	512K system board memory	First 512K of system board memory
080000 to 09FFFF	128K system board memory	System board memory (512K to 640K) May be disabled with jumper J10.
0A0000 to 0BFFFF	128K video RAM	Reserved for graphics display buffer
0C0000 to 0DFFFF	128K I/O expansion ROM	Reserved for ROM on I/O adapters
0E0000 to 0EFFFF	64K reserved on system board	Duplicated code assignment at address FE0000
0F0000 to 0FFFFF	64K ROM on the system board	Duplicated code assignment at address FF0000
100000 to FDFFFF	Maximum memory 15M	I/O channel memory - 640K to 15M installed on memory expansion options
FE0000 to FEFFFF	64K reserved on system board	Duplicated code assignment at address 0E0000
FF0000 to FFFFFF	64K ROM on the system board	Duplicated code assignment at address 0F0000

System Memory

Direct Memory Access

The system supports seven direct memory access (DMA) channels. Two Intel 8237A-5 DMA Controller chips are used, with four channels for each chip. The DMA channels are assigned as follows:

Controller 1	Controller 2
Ch 0 - Reserved Ch 1 - SDLC Ch 2 - Diskette Ch 3 - LAN	Ch 4 - Cascade for Ctlr 1 Ch 5 - Reserved Ch 6 - Reserved Ch 7 - Reserved

DMA Channels

DMA controller 1 contains channels 0 through 3. These channels support 8-bit data transfers between 8-bit I/O adapters and 8- or 16-bit system memory. Each channel can transfer data throughout the 16M system-address space in 64K blocks.

The following figures show address generation for the DMA channels.

Source	DMA Page Registers	Controller
Address	A23<----------->A16	A15<----------->A0

Address Generation for DMA Channels 0 through 3

Note: The addressing signal, 'byte high enable' (BHE), is generated by inverting address line A0.

DMA controller 1 command code addresses follow.

Hex Address	Register Function
000	CH0 base and current address
001	CH0 base and current word count
002	CH1 base and current address
003	CH1 base and current word count
004	CH2 base and current address
005	CH2 base and current word count
006	CH3 base and current address
007	CH3 base and current word count
008	Read Status Register/Write Command Register
009	Write Request Register
00A	Write Single Mask Register Bit
00B	Write Mode Register
00C	Clear Byte Pointer Flip-Flop
00D	Read Temporary Register/Write Master Clear
00E	Clear Mask Register
00F	Write All Mask Register Bits

DMA Controller 1 (Channels 0-3)

DMA controller 2 contains channels 4 through 7. Channel 4 is used to cascade channels 0 through 3 to the microprocessor. Channels 5, 6, and 7 support 16-bit data transfers between 16-bit I/O adapters and 16-bit system memory. These DMA channels can transfer data throughout the 16M system-address space in 128K blocks. Channels 5, 6, and 7 cannot transfer data on odd-byte boundaries.

Source	DMA Page Registers	Controller
Address	A23<----------->A17	A16<----------->A1

Address Generation for DMA Channels 5 through 7

Note: The addressing signals, BHE and A0, are forced to a logical 0.

The following figure shows the addresses for the page register.

Page Register	I/O Hex Address
DMA Channel 0	0087
DMA Channel 1	0083
DMA Channel 2	0081
DMA Channel 3	0082
DMA Channel 5	008B
DMA Channel 6	0089
DMA Channel 7	008A
Refresh	008F

Page Register Addresses

Addresses for all DMA channels do not increase or decrease through page boundaries (64K for channels 0 through 3, and 128K for channels 5 through 7).

DMA channels 5 through 7 perform 16-bit data transfers. Access can be gained only to 16-bit devices (I/O or memory) during the DMA cycles of channels 5 through 7. Access to the DMA controller, which controls these channels, is through I/O addresses hex 0C0 through 0DF.

DMA controller 2 command code addresses follow.

Hex Address	Register Function
0C0	CH4 base and current address
0C2	CH4 base and current word count
0C4	CH5 base and current address
0C6	CH5 base and current word count
0C8	CH6 base and current address
0CA	CH6 base and current word count
0CC	CH7 base and current address
0CE	CH7 base and current word count
0D0	Read Status Register/Write Command Register
0D2	Write Request Register
0D4	Write Single Mask Register Bit
0D6	Write Mode Register
0D8	Clear Byte Pointer Flip-Flop
0DA	Read Temporary Register/Write Master Clear
0DC	Clear Mask Register
0DE	Write All Mask Register Bits

DMA Controller 2 (DMA Channels 4-7)

All DMA memory transfers made with channels 5 through 7 must occur on even-byte boundaries. When the base address for these

channels is programmed, the real address divided by 2 is the data written to the base address register. Also, when the base word count for channels 5 through 7 is programmed, the count is the number of 16-bit words to be transferred. Therefore, DMA channels 5 through 7 can transfer 65,536 words, or 128Kb maximum, for any selected page of memory. These DMA channels divide the 16M memory space into 128K pages. When the DMA page registers for channels 5 through 7 are programmed, data bits D7 through D1 contain the high-order seven address bits (A23 through A17) of the desired memory space. Data bit D0 of the page registers for channels 5 through 7 is not used in the generation of the DMA memory address.

At power-on time, all internal locations, especially the mode registers, should be loaded with some valid value. This is done even if some channels are unused.

System Interrupts

The 80286 microprocessor's non maskable interrupt (NMI) and two 8259A controller chips provide 16 levels of system interrupts.

Note: Any or all interrupts may be masked (including the microprocessor's NMI).

Hardware Interrupt Listing

The following shows the interrupt-level assignments in decreasing priority.

Level	Function
Microprocessor NMI	Parity or I/O Channel Check
Interrupt Controllers CTRL 1 CTRL 2	
IRQ 0	Timer Output 0
IRQ 1	Keyboard (Output Buffer Full)
IRQ 2 ←	Interrupt from CTRL 2
IRQ 8	Realtime Clock Interrupt
IRQ 9	Software Redirected to INT 0AH PC Network * PC Network(Alt.) *
IRQ 10	Reserved
IRQ 11	Reserved
IRQ 12	Reserved
IRQ 13	Coprocessor
IRQ 14	Fixed Disk Controller
IRQ 15	Reserved
IRQ 3	Serial Port 2 BSC BSC (Alt.) PC Network * PC Network (Alt.) * SDLC
IRQ 4	Serial Port 1 BSC BSC (Alt.) SDLC
IRQ 5	Parallel Port 2
IRQ 6	Diskette Controller Fixed Disk and Diskette Drive
IRQ 7	Parallel Port 1 Data Acquisition and Control ** GPIB *** Voice Communications Adapter ****

* The PC Network is jumper selectable.
** The Data Acquisition Adapter can be set to interrupts 3 through 7. The default interrupt is 7.
*** The GPIB Adapter can be set to interrupts 2 through 7.
**** The Voice Communications Adapter can be set to interrupts 2, 3, 4, or 7 (Interrupt level 7 is recommended).

Hardware Interrupt Listing

Interrupt Sharing

A definition for standardized hardware design has been established that enables multiple adapters to share an interrupt level. This section describes this design and discusses the programming support required.

> **Note:** Since interrupt routines do not exist in ROM for protected mode operations, this design is intended to run only in the microprocessor's real address mode.

Design Overview

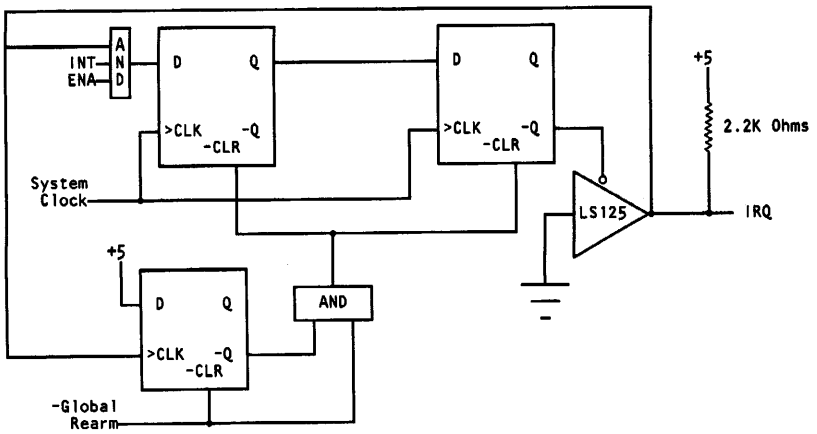
Most interrupt-supporting adapters hold the 'interrupt request' line (IRQ) at a low level and then drive the line high to cause an interrupt. In contrast, the shared-interrupt hardware design allows IRQ to float high through pull-up resistors on each adapter. Each adapter on the line may cause an interrupt by momentarily pulsing the line to a low level. The high-to-low transition arms the 8259A Interrupt Controller; the low-to-high transition generates the interrupt. The duration of this pulse must be between 125 and 1,000 nanoseconds.

The adapters must have an 'interrupt' status bit (INT) and a 'interrupt enable' bit (ENA) that can be controlled and monitored by its software.

Each adapter sharing an interrupt level must monitor the IRQ line. When any adapter drives the line low, all other adapters on that line must be prevented from issuing an interrupt request until they are rearmed.

If an adapter's INT status bit is at a high level when the interrupt sharing logic is rearmed, the adapter must reissue the interrupt. This prevents lost interrupts if two adapters issue an interrupt at the same time and an interrupt handler issues a Global Rearm after servicing one of the adapters.

The following diagram is an example of the shared interrupt logic.



Shared Interrupt Logic Diagram

Program Support

During multitasking, tasks are constantly being activated and deactivated in no particular order. The interrupt-sharing program support described in this section provides for an orderly means to:

- Link a task's interrupt handler to a chain of interrupt handlers
- Share the interrupt level while the task is active
- Unlink the interrupt handler from the chain when the task is deactivated.

Linking to a Chain

Each newly activated task replaces the interrupt vector in low memory with a pointer to its own interrupt handler. The old interrupt vector is used as a forward pointer (FPTR) and is stored at a fixed offset from the new task's interrupt handler.

Sharing the Interrupt Level

When the new task's handler gains control as a result of an interrupt, the handler reads the contents of the adapter's interrupt status register to determine if its adapter caused the interrupt. If it did, the handler services the interrupt, disables the interrupts (CLI), issues a non-specific End of Interrupt (EOI), and then, to rearm the interrupt hardware, writes to address 02F*X*, where *X* corresponds to interrupt levels 3 through 7, and 9 (IRQ9 is 02F2). A write to address 06F*X*, where *X* may be 2 through 7, is required for interrupt levels 10 through 15, respectively. Each adapter in the chain decodes the address which results in a Global Rearm. An adapter is required to decode the least significant 11 bits for this Global Rearm command. The handler then issues a Return From Interrupt (IRET).

If its adapter did not cause the interrupt, the handler passes control to the next interrupt handler in the chain.

Unlinking from the Chain

To unlink from the chain, a task must first locate its handler's position within the chain. By starting at the interrupt vector in low memory, and using the offset of each handler's FPTR to find the entry point of each handler, the chain can be methodically searched until the task finds its own handler. The FPTR of the previous handler in the chain is replaced by the task's FPTR, thus removing the handler from the chain.

Error Recovery

Should the unlinking routine discover that the interrupt chain has been corrupted (an interrupt handler is linked but does not have a valid SIGNATURE), an unlinking error-recovery procedure must be in place. Each application can incorporate its own unlinking error procedure into the unlinking routine. One application may choose to display an error message requiring the operator to either correct the situation or power down the system. Another application may choose an error recovery procedure that restores the original interrupt vector in low memory, and bypasses the corrupt portion of the interrupt chain. This error recovery

procedure may not be suitable when adapters that are being serviced by the corrupt handler are actively generating interrupts, since unserviced interrupts lock up that interrupt level.

ROS Considerations

Adapters with their handlers residing in ROS may choose to implement chaining by storing the 4 byte FPTR (plus the FIRST flag if it is sharing interrupt 7 or 15) in on-adapter latches or ports. Adapter ROS without this feature must first test to see that it is the first in the chain. If it is the first in the chain, the adapter can complete the link; if not, the adapter must exit its routine without linking.

Precautions

The following precautions must be taken when designing hardware or programs using shared interrupts:

- Hardware designers should ensure the adapters:
  - Do not power up with the ENA line active or an interrupt pending.
  - Do not generate interrupts that are not serviced by a handler. Generating interrupts when a handler is not active to service the adapter causes the interrupt level to lock up. The design relies on the handler to clear its adapter's interrupt and issue the Global Rearm.
  - Can be disabled so that they do not remain active after their application has terminated.
- Programmers should:
  - Ensure that their programs have a short routine that can be executed with the AUTOEXEC.BAT to disable their adapter's interrupts. This precaution ensures that the adapters are deactivated if the user reboots the system.
  - Treat words as words, not bytes. Remember that data is stored in memory using the Intel format (word 424B is stored as 4B42).

Interrupt Chaining Structure

```
ENTRY:  JMP          SHORT PAST          ; Jump around structure
        FPTR         DD       0          ; Forward Pointer
        SIGNATURE    DW       424BH      ; Used when unlinking to identify
                                         ;  compatible interrupt handlers
        FLAGS        DB                  ; Flags
        FIRST        EQU      80H        ; Flag for being first in chain
        JMP          SHORT    RESET
        RES_BYTES    DB       DUP 7 (0)  ; Future expansion
PAST:   ...                              ; Actual start of code
```

The interrupt chaining structure is a 16-byte format containing FPTR, SIGNATURE, and RES_BYTES. It begins at the third byte from the interrupt handler's entry point. The first instruction of every handler is a short jump around the structure to the start of the routine. Since the position of each interrupt handler's chaining structure is known (except for the handlers on adapter ROS), the FPTRs can be updated when unlinking.

The FIRST flag is used to determine the handler's position in the chain when unlinking when sharing interrupts 7 and 15. The RESET routine, an entry point for the operating system, must disable the adapter's interrupt and RETURN FAR to the operating system.

> **Note:** All handlers designed for interrupt sharing must use 424B as the signature to avoid corrupting the chain.

Examples

In the following examples, notice that interrupts are disabled before control is passed to the next handler on the chain. The next handler receives control as if a hardware interrupt had caused it to receive control. Also, notice that the interrupts are disabled before the non-specific EOI is issued, and not reenabled in the interrupt handler. This ensures that the IRET is executed (at which point the flags are restored and the interrupts reenabled) before another interrupt is serviced, protecting the stack from excessive build up.

Example of an Interrupt Handler

```
YOUR_CARD  EQU     xxxx              ; Location of your card's interrupt
                                     ;  control/status register
ISB        EQU     xx                ; Interrupt bit in your card's interrupt
                                     ;  control status register
REARM      EQU     2F7H              ; Global Rearm location for interrupt
                                     ;  level 7
SPC_EOI    EQU     67H               ; Specific EOI for 8259's interrupt
                                     ;  level 7
EOI        EQU     20H               ; Non-specific EOI
OCR        EQU     20H               ; Location of 8259 operational control
                                     ;  register
IMR        EQU     21H               ; Location of 8259 interrupt mask
                                     ;  register

MYCSEG     SEGMENT PARA
           ASSUME  CS:MYCSEG,DS:DSEG
ENTRY      PROC    FAR
           JMP     SHORT PAST        ; Entry point of handler
FPTR       DD      0                 ; Forward Pointer
SIGNATURE  DW      424BH             ; Used when unlinking to identify
                                     ;  compatible interrupt handlers
FLAGS      DB      0                 ; Flags
FIRST      EQU     80H
JMP        SHORT   RESET
RES_BYTES  DB      DUP 7 (0)         ; Future expansion
PAST:      STI                       ; Actual start of handler code
           PUSH    ...               ; Save needed registers

           MOV     DX,YOUR_CARD      ; Select your status register
           IN      AL,DX             ; Read the status register
           TEST    AL,ISB            ; Your card caused the interrupt?
           JNZ     SERVICE           ; Yes, branch to service logic
           TEST    CS:FLAGS,FIRST    ; Are we the first ones in?
           JNZ     EXIT              ; If yes, branch for EOI and Rearm
           POP     ...               ; Restore registers
           CLI                       ; Disable interrupts
           JMP     DWORD PTR CS:FPTR ; Pass control to next guy on chain

SERVICE:   ...                       ; Service the interrupt
EXIT:
           CLI                       ; Disable the interrupts
           MOV     AL,EOI
           OUT     OCR,AL            ; Issue non-specific EOI to 8259
           MOV     DX,REARM          ; Rearm the cards
           OUT     DX,AL
           POP     ...               ; Restore registers
           IRET
RESET:     ...                       ; Disable your card
           RET                       ; Return FAR to operating system
ENTRY      ENDP
           MYCSEG  ENDS
           END     ENTRY
```

Linking Code Example

```
          PUSH    ES
          CLI                             ; Disable interrupts
; Set forward pointer to value of interrupt vector in low memory
          ASSUME  CS:CODESEG,DS:CODESEG
          PUSH    ES
          MOV     AX,350FH                ; DOS get interrupt vector
          INT     21H
          MOV     SI,OFFSET CS:FPTR       ; Get offset of your forward pointer
                                          ;  in an indexable register
          MOV     CS:[SI],BX              ; Store the old interrupt vector
          MOV     CS:[SI+2],ES            ;  in your forward pointer for chaining
          CMP     ES:BYTE PTR[BX],CFH ; Test for IRET
          JNZ     SETVECTR
          MOV     CS:FLAGS,FIRST          ; Set up first in chain flag
SETVECTR: POP     ES
          PUSH    DS
; Make interrupt vector in low memory point to your handler
          MOV     DX,OFFSET ENTRY         ; Make interrupt vector point to your handler
          MOV     AX,SEG ENTRY            ; If DS not = CS, get it
          MOV     DS,AX                   ;  and put it in DS
          MOV     AX,250FH                ; DOS set interrupt vector
          INT     21H
          POP     DS
; Unmask (enable) interrupts for your level
          IN      AL,IMR                  ; Read interrupt mask register
          JMP     $+2                     ; IO delay
          AND     AL,07FH                 ; Unmask interrupt level 7
          OUT     IMR,AL                  ; Write new interrupt mask

          MOV     AL,SPC_EOI              ; Issue specific EOI for level 7
          JMP     $+2                     ;  to allow pending level 7 interrupts
OUT       OCR,AL              ;  (if any) to be serviced
          STI                             ; Enable interrupts
          POP     ES                      ;
```

Unlinking Code Example

```
          PUSH    DS
          PUSH    ES
          CLI                         ; Disable interrupts
          MOV     AX,350FH            ; DOS get interrupt vector
          INT     21H                 ; ES:BX points to first of chain
          MOV     CX,ES               ; Pickup segment part of interrupt vector
; Are we the first handler in the chain?
          MOV     AX,CS               ; Get code seg into comparable register
          CMP     BX,OFFSET ENTRY     ; Interrupt vector in low memory
                                      ;  pointing to your handler's offset?

          JNE     UNCHAIN_A           ; No, branch
          CMP     AX,CX               ; Vector pointing to your
                                      ;  handler's segment?

          JNE     UNCHAIN_A           ; No, branch
; Set interrupt vector in low memory to point to the handler
;  pointed to by your pointer

          PUSH    DS
          MOV     DX,WORD PTR CS:FPTR
          MOV     DS,WORD PTR CS FPTR[2]
          MOV     AX,250FH            ; DOS set interrupt vector
          INT     21H
          POP     DS

          JMP     UNCHAIN_X

UNCHAIN_A:    ; BX = FPTR offset, ES = FPTR segment, CX = CS
          CMP     ES:[BX+6],4B42H     ; Is handler using the appropriate
                                      ;  conventions (Is SIGNATURE present in
                                      ;   the interrupt chaining structure)?
          JNE     exception           ; No, invoke error exception handler
          LDS     SI,ES:[BX+2]        ; Get FPTR's segment and offset
          CMP     SI,OFFSET ENTRY     ; Is this forward pointer pointing to
                                      ;  your handler's offset?

          JNE     UNCHAIN_B           ; No, branch
          MOV     CX,DS               ; Move to compare
          CMP     AX,CX               ; Is this forward pointer pointing to
                                      ;  your handler's segment?

          JNE     UNCHAIN_B           ; No, branch
; Located your handler in the chain
          MOV     AX,WORD PTR CS:FPTR ; Get your FPTR's offset
          MOV     ES:[BX+2],AX        ; Replace offset of FPTR of handler
                                      ;  that points to you
          MOV     AX,WORD PTR CS:FPTR[2] ; Get your FPTR's segment
          MOV     ES:[BX+4],AX        ; Replace segment of FPTR of handler
                                      ;  that points to you
          MOV     AL,CS:FLAGS         ; Get your flags
          AND     AL,FIRST            ; Isolate FIRST flag
          OR      ES:[BX + 6],AL      ; Set your first flag into prior routine

          JMP     UNCHAIN_X

UNCHAIN_B: MOV    BX,SI               ; Move new offset to BX
          PUSH    DS
          PUSH    ES

          JMP     UNCHAIN_A           ; Examine next handler in chain

UNCHAIN_X: STI                        ; Enable interrupts
          POP     ES
          POP     DS
```

System Timers

The system has three programmable timer/counters, Channels 0 through 2. They are controlled by an Intel 8254-2 Timer/Counter chip, and are defined as follows:

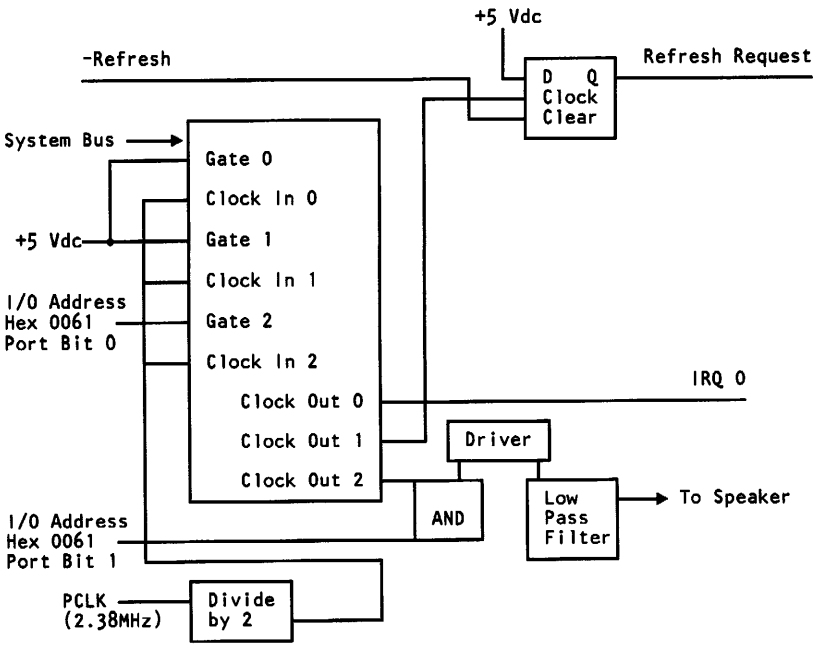
Channel 0	**System Timer**
GATE 0	Tied on
CLK IN 0	1.193182 MHz OSC
CLK OUT 0	8259A IRQ 0

Channel 1	**Refresh Request Generator**
GATE 1	Tied on
CLK IN 1	1.193182 MHz OSC
CLK OUT 1	Request refresh cycle

Note: Channel 1 is programmed as a rate generator to produce a 15-microsecond period signal.

Channel 2	**Tone Generation for Speaker**
GATE 2	Controlled by bit 0 of port hex 61, PPI bit
CLK IN 2	1.193182 MHz OSC
CLK OUT 2	Used to drive the speaker

The 8254-2 Timer/Counter is a programmable interval timer/counter that system programs treat as an arrangement of four external I/O ports. Three ports are treated as counters; the fourth is a control register for mode programming. The following is a system-timer block diagram.



System-Timer Block Diagram

System Clock

The 82284 System Clock Generator is driven by a 12-MHz crystal. Its output 'clock' signal (CLK) is the input to the system microprocessor and the I/O channel.

ROM Subsystem

The system board's ROM subsystem consists of two 32K by 8-bit ROM/EPROM modules in a 32K-by-16-bit arrangement. The code for odd and even addresses resides in separate modules. ROM is assigned at the top of the first and last 1M address space (0F0000 and FF0000). ROM is not parity-checked. Its maximum access time is 170 nanoseconds and its maximum cycle time is 333 nanoseconds.

RAM Subsystem

The system board's RAM subsystem starts at address 000000 of the 16M address space and consists of 640K of read/write (R/W) memory. The 640K memory is composed of two 256K-by-9 random access memory module packages (512K), plus two banks of 64K-by-4 RAM modules (128K). The 64K-by-4 RAM modules may be disabled at jumper J10, located on the system board. Memory access time is 150 nanoseconds and the cycle time is 275 nanoseconds.

Memory refresh requests one memory cycle every 15 microseconds through the timer/counter (channel 1). The RAM initialization program performs the following functions:

- Initializes channel 1 of the timer/counter to the rate generation mode, with a period of 15 microseconds
- Performs a memory write operation to any memory location.

  Note: The memory must be accessed or refreshed eight times before it can be used.

I/O Channel

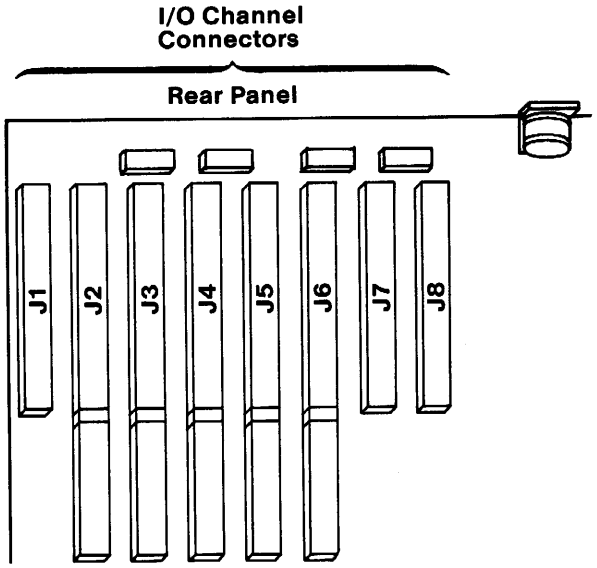
The I/O channel supports:

- I/O address space hex 100 to hex 3FF
- 24-bit memory addresses (16M)
- Selection of data accesses (either 8- or 16-bit)
- Interrupts
- DMA channels
- I/O wait-state generation
- Open-bus structure (allowing multiple microprocessors to share the system's resources, including memory)
- Refresh of system memory from channel microprocessors.

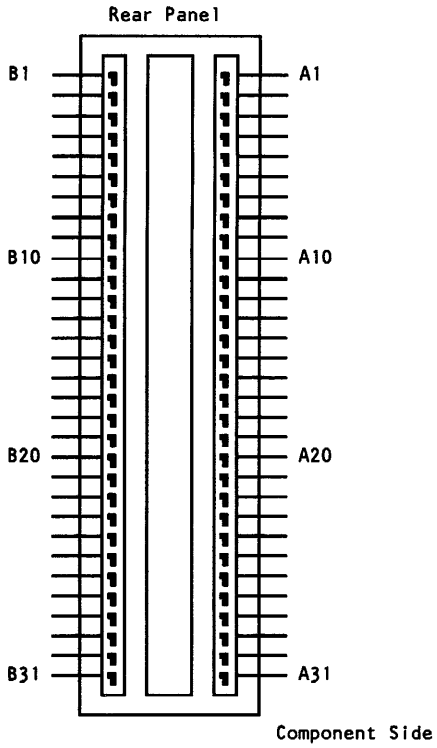
Connectors

The following figure shows the location and the numbering of the I/O channel connectors. These connectors consist of five 98-pin and three 62-pin edge connector sockets.

Note: The three 62-pin positions can support only 62-pin I/O bus adapters.

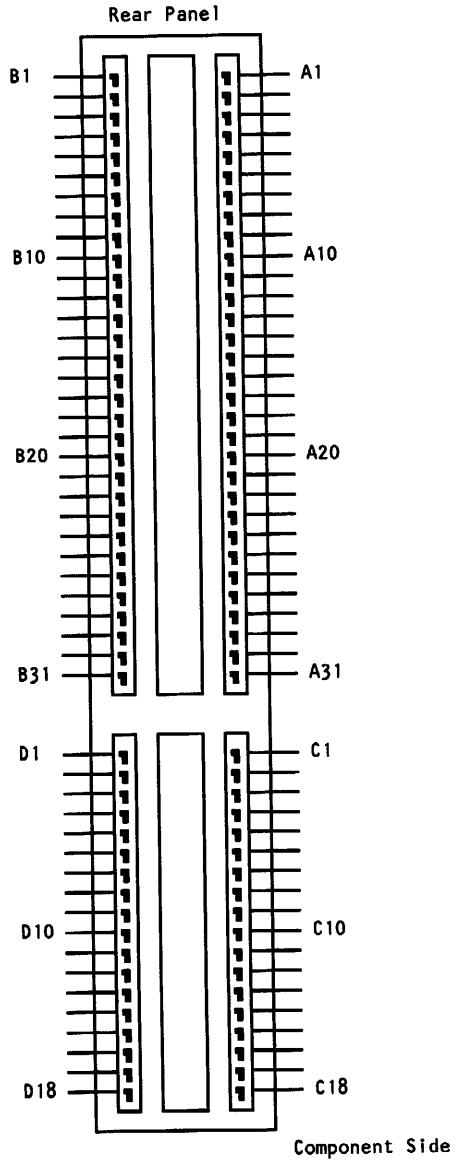


The following figure shows the pin numbering for the 62-pin I/O channel connectors J1, J7 and J8.



I/O Channel Pin Numbering (J1, J7 and J8)

The following figure shows the pin numbering for the 98-pin I/O channel connectors J2 through J6.



I/O Channel Pin Numbering (J2-J6)

The following figures summarize pin assignments for the I/O channel connectors.

I/O Pin	Signal Name	I/O
A1	-I/O CH CK	I
A2	SD7	I/O
A3	SD6	I/O
A4	SD5	I/O
A5	SD4	I/O
A6	SD3	I/O
A7	SD2	I/O
A8	SD1	I/O
A9	SD0	I/O
A10	+I/O CH RDY	I
A11	AEN	O
A12	SA19	I/O
A13	SA18	I/O
A14	SA17	I/O
A15	SA16	I/O
A16	SA15	I/O
A17	SA14	I/O
A18	SA13	I/O
A19	SA12	I/O
A20	SA11	I/O
A21	SA10	I/O
A22	SA9	I/O
A23	SA8	I/O
A24	SA7	I/O
A25	SA6	I/O
A26	SA5	I/O
A27	SA4	I/O
A28	SA3	I/O
A29	SA2	I/O
A30	SA1	I/O
A31	SA0	I/O

I/O Channel (A-Side)

I/O Pin	Signal Name	I/O
B1	GND	Ground
B2	RESET DRV	O
B3	+5 Vdc	Power
B4	IRQ 9	I
B5	-5 Vdc	Power
B6	DRQ2	I
B7	-12 Vdc	Power
B8	OWS	I
B9	+12 Vdc	Power
B10	GND	Ground
B11	-SMEMW	O
B12	-SMEMR	O
B13	-IOW	I/O
B14	-IOR	I/O
B15	-DACK3	O
B16	DRQ3	I
B17	-DACK1	O
B18	DRQ1	I
B19	-REFRESH	I/O
B20	CLK	O
B21	IRQ7	I
B22	IRQ6	I
B23	IRQ5	I
B24	IRQ4	I
B25	IRQ3	I
B26	-DACK2	O
B27	T/C	O
B28	BALE	O
B29	+5Vdc	Power
B30	14.318MHz OSC	O
B31	GND	Ground

I/O Channel (B-Side)

I/O Pin	Signal Name	I/O
C1	-SBHE	I/O
C2	LA23	I/O
C3	LA22	I/O
C4	LA21	I/O
C5	LA20	I/O
C6	LA19	I/O
C7	LA18	I/O
C8	LA17	I/O
C9	-MEMR	I/O
C10	-MEMW	I/O
C11	SD08	I/O
C12	SD09	I/O
C13	SD10	I/O
C14	SD11	I/O
C15	SD12	I/O
C16	SD13	I/O
C17	SD14	I/O
C18	SD15	I/O

I/O Channel (C-Side, J2 through J6 only)

I/O Pin	Signal Name	I/O
D1	-MEM CS16	I
D2	-I/O CS16	I
D3	IRQ10	I
D4	IRQ11	I
D5	IRQ12	I
D6	IRQ15	I
D7	IRQ14	I
D8	-DACK0	O
D9	DRQ0	I
D10	-DACK5	O
D11	DRQ5	I
D12	-DACK6	O
D13	DRQ6	I
D14	-DACK7	O
D15	DRQ7	I
D16	+5 Vdc	POWER
D17	-MASTER	I
D18	GND	GROUND

I/O Channel (D-Side, J2 through J6 only)

I/O Channel Signal Description

The following is a description of the system board's I/O channel signals. All signal lines are TTL compatible. I/O adapters should be designed with a maximum of two low-power Shottky (LS) loads per line and be capable of driving the data and address lines similar to a 74LS245 driver.

SA0 through SA19 (I/O)

Address signals 0 through 19 are used to address memory and I/O devices within the system. These 20 address lines, in addition to LA17 through LA23, allow access of up to 16M of memory. SA0 through SA19 are gated on the system bus when 'buffered address latch enable' signal (BALE) is high and are latched on the falling edge of BALE. These signals are generated by the microprocessor or DMA Controller. They also may be driven by other microprocessors or DMA controllers that reside on the I/O channel.

LA17 through LA23 (I/O)

These signals (unlatched) are used to address memory and I/O devices within the system. They give the system up to 16M of addressability. These signals are valid from the leading edge of BALE to the trailing edge of the '-I/O Read' (-IOR) or '-I/O Write' (-IOW) command cycle. These decodes should be latched by I/O adapters on the falling edge of the 'buffered address latch enable' signal (BALE).

These signals also may be driven by other microprocessors or DMA controllers that reside on the I/O channel.

CLK (O)

This is the system 'clock' signal. It is a synchronous microprocessor cycle clock with a cycle time of 167 nanoseconds. The clock has a 50% duty cycle. This signal should be used only for synchronization. It is not intended for uses requiring a fixed frequency.

RESET DRV (O)

The 'reset drive' signal is used to reset or initialize system logic at power-up time or during a low voltage condition. This signal is active high.

SD0 through SD15 (I/O)

These signals provide data bits 0 through 15 for the microprocessor, memory, and I/O devices. D0 is the least-significant bit and D15 is the most-significant bit. All 8-bit devices on the I/O channel should use D0 through D7 for communications to the microprocessor. The 16-bit devices will use D0 through D15. To support 8-bit devices, the data on D8 through D15 will be gated to D0 through D7 during 8-bit transfers to these devices; 16-bit microprocessor transfers to 8-bit devices will be converted to two 8-bit transfers.

BALE (O) (buffered)

The 'buffered address latch enable' signal is available to the I/O channel as an indicator of a valid microprocessor or DMA address (when used with 'address enable' signal, AEN). Microprocessor addresses SA0 through SA19 are latched with the falling edge of BALE. BALE is forced high (active) during DMA cycles. From the trailing edge of a command cycle (for example, the trailing edge of -IOR or -IOW) to the leading edge of BALE, the address lines are in transition and are not stable.

-I/O CH CK (I)

The '-I/O channel check' signal provides the system board with parity (error) information about memory or devices on the I/O channel. When this signal is active (low), it indicates a non-correctable system error.

I/O CH RDY (I)

The 'I/O channel ready' signal is pulled low (not ready) by a memory or I/O device to lengthen I/O or memory cycles. Any slow device using this line should drive it low immediately upon detecting its valid address and a Read or Write command. Machine cycles are extended by an integral number of clock cycles (167 nanoseconds). This signal should be held low for no more than 2.5 microseconds.

IRQ3-IRQ7, IRQ9-IRQ12, IRQ14, and IRQ15 (I)

Interrupt requests 3 through 7, 9 through 12, 14, and 15 are used to signal the microprocessor that an I/O device needs attention. The interrupt requests are prioritized, with IRQ9 through IRQ12, IRQ14, and IRQ15 having the highest priority (IRQ9 is the highest), and IRQ3 through IRQ7 having the lowest priority (IRQ7 is the lowest). An interrupt request is generated when an IRQ line is raised from low to high. The line is high until the microprocessor acknowledges the interrupt request (Interrupt Service routine). See the figure on page 1-13 for additional information.

Note: Interrupt requests IRQ0-IRQ2, IRQ8, IRQ13 are used on the system board and are not available on the I/O channel.

-IOR (I/O)

The '-I/O read' signal instructs an I/O device to drive its data onto the data bus. This signal may be driven by the system microprocessor or DMA controller, or by a microprocessor or DMA controller resident on the I/O channel. This signal is active low.

-IOW (I/O)

The '-I/O write' signal instructs an I/O device to read the data off the data bus. It may be driven by any microprocessor or DMA controller in the system. This signal is active low.

-SMEMR (O) -MEMR (I/O)

These signals instruct the memory devices to drive data onto the data bus. -SMEMR is active only when the memory decode is within the low 1M of memory space. -MEMR is active on all memory read cycles. -MEMR may be driven by any microprocessor or DMA controller in the system. -SMEMR is derived from -MEMR and the decode of the low 1M of memory. When a microprocessor on the I/O channel wishes to drive -MEMR, it must have the address lines valid on the bus for one clock cycle before driving -MEMR active. Both signals are active low.

-SMEMW (O) -MEMW (I/O)

These signals instruct the memory devices to store the data present on the data bus. -SMEMW is active only when the memory decode is within the low 1M of the memory space. -MEMW is active on all memory write cycles. -MEMW may be driven by any microprocessor or DMA controller in the system. -SMEMW is derived from -MEMW and the decode of the low 1M of memory. When a microprocessor on the I/O channel wishes to drive -MEMW, it must have the address lines valid on the bus for one clock cycle before driving -MEMW active. Both signals are active low.

DRQ0-DRQ3 and DRQ5-DRQ7 (I)

The 'DMA request' signals 0 through 3 and 5 through 7 are asynchronous channel requests used by peripheral devices and a microprocessor to gain DMA service (or control of the system). They are prioritized, with DRQ0 having the highest priority and DRQ7 the lowest. A request is generated by bringing a DRQ line to an active (high) level. A DRQ line is held high until the corresponding 'DMA acknowledge' (DACK) line goes active. DRQ0 through DRQ3 perform 8-bit DMA transfers; DRQ5 through DRQ7 perform 16-bit transfers. DRQ4 is used on the system board and is not available on the I/O channel.

-DACK0 to -DACK3 and -DACK5 to -DACK7 (O)

-DMA acknowledge 0 through 3 and 5 through 7 are used to acknowledge DMA requests. These signals are active low.

AEN (O)

The 'address enable' signal is used to degate the microprocessor and other devices from the I/O channel to allow DMA transfers to take place. When this line is active, the DMA controller has control of the address bus, the data bus, Read command lines (memory and I/O), and the Write command lines (memory and I/O). This signal is active high.

-REFRESH (I/O)

This signal is used to indicate a refresh cycle and can be driven by a microprocessor on the I/O channel. This signal is active low.

T/C (O)

The 'terminal count' signal provides a high pulse when the terminal count for any DMA channel is reached.

-SBHE (I/O)

The 'system bus high enable' signal indicates a transfer of data on the upper byte of the data bus, SD8 through SD15. Sixteen-bit devices use -SBHE to condition data bus buffers tied to SD8 through SD15. This signal is active low.

-MASTER (I)

This signal is used with a DRQ line to gain control of the system. A processor or DMA controller on the I/O channel may issue a DRQ to a DMA channel in cascade mode and receive a -DACK. Upon receiving the -DACK, a microprocessor may pull -MASTER active (low), which will allow it to control the system address, data, and control lines (a condition known as *tri-state*). After -MASTER is low, the microprocessor must wait one clock cycle before driving the address and data lines, and two clock cycles before issuing a Read or Write command. If this signal is held low for more than 15 microseconds, the system memory may be lost because of a lack of refresh.

-MEM CS16 (I)

The '-memory 16-bit chip select' signal indicates to the system that the present data transfer is a 1 wait-state, 16-bit, memory cycle. It must be derived from the decode of LA17 through LA23. -MEM CS16 is active low and should be driven with an open collector or tri-state driver capable of sinking 20 mA.

-I/O CS16 (I)

The '-I/O 16-bit chip select' signal indicates to the system that the present data transfer is a 16-bit, 1 wait-state, I/O cycle. It is derived from an address decode. -I/O CS16 is active low and should be driven with an open collector or tri-state driver capable of sinking 20 mA.

14.318MHz OSC (0)

The '14.318MHz oscillator' signal is a high-speed clock with a 70-nanosecond period (14.31818 MHz). This signal is not synchronous with the system clock. It has a 50% duty cycle.

0WS (I)

The 'zero wait state' signal tells the microprocessor that it can complete the present bus cycle without inserting any additional wait cycles. In order to run a memory cycle to a 16-bit device without wait cycles, 0WS is derived from an address decode gated with a Read or Write command. 0WS is active low and should be driven with an open collector or tri-state driver capable of sinking 20 mA.

I/O Addresses

The following describes the system board's I/O addresses.

Hex Range	Device
000-01F	DMA Controller 1, 8237A-5
020-03F	Interrupt Controller 1, 8259A, Master
040-05F	Timer, 8254-2
060	8042 (Keyboard)
061	System Board I/O port
064	8042 (Keyboard)
070-07F	Real-Time Clock, NMI (Non-maskable Interrupt) Mask
080-09F	DMA Page Register , 74LS612
0A0-0BF	Interrupt Controller 2, 8259A
0C0-0DF	DMA Controller 2, 8237A-5
0F0	Clear Math Coprocessor Busy
0F1	Reset Math Coprocessor
0F8-0FF	Math Coprocessor
1F0-1F8	Fixed Disk
20C-20D	Reserved
21F	Voice Communications Adapter
278-27F	Parallel Printer Port 2
2B0-2DF	Alternate Enhanced Graphics Adapter
2E1	GPIB (Adapter 0)
2E2 & 2E3	Data Acquisition (Adapter 0)
2F8-2FF	Serial Port 2
300-31F	Prototype Adapter
360-363	PC Network (low address)
364-367	Reserved
368-36B	PC Network (high address)
36C-36F	Reserved
378-37F	Parallel Printer Port 1
380-38F	SDLC, Bisynchronous 2
3A0-3AF	Bisynchronous 1
3B0-3BF	Monochrome Display and Printer Adapter
3C0-3CF	Enhanced Graphics Adapter
3D0-3DF	Color/Graphics Monitor Adapter
3F0-3F7	Diskette Controller
3F8-3FF	Serial Port 1
6E2 & 6E3	Data Acquisition (Adapter 1)
AE2 & AE3	Data Acquisition (Adapter 2)
EE2 & EE3	Data Acquisition (Adapter 3)
22E1	GPIB (Adapter 1)
42E1	GPIB (Adapter 2)
62E1	GPIB (Adapter 3)
82E1	GPIB (Adapter 4)
A2E1	GPIB (Adapter 5)
C2E1	GPIB (Adapter 6)
E2E1	GPIB (Adapter 7)

Note: I/O Addresses, hex 000 to 0FF, are reserved for the system board I/O. Hex 100 to 3FF are available on the I/O channel. The system board decodes up to 10 bits of I/O address information. I/O addresses above 3FF must not conflict with the system board I/O addresses.

I/O Address Map

NMI Controls

During POST, the non-maskable interrupt (NMI) into the 80286 is masked off. The mask bit can be set and reset with system programs as follows:

Mask On (Disable NMI)	Write to I/O address hex 070, with data bit 7 equal to a logical 1.
Mask Off (Enable NMI)	Write to I/O address hex 070, with data bit 7 equal to a logical 0.

Note: At the end of POST, the system enables NMI.

The '-I/O channel check' signal (-I/O CH CK) is used to report noncorrectable errors on RAM adapters on the I/O channel. This check creates an NMI if the NMI is enabled. During POST, the NMI is masked off and -I/O CH CK is disabled. Follow these steps when enabling -I/O CH CK and the NMI.

1. Write data in all I/O RAM-adapter memory locations; this establishes good parity at all locations.
2. Enable -I/O CH CK.
3. Enable the NMI.

Note: All three of these functions are performed by POST.

When a check occurs, an interrupt (NMI) results. Read the status bits to determine the source of the NMI (see the figure, "I/O Address Map" on page 1-38). To determine the location of the failing adapter, write to any memory location within a given adapter. If the parity check was from that adapter, -I/O CH CK is reset to inactive.

I/O Port (Read/Write)

Address hex 061 is a read/write port on the system board.

Input (Read)

The following are the input (read) bit descriptions for this I/O port.

Bit 7 +RAM Parity Check—System board memory parity check.

0 No memory parity check error has occurred.

1 A memory parity check error has occurred.

A non-maskable interrupt (NMI) will occur if NMI is enabled. The error bit can be reset by toggling output port hex 061, bit 2, to a 1 and then back to a 0.

Bit 6 +I/O Channel Check—Error on an I/O channel adapter (memory parity error or adapter errors).

0 No I/O channel error has occurred.

1 An I/O channel error has occurred.

A non-maskable interrupt (NMI) will occur if NMI is enabled. The error bit can be reset by toggling output port hex 061, bit 3, to a 1 and then back to a 0.

Bit 5 Timer 2 Channel Out—Reflects the level of the Timer 2 output.

Bit 4 Refresh Detect—Toggles every 15 microseconds, indicating normal Refresh activity

Bits 3 to 0 Read the status of bits 3, 2, 1, and 0, respectively, written to output port hex 061.

Output (Write)

The following are the output (write) bit descriptions for the system board I/O port.

Bits 7 to 4 Not used

Bit 3 -Enable I/O Channel Check

0 Enables I/O Channel Check errors.

1 Disables I/O Channel Check errors.

During power-up this bit is toggled to a 1, then back to a 0, to clear the I/O channel check flip-flop of previous errors.

Bit 2 -Enable System Board RAM Parity Check

0 Enables system board RAM parity check.

1 Disables system board RAM parity check.

During power-up this bit is toggled to a 1, then back to a 0, to clear the RAM parity check flip-flop before BIOS checks the system memory for parity errors.

Bit 1 +Speaker Data—Controls the speaker output (along with Timer 2 Clock output).

Bit 0 +Timer 2 Gate Speaker

0 Disables 8254 Timer 2 clock input (1.19 MHz).

1 Enables 8254 Timer 2 clock input (1.19 MHz).

Diagnostic-Checkpoint Port

I/O address hex 080 is used as a diagnostic-checkpoint port or register. This port corresponds to a read/write register in the DMA page register (74LS612). This port is used by POST during power up.

Coprocessor Controls

The following is a description of the Math Coprocessor controls.

0F0 An 8-bit Out command to port F0 will clear the latched Math Coprocessor '-busy' signal. The '-busy' signal will be latched if the coprocessor asserts its '-error' signal while it is busy. The data output should be zero.

0F1 An 8-bit Out command to port F1 will reset the Math Coprocessor. The data output should be zero.

Other Circuits

Speaker

The system unit has a 2-1/4 inch permanent-magnet speaker, which can be driven from:

- The I/O-port output bit
- The timer/counter's CLK OUT 2
- Both of the above

128K RAM Jumper (J10)

The system board has a three-pin, Berg-strip connector (J10). From the rear of the system to the front, the pins are numbered 1 through 3. Jumper placement across these pins determines whether the last 128K RAM (512KB to 640KB) of system board memory is enabled or disabled.

Pin	Assignments
1	No Connection
2	Ground
3	RAM Select

RAM Jumper Connector (J10)

With the jumper on pins 1 and 2, the 128K RAM is enabled. When the jumper is on pins 2 and 3, the 128K RAM is disabled.

Note: The normal mode is the enabled mode.

Display Switch

Set the slide switch on the system board to select the primary display adapter. Its positions are assigned as follows:

On (toward the front of the system unit): The primary display is attached to the Color/Graphics Monitor Adapter.

Off (toward the rear of the system unit): The primary display is attached to the Monochrome Display and Printer Adapter.

The switch may be set to either position if the primary display is attached to an Enhanced Graphics Adapter.

> **Note:** The primary display is activated when the system is powered on.

Keyboard Controller

The keyboard controller is a single-chip microcomputer (Intel 8042, or EPROM version 8742) that is programmed to support the keyboard serial interface. The keyboard controller receives serial data from the keyboard, checks the parity of the data, translates scan codes, and presents the data to the system as a byte of data in its output buffer. The controller can interrupt the system when data is placed in its output buffer, or wait for the system to poll its status register to determine when data is available.

Data is sent to the keyboard by first polling the controller's status register to determine when the input buffer is ready to accept data and then writing to the input buffer. Each byte of data is sent to the keyboard serially with an odd parity bit automatically inserted. Since the keyboard is required to acknowledge all data transmissions, another byte of data should not be sent to the keyboard until acknowledgement is received for the previous byte sent. The output-buffer-full interrupt may be used for both send and receive routines.

Keyboard Controller Initialization

At power-on, the keyboard controller sets the system flag bit to 0. After a power-on reset or the execution of the Self Test command, the keyboard controller disables the keyboard interface by forcing the 'keyboard clock' line low. The keyboard interface parameters are specified at this time by writing to locations within the 8042 RAM. The keyboard-inhibit function is then disabled by setting the inhibit-override bit in the command byte. A hex 55 is then placed in the output buffer if no errors are detected during the self test. Any value other than hex 55 indicates that the 8042 is defective. The keyboard interface is now enabled by lifting the 'keyboard data' and 'keyboard clock' signal lines, and the system flag is set to 1. The keyboard controller is then ready to accept commands from the system unit microprocessor or receive keyboard data.

The initialization sequence causes the keyboard to establish Mode 2 protocol (see "Data Stream" on page 4-27).

Receiving Data from the Keyboard

The keyboard sends data in a serial format using an 11-bit frame. The first bit is a start bit, and is followed by eight data bits, an odd parity bit, and a stop bit. Data sent is synchronized by a clock supplied by the keyboard. At the end of a transmission, the keyboard controller disables the interface until the system accepts the byte. If the byte of data is received with a parity error, a Resend command is automatically sent to the keyboard. If the keyboard controller is unable to receive the data correctly after a set number of retries, a hex FF is placed in its output buffer, and the parity bit in the status register is set to 1, indicating a receive parity error. The keyboard controller will also time a byte of data from the keyboard. If a keyboard transmission does not end within 2 milliseconds, a hex FF is placed in the keyboard controller's output buffer, and the receive time-out bit in the status register is set. No retries will be attempted on a receive time-out error.

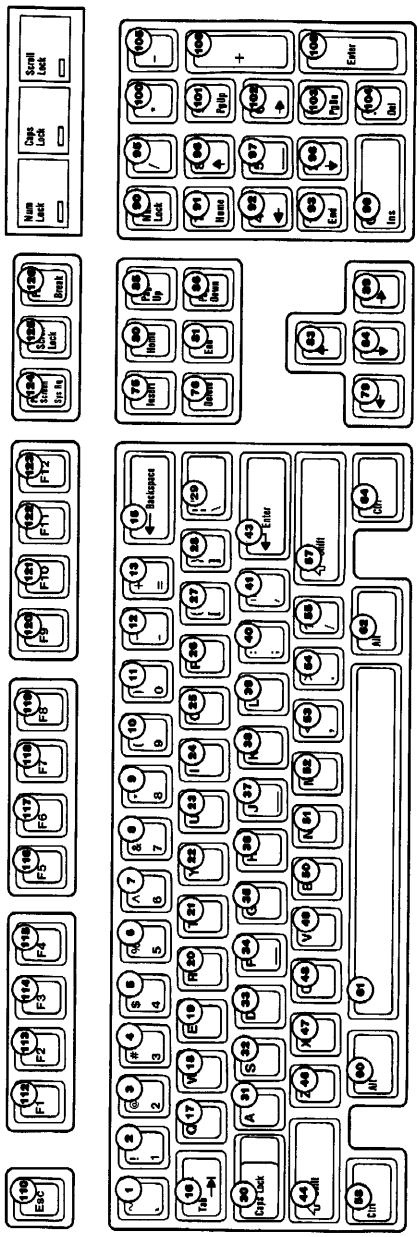
> **Note:** When a receive error occurs in the default mode (bits 5, 6, and 7 of the command byte set to 0), hex 00 is placed in the output buffer instead of hex FF. See

"Commands (I/O Address Hex 64)" on page 1-56 for a detailed description of the command byte.

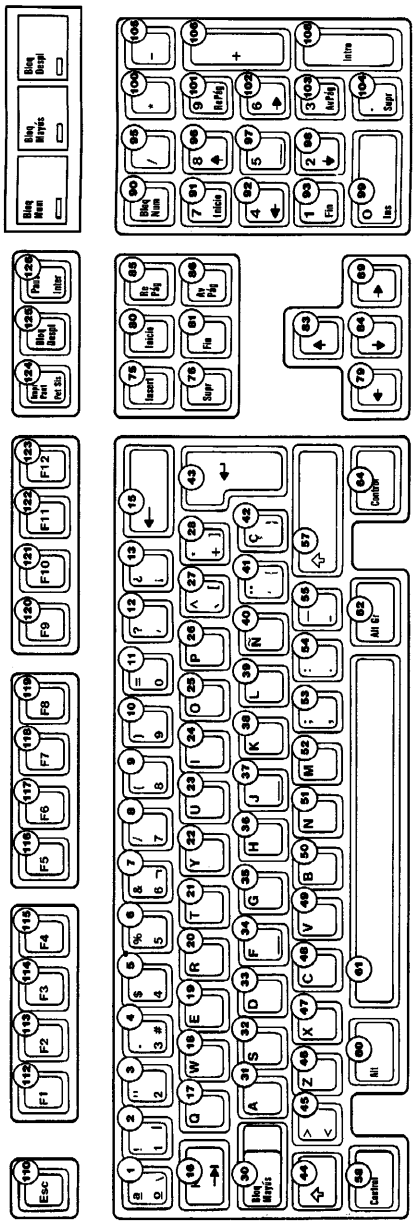
Scan Code Translation

Scan codes received from the keyboard are converted by the keyboard controller before being placed into the controller's output buffer. The following figures show the keyboard layouts. Each key position is numbered for reference.

101-Key Keyboard



102-Key Keyboard



The following figure is the scan-code translation table.

System Scan Code	Keyboard Scan Code	Key (101/102-key)
01	76	110
02	16	2
03	1E	3
04	26	4
05	25	5
06	2E	6
07	36	7
08	3D	8
09	3E	9
0A	46	10
0B	45	11
0C	4E	12
0D	55	13
0E	66	15
0F	0D	16
10	15	17
11	1D	18
12	24	19
13	2D	20
14	2C	21
15	35	22
16	3C	23
17	43	24
18	44	25
19	4D	26
1A	54	27
1B	5B	28
1C	5A	43
1D	14	58
1E	1C	31
1F	1B	32
20	23	33
21	2B	34
22	34	35
23	33	36
24	3B	37
25	42	38
26	4B	39
27	4C	40
28	52	41
29	0E	1
2A	12	44
2B	5D	29 (U.S. only) 42 (except U.S.)
2C	1A	46
2D	22	47
2E	21	48
2F	2A	49

Scan-Code Translation Table (Part 1 of 3)

System Scan Code	Keyboard Scan Code	Key (101/102-key)
30	32	50
31	31	51
32	3A	52
33	41	53
34	49	54
35	4A	55
36	59	57
38	11	60
39	29	61
3A	58	30
3B	05	112
3C	06	113
3D	04	114
3E	0C	115
3F	03	116
40	0B	117
41	83	118
42	0A	119
43	01	120
44	09	121
45	77	90
46	7E	125
47	6C	91
48	75	96
49	7D	101
4A	7B	105
4B	6B	92
4C	73	97
4D	74	102
4E	79	106
4F	69	93
50	72	98
51	7A	103
52	70	99
53	71	104
54	7F or 84	-
56	61	45 (except U.S.)
57	78	122
58	07	123
FF	00	-
E0 2A E0 37	E0 12 E0 7C	124
E0 1C	E0 5A	108
E0 1D	E0 14	64
E0 35	E0 4A	95
E0 37	7C	100
E0 38	E0 11	62
E0 47	E0 6C	80

Scan-Code Translation Table (Part 2 of 3)

System Scan Code	Keyboard Scan Code	Key (101/102-key)
E0 48	F0 47 75	83
E0 49	F0 47 7D	85
E0 4B	F0 47 6B	79
E0 4D	F0 47 74	89
E0 4F	F0 47 69	81
E0 50	F0 47 72	84
E0 51	F0 47 7A	86
E0 52	F0 47 70	75
E0 53	F0 47 71	76
E1 1D 45 E1 9D C5	E1 14 77 E1 F0 14 F0 77	126

Scan-Code Translation Table (Part 3 of 3)

The following scan codes are reserved.

Key	Keyboard Scan Code	System Scan Code
Reserved	60	55
Reserved	61	56
Reserved	78	57
Reserved	07	58
Reserved	0F	59
Reserved	17	5A
Reserved	1F	5B
Reserved	27	5C
Reserved	2F	5D
Reserved	37	5E
Reserved	3F	5F
Reserved	47	60
Reserved	4F	61
Reserved	56	62
Reserved	5E	63
Reserved	08	64
Reserved	10	65
Reserved	18	66
Reserved	20	67
Reserved	28	68
Reserved	30	69
Reserved	38	6A
Reserved	40	6B
Reserved	48	6C
Reserved	50	6D
Reserved	57	6E
Reserved	6F	6F
Reserved	13	70
Reserved	19	71
Reserved	39	72
Reserved	51	73
Reserved	53	74
Reserved	5C	75
Reserved	5F	76
Reserved	62	77
Reserved	63	78
Reserved	64	79
Reserved	65	7A
Reserved	67	7B
Reserved	68	7C
Reserved	6A	7D
Reserved	6D	7E
Reserved	6E	7F

Reserved Scan-Code Translation Table

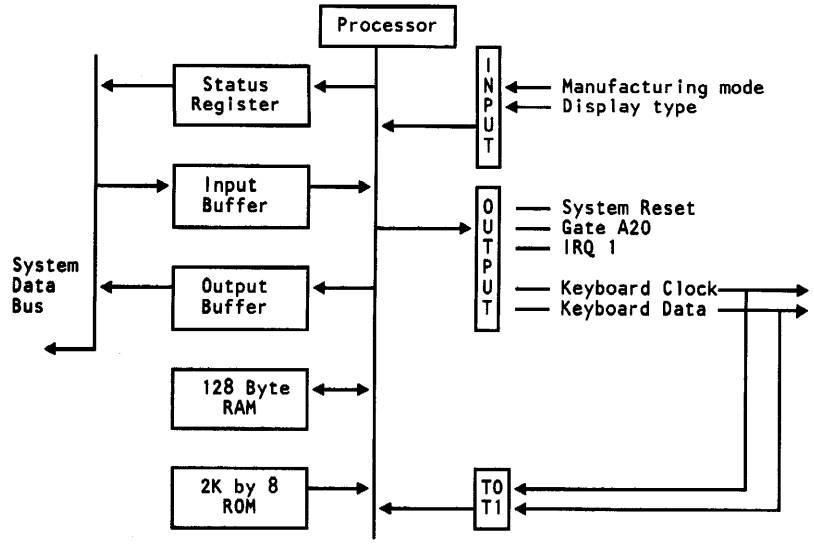
Sending Data to the Keyboard

The keyboard sends data in the same serial format used to receive data from the keyboard. A parity bit is automatically inserted by the keyboard controller. If the keyboard does not start clocking the data from the keyboard controller within 15 milliseconds, or complete that clocking within 2 milliseconds, a hex FE is placed in the keyboard controller's output buffer, and the transmit time-out error bit is set in the status register.

The keyboard is required to respond to all transmissions. The keyboard responds to any valid command and parameter, other than Echo and Resend, with an Acknowledge (ACK) response, hex FA. If the response contains a parity error, the keyboard controller places a hex FE in its output buffer, and the transmit time-out and parity error bits are set in the status register. The keyboard controller is programmed to set a 25-millisecond time limit for the keyboard to respond. If this time limit is exceeded, the keyboard controller places a hex FE in its output buffer and sets the transmit time-out and receive time-out error bits in the status register. No retries are attempted by the keyboard controller for any transmission error.

Keyboard Controller System Interface

The keyboard controller communicates with the system through a status register, an output buffer, and an input buffer. The following figure is a block diagram of the keyboard interface.



Keyboard Controller Interface Block Diagram

Status Register

The status register is an 8-bit read-only register at I/O address hex 64. It has information about the state of the keyboard controller (8042) and interface. It may be read at any time.

Status-Register Bit Definition

Bit 7 Parity Error—A 0 indicates the last byte of data received from the keyboard had odd parity. A 1 indicates the last byte had even parity. The keyboard should send data with odd parity.

Bit 6 Receive Time-Out—A 1 indicates that a transmission was started by the keyboard but did not finish within the programmed receive time-out delay.

Bit 5 Transmit Time-Out—A 1 indicates that a transmission started by the keyboard controller was not properly completed. If the transmit byte was not clocked out within the specified time limit, this will be the only error bit on.

If the transmit byte was clocked out but a response was not received within the programmed time limit, the transmit time-out and receive time-out error bits are set to 1. If the transmit byte was clocked out but the response was received with a parity error, the transmit time-out and parity error bits are set to 1.

Bit 4 Always set to 1.

Bit 3 Command/Data—The keyboard controller's input buffer may be addressed as either I/O address hex 60 or 64. Address hex 60 is defined as the data port, and address hex 64 is defined as the command port. Writing to address hex 64 sets this bit to 1; writing to address hex 60 sets this bit to 0. The controller uses this bit to determine if the byte in its input buffer should be interpreted as a command byte or a data byte.

Bit 2 System Flag—This bit is monitored by the system during the reset routine. If it is a 0, the reset was caused by a power on. The controller sets this bit to 0 at power on and it is set to 1 after a successful self test. This bit can be changed by writing to the system flag bit in the command byte (hex 64).

Bit 1 Input Buffer Full—A 0 indicates that the keyboard controller's input buffer (I/O address hex 60 or 64) is empty. A 1 indicates that data has been written into the buffer but the controller has not read the data. When the controller reads the input buffer, this bit will return to 0.

Bit 0 Output Buffer Full—A 0 indicates that the keyboard controller's output buffer has no data. A 1 indicates that the controller has placed data into its output buffer but the system has not yet read the data. When the system reads the output buffer (I/O address hex 60), this bit will return to a 0.

Output Buffer

The output buffer is an 8-bit read-only register at I/O address hex 60. The keyboard controller uses the output buffer to send scan codes received from the keyboard, and data bytes requested by command, to the system. The output buffer should be read only when the output-buffer-full bit in the status register is 1.

Input Buffer

The input buffer is an 8-bit write-only register at I/O address hex 60 or 64. Writing to address hex 60 sets a flag, which indicates a data write; writing to address hex 64 sets a flag, indicating a command write. Data written to I/O address hex 60 is sent to the keyboard, unless the keyboard controller is expecting a data byte following a controller command. Data should be written to the controller's input buffer only if the input buffer's full bit in the status register is 0. The following are valid keyboard controller commands.

Commands (I/O Address Hex 64)

20 Read Keyboard Controller's Command Byte—The controller sends its current command byte to its output buffer.

60 Write Keyboard Controller's Command Byte—The next byte of data written to I/O address hex 60 is placed in the controller's command byte. Bit definitions of the command byte are as follows:

Bit 7 Reserved—Should be written as a 0.

Bit 6 IBM Personal Computer Compatibility Mode—Writing a 1 to this bit causes the controller to convert the scan codes received from the keyboard to those used by the IBM Personal Computer. This includes converting a 2-byte break sequence to the 1-byte IBM Personal Computer format.

Bit 5 IBM Personal Computer Mode—Writing a 1 to this bit programs the keyboard to support the IBM Personal Computer keyboard interface. In this mode the controller does not check parity or convert scan codes.

Bit 4 Disable Keyboard—Writing a 1 to this bit disables the keyboard interface by driving the 'clock' line low. Data is not sent or received.

Bit 3 Not used.

Bit 2 System Flag—The value written to this bit is placed in the system flag bit of the controller's status register.

Bit 1 Reserved—Should be written as a 0.

Bit 0 Enable Output-Buffer-Full Interrupt—Writing a 1 to this bit causes the controller to generate an interrupt when it places data into its output buffer.

AA Self-Test—This commands the controller to perform internal diagnostic tests. A hex 55 is placed in the output buffer if no errors are detected.

AB Interface Test—This commands the controller to test the 'keyboard clock' and 'keyboard data' lines. The test result is placed in the output buffer as follows:

00 No error detected.
01 The 'keyboard clock' line is stuck low.
02 The 'keyboard clock' line is stuck high.
03 The 'keyboard data' line is stuck low.
04 The 'keyboard data' line is stuck high.

AD Disable Keyboard Feature—This command sets bit 4 of the controller's command byte. This disables the keyboard interface by driving the clock line low. Data will not be sent or received.

AE Enable Keyboard Interface—This command clears bit 4 of the command byte, which releases the keyboard interface.

C0 Read Input Port—This commands the controller to read its input port and place the data in its output buffer. This command should be used only if the output buffer is empty.

D0 Read Output Port—This command causes the controller to read its output port and place the data in its output buffer. This command should be issued only if the output buffer is empty.

D1 Write Output Port—The next byte of data written to I/O address hex 60 is placed in the controller's output port.

> **Note:** Bit 0 of the controller's output port is connected to System Reset. This bit should not be written low as it will reset the microprocessor.

E0 Read Test Inputs—This command causes the controller to read its T0 and T1 inputs. This data is placed in the output buffer. Data bit 0 represents T0, and data bit 1 represents T1.

F0–FF Pulse Output Port—Bits 0 through 3 of the controller's output port may be pulsed low for approximately 6 microseconds. Bits 0 through 3 of this command indicate which bits are to be pulsed. A 0 indicates that the bit should be pulsed, and a 1 indicates the bit should not be modified.

> **Note:** Bit 0 of the controller's output port is connected to System Reset. Pulsing this bit resets the microprocessor.

I/O Ports

The keyboard controller has two I/O ports, one assigned for input and the other for output. Two test inputs are used by the controller to read the state of the keyboard's 'clock' (T0) and 'data' (T1) lines.

The following figures show bit definitions for the input and output ports, and the test-inputs.

Bit 7	Always set to 1
Bit 6	Display switch - Primary display attached to: 0 = Color/Graphics adapter 1 = Monochrome adapter
Bit 5	Manufacturing Jumper 0 = Manufacturing jumper installed 1 = Jumper not installed
Bit 4	Always set to 1
Bit 3	Reserved
Bit 2	Reserved
Bit 1	Reserved
Bit 0	Reserved

Input-Port Bit Definitions

Bit 7	Keyboard data (output)
Bit 6	Keyboard clock (output)
Bit 5	Input buffer empty
Bit 4	Output buffer full
Bit 3	Reserved
Bit 2	Reserved
Bit 1	Gate A20
Bit 0	System reset

Output-Port Bit Definitions

Note: In the real address mode Gate A20 prevents address line A20 from being set, maintaining compatibility with the 8088 microprocessor. When in the protected (virtual address) mode, Gate A20 allows addressing above the 1M range.

T1	Keyboard data (input)
T0	Keyboard clock (input)

Test-Input Bit Definitions

Real-Time Clock CMOS RAM Information

The RTC (Real-time Clock) CMOS RAM chip (Motorola MC146818A) contains the real-time clock and 64 bytes of CMOS RAM. The internal clock circuitry uses 14 bytes of this RAM, and the rest is allocated to configuration information. The following figure shows the CMOS RAM addresses.

Addresses	Description
00 - 0D	* Real-time clock information
0E	* Diagnostic status byte
0F	* Shutdown status byte
10	Diskette drive type byte - drives A and B
11	Reserved
12	Fixed disk types byte - drives C and D
13	Reserved
14	Equipment byte
15	Low base memory byte
16	High base memory byte
17	Low expansion memory byte
18	High expansion memory byte
19	Disk C extended byte
1A	Disk D extended byte
1B - 2D	Reserved
2E - 2F	2-byte CMOS checksum
30	* Low expansion memory byte
31	* High expansion memory byte
32	* Date century byte
33	* Information flags (set during power on)
34 - 3F	Reserved

CMOS RAM Internal Address Map

* These bytes are not included in the checksum calculation and are not part of the configuration record.

Real-Time Clock Information

The following figure describes real-time clock bytes and specifies their addresses.

Byte	Function	Address
0	Seconds	00
1	Second Alarm	01
2	Minutes	02
3	Minute Alarm	03
4	Hours	04
5	Hour Alarm	05
6	Day of Week	06
7	Date of Month	07
8	Month	08
9	Year	09
10	Status Register A	0A
11	Status Register B	0B
12	Status Register C	0C
13	Status Register D	0D

Real-Time Clock Internal Addresses 00 - 0D

Note: The setup program initializes registers A, B, C, and D when the time and date are set. Also Interrupt 1A is the BIOS interface to read/set the time and date. It initializes the status bytes the same as the Setup program.

Status Register A

Bit 7 Update in Progress (UIP)—A 1 indicates the time update cycle is in progress. A 0 indicates the current date and time are available to read.

Bit 6–Bit 4 22-Stage Divider (DV2 through DV0)—These three divider-selection bits identify which time-base frequency is being used. The system initializes the stage divider to 010, which selects a 32.768-kHz time base.

Bit 3–Bit 0 Rate Selection Bits (RS3 through RS0)—These bits allow the selection of a divider output frequency. The system initializes the rate selection bits to 0110, which selects a 1.024-kHz square wave output frequency and a 976.562-microsecond periodic interrupt rate.

Status Register B

Bit 7 Set—A 0 updates the cycle normally by advancing the counts at one-per-second. A 1 aborts any update cycle in progress and the program can initialize the 14 time-bytes without any further updates occurring until a 0 is written to this bit.

Bit 6 Periodic Interrupt Enable (PIE)—This bit is a read/write bit that allows an interrupt to occur at a rate specified by the rate and divider bits in register A. A 1 enables an interrupt, and a 0 disables it. The system initializes this bit to 0.

Bit 5 Alarm Interrupt Enable (AIE)—A 1 enables the alarm interrupt, and a 0 disables it. The system initializes this bit to 0.

Bit 4 Update-Ended Interrupt Enabled (UIE)—A 1 enables the update-ended interrupt, and a 0 disables it. The system initializes this bit to 0.

Bit 3 Square Wave Enabled (SQWE)—A 1 enables the the square-wave frequency as set by the rate selection bits in register A, and a 0 disables the square wave. The system initializes this bit to 0.

Bit 2 Date Mode (DM)—This bit indicates whether the time and date calendar updates are to use binary or binary coded decimal (BCD) formats. A 1 indicates binary, and a 0 indicates BCD. The system initializes this bit to 0.

Bit 1 24/12—This bit indicates whether the hours byte is in the 24-hour or 12-hour mode. A 1 indicates the 24-hour mode and a 0 indicates the 12-hour mode. The system initializes this bit to 1.

Bit 0 Daylight Savings Enabled (DSE)—A 1 enables daylight savings and a 0 disables daylight savings (standard time). The system initializes this bit to 0.

Status Register C

Bit 7–Bit 4 IRQF, PF, AF, UF—These flag bits are read-only and are affected when the AIE, PIE, and UIE bits in register B are set to 1.

Bit 3–Bit 0 Reserved—Should be written as a 0.

Status Register D

Bit 7 Valid RAM Bit (VRB)—This bit is read-only and indicates the status of the power-sense pin (battery level). A 1 indicates battery power to the real-time clock is good. A 0 indicates the battery is dead, so RAM is not valid.

Bits 6–Bit 0 Reserved—Should be written as a 0.

CMOS RAM Configuration Information

The following lists show bit definitions for the CMOS configuration bytes (addresses hex 0E – 3F).

Diagnostic Status Byte (Hex 0E)

Bit 7 Power status of the real-time clock chip—A 0 indicates that the chip has not lost power (battery good), and a 1 indicates that the chip lost power (battery bad).

Bit 6 Configuration Record (Checksum Status Indicator)—A 0 indicates that checksum is good, and a 1 indicates it is bad.

Bit 5 Incorrect Configuration Information—This is a check, at power-on time, of the equipment byte of the configuration record. A 0 indicates that the configuration information is valid, and a 1 indicates it is invalid. Power-on checks require:

- At least one diskette drive to be installed (bit 0 of the equipment byte set to 1).
- The primary display adapter setting in configuration matches the system board's display switch setting and the actual display adapter hardware in the system.

Bit 4 Memory Size Comparison—A 0 indicates that the power-on check determined the same memory size as in the configuration record, and a 1 indicates the memory size is different.

Bit 3 Fixed Disk Adapter/Drive C Initialization Status—A 0 indicates that the adapter and drive are functioning properly and the system can attempt "boot up." A 1 indicates that the adapter

and/or drive C failed initialization, which prevents the system from attempting to "boot up."

Bit 2 Time Status Indicator (POST validity check)— A 0 indicates that the time is valid, and a 1 indicates that it is invalid.

Bit 1–Bit 0 Reserved

Shutdown Status Byte (Hex 0F)

The bits in this byte are defined by the power on diagnostics. For more information about this byte, refer to "System BIOS".

Diskette Drive Type Byte (Hex 10)

Bit 7–Bit 4 Type of first diskette drive installed:

0000 No drive is present.
0001 Double Sided Diskette Drive (48 TPI).
0010 High Capacity Diskette Drive (96 TPI).
0011 720KB Diskette Drive (3.5 inch).

Note: 0100 through 1111 are reserved.

Bit 3–Bit 0 Type of second diskette drive installed:

0000 No drive is present.
0001 Double Sided Diskette Drive (48 TPI).
0010 High Capacity Diskette Drive (96 TPI).
0011 720KB Diskette Drive (3.5 inch).

Note: 0100 through 1111 are reserved.

Hex address 11 contains a reserved byte.

Fixed Disk Type Byte (Hex 12)

Bit 7–Bit 4 Defines the type of fixed disk drive installed (drive C):

0000 No fixed disk drive is present.

0001 to 1110 Define type 1 through type 14 as shown in the following table (also see BIOS listing at label FD_TBL)

1111 Type 16 through 255. See "Drive C Extended Byte (Hex 19)" on page 1-68.

Bit 3–Bit 0 Defines the type of second fixed disk drive installed (drive D):

0000 No fixed disk drive is present.

0001 to 1110 Define type 1 through type 14 as shown in the following table (also see BIOS listing at label FD_TBL)

1111 Type 16 through 255. See "Drive D Extended Byte (Hex 1A)" on page 1-68.

The following table shows the BIOS fixed disk parameters.

Type	Cylinders	Heads	Write Precomp	Landing Zone
1	306	4	128	305
2	615	4	300	615
3	615	6	300	615
4	940	8	512	940
5	940	6	512	940
6	615	4	None	615
7	462	8	256	511
8	733	5	None	733
9	900	15	None	901
10	820	3	None	820
11	855	5	None	855
12	855	7	None	855
13	306	8	128	319
14	733	7	None	733
15	Extended Parameters (hex 19 and 1A)			

BIOS Fixed Disk Parameters

Hex address 13 contains a reserved byte.

Equipment Byte (Hex 14)

Bit 7–Bit 6 Indicates the number of diskette drives installed:

00 1 drive
01 2 drives
10 Reserved
11 Reserved

Bit 5–Bit 4 Primary display

00 Primary display is attached to an adapter that has its own BIOS, such as the Enhanced Graphics Adapter

01 Primary display is in the 40-column mode and attached to the Color/Graphics Monitor Adapter.

10 Primary display is in the 80-column mode and attached to the Color/Graphics Monitor Adapter.

11 Primary display is attached to the Monochrome Display and Printer Adapter.

Bit 3–Bit 2 Not used.

Bit 1 Math Coprocessor presence bit:

0 Math Coprocessor not installed
1 Math Coprocessor installed

Bit 0 Diskette drive presence bit:

0 Diskette drive not installed
1 Diskette drive installed

Note: The equipment byte defines basic equipment in the system for power-on diagnostics.

Low and High Base Memory Bytes (Hex 15 and 16)

Bit 7–Bit 0 Address hex 15—Low-byte base size

Bit 7–Bit 0 Address hex 16—High-byte base size

Valid Sizes:

0200H 512K–system board RAM
0280H 640K–system board RAM.

Low and High Expansion Memory Bytes (Hex 17 and 18)

Bit 7–Bit 0 Address hex 17—Low-byte expansion size

Bit 7-Bit 0 Address hex 18—High-byte expansion size

Valid Sizes:

0200H 512K–Expansion Memory
0400H 1024K–Expansion Memory
0600H 1536K–Expansion Memory
through
3C00H 15360K–Expansion Memory (15M maximum).

Drive C Extended Byte (Hex 19)

Bit 7–Bit 0 Defines the type of first fixed disk drive installed (drive C):

00000000 through 00001111 are reserved.

00010000 to 11111111 define type 16 through 255 as shown in the following table (see BIOS listing at label FD_TBL).

Drive D Extended Byte (Hex 1A)

Bit 7–Bit 0 Defines the type of second fixed disk drive installed (drive D):

00000000 through 00001111 are reserved.

00010000 to 11111111 define type 16 through 255 as shown in the following table (see BIOS listing at label FD_TBL).

The following table shows the BIOS fixed disk parameters for fixed disk drive types 16 through 24.

Note: Types 25 through 255 are reserved.

Type	Cylinders	Heads	Write Precomp	Landing Zone
16	612	4	All Cylinders	663
17	977	5	300	977
18	977	7	None	977
19	1024	7	512	1023
20	733	5	300	732
21	733	7	300	732
22	733	5	300	733
23	306	4	None	336
24	612	4	305	663
25	Reserved			
.	.			
.	.			
255	Reserved			

BIOS Fixed Disk Parameters (Extended)

Hex addresses 1B through 2D are reserved.

Checksum (Hex 2E and 2F)

Bit 7–Bit 0 Address hex 2E—High byte of checksum

Bit 7–Bit 0 Address hex 2F—Low byte of checksum

Note: Checksum is calculated on addresses hex 10-2D.

Low and High Expansion Memory Bytes (Hex 30 and 31)

Bit 7–Bit 0 Address hex 30—Low-byte expansion size

Bit 7–Bit 0 Address hex 31—High-byte expansion size

Valid Sizes:

0200H 512K–Expansion Memory
0400H 1024K–Expansion Memory
0600H 1536K–Expansion Memory
through
3C00H 15360K–Expansion Memory (15M maximum).

Note: These bytes reflect the total expansion memory above the 1M address space as determined at power-on time. This expansion memory size can be determined through system interrupt 15 (see the BIOS listing). The base memory at power-on time is determined through the system memory-size-determine interrupt (hex 12).

Date Century Byte (Hex 32)

Bit 7–Bit 0 BCD value for the century (BIOS interface to read and set).

Information Flag (Hex 33)

Bit 7	When set, this bit indicates that the top 128K of base memory is installed.
Bit 6	This bit is set to instruct the Setup utility to put out a first user message after initial setup.
Bit 5–Bit 0	Reserved

Hex addresses 34 through 3F are reserved.

I/O Operations

Writing to RTC CMOS RAM involves two steps:

1. OUT to port hex 70 with the internal CMOS address. Bits D0 - D5 contain the required address.

   Note: Bits D6 and D7 do not go to RTC CMOS RAM. D6 is a "don't care" bit. D7 is the value of the NMI mask: writing a 1 to D7 disables NMI; writing a 0 to D7 enables NMI.

2. OUT to port hex 71 with the data to be written.

Reading CMOS RAM also requires two steps:

1. OUT to port hex 70 with the internal CMOS address. Bits D0 - D5 contain the required address.

   Note: Bits D6 and D7 do not go to RTC CMOS RAM. D6 is a "don't care" bit. D7 is the value of the NMI mask: writing a 1 to D7 disables NMI; writing a 0 to D7 enables NMI.

2. IN from port hex 71, and the data read is returned in the AL register.

Note: Execute the steps in the order shown to ensure acknowledgement of the MC146818A Standby lead during system power-downs.

Specifications

System Unit

Size

- Length: 500 millimeters (19.6 inches)
- Depth: 410 millimeters (16.1 inches)
- Height: 142 millimeters (5.5 inches)

Weight

- 12.7 kilograms (28 pounds)

Power Cables

- Length: 1.8 meters (6 feet)

Environment

- Air Temperature
  - System On: 15.6 to 32.2 degrees C (60 to 90 degrees F)
  - System Off: 10 to 43 degrees C (50 to 110 degrees F)
- Wet Bulb Temperature
  - System On: 22.8 degrees C (73 degrees F)
  - System Off: 26.7 degrees C (80 degrees F)
- Humidity
  - System On: 8% to 80%

  - System Off: 20% to 80%

- Altitude
  - Maximum altitude: 2545.1 meters (8350 feet)

Heat Output

- 824 British Thermal Units (BTU) per hour

Noise Level

- Operating (without display or printer) - 46 decibels (dba) maximum noise level.

Electrical

- Range 1 (57-63 Hz)
  - Nominal: 115 Vac
  - Minimum: 90 Vac
  - Maximum: 137 Vac
- Range 2 (47-53 Hz)
  - Nominal: 230 Vac
  - Minimum: 180 Vac
  - Maximum: 265 Vac
- Lithium Battery
  - 6.0 Vdc
  - 1 Ampere/Hour Capacity
  - UL Approved.

Connectors

The system board has the following additional connectors:

- One power supply connector (P1)
- Battery connector (P2)
- Speaker connector (P3)
- Keyboard connector (J9).

The pin assignments for the system board connector (P1) and the power supply connectors P8 and P9, are as follows. Beginning at the rear of the system, the pins on the system board connector are numbered 1 through 12. Power supply connector P8 attaches to system board connector P1, pins 1 through 6. P9 connects to P1, pins 7 through 12.

System Board Connector	Pin	Assignments	Power Supply Connector	Pin
P1	1	Power Good	P8	1
	2	+5 Vdc		2
	3	+12 Vdc		3
	4	-12 Vdc		4
	5	Ground		5
	6	Ground		6
	7	Ground	P9	1
	8	Ground		2
	9	-5 Vdc		3
	10	+5 Vdc		4
	11	+5 Vdc		5
	12	+5 Vdc		6

System Board Connector (P1) to Power Supply Connectors (P8 and P9)

The battery connector, P2, is a four-pin, keyed, Berg strip. The pins are numbered 1 through 4 from the rear of the system. The pin assignments are:

Pin	Assignments
1	+6 Vdc
2	Key
3	Ground
4	Ground

Battery Connector (P2)

The speaker connector, P3, is a four-pin, keyed, Berg strip. The pins are numbered 1 through 4 from the rear of the system. The pin assignments are:

Pin	Function
1	Data out
2	Key
3	Ground
4	+5 Vdc

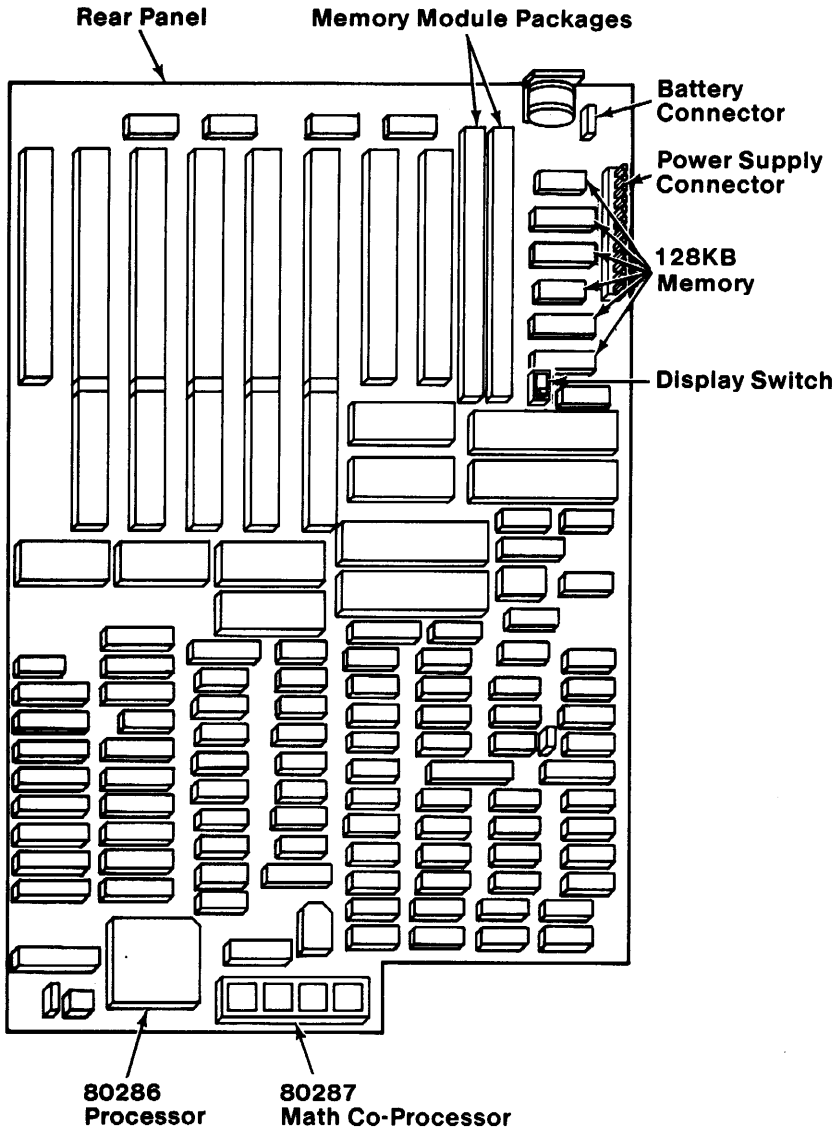
Speaker Connector (P3)

The keyboard connector, J9, is a five-pin, 90-degree Printed Circuit Board (PCB) mounting, DIN connector. For pin numbering, see the "Keyboard" Section. The pin assignments are:

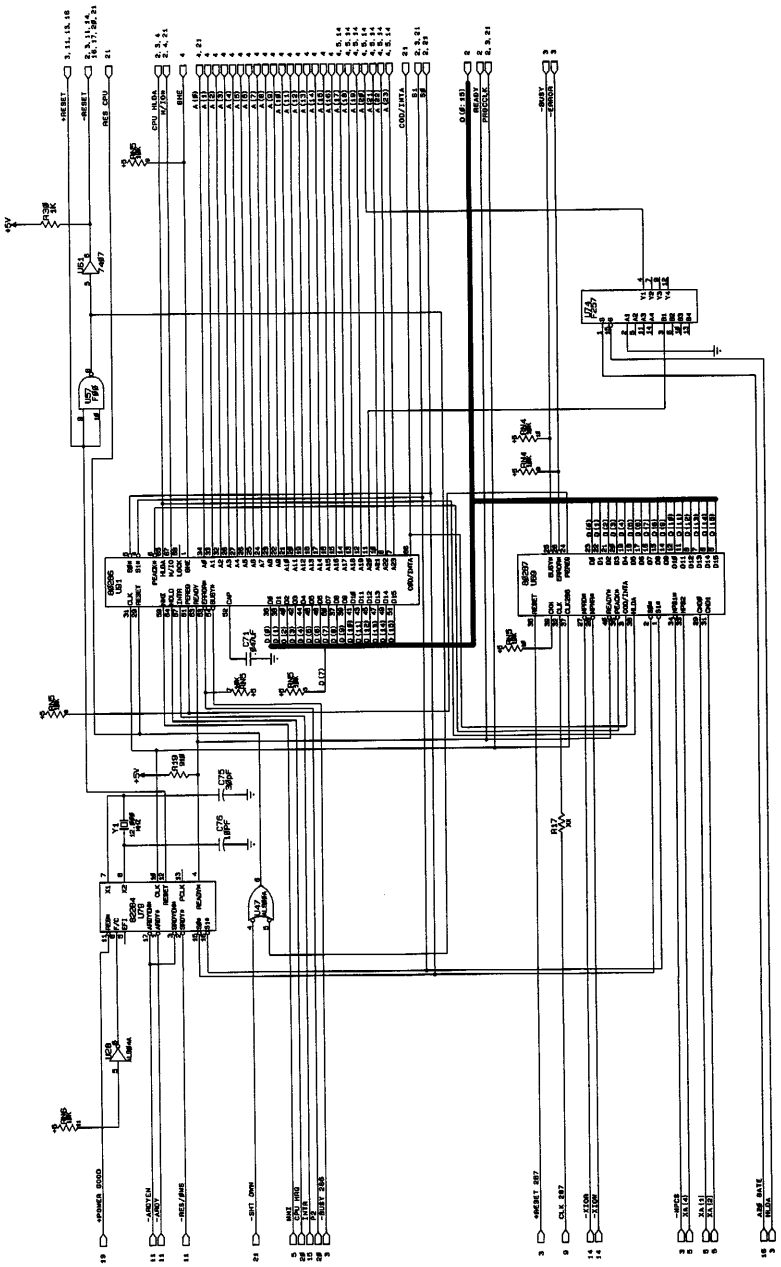
Pin	Assignments
1	Keyboard Clock
2	Keyboard Data
3	Reserved
4	Ground
5	+5 Vdc

Keyboard Connector (J9)

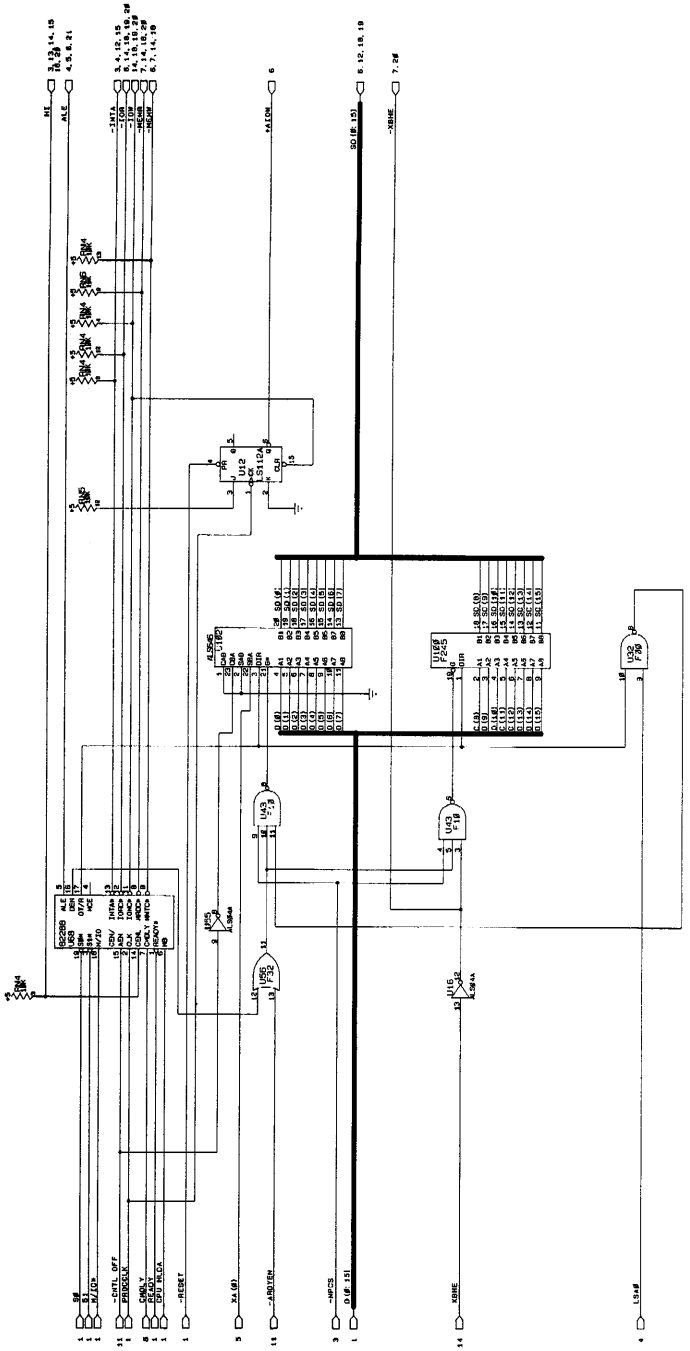
The following figure shows the layout of the system board.



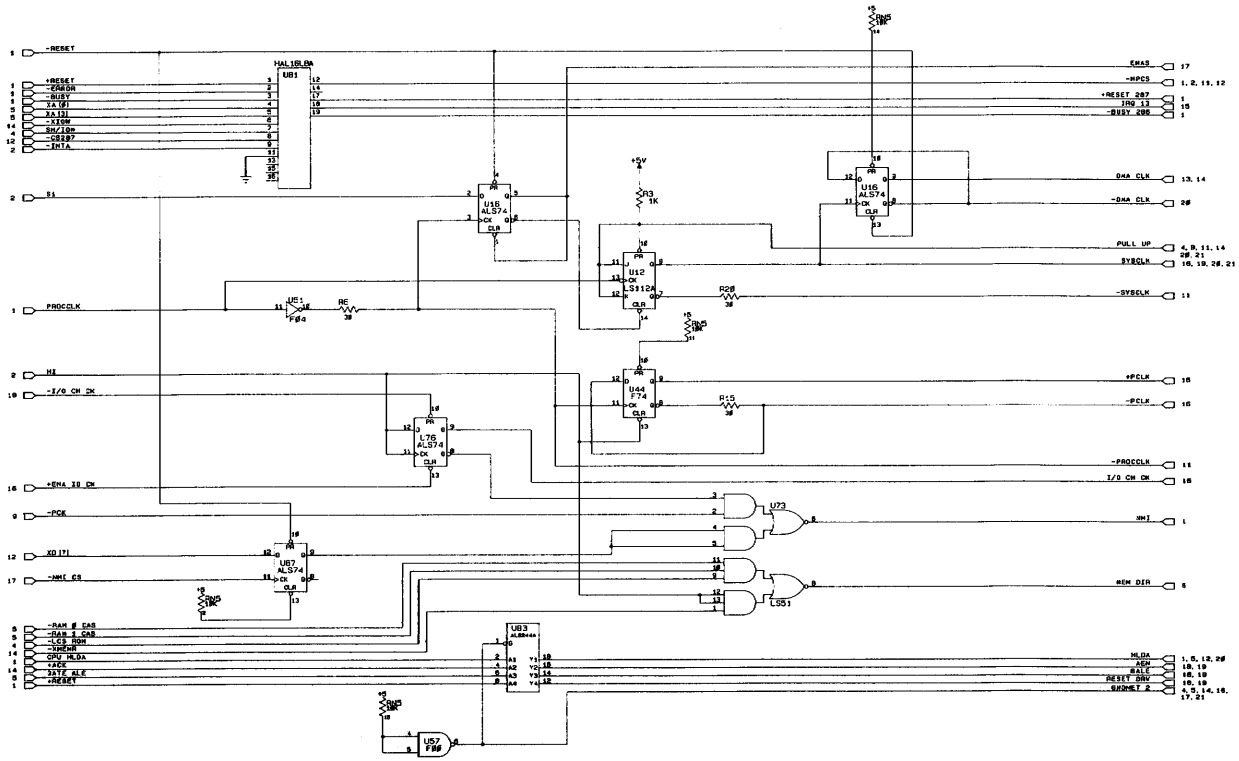
Logic Diagrams



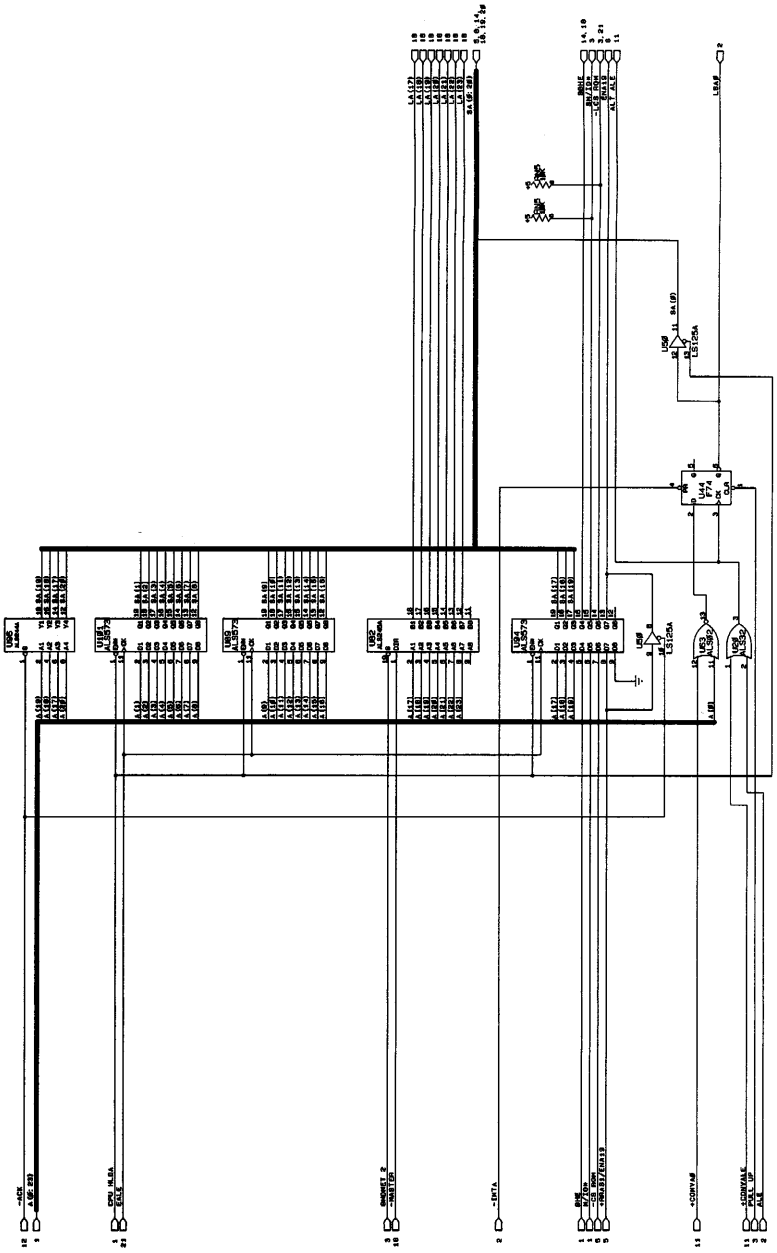
System Board (Sheet 1 of 22)



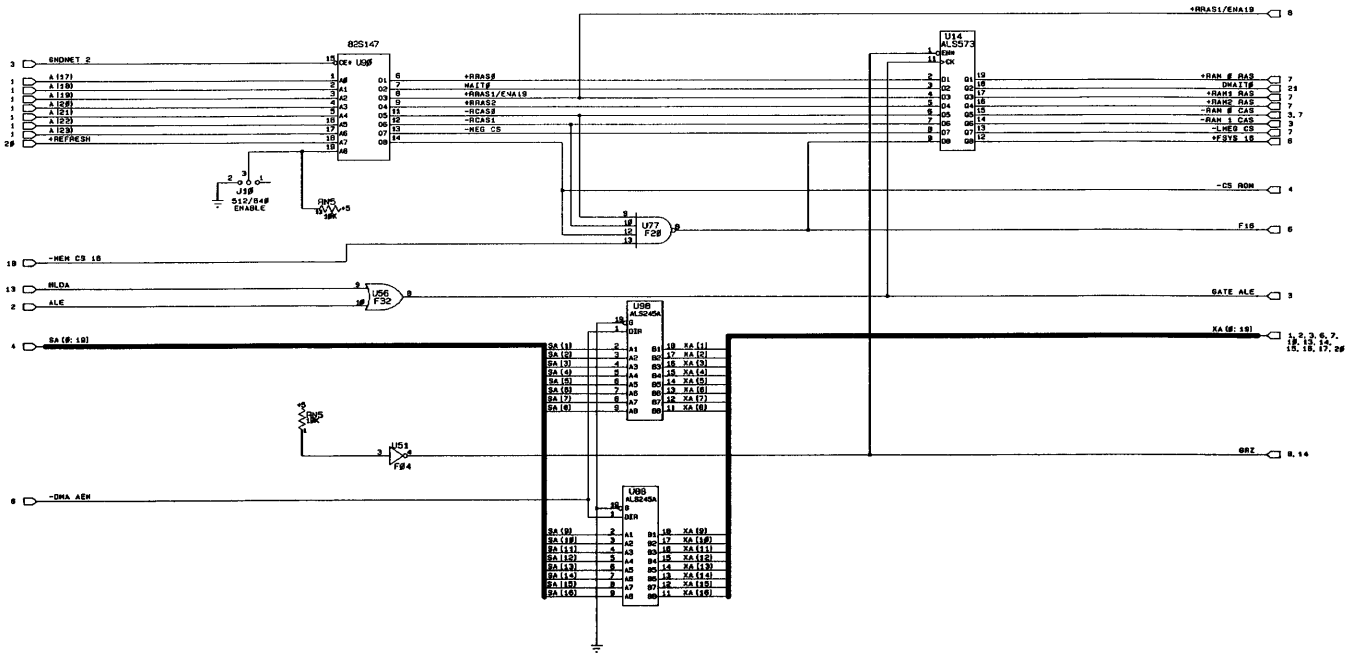
System Board (Sheet 2 of 22)



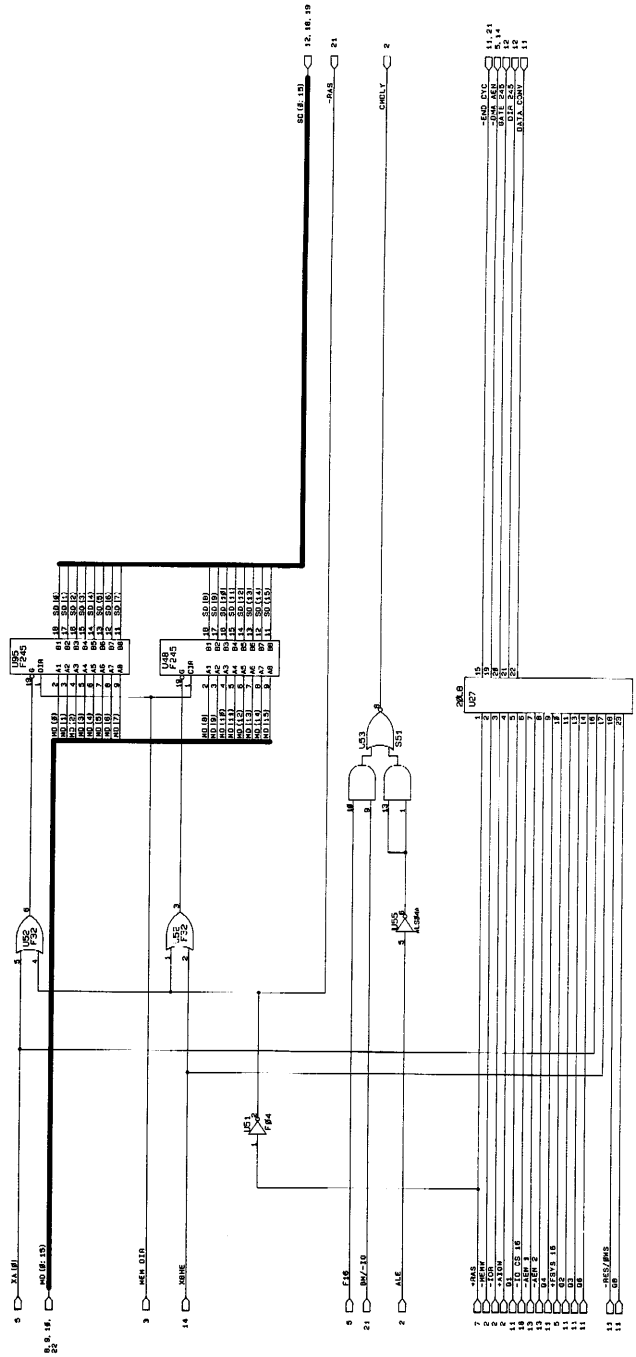
System Board (Sheet 3 of 22)



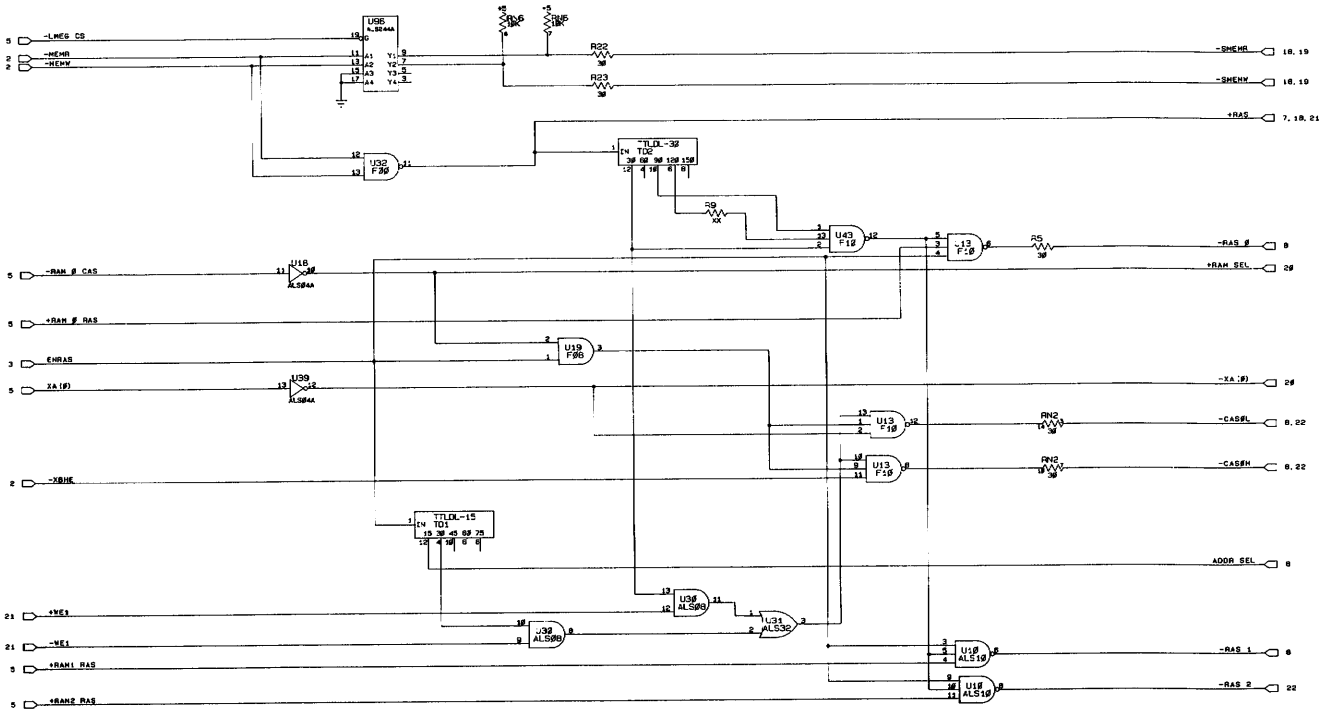
System Board (Sheet 4 of 22)



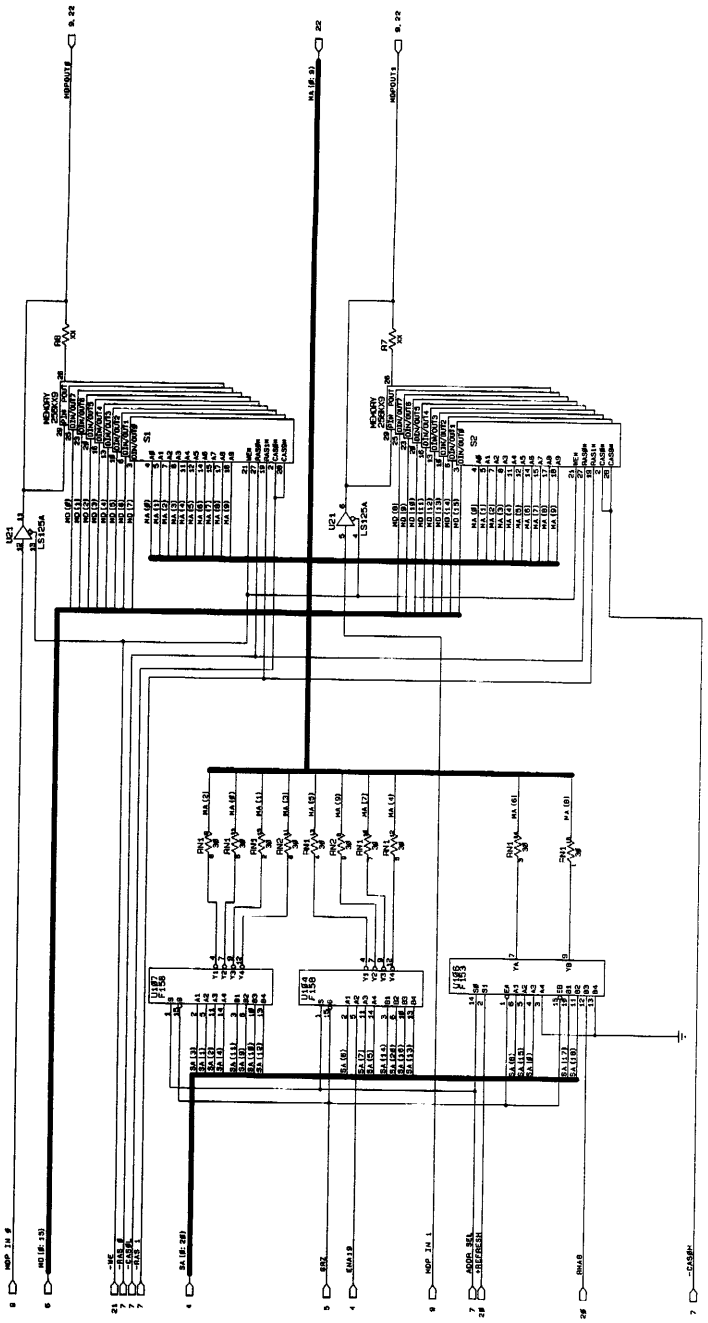
System Board (Sheet 5 of 22)



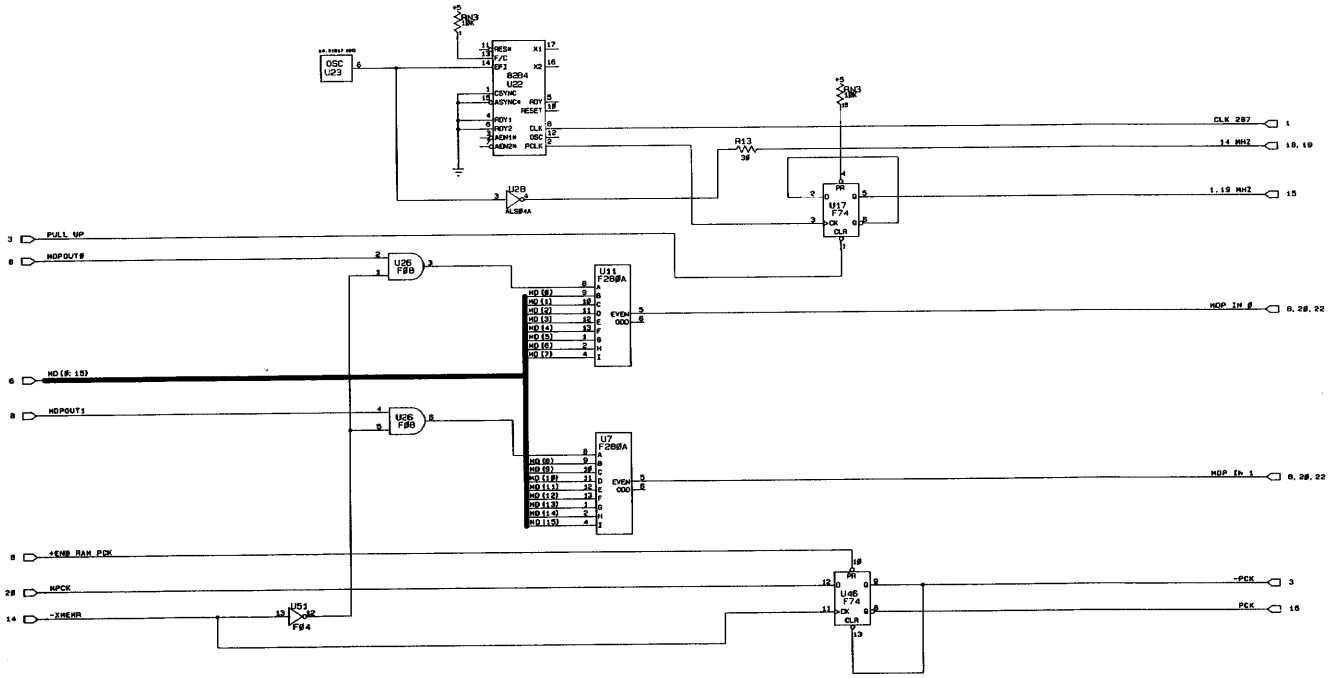
System Board (Sheet 6 of 22)



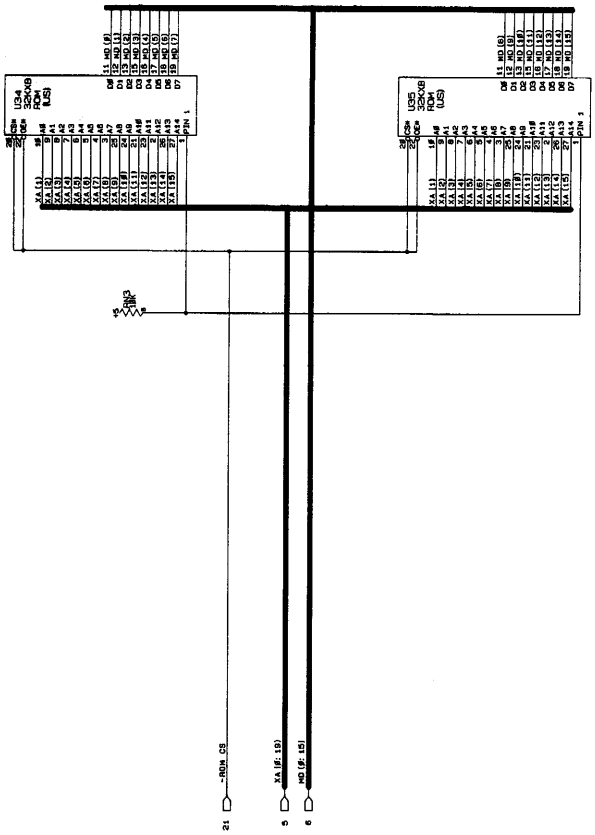
System Board (Sheet 7 of 22)



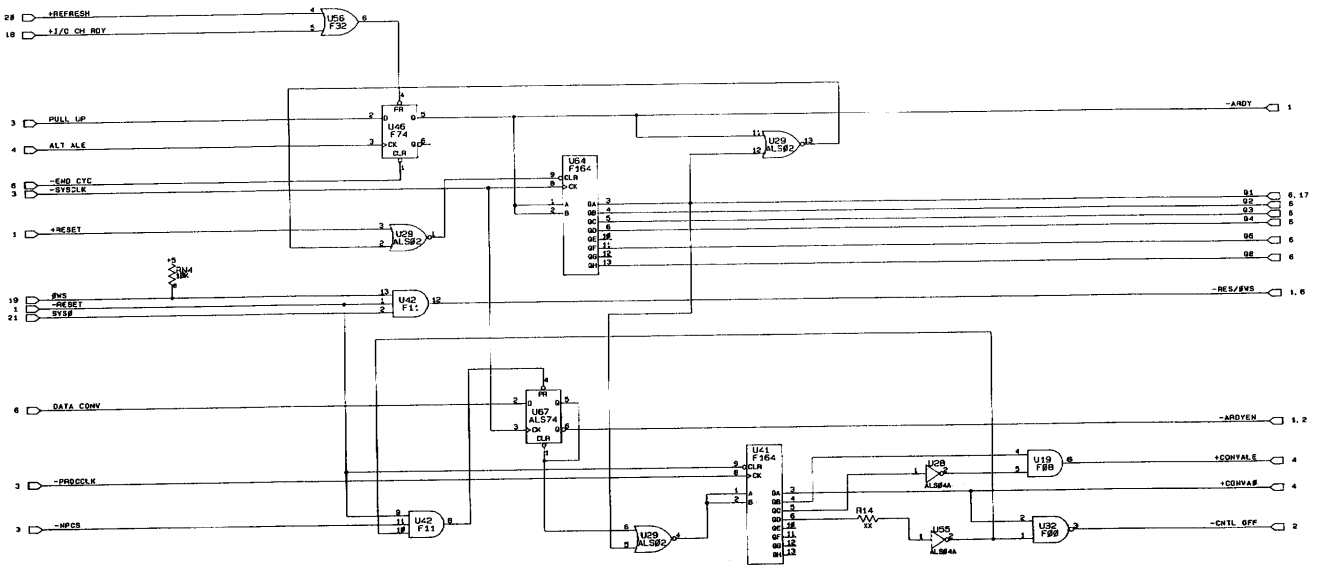
System Board (Sheet 8 of 22)



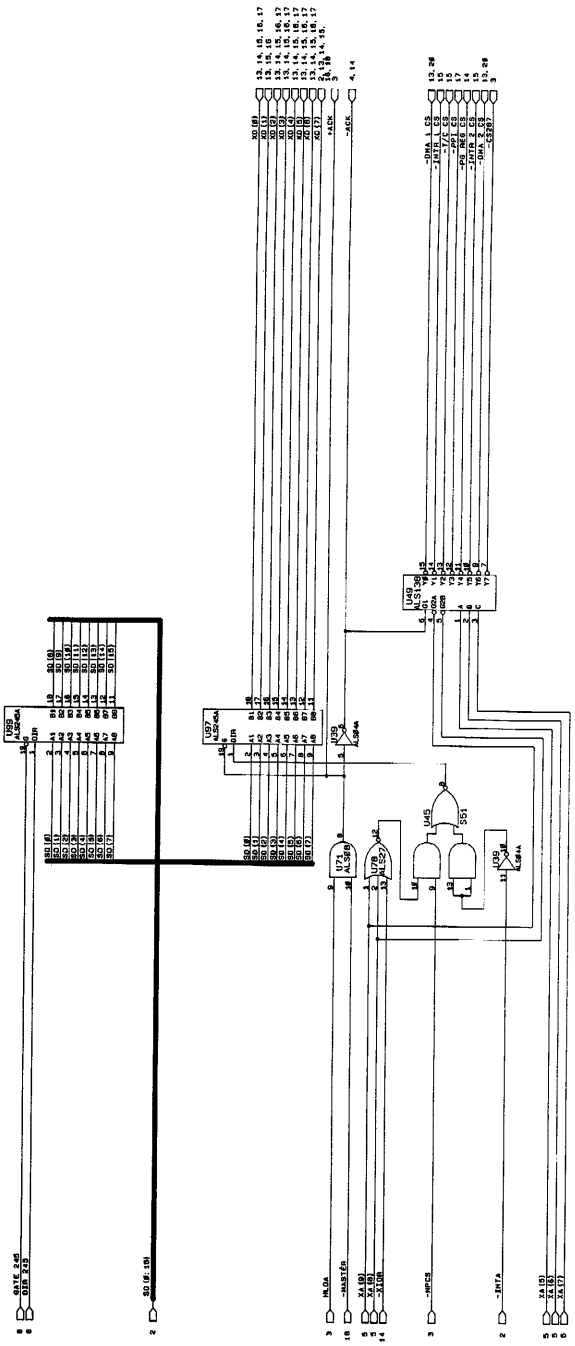
System Board (Sheet 9 of 22)



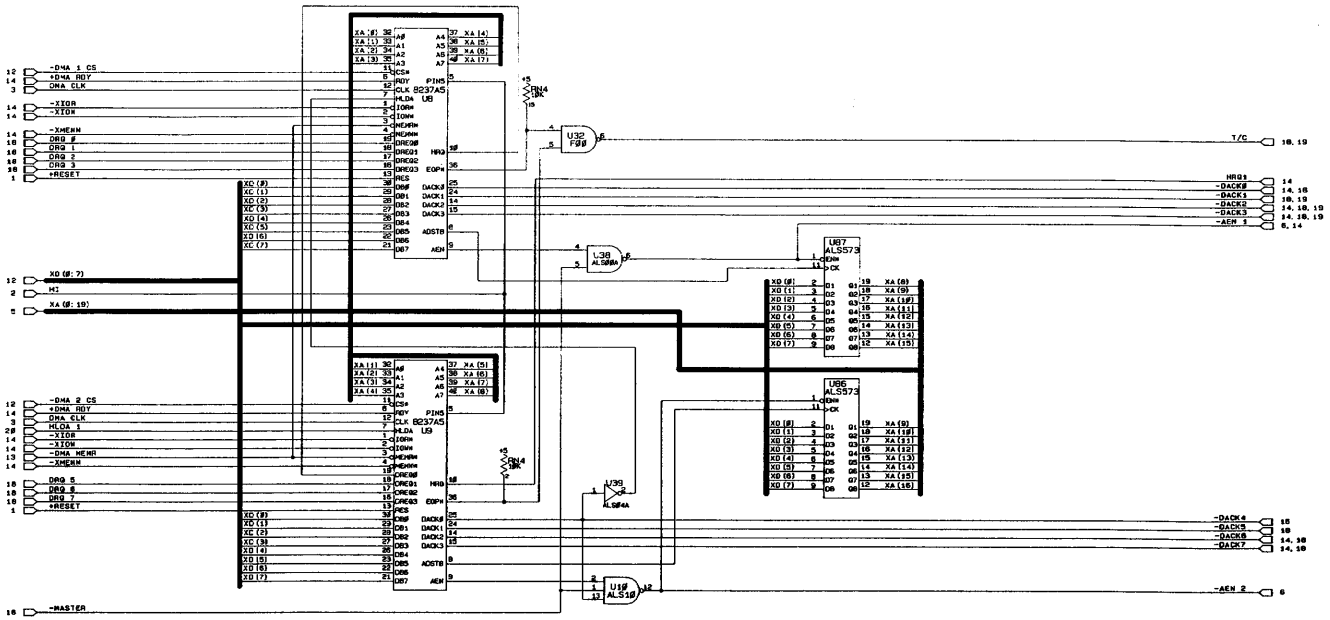
System Board (Sheet 10 of 22)



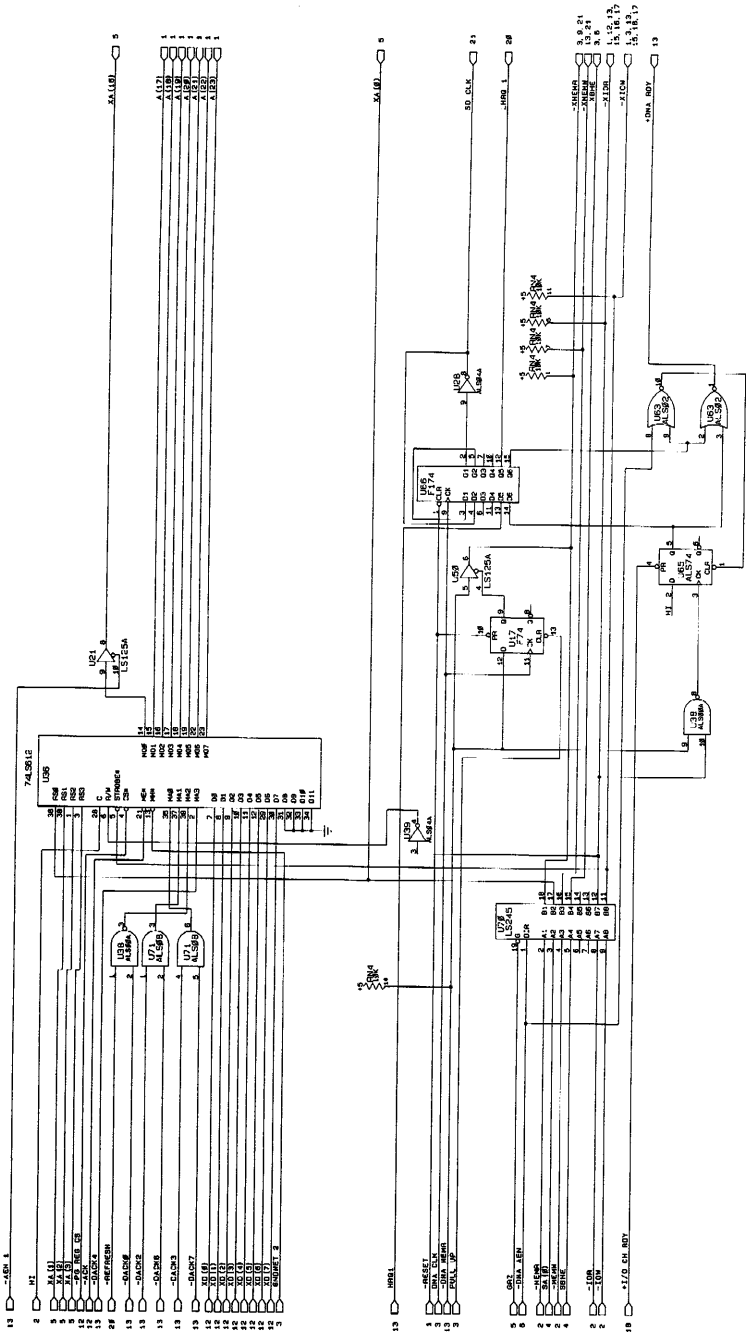
System Board (Sheet 11 of 22)



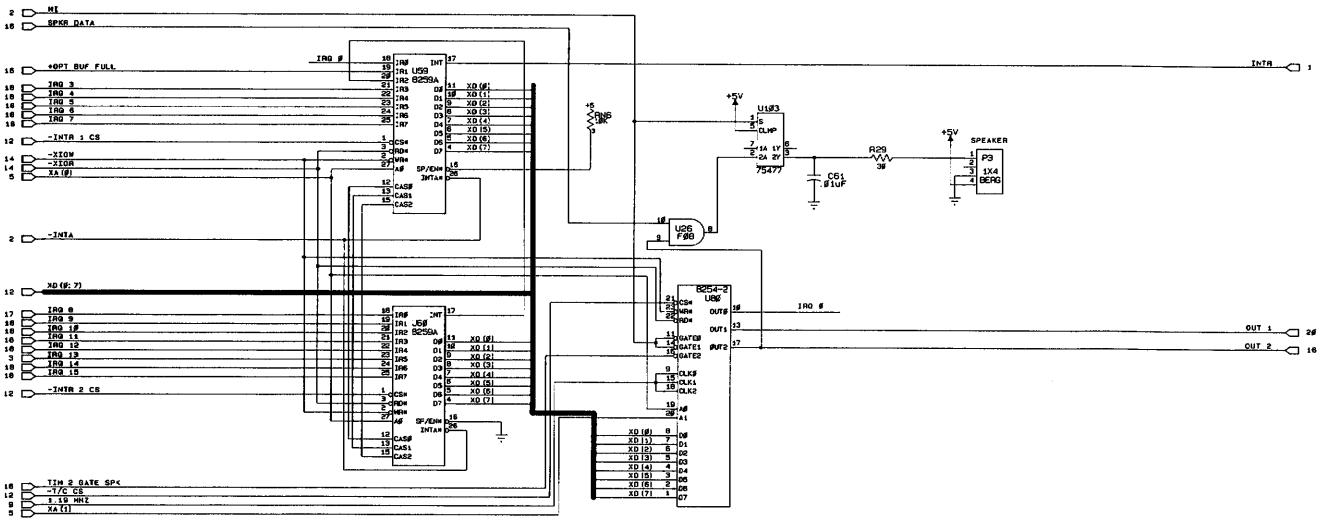
System Board (Sheet 12 of 22)



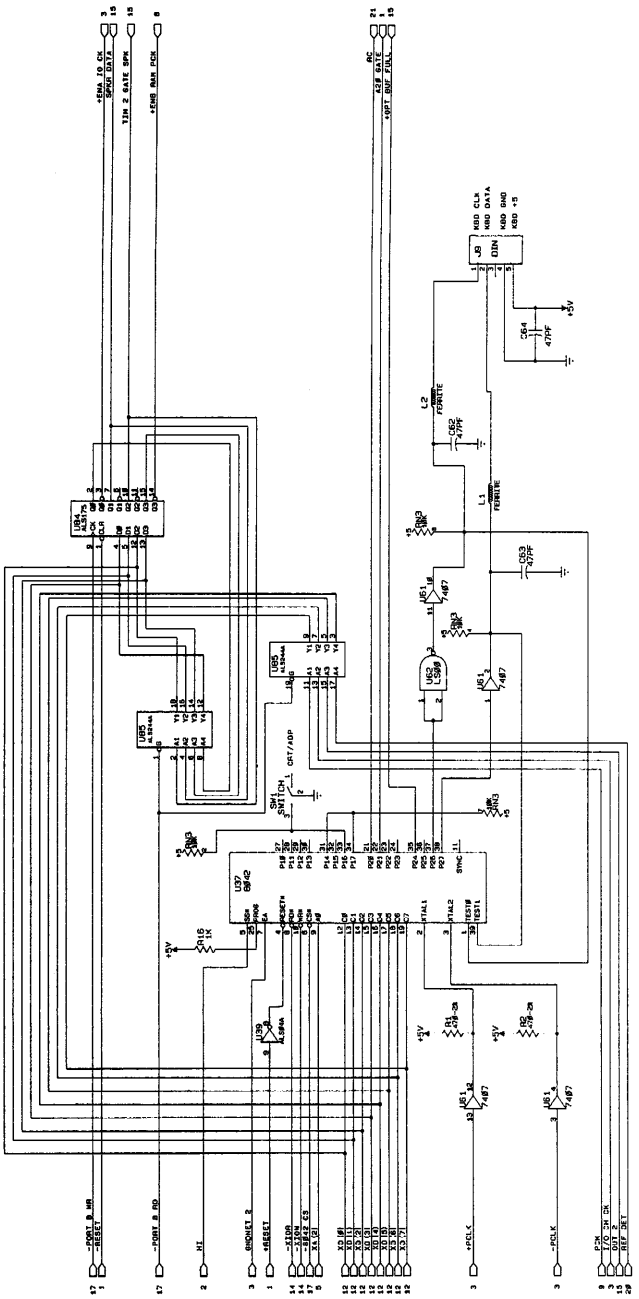
System Board (Sheet 13 of 22)



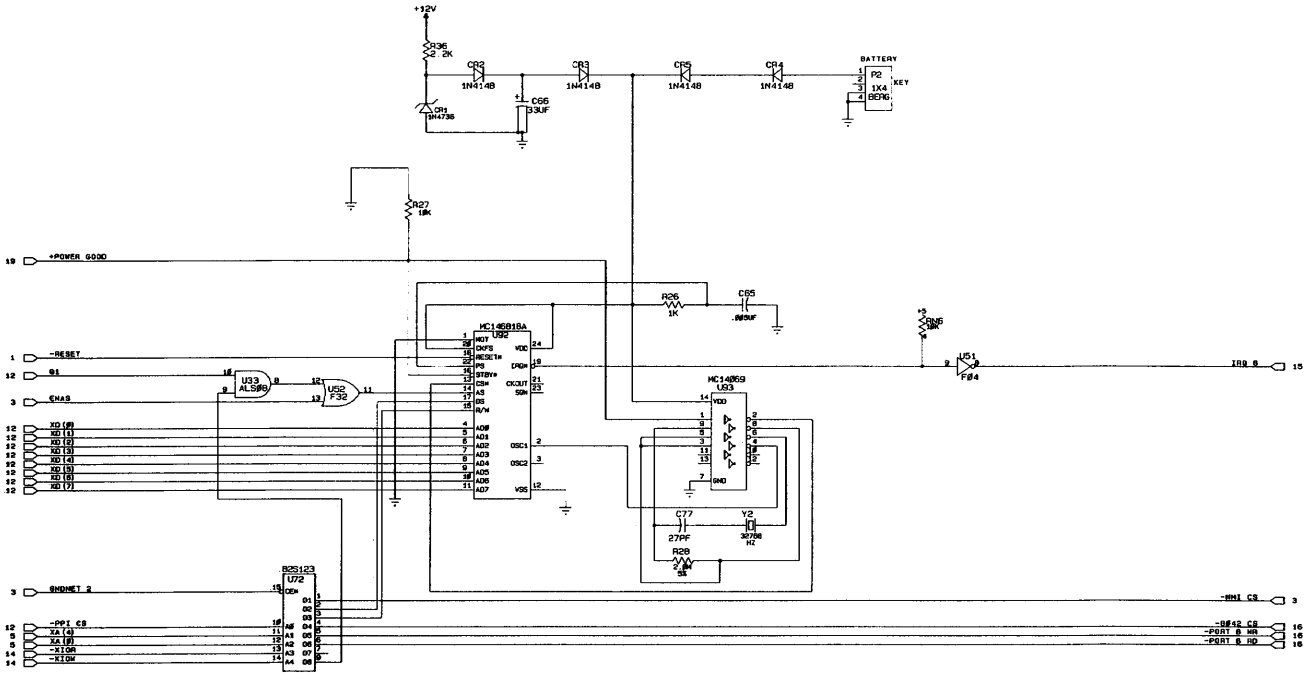
System Board (Sheet 14 of 22)



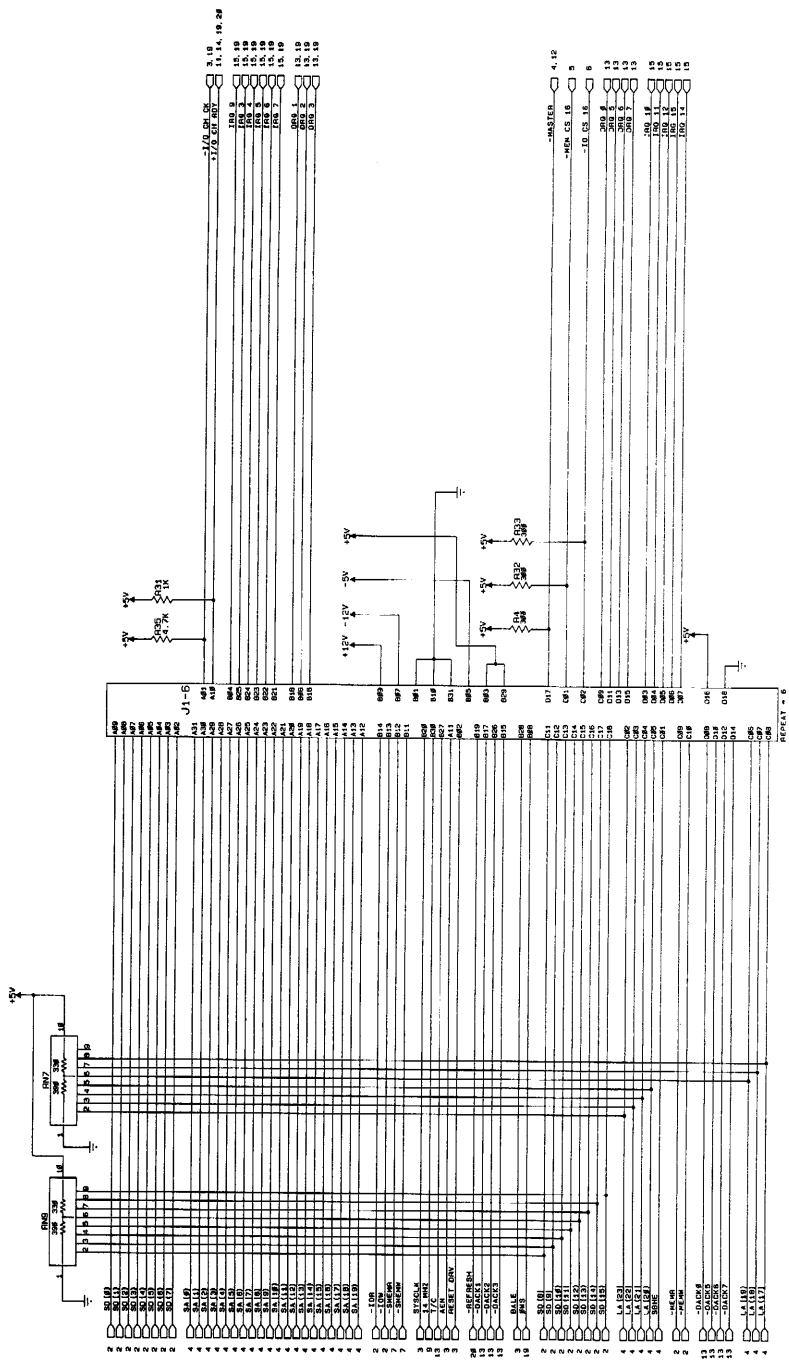
System Board (Sheet 15 of 22)



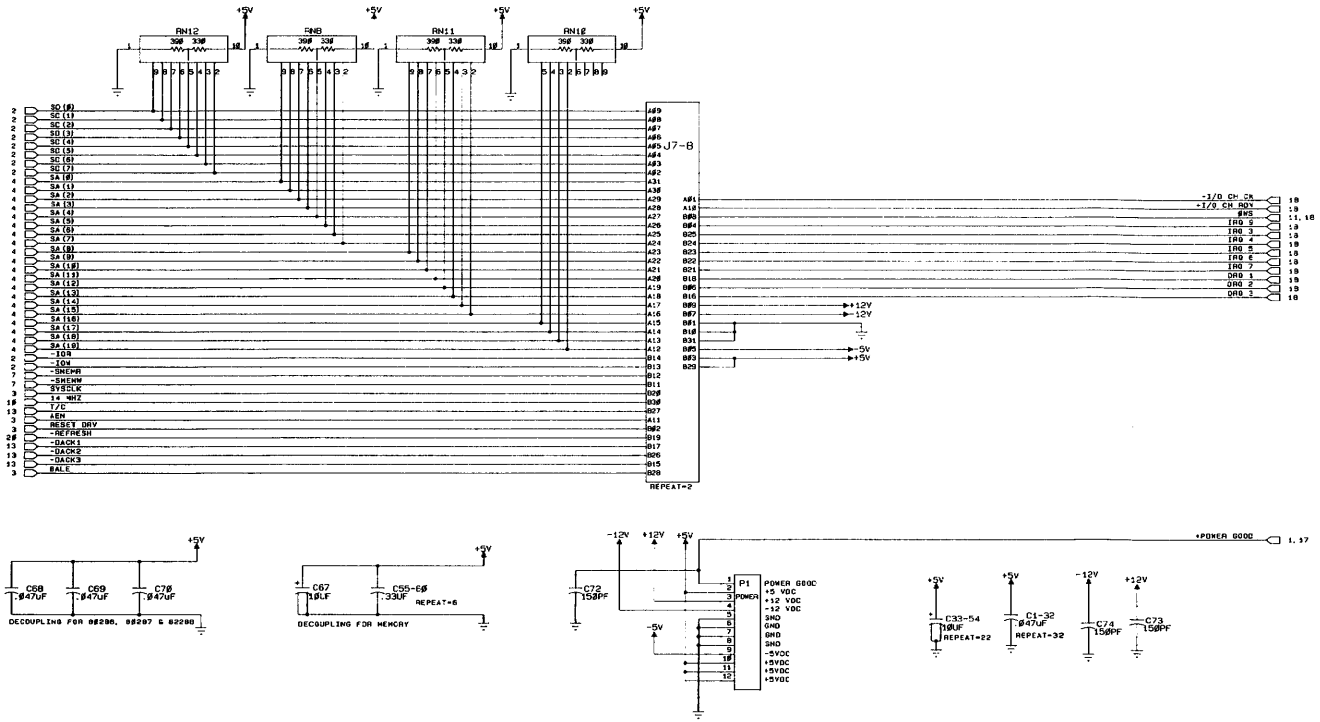
System Board (Sheet 16 of 22)



System Board (Sheet 17 of 22)

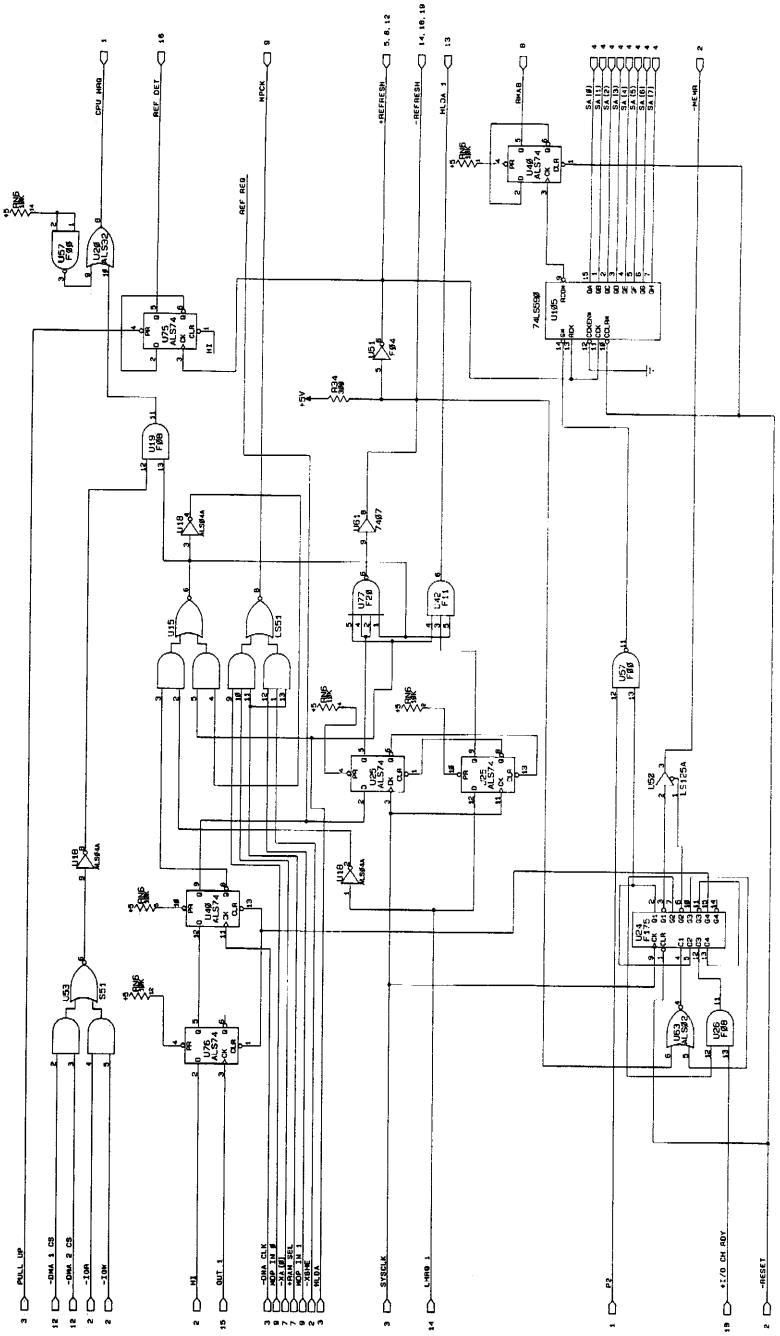


System Board (Sheet 18 of 22)

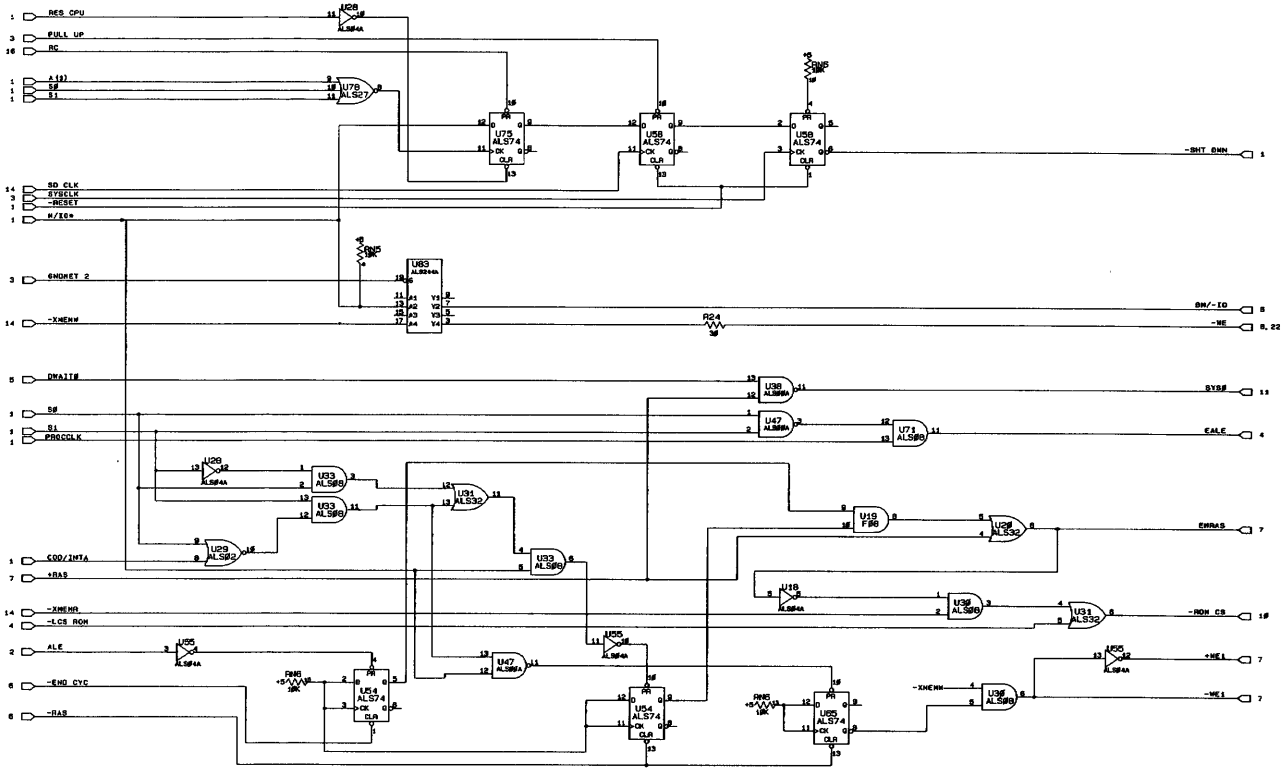


System Board (Sheet 19 of 22)

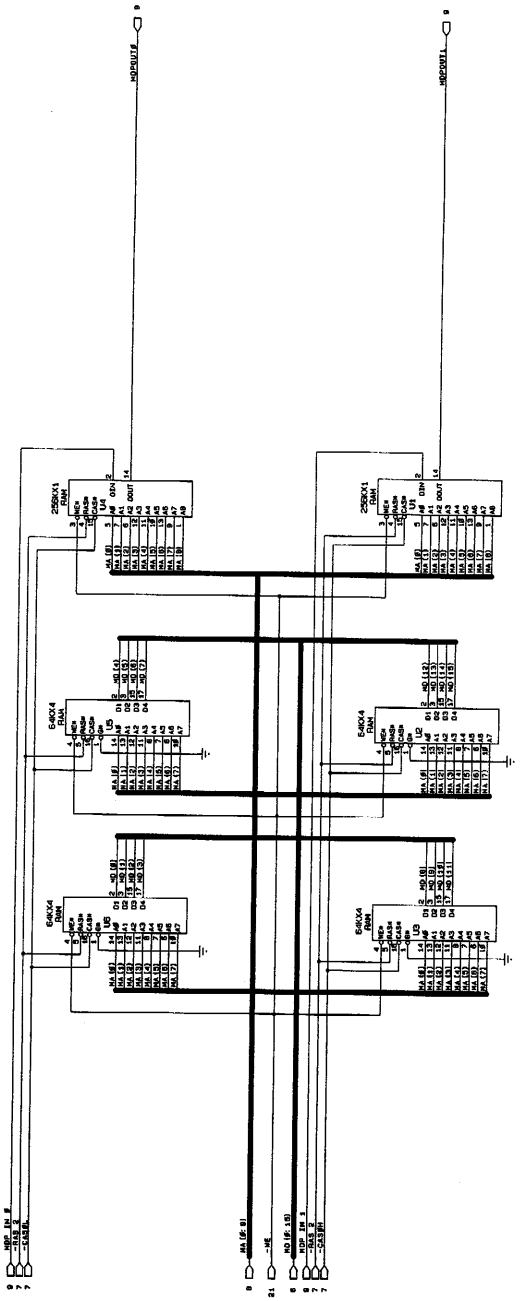
SECTION 1



System Board (Sheet 20 of 22)



System Board (Sheet 21 of 22)



System Board (Sheet 22 of 22)

SECTION 2. COPROCESSOR

Notes:

Description

The Math Coprocessor (80287) enables the IBM Personal Computer XT Model 286 to perform high-speed arithmetic, logarithmic functions, and trigonometric operations.

The coprocessor works in parallel with the microprocessor. The parallel operation decreases operating time by allowing the coprocessor to do mathematical calculations while the microprocessor continues to do other functions.

The coprocessor works with seven numeric data types, which are divided into the following three classes:

- Binary integers (3 types)
- Decimal integers (1 type)
- Real numbers (3 types).

Programming Interface

The coprocessor offers extended data types, registers, and instructions to the microprocessor.

The coprocessor has eight 80–bit registers, which provide the equivalent capacity of forty 16–bit registers. This register space allows constants and temporary results to be held in registers during calculations, thus reducing memory access and improving speed as well as bus availability. The register space can be used as a stack or as a fixed register set. When used as a stack, only the top two stack elements are operated on.

The following figure shows representations of large and small numbers in each data type.

Data Type	Bits	Significant Digits (Decimal)	Approximate Range (Decimal)
Word Integer	16	4	$-32{,}768 \leq X \leq +32{,}767$
Short Integer	32	9	$-2 \times 10^{9} \leq X \leq +2 \times 10^{9}$
Long Integer	64	18	$-9 \times 10^{18} \leq X \leq +9 \times 10^{18}$
Packed Decimal	80	18	-9..99 ≤ X ≤ +9..99 (18 digits)
Short Real *	32	6-7	$8.43 \times 10^{-37} \leq \lvert X \rvert \leq 3.37 \times 10^{38}$
Long Real *	64	15-16	$4.19 \times 10^{-307} \leq \lvert X \rvert \leq 1.67 \times 10^{308}$
Temporary Real	80	19	$3.4 \times 10^{-4932} \leq \lvert X \rvert \leq 1.2 \times 10^{4932}$

Data Types

* The Short Real and Long Real data types correspond to the single and double precision data types.

Hardware Interface

The coprocessor uses a 4.77 MHz clock (generated by a 14.318 MHz clock generator divided by three). The coprocessor is wired so that it functions as an I/O device through I/O port addresses hex 00F8, 00FA, and 00FC. The microprocessor sends OP codes and operands through these I/O ports. The microprocessor also receives and stores results through the same I/O ports. The coprocessor's 'busy' signal informs the microprocessor that it is executing; the microprocessor's Wait instruction forces the microprocessor to wait until the coprocessor is finished executing.

The coprocessor detects six different exception conditions that can occur during instruction execution. If the appropriate exception mask within the coprocessor is not set, the coprocessor sets its error signal. This error signal generates a hardware interrupt (interrupt 13) and causes the 'busy' signal to the coprocessor to be held in the busy state. The 'busy' signal may be cleared by an 8-bit I/O Write command to address hex F0 with D0 through D7 equal to 0.

The power-on self-test code in the system ROM enables IRQ 13 and sets up its vector to point to a routine in ROM. The ROM routine clears the 'busy' signal's latch and then transfers control

to the address pointed to by the NMI interrupt vector. This allows code written for any IBM Personal Computer to work on an IBM Personal Computer XT Model 286. The NMI interrupt handler should read the coprocessor's status to determine if the NMI was caused by the coprocessor. If the interrupt was not generated by the coprocessor, control should be passed to the original NMI interrupt handler.

The coprocessor has two operating modes similar to the two modes of the microprocessor. When reset by a power-on reset, system reset, or an I/O write operation to port hex 00F1, the coprocessor is in the real address mode. This mode is compatible with the 8087 Math Coprocessor used in other IBM Personal Computers. The coprocessor can be placed in the protected mode by executing the SETPM ESC instruction. It can be placed back in the real mode by an I/O write operation to port hex 00F1, with D7 through D0 equal to 0.

The coprocessor instruction extensions to the microprocessor can be found in Section 6 of this manual.

Detailed information for the internal functions of the Intel 80287 Coprocessor can be found in books listed in the bibliography.

Notes:

SECTION 3. POWER SUPPLY

Notes:

The system power supply is located *inside* the system unit and provides power for the system board, the adapters, the diskette drives, the fixed disk drive, the keyboard, and the IBM Monochrome Display.

Inputs

The power supply can operate at 110 Vac, 4.6 A or 220/240 Vac, 2.3 A at frequencies of either 60 ±3 Hz or 50 ±3 Hz. The power supply automatically adjusts to input voltages of 110 Vac or 220 Vac. The following figure shows the input requirements.

Range	Voltage (Vac)	Current (Amperes)
115 Vac	Minimum 90 Maximum 137	Maximum 4.6
230 Vac	Minimum 180 Maximum 265	Maximum 2.3

Input Requirements

Outputs

The power supply provides +5, -5, +12, and -12 Vdc. The following figure shows the load current and regulation tolerance for these voltages. The power to the IBM Monochrome Display display is controlled by the power supply.

Warning: The voltage provided to the monochrome display from the power supply is the same as the input line voltage to the power supply. Ensure that the monochrome display is the correct model for the input line voltage.

Nominal Output	Load Current (A) Minimum	Maximum	Regulation Tolerance
+5 Vdc	4.0	20.0	+5% to -4%
-5 Vdc	0.0	0.3	+10% to -8%
+12 Vdc	1.0	4.2	+5% to -4%
-12 Vdc	0.0	0.25	+10% to -9%

DC Load Requirements

DC Output Protection

An overcurrent condition will not damage the power supply.

Output Voltage Sequencing

Under normal conditions, the output voltage levels track within 50 milliseconds of each other when power is applied to, or removed from the power supply, provided at least minimum loading is present.

No-Load Operation

No damage or hazardous conditions occur when primary power is applied with no load on any output level. In such cases, the power supply may switch off, and a power-on reset will be required. The power supply requires a minimum load for proper operation.

Power-Good Signal

The power supply provides a 'power-good' signal to indicate proper operation of the power supply.

When the supply is switched off for a minimum of one second and then switched on, the 'power-good' signal is generated, assuming there are no problems. This signal is a logical AND of the dc output-voltage sense signal and the ac input-voltage sense signal. The 'power-good' signal is also a TTL-compatible high level for normal operation, and a low level for fault conditions. The ac fail signal causes 'power-good' to go to a low level at least one millisecond before any output voltage falls below the regulation limits. The operating point used as a reference for measuring the one millisecond is normal operation at minimum line voltage and maximum load.

The dc output-voltage sense signal holds the 'power-good' signal at a low level when power is switched on until all output voltages have reached their minimum sense levels. The 'power-good' signal has a turn-on delay of at least 100 milliseconds but not

longer than 500 milliseconds and is capable of sourcing 2 milliamperes and sinking 10 milliamperes.

The following figure shows the minimum sense levels for the output voltages.

Level (Vdc)	Minimum (Vdc)
+5	+4.5
-5	-4.3
+12	+10.8
-12	-10.2

Sense Level

Connectors

The following figure shows the pin assignments for the power-supply output connectors.

Load Point	Voltage (Vdc)
P8-1	Power Good *
P8-2	+5
P8-3	+12
P8-4	-12
P8-5	Ground
P8-6	Ground
P9-1	Ground
P9-2	Ground
P9-3	-5
P9-4	+5
P9-5	+5
P9-6	+5
P10-1	+12
P10-2	Ground
P10-3	Ground
P10-4	+5
P11-1	+12
P11-2	Ground
P11-3	Ground
P11-4	+5
* see "Power-Good Signal"	

Power Supply Output Connectors

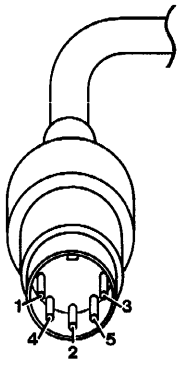
SECTION 4. KEYBOARD

Description

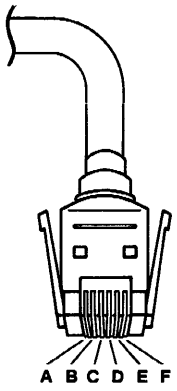
The keyboard has 101 keys (102 in countries outside the U. S.). At system power-on, the keyboard monitors the signals on the 'clock' and 'data' lines and establishes its line protocol. A bidirectional serial interface in the keyboard converts the 'clock' and 'data' signals and sends this information to and from the keyboard through the keyboard cable.

Cabling

The keyboard cable connects to the system with a five-pin DIN connector, and to the keyboard with a six-position SDL connector. The following table shows the pin configuration and signal assignments.



DIN Connector



SDL Connector

DIN Connector Pins	SDL Connector Pins	Signal Name	Signal Type
1	D	+KBD CLK	Input/Output
2	B	+KBD DATA	Input/Output
3	F	Reserved	
4	C	Ground	Ground
5	E	+5.0 Vdc	Power
	A	Not used	
Shield	Shield	Frame Ground	

Sequencing Key-Code Scanning

The keyboard detects all keys pressed, and sends each scan code in the correct sequence. When not serviced by the system, the keyboard stores the scan codes in its buffer.

Keyboard Buffer

A 16-byte first-in-first-out (FIFO) buffer in the keyboard stores the scan codes until the system is ready to receive them.

A buffer-overrun condition occurs when more than 16 bytes are placed in the keyboard buffer. An overrun code replaces the 17th byte. If more keys are pressed before the system allows keyboard output, the additional data is lost.

When the keyboard is allowed to send data, the bytes in the buffer will be sent as in normal operation, and new data entered is detected and sent. Response codes do not occupy a buffer position.

If keystrokes generate a multiple-byte sequence, the entire sequence must fit into the available buffer space or the keystroke is discarded and a buffer-overrun condition occurs.

Keys

With the exception of the Pause key, all keys are *make/break*. The make scan code of a key is sent to the keyboard controller when the key is pressed. When the key is released, its break scan code is sent.

Additionally, except for the Pause key, all keys are *typematic*. When a key is pressed and held down, the keyboard sends the make code for that key, delays 500 milliseconds ± 20%, and begins sending a make code for that key at a rate of 10.9 characters per second ± 20%. The typematic rate and delay can be modified [see "Set Typematic Rate/Delay (Hex F3)" on page 4-11].

If two or more keys are held down, only the last key pressed repeats at the typematic rate. Typematic operation stops when

the last key pressed is released, even if other keys are still held down. If a key is pressed and held down while keyboard transmission is inhibited, only the first make code is stored in the buffer. This prevents buffer overflow as a result of typematic action.

Note: Scan code set 3 allows key types to be changed by the system. See "Scan Code Tables (Set 3)" on page 4-24 for the default settings. Commands to change the default settings are listed in "Commands from the System" on page 4-6.

Power-On Routine

The following activities take place when power is first applied to the keyboard.

Power-On Reset

The keyboard logic generates a 'power-on reset' signal (POR) when power is first applied to the keyboard. POR occurs a minimum of 150 milliseconds and a maximum of 2.0 seconds from the time power is first applied to the keyboard.

Basic Assurance Test

The basic assurance test (BAT) consists of a keyboard processor test, a checksum of the read-only memory (ROM), and a random-access memory (RAM) test. During the BAT, activity on the 'clock' and 'data' lines is ignored. The LEDs are turned on at the beginning and off at the end of the BAT. The BAT takes a minimum of 300 milliseconds and a maximum of 500 milliseconds. This is in addition to the time required by the POR.

Upon satisfactory completion of the BAT, a completion code (hex AA) is sent to the system, and keyboard scanning begins. If a BAT failure occurs, the keyboard sends an error code to the system. The keyboard is then disabled pending command input. Completion codes are sent between 450 milliseconds and 2.5 seconds after POR, and between 300 and 500 milliseconds after a Reset command is acknowledged.

Immediately following POR, the keyboard monitors the signals on the keyboard 'clock' and 'data' lines and sets the line protocol.

Commands from the System

The following table shows the commands that the system may send and their hexadecimal values.

Command	Hex Value
Set/Reset Status Indicators	ED
Echo	EE
Invalid Command	EF
Select Alternate Scan Codes	F0
Invalid Command	F1
Read ID	F2
Set Typematic Rate/Delay	F3
Enable	F4
Default Disable	F5
Set Default	F6
Set All Keys - Typematic	F7
- Make/Break	F8
- Make	F9
- Typematic/Make/Break	FA
Set Key Type - Typematic	FB
- Make/Break	FC
- Make	FD
Resend	FE
Reset	FF

The commands may be sent to the keyboard at any time. The keyboard will respond within 20 milliseconds, except when performing the basic assurance test (BAT), or executing a Reset command.

Note: Mode 1 will accept only the 'reset' command.

The commands are described below, in alphabetic order. They have different meanings when issued by the keyboard (see "Commands to the System" on page 4-13).

Default Disable (Hex F5)

The Default Disable command resets all conditions to the power-on default state. The keyboard responds with ACK, clears its output buffer, sets the default key types (scan code set 3 operation only) and typematic rate/delay, and clears the last typematic key. The keyboard stops scanning, and awaits further instructions.

Echo (Hex EE)

Echo is a diagnostic aid. When the keyboard receives this command, it issues a hex EE response and, if the keyboard was previously enabled, continues scanning.

Enable (Hex F4)

Upon receipt of this command, the keyboard responds with ACK, clears its output buffer, clears the last typematic key, and starts scanning.

Invalid Command (Hex EF and F1)

Hex EF and hex F1 are invalid commands and are not supported. If one of these is sent, the keyboard does not acknowledge the command, but returns a Resend command and continues in its prior scanning state. No other activities occur.

Read ID (Hex F2)

This command requests identification information from the keyboard. The keyboard responds with ACK, discontinues scanning, and sends the two keyboard ID bytes. The second byte must follow completion of the first by no more than 500 microseconds. After the output of the second ID byte, the keyboard resumes scanning.

Resend (Hex FE)

The system sends this command when it detects an error in any transmission from the keyboard. It is sent only after a keyboard transmission and before the system allows the next keyboard output. When a Resend is received, the keyboard sends the previous output again (unless the previous output was Resend, in which case the keyboard sends the last byte before the Resend command).

Reset (Hex FF)

The system issues a Reset command to start a program reset and a keyboard internal self test. The keyboard acknowledges the command with an ACK and ensures the system accepts ACK before executing the command. The system signals acceptance of ACK by raising the 'clock' and 'data' lines for a minimum of 500 microseconds. The keyboard is disabled from the time it receives the Reset command until ACK is accepted, or until another command is sent that overrides the previous command.

Following acceptance of ACK, the keyboard is re-initialized and performs the BAT. After returning the completion code, the keyboard defaults to scan code set 2.

Select Alternate Scan Codes (Hex F0)

This command instructs the keyboard to select one of three sets of scan codes. The keyboard acknowledges receipt of this command with ACK, clears both the output buffer and the typematic key (if one is active). The system then sends the option byte and the keyboard responds with another ACK. An option byte value of hex 01 selects scan code set 1, hex 02 selects set 2, and hex 03 selects set 3.

An option byte value of hex 00 causes the keyboard to acknowledge with ACK and send a byte telling the system which scan code set is currently in use.

After establishing the new scan code set, the keyboard returns to the scanning state it was in before receiving the Select Alternate Scan Codes command.

Set All Keys (Hex F7, F8, F9, FA)

These commands instruct the keyboard to set all keys to the type listed below:

Hex Value	Command
F7	Set All Keys - Typematic
F8	Set All Keys - Make/Break
F9	Set All Keys - Make
FA	Set All Keys - Typematic/Make/Break

The keyboard responds with ACK, clears its output buffer, sets all keys to the type indicated by the command, and continues scanning (if it was previously enabled). Although these commands can be sent using any scan code set, they affect only scan code set 3 operation.

Set Default (Hex F6)

The Set Default command resets all conditions to the power-on default state. The keyboard responds with ACK, clears its output buffer, sets the default key types (scan code set 3 operation only) and typematic rate/delay, clears the last typematic key, and continues scanning.

Set Key Type (Hex FB, FC, FD)

These commands instruct the keyboard to set individual keys to the type listed below:

Hex Value	Command
FB	Set Key Type - Typematic
FC	Set Key Type - Make/Break
FD	Set Key Type - Make

The keyboard responds with ACK, clears its output buffer, and prepares to receive key identification. Key identification is accomplished by the system identifying each key by its scan code value as defined in scan code set 3. Only scan code set 3 values are valid for key identification. The type of each identified key is set to the value indicated by the command.

These commands can be sent using any scan code set, but affect only scan code set 3 operation.

Set/Reset Status Indicators (Hex ED)

Three status indicators on the keyboard— Num Lock, Caps Lock, and Scroll Lock—are accessible by the system. The keyboard activates or deactivates these indicators when it receives a valid command-code sequence from the system. The command sequence begins with the command byte (hex ED). The keyboard responds to the command byte with ACK, discontinues scanning, and waits for the option byte from the system. The bit assignments for this option byte are as follows:

Bit	Indicator
0	Scroll Lock Indicator
1	Num Lock Indicator
2	Caps Lock Indicator
3-7	Reserved (must be 0s)

If a bit for an indicator is set to 1, the indicator is turned on. If a bit is set to 0, the indicator is turned off.

The keyboard responds to the option byte with ACK, sets the indicators and, if the keyboard was previously enabled, continues scanning. The state of the indicators will reflect the bits in the option byte and can be activated or deactivated in any combination. If another command is received in place of the option byte, execution of the Set/Reset Mode Indicators command is stopped, with no change to the indicator states, and the new command is processed.

Immediately after power-on, the lights default to the Off state. If the Set Default and Default Disable commands are received, the lamps remain in the state they were in before the command was received.

Set Typematic Rate/Delay (Hex F3)

The system issues the Set Typematic Rate/Delay command to change the typematic rate and delay. The keyboard responds to the command with ACK, stops scanning, and waits for the system to issue the rate/delay value byte. The keyboard responds to the rate/delay value byte with another ACK, sets the rate and delay to the values indicated, and continues scanning (if it was previously enabled). Bits 6 and 5 indicate the delay, and bits 4, 3, 2, 1, and 0 (the least-significant bit) the rate. Bit 7, the most-significant bit, is always 0. The delay is equal to 1 plus the binary value of bits 6 and 5, multiplied by 250 milliseconds ± 20%.

The period (interval from one typematic output to the next) is determined by the following equation:

Period = (8 + A) X (2^B) X 0.00417 seconds.
where:
A = binary value of bits 2, 1, and 0.
B = binary value of bits 4 and 3.

The typematic rate (make codes per second) is 1 for each period and are listed in the following table.

Bit	Typematic Rate ± 20%	Bit	Typematic Rate ± 20%
00000	30.0	10000	7.5
00001	26.7	10001	6.7
00010	24.0	10010	6.0
00011	21.8	10011	5.5
00100	20.0	10100	5.0
00101	18.5	10101	4.6
00110	17.1	10110	4.3
00111	16.0	10111	4.0
01000	15.0	11000	3.7
01001	13.3	11001	3.3
01010	12.0	11010	3.0
01011	10.9	11011	2.7
01100	10.0	11100	2.5
01101	9.2	11101	2.3
01110	8.0	11110	2.1
01111	8.0	11111	2.0

The default values for the system keyboard are as follows:

Typematic rate = 10.9 characters per second ± 20%.

Delay = 500 milliseconds ± 20%.

The execution of this command stops without change to the existing rate if another command is received instead of the rate/delay value byte.

Commands to the System

The following table shows the commands that the keyboard may send to the system, and their hexadecimal values.

Command	Hex Value
Key Detection Error/Overrun	00 (Code Sets 2 and 3)
Keyboard ID	83AB
BAT Completion Code	AA
BAT Failure Code	FC
Echo	EE
Acknowledge (ACK)	FA
Resend	FE
Key Detection Error/Overrun	FF (Code Set 1)

The commands the keyboard sends to the system are described below, in alphabetic order. They have different meanings when issued by the system (see "Commands from the System" on page 4-6).

Acknowledge (Hex FA)

The keyboard issues Acknowledge (ACK) to any valid input other than an Echo or Resend command. If the keyboard is interrupted while sending ACK, it discards ACK and accepts and responds to the new command.

BAT Completion Code (Hex AA)

Following satisfactory completion of the BAT, the keyboard sends hex AA. Any other code indicates a failure of the keyboard.

BAT Failure Code (Hex FC)

If a BAT failure occurs, the keyboard sends this code, discontinues scanning, and waits for a system response or reset.

Echo (Hex EE)

The keyboard sends this code in response to an Echo command.

Keyboard ID (Hex 83AB)

The Keyboard ID consists of 2 bytes, hex 83AB. The keyboard responds to the Read ID with ACK, discontinues scanning, and sends the 2 ID bytes. The low byte is sent first followed by the high byte. Following output of Keyboard ID, the keyboard begins scanning.

Key Detection Error (Hex 00 or FF)

The keyboard sends a key detection error character if conditions in the keyboard make it impossible to identify a switch closure. If the keyboard is using scan code set 1, the code is hex FF. For sets 2 and 3, the code is hex 00.

Overrun (Hex 00 or FF)

An overrun character is placed in the keyboard buffer and replaces the last code when the buffer capacity has been exceeded. The code is sent to the system when it reaches the top of the buffer queue. If the keyboard is using scan code set 1, the code is hex FF. For sets 2 and 3, the code is hex 00.

Resend (Hex FE)

The keyboard issues a Resend command following receipt of an invalid input or any input with incorrect parity. If the system sends nothing to the keyboard, no response is required.

Keyboard Scan Codes

The following tables list the key numbers of the three scan code sets and their hexadecimal values. The system defaults to scan set 2, but can be switched to set 1 or set 3 (see "Select Alternate Scan Codes (Hex F0)" on page 4-8).

Scan Code Set 1

In scan code set 1, each key is assigned a base scan code and, in some cases, extra codes to generate artificial shift states in the system. The typematic scan codes are identical to the base scan code for each key.

Scan Code Tables (Set 1)

The following keys send the codes as shown, regardless of any shift states in the keyboard or the system. Refer to "Keyboard Layouts" beginning on page 4-40 to determine the character associated with each key number.

Key Number	Make Code	Break Code
1	29	A9
2	02	82
3	03	83
4	04	84
5	05	85
6	06	86
7	07	87
8	08	88
9	09	89
10	0A	8A
11	0B	8B
12	0C	8C
13	0D	8D
15	0E	8E
16	0F	8F
17	10	90
18	11	91
19	12	92
20	13	93
21	14	94
22	15	95
23	16	96
24	17	97
25	18	98
26	19	99
27	1A	9A
28	1B	9B
29 *	2B	AB
30	3A	BA
31	1E	9E
32	1F	9F
33	20	A0

* 101-key keyboard only.

Key Number	Make Code	Break Code
34	21	A1
35	22	A2
36	23	A3
37	24	A4
38	25	A5
39	26	A6
40	27	A7
41	28	A8
42 **	2B	AB
43	1C	9C
44	2A	AA
45 **	56	D6
46	2C	AC
47	2D	AD
48	2E	AE
49	2F	AF
50	30	B0
51	31	B1
52	32	B2
53	33	B3
54	34	B4
55	35	B5
57	36	B6
58	1D	9D
60	38	B8
61	39	B9
62	E0 38	E0 B8
64	E0 1D	E0 9D
90	45	C5
91	47	C7
92	4B	CB
93	4F	CF
96	48	C8
97	4C	CC
98	50	D0
99	52	D2
100	37	B7
101	49	C9
102	4D	CD
103	51	D1
104	53	D3
105	4A	CA
106	4E	CE
108	E0 1C	E0 9C
110	01	81
112	3B	BB
113	3C	BC
114	3D	BD
115	3E	BE
116	3F	BF
117	40	C0
118	41	C1
119	42	C2

** 102-key keyboard only.

Key Number	Make Code	Break Code
120	43	C3
121	44	C4
122	57	D7
123	58	D8
125	46	C6

The remaining keys send a series of codes dependent on the state of the various shift keys (Ctrl, Alt, and Shift), and the state of Num Lock (On or Off). Because the base scan code is identical to that of another key, an extra code (hex E0) has been added to the base code to make it unique.

Key No.	Base Case, or Shift+Num Lock Make/Break	Shift Case Make/Break *	Num Lock on Make/Break
75	E0 52 /E0 D2	E0 AA E0 52 /E0 D2 E0 2A	E0 2A E0 52 /E0 D2 E0 AA
76	E0 53 /E0 D3	E0 AA E0 53 /E0 D3 E0 2A	E0 2A E0 53 /E0 D3 E0 AA
79	E0 4B /E0 CB	E0 AA E0 4B /E0 CB E0 2A	E0 2A E0 4B /E0 CB E0 AA
80	E0 47 /E0 C7	E0 AA E0 47 /E0 C7 E0 2A	E0 2A E0 47 /E0 C7 E0 AA
81	E0 4F /E0 CF	E0 AA E0 4F /E0 CF E0 2A	E0 2A E0 4F /E0 CF E0 AA
83	E0 48 /E0 C8	E0 AA E0 48 /E0 C8 E0 2A	E0 2A E0 48 /E0 C8 E0 AA
84	E0 50 /E0 D0	E0 AA E0 50 /E0 D0 E0 2A	E0 2A E0 50 /E0 D0 E0 AA
85	E0 49 /E0 C9	E0 AA E0 49 /E0 C9 E0 2A	E0 2A E0 49 /E0 C9 E0 AA
86	E0 51 /E0 D1	E0 AA E0 51 /E0 D1 E0 2A	E0 2A E0 51 /E0 D1 E0 AA
89	E0 4D /E0 CD	E0 AA E0 4D /E0 CD E0 2A	E0 2A E0 4D /E0 CD E0 AA

* If the left Shift key is held down, the AA/2A shift make and break is sent with the other scan codes. If the right Shift key is held down, B6/36 is sent. If both Shift keys are down, both sets of codes are sent with the other scan code.

Key No.	Scan Code Make/Break	Shift Case Make/Break *
95	E0 35/E0 B5	E0 AA E0 35/E0 B5 E0 2A
* If the left Shift key is held down, the AA/2A shift make and break is sent with the other scan codes. If the right Shift key is held down, B6/36 is sent. If both Shift keys are down, both sets of codes are sent with the other scan code.		

Key No.	Scan Code Make/Break	Ctrl Case, Shift Case Make/Break	Alt Case Make/Break
124	E0 2A E0 37 /E0 B7 E0 AA	E0 37/E0 B7	54/D4

Key No.	Make Code	Ctrl Key Pressed
126 *	E1 1D 45 E1 9D C5	E0 46 E0 C6
* This key is not typematic. All associated scan codes occur on the make of the key.		

Scan Code Set 2

In scan code set 2, each key is assigned a unique 8-bit make scan code, which is sent when the key is pressed. Each key also sends a break code when the key is released. The break code consists of 2 bytes, the first of which is the break code prefix, hex F0; the second byte is the same as the make scan code for that key. The typematic scan code for a key is the same as the key's make code.

Scan Code Tables (Set 2)

The following keys send the codes shown, regardless of any shift states in the keyboard or system. Refer to "Keyboard Layouts" beginning on page 4-40 to determine the character associated with each key number.

Key Number	Make Code	Break Code
1	0E	F0 0E
2	16	F0 16
3	1E	F0 1E
4	26	F0 26
5	25	F0 25
6	2E	F0 2E
7	36	F0 36
8	3D	F0 3D
9	3E	F0 3E
10	46	F0 46
11	45	F0 45
12	4E	F0 4E
13	55	F0 55
15	66	F0 66
16	0D	F0 0D
17	15	F0 15
18	1D	F0 1D
19	24	F0 24
20	2D	F0 2D
21	2C	F0 2C
22	35	F0 35
23	3C	F0 3C
24	43	F0 43
25	44	F0 44
26	4D	F0 4D
27	54	F0 54
28	5B	F0 5B
29 *	5D	F0 5D
30	58	F0 58
31	1C	F0 1C

* 101-key keyboard only.

Key Number	Make Code	Break Code
32	1B	F0 1B
33	23	F0 23
34	2B	F0 2B
35	34	F0 34
36	33	F0 33
37	3B	F0 3B
38	42	F0 42
39	4B	F0 4B
40	4C	F0 4C
41	52	F0 52
42 **	5D	F0 5D
43	5A	F0 5A
44	12	F0 12
45 **	61	F0 61
46	1A	F0 1A
47	22	F0 22
48	21	F0 21
49	2A	F0 2A
50	32	F0 32
51	31	F0 31
52	3A	F0 3A
53	41	F0 41
54	49	F0 49
55	4A	F0 4A
57	59	F0 59
58	14	F0 14
60	11	F0 11
61	29	F0 29
62	E0 11	E0 F0 11
64	E0 14	E0 F0 14
90	77	F0 77
91	6C	F0 6C
92	6B	F0 6B
93	69	F0 69
96	75	F0 75
97	73	F0 73
98	72	F0 72
99	70	F0 70
100	7C	F0 7C
101	7D	F0 7D
102	74	F0 74
103	7A	F0 7A
104	71	F0 71
105	7B	F0 7B
106	79	F0 79
108	E0 5A	E0 F0 5A
110	76	F0 76
112	05	F0 05
113	06	F0 06
114	04	F0 04
115	0C	F0 0C
116	03	F0 03
117	0B	F0 0B
118	83	F0 83
119	0A	F0 0A

** 102-key keyboard only.

Key Number	Make Code	Break Code
120	01	F0 01
121	09	F0 09
122	78	F0 78
123	07	F0 07
125	7E	F0 7E

The remaining keys send a series of codes dependent on the state of the various shift keys (Ctrl, Alt, and Shift), and the state of Num Lock (On or Off). Because the base scan code is identical to that of another key, an extra code (hex E0) has been added to the base code to make it unique.

Key No.	Base Case, or Shift+Num Lock Make/Break	Shift Case Make/Break *	Num Lock on Make/Break
75	E0 70 /E0 F0 70	E0 F0 12 E0 70 /E0 F0 70 E0 12	E0 12 E0 70 /E0 F0 70 E0 F0 12
76	E0 71 /E0 F0 71	E0 F0 12 E0 71 /E0 F0 71 E0 12	E0 12 E0 71 /E0 F0 71 E0 F0 12
79	E0 6B /E0 F0 6B	E0 F0 12 E0 6B /E0 F0 6B E0 12	E0 12 E0 6B /E0 F0 6B E0 F0 12
80	E0 6C /E0 F0 6C	E0 F0 12 E0 6C /E0 F0 6C E0 12	E0 12 E0 6C /E0 F0 6C E0 F0 12
81	E0 69 /E0 F0 69	E0 F0 12 E0 69 /E0 F0 69 E0 12	E0 12 E0 69 /E0 F0 69 E0 F0 12
83	E0 75 /E0 F0 75	E0 F0 12 E0 75 /E0 F0 75 E0 12	E0 12 E0 75 /E0 F0 75 E0 F0 12
84	E0 72 /E0 F0 72	E0 F0 12 E0 72 /E0 F0 72 E0 12	E0 12 E0 72 /E0 F0 72 E0 F0 12
85	E0 7D /E0 F0 7D	E0 F0 12 E0 7D /E0 F0 7D E0 12	E0 12 E0 7D /E0 F0 7D E0 F0 12
86	E0 7A /E0 F0 7A	E0 F0 12 E0 7A /E0 F0 7A E0 12	E0 12 E0 7A /E0 F0 7A E0 F0 12
89	E0 74 /E0 F0 74	E0 F0 12 E0 74 /E0 F0 74 E0 12	E0 12 E0 74 /E0 F0 74 E0 F0 12

* If the left Shift key is held down, the F0 12/12 shift make and break is sent with the other scan codes. If the right Shift key is held down, F0 59/59 is sent. If both Shift keys are down, both sets of codes are sent with the other scan code.

Key No.	Scan Code Make/Break	Shift Case Make/Break *
95	E0 4A/E0 F0 4A	E0 F0 12 4A/E0 12 F0 4A
* If the left Shift key is held down, the F0 12/12 shift make and break is sent with the other scan codes. If the right Shift key is held down, F0 59/59 is sent. If both Shift keys are down, both sets of codes are sent with the other scan code.		

Key No.	Scan Code Make/Break	Ctrl Case, Shift Case Make/Break	Alt Case Make/Break
124	E0 12 E0 7C /E0 F0 7C E0 F0 12	E0 7C/E0 F0 7C	84/F0 84

Key No.	Make Code	Ctrl Key Pressed
126 *	E1 14 77 E1 F0 14 F0 77	E0 7E E0 F0 7E
* This key is not typematic. All associated scan codes occur on the make of the key.		

Scan Code Set 3

In scan code set 3, each key is assigned a unique 8-bit make scan code, which is sent when the key is pressed. Each key also sends a break code when the key is released. The break code consists of 2 bytes, the first of which is the break-code prefix, hex F0; the second byte is the same as the make scan code for that key. The typematic scan code for a key is the same as the key's make code. With this scan code set, each key sends only one scan code, and no keys are affected by the state of any other keys.

Scan Code Tables (Set 3)

The following keys send the codes shown, regardless of any shift states in the keyboard or system. Refer to "Keyboard Layouts" beginning on page 4-40 to determine the character associated with each key number.

Key Number	Make Code	Break Code	Default Key State
1	0E	F0 0E	Typematic
2	16	F0 16	Typematic
3	1E	F0 1E	Typematic
4	26	F0 26	Typematic
5	25	F0 25	Typematic
6	2E	F0 2E	Typematic
7	36	F0 36	Typematic
8	3D	F0 3D	Typematic
9	3E	F0 3E	Typematic
10	46	F0 46	Typematic
11	45	F0 45	Typematic
12	4E	F0 4E	Typematic
13	55	F0 55	Typematic
15	66	F0 66	Typematic
16	0D	F0 0D	Typematic
17	15	F0 15	Typematic
18	1D	F0 1D	Typematic
19	24	F0 24	Typematic
20	2D	F0 2D	Typematic
21	2C	F0 2C	Typematic
22	35	F0 35	Typematic
23	3C	F0 3C	Typematic
24	43	F0 43	Typematic
25	44	F0 44	Typematic
26	4D	F0 4D	Typematic
27	54	F0 54	Typematic
28	5B	F0 5B	Typematic

Key Number	Make Code	Break Code	Default Key State
29 *	5C	F0 5C	Typematic
30	14	F0 14	Make/Break
31	1C	F0 1C	Typematic
32	1B	F0 1B	Typematic
33	23	F0 23	Typematic
34	2B	F0 2B	Typematic
35	34	F0 34	Typematic
36	33	F0 33	Typematic
37	3B	F0 3B	Typematic
38	42	F0 42	Typematic
39	4B	F0 4B	Typematic
40	4C	F0 4C	Typematic
41	52	F0 52	Typematic
42 **	53	F0 53	Typematic
43	5A	F0 5A	Typematic
44	12	F0 12	Make/Break
45 **	13	F0 13	Typematic
46	1A	F0 1A	Typematic
47	22	F0 22	Typematic
48	21	F0 21	Typematic
49	2A	F0 2A	Typematic
50	32	F0 32	Typematic
51	31	F0 31	Typematic
52	3A	F0 3A	Typematic
53	41	F0 41	Typematic
54	49	F0 49	Typematic
55	4A	F0 4A	Typematic
57	59	F0 59	Make/Break
58	11	F0 11	Make/Break
60	19	F0 19	Make/Break
61	29	F0 29	Typematic
62	39	F0 39	Make only
64	58	F0 58	Make only
75	67	F0 67	Make only
76	64	F0 64	Typematic
79	61	F0 61	Typematic
80	6E	F0 6E	Make only
81	65	F0 65	Make only
83	63	F0 63	Typematic
84	60	F0 60	Typematic
85	6F	F0 6F	Make only
86	6D	F0 6D	Make only
89	6A	F0 6A	Typematic
90	76	F0 76	Make only
91	6C	F0 6C	Make only
92	6B	F0 6B	Make only
93	69	F0 69	Make only
95	77	F0 77	Make only
96	75	F0 75	Make only
97	73	F0 73	Make only
98	72	F0 72	Make only

* 101-key keyboard only.
** 102-key keyboard only.

Key Number	Make Code	Break Code	Default Key State
99	70	F0 70	Make only
100	7E	F0 7E	Make only
101	7D	F0 7D	Make only
102	74	F0 74	Make only
103	7A	F0 7A	Make only
104	71	F0 71	Make only
105	84	F0 84	Make only
106	7C	F0 7C	Typematic
108	79	F0 79	Make only
110	08	F0 08	Make only
112	07	F0 07	Make only
113	0F	F0 0F	Make only
114	17	F0 17	Make only
115	1F	F0 1F	Make only
116	27	F0 27	Make only
117	2F	F0 2F	Make only
118	37	F0 37	Make only
119	3F	F0 3F	Make only
120	47	F0 47	Make only
121	4F	F0 4F	Make only
122	56	F0 56	Make only
123	5E	F0 5E	Make only
124	57	F0 57	Make only
125	5F	F0 5F	Make only
126	62	F0 62	Make only

Clock and Data Signals

The keyboard and system communicate over the 'clock' and 'data' lines. The source of each of these lines is an open-collector device on the keyboard that allows either the keyboard or the system to force a line to an inactive (low) level. When no communication is occurring, the 'clock' line is at an active (high) level. The state of the 'data' line is held active(high) by the keyboard.

When the system sends data to the keyboard, it forces the 'data' line to an inactive level and allows the 'clock' line to go to an active level.

An inactive signal will have a value of at least 0, but not greater than +0.7 volts. A signal at the inactive level is a logical 0. An active signal will have a value of at least +2.4, but not greater than +5.5 volts. A signal at the active level is a logical 1. Voltages are measured between a signal source and the dc network ground.

The keyboard 'clock' line provides the clocking signals used to clock serial data to and from the keyboard. If the host system forces the 'clock' line to an inactive level, keyboard transmission is inhibited.

When the keyboard sends data to, or receives data from the system, it generates the 'clock' signal to time the data. The system can prevent the keyboard from sending data by forcing the 'clock' line to an inactive level; the 'data' line may be active or inactive during this time.

During the BAT, the keyboard allows the 'clock' and 'data' lines to go to an active level.

Data Stream

Data transmissions to and from the keyboard consist of an 11-bit data stream (Mode 2) sent serially over the 'data' line. A logical 1 is sent at an active (high) level. The following table shows the functions of the bits.

Bit	Function
1	Start bit (always 0)
2	Data bit 0 (least-significant)
3	Data bit 1
4	Data bit 2
5	Data bit 3
6	Data bit 4
7	Data bit 5
8	Data bit 6
9	Data bit 7 (most-significant)
10	Parity bit (odd parity)
11	Stop bit (always 1)

The parity bit is either 1 or 0, and the 8 data bits, plus the parity bit, always have an odd number of 1's.

Note: Mode 1 is a 9-bit data stream that does not have a parity bit or stop bit and the start bit is always 1.

Keyboard Data Output

When the keyboard is ready to send data, it first checks for a keyboard-inhibit or system request-to-send status on the 'clock' and 'data' lines. If the 'clock' line is inactive (low), data is stored in the keyboard buffer. If the 'clock' line is active (high) and the 'data' line is inactive (request-to-send), data is stored in the keyboard buffer, and the keyboard receives system data.

If the 'clock' and 'data' lines are both active, the keyboard sends the 0 start bit, 8 data bits, the parity bit, and the stop bit. Data will be valid before the trailing edge and beyond the leading edge of the clock pulse. During transmission, the keyboard checks the 'clock' line for an active level at least every 60 milliseconds. If the system lowers the 'clock' line from an active level after the keyboard starts sending data, a condition known as *line contention* occurs, and the keyboard stops sending data. If line contention occurs before the leading edge of the 10th clock signal (parity bit), the keyboard buffer returns the 'clock' and 'data' lines to an active level. If contention does not occur by the 10th clock signal, the keyboard completes the transmission. Following line contention, the system may or may not request the keyboard to resend the data.

Following a transmission, the system can inhibit the keyboard until the system processes the input, or until it requests that a response be sent.

Keyboard Data Input

When the system is ready to send data to the keyboard, it first checks to see if the keyboard is sending data. If the keyboard is sending, but has not reached the 10th 'clock' signal, the system can override the keyboard output by forcing the keyboard 'clock' line to an inactive (low) level. If the keyboard transmission is beyond the 10th 'clock' signal, the system must receive the transmission.

If the keyboard is not sending, or if the system elects to override the keyboard's output, the system forces the keyboard 'clock' line to an inactive level for more than 60 microseconds while preparing to send data. When the system is ready to send the start bit (the 'data' line will be inactive), it allows the 'clock' line to go to an active (high) level.

The keyboard checks the state of the 'clock' line at intervals of no more than 10 milliseconds. If a system request-to-send (RTS) is detected, the keyboard counts 11 bits. After the 10th bit, the keyboard checks for an active level on the 'data' line, and if the line is active, forces it inactive, and counts one more bit. This action signals the system that the keyboard has received its data. Upon receipt of this signal, the system returns to a ready state, in which it can accept keyboard output, or goes to the inhibited state until it is ready.

If the keyboard 'data' line is found at an inactive level following the 10th bit, a framing error has occurred, and the keyboard continues to count until the 'data' line becomes active. The keyboard then makes the 'data' line inactive and sends a Resend.

Each system command or data transmission to the keyboard requires a response from the keyboard before the system can send its next output. The keyboard will respond within 20 milliseconds unless the system prevents keyboard output. If the keyboard response is invalid or has a parity error, the system sends the command or data again. However, the two byte commands require special handling. If hex F3 (Set Typematic Rate/Delay),

hex F0 (Select Alternate Scan Codes), or hex ED (Set/Reset Mode Indicators) have been sent and acknowledged, and the value byte has been sent but the response is invalid or has a parity error, the system will resend both the command and the value byte.

Keyboard Encoding and Usage

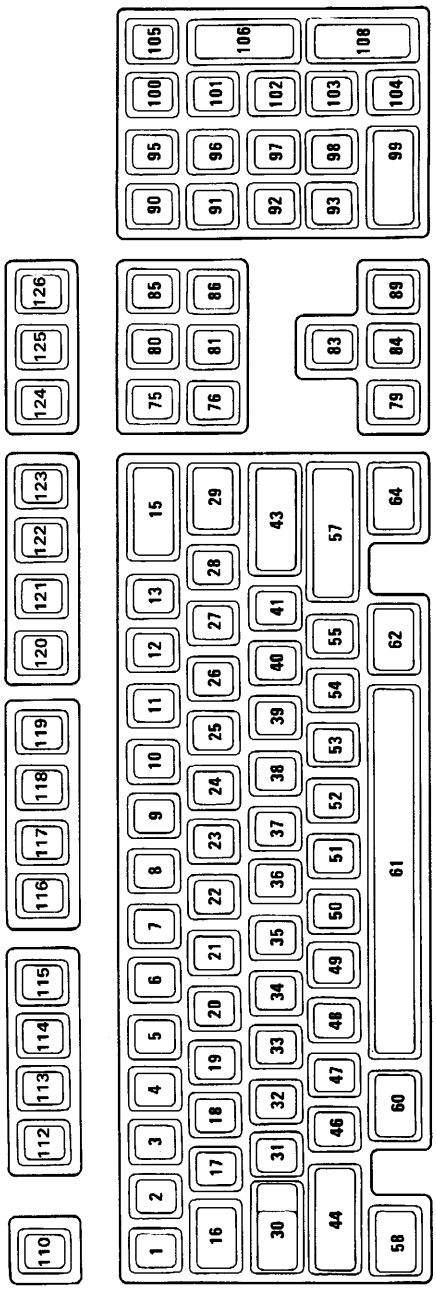
The keyboard routine, provided by IBM in the ROM BIOS, is responsible for converting the keyboard scan codes into what will be termed *Extended ASCII*. The extended ASCII codes returned by the ROM routine are mapped to the U.S. English keyboard layout. Some operating systems may make provisions for alternate keyboard layouts by providing an interrupt replacer, which resides in the read/write memory. This section discusses only the ROM routine.

Extended ASCII encompasses 1-byte character codes, with possible values of 0 to 255, an extended code for certain extended keyboard functions, and functions handled within the keyboard routine or through interrupts.

Character Codes

The character codes described later are passed through the BIOS keyboard routine to the system or application program. A "-1" means the combination is suppressed in the keyboard routine. The codes are returned in the AL register. See "Characters, Keystrokes, and Color" later in this manual for the exact codes.

The following figure shows the keyboard layout and key positions.



Key	Base Case	Uppercase	Ctrl	Alt
1	`	~	-1	(*)
2	1	!	-1	(*)
3	2	@	Nul(000) (*)	(*)
4	3	#	-1	(*)
5	4	$	-1	(*)
6	5	%	-1	(*)
7	6	^	RS(030)	(*)
8	7	&	-1	(*)
9	8	*	-1	(*)
10	9	(	-1	(*)
11	0	)	-1	(*)
12	-	_	US(031)	(*)
13	=	+	-1	(*)
15	Backspace (008)	Backspace (008)	Del(127)	(*)
16	→| (009)	|← (*)	(*)	(*)
17	q	Q	DC1(017)	(*)
18	w	W	ETB(023)	(*)
19	e	E	ENQ(005)	(*)
20	r	R	DC2(018)	(*)
21	t	T	DC4(020)	(*)
22	y	Y	EM(025)	(*)
23	u	U	NAK(021)	(*)
24	i	I	HT(009)	(*)
25	o	O	SI(015)	(*)
26	p	P	DLE(016)	(*)
27	[	{	Esc(027)	(*)
28	]	}	GS(029)	(*)
29	\	|	FS(028)	(*)
30 Caps Lock	-1	-1	-1	-1
31	a	A	SOH(001)	(*)
32	s	S	DC3(019)	(*)
33	d	D	EOT(004)	(*)
34	f	F	ACK(006)	(*)
35	g	G	BEL(007)	(*)
36	h	H	BS(008)	(*)
37	j	J	LF(010)	(*)
38	k	K	VT(011)	(*)
39	l	L	FF(012)	(*)
40	; '	: "	-1	(*)
41			-1	(*)
43	CR(013)	CR(013)	LF(010)	(*)
44 Shift (Left)	-1	-1	-1	-1
46	z	Z	SUB(026)	(*)
47	x	X	CAN(024)	(*)
48	c	C	ETX(003)	(*)

Notes:
(*) Refer to "Extended Functions" in this section.
(**) Refer to "Special Handling" in this section.

Character Codes (Part 1 of 2)

Key	Base Case	Uppercase	Ctrl	Alt
49	v	V	SYN(022)	(*)
50	b	B	STX(002)	(*)
51	n	N	SO(014)	(*)
52	m	M	CR(013)	(*)
53	,	<	-1	(*)
54	.	>	-1	(*)
55	/	?	-1	(*)
57 Shift (Right)	-1	-1	-1	-1
58 Ctrl (Left)	-1	-1	-1	-1
60 Alt (Left)	-1	-1	-1	-1
61	Space	Space	Space	Space
62 Alt (Right)	-1	-1	-1	-1
64 Ctrl (Right)	-1	-1	-1	-1
90 Num Lock	-1	-1	-1	-1
95	/	/	(*)	(*)
100	*	*	(*)	(*)
105	-	-	(*)	(*)
106	+	+	(*)	(*)
108	Enter	Enter	LF(010)	(*)
110	Esc	Esc	Esc	(*)
112	Null (*)	Null (*)	Null (*)	Null(*)
113	Null (*)	Null (*)	Null (*)	Null(*)
114	Null (*)	Null (*)	Null (*)	Null(*)
115	Null (*)	Null (*)	Null (*)	Null(*)
116	Null (*)	Null (*)	Null (*)	Null(*)
117	Null (*)	Null (*)	Null (*)	Null(*)
118	Null (*)	Null (*)	Null (*)	Null(*)
119	Null (*)	Null (*)	Null (*)	Null(*)
120	Null (*)	Null (*)	Null (*)	Null(*)
121	Null (*)	Null (*)	Null (*)	Null(*)
122	Null (*)	Null (*)	Null (*)	Null(*)
123	Null (*)	Null (*)	Null (*)	Null(*)
125 Scroll Lock	-1	-1	-1	-1
126	Pause(**)	Pause(**)	Break(**)	Pause(**)

Notes:
(*) Refer to "Extended Functions" in this section.
(**) Refer to "Special Handling" in this section.

Character Codes (Part 2 of 2)

The following table lists keys that have meaning only in Num Lock, Shift, or Ctrl states. The Shift key temporarily reverses the current Num Lock state.

Key	Num Lock	Base Case	Alt	Ctrl
91	7	Home (*)	-1	Clear Screen
92	4	← (*)	-1	Reverse Word(*)
93	1	End (*)	-1	Erase to EOL(*)
96	8	↑ (*)	-1	(*)
97	5	(*)	-1	(*)
98	2	↓ (*)	-1	(*)
99	0	Ins	-1	(*)
101	9	Page Up (*)	-1	Top of Text and Home
102	6	→ (*)	-1	Advance Word (*)
103	3	Page Down (*)	-1	Erase to EOS (*)
104	.	Delete (*,**)	(**)	(**)

Notes:
(*) Refer to "Extended Functions" in this section.
(**) Refer to "Special Handling" in this section.

Special Character Codes

Extended Functions

For certain functions that cannot be represented by a standard ASCII code, an extended code is used. A character code of 000 (null) is returned in AL. This indicates that the system or application program should examine a second code, which will indicate the actual function. Usually, but not always, this second code is the scan code of the primary key that was pressed. This code is returned in AH.

The following table is a list of the extended codes and their functions.

Second Code	Function
1	Alt Esc
3	Nul Character
14	Alt Backspace
15	⇤ (Back-tab)
16-25	Alt Q, W, E, R, T, Y, U, I, O, P
26-28	Alt [ ] ↵
30-38	Alt A, S, D, F, G, H, J, K, L
39-41	Alt ;
43	Alt \
44-50	Alt Z, X, C, V, B, N, M
51-53	Alt , . /
55	Alt Keypad *
59-68	F1 to F10 Function Keys (Base Case)
71	Home
72	↑ (Cursor Up)
73	Page Up
74	Alt Keypad -
75	← (Cursor Left)
76	Center Cursor
77	→ (Cursor Right)
78	Alt Keypad +
79	End
80	↓ (Cursor Down)
81	Page Down
82	Ins (Insert)
83	Del (Delete)
84-93	Shift F1 to F10
94-103	Ctrl F1 to F10
104-113	Alt F1 to F10
114	Ctrl PrtSc (Start/Stop Echo to Printer)
115	Ctrl ← (Reverse Word)
116	Ctrl → (Advance Word)
117	Ctrl End (Erase to End of Line-EOL)
118	Ctrl PgDn (Erase to End of Screen-EOS)
119	Ctrl Home (Clear Screen and Home)
120-131	Alt 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, -, = keys 2-13
132	Ctrl PgUp (Top 25 Lines of Text and Cursor Home)
133-134	F11, F12
135-136	Shift F11, F12
137-138	Ctrl F11, F12
139-140	Alt F11, F12
141	Ctrl Up/8
142	Ctrl Keypad -
143	Ctrl Keypad 5
144	Ctrl Keypad +
145	Ctrl Down/2
146	Ctrl Ins/0
147	Ctrl Del/.
148	Ctrl Tab
149	Ctrl Keypad /
150	Ctrl Keypad *

Keyboard Extended Functions (Part 1 of 2)

Second Code	Function
151	Alt Home
152	Alt Up
153	Alt Page Up
155	Alt Left
157	Alt Right
159	Alt End
160	Alt Down
161	Alt Page Down
162	Alt Insert
163	Alt Delete
164	Alt Keypad /
165	Alt Tab
166	Alt Enter

Keyboard Extended Functions (Part 2 of 2)

Shift States

Most shift states are handled within the keyboard routine, and are not apparent to the system or application program. In any case, the current status of active shift states is available by calling an entry point in the BIOS keyboard routine. The following keys result in altered shift states:

Shift: This key temporarily shifts keys 1 through 13, 16 through 29, 31 through 41, and 46 through 55, to uppercase (base case if in Caps Lock state). Also, the Shift temporarily reverses the Num Lock or non-Num Lock state of keys 91 through 93, 96, 98, 99, and 101 through 104.

Ctrl: This key temporarily shifts keys 3, 7, 12, 15 through 29, 31 through 39, 43, 46 through 52, 75 through 89, 91 through 93, 95 through 108, 112 through 124 and 126 to the Ctrl state. The Ctrl key is also used with the Alt and Del keys to cause the system-reset function; with the Scroll Lock key to cause the break function; and with the Num Lock key to cause the pause function. The system-reset, break, and pause functions are described under "Special Handling" later in this section.

Alt: This key temporarily shifts keys 1 through 29, 31 through 43, 46 through 55, 75 through 89, 95, 100, and 105 through 124 to the Alt state. The Alt key is also used with the Ctrl and Del keys to cause a system reset.

The Alt key also allows the user to enter any character code from 1 to 255. The user holds down the Alt key and types the decimal value of the characters desired on the numeric keypad (keys 91 through 93, 96 through 99, and 101 through 103). The Alt key is then released. If the number is greater than 255, a modulo-256 value is used. This value is interpreted as a character code and is sent through the keyboard routine to the system or application program. Alt is handled internal to the keyboard routine.

Caps Lock: This key shifts keys 17 through 26, 31 through 39, and 46 through 52 to uppercase. When Caps Lock is pressed again, it reverses the action. Caps Lock is handled internal to the keyboard routine. When Caps Lock is pressed, it changes the Caps Lock Mode indicator. If the indicator was on, it will go off; and if it was off, it will go on.

Scroll Lock: When interpreted by appropriate application programs, this key indicates that the cursor-control keys will cause windowing over the text rather than moving the cursor. When the Scroll Lock key is pressed again, it reverses the action. The keyboard routine simply records the current shift state of the Scroll Lock key. It is the responsibility of the application program to perform the function. When Scroll Lock is pressed, it changes the Scroll Lock Mode indicator. If the indicator was on, it will go off; and if it was off, it will go on.

Num Lock: This key shifts keys 91 through 93, 96 through 99, and 101 through 104 to uppercase. When Num Lock is pressed again, it reverses the action. Num Lock is handled internal to the keyboard routine. When Num Lock is pressed, it changes the Num Lock Mode indicator. If the indicator was on, it will go off; if it was off, it will go on.

Shift Key Priorities and Combinations: If combinations of the Alt, Ctrl, and Shift keys are pressed and only one is valid, the priority is as follows: the Alt key is first, the Ctrl key is second, and the Shift key is third. The only valid combination is Alt and Ctrl, which is used in the system-reset function.

Special Handling

System Reset

The combination of any Alt, Ctrl, and Del keys results in the keyboard routine that starts a system reset or restart. System reset is handled by BIOS.

Break

The combination of the Ctrl and Pause/Break keys results in the keyboard routine signaling interrupt hex 1B. The extended characters AL=hex 00, and AH=hex 00 are also returned.

Pause

The Pause key causes the keyboard interrupt routine to loop, waiting for any character or function key to be pressed. This provides a method of temporarily suspending an operation, such as listing or printing, and then resuming the operation. The method is not apparent to either the system or the application program. The key stroke used to resume operation is discarded. Pause is handled internal to the keyboard routine.

Print Screen

The Print Screen key results in an interrupt invoking the print-screen routine. This routine works in the alphameric or graphics mode, with unrecognizable characters printing as blanks.

System Request

When the System Request (Alt and Print Screen) key is pressed, a hex 8500 is placed in AX, and an interrupt hex 15 is executed. When the SysRq key is released, a hex 8501 is placed in AX, and another interrupt hex 15 is executed. If an application is to use System Request, the following rules must be observed:

Save the previous address.

Overlay interrupt vector hex 15.

Check AH for a value of hex 85:

If yes, process may begin.
If no, go to previous address.

The application program must preserve the value in all registers, except AX, upon return. System Request is handled internal to the keyboard routine.

Other Characteristics

The keyboard routine does its own buffering, and the keyboard buffer is large enough to support entries by a fast typist. However, if a key is pressed when the buffer is full, the key will be ignored and the "alarm" will sound.

The keyboard routine also suppresses the typematic action of the following keys: Ctrl, Shift, Alt, Num Lock, Scroll Lock, Caps Lock, and Ins.

During each interrupt hex 09 from the keyboard, an interrupt hex 15, function (AH)=hex 4F is generated by the BIOS after the scan code is read from the keyboard adapter. The scan code is passed in the (AL) register with the carry flag set. This is to allow an operating system to intercept each scan code prior to its being handled by the interrupt hex 09 routine, and have a chance to change or act on the scan code. If the carry flag is changed to 0 on return from interrupt hex 15, the scan code will be ignored by the interrupt handler.

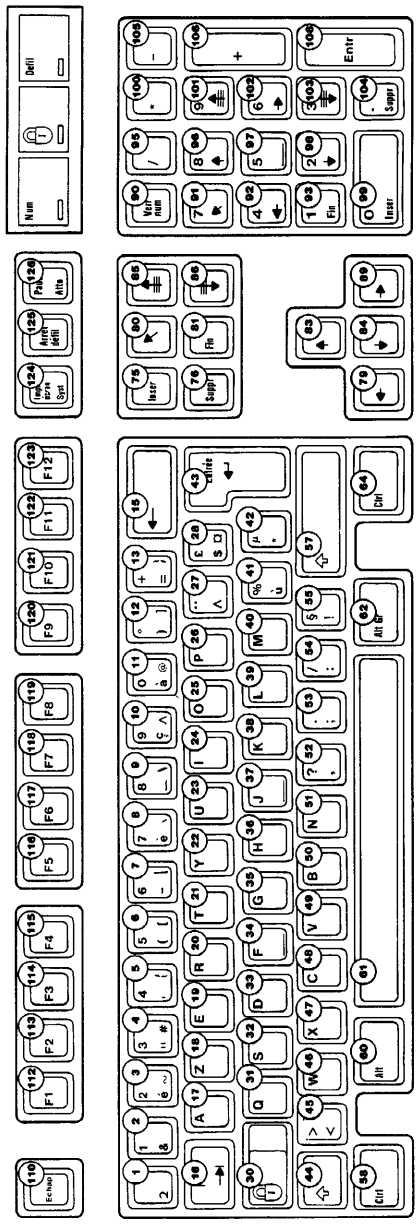
Keyboard Layouts

The keyboard is available in six layouts:

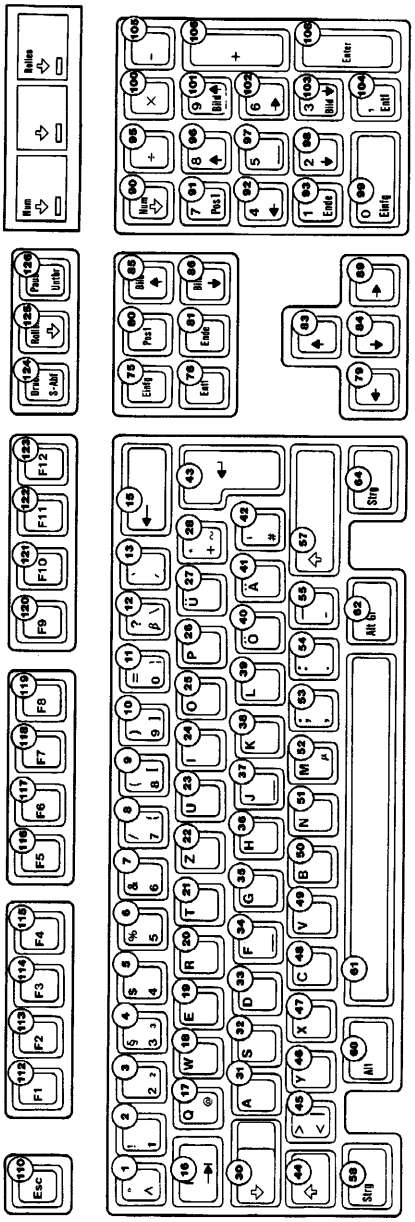
- French
- German
- Italian
- Spanish
- U.K. English
- U.S. English

The various layouts are shown in alphabetic order on the following pages. Nomenclature is on both the top and front face of the keybuttons. The number to the upper right designates the keybutton position.

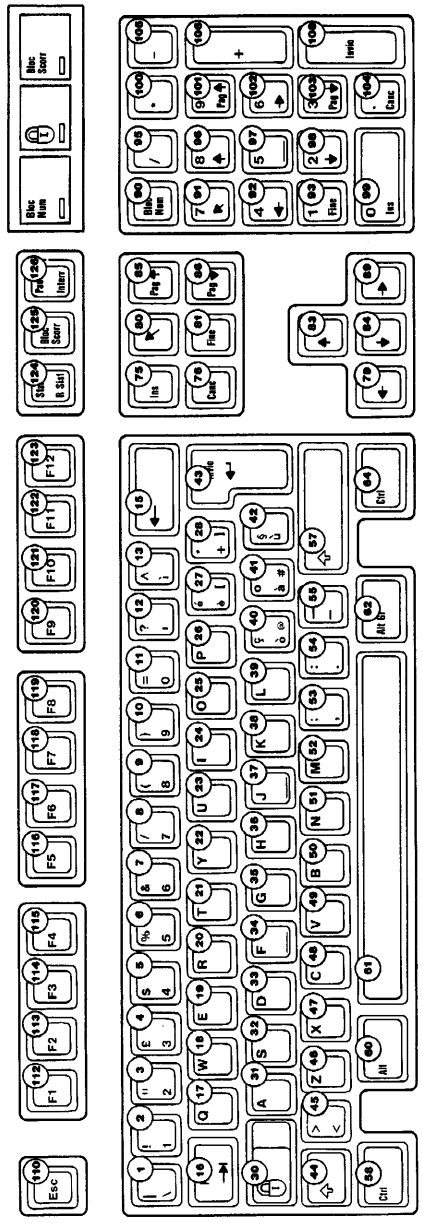
French Keyboard



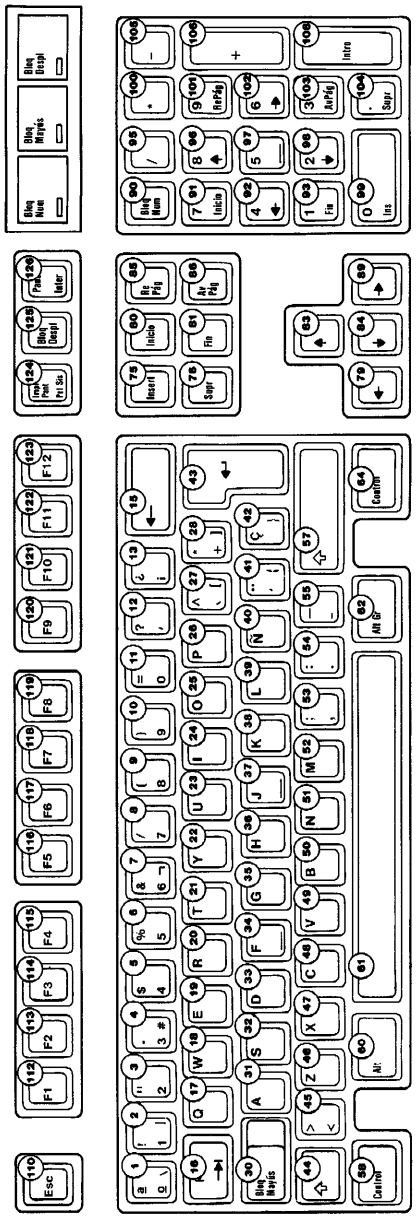
German Keyboard



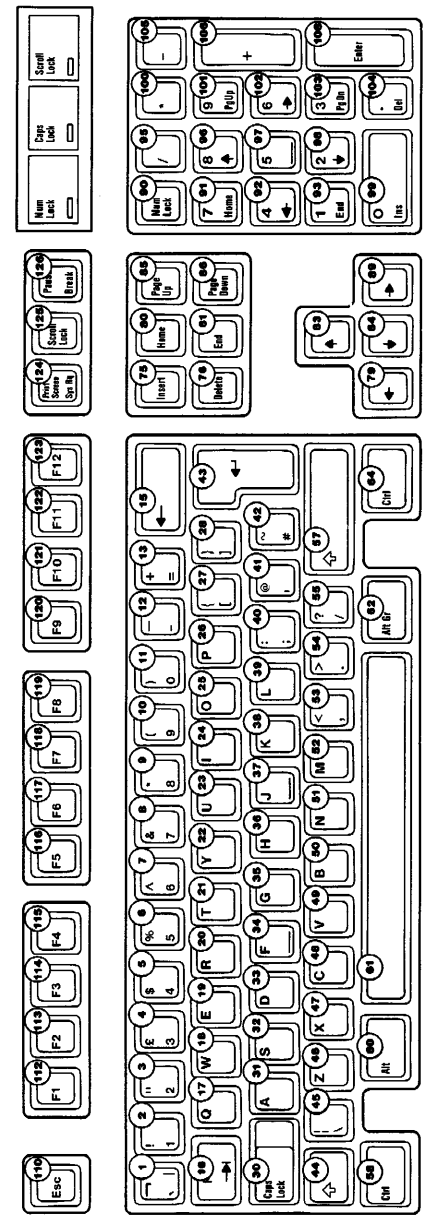
Italian Keyboard



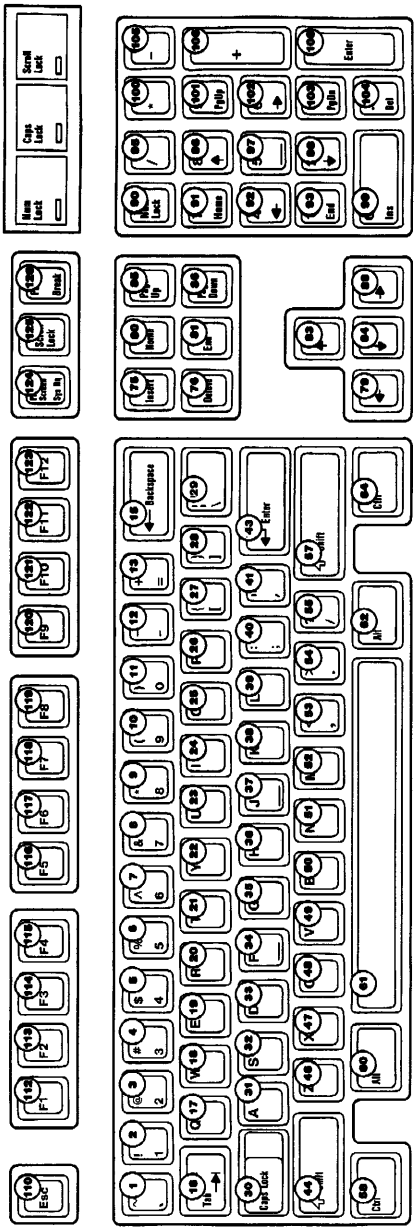
Spanish Keyboard



U.K. English Keyboard



U.S. English Keyboard



Specifications

The specifications for the keyboard are as follows.

Power Requirements

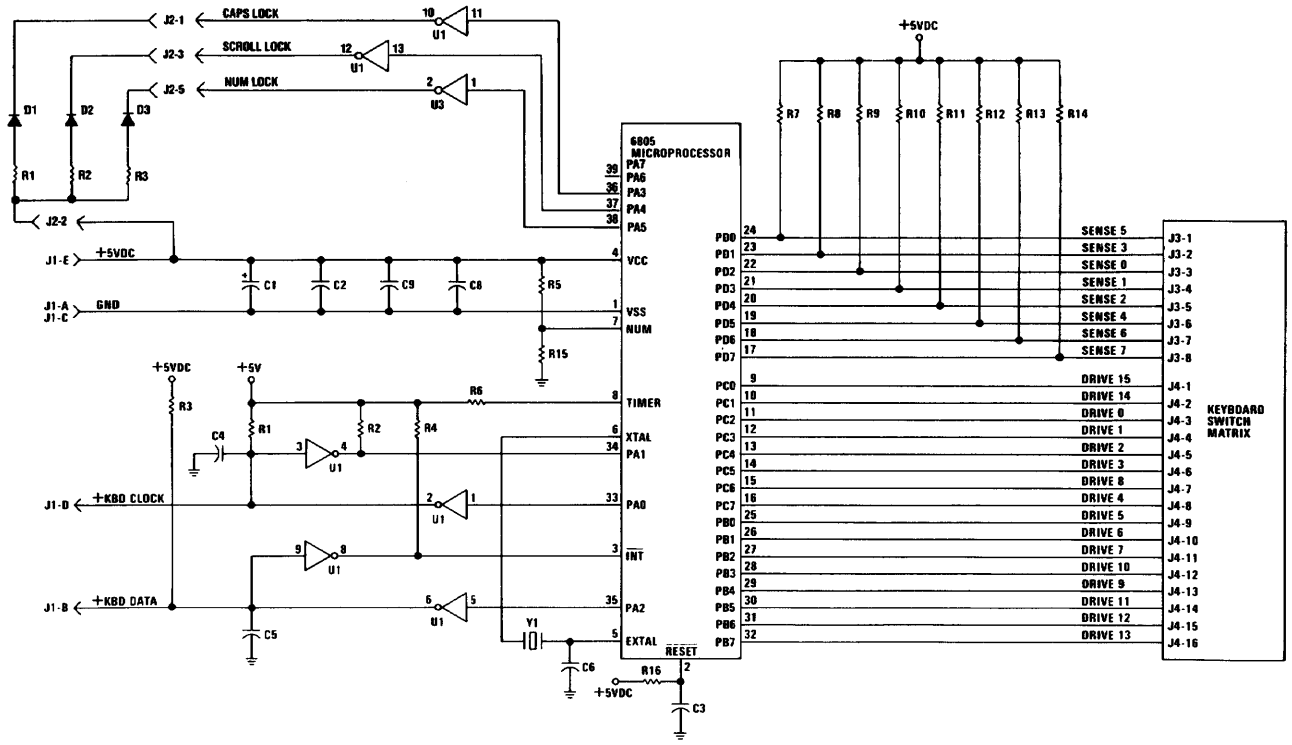
- +5 Vdc ± 10%
- Current cannot exceed 275 mA

Size

- Length: 492 millimeters (19.4 inches)
- Depth: 210 millimeters (8.3 inches)
- Height: 58 millimeters (2.3 inches), legs extended

Weight

2.25 kilograms (5.0 pounds)



101/102-KEY KEYBOARD

SECTION 5. SYSTEM BIOS

Notes:

System BIOS Usage

The basic input/output system (BIOS) resides in ROM on the system board and provides low level control for the major I/O devices in the system and provides system services, such as time-of-day and memory size determination. Additional ROM modules may be placed on option adapters to provide device-level control for that option adapter. BIOS routines enable the assembly language programmer to perform block (disk or diskette) or character-level I/O operations without concern for device address and characteristics.

During POST, a test is made for valid code starting at address hex E0000 and ending at hex EFFFF.

The goal of the BIOS is to provide an operational interface to the system and relieve the programmer of concern about the characteristics of hardware devices. The BIOS interface isolates the user from the hardware, allowing new devices to be added to the system, yet retaining the BIOS level interface to the device. In this manner, hardware modifications and enhancements are not apparent to user programs.

The IBM Personal Computer *Macro Assembler* manual and the IBM Personal Computer *Disk Operating System (DOS)* manual provide useful programming information related to this section. A complete listing of the BIOS is given later in this section.

Access to the BIOS is through program interrupts of the microprocessor in the real mode. Each BIOS entry point is available through its own interrupt. For example, to determine the amount of base RAM available in the system with the microprocessor in the real mode, INT 12H invokes the BIOS routine for determining the memory size and returns the value to the caller.

Parameter Passing

All parameters passed to and from the BIOS routines go through the 80286 registers. The prolog of each BIOS function indicates the registers used on the call and return. For the memory size example, no parameters are passed. The memory size, in 1K increments, is returned in the AX register.

If a BIOS function has several possible operations, the AH register is used at input to indicate the desired operation. For example, to set the time of day, the following code is required:

```
MOV   AH,1              ; function is to set time-of-day
MOV   CX,HIGH_COUNT     ; establish the current time
MOV   DX,LOW_COUNT
INT   1AH               ; set the time
```

To read the time of day:

```
MOV   AH,0              ; function is to read time-of-day
INT   1AH               ; read the timer
```

The BIOS routines save all registers except for AX and the flags. Other registers are modified on return only if they are returning a value to the caller. The exact register usage can be seen in the prolog of each BIOS function.

The following figure shows the interrupts with their addresses and functions.

Int	Address	Name	BIOS Entry
0	0-3	Divide by Zero	D11
1	4-7	Single Step	D11
2	8-B	Non-maskable	NMI INT
3	C-F	Breakpoint	D11
4	10-13	Overflow	D11
5	14-17	Print Screen	PRINT_SCREEN
6	18-1B	Reserved	D11
7	1C-1F	Reserved	D11
8	20-23	Time of Day	TIMER_INT
9	24-27	Keyboard	KB_INT
A	28-2B	Reserved	D11
B	2C-2F	Communications	D11
C	30-33	Communications	D11
D	34-37	Alternate Printer	D11
E	38-3B	Diskette	DISK_INT
F	3C-3F	Printer	D11
10	40-43	Video	VIDEO_IO
11	44-47	Equipment Check	EQUIPMENT
12	48-4B	Memory	MEMORY_SIZE_ DETERMINE
13	4C-4F	Diskette/Disk	DISKETTE_IO
14	50-53	Communications	RS232_IO
15	54-57	Cassette	CASSETTE IO/System Extensions
16	58-5B	Keyboard	KEYBOARD_IO
17	5C-5F	Printer	PRINTER_IO
18	60-63	Resident BASIC	F600:0000
19	64-67	Bootstrap	BOOTSTRAP
1A	68-6B	Time of Day	TIME_OF_DAY
1B	6C-6F	Keyboard Break	DUMMY_RETURN
1C	70-73	Timer Tick	DUMMY_RETURN
1D	74-77	Video Initialization	VIDEO_PARMS
1E	78-7B	Diskette Parameters	DISK_BASE
1F	7C-7F	Video Graphics Chars	0

80286-2 Program Interrupt Listing (Real Mode Only)

Note: For BIOS index, see the BIOS Quick Reference on page 5-14.

The following figure shows hardware, BASIC, and DOS reserved interrupts.

Interrupt	Address	Function
20	80-83	DOS program terminate
21	84-87	DOS function call
22	88-8B	DOS terminate address
23	8C-8F	DOS Ctrl Break exit address
24	90-93	DOS fatal error vector
25	94-97	DOS absolute disk read
26	98-9B	DOS absolute disk write
27	9C-9F	DOS terminate, fix in storage
28-3F	A0-FF	Reserved for DOS
40-5F	100-17F	Reserved for BIOS
60-67	180-19F	Reserved for user program interrupts
68-6F	1A0-1BF	Not used
70	1C0-1C3	IRQ 8 Realtime clock INT (BIOS entry RTC_INT)
71	1C4-1C7	IRQ 9 (BIOS entry RE_DIRECT)
72	1C8-1CB	IRQ 10 (BIOS entry DT1)
73	1CC-1CF	IRQ 11 (BIOS entry D11)
74	1D0-1D3	IRQ 12 (BIOS entry D11)
75	1D4-1D7	IRQ 13 BIOS Redirect to NMI interrupt (BIOS entry INT_287)
76	1D8-1DB	IRQ 14 (BIOS entry D11)
77	1DC-1DF	IRQ 15 (BIOS entry D11)
78-7F	1E0-1FF	Not used
80-85	200-217	Reserved for BASIC
86-F0	218-3C3	Used by BASIC interpreter while BASIC is running
F1-FF	3C4-3FF	Not used

Hardware, Basic, and DOS Interrupts

Vectors with Special Meanings

Interrupt 15—Cassette I/O: This vector points to the following functions:

- Device open
- Device closed
- Program termination
- Event wait
- Joystick support
- System Request key pressed

- Wait
- Move block
- Extended memory size determination
- Processor to protected mode

Additional information about these functions may be found in the BIOS listing.

Interrupt 1B—Keyboard Break Address: This vector points to the code that is executed when the Ctrl and Break keys are pressed. The vector is invoked while responding to a keyboard interrupt, and control should be returned through an IRET instruction. The power-on routines initialize this vector to point to an IRET instruction so that nothing will occur when the Ctrl and Break keys are pressed unless the application program sets a different value.

This routine may retain control with the following considerations:

- The Break may have occurred during interrupt processing, so that one or more End of Interrupt commands must be sent to the 8259 controller.
- All I/O devices should be reset in case an operation was underway at the same time.

Interrupt 1C—Timer Tick: This vector points to the code that will be executed at every system-clock tick. This vector is invoked while responding to the timer interrupt, and control should be returned through an IRET instruction. The power-on routines initialize this vector to point to an IRET instruction, so that nothing will occur unless the application modifies the pointer. The application must save and restore all registers that will be modified. When control is passed to an application with this interrupt, all hardware interrupts from the 8259 interrupt controller are disabled.

Interrupt 1D—Video Parameters: This vector points to a data region containing the parameters required for the initialization of the 6845 on the video adapter. Notice that there are four separate tables, and all four must be reproduced if all modes of operation are to be supported. The power-on routines initialize this vector to point to the parameters contained in the ROM video routines.

Interrupt 1E—Diskette Parameters: This vector points to a data region containing the parameters required for the diskette drive. The power-on routines initialize this vector to point to the parameters contained in the ROM diskette routine. These default parameters represent the specified values for any IBM drives attached to the system. Changing this parameter block may be necessary to reflect the specifications of other drives attached.

Interrupt 1F—Graphics Character Extensions: When operating in graphics modes 320 x 200 or 640 x 200, the read/write character interface will form a character from the ASCII code point, using a set of dot patterns. ROM contains the dot patterns for the first 128 code points. For access to the second 128 code points, this vector must be established to point at a table of up to 1K, where each code point is represented by 8 bytes of graphic information. At power-on time, this vector is initialized to 000:0, and the user must change this vector if the additional code points are required.

Interrupt 40—Reserved: When a Fixed Disk and Diskette Drive Adapter is installed, the BIOS routines use interrupt 40 to revector the diskette pointer.

Interrupt 41 and 46—Fixed Disk Parameters: These vectors point to the parameters for the fixed disk drives, 41 for the first drive and 46 for the second. The power-on routines initialize the vectors to point to the appropriate parameters in the ROM disk routine if CMOS is valid. The drive type codes in CMOS are used to select which parameter set each vector is pointed to. Changing this parameter hook may be necessary to reflect the specifications of other fixed drives attached.

Other Read/Write Memory Usage

The IBM BIOS routines use 256 bytes of memory from absolute hex 400 to hex 4FF. Locations hex 400 to 407 contain the base addresses of any RS-232C adapters installed in the system. Locations hex 408 to 40F contain the base addresses of any printer adapters.

Memory locations hex 300 to hex 3FF are used as a stack area during the power-on initialization and bootstrap, when control is passed to it from power-on. If the user desires the stack to be in a different area, that area must be set by the application.

The following figure shows the reserved memory locations.

Address	Mode	Function
400-4A1	ROM BIOS	See BIOS listing
4A2-4EF		Reserved
4F0-4FF		Reserved as intra-application communication area for any application
500-5FF		Reserved for DOS and BASIC
500	DOS	Print screen status flag store 0=Print screen not active or successful print screen operation 1=Print screen in progress 255=Error encountered during print screen operation
504	DOS	Single drive mode status byte
510-511	BASIC	BASIC's segment address store
512-515	BASIC	Clock interrupt vector segment:offset store
516-519	BASIC	Break key interrupt vector segment:offset store
51A-51D	BASIC	Disk error interrupt vector segment:offset store

Reserved Memory Locations

SECTION 5

The following is the BASIC workspace for DEF SEG (default workspace).

Offset	Length	
2E	2	Line number of current line being executed
347	2	Line number of last error
30	2	Offset into segment of start of program text
358	2	Offset into segment of start of variables (end of program text 1-1)
6A	1	Keyboard buffer contents 0=No characters in buffer 1=Characters in buffer
4E	1	Character color in graphics mode*
*Set to 1, 2, or 3 to get text in colors 1-3. Do not set to 0. The default is 3.		

Basic Workspace Variables

Example

100 PRINT PEEK (&H2E) + 256 x PEEK (&H2F)

L	H
Hex 64	Hex 00

The following is a BIOS memory map.

Starting Address	
00000	BIOS interrupt vectors
001E0	Available interrupt vectors
00400	BIOS data area
00500	User read/write memory
E0000	Read only memory
F0000	BIOS program area

BIOS Memory Map

BIOS Programming Hints

The BIOS code is invoked through program interrupts. The programmer should not "hard code" BIOS addresses into applications. The internal workings and absolute addresses within BIOS are subject to change without notice.

If an error is reported by the disk or diskette code, reset the drive adapter and retry the operation. A specified number of retries

should be required for diskette reads to ensure the problem is not due to motor startup.

When altering I/O-port bit values, the programmer should change only those bits necessary to the current task. Upon completion, the original environment should be restored. Failure to adhere to this practice may cause incompatibility with present and future applications.

Additional information for BIOS programming can be found in Section 8 of this manual.

Move Block BIOS

The Move Block BIOS was designed to make use of the memory above the 1M address boundary while operating with IBM DOS. The Block Move is done with the Intel 80286 Microprocessor operating in the protected mode.

Because the interrupts are disabled in the protected mode, Move Block BIOS may demonstrate a data overrun or lost interrupt situation in certain environments.

Communication devices, while receiving data, are sensitive to these interrupt routines; therefore, the timing of communication and the Block Move should be considered. The following table shows the interrupt servicing requirements for communication devices.

Baud Rate	11 Bit (ms)	9 bit (ms)
300	33.33	30.00
1200	8.33	7.50
2400	4.16	7.50
4800	2.08	1.87
9600	1.04	0.93
Times are approximate		

Communication Interrupt Intervals

The following table shows the time required to complete a Block Move.

Block Size	Buffer Addresses	Time in ms
Normal 512 Byte	Both even	0.98
	Even and odd	1.04
	Both odd	1.13
Maximum 64K	Both even	37.0
	Even and odd	55.0
	Both odd	72.0
Time is approximate		

Move Block BIOS Timing

Following are some ways to avoid data overrun errors and loss of interrupts:

- Do not use the Block Move while communicating, or
- Restrict the block size to 512 bytes or less while communicating, or
- Use even address buffers for both the source and the destination to keep the time for a Block Move to a minimum.

Adapters with System-Accessible ROM Modules

The ROM BIOS provides a way to integrate adapters with on-board ROM code into the system. During POST, interrupt vectors are established for the BIOS calls. After the default vectors are in place, a scan for additional ROM modules occurs. At this point, a ROM routine on an adapter may gain control and establish or intercept interrupt vectors to hook themselves into the system.

The absolute addresses hex C8000 through E0000 are scanned in 2K blocks in search of a valid adapter ROM. A valid ROM is defined as follows:

Byte 0 Hex 55
Byte 1 Hex AA

Byte 2 A length indicator representing the number of 512-byte blocks in the ROM

Byte 3 Entry by a CALL FAR

A checksum is also done to test the integrity of the ROM module. Each byte in the defined ROM module is summed modulo hex 100. This sum must be 0 for the module to be valid.

When the POST identifies a valid ROM, it does a CALL FAR to byte 3 of the ROM, which should be executable code. The adapter can now perform its power-on initialization tasks. The adapter's ROM should then return control to the BIOS routines by executing a RETURN FAR.

Quick Reference

Address		Publics by Name
F000:E729		A1
F000:3DE6		ACT_DISP_PAGE
F000:6000		BASIC
F000:1A04		BEEP
F000:1B2E		BLINK_INT
F000:2038		BOOT_STRAP_1
F000:0CA3		C21
F000:039D		C8042
F000:4503		CASSETTE_IO_1
F000:194F		CMOS_READ
F000:196B		CMOS_WRITE
F000:1A59		CONFIG_BAD
F000:E6F5		CONF_TBL
F000:FA6E		CRT_CHAR_GEN
F000:E020		D1
F000:1BE0		D11
F000:E030		D2
F000:E040		D2A
F000:1989		DDS
F000:2159		DISKETTE_IO_1
F000:EFC7		DISK_BASE
F000:2C72		DISK_INT_1
F000:2E86		DISK_IO
F000:2CDD		DISK_SETUP
F000:2C89		DSKETTE_SETUP
F000:FF53		DUMMY_RETURN
F000:1C2E		DUMMY_RETURN_1
F000:E05E		E101
F000:E07T		E102
F000:E090		E103
F000:E0A9		E104
F000:E0C2		E105
F000:E0DB		E106
F000:E0F4		E107
F000:E10D		E108
F000:E126		E109
F000:E13F		E161
F000:E168		E162
F000:E191		E163
F000:E1B7		E164
F000:E1DB		E201
F000:E1EE		E202
F000:E209		E203
F000:E224		E301
F000:E239		E302
F000:E2C6		E303
F000:E2EA		E304
F000:E30E		E401
F000:E31E		E501
F000:E32E		E601
F000:E343		E602
F000:4475		EQUIPMENT_1
F000:19C6		ERR_BEEP
F000:1991		E_MSG
F000:E364		FT780
F000:E379		F1781
F000:E38E		F1782
F000:E3AC		F1790
F000:E3BF		F1791
F000:E3D2		F3A
F000:E26D		F3D
F000:E3DF		F3D1
F000:E401		FD_TBL
F000:4C76		FILL
F000:FF5E		FLOPPY
F000:48DD		GATE_A20
F000:33B7		HD_INT
F000:FF5A		HRD
F000:1C38		INT_287
F000:E6E6		K10
F000:E98A		K11
F000:E9C4		K12
F000:E9D0		K14
F000:E92C		K15
F000:35AE		K16
F000:E87E		K6
F000:0008	Abs	K6L
F000:E886		K7
F000:E88E		K8
F000:1B03		KBD_RESET
F000:354F		KB_INT_1
F000:33DA		KEYBOARD_IO_1
F000:0010	Abs	M4
F000:F0E4		M5
F000:F0EC		M6
F000:F0F4		M7
F000:446B		MEMORY_SIZE_DET_1
F000:E2C3		NMI_INT
F000:447F		NMI_INT_1
F000:03A9		OBF_42
F000:0000		POST1
F000:0CA3		POST2
F000:167F		POST3
F000:194F		POST4
F000:1C4E		POST5
F000:1ECB		POST6
F000:3AC0		PRINTER_IO_1
F000:FF54		PRINT_SCREEN
F000:4B97		PRINT_SCREEN_1
F000:1C47		PROC_SHUTDOWN
F000:1A99		PROT_PRT_HEX
F000:1A7D		PRT_HEX
F000:1A84		PRT_SEG
F000:19B8		P_MSG
F000:FFF0		P_O_R
F000:3F3F		READ_AC_CURRENT
F000:3DCF		READ_CURSOR
F000:407F		READ_DOT
F000:43C5		READ_LPEN
F000:1C2F		RE_DIRECT
F000:1AD1		ROM_CHECK
F000:1AC5		ROM_CHECKSUM
F000:1FCB		ROM_ERR
F000:3B4F		RS232_IO_1
F000:4B1B		RTC_INT
F000:3EED		SCROLL_DOWN
F000:3E4F		SCROLL_UP

Address		Publics by Value
F000:0000		POST1
F000:0008	Abs	K6L
F000:0010	Abs	M4
F000:0050		START_1
F000:039D		C8042
F000:03A9		OBF_42
F000:0CA3		C21
F000:0CA3		POST2
F000:105F		SHUT3
F000:10C3		SHUT2
F000:10C6		SHUT7
F000:10E7		SHUT6
F000:1620		SHUT4
F000:167F		POST3
F000:194F		CMOS_READ
F000:194F		POST4
F000:196B		CMOS_WRITE
F000:1989		DDS
F000:1991		E_MSG
F000:19B8		P_MSG
F000:19C6		ERR_BEEP
F000:1A04		BEEP
F000:1A4A		WAITF
F000:1A59		CONFIG_BAD
F000:1A6D		XPC_BYTE
F000:1A7D		PRT_HEX
F000:1A84		PRT_SEG
F000:1A99		PROT_PRT_HEX
F000:1AC5		ROM_CHECKSUM
F000:1AD1		ROM_CHECK
F000:1B03		KBD_RESET
F000:1B2E		BLINK_INT
F000:1B3C		SET_TOD
F000:1BE0		D11
F000:1C2E		DUMMY_RETURN_1
F000:1C2F		RE_DIRECT
F000:1C38		INT_287
F000:1C47		PROC_SHUTDOWN
F000:1C4E		POST5
F000:1D40		SYSINIT1
F000:1ECB		POST6
F000:1ECB		STGTST_CNT
F000:1FCB		ROM_ERR
F000:1FF7		XMIT_8042
F000:2038		BOOT_STRAP_1
F000:2159		DISKETTE_IO_1
F000:2B1D		SEEK
F000:2C72		DISK_INT_1
F000:2C89		DSKETTE_SETUP
F000:2CDD		DISK_SETUP
F000:2E86		DISK_IO
F000:33B7		HD_INT
F000:33DA		KEYBOARD_IO_1
F000:354F		KB_INT_1
F000:36AE		K16
F000:3A25		SND_DATA
F000:3AC0		PRINTER_IO_1
F000:3B4F		RS232_IO_1
F000:3C64		VIDEO_IO_1
F000:3C98		SET_MODE
F000:3D62		SET_CTYPE
F000:3DA7		SET_CPOS
F000:3DCF		READ_CURSOR
F000:3DE6		ACT_DISP_PAGE
F000:3E08		SET_COLOR
F000:3E2E		VIDEO_STATE
F000:3E4F		SCROLL_UP
F000:3EED		SCROLL_DOWN
F000:3F3F		READ_AC_CURRENT
F000:3F9C		WRITE_AC_CURRENT
F000:3FCE		WRITE_C_CURRENT
F000:407F		READ_DOT
F000:4090		WRITE_DOT
F000:433E		WRITE_TTY
F000:43C5		READ_LPEN
F000:446B		MEMORY_SIZE_DET_1
F000:4475		EQUIPMENT_1
F000:447F		NMI_INT_1
F000:4503		CASSETTE_IO_1
F000:4799		SHUT9
F000:48DD		GATE_A20
F000:4999		TIME_OF_DAY_1
F000:4B1B		RTC_INT
F000:4B97		PRINT_SCREEN_1
F000:4C2D		TIMER_INT_1
F000:4C76		FILL
F000:6000		BASIC
F000:E020		D1
F000:E030		D2
F000:E040		D2A
F000:E05E		E101
F000:E07T		E102
F000:E090		E103
F000:E0A9		E104
F000:E0C2		E105
F000:E0DB		E106
F000:E0F4		E107
F000:E10D		E108
F000:E126		E109
F000:E13F		E161
F000:E168		E162
F000:E191		E163
F000:E1B7		E164
F000:E1DB		E201
F000:E1EE		E202
F000:E209		E203
F000:E224		E301
F000:E239		E302
F000:E25D		F3D
F000:E2C3		NMI_INT
F000:E2C6		E303
F000:E2EA		E304
F000:E30E		E401
F000:E31E		E501
F000:E32E		E601
F000:E343		E602
F000:E364		F1780

```
F000:2B1D       SEEK                        F000:E379       F1781
F000:FF62       SEEKS_1                     F000:E38E       F1782
F000:3E08       SET_COLOR                   F000:E3AC       F1790
F000:3DA7       SET_CPOS                    F000:E3BF       F1791
F000:3D82       SET_CTYPE                   F000:E3D2       F3A
F000:3C9B       SET_MODE                    F000:E3DF       F3D1
F000:1B3C       SET_TOD                     F000:E401       FD_TBL
F000:10C3       SHUT2                       F000:E6F5       CONF_TBL
F000:105F       SHUT3                       F000:E729       A1
F000:1620       SHUT4                       F000:E87E       K6
F000:10E7       SHUT6                       F000:E886       K7
F000:10C6       SHUT7                       F000:E88E       K8
F000:4799       SHUT9                       F000:E8E6       K10
F000:FF23       SLAVE_VECTOR_TABLE          F000:E92C       K15
F000:3A25       SND_DATA                    F000:E98A       K11
F000:0050       START_1                     F000:E9C4       K12
F000:1ECB       STGTST_CNT                  F000:E9D0       K14
F000:1D40       SYSINIT1                    F000:EFC7       DISK_BASE
F000:4C2D       TIMER_INT_1                 F000:F0A4       VIDEO_PARMS
F000:4999       TIME_OF_DAY_1               F000:F0E4       M5
F000:FF66       TUTOR                       F000:F0EC       M6
F000:FEF3       VECTOR_TABLE                F000:F0F4       M7
F000:3C64       VIDEO_IO_1                  F000:FA6E       CRT_CHAR_GEN
F000:F0A4       VIDEO_PARMS                 F000:FEF3       VECTOR_TABLE
F000:3E2E       VIDEO_STATE                 F000:FF23       SLAVE_VECTOR_TABLE
F000:1A4A       WAITF                       F000:FF53       DUMMY_RETURN
F000:3F9C       WRITE_AC_CURRENT            F000:FF54       PRINT_SCREEN
F000:3FCE       WRITE_C_CURRENT             F000:FF5A       HRD
F000:4090       WRITE_DOT                   F000:FF5E       FLOPPY
F000:433E       WRITE_TTY                   F000:FF62       SEEKS_1
F000:1FF7       XMIT_8042                   F000:FF66       TUTOR
F000:1A6D       XPC_BYTE                    F000:FFF0       P_O_R
```

THE BIOS ROUTINES ARE MEANT TO BE ACCESSED THROUGH SOFTWARE INTERRUPTS ONLY. ANY ADDRESSES PRESENT IN THESE LISTINGS ARE INCLUDED ONLY FOR COMPLETENESS, NOT FOR REFERENCE. APPLICATIONS WHICH REFERENCE ANY ABSOLUTE ADDRESSES WITHIN THE CODE SEGMENTS OF BIOS VIOLATE THE STRUCTURE AND DESIGN OF BIOS.

```
                        PAGE 118,121
                        TITLE TEST1 ---- 04/21/86  POWER ON SELF TEST (POST)
                        .286C
                        ;--------------------------------------------------------------------------------
                        ;                                                                              :
                        ; BIOS I/O INTERFACE                                                           :
                        ;                                                                              :
                        ;       THESE LISTINGS PROVIDE INTERFACE INFORMATION FOR ACCESSING              :
                        ;       THE BIOS ROUTINES.  THE POWER ON SELF TEST IS INCLUDED.                :
                        ;                                                                              :
                        ;       THE  BIOS  ROUTINES  ARE  MEANT  TO  BE  ACCESSED  THROUGH            :
                        ;       SOFTWARE  INTERRUPTS  ONLY.  ANY  ADDRESSES  PRESENT  IN              :
                        ;       THESE  LISTINGS  ARE  INCLUDED  ONLY  FOR  COMPLETENESS,              :
                        ;       NOT  FOR  REFERENCE.  APPLICATIONS  WHICH  REFERENCE  ANY             :
                        ;       ABSOLUTE  ADDRESSES  WITHIN  THE  CODE  SEGMENTS  OF  BIOS            :
                        ;       VIOLATE  THE  STRUCTURE  AND  DESIGN  OF  BIOS.                       :
                        ;                                                                              :
                        ;--------------------------------------------------------------------------------


                        ;--------------------------------------------------------------------------------
                        ;   MODULE REFERENCE                                                           :
                        ;                                                                              :
                        ;    TEST1.ASM      -->  POST AND MANUFACTURING TEST ROUTINES                  :
                        ;      DSEG.INC     -->    DATA SEGMENTS LOCATIONS                             :
                        ;      POSTEQU.INC  -->    COMMON EQUATES FOR POST AND BIOS                    :
                        ;      SYSDATA.INC  -->    POWER ON SELF TEST EQUATES FOR PROTECTED MODE       :
                        ;                            POST TEST.01 THROUGH TEST.16                      :
                        ;    TEST2.ASM      -->  POST TEST AND INITIALIZATION ROUTINES                 :
                        ;                            POST TEST.17 THROUGH TEST.22                      :
                        ;    TEST3.ASM      -->  POST EXCEPTION INTERRUPT TESTS                        :
                        ;    TEST4.ASM      -->  POST AND BIOS UTILITY ROUTINES                        :
                        ;                          CMOS_READ       - READ CMOS LOCATION ROUTINE        :
                        ;                          CMOS_WRITE      - WRITE CMOS LOCATION ROUTINE       :
                        ;                          DDS             - LOAD (DS:) WITH DATA SEGMENT      :
                        ;                          E_MSG           - POST ERROR MESSAGE HANDLER        :
                        ;                          MFG_HALT        - MANUFACTURING ERROR TRAP          :
                        ;                          P_MSG           - POST STRING DISPLAY ROUTINE       :
                        ;                          ERR_BEEP        - POST ERROR BEEP PROCEDURE         :
                        ;                          BEEP            - SPEAKER BEEP CONTROL ROUTINE      :
                        ;                          WAITF           - FIXED TIME WAIT ROUTINE           :
                        ;                          CONFIG_BAD      - SET BAD_CONFIG IN CMOS_DIAG       :
                        ;                          XPC_BYTE        - DISPLAY HEX BYTE AS 00 - FF       :
                        ;                          PRT_HEX         - DISPLAY CHARACTER                 :
                        ;                          PRT_SEG         - DISPLAY SEGMENT FORMAT ADDRESS    :
                        ;                          PROT_PRT_HEX    - POST PROTECTED MODE DISPLAY       :
                        ;                          ROM_CHECKSUM    - CHECK ROM MODULES FOR CHECKSUM    :
                        ;                          ROM_CHECK       - ROM SCAN AND INITIALIZE           :
                        ;                          KBD_RESET       - POST KEYBOARD RESET ROUTINE       :
                        ;                          BLINK_INT       - MANUFACTURING TOGGLE BIT ROUTINE  :
                        ;                          SET_TOD         - SET TIMER FROM CMOS RTC           :
                        ;                          D11             - DUMMY INTERRUPT HANDLER   ->INT 77H :
                        ;                          RE_DIRECT       - HARDWARE INT  9 REDIRECT (L 2)    :
                        ;                          INT_287         - HARDWARE INT 13 REDIRECT (287)    :
                        ;                          PROC_SHUTDOWN   - 80286 RESET ROUTINE               :
                        ;    TEST5.ASM      -->  EXCEPTION INTERRUPT TEST HANDLERS FOR POST TESTS      :
                        ;                          SYSINIT1        - BUILD PROTECTED MODE POINTERS     :
                        ;                          GDT_BLD         - BUILD THE GDT FOR POST            :
                        ;                          SIDT_BLD        - BUILD THE IDT FOR POST            :
                        ;    TEST6.ASM      -->  POST TESTS AND SYSTEM BOOT STRAP                      :
                        ;                          STGTST_CNT      - SEGMENT STORAGE TEST              :
                        ;                          ROM_ERR         - ROM ERROR DISPLAY ROUTINE         :
                        ;                          XMIT_8042       - KEYBOARD DIAGNOSTIC OUTPUT        :
                        ;                          BOOT_STRAP      - BOOT STRAP LOADER          -INT 19H :
                        ;                                                                              :
                        ;    DSKETTE.ASM    -->  DISKETTE BIOS                                         :
                        ;                          DISKETTE_IO_1 - INT 13H BIOS ENTRY (40H)   -INT 13H :
                        ;                          DISK_INT_1    - HARDWARE INTERRUPT HANDLER -INT 0EH :
                        ;                          DSKETTE_SETUP - POST SETUP DRIVE TYPES              :
                        ;    DISK.ASM       -->  FIXED DISK BIOS                                       :
                        ;                          DISK_SETUP      - SETUP DISK VECTORS AND TEST       :
                        ;                          DISK_IO         - INT 13H BIOS ENTRY        -INT 13H :
                        ;                          HD_INT          - HARDWARE INTERRUPT HANDLER -INT 76H :
                        ;    KYBD.ASM       -->  KEYBOARD BIOS                                         :
                        ;                          KEYBOARD_IO_1 - INT 16H BIOS ENTRY          -INT 16H :
                        ;                          KB_INT_1      - HARDWARE INTERRUPT          -INT 09H :
                        ;                          SND_DATA      - KEYBOARD TRANSMISSION               :
                        ;    PRT.ASM        -->  PRINTER ADAPTER BIOS                          -INT 17H :
                        ;    RS232.ASM      -->  COMMUNICATIONS BIOS FOR RS232                 -INT 14H :
                        ;    VIDEO1.ASM     -->  VIDEO BIOS                                    -INT 10H :
                        ;    BIOS.ASM       -->  BIOS ROUTINES                                         :
                        ;                          MEMORY_SIZE_DET_1 - REAL MODE SIZE          -INT 12H :
                        ;                          EQUIPMENT_1   - EQUIPMENT DETERMINATION     -INT 11H :
                        ;                          NMI_INT_1     - NMI HANDLER                 -INT 02H :
                        ;    BIOS1.ASM      -->  INTERRUPT 15H BIOS ROUTINES                   -INT 15H :
                        ;                          DEV_OPEN        - NULL DEVICE OPEN HANDLER          :
                        ;                          DEV_CLOSE       - NULL DEVICE CLOSE HANDLER         :
                        ;                          PROG_TERM       - NULL PROGRAM TERMINATION          :
                        ;                          EVENT_WAIT      - RTC EVENT WAIT/TIMEOUT ROUTINE    :
                        ;                          JOY_STICK       - JOYSTICK PORT HANDLER             :
                        ;                          SYS_REQ         - NULL SYSTEM REQUEST KEY           :
                        ;                          WAIT            - RTC TIMED WAIT ROUTINE            :
                        ;                          BLOCKMOVE       - EXTENDED MEMORY MOVE INTERFACE    :
                        ;                          GATE_A20        - ADDRESS BIT 20 CONTROL            :
                        ;                          EXT_MEMORY      - EXTENDED MEMORY SIZE DETERMINE    :
                        ;                          SET_VMODE       - SWITCH PROCESSOR TO VIRTUAL MODE  :
                        ;                          DEVICE_BUSY     - NULL DEVICE BUSY HANDLER          :
                        ;                          INT_COMPLETE    - NULL INTERRUPT COMPLETE HANDLER   :
                        ;    BIOS2.ASM      -->  BIOS INTERRUPT ROUTINES                               :
                        ;                          TIME_OF_DAY_1 - TIME OF DAY ROUTINES        -INT 1AH :
                        ;                          RTC_INT       - IRQ LEVEL 8 ALARM HANDLER   -INT 70H :
                        ;                          PRINT_SCREEN1 - PRINT SCREEN ROUTINE        -INT 05H :
                        ;                          TIMER_INT_1   - TIMER1 INTERRUPT HANDLER  ->INT 1CH :
                        ;    ORGS.ASM       -->  COMPATIBILITY MODULE                                  :
                        ;                          POST ERROR MESSAGES                                 :
                        ;                          DISKETTE - DISK - VIDEO DATA TABLES                 :
                        ;--------------------------------------------------------------------------------
                        .LIST
```

```
                        PAGE
                C       INCLUDE DSEG.INC
                C       ;--------------------------------------
                C       ;       80286 INTERRUPT LOCATIONS      :
                C       ;       REFERENCED BY POST & BIOS      :
                C       ;--------------------------------------
                C
0000            C       ABS0            SEGMENT AT 0            ; ADDRESS= 0000:0000
                C
0000 ??         C       @STG_LOC0       DB      ?               ; START OF INTERRUPT VECTOR TABLE
                C
0008            C                       ORG     4*002H
0008 ????????   C       @NMI_PTR        DD      ?               ; NON-MASKABLE INTERRUPT VECTOR
                C
0014            C                       ORG     4*005H
0014 ????????   C       @INT5_PTR       DD      ?               ; PRINT SCREEN INTERRUPT VECTOR
                C
0020            C                       ORG     4*008H
0020 ????????   C       @INT_PTR        DD      ?               ; HARDWARE INTERRUPT POINTER (8-F)
                C
0040            C                       ORG     4*010H
0040 ????????   C       @VIDEO_INT      DD      ?               ; VIDEO I/O INTERRUPT VECTOR
                C
004C            C                       ORG     4*013H
004C ????????   C       @ORG_VECTOR     DD      ?               ; DISKETTE/DISK INTERRUPT VECTOR
                C
0060            C                       ORG     4*018H
0060 ????????   C       @BASIC_PTR      DD      ?               ; POINTER TO CASSETTE BASIC
                C
0074            C                       ORG     4*01DH
0074 ????????   C       @PARM_PTR       DD      ?               ; POINTER TO VIDEO PARAMETERS
                C
0078            C                       ORG     4*01EH
0078 ????????   C       @DISK_POINTER   DD      ?               ; POINTER TO DISKETTE PARAMETER TABLE
                C
007C            C                       ORG     4*01FH
007C ????????   C       @EXT_PTR        DD      ?               ; POINTER TO GRAPHIC CHARACTERS 128-255
                C
0100            C                       ORG     4*040H
0100 ????????   C       @DISK_VECTOR    DD      ?               ; POINTER TO DISKETTE INTERRUPT CODE
                C
0104            C                       ORG     4*041H
0104 ????????   C       @HF_TBL_VEC     DD      ?               ; POINTER TO FIRST DISK PARAMETER TABLE
                C
0118            C                       ORG     4*046H
0118 ????????   C       @HF1_TBL_VEC    DD      ?               ; POINTER TO SECOND DISK PARAMETER TABLE
                C
01C0            C                       ORG     4*070H
01C0 ????????   C       @SLAVE_INT_PTR  DD      ?               ; POINTER TO SLAVE INTERRUPT HANDLER
                C
01D8            C                       ORG     4*076H
01D8 ????????   C       @HDISK_INT      DD      ?               ; POINTER TO FIXED DISK INTERRUPT CODE
                C
0400            C                       ORG     0400H
0400 ????       C       @TOS            DW      ?               ; STACK -- USED DURING POST ONLY
                C                                               ;   USE WILL OVERLAY INTERRUPTS VECTORS
                C
0500            C                       ORG     0500H
0500            C       @MFG_TEST_RTN   LABEL   FAR             ; LOAD LOCATION FOR MANUFACTURING TESTS
                C
7C00            C                       ORG     7C00H
7C00            C       @BOOT_LOCN      LABEL   FAR             ; BOOT STRAP CODE LOAD LOCATION
                C
7C00            C       ABS0            ENDS
```

```
                        C  PAGE
                        C  ;--------------------------------------
                        C  ;       ROM BIOS DATA AREAS           :
                        C  ;--------------------------------------
                        C
0000                    C  DATA            SEGMENT AT 40H          ; ADDRESS= 0040:0000
                        C
0000 ????               C  @RS232_BASE     DW      ?               ; BASE ADDRESSES OF RS232 ADAPTERS
0002 ????               C                  DW      ?               ;  SECOND LOGICAL RS232 ADAPTER
0004 ????               C                  DW      ?               ;   RESERVED
0006 ????               C                  DW      ?               ;   RESERVED
0008 ????               C  @PRINTER_BASE   DW      ?               ; BASE ADDRESSES OF PRINTER ADAPTERS
000A ????               C                  DW      ?               ;  SECOND LOGICAL PRINTER ADAPTER
000C ????               C                  DW      ?               ;  THIRD LOGICAL PRINTER ADAPTER
000E ????               C                  DW      ?               ;   RESERVED
0010 ????               C  @EQUIP_FLAG     DW      ?               ; INSTALLED HARDWARE FLAGS
0012 ??                 C  @MFG_TST        DB      ?               ; INITIALIZATION FLAGS
0013 ????               C  @MEMORY_SIZE    DW      ?               ; BASE MEMORY SIZE IN K BYTES  (X 1024)
0015 ??                 C  @MFG_ERR_FLAG   DB      ?               ; SCRATCHPAD FOR MANUFACTURING
0016 ??                 C                  DB      ?               ;  ERROR CODES
                        C
                        C  ;--------------------------------------
                        C  ;       KEYBOARD DATA AREAS           :
                        C  ;--------------------------------------
                        C
0017 ??                 C  @KB_FLAG        DB      ?               ; KEYBOARD SHIFT STATE AND STATUS FLAGS
0018 ??                 C  @KB_FLAG_1      DB      ?               ; SECOND BYTE OF KEYBOARD STATUS
0019 ??                 C  @ALT_INPUT      DB      ?               ; STORAGE FOR ALTERNATE KEY PAD ENTRY
001A ????               C  @BUFFER_HEAD    DW      ?               ; POINTER TO HEAD OF KEYBOARD BUFFER
001C ????               C  @BUFFER_TAIL    DW      ?               ; POINTER TO TAIL OF KEYBOARD BUFFER
                        C
                        C  ;------      HEAD = TAIL    INDICATES THAT THE BUFFER IS EMPTY
                        C
001E   10 [             C  @KB_BUFFER      DW      16 DUP(?)       ; ROOM FOR 15 SCAN CODE ENTRIES
        ????
               ]        C
                        C
                        C
                        C  ;--------------------------------------
                        C  ;       DISKETTE DATA AREAS           :
                        C  ;--------------------------------------
                        C
003E ??                 C  @SEEK_STATUS    DB      ?               ; DRIVE RECALIBRATION STATUS
                        C                                          ; BIT 3-0 = DRIVE 3-0 RECALIBRATION
                        C                                          ; BEFORE NEXT SEEK IF BIT IS = 0
003F ??                 C  @MOTOR_STATUS   DB      ?               ; MOTOR STATUS
                        C                                          ; BIT 3-0 = DRIVE 3-0 CURRENTLY RUNNING
                        C                                          ; BIT 7 = CURRENT OPERATION IS A WRITE
0040 ??                 C  @MOTOR_COUNT    DB      ?               ; TIME OUT COUNTER FOR MOTOR(S) TURN OFF
0041 ??                 C  @DSKETTE_STATUS DB      ?               ; RETURN CODE STATUS BYTE
                        C                                          ; CMD_BLOCK  IN STACK FOR DISK OPERATION
0042   07 [             C  @NEC_STATUS     DB      7 DUP(?)        ; STATUS BYTES FROM DISKETTE OPERATION
        ??
               ]        C
                        C
                        C
                        C  ;--------------------------------------
                        C  ;       VIDEO DISPLAY DATA AREA       :
                        C  ;--------------------------------------
                        C
0049 ??                 C  @CRT_MODE       DB      ?               ; CURRENT DISPLAY MODE (TYPE)
004A ????               C  @CRT_COLS       DW      ?               ; NUMBER OF COLUMNS ON SCREEN
004C ????               C  @CRT_LEN        DW      ?               ; LENGTH OF REGEN BUFFER IN BYTES
004E ????               C  @CRT_START      DW      ?               ; STARTING ADDRESS IN REGEN BUFFER
0050   08 [             C  @CURSOR_POSN    DW      8 DUP(?)        ; CURSOR FOR EACH OF UP TO 8 PAGES
        ????
               ]        C
                        C
0060 ????               C  @CURSOR_MODE    DW      ?               ; CURRENT CURSOR MODE SETTING
0062 ??                 C  @ACTIVE_PAGE    DB      ?               ; CURRENT PAGE BEING DISPLAYED
0063 ????               C  @ADDR_6845      DW      ?               ; BASE ADDRESS FOR ACTIVE DISPLAY CARD
0065 ??                 C  @CRT_MODE_SET   DB      ?               ; CURRENT SETTING OF THE 3X8 REGISTER
0066 ??                 C  @CRT_PALETTE    DB      ?               ; CURRENT PALETTE SETTING - COLOR CARD
                        C
                        C  ;--------------------------------------
                        C  ;       POST AND BIOS WORK DATA AREA  :
                        C  ;--------------------------------------
                        C
                        C                                          ; STACK SAVE, etc.
0067 ????               C  @IO_ROM_INIT    DW      ?               ; POINTER TO ROM INITIALIZATION ROUTINE
0069 ????               C  @IO_ROM_SEG     DW      ?               ; POINTER TO I/O ROM SEGMENT
006B ??                 C  @INTR_FLAG      DB      ?               ; FLAG INDICATING AN INTERRUPT HAPPENED
                        C
                        C  ;--------------------------------------
                        C  ;       TIMER DATA AREA               :
                        C  ;--------------------------------------
                        C
006C ????               C  @TIMER_LOW      DW      ?               ; LOW WORD OF TIMER COUNT
006E ????               C  @TIMER_HIGH     DW      ?               ; HIGH WORD OF TIMER COUNT
0070 ??                 C  @TIMER_OFL      DB      ?               ; TIMER HAS ROLLED OVER SINCE LAST READ
                        C
                        C  ;--------------------------------------
                        C  ;       SYSTEM DATA AREA              :
                        C  ;--------------------------------------
                        C
0071 ??                 C  @BIOS_BREAK     DB      ?               ; BIT 7=1 IF BREAK KEY HAS BEEN PRESSED
0072 ????               C  @RESET_FLAG     DW      ?               ; WORD=1234H IF KEYBOARD RESET UNDERWAY
                        C
                        C  ;--------------------------------------
                        C  ;       FIXED DISK DATA AREAS         :
                        C  ;--------------------------------------
                        C
0074 ??                 C  @DISK_STATUS1   DB      ?               ; FIXED DISK STATUS
0075 ??                 C  @HF_NUM         DB      ?               ; COUNT OF FIXED DISK DRIVES
0076 ??                 C  @CONTROL_BYTE   DB      ?               ; HEAD CONTROL BYTE
0077 ??                 C  @PORT_OFF       DB      ?               ;  RESERVED (PORT OFFSET)
```

```
                        C  PAGE
                        C  ;--------------------------------------
                        C  ;       TIME-OUT VARIABLES             :
                        C  ;--------------------------------------
                        C
0078 ??                 C  @PRINT_TIM_OUT  DB      ?               ; TIME OUT COUNTERS FOR PRINTER RESPONSE
0079 ??                 C                  DB      ?               ;  SECOND LOGICAL PRINTER ADAPTER
007A ??                 C                  DB      ?               ;  THIRD LOGICAL PRINTER ADAPTER
007B ??                 C                  DB      ?               ;   RESERVED
007C ??                 C  @RS232_TIM_OUT  DB      ?               ; TIME OUT COUNTERS FOR RS232 RESPONSE
007D ??                 C                  DB      ?               ;  SECOND LOGICAL RS232 ADAPTER
007E ??                 C                  DB      ?               ;   RESERVED
007F ??                 C                  DB      ?               ;   RESERVED
                        C
                        C  ;--------------------------------------
                        C  ;       ADDITIONAL KEYBOARD DATA AREA  :
                        C  ;--------------------------------------
                        C
                        C                                          ; BUFFER LOCATION WITHIN SEGMENT 40H
0080 ????               C  @BUFFER_START   DW      ?               ; OFFSET OF KEYBOARD BUFFER START
0082 ????               C  @BUFFER_END     DW      ?               ; OFFSET OF END OF BUFFER
                        C
                        C  ;--------------------------------------
                        C  ;       EGA/PGA DISPLAY WORK AREA      :
                        C  ;--------------------------------------
                        C
0084 ??                 C  @ROWS           DB      ?               ; ROWS ON THE ACTIVE SCREEN (LESS 1)
0085 ????               C  @POINTS         DW      ?               ; BYTES PER CHARACTER
0087 ??                 C  @INFO           DB      ?               ; MODE OPTIONS
0088 ??                 C  @INFO_3         DB      ?               ; FEATURE BIT SWITCHES
0089 ??                 C                  DB      ?               ; RESERVED FOR DISPLAY ADAPTERS
008A ??                 C                  DB      ?               ; RESERVED FOR DISPLAY ADAPTERS
                        C
                        C  ;--------------------------------------
                        C  ;       ADDITIONAL MEDIA DATA          :
                        C  ;--------------------------------------
                        C
008B ??                 C  @LASTRATE       DB      ?               ; LAST DISKETTE DATA RATE SELECTED
008C ??                 C  @HF_STATUS      DB      ?               ; STATUS REGISTER
008D ??                 C  @HF_ERROR       DB      ?               ; ERROR REGISTER
008E ??                 C  @HF_INT_FLAG    DB      ?               ; FIXED DISK INTERRUPT FLAG
008F ??                 C  @HF_CNTRL       DB      ?               ; COMBO FIXED DISK/DISKETTE CARD BIT 0=1
0090 ??                 C  @DSK_STATE      DB      ?               ; DRIVE 0 MEDIA STATE
0091 ??                 C                  DB      ?               ; DRIVE 1 MEDIA STATE
0092 ??                 C                  DB      ?               ; DRIVE 0 OPERATION START STATE
0093 ??                 C                  DB      ?               ; DRIVE 1 OPERATION START STATE
0094 ??                 C  @DSK_TRK        DB      ?               ; DRIVE 0 PRESENT CYLINDER
0095 ??                 C                  DB      ?               ; DRIVE 1 PRESENT CYLINDER
                        C
                        C  ;--------------------------------------
                        C  ;       ADDITIONAL KEYBOARD FLAGS      :
                        C  ;--------------------------------------
                        C
0096 ??                 C  @KB_FLAG_3      DB      ?               ; KEYBOARD MODE STATE AND TYPE FLAGS
0097 ??                 C  @KB_FLAG_2      DB      ?               ; KEYBOARD LED FLAGS
                        C
                        C  ;--------------------------------------
                        C  ;       REAL TIME CLOCK DATA AREA      :
                        C  ;--------------------------------------
                        C
0098 ????               C  @USER_FLAG      DW      ?               ; OFFSET ADDRESS OF USERS WAIT FLAG
009A ????               C  @USER_FLAG_SEG  DW      ?               ; SEGMENT ADDRESS OF USER WAIT FLAG
009C ????               C  @RTC_LOW        DW      ?               ; LOW WORD OF USER WAIT FLAG
009E ????               C  @RTC_HIGH       DW      ?               ; HIGH WORD OF USER WAIT FLAG
00A0 ??                 C  @RTC_WAIT_FLAG  DB      ?               ; WAIT ACTIVE FLAG (01=BUSY, 80=POSTED)
                        C                                          ;                  (00=POST ACKNOWLEDGED)
                        C  ;--------------------------------------
                        C  ;       AREA FOR NETWORK ADAPTER       :
                        C  ;--------------------------------------
                        C
00A1   07 [             C  @NET            DB      7 DUP(?)        ; RESERVED FOR NETWORK ADAPTERS
          ??            C
               ]        C
                        C
                        C  ;--------------------------------------
                        C  ;       EGA/PGA PALETTE POINTER        :
                        C  ;--------------------------------------
                        C
00A8 ????????           C  @SAVE_PTR       DD      ?               ; POINTER TO EGA PARAMETER CONTROL BLOCK
                        C                                          ; RESERVED
                        C  ;--------------------------------------
                        C  ;       DATA AREA - PRINT SCREEN       :
                        C  ;--------------------------------------
                        C
0100                    C                  ORG     100H            ; ADDRESS= 0040:0100   (REF 0050:0000)
                        C
0100 ??                 C  @STATUS_BYTE    DB      ?               ; PRINT SCREEN STATUS BYTE
                        C                                          ;  00=READY/OK,  01=BUSY,  FF=ERROR
                        C
0101                    C  DATA            ENDS                    ; END OF BIOS DATA SEGMENT

                           .LIST
```

SECTION 5

```
                        PAGE
                C       INCLUDE POSTEQU.INC
                C       ;--------------------------------------
                C       ;       EQUATES USED BY POST AND BIOS   :
                C       ;--------------------------------------
                C
= 00FC          C       MODEL_BYTE      EQU     0FCH            ; SYSTEM MODEL BYTE
= 0002          C       SUB_MODEL_BYTE  EQU     002H            ; SYSTEM SUB-MODEL TYPE
= 0000          C       BIOS_LEVEL      EQU     000H            ; BIOS REVISION LEVEL
= F780          C       RATE_UPPER      EQU     0F780H          ; UPPER LIMIT + 13%
= F9FD          C       RATE_LOWER      EQU     0F9FDH          ; LOWER LIMIT - 20%
                C
                C       ;--------- 8042 KEYBOARD INTERFACE AND DIAGNOSTIC CONTROL REGISTERS ------------
= 0060          C       PORT_A          EQU     060H            ; 8042 KEYBOARD SCAN CODE/CONTROL PORT
= 0061          C       PORT_B          EQU     061H            ; PORT B READ/WRITE DIAGNOSTIC REGISTER
= 00F3          C       RAM_PAR_ON      EQU     11110011B       ; AND MASK FOR PARITY CHECKING ENABLE ON
= 000C          C       RAM_PAR_OFF     EQU     00001100B       ; OR MASK FOR PARITY CHECKING ENABLE OFF
= 00C0          C       PARITY_ERR      EQU     11000000B       ; R/W MEMORY - I/O CHANNEL PARITY ERROR
= 0001          C       GATE2           EQU     00000001B       ; TIMER 2 INPUT GATE CLOCK BIT
= 0002          C       SPK2            EQU     00000010B       ; SPEAKER OUTPUT DATA ENABLE BIT
= 0010          C       REFRESH_BIT     EQU     00010000B       ; REFRESH TEST BIT
= 0020          C       OUT2            EQU     00100000B       ; SPEAKER TIMER OUT2 INPUT BIT
= 0040          C       IO_CHECK        EQU     01000000B       ; I/O (MEMORY) CHECK OCCURRED BIT MASK
= 0080          C       PARITY_CHECK    EQU     10000000B       ; MEMORY PARITY CHECK OCCURRED BIT MASK
= 0064          C       STATUS_PORT     EQU     064H            ; 8042 STATUS PORT
= 0001          C       OUT_BUF_FULL    EQU     00000001B       ;   0 = +OUTPUT BUFFER FULL
= 0002          C       INPT_BUF_FULL   EQU     00000010B       ;   1 = +INPUT BUFFER FULL
= 0004          C       SYS_FLAG        EQU     00000100B       ;   2 = -SYSTEM FLAG -POST/-SELF TEST
= 0008          C       CMD_DATA        EQU     00001000B       ;   3 = -COMMAND/+DATA
= 0010          C       KYBD_INH        EQU     00010000B       ;   4 = +KEYBOARD INHIBITED
= 0020          C       TRANS_TMOUT     EQU     00100000B       ;   5 = +TRANSMIT TIMEOUT
= 0040          C       RCV_TMOUT       EQU     01000000B       ;   6 = +RECEIVE TIME OUT
= 0080          C       PARITY_EVEN     EQU     10000000B       ;   7 = +PARITY IS EVEN
                C
                C       ;--------- 8042 INPUT PORT BIT DEFINITION SAVED IN @MFG_TST -------------------
= 0008          C       BASE_MEM8       EQU     00001000B       ; BASE PLANAR R/W MEMORY EXTENSION 640/X
= 0010          C       BASE_MEM        EQU     00010000B       ; BASE PLANAR R/W MEMORY SIZE 256/512
= 0020          C       MFG_LOOP        EQU     00100000B       ; LOOP POST JUMPER BIT FOR MANUFACTURING
= 0040          C       DSP_JMP         EQU     01000000B       ; DISPLAY TYPE SWITCH JUMPER BIT
= 0080          C       KEY_BD_INHIB    EQU     10000000B       ; KEYBOARD INHIBIT SWITCH BIT
                C
                C       ;--------- 8042 COMMANDS -----------------------------------------------------
= 0060          C       WRITE_8042_LOC  EQU     060H            ; WRITE 8042 COMMAND BYTE
= 00AA          C       SELF_TEST       EQU     0AAH            ; 8042 SELF TEST
= 00AB          C       INTR_FACE_CK    EQU     0ABH            ; CHECK 8042 INTERFACE COMMAND
= 00AD          C       DIS_KBD         EQU     0ADH            ; DISABLE KEYBOARD COMMAND
= 00AE          C       ENA_KBD         EQU     0AEH            ; ENABLE KEYBOARD COMMAND
= 00C0          C       READ_8042_INPUT EQU     0C0H            ; READ 8042 INPUT PORT
= 00DD          C       DISABLE_BIT20   EQU     0DDH            ; DISABLE ADDRESS LINE BIT 20
= 00DF          C       ENABLE_BIT20    EQU     0DFH            ; ENABLE ADDRESS LINE BIT 20
= 00E0          C       KYBD_CLK_DATA   EQU     0E0H            ; GET KEYBOARD CLOCK AND DATA COMMAND
= 00FE          C       SHUT_CMD        EQU     0FEH            ; CAUSE A SHUTDOWN COMMAND
= 0001          C       KYBD_CLK        EQU     001H            ; KEYBOARD CLOCK BIT 0
                C
                C       ;---------- KEYBOARD/LED COMMANDS ----------------------------------------------
= 00FF          C       KB_RESET        EQU     0FFH            ; SELF DIAGNOSTIC COMMAND
= 00FE          C       KB_RESEND       EQU     0FEH            ; RESEND COMMAND
= 00FA          C       KB_MAKE_BREAK   EQU     0FAH            ; TYPAMATIC COMMAND
= 00F4          C       KB_ENABLE       EQU     0F4H            ; KEYBOARD ENABLE
= 00F3          C       KB_TYPA_RD      EQU     0F3H            ; TYPAMATIC RATE/DELAY COMMAND
= 00F2          C       KB_READ_ID      EQU     0F2H            ; READ KEYBOARD ID COMMAND
= 00EE          C       KB_ECHO         EQU     0EEH            ; ECHO COMMAND
= 00ED          C       LED_CMD         EQU     0EDH            ; LED WRITE COMMAND
                C
                C       ;---------- 8042 KEYBOARD RESPONSE ---------------------------------------------
= 00FF          C       KB_OVER_RUN     EQU     0FFH            ; OVER RUN SCAN CODE
= 00FE          C       KB_RESEND       EQU     0FEH            ; RESEND REQUEST
= 00FA          C       KB_ACK          EQU     0FAH            ; ACKNOWLEDGE FROM TRANSMISSION
= 00F0          C       KB_BREAK        EQU     0F0H            ; KEYBOARD BREAK CODE
= 00AA          C       KB_OK           EQU     0AAH            ; RESPONSE FROM SELF DIAGNOSTIC
                C
                C       ;---------- KEYBOARD SCAN CODES ------------------------------------------------
= 0045          C       NUM_KEY         EQU     69              ; SCAN CODE FOR  NUMBER LOCK KEY
= 0046          C       SCROLL_KEY      EQU     70              ; SCAN CODE FOR  SCROLL LOCK KEY
= 0038          C       ALT_KEY         EQU     56              ; SCAN CODE FOR  ALTERNATE SHIFT KEY
= 001D          C       CTL_KEY         EQU     29              ; SCAN CODE FOR  CONTROL KEY
= 003A          C       CAPS_KEY        EQU     58              ; SCAN CODE FOR  SHIFT LOCK KEY
= 0053          C       DEL_KEY         EQU     83              ; SCAN CODE FOR  DELETE KEY
= 0052          C       INS_KEY         EQU     82              ; SCAN CODE FOR  INSERT KEY
= 002A          C       LEFT_KEY        EQU     42              ; SCAN CODE FOR  LEFT SHIFT
= 0036          C       RIGHT_KEY       EQU     54              ; SCAN CODE FOR  RIGHT SHIFT
= 0054          C       SYS_KEY         EQU     84              ; SCAN CODE FOR  SYSTEM KEY
                C
                C       ;---------- ENHANCED KEYBOARD SCAN CODES ---------------------------------------
= 00AB          C       ID_1            EQU     0ABH            ; 1ST ID CHARACTER FOR KBX
= 0041          C       ID_2            EQU     041H            ; 2ND ID CHARACTER FOR KBX
= 0054          C       ID_2A           EQU     054H            ; ALTERNATE 2ND ID CHAR FOR KBX
= 0057          C       F11_M           EQU     87              ; F11 KEY MAKE
= 0058          C       F12_M           EQU     88              ; F12 KEY MAKE
= 00E0          C       MC_E0           EQU     224             ; GENERAL MARKER CODE
= 00E1          C       MC_E1           EQU     225             ; PAUSE KEY MARKER CODE
```

```
                        C  PAGE
                        C  ;---------- FLAG EQUATES WITHIN  @KB_FLAG -----------------------------------------
= 0001                  C  RIGHT_SHIFT      EQU     00000001B       ; RIGHT SHIFT KEY DEPRESSED
= 0002                  C  LEFT_SHIFT       EQU     00000010B       ; LEFT SHIFT KEY DEPRESSED
= 0004                  C  CTL_SHIFT        EQU     00000100B       ; CONTROL SHIFT KEY DEPRESSED
= 0008                  C  ALT_SHIFT        EQU     00001000B       ; ALTERNATE SHIFT KEY DEPRESSED
= 0010                  C  SCROLL_STATE     EQU     00010000B       ; SCROLL LOCK STATE IS ACTIVE
= 0020                  C  NUM_STATE        EQU     00100000B       ; NUM LOCK STATE IS ACTIVE
= 0040                  C  CAPS_STATE       EQU     01000000B       ; CAPS LOCK STATE IS ACTIVE
= 0080                  C  INS_STATE        EQU     10000000B       ; INSERT STATE IS ACTIVE
                        C
                        C  ;---------- FLAG EQUATES WITHIN  @KB_FLAG_1 ---------------------------------------
= 0001                  C  L_CTL_SHIFT      EQU     00000001B       ; LEFT CTL KEY DOWN
= 0002                  C  L_ALT_SHIFT      EQU     00000010B       ; LEFT ALT KEY DOWN
= 0004                  C  SYS_SHIFT        EQU     00000100B       ; SYSTEM KEY DEPRESSED AND HELD
= 0008                  C  HOLD_STATE       EQU     00001000B       ; SUSPEND KEY HAS BEEN TOGGLED
= 0010                  C  SCROLL_SHIFT     EQU     00010000B       ; SCROLL LOCK KEY IS DEPRESSED
= 0020                  C  NUM_SHIFT        EQU     00100000B       ; NUM LOCK KEY IS DEPRESSED
= 0040                  C  CAPS_SHIFT       EQU     01000000B       ; CAPS LOCK KEY IS DEPRESSED
= 0080                  C  INS_SHIFT        EQU     10000000B       ; INSERT KEY IS DEPRESSED
                        C
                        C  ;---------- FLAGS EQUATES WITHIN  @KB_FLAG_2 ---------------------------------------
= 0007                  C  KB_LEDS          EQU     00000111B       ; KEYBOARD LED STATE BITS
                        C  ;                                        ; SCROLL LOCK INDICATOR
                        C  ;                                        ; NUM LOCK INDICATOR
                        C  ;                                        ; CAPS LOCK INDICATOR
                        C  ;                EQU     00001000B       ; RESERVED (MUST BE ZERO)
= 0010                  C  KB_FA            EQU     00010000B       ; ACKNOWLEDGMENT RECEIVED
= 0020                  C  KB_FE            EQU     00100000B       ; RESEND RECEIVED FLAG
= 0040                  C  KB_PR_LED        EQU     01000000B       ; MODE INDICATOR UPDATE
= 0080                  C  KB_ERR           EQU     10000000B       ; KEYBOARD TRANSMIT ERROR FLAG
                        C
                        C  ;---------- FLAGS EQUATES WITHIN  @KB_FLAG_3 ---------------------------------------
= 0001                  C  LC_E1            EQU     00000001B       ; LAST CODE WAS THE E1 HIDDEN CODE
= 0002                  C  LC_E0            EQU     00000010B       ; LAST CODE WAS THE E0 HIDDEN CODE
= 0004                  C  R_CTL_SHIFT      EQU     00000100B       ; RIGHT CTL KEY DOWN
= 0008                  C  R_ALT_SHIFT      EQU     00001000B       ; RIGHT ALT KEY DOWN
= 0008                  C  GRAPH_ON         EQU     00001000B       ; ALT GRAPHICS KEY DOWN (WT ONLY)
= 0010                  C  KBX              EQU     00010000B       ; ENHANCED KEYBOARD INSTALLED
= 0020                  C  SET_NUM_LK       EQU     00100000B       ; FORCE NUM LOCK IF READ ID AND KBX
= 0040                  C  LC_AB            EQU     01000000B       ; LAST CHARACTER WAS FIRST ID CHARACTER
= 0080                  C  RD_ID            EQU     10000000B       ; DOING A READ ID (MUST BE BIT0)
```

```
                        C  PAGE
                        C  ;--------------------------------------
                        C  ;       CMOS EQUATES FOR THIS SYSTEM   ;
                        C  ;--------------------------------------------------------------------------------
0070                    C  CMOS_PORT        EQU     070H            ; I/O ADDRESS OF CMOS ADDRESS PORT
0071                    C  CMOS_DATA        EQU     071H            ; I/O ADDRESS OF CMOS DATA PORT
0080                    C  NMI              EQU     10000000B       ; DISABLE NMI INTERRUPTS MASK -
                        C                                           ;  HIGH BIT OF CMOS LOCATION ADDRESS
                        C
                        C  ;---------- CMOS TABLE LOCATION ADDRESS'S ## ------------------------------------
0000                    C  CMOS_SECONDS     EQU     000H            ; SECONDS
0001                    C  CMOS_SEC_ALARM   EQU     001H            ; SECONDS ALARM  ## NOTE:  ALL LOCATIONS
0002                    C  CMOS_MINUTES     EQU     002H            ; MINUTES                | IN THE CMOS AREA
0003                    C  CMOS_MIN_ALARM   EQU     003H            ; MINUTES ALARM          | ARE IBM USE ONLY
0004                    C  CMOS_HOURS       EQU     004H            ; HOURS                  | AND  SUBJECT  TO
0005                    C  CMOS_HR_ALARM    EQU     005H            ; HOURS ALARM            | CHANGE.  ONLY THE
0006                    C  CMOS_DAY_WEEK    EQU     006H            ; DAY OF THE WEEK        | POST & BIOS CODE
0007                    C  CMOS_DAY_MONTH   EQU     007H            ; DAY OF THE MONTH       | SHOULD  DIRECTLY
0008                    C  CMOS_MONTH       EQU     008H            ; MONTH                  | ACCESS LOCATIONS
0009                    C  CMOS_YEAR        EQU     009H            ; YEAR (TWO DIGITS)      | IN CMOS STORAGE.
000A                    C  CMOS_REG_A       EQU     00AH            ; STATUS REGISTER A      *-----------------
000B                    C  CMOS_REG_B       EQU     00BH            ; STATUS REGISTER B  ALARM
000C                    C  CMOS_REG_C       EQU     00CH            ; STATUS REGISTER C  FLAGS
000D                    C  CMOS_REG_D       EQU     00DH            ; STATUS REGISTER D  BATTERY
000E                    C  CMOS_DIAG        EQU     00EH            ; POST DIAGNOSTIC STATUS RESULTS BYTE
000F                    C  CMOS_SHUT_DOWN   EQU     00FH            ; SHUTDOWN STATUS COMMAND BYTE
0010                    C  CMOS_DISKETTE    EQU     010H            ; DISKETTE DRIVE TYPE BYTE              ;
                        C  ;                EQU     011H            ; - RESERVED                            ;C
0012                    C  CMOS_DISK        EQU     012H            ; FIXED DISK TYPE BYTE                  ;H
                        C  ;                EQU     013H            ; - RESERVED                            ;E
0014                    C  CMOS_EQUIP       EQU     014H            ; EQUIPMENT WORD LOW BYTE               ;C
0015                    C  CMOS_B_M_S_LO    EQU     015H            ; BASE MEMORY SIZE - LOW BYTE (X1024)   ;K
0016                    C  CMOS_B_M_S_HI    EQU     016H            ; BASE MEMORY SIZE - HIGH BYTE          ;S
0017                    C  CMOS_E_M_S_LO    EQU     017H            ; EXPANSION MEMORY SIZE - LOW BYTE      ;U
0018                    C  CMOS_E_M_S_HI    EQU     018H            ; EXPANSION MEMORY SIZE - HIGH BYTE     ;M
0019                    C  CMOS_DTSK_T      EQU     019H            ; FIXED DISK TYPE - DRIVE C EXTENSION   ;E
001A                    C  CMOS_DISK_2      EQU     01AH            ; FIXED DISK TYPE - DRIVE D EXTENSION   ;D
                        C  ;                EQU     01BH            ; - 1BH THROUGH 2DH - RESERVED          ;
002E                    C  CMOS_CKSUM_HI    EQU     02EH            ; CMOS CHECKSUM - HIGH BYTE             ;*
002F                    C  CMOS_CKSUM_LO    EQU     02FH            ; CMOS CHECKSUM - LOW BYTE              ;*
0030                    C  CMOS_U_M_S_LO    EQU     030H            ; USABLE MEMORY ABOVE 1 MEG - LOW BYTE
0031                    C  CMOS_U_M_S_HI    EQU     031H            ; USABLE MEMORY ABOVE 1 MEG - HIGH BYTE
0032                    C  CMOS_CENTURY     EQU     032H            ; DATE CENTURY BYTE (BCD)
0033                    C  CMOS_INFO128     EQU     033H            ; 128KB INFORMATION STATUS FLAG BYTE
                        C  ;                EQU     034H            ; - 34H THROUGH 3FH - RESERVED
                        C
                        C  ;---------- CMOS DIAGNOSTIC STATUS ERROR FLAGS WITHIN CMOS_DIAG ----------------
0004                    C  CMOS_CLK_FAIL    EQU     00000100B       ; CMOS CLOCK NOT UPDATING OR NOT VALID
0008                    C  HF_FAIL          EQU     00001000B       ; FIXED DISK FAILURE ON INITIALIZATION
0010                    C  W_MEM_SIZE       EQU     00010000B       ; MEMORY SIZE NOT EQUAL TO CONFIGURATION
0020                    C  BAD_CONFIG       EQU     00100000B       ; MINIMUM CONFIG USED INSTEAD OF CMOS
0040                    C  BAD_CKSUM        EQU     01000000B       ; CHECKSUM ERROR
0080                    C  BAD_BAT          EQU     10000000B       ; DEAD BATTERY - CMOS LOST POWER
                        C
                        C  ;---------- CMOS INFORMATION FLAGS ---------------------------------------------
0080                    C  M640K            EQU     10000000B       ; 512K -> 640K OPTION INSTALLED  (128K)
                        C  ;                EQU     01000000B       ; FLAG USED BY CMOS SETUP UTILITY
                        C
                        C
                        C
                        C  ;---------- DISKETTE EQUATES ---------------------------------------------------
0001                    C  DUAL             EQU     00000001B       ; MASK FOR COMBO/DSP ADAPTER
0080                    C  INT_FLAG         EQU     10000000B       ; INTERRUPT OCCURRENCE FLAG
0080                    C  DSK_CHG          EQU     10000000B       ; DISKETTE CHANGE FLAG MASK BIT
0010                    C  DETERMINED       EQU     00010000B       ; SET STATE DETERMINED IN STATE BITS
0010                    C  HOME             EQU     00010000B       ; TRACK 0 MASK
0004                    C  SENSE_DRV_ST     EQU     00000100B       ; SENSE DRIVE STATUS COMMAND
0030                    C  TRK_SLAP         EQU     030H            ; CRASH STOP (48 TPI DRIVES)
000A                    C  QUIET_SEEK       EQU     00AH            ; SEEK TO TRACK 10
0002                    C  MAX_DRV          EQU     2               ; MAX NUMBER OF DRIVES
000F                    C  HD12_SETTLE      EQU     15              ; 1.2 M HEAD SETTLE TIME
0014                    C  HD320_SETTLE     EQU     20              ; 320 K HEAD SETTLE TIME
0025                    C  MOTOR_WAIT       EQU     37              ; 2 SECONDS OF COUNTS FOR MOTOR TURN OFF
                        C
                        C
                        C  ;---------- DISKETTE ERRORS ----------------------------------------------------
0080                    C  TIME_OUT         EQU     080H            ; ATTACHMENT FAILED TO RESPOND
0040                    C  BAD_SEEK         EQU     040H            ; SEEK OPERATION FAILED
0020                    C  BAD_NEC          EQU     020H            ; DISKETTE CONTROLLER HAS FAILED
0010                    C  BAD_CRC          EQU     010H            ; BAD CRC ON DISKETTE READ
000C                    C  MED_NOT_FND      EQU     00CH            ; MEDIA TYPE NOT FOUND
0009                    C  DMA_BOUNDARY     EQU     009H            ; ATTEMPT TO DMA ACROSS 64K BOUNDARY
0008                    C  BAD_DMA          EQU     008H            ; DMA OVERRUN ON OPERATION
0006                    C  MEDIA_CHANGE     EQU     006H            ; MEDIA REMOVED ON DUAL ATTACH CARD
0004                    C  RECORD_NOT_FND   EQU     004H            ; REQUESTED SECTOR NOT FOUND
0003                    C  WRITE_PROTECT    EQU     003H            ; WRITE ATTEMPTED ON WRITE PROTECT DISK
0002                    C  BAD_ADDR_MARK    EQU     002H            ; ADDRESS MARK NOT FOUND
0001                    C  BAD_CMD          EQU     001H            ; BAD COMMAND PASSED TO DISKETTE I/O
                        C
                        C  ;---------- DISK CHANGE LINE EQUATES -------------------------------------------
0001                    C  NOCHGLN          EQU     001H            ; NO DISK CHANGE LINE AVAILABLE
0002                    C  CHGLN            EQU     002H            ; DISK CHANGE LINE AVAILABLE
                        C
                        C  ;---------- MEDIA/DRIVE STATE INDICATORS ----------------------------------------
0001                    C  TRK_CAPA         EQU     00000001B       ; 80 TRACK CAPABILITY
0002                    C  FMT_CAPA         EQU     00000010B       ; MULTIPLE FORMAT CAPABILITY (1.2M)
0004                    C  DRV_DET          EQU     00000100B       ; DRIVE DETERMINED
0010                    C  MED_DET          EQU     00010000B       ; MEDIA DETERMINED BIT
0020                    C  DBL_STEP         EQU     00100000B       ; DOUBLE STEP BIT
00C0                    C  RATE_MSK         EQU     11000000B       ; MASK FOR CLEARING ALL BUT RATE
0000                    C  RATE_500         EQU     00000000B       ; 500 KBS DATA RATE
0040                    C  RATE_300         EQU     01000000B       ; 300 KBS DATA RATE
0080                    C  RATE_250         EQU     10000000B       ; 250 KBS DATA RATE
000C                    C  STRT_MSK         EQU     00001100B       ; OPERATION START RATE MASK
00C0                    C  SEND_MSK         EQU     11000000B       ; MASK FOR SEND RATE BITS
                        C
                        C
                        C  ;---------- MEDIA/DRIVE STATE INDICATORS COMPATIBILITY --------------------------
0000                    C  M3D3U            EQU     00000000B       ; 360 MEDIA/DRIVE NOT ESTABLISHED
0001                    C  M3D1U            EQU     00000001B       ; 360 MEDIA,1.2DRIVE NOT ESTABLISHED
0002                    C  M1D1U            EQU     00000010B       ; 1.2 MEDIA/DRIVE NOT ESTABLISHED
0007                    C  MED_UNK          EQU     00000111B       ; NONE OF THE ABOVE
```

```
                        C  PAGE
                        C  ;---------- INTERRUPT EQUATES ----------------------------------------------------------
= 0020                  C  EOI             EQU     020H            ; END OF INTERRUPT COMMAND TO 8259
= 0020                  C  INTA00          EQU     020H            ; 8259 PORT
= 0021                  C  INTA01          EQU     021H            ; 8259 PORT
= 00A0                  C  INTB00          EQU     0A0H            ; 2ND 8259
= 00A1                  C  INTB01          EQU     0A1H            ;
= 0070                  C  INT_TYPE        EQU     070H            ; START OF 8259 INTERRUPT TABLE LOCATION
= 0010                  C  INT_VIDEO       EQU     010H            ; VIDEO VECTOR
                        C  ;--------------------------------------------------------------------------------
= 0008                  C  DMA08           EQU     008H            ; DMA STATUS REGISTER PORT ADDRESS
= 0000                  C  DMA             EQU     000H            ; DMA CH.0 ADDRESS REGISTER PORT ADDRESS
= 00D0                  C  DMA18           EQU     0D0H            ; 2ND DMA STATUS PORT ADDRESS
= 00C0                  C  DMA1            EQU     0C0H            ; 2ND DMA CH.0 ADDRESS REGISTER ADDRESS
                        C  ;--------------------------------------------------------------------------------
= 0040                  C  TIMER           EQU     040H            ; 8254 TIMER - BASE ADDRESS
                        C
                        C  ;--------- MANUFACTURING PORT ----------------------------------------------------
= 0080                  C  MFG_PORT        EQU     80H             ; MANUFACTURING AND POST CHECKPOINT PORT
                        C                                          ;  DMA CHANNEL 0 PAGE REGISTER ADDRESS
                        C
                        C  ;--------- MANUFACTURING BIT DEFINITION FOR @MFG_ERR_FLAG+1 ----------------------
= 0001                  C  MEM_FAIL        EQU     00000001B       ; STORAGE TEST FAILED          (ERROR 20X)
= 0002                  C  PRO_FAIL        EQU     00000010B       ; VIRTUAL MODE TEST FAILED     (ERROR 104)
= 0004                  C  LMCS_FAIL       EQU     00000100B       ; LOW MEG CHIP SELECT FAILED   (ERROR 109)
= 0008                  C  KYCLK_FAIL      EQU     00001000B       ; KEYBOARD CLOCK TEST FAILED   (ERROR 304)
= 0010                  C  KY_SYS_FAIL     EQU     00010000B       ; KEYBOARD OR SYSTEM FAILED    (ERROR 303)
= 0020                  C  KYBD_FAIL       EQU     00100000B       ; KEYBOARD FAILED              (ERROR 301)
= 0040                  C  DSK_FAIL        EQU     01000000B       ; DISKETTE TEST FAILED         (ERROR 601)
= 0080                  C  KEY_FAIL        EQU     10000000B       ; KEYBOARD LOCKED              (ERROR 302)
                        C
                        C  ;--------------------------------------------------------------------------------
= 0081                  C  DMA_PAGE        EQU     081H            ; START OF DMA PAGE REGISTERS
= 008F                  C  LAST_DMA_PAGE   EQU     08FH            ; LAST DMA PAGE REGISTER
                        C
                        C  ;--------------------------------------------------------------------------------
= 00F0                  C  X287            EQU     0F0H            ; MATH COPROCESSOR CONTROL PORT
                        C
                        C  ;--------------------------------------------------------------------------------
= 0000                  C  POST_SS         EQU     00000H          ; POST STACK SEGMENT
= 8000                  C  POST_SP         EQU     08000H          ; POST STACK POINTER
                        C
                        C  ;--------------------------------------------------------------------------------
= 000D                  C  CR              EQU     000DH           ; CARRIAGE RETURN CHARACTER
= 000A                  C  LF              EQU     000AH           ; LINE FEED CHARACTER
= 0008                  C  RVRT            EQU     00001000B       ; VIDEO VERTICAL RETRACE BIT
= 0001                  C  RHRZ            EQU     00000001B       ; VIDEO HORIZONTAL RETRACE BIT
= 0100                  C  H               EQU     256             ; HIGH BYTE FACTOR (X 100H)
= 0101                  C  X               EQU     H+1             ; HIGH AND LOW BYTE FACTOR (X 101H)

                            .LIST
```

```
                        PAGE
                        INCLUDE SYSDATA.INC
                        ;--------------------------------------------------------------------------------
                        ;       PROTECTED MODE EQUATES FOR POST TESTS AND BIOS ROUTINES                  :
                        ;--------------------------------------------------------------------------------

                        ;-----          LENGTH EQUATES FOR PROTECTED MODE TESTS

= 0300                  SDA_LEN         EQU     00300H          ; SYSTEM DATA AREA LENGTH
= 0800                  SYS_IDT_LEN     EQU     256*8           ; 256 SYSTEM IDT ENTRIES, 8 BYTES EACH
= 0008                  GDT_LEN         EQU     TYPE GDT_DEF    ; GDT STRUCTURE LENGTH
= 0008                  DESC_LEN        EQU     TYPE DATA_DESC  ; LENGTH OF A DESCRIPTOR
= 1000                  MCRT_SIZE       EQU     4*1024          ; MONOCHROME CRT SIZE
= 4000                  CCRT_SIZE       EQU     16*1024         ; COMPATIBLE COLOR CRT SIZE
= FFFF                  ECCRT_SIZE      EQU     0FFFFH          ; SIZE OF EACH PORTION OF THE ENHANCED
= FFFF                  MAX_SEG_LEN     EQU     0FFFFH          ; MAXIMUM SEGMENT LENGTH = 64K
= 0000                  NULL_SEG_LEN    EQU     00000H          ; NULL SEGMENT LENGTH = 0

                        ;-----          LOCATION EQUATES FOR PROTECTED MODE TESTS

= D0A0                  SYS_IDT_LOC     EQU     0D0A0H          ; THE SYSTEM IDT IS AT THE BOTTOM
= 0400                  SDA_LOC         EQU     00400H          ; SAME AS REAL
= D8A0                  GDT_LOC         EQU     (SYS_IDT_LOC + SYS_IDT_LEN)
= 0000                  MCRT@_LO        EQU     0000H           ; MONOCHROME CRT ADDRESS
= 000B                  MCRT@_HI        EQU     0BH             ; (0B0000H)
= 8000                  CCRT@_LO        EQU     8000H           ; COMPATIBLE COLOR CRT ADDRESS
= 000B                  CCRT@_HI        EQU     0BH             ; (0B8000H)
= 0000                  ECCRT@_LO_LO    EQU     0000H
= 000A                  ECCRT@_LO_HI    EQU     0AH             ; (0A0000H)
= 0000                  ECCRT@_HI_LO    EQU     0000H
= 000B                  ECCRT@_HI_HI    EQU     0BH             ; (0B0000H)
= 0000                  CSEG@_LO        EQU     0000H           ; CODE SEGMENT POST/BIOS
= 000F                  CSEG@_HI        EQU     0FH             ; (0F0000H) FOR TESTS
= 0000                  NSEG@_LO        EQU     0000H           ; ABS0
= 0000                  NSEG@_HI        EQU     00H

                        ;-----  DEFINITIONS FOR ACCESS RIGHTS BYTES

= 00F3                  CPL3_DATA_ACCESS        EQU     11110011B       ; PRESENT
                                                                        ; DPL = 3
                                                                        ; CODE/DATA SEGMENT
                                                                        ; NOT EXECUTABLE
                                                                        ; GROW-UP (OFFSET <= LIMIT)
                                                                        ; WRITABLE
                                                                        ; ACCESSED
= 0093                  CPL0_DATA_ACCESS        EQU     10010011B       ; DPL = 0
= 009B                  CPL0_CODE_ACCESS        EQU     10011011B       ; CPL 0 - NON-CONFORMING
= 00E2                  LDT_DESC                EQU     11100010B
= 0081                  FREE_TSS                EQU     10000001B
= 0086                  INT_GATE                EQU     10000110B
= 0087                  TRAP_GATE               EQU     10000111B

= 0001                  VIRTUAL_ENABLE          EQU     0000000000000001B       ; PROTECTED MODE ENABLE

                        ;-----  THE GLOBAL DESCRIPTOR TABLE DEFINITION FOR POWER ON SELF TESTS

                        GDT_DEF         STRUC
0000 ????????????????   GDT_PTR          DQ      ?               ; UNUSED ENTRY
0008 ????????????????   SYS_IDT_PTR      DQ      ?               ; THIS ENTRY POINTS TO THIS TABLE
0010 ????????????????   RSDA_PTR         DQ      ?               ; POST INTERRUPT DESCRIPTOR TABLE
0018 ????????????????   C_BWCRT_PTR      DQ      ?               ; THE REAL SYSTEM DATA AREA FOR POST
0020 ????????????????   C_CCRT_PTR       DQ      ?               ; COMPATIBLE BW CRT FOR POST
0028 ????????????????   E_CCRT_PTR       DQ      ?               ; COMPATIBLE COLOR CRT FOR POST
0030 ????????????????   E_CCRT_PTR2      DQ      ?               ; ENHANCED COLOR GRAPHICS CRT (16 BYTES)
0038 ????????????????   SYS_ROM_CS       DQ      ?               ; CS - POST IDT, ROM RESIDENT
0040 ????????????????   ES_TEMP          DQ      ?               ; DYNAMIC POINTER FOR ES
0048 ????????????????   CS_TEMP          DQ      ?               ; DYNAMIC POINTER FOR CS
0050 ????????????????   SS_TEMP          DQ      ?               ; DYNAMIC POINTER FOR SS
0058 ????????????????   DS_TEMP          DQ      ?               ; DYNAMIC POINTER FOR DS
0060 ????????????????   POST_TR          DQ      ?               ; TR VALUE FOR THIS MACHINE'S TSS
0068 ????????????????   POST_TSS_PTR     DQ      ?
0070 ????????????????   POST_LDTR        DQ      ?               ; LDTR VALUE FOR THIS MACHINE'S LDT
0078 ????????????????   POST_LDT_PTR     DQ      ?
0080 ????????????????   GDT_DEF         ENDS
0088

                        ;-----  SEGMENT DESCRIPTOR TABLE ENTRY STRUCTURE

                        DATA_DESC       STRUC
0000 ????               SEG_LIMIT       DW      ?               ; SEGMENT LIMIT (1 - 65535 BYTES)
0002 ????               BASE_LO_WORD    DW      ?               ; 24 BIT SEGMENT PHYSICAL
0004 ??                 BASE_HI_BYTE    DB      ?               ;       ADDRESS (0 - (16M-1))
0005 ??                 DATA_ACC_RIGHTS DB      ?               ; ACCESS RIGHTS BYTE
0006 ????               DATA_RESERVED   DW      ?               ; RESERVED - MUST BE 0000 FOR THE 80286
0008                    DATA_DESC       ENDS

                        ;-----  GATE DESCRIPTOR TABLE ENTRY STRUCTURE

                        GATE_DESC       STRUC
0000 ????               ENTRY_POINT     DW      ?               ; DESTINATION ROUTINE ENTRY POINT
0002 ????               CS_SELECTOR     DW      ?               ; SELECTOR FOR DESTINATION SEGMENT
0004 ??                 WORD_COUNT      DB      ?               ; NUMBER OF WORDS TO COPY FROM STACK
0005 ??                 GATE_ACC_RIGHTS DB      ?               ; ACCESS RIGHTS BYTE
0006 ????               GATE_RESERVED   DW      ?               ; RESERVED - MUST BE 0000 FOR THE 80286
0008                    GATE_DESC       ENDS

                        .LIST
```

```
                        PAGE
0000                    CODE    SEGMENT WORD PUBLIC

                                PUBLIC  C8042
                                PUBLIC  OBF_42
                                PUBLIC  POST1
                                PUBLIC  START_1

                                EXTRN   CMOS_READ:NEAR
                                EXTRN   CMOS_WRITE:NEAR
                                EXTRN   CONFIG_BAD:NEAR
                                EXTRN   D11:NEAR
                                EXTRN   DDS:NEAR
                                EXTRN   DUMMY_RETURN:NEAR
                                EXTRN   ERR_BEEP:NEAR
                                EXTRN   GATE_A20:NEAR
                                EXTRN   KBD_RESET:NEAR
                                EXTRN   NMI_INT:NEAR
                                EXTRN   POST2:NEAR
                                EXTRN   PRINT_SCREEN:NEAR
                                EXTRN   PROC_SHUTDOWN:NEAR
                                EXTRN   ROM_CHECK:NEAR
                                EXTRN   SHUT2:NEAR
                                EXTRN   SHUT3:NEAR
                                EXTRN   SHUT4:NEAR
                                EXTRN   SHUT6:NEAR
                                EXTRN   SHUT7:NEAR
                                EXTRN   SHUT9:NEAR
                                EXTRN   SLAVE_VECTOR_TABLE:NEAR
                                EXTRN   STGTST_CNT:NEAR
                                EXTRN   SYSINIT1:NEAR
                                EXTRN   VECTOR_TABLE:NEAR
                                EXTRN   VIDEO_PARMS:BYTE

                                ASSUME  CS:CODE,DS:NOTHING,ES:NOTHING,SS:NOTHING

0000                    POST1   PROC    NEAR

= 0000                  BEGIN   EQU     $
0000 37 38 58 37 34 36          DB      '78X7462COPR. IBM CORP. 1981,1986  '           ;COPYRIGHT NOTICE
     32 43 4F 50 52 2E
     20 49 42 4D 20 43
     4F 52 50 2E 20 31
     39 38 31 2C 31 39
     38 36 20 20

                                EVEN                                                    ;EVEN BOUNDARY
                                ;        7 8 X 7 4 6 2   C O P R .   I B M   1 9 8 6   ;EVEN MODULE
                                ;         7 8 X 7 4 6 3   C O P R .   I B M   1 9 8 6  ;ODD  MODULE
0022 37 37 38 38 58 58          DB      '7788XX77446623  CCOOPPRR..  IIBBMM  II998866' ;COPYRIGHT NOTICE
     37 37 34 34 36 36
     32 33 20 20 43 43
     4F 4F 50 50 52 52
     2E 2E 20 20 49 49
     42 42 4D 4D 20 20
     31 31 39 39 38 38
     36 36
004E 20 20                      DB      '  '                                            ;PAD

                        ;-----------------------------------------------
                        ;       INITIAL RELIABILITY TESTS -- (POST1)    :
                        ;-----------------------------------------------

                        ;---------------------------------------
                        ; TEST.01                              :
                        ;       80286 PROCESSOR TEST (REAL MODE) :
                        ; DESCRIPTION                         :
                        ;       VERIFY FLAGS, REGISTERS       :
                        ;       AND CONDITIONAL JUMPS.        :
                        ;---------------------------------------

                                ASSUME  DS:DATA

0050                    START_1:
0050 FA                         CLI                             ; DISABLE INTERRUPTS
0051 B8 D58D                    MOV     AX,0D500H+CMOS_REG_D+NMI; FLAG MASK IN (AH) AND NMI MASK IN (AL)
0054 E6 70                      OUT     CMOS_PORT,AL            ; DISABLE NMI INTERRUPTS, LATCH STANDBY
0056 9E                         SAHF                            ; SET "SF" "ZF" "AF" "PF" "CF" FLAGS ON
0057 73 27                      JNC     ERR02                   ; GO TO ERROR ROUTINE IF "CF" NOT SET
0059 75 25                      JNZ     ERR02                   ; GO TO ERROR ROUTINE IF "ZF" NOT SET
005B 7B 23                      JNP     ERR02                   ; GO TO ERROR ROUTINE IF "PF" NOT SET
005D 79 21                      JNS     ERR02                   ; GO TO ERROR ROUTINE IF "SF" NOT SET
005F 9F                         LAHF                            ; LOAD FLAG IMAGE TO (AH)
0060 B1 05                      MOV     CL,5                    ; LOAD COUNT REGISTER WITH SHIFT COUNT
0062 D2 EC                      SHR     AH,CL                   ; SHIFT "AF" INTO CARRY BIT POSITION
0064 73 1A                      JNC     ERR02                   ; GO TO ERROR ROUTINE IF "AF" NOT SET
0066 B0 40                      MOV     AL,40H                  ; SET THE "OF" FLAG ON
0068 D0 E0                      SHL     AL,1                    ; SETUP FOR TESTING
006A 71 14                      JNO     ERR02                   ; GO TO ERROR ROUTINE IF "OF" NOT SET
006C 32 E4                      XOR     AH,AH                   ; SET (AH) = 0
006E 9E                         SAHF                            ; CLEAR "SF", "CF", "ZF", AND "PF"
006F 76 0F                      JBE     ERR02                   ; GO TO ERROR ROUTINE IF "CF" ON
                                                                ; GO TO ERROR ROUTINE IF "ZF" ON
0071 78 0D                      JS      ERR02                   ; GO TO ERROR ROUTINE IF "SF" ON
0073 7A 0B                      JP      ERR02                   ; GO TO ERROR ROUTINE IF "PF" ON
0075 9F                         LAHF                            ; LOAD FLAG IMAGE TO (AH)
0076 D2 EC                      SHR     AH,CL                   ; SHIFT "AF" INTO CARRY BIT POSITION
0078 72 06                      JC      ERR02                   ; GO TO ERROR ROUTINE IF ON
007A D0 E4                      SHL     AH,1                    ; CHECK THAT "OF" IS CLEAR
007C 70 02                      JO      ERR02                   ; GO TO ERROR ROUTINE IF ON
007E 74 03                      JZ      CTA                     ; CONTINUE CONFIDENCE TESTS IF "ZF" SET
0080                    ERR02:
0080 F4                         HLT                             ;       ERROR HALT
0081 EB FD                      JMP     ERR02                   ; ERROR LOOP TRAP

0083                    CTA:
0083 B8 ---- R                  MOV     AX,DATA                 ; SET DATA SEGMENT
0086 8E D8                      MOV     DS,AX                   ; INTO THE (DS) SEGMENT REGISTER

                        ;-----  CHECK FOR PROCESSOR SHUTDOWN

0088 E4 64                      IN      AL,STATUS_PORT          ; READ CURRENT KEYBOARD PROCESSOR STATUS
008A A8 04                      TEST    AL,SYS_FLAG             ; CHECK FOR SHUTDOWN IN PROCESS FLAG
008C 75 03                      JNZ     CTB                     ; GO IF YES
008E E9 0123 R                  JMP     SHUT0                   ; ELSE CONTINUE NORMAL POWER ON CODE
```

```
                        PAGE
                        ;-----  CHECK FOR SHUTDOWN 09
                        C7B:
0091                            MOV     AL,CMOS_SHUT_DOWN+NMI   ; CMOS ADDRESS FOR SHUTDOWN BYTE
0091 B0 8F                      OUT     CMOS_PORT,AL
0093 E6 70                      JMP     $+2                     ; I/O DELAY
0095 EB 00                      IN      AL,CMOS_DATA            ; GET REQUEST NUMBER
0097 E4 71                      CMP     AL,09H                  ; WAS IT SHUTDOWN REQUEST 9?
0099 3C 09                      XCHG    AL,AH                   ; SAVE THE SHUTDOWN REQUEST
009B 86 C4                      JE      C7C                     ; BYPASS INITIALIZING INTERRUPT CHIPS
009D 74 41

                        ;-----  CHECK FOR SHUTDOWN 0A

009F 80 FC 0A                   CMP     AH,0AH                  ; WAS IT SHUTDOWN REQUEST A?
00A2 74 3C                      JE      C7C                     ; BYPASS INITIALIZING INTERRUPT CHIPS

00A4 2A C0                      SUB     AL,AL                   ; INSURE MATH PROCESSOR RESET
00A6 E6 F1                      OUT     X287+1,AL

                        ;--------------------------------------------------------
                        ;   RE-INITIALIZE THE 8259 INTERRUPT #1 CONTROLLER CHIP :
                        ;--------------------------------------------------------
00A8 B0 11                      MOV     AL,11H                  ; ICW1 - EDGE, MASTER, ICW4
00AA E6 20                      OUT     INTA00,AL
00AC EB 00                      JMP     $+2                     ; WAIT STATE FOR I/O
00AE B0 08                      MOV     AL,08H                  ; SETUP ICW2 - INTERRUPT TYPE 8H (8-F)
00B0 E6 21                      OUT     INTA01,AL
00B2 EB 00                      JMP     $+2                     ; WAIT STATE FOR I/O
00B4 B0 04                      MOV     AL,04H                  ; SETUP ICW3 - MASTER LEVEL 2
00B6 E6 21                      OUT     INTA01,AL               ;
00B8 EB 00                      JMP     $+2                     ; I/O WAIT STATE
00BA B0 01                      MOV     AL,01H                  ; SETUP ICW4 - MASTER,8086 MODE
00BC E6 21                      OUT     INTA01,AL               ;
00BE EB 00                      JMP     $+2                     ; WAIT STATE FOR I/O
00C0 B0 FF                      MOV     AL,0FFH                 ; MASK ALL INTERRUPTS OFF
00C2 E6 21                      OUT     INTA01,AL               ; (VIDEO ROUTINE ENABLES INTERRUPTS)
                        ;--------------------------------------------------------
                        ;   RE-INITIALIZE THE 8259 INTERRUPT #2 CONTROLLER CHIP  :
                        ;--------------------------------------------------------
00C4 B0 11                      MOV     AL,11H                  ; ICW1 - EDGE, SLAVE ICW4
00C6 E6 A0                      OUT     INTB00,AL
00C8 EB 00                      JMP     $+2                     ; WAIT STATE FOR I/O
00CA B0 70                      MOV     AL,INT_TYPE             ; SETUP ICW2 - INTERRUPT TYPE 70 (70-7F)
00CC E6 A1                      OUT     INTB01,AL
00CE B0 02                      MOV     AL,02H                  ; SETUP ICW3 - SLAVE LEVEL 2
00D0 EB 00                      JMP     $+2
00D2 E6 A1                      OUT     INTB01,AL
00D4 EB 00                      JMP     $+2                     ; I/O DELAY
00D6 B0 01                      MOV     AL,01H                  ; SETUP ICW4 - 8086 MODE, SLAVE
00D8 E6 A1                      OUT     INTB01,AL
00DA EB 00                      JMP     $+2                     ; WAIT STATE FOR I/O
00DC B0 FF                      MOV     AL,0FFH                 ; MASK ALL INTERRUPTS OFF
00DE E6 A1                      OUT     INTB01,AL
                        ;--------------------------------------------------------------------------------
                        ;  SHUTDOWN - RESTART                                                            :
                        ;       RETURN CONTROL AFTER A SHUTDOWN COMMAND IS ISSUED                        :
                        ;  DESCRIPTION                                                                   :
                        ;       A TEST IS MADE FOR THE SYSTEM FLAG BEING SET.  IF THE SYSTEM FLAG IS     :
                        ;       SET, THE SHUTDOWN BYTE IN CMOS IS USED TO DETERMINE WHERE CONTROL IS     :
                        ;       RETURNED.                                                                :
                        ;                                                                                :
                        ;       CMOS = 0   SOFT RESET OR UNEXPECTED SHUTDOWN                             :
                        ;       CMOS = 1   SHUT DOWN AFTER MEMORY SIZE                                   :
                        ;       CMOS = 2   SHUT DOWN AFTER MEMORY TEST                                   :
                        ;       CMOS = 3   SHUT DOWN WITH MEMORY ERROR                                   :
                        ;       CMOS = 4   SHUT DOWN WITH BOOT LOADER REQUEST                            :
                        ;       CMOS = 5   JMP DWORD REQUEST - (INTERRUPT CHIPS & 287 ARE INITIALIZED)   :
                        ;       CMOS = 6   PROTECTED MODE TEST3 PASSED                                   :
                        ;       CMOS = 7   PROTECTED MODE TEST3 FAILED                                   :
                        ;       CMOS = 8   PROTECTED MODE TEST1 FAILED                                   :
                        ;       CMOS = 9   BLOCK MOVE SHUTDOWN REQUEST                                   :
                        ;       CMOS = A   JMP DWORD REQUEST - (W/O INTERRUPT CHIPS INITIALIZED)          :
                        ;                                                                                :
                        ;          NOTES:  RETURNS ARE MADE WITH INTERRUPTS AND NMI DISABLED.            :
                        ;                  USER MUST RESTORE SS:SP (POST DEFAULT SET = 0000:0400),       :
                        ;                  ENABLE NON-MASKABLE INTERRUPTS (NMI) WITH AN OUT TO           :
                        ;                  PORT 70H WITH HIGH ORDER BIT OFF, AND THEN ISSUE A            :
                        ;                  STI TO ENABLE INTERRUPTS.  FOR SHUTDOWN (5) THE USER          :
                        ;                  MUST ALSO RESTORE THE INTERRUPT MASK REGISTERS.                :
                        ;--------------------------------------------------------------------------------

                        ;-----  CHECK FROM WHERE
00E0                    C7C:
00E0 B0 8F                      MOV     AL,CMOS_SHUT_DOWN+NMI   ; CLEAR CMOS BYTE
00E2 E6 70                      OUT     CMOS_PORT,AL
00E4 90                         NOP                             ; I/O DELAY
00E5 2A C0                      SUB     AL,AL                   ; SET BYTE TO 0
00E7 E6 71                      OUT     CMOS_DATA,AL
00E9 86 E0                      XCHG    AH,AL
00EB 3C 0A                      CMP     AL,0AH                  ; COMPARE WITH MAXIMUM TABLE ENTRIES
00ED 77 34                      JA      SHUT0                   ; SKIP TO POST IF GREATER THAN MAXIMUM
00EF BE 0103 R                  MOV     SI,OFFSET BRANCH        ; POINT TO THE START OF THE BRANCH TABLE
00F2 03 F0                      ADD     SI,AX
00F4 03 F0                      ADD     SI,AX                   ; POINT TO BRANCH ADDRESS
00F6 2E: 8B 1C                  MOV     BX,CS:[SI]              ; MOVE BRANCH TO ADDRESS TO BX REGISTER

                        ;-----  SET TEMPORARY STACK FOR POST

00F9 B8 ---- R                  MOV     AX,ABS0                 ; SET STACK SEGMENT TO ABS0 SEGMENT
00FC 8E D0                      MOV     SS,AX
00FE BC 0400 R                  MOV     SP,OFFSET @TOS          ; SET STACK POINTER TO END OF VECTORS
0101 FF E3                      JMP     BX                      ; JUMP BACK TO RETURN ROUTINE

0103 0123 R             BRANCH: DW      SHUT0                   ; NORMAL POWER UP/UNEXPECTED SHUTDOWN
0105 0992 R                     DW      SHUT1                   ; SHUT DOWN AFTER MEMORY SIZE
0107 0000 E                     DW      SHUT2                   ; SHUT DOWN AFTER MEMORY TEST
0109 0000 E                     DW      SHUT3                   ; SHUT DOWN WITH MEMORY ERROR
010B 0000 E                     DW      SHUT4                   ; SHUT DOWN WITH BOOT LOADER REQUEST
010D 0119 R                     DW      SHUT5                   ; JMP DWORD REQUEST WITH INTERRUPT INIT
010F 0000 E                     DW      SHUT6                   ; PROTECTED MODE TEST7 PASSED
0111 0000 E                     DW      SHUT7                   ; PROTECTED MODE TEST7 FAILED
0113 0795 R                     DW      SHUT8                   ; PROTECTED MODE TEST1 FAILED
0115 0000 E                     DW      SHUT9                   ; BLOCK MOVE SHUTDOWN REQUEST
0117 011F R                     DW      SHUTA                   ; JMP DWORD REQUEST (W/O INTERRUPT INIT)
```

```
                                PAGE
                        ;-----  @IO_ROM_INIT MUST BE INITIALIZED BY THE USER FOR VECTORED REQUESTS

                        SHUT5:
0119 E4 60                      IN      AL,PORT_A               ; FLUSH THE KEYBOARD BUFFER
011B B0 20                      MOV     AL,EOI                  ; FLUSH LAST TIMER REQUEST IF PENDING
011D E6 20                      OUT     INTA00,AL               ;  - TO ALLOW TIMER INTERRUPTS
011F                    SHUTA:
011F FF 2E 0067 R               JMP     DWORD PTR @IO_ROM_INIT  ; FAR JUMP TO USER DEFINED LOCATION
                                                                ;  AFTER SHUTDOWN TO REAL MODE CODE
                                                                ;  WITH INTERRUPTS AND NMI DISABLED
                        ;-----  CHECKPOINT 01

                        SHUT0:
0123 B0 01                      MOV     AL,01H                  ;       <><><><><><><><><>
0125 E6 80                      OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  01 <><>

                        ;-----  READ/WRITE/TEST THE 80286 REGISTERS WITH ONE'S AND ZERO'S

0127 B8 FFFF                    MOV     AX,0FFFFH               ; SETUP ONE'S PATTERN IN (AX)
012A F9                         STC                             ; SET CARRY FLAG
012B 73 21                      JNC     ERR01                   ; GO IF NO CARRY
012D                    C8:
012D 8E D8                      MOV     DS,AX                   ; WRITE PATTERN TO ALL REGISTERS
012F 8C DB                      MOV     BX,DS
0131 8E C3                      MOV     ES,BX
0133 8C C1                      MOV     CX,ES
0135 8E D1                      MOV     SS,CX
0137 8C D2                      MOV     DX,SS
0139 8B E2                      MOV     SP,DX
013B 8B F5                      MOV     BP,SP
013D 8B F6                      MOV     SI,BP
013F 8B FE                      MOV     DI,SI
0141 73 07                      JNC     C9
0143 33 C7                      XOR     AX,DI                   ; PATTERN MAKE IT THROUGH ALL REGISTERS
0145 75 07                      JNZ     ERR01                   ; NO - GO TO ERROR ROUTINE
0147 F8                         CLC                             ; CLEAR CARRY FLAG
0148 EB E3                      JMP     C8
014A                    C9:                                     ; TSTIA
014A 0B C7                      OR      AX,DI                   ; ZERO PATTERN MAKE IT THROUGH ?
014C 74 01                      JZ      C10A                    ; YES - GO TO NEXT TEST
014E                    ERR01:
014E F4                         HLT                             ; HALT SYSTEM

                        ;-----  INSURE THAT CMOS CLOCK INTERRUPTS ARE DISABLED
014F                    C10A:
014F B0 8B                      MOV     AL,(CMOS_REG_B+NMI)     ; ADDRESS CMOS ALARM BYTE WITH NMI=OFF
0151 E6 70                      OUT     CMOS_PORT,AL            ;
0153 90                         NOP                             ; I/O DELAY
0154 E4 71                      IN      AL,CMOS_DATA            ; READ DATA TO OPEN STANDBY LATCH
0156 B8 8B8B                    MOV     AX,X*(CMOS_REG_B+NMI)   ; ADDRESS TO BOTH (AH) AND (AL)
0159 E6 70                      OUT     CMOS_PORT,AL            ; ADDRESS CMOS ALARM BYTE WITH NMI=OFF
015B 90                         NOP                             ; I/O DELAY
015C E4 71                      IN      AL,CMOS_DATA            ; GET THE CURRENT CONTROL REGISTER
015E 24 07                      AND     AL,00000111B            ; CLEAR SET,PIE,AIE, AND SQWE BITS
0160 86 C4                      XCHG    AL,AH                   ; SAVE IT
0162 E6 70                      OUT     CMOS_PORT,AL
0164 86 C4                      XCHG    AL,AH
0166 E6 71                      OUT     CMOS_DATA,AL

0168 B0 8C                      MOV     AL,CMOS_REG_C+NMI       ; ADDRESS CMOS FLAGS BYTE WITH NMI=OFF
016A 90                         NOP                             ; I/O DELAY
016B E6 70                      OUT     CMOS_PORT,AL
016D 90                         NOP                             ; I/O DELAY
016E E4 71                      IN      AL,CMOS_DATA            ; READ STATUS TO CLEAR PENDING INTERRUPT

                        ;-----  RESET VIDEO
0170 B0 00                      MOV     AL,0                    ; CLEAR DATA BYTE TO DISABLE VIDEO
0172 BA 03D8                    MOV     DX,03D8H                ; GET COLOR MODE CONTROL PORT ADDRESS
0175 EE                         OUT     DX,AL                   ; DISABLE COLOR VIDEO
0176 FE C0                      INC     AL                      ; MONOCHROME MODE RESET MASK
0178 B2 B8                      MOV     DL,0B8H                 ; GET ADDRESS OF MONOCHROME MODE CONTROL
017A EE                         OUT     DX,AL                   ; DISABLE B/W VIDEO, ENABLE HIGH RES
017B B2 BA                      MOV     DL,0BAH                 ; ADDRESS OF MONOCHROME STATUS REGISTER
017D EC                         IN      AL,DX                   ; READ STATUS TO DISABLE EGA VIDEO
017E B2 DA                      MOV     DL,0DAH                 ; ADDRESS OF COLOR MODE STATUS REGISTER
0180 EC                         IN      AL,DX                   ; READ STATUS TO DISABLE EGA VIDEO
0181 B0 00                      MOV     AL,0                    ; SELECT ATTRIBUTE PALETTE REGISTER 0
0183 B2 C0                      MOV     DL,0C0H                 ; WRITE 0 TO ATTRIBUTE ADDRESS REGISTER
0185 EE                         OUT     DX,AL                   ; TO DISABLE EGA VIDEO
0186 B0 FC                      MOV     AL,11111100B            ; DISABLE PARITY CHECKERS
0188 E6 61                      OUT     PORT_B,AL

                        ;--------------------------------------
                        ; TEST.02                              :
                        ;       ROM CHECKSUM TEST I            :
                        ; DESCRIPTION                          :
                        ;       A CHECKSUM IS DONE FOR THE 32K :
                        ;       READ ONLY MEMORY MODULES (TWO) :
                        ;       CONTAINING POST, BASIC AND BIOS.:
                        ;--------------------------------------

                        ;-----  CHECKPOINT 02

018A B0 02                      MOV     AL,02H                  ;       <><><><><><><><><>
018C E6 80                      OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  02 <><>

                                ASSUME  SS:CODE
018E 8C C8                      MOV     AX,CS                   ; SETUP SS SEGMENT REGISTER
0190 8E D0                      MOV     SS,AX
0192 8E D8                      MOV     DS,AX                   ; SET UP DATA SEGMENT TO POINT TO
0194 33 F6                      XOR     SI,SI                   ; ROM ADDRESS START
0196 33 DB                      XOR     BX,BX                   ; CLEAR CHECK REGISTER
0198 B5 80                      MOV     CH,080H                 ; COUNT FOR 32K WORDS
019A                    C11:
019A AD                         LODSW                           ; MOVE TWO BYTES INTO AX -- SI=SI+2
019B 02 DC                      ADD     BL,AH                   ; ADD ODD BYTE AT DS:SI+1 TO CHECKSUM
019D 02 D8                      ADD     BL,AL                   ; ADD EVEN BYTE AT DS:SI TO CHECKSUM
019F E2 F9                      LOOP    C11                     ; LOOP COUNT FOR 65K BYTES (32K WORDS)
01A1 73 02                      JNC     C11E                    ;  EXIT IF "LOOP" RESET THE CARRY FLAG
                                                                ;  (NOTE: MODEL BYTE MUST NOT = ZERO)
01A3 74 01                      JZ      C11A                    ; CONTINUE IF CHECKSUM VALID (ZERO)
01A5                    C11E:
01A5 F4                         HLT                             ; ELSE HALT IF CHECKSUM PROBLEM
```

SECTION 5

```
                        ;--------------------------------------
                        ; TEST.03                             :
                        ;       VERIFY CMOS SHUTDOWN BYTE     :
                        ; DESCRIPTION                         :
                        ;       ROLLING BIT WRITTEN AND       :
                        ;       VERIFIED AT SHUTDOWN ADDRESS. :
                        ;--------------------------------------

                        ;-----  VERIFY AND CLEAR SHUTDOWN FLAG

 01A6                   C11A:
 01A6 B0 03                     MOV     AL,03H                  ;       <><><><><><><><><>
 01A8 E6 80                     OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  03 <><>

 01AA B9 0009                   MOV     CX,09H                  ; LOOP COUNT
 01AD B4 01                     MOV     AH,1                    ; START WITH BIT 0
 01AF                   C11B:
 01AF B0 8F                     MOV     AL,CMOS_SHUT_DOWN+NMI
 01B1 E6 70                     OUT     CMOS_PORT,AL
 01B3 8A C4                     MOV     AL,AH                   ; OUTPUT ROLLING BIT
 01B5 E6 71                     OUT     CMOS_DATA,AL
 01B7 B0 8F                     MOV     AL,CMOS_SHUT_DOWN+NMI   ; READ CMOS
 01B9 90                        NOP                             ; I/O DELAY
 01BA E6 70                     OUT     CMOS_PORT,AL
 01BC 90                        NOP                             ; I/O DELAY
 01BD E4 71                     IN      AL,CMOS_DATA
 01BF 3A C4                     CMP     AL,AH                   ; MUST BE THE SAME
 01C1 75 89                     JNZ     ERR01                   ; ERROR IF NOT
 01C3 D0 D4                     RCL     AH,1                    ; ROLL A BIT THROUGH SHUTDOWN BYTE
 01C5 E2 E8                     LOOP    C11B                    ; LOOP TILL DONE

                        ;--------------------------------------
                        ; TEST.04                             :
                        ;       8254 CHECK TIMER 1            :
                        ; DESCRIPTION                         :
                        ;       SET TIMER COUNT TO            :
                        ;       CHECK THAT TIMER 1 ALL BITS ON :
                        ;--------------------------------------

                                ASSUME  DS:DATA
 01C7 B8 ---- R                 MOV     AX,DATA                 ; SET DATA SEGMENT
 01CA 8E D8                     MOV     DS,AX
 01CC B0 04                     MOV     AL,04H                  ;       <><><><><><><><><>
 01CE E6 80                     OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  04 <><>

                        ;-----  DISABLE DMA CONTROLLER

 01D0 E6 08                     OUT     DMA08,AL                ; (AL) ALREADY = 04H
 01D2 E6 D0                     OUT     DMA18,AL                ; DISABLE DMA CONTROLLER 1
                                                                ; DISABLE DMA CONTROLLER 2

                        ;-----  VERIFY THAT TIMER 1 FUNCTIONS OK

 01D4 8B 16 0072 R              MOV     DX,@RESET_FLAG          ; SAVE RESET FLAG WHILE REFRESH IS OFF
 01D8 B0 54                     MOV     AL,54H                  ; SELECT TIMER 1,LSB,MODE 2
 01DA E6 43                     OUT     TIMER+3,AL
 01DC EB 00                     JMP     $+2                     ; I/O DELAY
 01DE B0 12                     MOV     AL,18                   ; SET INITIAL TIMER COUNT TO 18
 01E0 E6 41                     OUT     TIMER+1,AL
 01E2 B7 05                     MOV     BH,05H                  ; LOOP COUNT
 01E4                   C12:                                    ; TIMER1_BITS_ON
 01E4 B0 40                     MOV     AL,40H                  ; LATCH TIMER 1 COUNT
 01E6 EB 00                     JMP     $+2                     ; I/O DELAY
 01E8 E6 43                     OUT     TIMER+3,AL
 01EA 80 FB 1F                  CMP     BL,01FH                 ; YES - SEE IF ALL BITS GO ON
 01ED 74 0B                     JE      C13                     ; TIMER1_BITS_OFF
 01EF E4 41                     IN      AL,TIMER+1              ; READ TIMER T COUNT
 01F1 0A D8                     OR      BL,AL                   ; ALL BITS ON IN TIMER
 01F3 E2 EF                     LOOP    C12                     ; TIMER1_BITS_ON
 01F5 FE CF                     DEC     BH
 01F7 75 EB                     JNZ     C12                     ; TRY AGAIN
 01F9 F4                        HLT                             ; TIMER 1 FAILURE, HALT SYSTEM
                                                                ; TIMER1_BITS_OFF
                        ;--------------------------------------
                        ; TEST.05                             :
                        ;       8254 CHECK TIMER 1 ALL BIT OFF :
                        ; DESCRIPTION                         :
                        ;       SET TIMER COUNT               :
                        ;       CHECK THAT TIMER 1 ALL BITS OFF :
                        ;--------------------------------------

                        ;-----  CHECKPOINT 05

 01FA B0 05             C13:    MOV     AL,05H                  ;       <><><><><><><><><>
 01FC E6 80                     OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  05 <><>

 01FE 8A C3                     MOV     AL,BL                   ; SET TIMER 1 COUNT
 0200 2B C9                     SUB     CX,CX
 0202 E6 41                     OUT     TIMER+1,AL
 0204 B7 05                     MOV     BH,05H                  ; SET TRY AGAIN COUNT
 0206                   C14:                                    ; TIMER_LOOP
 0206 B0 40                     MOV     AL,40H                  ; LATCH TIMER 1 COUNT
 0208 E6 43                     OUT     TIMER+3,AL
 020A EB 00                     JMP     $+2                     ; DELAY FOR TIMER
 020C EB 00                     JMP     $+2                     ; ADDED DELAY FOR TIMER
 020E E4 41                     IN      AL,TIMER+1              ; READ TIMER 1 COUNT
 0210 22 D8                     AND     BL,AL
 0212 74 07                     JZ      C15                     ; GO TO WRAP DMA REGISTER TESTS
 0214 E2 F0                     LOOP    C14                     ; TIMER_LOOP
 0216 FE CF                     DEC     BH
 0218 75 EC                     JNZ     C14
 021A F4                        HLT                             ; HALT SYSTEM

                        ;--------------------------------------
                        ; TEST.06                             :
                        ;       8237 DMA 0 INITIALIZATION     :
                        ;       CHANNEL REGISTER TEST         :
                        ; DESCRIPTION                         :
                        ;       DISABLE THE 8237 DMA CONTROLLER.:
                        ;       WRITE/READ THE CURRENT ADDRESS :
                        ;       AND WORD COUNT REGISTERS FOR  :
                        ;       ALL CHANNELS.                 :
                        ;--------------------------------------
```

```
                        ;-----  CHECKPOINT 06
0221                    C15:
0221 B8 ---- R                  MOV     AX,DATA                 ; SET DATA SEGMENT
0224 8E D8                      MOV     DS,AX
0226 B0 06                      MOV     AL,06H                  ;       <><><><><><><><><>
0228 E6 80                      OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  06 <><>
022A 89 16 0072 R               MOV     @RESET_FLAG,DX          ; RESTORE SOFT RESET FLAG
022E E6 0D                      OUT     DMA+0DH,AL              ; SEND MASTER CLEAR TO DMA
                        ;-----  WRAP DMA 0 CHANNEL ADDRESS AND COUNT REGISTERS
0231 B0 FF                      MOV     AL,0FFH                 ; WRITE PATTERN "FF" TO ALL REGISTERS
0233 8A D8              C16:    MOV     BL,AL                   ; SAVE PATTERN FOR COMPARE
0235 8A F8                      MOV     BH,AL
0237 B9 0008                    MOV     CX,8                    ; SETUP LOOP COUNT
023A BA 0000                    MOV     DX,DMA                  ; SETUP I/O PORT ADDRESS OF REGISTER
023D EE                 C17:    OUT     DX,AL                   ; WRITE PATTERN TO REGISTER, LSB
023E EB 00                      JMP     $+2                     ; I/O DELAY
0240 EE                         OUT     DX,AL                   ; MSB OF 16 BIT REGISTER
0241 B0 01                      MOV     AL,01H                  ; AL TO ANOTHER PATTERN BEFORE READ
0243 EB 00                      JMP     $+2                     ; I/O DELAY
0245 EC                         IN      AL,DX                   ; READ 16-BIT DMA CH REG, LSB  2ST DMA
0246 EB 00                      JMP     $+2                     ; I/O DELAY
0248 8A E0                      MOV     AH,AL                   ; SAVE LSB OF 16-BIT REGISTER
024A EC                         IN      AL,DX                   ; READ MSB OF DMA CHANNEL REGISTER
024B 3B D8                      CMP     BX,AX                   ; PATTERN READ AS WRITTEN?
024D 74 01                      JE      C18                     ; YES - CHECK NEXT REGISTER
024F F4                         HLT                             ; NO - HALT THE SYSTEM
0250                    C18:                                    ; NXT_DMA_CH
0250 42                         INC     DX                      ; SET I/O PORT TO NEXT CHANNEL REGISTER
0251 E2 EA                      LOOP    C17                     ; WRITE PATTERN TO NEXT REGISTER
0253 FE C0                      INC     AL                      ; SET PATTERN TO 0
0255 74 DC                      JZ      C16                     ; YES CONTINUE
                        ;-----  WRITE DMA WITH 55 PATTERN
0257 80 FB 55                   CMP     BL,055H                 ; CHECK IF "55" PATTERN DONE
025A 74 09                      JZ      C19                     ; GO IF YES
025C 80 FB AA                   CMP     BL,0AAH                 ; CHECK IF "AA" PATTERN DONE
025F 74 08                      JZ      C20                     ; GO IF YES
0261 B0 55                      MOV     AL,055H
0263 EB CE                      JMP     C16
                        ;-----  WRITE DMA WITH AA PATTERN
0265 B0 AA              C19:    MOV     AL,0AAH
0267 EB CA                      JMP     C16
                        ;---------------------------------------
                        ; TEST.07                              :
                        ;       8237 DMA 1 INITIALIZATION      :
                        ;       CHANNEL REGISTER TEST          :
                        ; DESCRIPTION                          :
                        ;       DISABLE 8237 DMA CONTROLLER 1. :
                        ;       WRITE/READ THE CURRENT DMA 1   :
                        ;       ADDRESS AND WORD COUNT         :
                        ;       REGISTERS FOR ALL CHANNELS.    :
                        ;---------------------------------------
                        ;-----  CHECKPOINT 07 - DMA 1
0269 B0 07              C20:    MOV     AL,07H                  ;       <><><><><><><><><>
026B E6 80                      OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  07 <><>
026D E6 DA                      OUT     DMAT+0DH*2,AL           ; SEND MASTER CLEAR TO 2ND DMA
                        ;-----  WRAP DMA 1 CHANNEL ADDRESS AND COUNT REGISTERS
026F B0 FF                      MOV     AL,0FFH                 ; WRITE PATTERN FF TO ALL REGISTERS
0271 8A D8              C16A:   MOV     BL,AL                   ; SAVE PATTERN FOR COMPARE
0273 8A F8                      MOV     BH,AL
0275 B9 0008                    MOV     CX,8                    ; SETUP LOOP COUNT
0278 BA 00C0                    MOV     DX,DMA1                 ; SETUP I/O PORT ADDRESS OF REGISTER
027B EE                 C17A:   OUT     DX,AL                   ; WRITE PATTERN TO REGISTER, LSB
027C EB 00                      JMP     $+2                     ; I/O DELAY
027E EE                         OUT     DX,AL                   ; MSB OF 16 BIT REGISTER
027F B0 01                      MOV     AL,01H                  ; AL TO ANOTHER PAT BEFORE RD
0281 EB 00                      JMP     $+2                     ; I/O DELAY
0283 EC                         IN      AL,DX                   ; READ 16-BIT DMA CH REG, LSB  2ST DMA
0284 EB 00                      JMP     $+2                     ; I/O DELAY
0286 8A E0                      MOV     AH,AL                   ; SAVE LSB OF 16-BIT REGISTER
0288 EC                         IN      AL,DX                   ; READ MSB OF DMA CH REGISTER
0289 3B D8                      CMP     BX,AX                   ; PATTERN READ AS WRITTEN?
028B 74 01                      JE      C18A                    ; YES - CHECK NEXT REGISTER
028D F4                         HLT                             ; NO - HALT THE SYSTEM
028E                    C18A:                                   ; NXT_DMA_CH
028E 83 C2 02                   ADD     DX,2                    ; SET I/O PORT TO NEXT CHANNEL REGISTER
0291 E2 E8                      LOOP    C17A                    ; WRITE PATTERN TO NEXT REGISTER
0293 FE C0                      INC     AL                      ; SET PATTERN TO 0
0295 74 DA                      JZ      C16A                    ; YES CONTINUE
                        ;-----  WRITE DMA WITH 55 PATTERN
0297 80 FB 55                   CMP     BL,55H                  ; CHECK IF 55 PATTERN DONE
029A 74 09                      JZ      C20A                    ; GO IF YES
029C 80 FB AA                   CMP     BL,0AAH                 ; CHECK IF AA PATTERN DONE
029F 74 08                      JZ      C21                     ; GO IF YES
02A1 B0 55                      MOV     AL,55H
02A3 EB CC                      JMP     C16A
                        ;-----  WRITE DMA WITH AA PATTERN
02A5 B0 AA              C20A:   MOV     AL,0AAH
02A7 EB C8                      JMP     C16A
                        ;-----  INITIALIZE AND START MEMORY REFRESH
02A9                    C21:
02A9 8B 1E 0072 R               MOV     BX,@RESET_FLAG          ; GET THE RESET FLAG
02AD A3 0010 R                  MOV     @EQUIP_FLAG,AX          ; DO A DUMMY MEMORY WRITE BEFORE REFRESH
02B0 B0 12                      MOV     AL,18                   ; START REFRESH TIMER
02B2 E6 41                      OUT     TIMER+1,AL
                        ;-----  SET DMA COMMAND
```

SECTION 5

```
02AD 2A C0                      SUB     AL,AL                   ; DACK SENSE LOW,DREQ SENSE HIGH
02AF E6 08                      OUT     DMA+8,AL                ; LATE WRITE, FIXED PRIORITY, NORMAL
                                                                ; TIMING, CONTROLLER ENABLE, CH0 ADDRESS
                                                                ; HOLD DISABLE, MEMORY TO MEMORY DISABLE
02B1 E6 D0                      OUT     DMA18,AL                ; SAME TO SECOND CONTROLLER

                        ;-----  MODE SET ALL DMA CHANNELS

02B3 B0 40                      MOV     AL,40H                  ; SET MODE FOR CHANNEL 0
02B5 E6 0B                      OUT     DMA+0BH,AL
02B7 B0 C0                      MOV     AL,0C0H                 ; SET CASCADE MODE ON CHANNEL 4
02B9 E6 D6                      OUT     DMA18+06H,AL
02BB EB 00                      JMP     $+2                     ; I/O DELAY
02BD B0 41                      MOV     AL,41H                  ; SET MODE FOR CHANNEL 1
02BF E6 0B                      OUT     DMA+0BH,AL
02C1 E6 D6                      OUT     DMA18+06H,AL            ; SET MODE FOR CHANNEL 5
02C3 EB 00                      JMP     $+2                     ; I/O DELAY
02C5 B0 42                      MOV     AL,42H                  ; SET MODE FOR CHANNEL 2
02C7 E6 0B                      OUT     DMA+0BH,AL
02C9 E6 D6                      OUT     DMA18+06H,AL            ; SET MODE FOR CHANNEL 6
02CB EB 00                      JMP     $+2                     ; I/O DELAY
02CD B0 43                      MOV     AL,43H                  ; SET MODE FOR CHANNEL 3
02CF E6 0B                      OUT     DMA+0BH,AL
02D1 E6 D6                      OUT     DMA18+06H,AL            ; SET MODE FOR CHANNEL 7

                        ;-----  RESTORE RESET FLAG

02D3 89 1E 0072 R               MOV     @RESET_FLAG,BX

                        ;--------------------------------------
                        ; TEST.08                             :
                        ;       DMA PAGE REGISTER TEST        :
                        ; DESCRIPTION                        :
                        ;       WRITE/READ ALL PAGE REGISTERS :
                        ;--------------------------------------

                        ;-----  CHECKPOINT 08

02D7 B0 08                      MOV     AL,08H                  ;       <><><><><><><><>
02D9 E6 80                      OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  08 <><>
02DB 2A C0                      SUB     AL,AL
02DD BA 0081                    MOV     DX,DMA_PAGE
02E0 B9 00FF                    MOV     CX,0FFH                 ; DO ALL DATA PATTERNS
02E3 EE                 C22A:   OUT     DX,AL
02E4 42                         INC     DX
02E5 FE C0                      INC     AL
02E7 81 FA 008F                 CMP     DX,8FH                  ; TEST DMA PAGES 81 THROUGH 8EH
02EB 75 F6                      JNZ     C22A
02ED 86 E0                      XCHG    AH,AL                   ; SAVE CURRENT DATA PATTERN
02EF FE CC                      DEC     AH                      ; CHECK LAST WRITTEN
02F1 4A                         DEC     DX                      ;
02F2 2A C0              C22B:   SUB     AL,AL                   ; CHANGE DATA BEFORE READ
02F4 EC                         IN      AL,DX
02F5 3A C4                      CMP     AL,AH                   ; DATA AS WRITTEN?
02F7 75 30                      JNZ     C26                     ; GO ERROR HALT IF NOT
02F9 FE CC                      DEC     AH
02FB 4A                         DEC     DX
02FC 81 FA 0080                 CMP     DX,MFG_PORT             ; CONTINUE TILL PORT 80
0300 75 F0                      JNZ     C22B
0302 FE C4                      INC     AH                      ; NEXT PATTERN TO RIPPLE
0304 8A C4                      MOV     AL,AH
0306 E2 DB                      LOOP    C22A

                        ;-----  TEST LAST DMA PAGE REGISTER (USED FOR ADDRESS LINES DURING REFRESH)

0308 B0 CC                      MOV     AL,0CCH                 ; WRITE AN CC TO PAGE REGISTERS
030A BA 008F            C22:    MOV     DX,LAST_DMA_PAGE
030D 8A E0                      MOV     AH,AL                   ; SAVE THE DATA PATTERN
030F EE                         OUT     DX,AL                   ; OUTPUT PAGE REGISTER

                        ;-----  VERIFY PAGE REGISTER 8F

0310 2A C0                      SUB     AL,AL                   ; CHANGE DATA PATTERN BEFORE READ
0312 EC                         IN      AL,DX                   ; GET THE DATA FROM PAGE REGISTER
0313 3A C4                      CMP     AL,AH
0315 75 12                      JNZ     C26                     ; GO IF ERROR
0317 80 FC CC                   CMP     AH,0CCH
031A 75 04                      JNZ     C25                     ; GO IF ERROR
031C B0 33                      MOV     AL,033H                 ; SET UP DATA PATTERN OF 33
031E EB EA                      JMP     C22                     ; DO DATA 33
0320                    C25:
0320 80 FC 00                   CMP     AH,0                    ; CHECK DONE
0323 74 06                      JZ      C27                     ; GO IF YES
0325 2A C0                      SUB     AL,AL                   ; SET UP FOR DATA PATTERN 00
0327 EB E1                      JMP     C22                     ; DO DATA 0

                        ;-----  ERROR HALT
0329                    C26:
0329 F4                         HLT                             ; HALT SYSTEM

                        ;--------------------------------------
                        ; TEST.09                             :
                        ;       STORAGE REFRESH TEST          :
                        ; DESCRIPTION                        :
                        ;       VERIFY REFRESH IS OCCURRING   :
                        ;--------------------------------------

                        ;-----  CHECKPOINT 09 - TEST MEMORY REFRESH

032A                    C27:
032A B0 09                      MOV     AL,09H                  ;       <><><><><><><><>
032C E6 80                      OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  09 <><>
032E 2B C9                      SUB     CX,CX                   ;
0330                    C28:
0330 E4 61                      IN      AL,PORT_B               ; INSURE REFRESH BIT IS TOGGLING
0332 A8 10                      TEST    AL,REFRESH_BIT
0334 E1 FA                      LOOPZ   C28                     ; INSURE REFRESH IS OFF
0336 E3 F1                      JCXZ    C26                     ; ERROR HALT IF TIMEOUT
0338                    C29:
0338 E4 61                      IN      AL,PORT_B
033A A8 10                      TEST    AL,REFRESH_BIT          ; INSURE REFRESH IS ON
033C E0 FA                      LOOPNZ  C29
033E E3 E9                      JCXZ    C26                     ; ERROR HALT IF NO REFRESH BIT
```

```
                        ;--------------------------------------
                        ; TEST.10                             :
                        ;       8042 INTERFACE TEST           :
                        ;       READ CONFIGURATION JUMPERS    :
                        ; DESCRIPTION                         :
                        ;       ISSUE A SELF TEST TO THE 8042.:
                        ;       INSURE A 55H IS RECEIVED.     :
                        ;       READ MANUFACTURING AND DISPLAY:
                        ;       JUMPERS AND SAVE IN MFG_TEST. :
                        ;--------------------------------------

                        ;-----  CHECKPOINT 0A

0340 B0 0A                      MOV     AL,0AH                  ;       <><><><><><><><>
0342 E6 80                      OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  0A <><

                        ;-----  SOFT RESET (HANDLE ALL POSSIBLE CONDITIONS)

0344 2B C9                      SUB     CX,CX                   ; 100 MILLISECONDS FOR THIS LOOP
0346 E4 64              TST1:   IN      AL,STATUS_PORT          ; CHECK FOR INPUT BUFFER FULL
0348 8A E0                      MOV     AH,AL
034A F6 C4 01                   TEST    AH,OUT_BUF_FULL
034D 74 02                      JZ      TST2                    ; GO IF NOT
034F E4 60                      IN      AL,PORT_A               ; FLUSH
0351 F6 C4 02           TST2:   TEST    AH,INPT_BUF_FULL        ; IS THE OUTPUT BUFFER ALSO FULL?
0354 E0 F0                      LOOPNZ  TST1                    ; TRY AGAIN
0356 74 01                      JZ      TST4                    ; CONTINUE IF OK

0358 F4                 ERR0:   HLT                             ; HALT SYSTEM IF BUFFER FULL

                        ;-----  ISSUE A RESET TO THE 8042

0359 B0 0B              TST4:   MOV     AL,0BH                  ;       <><><><><><><><>
035B E6 80                      OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  0B <><

035D B0 AA                      MOV     AL,SELF_TEST            ; SELF TEST COMMAND
035F BC 03F5 R                  MOV     SP,OFFSET C8042A        ; SET RETURN ADDRESS
0362 EB 39                      JMP     SHORT C8042
0364 A8 01              TST4_B: TEST    AL,OUT_BUF_FULL         ; IS THE OUTPUT BUFFER FULL?
0366 74 02                      JZ      TST4_A                  ; GO IF NOT
0368 E4 60                      IN      AL,PORT_A               ; FLUSH
036A BC 03F7 R          TST4_A: MOV     SP,OFFSET OBF_42A       ; SET RETURN ADDRESS
036D EB 3A                      JMP     SHORT OBF_42            ; GO WAIT FOR BUFFER
036F E4 60              TST4_C: IN      AL,PORT_A               ; GET THE ENDING RESPONSE
0371 3C 55                      CMP     AL,55H

0373 B0 0C                      MOV     AL,0CH                  ;       <><><><><><><><>
0375 E6 80                      OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  0C <><

0377 75 DF                      JNZ     ERR0                    ; GO IF NOT OK

                        ;-----  GET THE SWITCH SETTINGS

0379 B0 C0                      MOV     AL,READ_8042_INPUT      ; READ INPUT COMMAND
037B BC 03FB R                  MOV     SP,OFFSET C8042C        ; SET RETURN ADDRESS
037E EB 1D                      JMP     SHORT C8042             ; ISSUE COMMAND
0380 BC 03FD R          E30B:   MOV     SP,OFFSET OBF_42B       ; SET RETURN ADDRESS
0383 EB 24                      JMP     SHORT OBF_42            ; GO WAIT FOR RESPONSE
0385 E4 60              E30C:   IN      AL,PORT_A               ; GET THE SWITCH
0387 E6 82                      OUT     DMA_PAGE+1,AL           ; SAVE TEMPORARY

                        ;-----  WRITE BYTE 0 OF 8042 MEMORY

0389 B0 60                      MOV     AL,WRITE_8042_LOC       ; WRITE BYTE COMMAND
038B BC 03F9 R                  MOV     SP,OFFSET C8042B        ; SET RETURN ADDRESS
038E EB 0D                      JMP     SHORT C8042             ; ISSUE THE COMMAND
0390 74 05              TST4_D: JZ      TST4_D1                 ; CONTINUE IF COMMAND ACCEPTED

0392 B0 0D                      MOV     AL,0DH                  ;       <><><><><><><><>
0394 E6 80                      OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  0D <><
0396 F4                         HLT
0397                    TST4_D1:
0397 B0 5D                      MOV     AL,5DH                  ; ENABLE OUTPUT BUFFER FULL INTERRUPT,
0399 E6 60                      OUT     PORT_A,AL               ; DISABLE KEYBOARD, SET SYSTEM FLAG,
039B EB 1D                      JMP     SHORT E30A              ; PC I COMPATIBILITY, INHIBIT OVERRIDE

                        ;-----  ISSUE THE COMMAND TO THE 8042

039D FA                 C8042:  CLI                             ; NO INTERRUPTS ALLOWED
039E E6 64                      OUT     STATUS_PORT,AL          ; SEND COMMAND IN AL REGISTER
03A0 2B C9                      SUB     CX,CX                   ; LOOP COUNT
03A2 E4 64              C42_1:  IN      AL,STATUS_PORT          ; WAIT FOR THE COMMAND ACCEPTED
03A4 A8 02                      TEST    AL,INPT_BUF_FULL
03A6 E0 FA                      LOOPNZ  C42_1
03A8 C3                         RET

                        ;-----  WAIT FOR 8042 RESPONSE

03A9 2B C9              OBF_42: SUB     CX,CX
03AB B3 06                      MOV     BL,6                    ; 200MS/PER LOOP * 6  =1200 MS +
03AD E4 64              C42_2:  IN      AL,STATUS_PORT          ; CHECK FOR RESPONSE
03AF A8 01                      TEST    AL,OUT_BUF_FULL
03B1 75 06                      JNZ     C42_3                   ; GO IF RESPONSE
03B3 E2 F8                      LOOP    C42_2                   ; TRY AGAIN
03B5 FE CB                      DEC     BL                      ; DECREMENT LOOP COUNT
03B7 75 F4                      JNZ     C42_2
03B9 C3                 C42_3:  RET                             ; RETURN TO CALLER

                        ;--------------------------------------
                        ; TEST.11                             :
                        ;       BASE 64K READ/WRITE MEMORY TEST:
                        ; DESCRIPTION                         :
                        ;       WRITE/READ/VERIFY DATA PATTERNS:
                        ;       AA,55,FF,01, AND 00 TO 1 ST 64K:
                        ;       OF STORAGE. VERIFY STORAGE    :
                        ;       ADDRESSABILITY.               :
                        ;--------------------------------------

                        ;-----  FILL MEMORY WITH DATA

03BA B0 0E              E30A:   MOV     AL,0EH                  ;       <><><><><><><><>
03BC E6 80                      OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  0E <><
```

```
                        MOV     AX,DATA                 ; GET THE SYSTEM SEGMENT
                        MOV     DS,AX                   ;  OF DATA
                        MOV     BX,@RESET_FLAG          ; SAVE @RESET_FLAG IN BX
                        CLD                             ; SET DIRECTION FLAG TO INCREMENT
                        MOV     CX,2000H*4              ; SET FOR 32K WORDS
                        SUB     DI,DI                   ; FIRST 16K
                        SUB     SI,SI
                        SUB     AX,AX
                        MOV     DS,AX
                        MOV     ES,AX
                        CMP     BX,1234H                ; WARM START?
                        JNZ     E30A_0                  ; GO IF NOT
                        JMP     CLR_STG

                ;-----  GET THE INPUT BUFFER (SWITCH SETTINGS)

E30A_0:         MOV     AL,0FH                  ;       <><><><><><><><>
                OUT     MFG_PORT,AL             ;       <><  CHECKPOINT  0F  <><

                MOV     AL,PARITY_CHECK         ; SET BASE MEMORY PARITY
                OUT     DMA_PAGE+6,AL           ; USE AS TEMPORARY SAVE
                MOV     SP,OFFSET C2            ; SET RETURN ADDRESS
                JMP     STGTST_CNT
C30:            MOV     BX,AX                   ; SAVE FAILING BIT PATTERN
                JNZ     C31
                JMP     C33                     ; STORAGE OK, CONTINUE

                ;-----  TEMPORARY STACK FOR POST ROUTINES

C2              DW      C30
C8042A          DW      TST4_B
0BF_42A         DW      TST4_C
C8042B          DW      TST4_D
C8042C          DW      E30B
0BF_42B         DW      E30C

                ;--------------------------------------
                ; BASE 64K STORAGE FAILURE
                ;   DISPLAY THE CHECKPOINT (MFG CHECKPOINT)
                ;     AND XOR EXPECTED WITH READ IN MFG_PORT
                ;   DISPLAY CHECKPOINT IN MFG_PORT+3
                ;   DISPLAY XOR'D DATA HIGH BYTE MFG_PORT+1
                ;     LOW BYTE IN MFG PORT+2
                ;   A READ/WRITE SCOPE LOOP OF THE FIRST
                ;   WORD FOR POSSIBLE ADDRESS LINE FAILURES
                ;--------------------------------------

C31:            MOV     AL,BH                   ; SAVE HIGH BYTE
                OUT     MFG_PORT+1,AL
                MOV     AL,BL                   ; SAVE LOW BYTE
                OUT     MFG_PORT+2,AL

                ;-----  CHECK FOR VIDEO ROM

                MOV     CX,0C000H               ; START OF I/O ROM
M1:             MOV     DS,CX                   ; POINT TO SEGMENT
                SUB     BX,BX                   ; GET THE FIRST 2 LOCATIONS
                MOV     AX,[BX]
                JMP     $+2                     ; BUS SETTLE
                CMP     AX,0AA55H               ; IS THE VIDEO ROM PRESENT?
                JZ      Z5                      ; GO IF YES
                ADD     CX,080H                 ; POINT TO NEXT 2K BLOCK
                CMP     CX,0C800H               ; TOP OF VIDEO ROM AREA YET?
                JL      M1                      ; TRY AGAIN
                AND     CX,CX                   ; SET NON ZERO FLAG
Z5:             JNZ     C32                     ; GO IF NOT
                JMP     C31_0                   ; BYPASS ERROR DISPLAY IF VIDEO ROM

                ;--------------------------------------------------
                ; SET VIDEO MODE TO DISPLAY MEMORY ERROR
                ;       THIS ROUTINE INITIALIZES THE ATTACHMENT TO
                ;       TO DISPLAY FIRST 64K STORAGE ERRORS.
                ; BOTH COLOR AND MONOCHROME ATTACHMENTS ARE INITIALIZED.
                ;--------------------------------------------------

                ;-----  INITIALIZE COLOR/MONOCHROME

C32:            MOV     DX,3D8H                 ; CONTROL REGISTER ADDRESS OF COLOR CARD
                SUB     AL,AL                   ; MODE SET
                OUT     DX,AL

                MOV     DX,03B8H                ; CONTROL REGISTER ADDRESS OF B/W CARD
                MOV     AL,1                    ; MODE SET FOR CARD
                OUT     DX,AL                   ; RESET VIDEO
                SUB     DX,4                    ; BACK TO BASE REGISTER

M4              EQU     10H

                MOV     BX,OFFSET VIDEO_PARMS+M4*3  ; POINT TO VIDEO PARAMETERS
                ASSUME  DS:CODE
Z_2:            MOV     CX,M4                   ; COUNT OF MONOCHROME VIDEO PARAMETERS

                ;-----  BX POINTS TO CORRECT ROW OF INITIALIZATION TABLE

                XOR     AH,AH                   ; AH IS REGISTER NUMBER DURING LOOP

                ;-----  LOOP THROUGH TABLE, OUTPUTTING REGISTER ADDRESS, THEN VALUE FROM TABLE

M10:            MOV     AL,AH                   ; GET 6845 REGISTER NUMBER
                OUT     DX,AL
                INC     DX                      ; POINT TO DATA PORT
                INC     AH                      ; NEXT REGISTER VALUE
                MOV     AL,CS:[BX]              ; GET TABLE VALUE
                OUT     DX,AL                   ; OUT TO CHIP
                INC     BX                      ; NEXT IN TABLE
                DEC     DX                      ; BACK TO POINTER REGISTER
                LOOP    M10                     ; DO THE WHOLE TABLE
                MOV     AH,DL                   ; CHECK IF COLOR CARD DONE
                AND     AH,0F0H                 ; STRIP UNWANTED BITS
                CMP     AH,0D0H                 ; IS IT THE COLOR CARD?
                JZ      Z_3                     ; CONTINUE IF COLOR
                MOV     BX,OFFSET VIDEO_PARMS   ; POINT TO VIDEO PARAMETERS
                MOV     DX,3D4H                 ; COLOR BASE
                JMP     Z_2                     ; CONTINUE
```

```
                        ;-----  FILL REGEN AREA WITH BLANK
                        Z_3:    XOR     DI,DI                   ; SET UP POINTER FOR REGEN
0461 B8 B000                    MOV     AX,0B000H               ; SET UP ES TO VIDEO REGEN
0464 8E C0                      MOV     ES,AX

0466 B9 0800                    MOV     CX,2048                 ; NUMBER OF WORDS IN MONOCHROME CARD
0469 B8 0720                    MOV     AX,' '+7*H              ; FILL CHARACTER FOR ALPHA + ATTRIBUTE
046C F3/ AB                     REP     STOSW                   ; FILL THE REGEN BUFFER WITH BLANKS

046E 33 FF                      XOR     DI,DI                   ; CLEAR COLOR VIDEO BUFFER MEMORY
0470 BB B800                    MOV     BX,0B800H               ; SET UP ES TO COLOR VIDEO MEMORY
0473 8E C3                      MOV     ES,BX
0475 B9 2000                    MOV     CX,8192
0478 F3/ AB                     REP     STOSW                   ; FILL WITH BLANKS

                        ;-----  ENABLE VIDEO AND CORRECT PORT SETTING

047A BA 03B8                    MOV     DX,3B8H
047D B0 29                      MOV     AL,29H
047F EE                         OUT     DX,AL                   ; SET VIDEO ENABLE PORT

                        ;-----  SET UP OVERSCAN REGISTER

0480 42                         INC     DX                      ; SET OVERSCAN PORT TO A DEFAULT
0481 B0 30                      MOV     AL,30H                  ; VALUE 30H FOR ALL MODES EXCEPT 640X200
0483 EE                         OUT     DX,AL                   ; OUTPUT THE CORRECT VALUE TO 3D9 PORT

                        ;-----  ENABLE COLOR VIDEO AND CORRECT PORT SETTING

0484 BA 03D8                    MOV     DX,3D8H
0487 B0 28                      MOV     AL,28H
0489 EE                         OUT     DX,AL                   ; SET VIDEO ENABLE PORT

                        ;-----  SET UP OVERSCAN REGISTER

048A 42                         INC     DX                      ; SET OVERSCAN PORT TO A DEFAULT
048B B0 30                      MOV     AL,30H                  ; VALUE 30H FOR ALL MODES EXCEPT 640X200
048D EE                         OUT     DX,AL                   ; OUTPUT THE CORRECT VALUE TO 3D9 PORT

                        ;-----  DISPLAY FAILING CHECKPOINT AND

048E 8C C8                      MOV     AX,CS                   ; SET STACK SEGMENT TO CODE SEGMENT
0490 8E D0                      MOV     SS,AX

0492 BB B000                    MOV     BX,0B000H
0495 8E DB                      MOV     DS,BX                   ; SET DS TO B/W DISPLAY BUFFER

0497 B0 30                      MOV     AL,'0'                  ; DISPLAY BANK 000000
0499 B9 0006                    MOV     CX,6
049C 2B FF                      SUB     DI,DI                   ; START AT 0
049E 88 05                      MOV     [DI],AL                 ; WRITE TO DISPLAY REGEN BUFFER
04A0 47                         INC     DI                      ; POINT TO NEXT POSITION
04A1 47                         INC     DI
04A2 E2 FA                      LOOP    Z

04A4 80 FF B8                   CMP     BH,0B8H                 ; CHECK THAT COLOR BUFFER WRITTEN
04A7 74 0C                      JZ      Z_1
04A9 2B FF                      SUB     DI,DI                   ; POINT TO START OF BUFFER

04AB BF B0                      MOV     BH,0B0H
04AD 8E C3                      MOV     ES,BX                   ; ES = MONOCHROME
04AF BF B8                      MOV     BH,0B8H                 ; SET SEGMENT TO COLOR
04B1 8E DB                      MOV     DS,BX                   ; DS = COLOR
04B3 EB E2                      JMP     Z_0

                        ;-----  PRINT FAILING BIT PATTERN

04B5 B0 20              Z_1:    MOV     AL,' '                  ; DISPLAY A BLANK
04B7 88 05                      MOV     [DI],AL                 ; WRITE TO COLOR BUFFER
04B9 26: 88 05                  MOV     ES:[DI],AL              ; WRITE TO MONOCHROME REGEN BUFFER
04BC 47                         INC     DI                      ; POINT TO NEXT POSITION
04BD 47                         INC     DI
04BE E4 81                      IN      AL,MFG_PORT+1           ; GET THE HIGH BYTE OF FAILING PATTERN
04C0 B1 04                      MOV     CL,4                    ; SHIFT COUNT
04C2 D2 E8                      SHR     AL,CL                   ; NIBBLE SWAP
04C4 BC 057F R                  MOV     SP,OFFSET Z1_0
04C7 EB 19                      JMP     SHORT PR

04C9 E4 81              Z1:     IN      AL,MFG_PORT+1
04CB 24 0F                      AND     AL,0FH                  ; ISOLATE TO LOW NIBBLE
04CD BC 0581 R                  MOV     SP,OFFSET Z2_0
04D0 EB 12                      JMP     SHORT PR
04D2 E4 82              Z2:     IN      AL,MFG_PORT+2           ; GET THE HIGH BYTE OF FAILING PATTERN
04D4 B1 04                      MOV     CL,4                    ; SHIFT COUNT
04D6 D2 E8                      SHR     AL,CL                   ; NIBBLE SWAP
04D8 BC 0583 R                  MOV     SP,OFFSET Z3_0
04DB EB 07                      JMP     SHORT PR
04DD E4 82              Z3:     IN      AL,MFG_PORT+2
04DF 24 0F                      AND     AL,0FH                  ; ISOLATE TO LOW NIBBLE
04E1 BC 0585 R                  MOV     SP,OFFSET Z4_0          ; RETURN TO Z4!

                        ;-----  CONVERT AND PRINT

                        PR:     ADD     AL,090H                 ; CONVERT 00-0F TO ASCII CHARACTER
04E4 04 90                      DAA                             ; ADD FIRST CONVERSION FACTOR
04E6 27                         ADC     AL,040H                 ; ADJUST FOR NUMERIC AND ALPHA RANGE
04E7 14 40                      DAA                             ; ADD CONVERSION AND ADJUST LOW NIBBLE
04E9 27                                                         ; ADJUST HIGH NIBBLE TO ASCII RANGE

04EA 88 05                      MOV     [DI],AL                 ; WRITE TO COLOR BUFFER
04EC 26: 88 05                  MOV     ES:[DI],AL              ; WRITE TO MONOCHROME BUFFER
04EF 47                         INC     DI                      ; POINT TO NEXT POSITION
04F0 47                         INC     DI
04F1 C3                         RET

                        ;-----  DISPLAY 201 ERROR

04F2 B0 20              Z4:     MOV     AL,' '                  ; DISPLAY A BLANK
04F4 88 05                      MOV     [DI],AL                 ; WRITE TO DISPLAY REGEN BUFFER
04F6 26: 88 05                  MOV     ES:[DI],AL              ; WRITE TO MONOCHROME BUFFER
04F9 47                         INC     DI                      ; POINT TO NEXT POSITION
04FA 47                         INC     DI
04FB B0 32                      MOV     AL,'2'                  ; DISPLAY 201 ERROR
```

```
1790 04FD 88 05                 MOV     [DI],AL                 ; WRITE TO DISPLAY REGEN BUFFER
1791 04FF 26: 88 05             MOV     ES:[DI],AL              ; WRITE TO MONOCHROME BUFFER
1792 0502 47                    INC     DI                      ; POINT TO NEXT POSITION
1793 0503 47                    INC     DI
1794 0504 B0 30                 MOV     AL,'0'
1795 0506 88 05                 MOV     [DI],AL                 ; WRITE TO DISPLAY REGEN BUFFER
1796 0508 26: 88 05             MOV     ES:[DI],AL              ; WRITE TO MONOCHROME BUFFER
1797 050B 47                    INC     DI                      ; POINT TO NEXT POSITION
1798 050C 47                    INC     DI
1799 050D B0 31                 MOV     AL,'1'
1800 050F 88 05                 MOV     [DI],AL                 ; WRITE TO DISPLAY REGEN BUFFER
1801 0511 26: 88 05             MOV     ES:[DI],AL              ; WRITE TO MONOCHROME BUFFER
1802
1803                            ;-----  ROLL ERROR CODE IN MFG_PORT --> FIRST THE CHECKPOINT
1804
1805 0514 B0 DD                 C31_0:  MOV     AL,0DDH                 ;       <><><><><><><><><>
1806 0516 E6 80                         OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  DD <><>
1807 0518 E6 83                         OUT     MFG_PORT+3,AL           ; ALSO DISPLAY CHECK POINT IN PORT 83
1808 051A 2B C9                         SUB     CX,CX
1809 051C                       C31_A:
1810
1811 051C 2B C0                         SUB     AX,AX                   ; SETUP SEGMENT
1812 051E 8E D8                         MOV     DS,AX
1813 0520 B8 AA55                       MOV     AX,0AA55H               ; WRITE AN AA55
1814 0523 2B FF                         SUB     DI,DI
1815 0525 89 05                         MOV     [DI],AX
1816 0527 8B 05                         MOV     AX,[DI]                 ; READ THE FIRST WORD
1817 0529 E2 F1                         LOOP    C31_A                   ; DISPLAY CHECKPOINT LONGER
1818 052B                       C31_B:
1819 052B 89 05                         MOV     [DI],AX
1820 052D 8B 05                         MOV     AX,[DI]
1821 052F E2 FA                         LOOP    C31_B
1822 0531                       C31_C:
1823 0531 89 05                         MOV     [DI],AX
1824 0533 8B 05                         MOV     AX,[DI]
1825 0535 E2 FA                         LOOP    C31_C
1826 0537                       C31_D:
1827 0537 89 05                         MOV     [DI],AX
1828 0539 8B 05                         MOV     AX,[DI]
1829 053B E2 FA                         LOOP    C31_D
1830 053D                       C31_E:
1831 053D 89 05                         MOV     [DI],AX
1832 053F 8B 05                         MOV     AX,[DI]
1833 0541 E2 FA                         LOOP    C31_E
1834
1835                            ;-----  ROLL ERROR CODE IN MFG_PORT --> NEXT THE HIGH BYTE
1836
1837 0543 E4 81                         IN      AL,MFG_PORT+1           ; XOR OF FAILING BIT PATTERN
1838 0545 E6 80                         OUT     MFG_PORT,AL             ; HIGH BYTE
1839 0547                       C31_G:
1840 0547 B8 AA55                       MOV     AX,0AA55H               ; WRITE AN AA55
1841 054A 89 05                         MOV     [DI],AX
1842 054C 8B 05                         MOV     AX,[DI]                 ; READ THE FIRST WORD
1843 054E E2 F7                         LOOP    C31_G
1844 0550                       C31_H:
1845 0550 89 05                         MOV     [DI],AX
1846 0552 8B 05                         MOV     AX,[DI]
1847 0554 E2 FA                         LOOP    C31_H
1848 0556                       C31_I:
1849 0556 89 05                         MOV     [DI],AX
1850 0558 8B 05                         MOV     AX,[DI]
1851 055A E2 FA                         LOOP    C31_I
1852
1853                            ;-----  ROLL ERROR CODE IN MFG_PORT --> THEN THE LOW BYTE
1854
1855 055C E4 82                         IN      AL,MFG_PORT+2           ; LOW BYTE
1856 055E E6 80                         OUT     MFG_PORT,AL
1857 0560 B8 AA55                       MOV     AX,0AA55H               ; WRITE AN AA55
1858 0563 2B FF                 C31_K:  SUB     DI,DI
1859 0565 89 05                         MOV     [DI],AX
1860 0567 8B 05                         MOV     AX,[DI]                 ; READ THE FIRST WORD
1861 0569 E2 F8                         LOOP    C31_K
1862 056B                       C31_L:
1863 056B 89 05                         MOV     [DI],AX
1864 056D 8B 05                         MOV     AX,[DI]
1865 056F E2 FA                         LOOP    C31_L
1866 0571                       C31_M:
1867 0571 89 05                         MOV     [DI],AX
1868 0573 8B 05                         MOV     AX,[DI]
1869 0575 E2 FA                         LOOP    C31_M
1870 0577                       C31_N:
1871 0577 89 05                         MOV     [DI],AX
1872 0579 8B 05                         MOV     AX,[DI]
1873 057B E2 FA                         LOOP    C31_N
1874 057D EB 95                         JMP     C31_0                   ; DO AGAIN
1875
1876 057F 04C9 R                Z1_0    DW      Z1                      ; TEMPORARY STACK
1877 0581 04D2 R                Z2_0    DW      Z2                      ; TEMPORARY STACK
1878 0583 04DD R                Z3_0    DW      Z3                      ; TEMPORARY STACK
1879 0585 04F2 R                Z4_0    DW      Z4                      ; TEMPORARY STACK
1880
1881
1882                            ;-----  CLEAR STORAGE ENTRY
1883
1884
1885 0587                       CLR_STG:
1886                                    ASSUME  DS:DATA
1887 0587 F3/ AB                        REP     STOSW                   ; STORE 32K WORDS OF 0000
1888 0589 B8 ---- R                     MOV     AX,DATA                 ; RESTORE DATA SEGMENT
1889 058C 8E D8                         MOV     DS,AX
1890 058E 89 1E 0072 R                  MOV     @RESET_FLAG,BX          ; RESTORE RESET FLAG
1891
1892                            ;-----  SETUP STACK SEGMENT AND SP
1893
1894 0592                       C33:
1895 0592 B8 ---- R                     MOV     AX,DATA                 ; SET DATA SEGMENT
1896 0595 8E D8                         MOV     DS,AX
1897 0597 BC 0000                       MOV     SP,POST_SS              ; GET STACK VALUE
1898 059A 8E D4                         MOV     SS,SP                   ; SET THE STACK UP
1899 059C BC 8000                       MOV     SP,POST_SP              ; STACK IS READY TO GO
1900
1901                            ;-----  INITIALIZE DISPLAY ROW COUNT
1902
1903 059F C6 06 0084 R 18               MOV     @ROWS,25-1              ; SET ROWS FOR PRINT SCREEN DEFAULT
```

```
                                
05A4 B0 11                      MOV     AL,11H                  ;       <><><><><><><><><><>
05A6 E6 80                      OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  11 <><>
                                
                        ;-----  VERIFY SPEED/REFRESH CLOCK RATES  ( ERROR = 1 LONG AND 1 SHORT BEEP )
                                
                                XOR     BL,BL                   ; CLEAR REFRESH CYCLE REPEAT COUNT
                                XOR     CX,CX                   ; INITIALIZE SPEED RATE REGISTER
                                EVEN                            ; PLACE ON EVEN WORD BOUNDARY
                        C34:
05AC E4 61                      IN      AL,PORT_B               ; READ REFRESH BIT REGISTER
05AE A8 10                      TEST    AL,REFRESH_BIT          ; MASK FOR BIT
05B0 E1 FA                      LOOPZ   C34                     ; DECREMENT LOOP COUNTER TILL ON
                        C35:
05B2 E4 61                      IN      AL,PORT_B               ; READ REFRESH BIT REGISTER
05B4 A8 10                      TEST    AL,REFRESH_BIT          ; MASK FOR BIT
05B6 E0 FA                      LOOPNZ  C35                     ; DECREMENT LOOP COUNTER TILL OFF
                                
05B8 FE CB                      DEC     BL                      ; DECREMENT REFRESH CYCLE REPEAT COUNT
05BA 75 F0                      JNZ     C34                     ; REPEAT TILL CYCLE COUNT DONE
                                
05BC 81 F9 F780                 CMP     CX,RATE_UPPER           ; CHECK FOR RATE BELOW UPPER LIMIT
05C0 73 07                      JAE     C36                     ; SKIP ERROR BEEP IF BELOW MAXIMUM
                        C36E:
05C2 BA 0101                    MOV     DX,0101H                ; GET BEEP COUNTS FOR REFRESH ERROR
05C5 E8 0000 E                  CALL    ERR_BEEP                ; CALL TO POST ERROR BEEP ROUTINES
05C8 F4                         HLT                             ; HALT SYSTEM - BAD REFRESH RATE
                        C36:
05C9                            
05C9 81 F9 F9FD                 CMP     CX,RATE_LOWER           ; CHECK FOR RATE ABOVE LOWER LIMIT
05CD 77 F3                      JA      C36E                    ; GO TO ERROR BEEP IF BELOW MINIMUM
                                
                        ;-----  GET THE INPUT BUFFER (SWITCH SETTINGS)
                                
05CF E4 82                      IN      AL,DMA_PAGE+1           ; GET THE SWITCH SETTINGS
05D1 24 F8                      AND     AL,KEY_BD_INHIB+DSP_JMP+MFG_LOOP+BASE_MEM+BASE_MEM8 ; STRIP BITS
05D3 A2 0012 R                  MOV     @MFG_TST,AL             ; SAVE SETTINGS
05D6 2A C0                      SUB     AL,AL                   ; RESET DMA_PAGE
05D8 E6 82                      OUT     DMA_PAGE+1,AL
                                
                        ;---------------------------------------
                        ; TEST.11A                             :
                        ;       VERIFY 286 LGDT/SGDT LIDT/SIDT  :
                        ;       INSTRUCTIONS                    :
                        ; DESCRIPTION                          :
                        ;       LOAD GDT AND IDT REGISTERS WITH :
                        ;       AA,55,00 AND VERIFY CORRECT.    :
                        ;---------------------------------------
                                
                        ;-----  VERIFY STATUS INDICATE COMPATIBILITY (REAL) MODE
                                
                                SMSW    AX                      ; GET THE CURRENT STATUS WORD
05DA 0F 01 E0           +       DB      00FH,001H,0E0H
05DD A9 000F                    TEST    AX,0FH                  ; PE/MP/EM/TS BITS SHOULD BE ZERO
05E0 75 34                      JNZ     ERR_PROT                ; GO IF STATUS NOT REAL MODE
                                
                        ;-----  TEST PROTECTED MODE REGISTERS
                                
05E2 B0 12                      MOV     AL,12H                  ;       <><><><><><><><><><>
05E4 E6 80                      OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  12 <><>
                                
05E6 1E                         PUSH    DS                      ; SET ES TO SAME SEGMENT AS DS
05E7 07                         POP     ES
05E8 BF D0A0                    MOV     DI,SYS_IDT_LOC          ; USE THIS AREA TO BUILD TEST PATTERN
05EB B9 0003                    MOV     CX,3
05EE B8 AAAA                    MOV     AX,0AAAAH               ; FIRST PATTERN
05F1 E8 0619 R                  CALL    WRT_PAT
05F4 B8 5555                    MOV     AX,05555H
05F7 E8 0619 R                  CALL    WRT_PAT                 ; WRITE NEXT PATTERN
05FA 2B C0                      SUB     AX,AX                   ; WRITE 0
05FC E8 0619 R                  CALL    WRT_PAT
                                
                        ;-----  TEST 286 CONTROL FLAGS
                                
05FF FD                         STD                             ; SET DIRECTION FLAG FOR DECREMENT
0600 9C                         PUSHF                           ; GET THE FLAGS
0601 58                         POP     AX
0602 A9 0200                    TEST    AX,0200H                ; INTERRUPT FLAG SHOULD BE OFF
0605 75 0F                      JNZ     ERR_PROT                ; GO IF NOT
0607 A9 0400                    TEST    AX,0400H                ; CHECK DIRECTION FLAG
060A 74 0A                      JZ      ERR_PROT                ; GO IF NOT SET
060C FC                         CLD                             ; CLEAR DIRECTION FLAG
060D 9C                         PUSHF                           ; INSURE DIRECTION FLAG IS RESET
060E 58                         POP     AX
060F A9 0400                    TEST    AX,0400H
0612 75 02                      JNZ     ERR_PROT                ; GO IF NOT
                                
0614 EB 3D                      JMP     SHORT C37A              ; TEST OK CONTINUE
                        ERR_PROT:
0616 F4                         HLT                             ; PROTECTED MODE REGISTER FAILURE
0617 EB FD                      JMP     SHORT ERR_PROT          ; INSURE NO BREAKOUT OF HALT
                                
                        ;-----  WRITE TO 286 REGISTERS
                                
                        WRT_PAT:MOV     CX,3
                                REP     STOSW                   ; STORE 6 BYTES OF PATTERN
                                MOV     BP,SYS_IDT_LOC
                                SEGOV   ES                      ; LOAD THE IDT
0621 26                 +       DB      026H
                                LIDT    [BP]                    ; REGISTER FROM THIS AREA
0622 0F                 +       DB      00FH
0623                    + ??0001 LABEL  BYTE
0623 8B 5E 00           +       MOV     BX,WORD PTR [BP]
0626                    + ??0002 LABEL  BYTE
0623                    +       ORG     OFFSET CS:??0001
0623 01                 +       DB      001H
0624                    +       ORG     OFFSET CS:??0002
0626 BD D0A0                    MOV     BP,SYS_IDT_LOC
                                SEGOV   ES                      ; LOAD THE GDT
0629 26                 +       DB      026H
                                LGDT    [BP]                    ; FROM THE SAME AREA
062A 0F                 +       DB      00FH
062B                    + ??0004 LABEL  BYTE
062B 8B 56 00           +       MOV     DX,WORD PTR [BP]
062E                    + ??0005 LABEL  BYTE
```

SECTION 5

```
2018 062B                       +       ORG     OFFSET CS:??0004
2019 062B 01                    +       DB      001H
2020 062E                       +       ORG     OFFSET CS:??0005
2021
2022                            ;-----  READ AND VERIFY 286 REGISTERS
2023
2024 062E BD D8A0                       MOV     BP,GDT_LOC              ; STORE THE REGISTERS HERE
2025                                    SEGOV   ES
2026 0631 26                    +       DB      026H
2027                                    SIDT    [BP]                    ; GET THE IDT REGISTERS
2028 0632 0F                    +       DB      00FH
2029 0633                       + ??0007 LABEL  BYTE
2030 0633 8B 4E 00              +       MOV     CX,[BP]
2031 0636                       + ??0008 LABEL  BYTE
2032 0633                       +       ORG     OFFSET CS:??0007
2033 0633 01                    +       DB      001H
2034 0636                       +       ORG     OFFSET CS:??0008
2035 0636 BD D8A6                       MOV     BP,GDT_LOC+6
2036                                    SEGOV   ES
2037 0639 26                    +       DB      026H
2038                                    SGDT    [BP]                    ; GET THE GDT REGISTERS
2039 063A 0F                    +       DB      00FH
2040 063B                       + ??000A LABEL  BYTE
2041 063B 03 46 00              +       ADD     AX,[BP]
2042 063E                       + ??000B LABEL  BYTE
2043 063B                       +       ORG     OFFSET CS:??000A
2044 063B 01                    +       DB      001H
2045 063E                       +       ORG     OFFSET CS:??000B
2046 063E BF D0A0                       MOV     DI,SYS_IDT_LOC
2047 0641 8B 05                         MOV     AX,[DI]                 ; GET THE PATTERN WRITTEN
2048 0643 B9 0005                       MOV     CX,5                    ; CHECK ALL REGISTERS
2049 0646 BE D8A0                       MOV     SI,GDT_LOC              ; POINT TO THE BEGINNING
2050 0649 26: 3B 04             C37B:   CMP     AX,ES:[SI]
2051 064C 75 C8                         JNZ     ERR_PROT                ; HALT IF ERROR
2052 064E 46                            INC     SI                      ; POINT TO NEXT WORD
2053 064F 46                            INC     SI
2054 0650 E2 F7                         LOOP    C37B                    ; CONTINUE TILL DONE
2055 0652 C3                            RET
2056
2057                            ;-------------------------------------------------------
2058                            ;       INITIALIZE THE 8259 INTERRUPT #1 CONTROLLER CHIP :
2059                            ;-------------------------------------------------------
2060 0653                       C37A:
2061 0653 2A C0                         SUB     AL,AL                   ; RESET MATH PROCESSOR
2062 0655 E6 F1                         OUT     X287+1,AL
2063 0657 B0 11                         MOV     AL,11H                  ; ICW1 - EDGE, MASTER, ICW4
2064 0659 E6 20                         OUT     INTA00,AL
2065 065B EB 00                         JMP     $+2                     ; I/O DELAY
2066 065D B0 08                         MOV     AL,8                    ; SETUP ICW2 - INTERRUPT TYPE 8 (8-F)
2067 065F E6 21                         OUT     INTA01,AL
2068 0661 EB 00                         JMP     $+2                     ; I/O DELAY
2069
2070 0663 B0 04                         MOV     AL,04H                  ; SETUP ICW3 - MASTER LEVEL 2
2071 0665 E6 21                         OUT     INTA01,AL
2072 0667 EB 00                         JMP     $+2                     ; I/O DELAY
2073 0669 B0 01                         MOV     AL,01H                  ; SETUP ICW4 - MASTER,8086 MODE
2074 066B E6 21                         OUT     INTA01,AL
2075 066D EB 00                         JMP     $+2                     ; I/O DELAY
2076 066F B0 FF                         MOV     AL,0FFH                 ; MASK ALL INTERRUPTS OFF
2077 0671 E6 21                         OUT     INTA01,AL               ; (VIDEO ROUTINE ENABLES INTERRUPTS)
2078
2079                            ;-------------------------------------------------------
2080                            ;       INITIALIZE THE 8259 INTERRUPT #2 CONTROLLER CHIP  :
2081                            ;-------------------------------------------------------
2082
2083 0673 B0 13                         MOV     AL,13H                  ;       <><><><><><><><>
2084 0675 E6 80                         OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  13 <><>
2085
2086 0677 B0 11                         MOV     AL,11H                  ; ICW1 - EDGE, SLAVE ICW4
2087 0679 E6 A0                         OUT     INTB00,AL
2088 067B EB 00                         JMP     $+2                     ; I/O DELAY
2089 067D B0 70                         MOV     AL,INT_TYPE             ; SETUP ICW2 - INTERRUPT TYPE 70 (70-7F)
2090 067F E6 A1                         OUT     INTB01,AL
2091 0681 B0 02                         MOV     AL,02H                  ; SETUP ICW3 - SLAVE LEVEL 2
2092 0683 EB 00                         JMP     $+2
2093 0685 E6 A1                         OUT     INTB01,AL
2094 0687 EB 00                         JMP     $+2                     ; I/O DELAY
2095 0689 B0 01                         MOV     AL,01H                  ; SETUP ICW4 - 8086 MODE, SLAVE
2096 068B E6 A1                         OUT     INTB01,AL
2097 068D EB 00                         JMP     $+2                     ; I/O DELAY
2098 068F B0 FF                         MOV     AL,0FFH                 ; MASK ALL INTERRUPTS OFF
2099 0691 E6 A1                         OUT     INTB01,AL
2100
2101                            ;-----  SET UP THE INTERRUPT VECTORS TO TEMPORARY INTERRUPT
2102
2103 0693 B0 14                         MOV     AL,14H                  ;       <><><><><><><><>
2104 0695 E6 80                         OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  14 <><>
2105
2106 0697 B9 0078                       MOV     CX,78H                  ; FILL ALL INTERRUPT LOCATIONS
2107 069A 2B FF                         SUB     DI,DI                   ; FIRST INTERRUPT LOCATION
2108 069C 8E C7                         MOV     ES,DI                   ; SET (ES) ALSO
2109 069E B6 0000 E             D3:     MOV     AX,OFFSET D11           ; GET ADDRESS OF INTERRUPT OFFSET
2110 06A1 AB                            STOSW                           ; PLACE IN INTERRUPT VECTOR LOCATION
2111 06A2 8C C8                         MOV     AX,CS                   ; GET THE CURRENT CODE SEGMENT
2112 06A4 AB                            STOSW                           ; PLACE CODE SEGMENT IN VECTOR LOCATION
2113 06A5 E2 F7                         LOOP    D3
2114
2115                            ;-----  ESTABLISH BIOS SUBROUTINE CALL INTERRUPT VECTORS
2116
2117 06A7 B0 15                         MOV     AL,15H                  ;       <><><><><><><><>
2118 06A9 E6 80                         OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  15 <><>
2119
2120
2121 06AB BF 0040 R                     MOV     DI,OFFSET @VIDEO_INT    ; SET VIDEO INTERRUPT AREA
2122 06AE 0E                            PUSH    CS
2123 06AF 1F                            POP     DS                      ; SET UP ADDRESS OF VECTOR TABLE
2124                            ;       MOV     AX,DS                   ; SET AX=SEGMENT
2125 06B0 BE 0010 E                     MOV     SI,OFFSET VECTOR_TABLE+16 ;START WITH VIDEO ENTRY
2126 06B3 B9 0010                       MOV     CX,16
2127
2128 06B6 A5                    D3A:    MOVSW                           ; MOVE VECTOR TABLE TO LOW MEMORY
2129 06B7 47                            INC     DI
2130 06B8 47                            INC     DI                      ; SKIP SEGMENT POINTER
2131 06B9 E2 FB                         LOOP    D3A
```

```
                        ;--------------------------------------
                        ; TEST.12                             :
                        ;       VERIFY CMOS CHECKSUM/BATTERY OK :
                        ; DESCRIPTION                         :
                        ;       DETERMINE IF CONFIG RECORD    :
                        ;       CAN BE USED FOR INITIALIZATION. :
                        ;--------------------------------------

                                ASSUME  DS:DATA
06BB E8 0000 E                  CALL    DDS                     ; SET THE DATA SEGMENT

06BE B0 16                      MOV     AL,16H                  ;       <><><><><><><><><>
06C0 E6 80                      OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  16 <><>

                        ;-----  IS THE BATTERY LOW THIS POWER UP?

06C2 B0 8D                      MOV     AL,CMOS_REG_D+NMI       ; CHECK BATTERY CONDITION
06C4 E8 0000 E                  CALL    CMOS_READ               ; READ THE BATTERY STATUS
06C7 A8 80                      TEST    AL,10000000B            ; IS THE BATTERY LOW?
06C9 74 0B                      JZ      CMOS1A                  ; ERROR IF YES

06CB B0 8E                      MOV     AL,CMOS_DIAG+NMI        ; GET THE OLD STATUS
06CD E8 0000 E                  CALL    CMOS_READ               ; FROM DIAGNOSTIC STATUS BYTE
06D0 A8 80                      TEST    AL,BAD_BAT              ; HAS CUSTOMER SETUP BEEN EXECUTED?
06D2 74 15                      JZ      CMOS1                   ; GO CHECK CHECKSUM IF YES

06D4 EB 64                      JMP     SHORT CMOS4             ; CONTINUE WITHOUT CONFIGURATION

                        ;-----  SET DEFECTIVE BATTERY FLAG

06D6 B0 17              CMOS1A: MOV     AL,17H                  ;       <><><><><><><><><>
06D8 E6 80                      OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  17 <><>

06DA B6 8E8E                    MOV     AX,X*(CMOS_DIAG+NMI)    ; CMOS DIAGNOSTIC STATUS BYTE
06DD E8 0000 E                  CALL    CMOS_READ               ; GET THE CURRENT STATUS
06E0 0C 80                      OR      AL,BAD_BAT              ; SET THE DEAD BATTERY FLAG
06E2 86 C4                      XCHG    AL,AH                   ; SAVE
06E4 E8 0000 E                  CALL    CMOS_WRITE              ; OUTPUT THE STATUS
06E7 EB 51                      JMP     SHORT  CMOS4            ; GO TO MINIMUM CONFIGURATION

                        ;-----  VERIFY CHECKSUM

06E9 B6 8E8E            CMOS1:  MOV     AX,X*(CMOS_DIAG+NMI)    ; CLEAR OLD STATUS
06EC E8 0000 E                  CALL    CMOS_READ               ; GET THE CURRENT STATUS
06EF 81 3E 0072 R 1234          CMP     @RESET_FLAG,1234H       ; IS THIS A SOFT RESET
06F5 75 04                      JNZ     CMOS1_A                 ; GO IF NOT

06F7 24 10                      AND     AL,W_MEM_SIZE           ; CLEAR ALL BUT THE CMOS/POR MEMORY SIZE
06F9 EB 02                      JMP     SHORT CMOS1_B
06FB                    CMOS1_A:
06FB 2A C0                      SUB     AL,AL                   ; CLEAR STATUS IF POWER ON RESET
06FD                    CMOS1_B:
06FD 86 C4                      XCHG    AL,AH                   ; SAVE THE CURRENT STATUS
06FF E8 0000 E                  CALL    CMOS_WRITE

0702 2B DB                      SUB     BX,BX
0704 2B C9                      SUB     CX,CX
0706 B1 90                      MOV     CL,CMOS_DISKETTE+NMI    ; SET START OF CMOS CHECKSUMED AREA
0708 B5 AE                      MOV     CH,CMOS_CKSUM_HI+NMI    ; SET END OF CMOS CHECKSUMED AREA +1
                                                                ;  (FIRST BYTE OF CHECKSUM)
070A 8A C1              CMOS2:  MOV     AL,CL
070C E8 0000 E                  CALL    CMOS_READ               ; ADDRESS THE BEGINNING
070F 2A E4                      SUB     AH,AH                   ; INSURE AH=0
0711 03 D8                      ADD     BX,AX                   ; ADD TO CURRENT VALUE
0713 FE C1                      INC     CL                      ; POINT TO NEXT BYTE ADDRESS IN CMOS
0715 3A E9                      CMP     CH,CL                   ; FINISHED?  (AT CHECKSUM BYTE HIGH)
0717 75 F1                      JNZ     CMOS2                   ; GO IF NOT
0719 0B DB                      OR      BX,BX                   ; BX MUST NOT BE 0
071B 74 10                      JZ      CMOS3                   ; CMOS BAD IF CHECKSUM=0
071D B0 AE                      MOV     AL,CMOS_CKSUM_HI+NMI    ; GET THE CHECK SUM HIGH BYTE
071F E8 0000 E                  CALL    CMOS_READ               ; FIRST BYTE OF CHECKSUM
0722 8A E0                      MOV     AH,AL                   ; SAVE IT
0724 B0 AF                      MOV     AL,CMOS_CKSUM_LO+NMI    ; SECOND BYTE OF CHECKSUM
0726 E8 0000 E                  CALL    CMOS_READ
0729 3B C3                      CMP     AX,BX                   ; IS THE CHECKSUM OK
072B 74 0D                      JZ      CMOS4                   ; GO IF YES

                        ;-----  SET CMOS CHECKSUM ERROR

072D B6 8E8E            CMOS3:  MOV     AX,X*(CMOS_DIAG+NMI)    ; ADDRESS DIAGNOSTIC STATUS
0730 E8 0000 E                  CALL    CMOS_READ               ; GET THE CURRENT STATUS
0733 0C 40                      OR      AL,BAD_CKSUM            ; SET BAD CHECKSUM FLAG
0735 86 C4                      XCHG    AL,AH                   ; SAVE IT
0737 E8 0000 E                  CALL    CMOS_WRITE              ; SET FLAG

                        ;-----  INSURE CMOS DIVIDERS SET

073A                    CMOS4:
073A B6 8A8A                    MOV     AX,X*(CMOS_REG_A+NMI)   ; ADDRESS CMOS REGISTER A
073D E8 0000 E                  CALL    CMOS_READ               ; GET CURRENT DIVISORS
0740 24 0F                      AND     AL,0FH                  ; LOOK AT PERIODIC RATE BITS
0742 75 07                      JNZ     CMOS9                   ; EXIT IF SET TO SOMETHING USEFUL

0744 B0 26                      MOV     AL,26H                  ; ELSE SET THE STANDARD DEFAULT USED BY
0746 86 C4                      XCHG    AL,AH                   ;  BIOS FOR THE 976.56 US RATE
0748 E8 0000 E                  CALL    CMOS_WRITE              ;  FOR THE PERIODIC CLOCK
074B                    CMOS9:
074B B0 18                      MOV     AL,18H                  ;       <><><><><><><><><>
074D E6 80                      OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  18 <><>

                        ;-----  ENABLE PROTECTED MODE

074F E4 61                      IN      AL,PORT_B               ; DISABLE MEMORY AND I/O PARITY CHECKS
0751 0C 0C                      OR      AL,RAM_PAR_OFF
0753 E6 61                      OUT     PORT_B,AL

                        ;-----  SET RETURN ADDRESS BYTE IN CMOS

0755 B0 19                      MOV     AL,19H                  ;       <><><><><><><><><>
0757 E6 80                      OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  19 <><>

0759 B6 018F                    MOV     AX,!*H+(CMOS_SHUT_DOWN+NMI)   ; SET THE RETURN ADDRESS FOR
075C E8 0000 E                  CALL    CMOS_WRITE              ; THE FIRST SHUTDOWN RETURN ADDRESS
```

SECTION 5

```
                                ;--------------------------------------------------------------------------------
                                ; TEST.13                                                                       :
                                ;  PROTECTED MODE TEST AND MEMORY SIZE DETERMINE  ( 0 --> 640K )               :
                                ;                                                                              :
                                ; DESCRIPTION:                                                                 :
                                ;     THIS ROUTINE RUNS IN PROTECTED MODE IN ORDER TO ADDRESS ALL OF STORAGE.   :
                                ;     IT CHECKS THE MACHINE STATUS WORD (MSW) FOR PROTECTED MODE AND THE BASE   :
                                ;     MEMORY SIZE IS DETERMINED AND SAVED.  BIT 4 OF THE CMOS DIAGNOSTIC       :
                                ;     STATUS BYTE IS SET IF 512K --> 640K MEMORY IS INSTALLED.                 :
                                ;     DURING A POWER UP SEQUENCE THE MEMORY SIZE DETERMINE IS DONE WITH        :
                                ;     PLANAR AND I/O PARITY CHECKS DISABLED.  DURING A SOFT RESET THE MEMORY   :
                                ;     SIZE DETERMINE WILL CHECK FOR PARITY ERRORS.                             :
                                ;--------------------------------------------------------------------------------

                                ;-----  INSURE PROTECTED MODE

 0784 0F 01 E0                  +       SMSW    AX                      ; GET THE MACHINE STATUS WORD
                                        DB      00FH,001H,0E0H
 0787 A9 0001                           TEST    AX,VIRTUAL_ENABLE       ; ARE WE IN PROTECTED MODE
 078A 75 0C                             JNZ     VIR_OK

 078C B8 088F                   SHUT_8: MOV     AX,8*H+(CMOS_SHUT_DOWN+NMI)     ; SET THE RETURN ADDRESS
 078F E8 0000 E                         CALL    CMOS_WRITE              ; AND SET SHUTDOWN 8
 0792 E9 0000 E                         JMP     PROC_SHUTDOWN           ; CAUSE A SHUTDOWN

                                ;-----  VIRTUAL MODE ERROR HALT

 0795 F4                        SHUT8:  HLT
 0796 EB FD                             JMP     SHUT8                   ; ERROR HALT

                                ;-----  64K SEGMENT LIMIT

 0798 C7 06 0048 FFFF           VIR_OK: MOV     DS:ES_TEMP.SEG_LIMIT,MAX_SEG_LEN

                                ;-----  CPL0, DATA ACCESS RIGHTS

 079E C6 06 004D 93                     MOV     BYTE PTR DS:(ES_TEMP.DATA_ACC_RIGHTS),CPL0_DATA_ACCESS

                                ;-----  START WITH SEGMENT ADDRESS 01-0000 (SECOND 64K)

 07A3 C6 06 004C 01                     MOV     BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE),01H
 07A8 C7 06 004A 0000                   MOV     DS:ES_TEMP.BASE_LO_WORD,0H

 07AE B0 1B                             MOV     AL,1BH                  ;       <><><><><><><><><>
 07B0 E6 80                             OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  1B <><>

 07B2 BB 0040                           MOV     BX,16*4                 ; SET THE FIRST 64K DONE

                                ;-----  START STORAGE SIZE/CLEAR

 07B5                           NOT_DONE:
 07B5 6A 48                             PUSH    BYTE PTR ES_TEMP        ; POINT ES TO DATA
 07B7 07                                POP     ES                      ; POINT TO SEGMENT TO TEST
 07B8 E8 07D4 R                         CALL    HOW_BIG                 ; DO THE FIRST 64K
 07BB 74 03                             JZ      NOT_FIN                 ; CHECK IF TOP OF MEMORY
 07BD E9 0872 R                         JMP     DONE

 07C0                           NOT_FIN:
 07C0 83 C3 40                          ADD     BX,16*4                 ; BUMP MEMORY COUNT BY 64K

                                ;-----  DO NEXT 64K (0X0000) BLOCK

 07C3 FE 06 004C                        INC     BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE)

                                ;-----  CHECK FOR END OF FIRST 640K (END OF BASE MEMORY)

 07C7 80 3E 004C 0A                     CMP     BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE),0AH
 07CC 75 E7                             JNZ     NOT_DONE                ; GO IF NOT
 07CE E8 084F R                         CALL    HOW_BIG_END             ; GO SET MEMORY SIZE
 07D1 E9 0872 R                         JMP     DONE

                                ;-----  FILL/CHECK LOOP

 07D4                           HOW_BIG:
 07D4 2B FF                             SUB     DI,DI
 07D6 B8 AA55                           MOV     AX,0AA55H               ; TEST PATTERN
 07D9 8B C8                             MOV     CX,AX                   ; SAVE PATTERN
 07DB 26: 89 05                         MOV     ES:[DI],AX              ; WRITE PATTERN TO MEMORY
 07DE B0 0F                             MOV     AL,0FH                  ; PUT SOMETHING IN AL
 07E0 26: 8B 05                         MOV     AX,ES:[DI]              ; GET PATTERN
 07E3 26: 89 05                         MOV     ES:[DI],AX              ; INSURE NO PARITY I/O CHECK
 07E6 33 C1                             XOR     AX,CX                   ; COMPARE PATTERNS
 07E8 75 65                             JNZ     HOW_BIG_END             ; GO END IF NO COMPARE

 07EA 1E                                PUSH    DS
 07EB 6A 18                             PUSH    BYTE PTR RSDA_PTR       ; POINT TO SYSTEM DATA AREA
 07ED 1F                                POP     DS                      ; GET (DS:)

                                ;-----  IS THIS A SOFT RESET

 07EE 81 3E 0072 R 1234                 CMP     @RESET_FLAG,1234H       ; SOFT RESET
 07F4 1F                                POP     DS                      ; RESTORE DS
 07F5 75 36                             JNZ     HOW_BIG_2               ; GO IF NOT SOFT RESET
```

```
                        ;-----  INSURE NO PARITY WITH PARITY BITS OFF

                                MOV     WORD PTR ES:[DI],0101H  ; TURN OFF BOTH PARITY BITS

                                IN      AL,PORT_B
                                OR      AL,RAM_PAR_OFF          ; TOGGLE PARITY CHECK ENABLES
                                OUT     PORT_B,AL
                                AND     AL,RAM_PAR_ON
                                OUT     PORT_B,AL
                                PUSH    BYTE PTR 0FFH           ; PLACE 0FFFFH IN STACK (BUS BITS ON)
                                POP     AX                      ; DELAY - CAUSING BUS BITS ON
                                MOV     AX,ES:[DI]              ; CHECK PARITY

                                IN      AL,PORT_B               ; CHECK FOR PLANAR OR I/O PARITY CHECK
                                AND     AL,PARITY_ERR
                                MOV     ES:[DI],AX              ; CLEAR POSSIBLE PARITY ERROR
                                JNZ     HOW_BIG_END             ; GO IF PLANAR OR I/O PARITY CHECK

                        ;-----  CHECK ALL BITS WRITE OK

                                MOV     WORD PTR ES:[DI],0FFFFH ; TURN ON ALL BITS
                                MOV     AX,ES:[DI]              ; CHECK FOR FFFFH
                                PUSH    AX                      ; SAVE RESULTS
                                IN      AL,PORT_B               ; CHECK FOR PLANAR OR I/O PARITY CHECK
                                AND     AL,PARITY_ERR
                                MOV     ES:[DI],AX              ; CLEAR POSSIBLE PARITY ERROR
                                POP     AX                      ; GET RESULTS
                                JNZ     HOW_BIG_END             ; GO IF PARITY CHECK
                                CMP     AX,0FFFFH
                                JNZ     HOW_BIG_END

                        ;-----  CHECK 64K BLOCK FOR PARITY CHECK

                        HOW_BIG_2:
                                SUB     AX,AX                   ; WRITE ZEROS
                                MOV     CX,2000H*4              ; SET COUNT FOR 32K WORDS
                                REP     STOSW                   ; FILL 32K WORDS

                                PUSH    DS
                                PUSH    ES
                                PUSH    ES                      ; GET ES TO DS
                                POP     DS
                                MOV     CX,2000H*4              ; SET COUNT FOR 32K WORDS
                                SUB     SI,SI
                                REP     LODSW
                                SUB     DI,DI                   ; SET TO BEGINNING OF BLOCK
                                IN      AL,PORT_B               ; CHECK FOR PLANAR OR I/O PARITY CHECK
                                AND     AL,PARITY_ERR
                                MOV     WORD PTR ES:[DI],0      ; CLEAR POSSIBLE PARITY ERROR
                                POP     ES                      ; RESTORE SEGMENTS
                                POP     DS
                                JNZ     HOW_BIG_END             ; GO IF PLANAR OR I/O PARITY CHECK

                                RET

                        HOW_BIG_END:
                                PUSHF                           ; SAVE THE CURRENT FLAGS
                                MOV     AL,1CH                  ;       <><><><><><><><><><>
                                OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  1C <><>

                        ;-----  SET OR RESET 512 TO 640 INSTALLED FLAG

                                MOV     AX,X*(CMOS_INFO128+NMI) ; SET/RESET 640K STATUS FLAG
                                CALL    CMOS_READ               ; GET THE DIAGNOSTIC STATUS
                                OR      AL,M640K
                                CMP     BX,512                  ; CHECK MEMORY SIZE
                                JA      K640                    ; SET FLAG FOR 512 -> 640 INSTALLED
                                AND     AL,NOT M640K
                        K640:
                                XCHG    AL,AH                   ; SAVE THE STATUS
                                CALL    CMOS_WRITE              ; RESTORE THE STATUS

                                PUSH    BYTE PTR RSDA_PTR       ; RESTORE THE DATA SEGMENT
                                POP     DS
                                MOV     @MEMORY_SIZE,BX         ; SAVE MEMORY SIZE
                                POPF                            ; RESTORE THE FLAG REGISTER
                                RET

                        ;--------------------------------------------------------------------------
                        ; TEST.13A                                                               :
                        ;  PROTECTED MODE TEST AND MEMORY SIZE DETERMINE  ( ABOVE 1024K )       :
                        ;                                                                       :
                        ; DESCRIPTION:                                                          :
                        ;     THIS ROUTINE RUNS IN PROTECTED MODE IN ORDER TO ADDRESS ABOVE 1 MEG.:
                        ;     THE MEMORY SIZE IS DETERMINED AND SAVED IN CMOS.                  :
                        ;     DURING A POWER UP SEQUENCE THE MEMORY SIZE DETERMINE IS DONE WITH :
                        ;     PLANAR AND I/O PARITY CHECKS DISABLED.  DURING A SOFT RESET THE MEMORY:
                        ;     SIZE DETERMINE WILL CHECK FOR PARITY ERRORS.                      :
                        ;--------------------------------------------------------------------------

                        DONE:
                                PUSH    BYTE PTR GDT_PTR        ; POINT DS TO THE DESCRIPTOR TABLE
                                POP     DS

                        ;-----  START WITH SEGMENT ADDRESS 10-0000 (ONE MEG AND ABOVE)

                                MOV     BYTE PTR DS:[ES_TEMP.BASE_HI_BYTE],10H
                                MOV     DS:ES_TEMP.BASE_LO_WORD,0H

                                MOV     AL,1DH                  ;       <><><><><><><><><><>
                                OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  1D <><>

                                SUB     BX,BX                   ; START WITH COUNT 0

                        ;-----  START STORAGE SIZE/CLEAR

                        NOT_DONE1:
                                PUSH    BYTE PTR ES_TEMP        ; POINT ES TO DATA
                                POP     ES                      ; POINT TO SEGMENT TO TEST
                                CALL    HOW_BIG1                ; DO THE FIRST 64K
                                JZ      DONEA                   ; CHECK IF TOP

                                JMP     DONE1                   ; GO IF TOP
```

SECTION 5

```
                        DONEA:  ADD     BX,16*4                 ; BUMP MEMORY COUNT BY 64K
                        ;-----  DO NEXT 64K (XX0000) BLOCK
                                INC     BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE)
                        ;-----  CHECK FOR TOP OF MEMORY (FE0000)
                                CMP     BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE),0FEH ; LAST OF MEMORY?
                                JNZ     NOT_DONE1                               ; GO IF NOT
                                CALL    HOW_BIG_END1                            ; GO SET MEMORY SIZE
                                JMP     DONE1
                        ;-----  FILL/CHECK LOOP
                        HOW_BIG1:
                                SUB     DI,DI
                                MOV     AX,0AA55H               ; TEST PATTERN
                                MOV     CX,AX                   ; SAVE PATTERN
                                MOV     ES:[DI],AX              ; SEND PATTERN TO MEMORY
                                MOV     AL,0FH                  ; PUT SOMETHING IN AL
                                MOV     AX,ES:[DI]              ; GET PATTERN
                                MOV     ES:[DI],AX              ; INSURE NO PARITY I/O CHECK
                                XOR     AX,CX                   ; COMPARE PATTERNS
                                JNZ     HOW_BIG_END1            ; GO END IF NO COMPARE
                        ;-----  IS THIS A SOFT RESET
                                PUSH    DS
                                PUSH    BYTE PTR RSDA_PTR       ; POINT TO SYSTEM DATA AREA
                                POP     DS
                                CMP     @RESET_FLAG,1234H       ; SOFT RESET
                                POP     DS                      ; RESTORE DS
                                JNZ     HOW_BIG_2A              ; GO IF NOT SOFT RESET
                        ;-----  CHECK PARITY WITH PARITY BITS OFF
                                MOV     WORD PTR ES:[DI],0101H  ; TURN OFF BOTH PARITY BITS
                                PUSH    BYTE PTR 0FFH           ; PLACE 0FFFFH IN STACK (BUS BITS ON)
                                POP     AX                      ; DELAY - CAUSING BUS BITS ON
                                MOV     AX,ES:[DI]              ; CHECK PARITY
                                IN      AL,PORT_B               ; CHECK FOR PLANAR OR I/O PARITY CHECK
                                AND     AL,PARITY_ERR
                                MOV     ES:[DI],AX              ; CLEAR POSSIBLE PARITY ERROR
                                JNZ     HOW_BIG_END1            ; GO IF PLANAR OR I/O PARITY CHECK
                        ;-----  CHECK ALL BITS
                                MOV     WORD PTR ES:[DI],0FFFFH ; TURN ON ALL BITS
                                PUSH    BYTE PTR 0              ; PLACE 00000H IN STACK (BUS BITS OFF)
                                POP     AX                      ; DELAY - CAUSING BUS BITS OFF
                                MOV     AX,ES:[DI]              ; CHECK FOR FFFFH
                                PUSH    AX                      ; SAVE RESULTS
                                IN      AL,PORT_B               ; CHECK FOR PLANAR OR I/O PARITY CHECK
                                AND     AL,PARITY_ERR
                                MOV     ES:[DI],AX              ; CLEAR POSSIBLE PARITY ERROR
                                POP     AX                      ; GET RESULTS
                                JNZ     HOW_BIG_END1            ; GO IF PLANAR OR I/O PARITY CHECK
                                CMP     AX,0FFFFH
                                JNZ     HOW_BIG_END1
                        ;-----  CLEAR 64K BLOCK OF MEMORY
                        HOW_BIG_2A:
                                SUB     AX,AX                   ; WRITE ZEROS
                                MOV     CX,2000H*4              ; SET COUNT FOR 32K WORDS
                                REP     STOSW                   ; FILL 32K WORDS
                        ;-----  CHECK 64K BLOCK FOR PARITY CHECK (VALID TEST DURING SOFT RESET ONLY)
                                PUSH    DS
                                PUSH    ES
                                PUSH    ES                      ; GET ES TO DS
                                POP     DS
                                MOV     CX,2000H*4              ; SET COUNT FOR 32K WORDS
                                SUB     SI,SI
                                REP     LODSW
                                SUB     DI,DI                   ; SET TO BEGINNING OF BLOCK
                                IN      AL,PORT_B               ; CHECK FOR PLANAR OR I/O PARITY CHECK
                                AND     AL,PARITY_ERR
                                MOV     WORD PTR ES:[DI],0      ; CLEAR POSSIBLE PARITY ERROR
                                POP     ES                      ; RESTORE SEGMENT
                                POP     DS
                                JNZ     HOW_BIG_END1            ; GO IF PLANAR OR I/O PARITY CHECK
                                RET
                        HOW_BIG_END1:
                                MOV     AL,1EH                  ;       <><><><><><><><>
                                OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  1E <><>
                        ;-----  SET EXPANSION MEMORY SIZE DETERMINED IN CMOS
                                MOV     AL,CMOS_U_M_S_LO+NMI    ; ADDRESS LOW BYTE
                                MOV     AH,BL                   ; GET LOW MEMORY SIZE
                                CALL    CMOS_WRITE              ; SET LOW BYTE
                                MOV     AL,CMOS_U_M_S_HI+NMI    ; ADDRESS HI BYTE
                                MOV     AH,BH                   ; GET THE HIGH MEMORY SIZE
                                CALL    CMOS_WRITE              ; PLACE IN CMOS
                                RET
                        ;-----  TEST ADDRESS LINES 19 - 23
                        DONE1:  MOV     AL,1FH                  ;       <><><><><><><><>
                                OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  1F <><>
                                MOV     BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE),00H
                                SUB     DI,DI                   ; SET LOCATION POINTER TO ZERO
                                MOV     DX,0FFFFH               ; WRITE FFFF AT ADDRESS 0
                                CALL    SDO
                                SUB     DX,DX                   ; WRITE 0
                                MOV     BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE),08H
                                CALL    SDO
```

```
2588 0947 C6 06 004C 10                 MOV     BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE),10H
2589 094C E8 0969 R                     CALL    SD0
2590 094F C6 06 004C 20                 MOV     BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE),20H
2591 0954 E8 0969 R                     CALL    SD0
2592 0957 C6 06 004C 40                 MOV     BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE),40H
2593 095C E8 0969 R                     CALL    SD0
2594 095F C6 06 004C 80                 MOV     BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE),80H
2595 0964 E8 0969 R                     CALL    SD0
2596
2597 0967 EB 18                         JMP     SHORT SD2               ; TEST PASSED CONTINUE
2598
2599 0969                       SD0:
2600 0969 6A 48                         PUSH    BYTE PTR ES_TEMP        ; POINT ES TO DATA
2601 096B 07                            POP     ES                      ; POINT TO SEGMENT TO TEST
2602 096C 26: 89 15                     MOV     ES:[DI],DX              ; WRITE THE PATTERN
2603
2604 096F C6 06 004C 00                 MOV     BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE),00H
2605
2606 0974 6A 48                         PUSH    BYTE PTR ES_TEMP        ; POINT ES TO DATA
2607 0976 07                            POP     ES                      ; POINT TO SEGMENT TO TEST
2608 0977 26: 83 3D FF                  CMP     WORD PTR ES:[DI],0FFFFH ; DID LOCATION 0 CHANGE?
2609 097B 74 03                         JZ      SD1                     ; CONTINUE IF NOT
2610 097D E9 078C R                     JMP     SHUT_8                  ; GO HALT IF YES
2611 0980                       SD1:
2612 0980 C3                            RET
2613
2614                            ;-----  CAUSE A SHUTDOWN
2615
2616 0981 B0 20                 SD2:    MOV     AL,20H                  ;       <><><><><><><><><><><>
2617 0983 E6 80                         OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  20 <><>
2618 0985 E4 61                         IN      AL,PORT_B
2619 0987 0C 0C                         OR      AL,RAM_PAR_OFF          ; TOGGLE PARITY CHECK ENABLES
2620 0989 E6 61                         OUT     PORT_B,AL
2621 098B 24 F3                         AND     AL,RAM_PAR_ON
2622 098D E6 61                         OUT     PORT_B,AL
2623 098F E9 0000 E                     JMP     PROC_SHUTDOWN           ; CAUSE A SHUTDOWN (RETURN VIA JUMP)
2624
2625                            ;--------------------------------------
2626                            ; RETURN ! FROM SHUTDOWN               :
2627                            ;--------------------------------------
2628
2629 0992 B0 21                 SHUT1:  MOV     AL,21H                  ;       <><><><><><><><><><><>
2630 0994 E6 80                         OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  21 <><>
2631 0996 BC ---- R                     MOV     SP,ABS0                 ; SET REAL MODE STACK
2632 0999 8E D4                         MOV     SS,SP
2633 099B BC 0400 R                     MOV     SP,OFFSET @TOS
2634
2635                            ;-----  SET DIVIDE 0 VECTOR OFFSET
2636
2637 099E 2B FF                         SUB     DI,DI                   ; POINT TO FIRST INTERRUPT LOCATION
2638 09A0 8E C7                         MOV     ES,DI                   ; SET ES TO ABS0 SEGMENT
2639 09A2 B8 0000 E                     MOV     AX,OFFSET D11           ; GET ADDRESS OF INTERRUPT OFFSET
2640 09A5 AB                            STOSW                           ; PLACE OFFSET IF NULL HANDLER IN VECTOR
2641
2642 09A6 E8 0000 E                     CALL    DDS                     ; SET UP THE REAL DATA AREA
2643
2644                            ;-----  GET THE CONFIGURATION FROM CMOS
2645
2646 09A9 B8 8E8E                       MOV     AX,X*(CMOS_DIAG+NMI)    ; CHECK CMOS GOOD
2647 09AC E8 0000 E                     CALL    CMOS_READ               ; GET THE STATUS
2648 09AF A8 C0                         TEST    AL,BAD_BAT+BAD_CKSUM    ; VALID CMOS ?
2649 09B1 74 03                         JZ      M_OK                    ; GO IF YES
2650 09B3 E9 0A3C R                     JMP     BAD_MOS                 ; GO IF NOT
2651 09B6                       M_OK:
2652 09B6 24 DF                         AND     AL,0DFH                 ; CLEAR THE MINIMUM CONFIG BIT
2653 09B8 86 C4                         XCHG    AL,AH                   ; SAVE THE STATUS BYTE
2654 09BA E8 0000 E                     CALL    CMOS_WRITE              ; BACK INTO CMOS
2655
2656                            ;-----  CHECK FOR CMOS RUN IN MODE
2657
2658 09BD 81 3E 0072 R 1234             CMP     @RESET_FLAG,1234H       ; CHECK FOR SOFT RESET
2659 09C3 74 10                         JE      M_OK_64                 ; BYPASS IF SOFT RESET
2660
2661 09C5 B0 96                         MOV     AL,CMOS_B_M_S_HI+NMI    ; GET THE BASE MEMORY SIZE HIGH BYTE
2662 09C7 E8 0000 E                     CALL    CMOS_READ
2663 09CA 24 C0                         AND     AL,0C0H                 ; MASK FOR MANUFACTURING TEST BITS
2664 09CC 3C C0                         CMP     AL,0C0H                 ; CHECK FOR MANUFACTURING TEST MODE SET
2665 09CE 75 05                         JNE     M_OK_64                 ; SKIP IF NOT MANUFACTURING LINE TEST
2666
2667 09D0 C6 06 0072 R 64               MOV     BYTE PTR @RESET_FLAG,64H; ELSE SET THE MFG TEST FLAG
2668
2669                            ;-----  INSURE CONFIGURATION HAS CORRECT VIDEO TYPE
2670
2671 09D5                       M_OK_64:
2672 09D5 B0 94                         MOV     AL,CMOS_EQUIP+NMI       ; GET THE EQUIPMENT BYTE
2673 09D7 E8 0000 E                     CALL    CMOS_READ
2674 09DA 8A E0                         MOV     AH,AL                   ; SAVE VIDEO TYPE
2675 09DC A8 30                         TEST    AL,030H                 ; ANY VIDEO?
2676 09DE 75 31                         JNZ     MOS_OK_1                ; CONTINUE
2677 09E0 E8 09EE R                     CALL    CHK_VIDEO               ; INSURE VIDEO ROM PRESENT
2678 09E3 74 4C                         JZ      MOS_OK                  ; CONTINUE
2679
2680 09E5 F6 06 0012 R 20               TEST    @MFG_TST,MFG_LOOP       ; EXCEPT IF MFG JUMPER IS INSTALLED
2681 09EA 74 6F                         JZ      NORMAL_CONFIG           ; GO IF INSTALLED
2682
2683 09EC EB 4E                         JMP     SHORT BAD_MOS           ; GO DEFAULT
2684
2685                            ;-----  ROUTINE CHECK FOR VIDEO FEATURE ROM PRESENT
2686
2687 09EE                       CHK_VIDEO:
2688 09EE B9 C000                       MOV     CX,0C000H               ; START OF FEATURE I/O ROM
2689 09F1                       CHK_VIDEO1:
2690 09F1 50                            PUSH    AX                      ; SAVE THE CONFIGURATION
2691 09F2 1E                            PUSH    DS                      ; SAVE THE DATA SEGMENT
2692 09F3 57                            PUSH    DI                      ; SAVE COMPARE REGISTER
2693 09F4 8E D9                         MOV     DS,CX                   ; GET ROM SEGMENT
2694 09F6 BF AA55                       MOV     DI,0AA55H               ; GET THE PRESENCE SIGNATURE
2695 09F9 2B DB                         SUB     BX,BX                   ; CLEAR INDEX POINTER
2696 09FB 8B 07                         MOV     AX,[BX]                 ; GET THE FIRST 2 LOCATIONS
2697 09FD 3B C7                         CMP     AX,DI                   ; IS THE VIDEO FEATURE ROM PRESENT?
2698 09FF 5F                            POP     DI                      ; RESTORE WORK REGISTER
2699 0A00 1F                            POP     DS                      ; RESTORE DATA SEGMENT
2700 0A01 58                            POP     AX                      ; GET THE CONFIGURATION
2701 0A02 74 0C                         JZ      CHK_VIDEO2              ; GO IF VIDEO ROM INSTALLED
```

```
                                
0A04 81 C1 0080                 ADD     CX,080H                 ; POINT TO NEXT 2K BLOCK
0A08 81 F9 C800                 CMP     CX,0C800H               ; TOP OF VIDEO ROM AREA YET?
0A0C 7C E3                      JL      CHK_VIDEO1              ; TRY AGAIN
0A0E 23 C9                      AND     CX,CX                   ; SET NON ZERO FLAG
0A10                            CHK_VIDEO2:
0A10 C3                         RET                             ; RETURN TO CALLER
                                
                                ;-----  CMOS VIDEO BITS NON ZERO (CHECK FOR PRIMARY DISPLAY AND NO VIDEO ROM)
                                
0A11                            MOS_OK_1:
0A11 E8 09EE R                  CALL    CHK_VIDEO               ; IS THE VIDEO ROM INSTALLED?
0A14 74 26                      JZ      BAD_MOS                 ; WRONG CONFIGURATION IN CONFIG BYTE
                                
0A16 8A C4                      MOV     AL,AH                   ; RESTORE CONFIGURATION
0A18 F6 06 0012 R 40            TEST    @MFG_TST,DSP_JMP        ; CHECK FOR DISPLAY JUMPER
0A1D 74 0A                      JZ      MOS_OK_2                ; GO IF COLOR CARD IS PRIMARY DISPLAY
                                
                                ;-----  MONOCHROME CARD IS PRIMARY DISPLAY     (NO JUMPER INSTALLED)
                                
0A1F 24 30                      AND     AL,30H                  ; INSURE MONOCHROME IS PRIMARY
0A21 3C 30                      CMP     AL,30H                  ; CONFIGURATION OK?
0A23 75 17                      JNZ     BAD_MOS                 ; GO IF NOT
0A25 8A C4                      MOV     AL,AH                   ; RESTORE CONFIGURATION
0A27 EB 08                      JMP     SHORT MOS_OK            ; USE THE CONFIGURATION BYTE FOR DISPLAY
                                
                                ;-----  COLOR CARD
                                
0A29                            MOS_OK_2:
0A29 24 30                      AND     AL,30H                  ; STRIP UNWANTED BITS
0A2B 3C 30                      CMP     AL,30H                  ; MUST NOT BE MONO WITH JUMPER INSTALLED
0A2D 8A C4                      MOV     AL,AH                   ; RESTORE CONFIGURATION
0A2F 74 0B                      JZ      BAD_MOS                 ; GO IF YES
                                
                                ;-----  CONFIGURATION MUST HAVE AT LEAST ONE DISKETTE
                                
0A31 A8 01                      MOS_OK: TEST    AL,01H          ; MUST HAVE AT LEAST ONE DISKETTE
0A33 75 26                      JNZ     NORMAL_CONFIG           ; GO SET CONFIGURATION IF OK
0A35 F6 06 0012 R 20            TEST    @MFG_TST,MFG_LOOP       ; EXCEPT IF MFG JUMPER IS INSTALLED
0A3A 74 1F                      JZ      NORMAL_CONFIG           ; GO IF INSTALLED
                                
                                ;-----  MINIMUM CONFIGURATION WITH BAD CMOS OR NON VALID VIDEO
                                
0A3C                            BAD_MOS:
0A3C B8 008E                    MOV     AX,CMOS_DIAG+NMI        ; GET THE DIAGNOSTIC STATUS
0A3F E8 0000 E                  CALL    CMOS_READ
0A42 A8 C0                      TEST    AL,BAD_BAT+BAD_CKSUM    ; WAS BATTERY DEFECTIVE OR BAD CHECKSUM
0A44 75 03                      JNZ     BAD_MOS1                ; GO IF YES
                                
0A46 E8 0000 E                  CALL    CONFIG_BAD              ; SET THE MINIMUM CONFIGURATION FLAG
0A49                            BAD_MOS1:
0A49 E8 09EE R                  CALL    CHK_VIDEO               ; CHECK FOR VIDEO ROM
0A4C B0 01                      MOV     AL,01H                  ; DISKETTE ONLY
0A4E 74 0B                      JZ      NORMAL_CONFIG           ; GO IF VIDEO ROM PRESENT
                                
0A50 F6 06 0012 R 40            TEST    @MFG_TST,DSP_JMP        ; CHECK FOR DISPLAY JUMPER
0A55 B0 11                      MOV     AL,11H                  ; DEFAULT TO 40X25 COLOR
0A57 74 02                      JZ      NORMAL_CONFIG           ; GO IF JUMPER IS INSTALLED
                                
0A59 B0 31                      MOV     AL,31H                  ; DISKETTE / B/W DISPLAY 80X25
                                
                                ;---------------------------------------
                                ;       CONFIGURATION AND MFG MODE      :
                                ;---------------------------------------
                                
0A5B                            NORMAL_CONFIG:
0A5B F6 06 0012 R 20            TEST    @MFG_TST,MFG_LOOP       ; IS THE MANUFACTURING JUMPER INSTALLED
0A60 75 02                      JNZ     NORM1                   ; GO IF NOT
0A62 24 3E                      AND     AL,03EH                 ; STRIP DISKETTE FOR MFG TEST
                                
0A64 2A E4                      NORM1:  SUB     AH,AH
0A66 A3 0010 R                  MOV     @EQUIP_FLAG,AX          ; SAVE SWITCH INFORMATION
0A69 81 3E 0072 R 1234          CMP     @RESET_FLAG,1234H       ; BYPASS IF SOFT RESET
0A6F 74 2C                      JZ      E6
                                
                                ;-----  GET THE FIRST SELF TEST RESULTS FROM KEYBOARD
                                
0A71 B0 60                      MOV     AL,WRITE_8042_LOC       ; ENABLE KEYBOARD
0A73 E8 039D R                  CALL    C8042                   ; ISSUE WRITE BYTE COMMAND
0A76 B0 4D                      MOV     AL,4DH                  ; ENABLE OUTPUT BUFFER FULL INTERRUPT,
                                                                ; SET SYSTEM FLAG, PC I COMPATIBILITY,
0A78 E6 60                      OUT     PORT_A,AL               ; INHIBIT OVERRIDE, ENABLE KEYBOARD
                                
0A7A 2B C9                      SUB     CX,CX                   ; WAIT FOR COMMAND ACCEPTED
0A7C E8 03A2 R                  CALL    C42_1
                                
0A7F B9 7FFF                    MOV     CX,07FFFH               ; SET LOOP COUNT FOR APPROXIMATELY 100MS
                                                                ; TO RESPOND
0A82 E4 64                      TST6:   IN      AL,STATUS_PORT  ; WAIT FOR OUTPUT BUFFER FULL
0A84 A8 01                      TEST    AL,OUT_BUF_FULL
0A86 E1 FA                      LOOPZ   TST6                    ; TRY AGAIN IF NOT
                                
0A88 9C                         PUSHF                           ; SAVE FLAGS
0A89 B0 AD                      MOV     AL,DIS_KBD              ; DISABLE KEYBOARD
0A8B E8 039D R                  CALL    C8042                   ; ISSUE THE COMMAND
0A8E 9D                         POPF                            ; RESTORE FLAGS
0A8F 74 0C                      JZ      E6                      ; CONTINUE WITHOUT RESULTS
                                
0A91 E4 60                      IN      AL,PORT_A               ; GET INPUT FROM KEYBOARD
0A93 A2 0072 R                  MOV     BYTE PTR @RESET_FLAG,AL ; TEMPORARY SAVE FOR AA RECEIVED
                                
                                ;-----  CHECK FOR MFG REQUEST
                                
0A96 3C 65                      CMP     AL,065H                 ; LOAD MANUFACTURING TEST REQUEST?
0A98 75 03                      JNE     E6                      ; CONTINUE IF NOT
0A9A E9 0C34 R                  JMP     MFG_BOOT                ; ELSE GO TO MANUFACTURING BOOTSTRAP
```

```
                        PAGE
                        ;--------------------------------------------------------
                        ; TEST.14                                               :
                        ;       INITIALIZE AND START CRT CONTROLLER (6845)      :
                        ;       TEST VIDEO READ/WRITE STORAGE.                  :
                        ; DESCRIPTION                                           :
                        ;       RESET THE VIDEO ENABLE SIGNAL.                  :
                        ;       SELECT ALPHANUMERIC MODE, 40 * 25, B & W.       :
                        ;       READ/WRITE DATA PATTERNS TO MEMORY. CHECK       :
                        ;       STORAGE ADDRESSABILITY.                         :
                        ; ERROR = 1 LONG AND 2 SHORT BEEPS                      :
                        ;--------------------------------------------------------

0A9D A1 0010 R          E6:     MOV     AX,@EQUIP_FLAG          ; GET SENSE INFORMATION
0AA0 50                         PUSH    AX                      ; SAVE IT
0AA1 B0 30                      MOV     AL,30H                  ; FORCE MONOCHROME TYPE
0AA3 A3 0010 R                  MOV     @EQUIP_FLAG,AX          ; INTO EQUIPMENT FLAG
0AA6 2B C0                      SUB     AX,AX                   ; MODE SET COMMAND FOR DEFAULT MODE
0AA8 CD 10                      INT     INT_VIDEO               ; SEND INITIALIZATION TO B/W CARD
0AAA B0 20                      MOV     AL,20H                  ; FORCE COLOR AT 80 BY 25
0AAC A3 0010 R                  MOV     @EQUIP_FLAG,AX          ; INTO EQUIPMENT FLAG TO CLEAR BUFFERS
0AAF B8 0003                    MOV     AX,0003H                ; AND INITIALIZATION COLOR CARD 80X25
0AB2 CD 10                      INT     INT_VIDEO               ; MODE SET 80 X 25
0AB4 B8 0001                    MOV     AX,0001H                ; SET COLOR 40 X 25 MODE
0AB7 CD 10                      INT     INT_VIDEO               ; SET DEFAULT COLOR MODE
0AB9 58                         POP     AX                      ; RECOVER REAL SWITCH INFORMATION
0ABA A3 0010 R                  MOV     @EQUIP_FLAG,AX          ; RESTORE IT
0ABD 24 30                      AND     AL,30H                  ; ISOLATE VIDEO SWITCHES
0ABF 75 11                      JNZ     E7                      ; VIDEO SWITCHES SET TO 0?
0AC1 1E                         PUSH    DS                      ; SAVE THE DATA SEGMENT
0AC2 50                         PUSH    AX
0AC3 2B C0                      SUB     AX,AX                   ; SET DATA SEGMENT TO 0
0AC5 8E D8                      MOV     DS,AX
0AC7 BF 0040 R                  MOV     DI,OFFSET @VIDEO_INT    ; SET INTERRUPT 10H TO DUMMY
0ACA C7 05 0000 E               MOV     WORD PTR [DI],OFFSET DUMMY_RETURN  ; RETURN IF NO VIDEO CARD
0ACE 58                         POP     AX                      ; RESTORE REGISTERS
0ACF 1F                         POP     DS
0AD0 EB 7F                      JMP     SHORT E18_1             ; BYPASS VIDEO TEST
0AD2                    E7:
0AD2 3C 30                      CMP     AL,30H                  ; B/W CARD ATTACHED?
0AD4 74 08                      JE      E8                      ; YES - SET MODE FOR B/W CARD
0AD6 FE C4                      INC     AH                      ; SET COLOR MODE FOR COLOR CARD
0AD8 3C 20                      CMP     AL,20H                  ; 80X25 MODE SELECTED?
0ADA 75 02                      JNE     E8                      ; NO - SET MODE FOR 40X25
0ADC B4 03                      MOV     AH,3                    ; SET MODE FOR 80X25
0ADE                    E8:
0ADE 86 E0                      XCHG    AH,AL
0AE0 50                         PUSH    AX                      ; SAVE VIDEO MODE ON STACK
0AE1 2A E4                      SUB     AH,AH                   ; INITIALIZE TO ALPHANUMERIC MD
0AE3 CD 10                      INT     INT_VIDEO               ; CALL VIDEO_IO
0AE5 58                         POP     AX                      ; RESTORE VIDEO SENSE SWITCHES IN AH
0AE6 50                         PUSH    AX                      ; SAVE VALUE
0AE7 BB B000                    MOV     BX,0B000H               ; STARTING VIDEO MEMORY ADDRESS B/W CARD
0AEA BA 03B8                    MOV     DX,3B8H                 ; MODE REGISTER FOR B/W
0AED B9 0800                    MOV     CX,2048                 ; MEMORY WORD COUNT FOR B/W CARD
0AF0 80 FC 30                   CMP     AH,30H                  ; B/W VIDEO CARD ATTACHED?
0AF3 74 07                      JE      E9                      ; YES - GO TEST VIDEO STORAGE
0AF5 B7 B8                      MOV     BH,0B8H                 ; STARTING MEMORY ADDRESS FOR COLOR CARD
0AF7 BA 03D8                    MOV     DX,3D8H                 ; MODE REGISTER FOR COLOR CARD
0AFA B5 20                      MOV     CH,20H                  ; MEMORY WORD COUNT FOR COLOR CARD
0AFC                    E9:
0AFC A0 0065 R                  MOV     AL,@CRT_MODE_SET        ; GET CURRENT MODE SET VALUE
0AFF 24 37                      AND     AL,037H                 ; SET VIDEO BIT OFF
0B01 EE                         OUT     DX,AL                   ; DISABLE VIDEO FOR COLOR CARD
0B02 8E C3                      MOV     ES,BX                   ; POINT ES TO VIDEO MEMORY
0B04 8E DB                      MOV     DS,BX                   ; POINT DS TO VIDEO MEMORY
0B06 D1 C9                      ROR     CX,1                    ; DIVIDE BY 2 FOR WORD COUNT
0B08 E8 0000 E                  CALL    STGTST_CNT              ; GO TEST VIDEO READ/WRITE STORAGE
0B0B 75 7B                      JNE     E17                     ; R/W MEMORY FAILURE - BEEP SPEAKER

                        ;--------------------------------------------------------
                        ; TEST.15                                               :
                        ;       SETUP VIDEO DATA ON SCREEN FOR VIDEO            :
                        ;       LINE TEST.                                      :
                        ; DESCRIPTION                                           :
                        ;       ENABLE VIDEO SIGNAL AND SET MODE.               :
                        ;       DISPLAY A HORIZONTAL BAR ON SCREEN.             :
                        ;--------------------------------------------------------

0B0D B0 22                      MOV     AL,22H                  ;       <><><><><><><><>
0B0F E6 80                      OUT     MFG_PORT,AL             ;       <> CHECKPOINT  22 <>
0B11 58                         POP     AX                      ; GET VIDEO SENSE SWITCHES (AH)
0B12 50                         PUSH    AX                      ; SAVE IT
0B13 B4 00                      MOV     AH,0                    ; ENABLE VIDEO AND SET MODE
0B15 CD 10                      INT     INT_VIDEO               ; VIDEO
0B17 B8 7020                    MOV     AX,7020H                ; WRITE BLANKS IN REVERSE VIDEO
0B1A 2B FF                      SUB     DI,DI                   ; SETUP STARTING LOCATION
0B1C B9 0028                    MOV     CX,40                   ; NUMBER OF BLANKS TO DISPLAY
0B1F F3/ AB                     REP     STOSW                   ; WRITE VIDEO STORAGE

                        ;--------------------------------------
                        ; TEST.16                              :
                        ;       CRT INTERFACE LINES TEST       :
                        ; DESCRIPTION                         :
                        ;       SENSE ON/OFF TRANSITION OF THE :
                        ;       VIDEO ENABLE AND HORIZONTAL    :
                        ;       SYNC LINES.                    :
                        ;--------------------------------------

0B21 58                         POP     AX                      ; GET VIDEO SENSE SWITCH INFORMATION
0B22 50                         PUSH    AX                      ; SAVE IT
0B23 80 FC 30                   CMP     AH,30H                  ; B/W CARD ATTACHED?
0B26 BA 03BA                    MOV     DX,03BAH                ; SETUP ADDRESS OF B/W STATUS PORT
0B29 74 03                      JE      E11                     ; YES - GO TEST LINES
0B2B BA 03DA                    MOV     DX,03DAH                ; COLOR CARD IS ATTACHED
0B2E                    E11:
0B2E B4 08                      MOV     AH,8
0B30                    E12:
0B30 2B C9                      SUB     CX,CX
0B32 EC                 E13:    IN      AL,DX                   ; READ CRT STATUS PORT
0B33 22 C4                      AND     AL,AH                   ; CHECK VIDEO/HORIZONTAL LINE
0B35 75 04                      JNZ     E14                     ; ITS ON - CHECK IF IT GOES OFF
0B37 E2 F9                      LOOP    E13                     ; LOOP UNTIL ON OR TIMEOUT
0B39 EB 4D                      JMP     SHORT E17               ; GO PRINT ERROR MESSAGE
```

SECTION 5

```
0B3B                    E14:
0B3B 2B C9                      SUB     CX,CX
0B3D EC                 E15:    IN      AL,DX                   ; READ CRT STATUS PORT
0B3E 22 C4                      AND     AL,AH                   ; CHECK VIDEO/HORIZONTAL LINE
0B40 74 04                      JZ      E16                     ; ITS ON - CHECK NEXT LINE
0B42 E2 F9                      LOOP    E15                     ; LOOP IF ON UNTIL IT GOES OFF
0B44 EB 42                      JMP     SHORT E17               ; GO ERROR BEEP
                        ;-----  CHECK HORIZONTAL LINE
0B46 B1 03              E16:    MOV     CL,3                    ; GET NEXT BIT TO CHECK
0B48 D2 EC                      SHR     AH,CL
0B4A 75 E4                      JNZ     E12                     ; CONTINUE
0B4C                    E18:
0B4C 58                         POP     AX                      ; GET VIDEO SENSE SWITCHES (AH)
0B4D B4 00                      MOV     AH,0                    ; SET MODE AND DISPLAY CURSOR
0B4F CD 10                      INT     INT_VIDEO               ; CALL VIDEO I/O PROCEDURE
                        ;-----  CHECK FOR THE ADVANCED VIDEO CARD
                                ASSUME DS:ABS0
0B51 BA 0000            E18_1:  MOV     DX,0                    ; SET DS TO VECTORS SEGMENT
0B54 8E DA                      MOV     DS,DX                   ; RESET SEGMENT VALUE
0B56 C7 06 007E R 0000          MOV     WORD PTR @EXT_PTR+2,0   ; FOR GRAPHIC CHARS !28 - 255
0B5C BA C000                    MOV     DX,0C000H               ; SET THE LOW SEGMENT VALUE
0B5F                    E18A:
0B5F B0 23                      MOV     AL,23H                  ;       <><><><><><><><><><>
0B61 E6 80                      OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  23 <><>
0B63 8E DA                      MOV     DS,DX
0B65 57                         PUSH    DI                      ; SAVE WORK REGISTER
0B66 BF AA55                    MOV     DI,0AA55H               ; PRESENCE SIGNATURE
0B69 2B DB                      SUB     BX,BX                   ; CLEAR POINTER
0B6B 8B 07                      MOV     AX,[BX]                 ; GET FIRST 2 LOCATIONS
0B6D 3B C7                      CMP     AX,DI                   ; PRESENT?
0B6F 5F                         POP     DI                      ; RECOVER REGISTER
0B70 75 05                      JNZ     E18B                    ; NO? GO LOOK FOR OTHER MODULES
0B72 E8 0000 E                  CALL    ROM_CHECK               ; GO SCAN MODULE
0B75 EB 04                      JMP     SHORT E18C
0B77                    E18B:
0B77 81 C2 0080                 ADD     DX,0080H                ; POINT TO NEXT 2K BLOCK
0B7B                    E18C:
0B7B 81 FA C800                 CMP     DX,0C800H               ; TOP OF VIDEO ROM AREA YET?
0B7F 7C DE                      JL      E18A                    ; GO SCAN FOR ANOTHER MODULE
0B81 B0 24                      MOV     AL,24H                  ;       <><><><><><><><><><>
0B83 E6 80                      OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  24 <><>
0B85 E9 0000 E                  JMP     POST2                   ; GO TO NEXT TEST
                                ASSUME  DS:DATA
                        ;-----  CRT ERROR SET MFG CHECKPOINT AND ERROR BEEP
0B88 E8 0000 E          E17:    CALL    DDS                     ; POINT TO DATA
                        ;-----  CHECKPOINT 0C = MONOCHROME FAILED
0B8B C6 06 0015 R 0C            MOV     @MFG_ERR_FLAG,0CH       ; <><> CRT ERROR CHECKPOINT  0C <><>
0B90 80 3E 0072 R 64            CMP     BYTE PTR @RESET_FLAG,064H ; IS THIS A MFG REQUEST?
0B95 74 0D                      JZ      E19                     ; BY PASS ERROR BEEP IF YES
0B97 F6 06 0012 R 20            TEST    @MFG_TST,MFG_LOOP       ; IS THE MFG LOOP JUMPER INSTALLED?
0B9C 74 06                      JZ      E19                     ; BY PASS ERROR BEEP IF YES
0B9E BA 0102                    MOV     DX,102H
0BA1 E8 0000 E                  CALL    ERR_BEEP                ; GO BEEP SPEAKER
0BA4                    E19:
0BA4 1E                         PUSH    DS
0BA5 A1 0010 R                  MOV     AX,@EQUIP_FLAG          ; GET THE CURRENT VIDEO
0BA8 24 30                      AND     AL,30H                  ; STRIP OTHER BITS
0BAA 3C 30                      CMP     AL,30H                  ; IS IT MONOCHROME ?
0BAC 74 30                      JZ      TRY_COLOR               ; GO IF YES
                        ;-----  COLOR FAILED TRY MONOCHROME - CHECKPOINT 0D = COLOR FAILED
0BAE C6 06 0015 R 0D            MOV     @MFG_ERR_FLAG,0DH       ; <><> CRT ERROR CHECKPOINT  0D <><>
0BB3 BA 03B8                    MOV     DX,3B8H                 ; DISABLE B/W
0BB6 B0 01                      MOV     AL,1
0BB8 EE                         OUT     DX,AL                   ; OUTPUT THE DISABLE
0BB9 BB B000                    MOV     BX,0B000H               ; CHECK FOR MONOCHROME VIDEO MEMORY
0BBC 8E DB                      MOV     DS,BX
0BBE B8 AA55                    MOV     AX,0AA55H               ; WRITE AN AA55
0BC1 2B DB                      SUB     BX,BX                   ; TO THE FIRST LOCATION
0BC3 89 07                      MOV     [BX],AX
0BC5 EB 00                      JMP     $+2                     ; ALLOW BUS TO SETTLE
0BC7 8B 07                      MOV     AX,[BX]                 ; READ THE FIRST LOCATION
0BC9 3D AA55                    CMP     AX,0AA55H               ; IS THE MONOCHROME VIDEO CARD THERE?
0BCC 1F                         POP     DS                      ; RESTORE THE DATA SEGMENT
0BCD 75 55                      JNZ     E17_3                   ; GO IF NOT
0BCF 81 0E 0010 R 0030          OR      @EQUIP_FLAG,30H         ; TURN ON MONOCHROME BITS IN EQUIP FLAG
0BD5 A1 0010 R                  MOV     AX,@EQUIP_FLAG          ; ENABLE VIDEO
0BD8 2A E4                      SUB     AH,AH
0BDA CD 10                      INT     INT_VIDEO
0BDC EB 34                      JMP     SHORT E17_1             ; CONTINUE
                        ;-----  MONOCHROME FAILED TRY COLOR
0BDE                    TRY_COLOR:
0BDE B0 01                      MOV     AL,01H                  ; SET MODE COLOR 40X25
0BE0 2A E4                      SUB     AH,AH
0BE2 CD 10                      INT     INT_VIDEO
0BE4 BA 03D8                    MOV     DX,3D8H                 ; DISABLE COLOR
0BE7 B0 00                      MOV     AL,0
0BE9 EE                         OUT     DX,AL                   ; OUTPUT THE DISABLE
0BEA BB B800                    MOV     BX,0B800H               ; CHECK FOR COLOR VIDEO MEMORY
0BED 8E DB                      MOV     DS,BX
0BEF B8 AA55                    MOV     AX,0AA55H               ; WRITE AN AA55
0BF2 2B DB                      SUB     BX,BX                   ; TO THE FIRST LOCATION
0BF4 89 07                      MOV     [BX],AX
0BF6 EB 00                      JMP     $+2                     ; ALLOW BUS TO SETTLE
0BF8 8B 07                      MOV     AX,[BX]                 ; READ THE FIRST LOCATION
0BFA 3D AA55                    CMP     AX,0AA55H               ; IS THE COLOR VIDEO CARD THERE?
0BFD 1F                         POP     DS                      ; RESTORE THE DATA SEGMENT
0BFE 75 24                      JNZ     E17_3                   ; GO IF NOT
```

```
0036 0C00 81 26 0010 R FFCF          AND     @EQUIP_FLAG,0FFCFH      ; TURN OFF VIDEO BITS
0037 0C06 81 0E 0010 R 0010          OR      @EQUIP_FLAG,10H         ; SET COLOR 40X24
0038 0C0C B0 01                      MOV     AL,01H
0039 0C0E 2A E4                      SUB     AH,AH
0040 0C10 CD 10                      INT     INT_VIDEO
0041 0C12                    E17_1:
0042 0C12 58                         POP     AX                      ; SET NEW VIDEO TYPE ON STACK
0043 0C13 A1 0010 R                  MOV     AX,@EQUIP_FLAG
0044 0C16 24 30                      AND     AL,30H
0045 0C18 3C 30                      CMP     AL,30H                  ; IS IT THE B/W?
0046 0C1A 2A C0                      SUB     AL,AL
0047 0C1C 74 02                      JZ      E17_2                   ; GO IF YES
0048 0C1E FE C0                      INC     AL                      ; INITIALIZE FOR 40X25
0049 0C20                    E17_2:
0050 0C20 50                         PUSH    AX
0051 0C21                    E17_4:
0052 0C21 E9 0B4C R                  JMP     E18
0053
0054                         ;-----  BOTH VIDEO CARDS FAILED SET DUMMY RETURN IF RETRACE FAILURE
0055
0056 0C24                    E17_3:
0057 0C24 1E                         PUSH    DS
0058 0C25 2B C0                      SUB     AX,AX                   ; SET DS SEGMENT TO 0
0059 0C27 8E D8                      MOV     DS,AX
0060 0C29 BF 0040 R                  MOV     DI,OFFSET @VIDEO_INT    ; SET INTERRUPT 10H TO DUMMY
0061 0C2C C7 05 0000 E               MOV     WORD PTR [DI],OFFSET DUMMY_RETURN  ; RETURN IF NO VIDEO CARD
0062 0C30 1F                         POP     DS
0063 0C31 E9 0B51 R                  JMP     E18_1                   ; BYPASS REST OF VIDEO TEST
```

```
                        PAGE
                        ;--------------------------------------------------------------------------------
                        ; MANUFACTURING BOOT TEST CODE ROUTINE                                          :
                        ;       LOAD A BLOCK OF TEST CODE THROUGH THE KEYBOARD PORT FOR MANUFACTURING   :
                        ;       TESTS.                                                                 :
                        ;       THIS ROUTINE WILL LOAD A TEST (MAX LENGTH=FAFFH) THROUGH THE KEYBOARD  :
                        ;       PORT. CODE WILL BE LOADED AT LOCATION  0000:0500.  AFTER LOADING,      :
                        ;       CONTROL WILL BE TRANSFERRED TO LOCATION  0000:0500.   THE STACK WILL   :
                        ;       BE LOCATED AT 0000:0400.  THIS ROUTINE ASSUMES THAT THE FIRST 2 BYTES  :
                        ;       TRANSFERRED CONTAIN THE COUNT OF BYTES TO BE LOADED                    :
                        ;         (BYTE 1=COUNT LOW, BYTE 2=COUNT HI.)                                 :
                        ;--------------------------------------------------------------------------------

                        ;-----  DEGATE ADDRESS LINE 20

0C34                    MFG_BOOT:
0C34 B4 DD                      MOV     AH,DISABLE_BIT20        ; DEGATE COMMAND FOR ADDRESS LINE 20
0C36 E8 0000 E                  CALL    GATE_A20                ; ISSUE TO KEYBOARD ADAPTER AND CLI

                        ;-----  SETUP HARDWARE INTERRUPT VECTOR TABLE LEVEL 0-7 AND SOFTWARE INTERRUPTS

0C39 68 ---- R                  PUSH    ABS0                    ; SET ES SEGMENT REGISTER TO ABS0
0C3C 07                         POP     ES
0C3D B9 0018                    MOV     CX,24                   ; GET VECTOR COUNT
0C40 8C C8                      MOV     AX,CS                   ; GET THE CURRENT CODE SEGMENT VALUE
0C42 8E D8                      MOV     DS,AX                   ; SETUP DS SEGMENT REGISTER TO
0C44 BE 0000 E                  MOV     SI,OFFSET VECTOR_TABLE  ; POINT TO THE ROUTINE ADDRESS TABLE
0C47 BF 0020 R                  MOV     DI,OFFSET @INT_PTR      ; SET DESTINATION TO FIRST USED VECTOR
0C4A                    MFG_B1:
0C4A A5                         MOVSW                           ; MOVE ONE ROUTINE OFFSET ADDRESS
0C4B AB                         STOSW                           ; INSERT CODE SEGMENT VALUE
0C4C E2 FC                      LOOP    MFG_B1                  ; MOVE THE NUMBER OF ENTRIES REQUIRED

                        ;-----  SETUP HARDWARE INTERRUPT VECTORS LEVEL 8-15 (VECTORS START AT INT 70 H)

0C4E B9 0008                    MOV     CX,08                   ; GET VECTOR COUNT
0C51 BE 0000 E                  MOV     SI,OFFSET SLAVE_VECTOR_TABLE
0C54 BF 01C0 R                  MOV     DI,OFFSET @SLAVE_INT_PTR
0C57                    MFG_B2:
0C57 A5                         MOVSW                           ; MOVE ONE ROUTINE OFFSET ADDRESS
0C58 AB                         STOSW                           ; INSERT CODE SEGMENT VALUE
0C59 E2 FC                      LOOP    MFG_B2

                        ;-----  SET UP OTHER INTERRUPTS AS NECESSARY

                                ASSUME  DS:ABS0,ES:ABS0
0C5B 06                         PUSH    ES                                      ; ES= ABS0
0C5C 1F                         POP     DS                                      ; SET DS TO ABS0
0C5D C7 06 0008 R 0000 E        MOV     WORD PTR @NMI_PTR,OFFSET NMI_INT        ; NMI INTERRUPT
0C63 C7 06 0014 R 0000 E        MOV     WORD PTR @INT5_PTR,OFFSET PRINT_SCREEN  ; PRINT SCREEN
0C69 C7 06 0062 R F600          MOV     WORD PTR @BASIC_PTR+2,0F600H            ; CASSETTE BASIC SEGMENT

                        ;-----  ENABLE KEYBOARD PORT

0C6F B0 60                      MOV     AL,60H                  ; WRITE 8042 MEMORY LOCATION 0
0C71 E8 039D R                  CALL    C8042                   ; ISSUE THE COMMAND
0C74 B0 09                      MOV     AL,00001001B            ; SET INHIBIT OVERRIDE/ENABLE OBF
0C76 E6 60                      OUT     PORT_A,AL               ;   INTERRUPT AND NOT PC COMPATIBLE

0C78 E8 0C9A R                  CALL    MFG_B4                  ; GET COUNT LOW
0C7B 8A F8                      MOV     BH,AL                   ; SAVE IT
0C7D E8 0C9A R                  CALL    MFG_B4                  ; GET COUNT HI
0C80 8A E8                      MOV     CH,AL
0C82 8A CF                      MOV     CL,BH                   ; CX NOW HAS COUNT
0C84 FC                         CLD                             ; SET DIRECTION FLAG TO INCREMENT
0C85 BF 0500 R                  MOV     DI,OFFSET @MFG_TEST_RTN ; SET TARGET OFFSET (DS=0000)
0C88                    MFG_B3:
0C88 E4 64                      IN      AL,STATUS_PORT          ; GET 8042 STATUS PORT
0C8A A8 01                      TEST    AL,OUT_BUF_FULL         ; KEYBOARD REQUEST PENDING?
0C8C 74 FA                      JZ      MFG_B3                  ; LOOP TILL DATA PRESENT
0C8E E4 60                      IN      AL,PORT_A               ; GET DATA
0C90 AA                         STOSB                           ; STORE IT
0C91 E6 80                      OUT     MFG_PORT,AL             ; DISPLAY CHARACTER AT MFG PORT
0C93 E2 F3                      LOOP    MFG_B3                  ; LOOP TILL ALL BYTES READ

0C95 EA 0500 ---- R             JMP     @MFG_TEST_RTN           ; FAR JUMP TO CODE THAT WAS JUST LOADED

0C9A                    MFG_B4:
0C9A E4 64                      IN      AL,STATUS_PORT          ; CHECK FOR OUTPUT BUFFER FULL
0C9C A8 01                      TEST    AL,OUT_BUF_FULL         ;   HANG HERE IF NO DATA AVAILABLE
0C9E E1 FA                      LOOPZ   MFG_B4

0CA0 E4 60                      IN      AL,PORT_A               ; GET THE COUNT
0CA2 C3                         RET

0CA3                    POST1   ENDP
0CA3                    CODE    ENDS
                                END
```

```
                        PAGE 118,121
                        TITLE TEST2 ---- 01/24/85  POST TESTS AND INITIALIZATION ROUTINES
                        .286C
                        .287
                        .LIST
0000                    CODE    SEGMENT BYTE PUBLIC
                                PUBLIC  C21
                                PUBLIC  POST2
                                PUBLIC  SHUT2
                                PUBLIC  SHUT3
                                PUBLIC  SHUT4
                                PUBLIC  SHUT6
                                PUBLIC  SHUT7

                                EXTRN   BLINK_INT:NEAR
                                EXTRN   C8042:NEAR
                                EXTRN   CMOS_READ:NEAR
                                EXTRN   CMOS_WRITE:NEAR
                                EXTRN   CONFTG_BAD:NEAR
                                EXTRN   D1:NEAR
                                EXTRN   D2:NEAR
                                EXTRN   DDS:NEAR
                                EXTRN   DISK_SETUP:NEAR
                                EXTRN   DSKETTE_SETUP:NEAR
                                EXTRN   ERR_BEEP:NEAR
                                EXTRN   E_MSG:NEAR
                                EXTRN   F3D:NEAR
                                EXTRN   F3D1:NEAR
                                EXTRN   GATE_A20:NEAR
                                EXTRN   HD_INT:NEAR
                                EXTRN   KBD_RESET:NEAR
                                EXTRN   NMI_INT:NEAR
                                EXTRN   OBF_42:NEAR
                                EXTRN   POST3:NEAR
                                EXTRN   PRINT_SCREEN:NEAR
                                EXTRN   PROC_SHUTDOWN:NEAR
                                EXTRN   PROT_PRT_HEX:NEAR
                                EXTRN   PRT_HEX:NEAR
                                EXTRN   P_MSG:NEAR
                                EXTRN   ROM_CHECK:NEAR
                                EXTRN   ROM_CHECKSUM:NEAR
                                EXTRN   SEEK:NEAR
                                EXTRN   SET_TOD:NEAR
                                EXTRN   SLAVE_VECTOR_TABLE:NEAR
                                EXTRN   SND_DATA:NEAR
                                EXTRN   START_1:NEAR
                                EXTRN   STGTST_CNT:NEAR
                                EXTRN   SYSINIT1:NEAR
                                EXTRN   VECTOR_TABLE:NEAR
                                EXTRN   WAITF:NEAR
                                EXTRN   XPC_BYTE:NEAR

                                EXTRN   E101:NEAR               ; 101 ERROR CODE - INTERRUPT FAILURE
                                EXTRN   E102:NEAR               ; 102 ERROR CODE - TIMER FAILURE
                                EXTRN   E103:NEAR               ; 103 ERROR CODE - TIMER INTERRUPT
                                EXTRN   E104:NEAR               ; 104 ERROR CODE - PROTECTED MODE ERROR
                                EXTRN   E105:NEAR               ; 105 ERROR CODE - 8042 COMMAND FAILURE
                                EXTRN   E106:NEAR               ; 106 ERROR CODE - CONVERTING LOGIC
                                EXTRN   E107:NEAR               ; 107 ERROR CODE - NMI ERROR
                                EXTRN   E108:NEAR               ; 108 ERROR CODE - TIMER BUS ERROR
                                EXTRN   E109:NEAR               ; 109 ERROR CODE - MEMORY SELECT ERROR
                                EXTRN   E161:NEAR               ; 161 ERROR CODE - BAD BATTERY
                                EXTRN   E162:NEAR               ; 162 ERROR CODE - CMOS CHECKSUM/CONFIG
                                EXTRN   E163:NEAR               ; 163 ERROR CODE - BAD REAL TIME CLOCK
                                EXTRN   E164:NEAR               ; 164 ERROR CODE - MEMORY SIZE WRONG
                                EXTRN   E201:NEAR               ; 201 ERROR CODE - MEMORY DATA ERROR
                                EXTRN   E202:NEAR               ; 202 ERROR CODE - MEMORY ADDRESS ERROR
                                EXTRN   E203:NEAR               ; 203 ERROR CODE - SEGMENT ADDRESS ERROR
                                EXTRN   E301:NEAR               ; 301 ERROR CODE - KEYBOARD ERROR
                                EXTRN   E302:NEAR               ; 302 ERROR CODE - LOCK IS ON
                                EXTRN   E303:NEAR               ; 303 ERROR CODE - KEYBOARD/PLANAR ERROR
                                EXTRN   E304:NEAR               ; 304 ERROR CODE - KEYBOARD/PLANAR ERROR
                                EXTRN   E401:NEAR               ; 401 ERROR CODE - MONOCHROME ADAPTER
                                EXTRN   E501:NEAR               ; 501 ERROR CODE - COLOR ADAPTER
                                EXTRN   E601:NEAR               ; 601 ERROR CODE - DISKETTE ADAPTER

                        ;--------------------------------------------------------
                        ; TEST.17                                               :
                        ;       8259 INTERRUPT CONTROLLER TEST                  :
                        ; DESCRIPTION                                           :
                        ;       READ/WRITE THE INTERRUPT MASK REGISTER (IMR)    :
                        ;       WITH ALL ONES AND ZEROES. ENABLE SYSTEM         :
                        ;       INTERRUPTS.  MASK DEVICE INTERRUPTS OFF. CHECK  :
                        ;       FOR HOT INTERRUPTS (UNEXPECTED).                :
                        ;--------------------------------------------------------

                                ASSUME  CS:CODE,DS:DATA

0000                    POST2   PROC    NEAR

0000 B0 0A              C21:    MOV     AL,10                   ; LINE FEED ON DISPLAY
0002 E8 0000 E                  CALL    PRT_HEX
0005 E8 0000 E                  CALL    DDS                     ;SET DATA SEGMENT

                        ;-----  CLEAR ERROR FLAG REGISTER (BP) <=> 0 FLAGS ERROR

0008 2B ED                      SUB     BP,BP                   ; CLEAR (BP) REGISTER AS ERROR FLAG REG

                        ;-----  TEST THE INTERRUPT MASK REGISTER REGISTERS

000A FA                 C21A:   CLI                             ; TURN OFF INTERRUPTS
000B B0 00                      MOV     AL,0                    ; SET INTERRUPT MASK REGISTER TO ZERO
000D E6 21                      OUT     INTA01,AL
000F E6 A1                      OUT     INTB01,AL               ; SEND TO 2ND INTERRUPT CONTROLLER ALSO
0011 EB 00                      JMP     $+2
0013 E4 21                      IN      AL,INTA01               ; READ INTERRUPT MASK REGISTER
0015 8A E0                      MOV     AH,AL                   ; SAVE RESULTS
0017 E4 A1                      IN      AL,INTB01               ; READ 2ND INTERRUPT MASK REGISTER

0019 0A E0                      OR      AH,AL                   ; BOTH IMR = 0?
001B 75 2C                      JNZ     D6                      ; GO TO ERR ROUTINE IF NOT 0

001D B0 25                      MOV     AL,25H                  ;       <><><><><><><><><><>
001F E6 80                      OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  25 <><>
```

```
                        MOV     AL,0FFH                 ; DISABLE DEVICE INTERRUPTS
0021 B0 FF              OUT     INTA01,AL               ; WRITE TO INTERRUPT MASK REGISTER
0023 E6 21              OUT     INTB01,AL               ; WRITE TO 2ND INTERRUPT MASK REGISTER
0025 E6 A1              JMP     $+2                     ; I/O DELAY
0027 EB 00              IN      AL,INTA01               ; READ INTERRUPT MASK REGISTER
0029 E4 21              MOV     AH,AL                   ; SAVE RESULTS
002B 8A E0              IN      AL,INTB01               ; READ 2ND INTERRUPT MASK REGISTER
002D E4 A1
                        ADD     AX,1                    ; ALL IMR BITS ON?
002F 05 0001            JNZ     D6                      ; NO - GO TO ERR ROUTINE
0032 75 15
                ;-----  CHECK FOR HOT INTERRUPTS

                ;-----  INTERRUPTS ARE MASKED OFF.  CHECK THAT NO INTERRUPTS OCCUR.

                        MOV     @INTR_FLAG,AL           ; CLEAR INTERRUPT FLAG
0034 A2 006B R
                        MOV     AL,26H                  ;       <><><><><><><><>
0037 B0 26              OUT     MFG_PORT,AL             ;       <> CHECKPOINT  26 <>
0039 E6 80
                        STI                             ; ENABLE EXTERNAL INTERRUPTS
003B FB                 MOV     CX,6628                 ; WAIT 100 MILLISECONDS FOR ANY
003C B9 19E4            CALL    WAITF                   ;  INTERRUPTS THAT OCCUR
003F E8 0000 E          CMP     @INTR_FLAG,00H          ; DID ANY INTERRUPTS OCCUR?
0042 80 3E 006B R 00    JZ      D7                      ; NO - GO TO NEXT TEST
0047 74 0D
                        MOV     @MFG_ERR_FLAG,05H       ;       <><><><><><><><>
0049 C6 06 0015 R 05    D6:     MOV     @MFG_ERR_FLAG,05H       ;       <><><><><><><><>
                                                        ;       <> CHECKPOINT   5 <>
004E BE 0000 E          MOV     SI,OFFSET E101          ; DISPLAY 101 ERROR
0051 E8 0000 E          D6A:    CALL    E_MSG
0054 FA                 CLI
0055 F4                 HLT                             ; HALT THE SYSTEM

                ;-----  CHECK THE CONVERTING LOGIC

                        D7:     MOV     AL,27H                  ;       <><><><><><><><>
0056 B0 27              OUT     MFG_PORT,AL             ;       <> CHECKPOINT  27 <>
0058 E6 80
                        MOV     AX,0AA55H
005A B8 AA55            OUT     MFG_PORT+2,AX           ; WRITE A WORD
005D E7 82              IN      AL,MFG_PORT+2           ;  GET THE FIRST BYTE
005F E4 82              XCHG    AL,AH                   ;  SAVE IT
0061 86 C4              IN      AL,MFG_PORT+3           ;  GET THE SECOND BYTE
0063 E4 83              CMP     AX,55AAH                ; IS IT OK?
0065 3D 55AA            JZ      D7_A                    ; GO IF YES
0068 74 05
                        MOV     SI,OFFSET E106          ; DISPLAY 106 ERROR
006A BE 0000 E          JMP     D6A
006D EB E2

                ;-----  CHECK FOR HOT NMI INTERRUPTS WITHOUT I/O-MEMORY PARITY ENABLED

006F                    D7_A:
                        MOV     AL,CMOS_SHUT_DOWN       ; TURN ON NMI
006F B0 0F              OUT     CMOS_PORT,AL            ; ADDRESS DEFAULT READ ONLY REGISTER
0071 E6 70              MOV     CX,7                    ; DELAY COUNT FOR 100 MICROSECONDS
0073 B9 0007            CALL    WAITF                   ; WAIT FOR HOT NMI TO PROCESS
0076 E8 0000 E          MOV     AL,CMOS_SHUT_DOWN+NMI   ; TURN NMI ENABLE BACK OFF
0079 B0 8F              OUT     CMOS_PORT,AL
007B E6 70              CMP     @INTR_FLAG,00H          ; DID ANY INTERRUPTS OCCUR?
007D 80 3E 006B R 00    JZ      D7_C                    ; CONTINUE IF NOT
0082 74 09
                        MOV     AL,28H                  ;       <><><><><><><><>
0084 B0 28              OUT     MFG_PORT,AL             ;       <> CHECKPOINT  28 <>
0086 E6 80
                        MOV     SI,OFFSET E107          ; DISPLAY 107 ERROR
0088 BE 0000 E          JMP     D6A
008B EB C4

                ;-----  TEST THE DATA BUS TO TIMER 2

                        D7_C:   MOV     AL,29H                  ;       <><><><><><><><>
008D B0 29              OUT     MFG_PORT,AL             ;       <> CHECKPOINT  29 <>
008F E6 80              IN      AL,PORT_B               ; GET CURRENT SETTING OF PORT
0091 E4 61              MOV     AH,AL                   ; SAVE THAT SETTING
0093 8A E0              AND     AL,0FCH                 ; INSURE SPEAKER OFF
0095 24 FC              OUT     PORT_B,AL
0097 E6 61
                        MOV     AL,10110000B            ; SELECT TIM 2,LSB,MSB,BINARY,MODE 0
0099 B0 B0              OUT     TIMER+3,AL              ; WRITE THE TIMER MODE REGISTER
009B E6 43              JMP     $+2                     ; I/O DELAY
009D EB 00              MOV     AX,0AA55H               ; WRITE AN AA55
009F B8 AA55            OUT     TIMER+2,AL              ; WRITE TIMER 2 COUNT - LSB
00A2 E6 42              JMP     $+2                     ; I/O DELAY
00A4 EB 00              MOV     AL,AH
00A6 8A C4              OUT     TIMER+2,AL              ; WRITE TIMER 2 COUNT - MSB
00A8 E6 42              JMP     $+2                     ; I/O DELAY
00AA EB 00              IN      AL,TIMER+2              ; GET THE LSB
00AC E4 42              XCHG    AH,AL                   ; SAVE IT
00AE 86 E0              JMP     $+2                     ; I/O DELAY
00B0 EB 00              IN      AL,TIMER+2              ; GET THE MSB
00B2 E4 42              CMP     AX,055AAH               ; BUS OK?
00B4 3D 55AA            JZ      D7_D                    ; GO IF OK
00B7 74 05
                        MOV     SI,OFFSET E108          ; DISPLAY 108 ERROR
00B9 BE 0000 E          JMP     D6A
00BC EB 93

                ;--------------------------------------------------------
                ; TEST.18                                               :
                ;       8254 TIMER CHECKOUT                             :
                ; DESCRIPTION                                          :
                ;       VERIFY THAT THE SYSTEM TIMER (0) DOESN'T COUNT  :
                ;       TOO FAST OR TOO SLOW.                           :
                ;--------------------------------------------------------

                        D7_D:   MOV     AL,2AH                  ;       <><><><><><><><>
00BE B0 2A              OUT     MFG_PORT,AL             ;       <> CHECKPOINT  2A <>
00C0 E6 80              CLI                             ;
00C2 FA                 MOV     AL,0FEH                 ; MASK ALL INTERRUPTS EXCEPT LEVEL 0
00C3 B0 FE              OUT     INTA01,AL               ; WRITE THE 8259 IMR
00C5 E6 21              MOV     AL,00010000B            ; SELECT TIM 0, LSB, MODE 0, BINARY
00C7 B0 10              OUT     TIMER+3,AL              ; WRITE TIMER CONTROL MODE REGISTER
00C9 E6 43              MOV     CX,2CH                  ; SET PROGRAM LOOP COUNT
00CB B9 002C
                        JMP     $+2                     ; I/O DELAY
00CE EB 00
```

```
00D0 8A C1                      MOV     AL,CL                   ; SET TIMER 0 COUNT REGISTER
00D2 E6 40                      OUT     TIMER+0,AL              ; WRITE TIMER 0 COUNT REGISTER
00D4 FB                         STI
00D5 F6 06 006B R 01    D8:     TEST    @INTR_FLAG,01H
                                                                ; DID TIMER 0 INTERRUPT OCCUR?
00DA 75 0D                      JNZ     D9                      ; CHECK TIMER OPERATION FOR SLOW TIME
00DC E2 F7                      LOOP    D8                      ; WAIT FOR INTERRUPT FOR SPECIFIED TIME

00DE C6 06 0015 R 02            MOV     @MFG_ERR_FLAG,02H       ;       <><><><><><><><><><><><><><><>
                                                                ;       <><> TIMER CHECKPOINT (2) <><>

00E3 BE 0000 E          D8_A:   MOV     SI,OFFSET E102          ; DISPLAY 102 ERROR
00E6 E9 0051 R                  JMP     D6A                     ; TIMER 0 INTERRUPT DID NOT OCCUR= ERROR


00E9 B0 2B              D9:     MOV     AL,2BH                  ;       <><><><><><><><><><><><><><><>
00EB E6 80                      OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  2B <><>

00ED FA                         CLI
00EE B1 0C                      MOV     CL,12                   ; SET PROGRAM LOOP COUNT
00F0 B0 FF                      MOV     AL,0FFH                 ; WRITE TIMER 0 COUNT REGISTER
00F2 E6 40                      OUT     TIMER+0,AL
00F4 C6 06 006B R 00            MOV     @INTR_FLAG,0            ; RESET INTERRUPT RECEIVED FLAG
00F9 B0 FE                      MOV     AL,0FEH                 ; RE-ENABLE TIMER 0 INTERRUPTS
00FB E6 21                      OUT     INTA01,AL
00FD FB                         STI
00FE F6 06 006B R 01    D10:    TEST    @INTR_FLAG,01H          ; DID TIMER 0 INTERRUPT OCCUR?
0103 75 DE                      JNZ     D8_A                    ; YES - TIMER COUNTING TOO FAST, ERROR
0105 E2 F7                      LOOP    D10                     ; WAIT FOR INTERRUPT FOR SPECIFIED TIME

                        ;-----  WAIT FOR INTERRUPT

0107 2B C9                      SUB     CX,CX

0109 B0 2C                      MOV     AL,2CH                  ;       <><><><><><><><><><><><><><><>
010B E6 80                      OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  2C <><>
010D                    D110:
010D F6 06 006B R 01            TEST    @INTR_FLAG,01H          ; DID TIMER 0 INTERRUPT OCCUR?
0112 75 08                      JNZ     D12                     ; GO IF YES
0114 E2 F7                      LOOP    D110                    ; TRY AGAIN

0116 BE 0000 E                  MOV     SI,OFFSET E103          ; DISPLAY 103 ERROR
0119 E9 0051 R                  JMP     D6A                     ; ERROR IF NOT

                        ;-----  SETUP TIMER 0 TO MODE 3

011C FA                 D12:    CLI
011D B0 FF                      MOV     AL,0FFH                 ; DISABLE ALL DEVICE INTERRUPTS
011F E6 21                      OUT     INTA01,AL
0121 B0 36                      MOV     AL,36H                  ; SELECT TIMER 0,LSB,MSB,MODE 3
0123 E6 43                      OUT     TIMER+3,AL              ; WRITE TIMER MODE REGISTER
0125 EB 00                      JMP     $+2                     ; I/O DELAY
0127 B0 00                      MOV     AL,0
0129 E6 40                      OUT     TIMER+0,AL              ; WRITE LSB TO TIMER 0 REGISTER
012B EB 00                      JMP     $+2                     ; I/O DELAY
012D E6 40                      OUT     TIMER+0,AL              ; WRITE MSB TO TIMER 0 REGISTER

                        ;-----  CHECK 8042 FOR LAST COMMAND ACCEPTED

012F 2B C9                      SUB     CX,CX                   ; SET WAIT TIME
0131 B0 2D                      MOV     AL,2DH                  ;       <><><><><><><><><><><><><><><>
0133 E6 80                      OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  2D <><>
0135 E4 64              D13:    IN      AL,STATUS_PORT          ; GET THE 8042 STATUS
0137 A8 02                      TEST    AL,INPT_BUF_FULL        ; HAS THE LAST COMMAND BEEN ACCEPTED?
0139 74 08                      JZ      E19                     ; GO IF YES
013B E2 F8                      LOOP    D13                     ; TRY AGAIN

                        ;-----  ERROR EXIT (MESSAGE 105)

013D BE 0000 E                  MOV     SI,OFFSET E105          ; PRINT 105 ERROR
0140 E9 0051 R                  JMP     D6A                     ; GO ERROR HALT

                        ;--------------------------------------------------------------------------
                        ; TEST.19                                                                 :
                        ;       ADDITIONAL READ/WRITE STORAGE TEST                                :
                        ;       ++++ MUST RUN IN PROTECTED MODE ++++                              :
                        ; DESCRIPTION                                                             :
                        ;       WRITE/READ DATA PATTERNS TO ANY READ/WRITE STORAGE AFTER THE       :
                        ;       FIRST 64K.  STORAGE ADDRESSABILITY IS CHECKED.                    :
                        ;--------------------------------------------------------------------------

                                ASSUME  DS:DATA
0143                    E19:
0143 E8 0000 E                  CALL    DDS                     ; SET DATA SEGMENT
0146 B0 2F                      MOV     AL,2FH                  ;       <><><><><><><><><><><><><><><>
0148 E6 80                      OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  2F <><>

014A 81 3E 0072 R 1234          CMP     @RESET_FLAG,1234H       ; WARM START?
0150 75 03                      JNE     E19A                    ; GO IF NOT
0152 E9 0420 R                  JMP     SHUT2                   ; GO TO NEXT TEST IF WARM START

                        ;-----  SET SHUTDOWN RETURN 2

0155 B0 30              E19A:   MOV     AL,30H                  ;       <><><><><><><><><><><><><><><>
0157 E6 80                      OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  30 <><>

0159 B8 028F                    MOV     AX,2*H+CMOS_SHUT_DOWN+NMI       ; ADDRESS FOR SHUTDOWN BYTE
015C E8 0000 E                  CALL    CMOS_WRITE              ; SECOND ENTRY IN SHUTDOWN TABLE

                        ;-----  ENABLE PROTECTED MODE

015F BC 0000                    MOV     SP,POST_SS              ; SET STACK FOR SYSINIT1
0162 8E D4                      MOV     SS,SP
0164 BC 8000                    MOV     SP,POST_SP

0167 E8 0000 E                  CALL    SYSINIT1                ; GO ENABLE PROTECTED MODE

016A B0 31                      MOV     AL,31H                  ;       <><><><><><><><><><><><><><><>
016C E6 80                      OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  31 <><>

                        ;-----  SET TEMPORARY STACK

016E 6A 08                      PUSH    BYTE PTR GDT_PTR
0170 07                         POP     ES
```

```
0171 26: C7 06 006A 0000          MOV     ES:SS_TEMP.BASE_LO_WORD,0
0178 26: C6 06 005C 00            MOV     BYTE PTR ES:(SS_TEMP.BASE_HI_BYTE),0
017E 8E 0058                      MOV     SI,SS_TEMP
0181 8E D6                        MOV     SS,SI
0183 BC FFFD                      MOV     SP,MAX_SEG_LEN-2

                                  ;-----  DATA SEGMENT TO SYSTEM DATA AREA

0186 6A 18                        PUSH    BYTE PTR RSDA_PTR       ; POINT TO DATA AREA
0188 1F                           POP     DS

0189 B0 80                        MOV     AL,PARITY_CHECK         ; SET CHECK PARITY
018B E6 87                        OUT     DMA_PAGE+6,AL           ; SAVE WHICH CHECK TO USE

                                  ;-----  PRINT 64 K BYTES OK

018D B8 0040                      MOV     AX,64                   ; STARTING AMOUNT OF MEMORY OK
0190 E8 09A0 R                    CALL    PRT_OK                  ; POST 65K OK MESSAGE

                                  ;-----  GET THE MEMORY SIZE DETERMINED (PREPARE BX AND DX FOR BAD CMOS)

0193 B8 B0B1                      MOV     AX,(CMOS_U_M_S_LO+NMI)*H+CMOS_U_M_S_HI+NMI
0196 E8 0000 E                    CALL    CMOS_READ               ; HIGH BYTE
0199 86 E0                        XCHG    AH,AL                   ; SAVE HIGH BYTE
019B E8 0000 E                    CALL    CMOS_READ               ; LOW BYTE
019E 8B 1E 0013 R                 MOV     BX,@MEMORY_SIZE         ; LOAD THE BASE MEMORY SIZE
01A2 8B D3                        MOV     DX,BX                   ; SAVE BASE MEMORY SIZE
01A4 03 D8                        ADD     BX,AX                   ; SET TOTAL MEMORY SIZE

                                  ;-----  IS CMOS GOOD?

01A6 B0 8E                        MOV     AL,CMOS_DIAG+NMI        ; DETERMINE THE CONDITION OF CMOS
01A8 E8 0000 E                    CALL    CMOS_READ               ; GET THE CMOS STATUS

01AB A8 C0                        TEST    AL,BAD_BAT+BAD_CKSUM    ; CMOS OK?
01AD 74 02                        JZ      E20B0                   ; GO IF YES
01AF EB 6B                        JMP     SHORT E20C              ; DEFAULT IF NOT

                                  ;-----  GET THE BASE 0->640K MEMORY SIZE FROM CONFIGURATION IN CMOS

01B1                              E20B0:
01B1 B8 9596                      MOV     AX,(CMOS_B_M_S_LO+NMI)*H+CMOS_B_M_S_HI+NMI
01B4 E8 0000 E                    CALL    CMOS_READ               ; HIGH BYTE
01B7 24 3F                        AND     AL,03FH                 ; MASK OFF THE MANUFACTURING TEST BITS
01B9 86 E0                        XCHG    AH,AL                   ; SAVE HIGH BYTE
01BB E8 0000 E                    CALL    CMOS_READ               ; LOW BYTE OF BASE MEMORY SIZE
01BE 3B D0                        CMP     DX,AX                   ; IS MEMORY SIZE GREATER THAN CONFIG?
01C0 74 13                        JZ      E20B1                   ; GO IF EQUAL

                                  ;-----  SET MEMORY SIZE DETERMINE NOT EQUAL TO CONFIGURATION

01C2 50                           PUSH    AX                      ; SAVE AX
01C3 B8 8E8E                      MOV     AX,X*(CMOS_DIAG+NMI)    ; ADDRESS THE STATUS BYTE
01C6 E8 0000 E                    CALL    CMOS_READ               ; GET THE STATUS
01C9 0C 10                        OR      AL,W_MEM_SIZE           ; SET CMOS FLAG
01CB 86 C4                        XCHG    AL,AH                   ; SAVE AL AND GET ADDRESS
01CD E8 0000 E                    CALL    CMOS_WRITE              ; WRITE UPDATED STATUS
01D0 58                           POP     AX                      ; RESTORE AX
01D1 3B D0                        CMP     DX,AX                   ; IS MEMORY SIZE GREATER THAN CONFIG ?
01D3 77 37                        JA      E20C                    ; DEFAULT TO MEMORY SIZE DETERMINED ?
01D5                              E20B1:
01D5 8B D8                        MOV     BX,AX                   ; SET BASE MEMORY SIZE IN TOTAL REGISTER
01D7 8B D0                        MOV     DX,AX                   ; SAVE IN BASE SIZE REGISTER

                                  ;-----  CHECK MEMORY SIZE ABOVE 640K FROM CONFIGURATION

01D9 B8 9798                      MOV     AX,(CMOS_E_M_S_LO+NMI)*H+(CMOS_E_M_S_HI+NMI)
01DC E8 0000 E                    CALL    CMOS_READ               ; HIGH BYTE
01DF 86 E0                        XCHG    AH,AL                   ; SAVE HIGH BYTE
01E1 E8 0000 E                    CALL    CMOS_READ               ; LOW BYTE
01E4 8B C8                        MOV     CX,AX                   ; SAVE THE ABOVE 640K MEMORY SIZE
                                  ;-----  ABOVE 640K SIZE FROM MEMORY SIZE DETERMINE
                                  ;-----  CX=CONFIG  AX=MEMORY SIZE DETERMINE
01E6 B8 B0B1                      MOV     AX,(CMOS_U_M_S_LO+NMI)*H+(CMOS_U_M_S_HI+NMI)
01E9 E8 0000 E                    CALL    CMOS_READ               ; HIGH BYTE
01EC 86 E0                        XCHG    AH,AL                   ; SAVE HIGH BYTE
01EE E8 0000 E                    CALL    CMOS_READ               ; LOW BYTE
                                  ;------ WHICH IS GREATER - AX = MEMORY SIZE DETERMINE
                                  ;------ CX = CONFIGURATION (ABOVE 640) BX = SIZE (BELOW 640)

01F1 3B C8                        CMP     CX,AX                   ; IS CONFIGURATION EQUAL TO DETERMINED?
01F3 74 0F                        JZ      SET_MEM1                ; GO IF EQUAL

                                  ;-----  SET MEMORY SIZE DETERMINE NOT EQUAL TO CONFIGURATION

01F5 50                           PUSH    AX                      ; SAVE AX
01F6 B8 8E8E                      MOV     AX,X*(CMOS_DIAG+NMI)    ; ADDRESS THE STATUS BYTE
01F9 E8 0000 E                    CALL    CMOS_READ               ; GET THE STATUS
01FC 0C 10                        OR      AL,W_MEM_SIZE           ; SET CMOS FLAG
01FE 86 C4                        XCHG    AL,AH                   ; SAVE AL
0200 E8 0000 E                    CALL    CMOS_WRITE              ; UPDATE STATUS BYTE
0203 58                           POP     AX                      ; RESTORE AX

0204                              SET_MEM1:
0204 3B C8                        CMP     CX,AX                   ; IS CONFIG GREATER THAN DETERMINED?
0206 77 02                        JA      SET_MEM                 ;  GO IF YES
0208 8B C8                        MOV     CX,AX                   ; USE MEMORY SIZE DETERMINE IF NOT
020A                              SET_MEM:
020A 03 D9                        ADD     BX,CX                   ; SET TOTAL MEMORY SIZE
020C                              E20C:
020C 81 FA 0201                   CMP     DX,513                  ; CHECK IF BASE MEMORY LESS 512K
0210 72 0D                        JB      NO_640                  ; GO IF YES

0212 B8 B3B3                      MOV     AX,X*(CMOS_INFO128+NMI) ; SET 640K BASE MEMORY BIT
0215 E8 0000 E                    CALL    CMOS_READ               ; GET THE CURRENT STATUS
0218 0C 80                        OR      AL,M640K                ; TURN ON 640K BIT IF NOT ALREADY ON
021A 86 C4                        XCHG    AL,AH                   ; SAVE THE CURRENT DIAGNOSTIC STATUS
021C E8 0000 E                    CALL    CMOS_WRITE              ; RESTORE THE STATUS
021F                              NO_640:
021F 89 1E 0017 R                 MOV     WORD PTR @KB_FLAG,BX    ; SAVE TOTAL SIZE FOR LATER TESTING
0223 C1 EB 06                     SHR     BX,6                    ; DIVIDE BY 64
0226 4B                           DEC     BX                      ; 1ST 64K ALREADY DONE
0227 C1 EA 06                     SHR     DX,6                    ; DIVIDE BY 64 FOR BASE
```

```
                        ;-----  SAVE COUNTS IN STACK FOR BOTH MEMORY AND ADDRESSING TESTS

022A 52                         PUSH    DX                      ; SAVE BASE MEMORY SIZE COUNT
022B 6A 40                      PUSH    BYTE PTR 64             ; SAVE STARTING AMOUNT OF MEMORY OK
022D 53                         PUSH    BX                      ; SAVE COUNT OF 64K BLOCKS TO BE TESTED

022E 52                         PUSH    DX                      ; SAVE BASE MEMORY SIZE COUNT
022F 6A 40                      PUSH    BYTE PTR 64             ; SAVE STARTING AMOUNT OF MEMORY OK
0231 53                         PUSH    BX                      ; SAVE COUNT OF 64K BLOCKS TO BE TESTED

                        ;-----  MODIFY DESCRIPTOR TABLES

0232 6A 08                      PUSH    BYTE PTR GDT_PTR        ; MODIFY THE DESCRIPTOR TABLE
0234 07                         POP     ES

                        ;-----  SET TEMPORARY ES DESCRIPTOR 64K SEGMENT LIMIT STARTING AT 000000

0235 26: C7 06 0048 FFFF        MOV     ES:ES_TEMP.SEG_LIMIT,MAX_SEG_LEN
023C 26: C7 06 004A 0000        MOV     ES:ES_TEMP.BASE_LO_WORD,0
0243 26: C6 06 004C 00          MOV     BYTE PTR ES:(ES_TEMP.BASE_HI_BYTE),0            ; FIRST 65K
0249 26: C6 06 004D 93          MOV     BYTE PTR ES:(ES_TEMP.DATA_ACC_RIGHTS),CPL0_DATA_ACCESS

                        ;-----  SET TEMPORARY DS DESCRIPTOR 64K SEGMENT LIMIT AT FIRST 65K BLOCK

024F 26: C7 06 0060 FFFF        MOV     ES:DS_TEMP.SEG_LIMIT,MAX_SEG_LEN
0256 26: C7 06 0062 0000        MOV     ES:DS_TEMP.BASE_LO_WORD,0
025D 26: C6 06 0064 00          MOV     BYTE PTR ES:(DS_TEMP.BASE_HI_BYTE),0
0263 26: C6 06 0065 93          MOV     BYTE PTR ES:(DS_TEMP.DATA_ACC_RIGHTS),CPL0_DATA_ACCESS

                        ;-----  TEMPORARY SEGMENT SAVE IN DMA PAGE REGISTER FOR SECOND 65K BLOCK

0269 2A C0                      SUB     AL,AL                   ; INITIALIZE VALUES TO 010000
026B E6 85                      OUT     DMA_PAGE+4,AL           ; HIGH BYTE OF LOW WORD OF SEGMENT
026D E6 86                      OUT     DMA_PAGE+5,AL           ; LOW BYTE OF LOW WORD OF SEGMENT
026F FE C0                      INC     AL                      ; SET HIGH BYTE OF SEGMENT WORD
0271 E6 84                      OUT     DMA_PAGE+3,AL           ; HIGH BYTE OF SEGMENT


                        ;-----  MEMORY TEST LOOP - POINT TO NEXT BLOCK OF 32K WORDS (64K)

0273                    E21:                                    ;       MEMORY TEST LOOP
0273 6A 08                      PUSH    BYTE PTR GDT_PTR        ; POINT TO START OF DESCRIPTOR TABLE
0275 1F                         POP     DS
0276 FE 06 0064                 INC     BYTE PTR DS:(DS_TEMP.BASE_HI_BYTE)      ; POINT TO NEXT BLOCK
027A FE 06 004C                 INC     BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE)

                        ;-----  CHECK FOR END OF 256K PLANAR MEMORY

027E 80 3E 0064 04              CMP     BYTE PTR DS:(DS_TEMP.BASE_HI_BYTE),04H
0283 72 04                      JB      E21_0                   ; GO IF STILL FIRST 256K OF BASE MEMORY

0285 B0 C0                      MOV     AL,PARITY_CHECK+IO_CHECK; CHECK FOR ANY TYPE OF PARITY ERROR
0287 E6 87                      OUT     DMA_PAGE+6,AL           ; AFTER FIRST 256K

                        ;-----  CHECK END OF FIRST 640K OR ABOVE (END OF MAXIMUM BASE MEMORY)
0289                    E21_0:
0289 80 3E 0064 0A              CMP     BYTE PTR DS:(DS_TEMP.BASE_HI_BYTE),0AH
028E 77 16                      JA      NEXT                    ; CONTINUE IF ABOVE 1 MEG

                        ;-----  CHECK FOR END OF BASE MEMORY TO BE TESTED

0290 59                         POP     CX                      ; GET COUNT
0291 5B                         POP     BX                      ; GET COUNT TESTED
0292 58                         POP     AX                      ; RECOVER COUNT OF BASE MEMORY BLOCKS
0293 50                         PUSH    AX                      ; SAVE BASE COUNT
0294 53                         PUSH    BX                      ; SAVE TESTED COUNT
0295 51                         PUSH    CX                      ; SAVE TOTAL COUNT
0296 38 06 0064                 CMP     BYTE PTR DS:(DS_TEMP.BASE_HI_BYTE),AL           ; MAX BASE COUNT
029A 72 0A                      JB      NEXT                    ; CONTINUE IF NOT DONE WITH BASE MEMORY

                        ;-----  DO ADDITIONAL STORAGE ABOVE 1 MEG

029C C6 06 0064 10              MOV     BYTE PTR DS:(DS_TEMP.BASE_HI_BYTE),10H
02A1 C6 06 004C 10              MOV     BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE),10H

                        ;-----  SAVE BASE_HI_BYTE IN DMA PAGE REGISTERS 3

02A6 A0 0064            NEXT:   MOV     AL,BYTE PTR DS:(DS_TEMP.BASE_HI_BYTE)
02A9 E6 84                      OUT     DMA_PAGE+3,AL           ; SAVE THE HIGH BYTE OF SEGMENT
                                                                ;  FOR POSSIBLE ERROR

                        ;-----  CHECK FOR TOP OF MEMORY (FE0000) 16 MEG

02AB 80 3E 004C FE              CMP     BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE),0FEH ; TOP OF MEMORY?
02B0 74 29                      JE      KB_LOOP3                ; EXIT NEXT TEST IF DONE

                        ;-----  SET ES AND DS REGISTERS TO MEMORY BLOCK

02B2 6A 60                      PUSH    BYTE PTR DS_TEMP
02B4 1F                         POP     DS
02B5 6A 48                      PUSH    BYTE PTR ES_TEMP
02B7 07                         POP     ES

02B8 B0 31                      MOV     AL,31H                  ;       <><><><><><><><><><>
02BA E6 80                      OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  31 <><>

02BC B9 8000                    MOV     CX,8000H                ; SET COUNT FOR 32K WORDS
02BF E8 0000 E                  CALL    STGTST_CNT
02C2 74 03                      JZ      N1                      ; SKIP IF OK
02C4 E9 0367 R                  JMP     E21A                    ; GO PRINT ERROR
02C7                    N1:
02C7 59                         POP     CX                      ; POP CX TO GET AX
02C8 58                         POP     AX                      ; RECOVER TESTED MEMORY

                        ;-----  WRITE THE CURRENT SIZE FOR (ADDRESS LINE 23-17 TEST) USED LATER

02C9 2B FF                      SUB     DI,DI                   ; POINT TO BEGINNING OR A BLOCK
02CB AB                         STOSW                           ; WRITE THE CURRENT SIZE
                                                                ;  AT THE STARTING ADDRESS
02CC 05 0040                    ADD     AX,64                   ; ADVANCE COUNT TO NEXT BLOCK
02CF 50                         PUSH    AX                      ; SAVE TESTED MEMORY
02D0 51                         PUSH    CX                      ; SAVE LOOP COUNT

02D1 E8 09A0 R                  CALL    PRT_OK                  ; DISPLAY "0XXXX OK" MESSAGE
```

```
02D4 59                         POP     CX                      ; RECOVER 64K BLOCK COUNT
02D5 49                         DEC     CX                      ; DECREMENT BLOCK COUNT FOR LOOP
02D6 E3 03                      JCXZ    KB_LOOP3                ; CONTINUE TO NEXT TEST IF DONE

02D8 51                         PUSH    CX                      ; SAVE LOOP COUNT
02D9 EB 98                      JMP     E21                     ; LOOP TILL ALL MEMORY CHECKED

02DB                    KB_LOOP3:                               ;       END MAIN TEST LOOP
02DB 58                         POP     AX                      ; CLEAR MAXIMUM BLOCK COUNT
02DC 58                         POP     AX                      ; CLEAR BASE SIZE COUNT FROM STACK
                                                                ; ADDRESS TEST VALUES ARE IN STACK
                        ;-----  ADDRESS LINE 16-23 TEST
                                                                ;       LET FIRST PASS BE SEEN
02DD B9 40B8                    MOV     CX,16571                ; COUNT FOR 250 MS FIXED TIME DELAY
02E0 E8 0000 E                  CALL    WAITF                   ; ALLOW SIX DISPLAY REFRESH CYCLES

                        ;-----  INITIALIZE DS DESCRIPTOR

02E3 6A 08                      PUSH    BYTE PTR GDT_PTR
02E5 07                         POP     ES
02E6 26: C6 06 0064 00          MOV     BYTE PTR ES:(DS_TEMP.BASE_HI_BYTE),0
02EC 26: C7 06 0062 0000        MOV     ES:DS_TEMP.BASE_LO_WORD,0

                        ;-----  TEMPORARY SEGMENT SAVE IN DMA PAGE REGISTER

02F3 2A C0                      SUB     AL,AL
02F5 E6 85                      OUT     DMA_PAGE+4,AL           ; HIGH BYTE OF LOW WORD OF SEGMENT
02F7 E6 86                      OUT     DMA_PAGE+5,AL           ; LOW BYTE OF LOW WORD OF SEGMENT
02F9 B0 01                      MOV     AL,01H                  ; SET HIGH BYTE OF SEGMENT WORD
02FB E6 84                      OUT     DMA_PAGE+3,AL           ; HIGH BYTE OF SEGMENT

                        ;-----  POINT TO NEXT BLOCK OF 64K

02FD                    E21_A:
02FD B0 33                      MOV     AL,33H                  ;       <><><><><><><><><>
02FF E6 80                      OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  33 <><>
0301 26: 80 06 0064 01          ADD     BYTE PTR ES:(DS_TEMP.BASE_HI_BYTE),01

                        ;-----  CHECK FOR END OF BASE MEMORY TO BE TESTED

0307 26: 80 3E 0064 0A          CMP     BYTE PTR ES:(DS_TEMP.BASE_HI_BYTE),0AH
030D 77 13                      JA      NEXT_A                  ; CONTINUE IF ABOVE 1 MEG

030F 59                         POP     CX                      ; GET COUNT
0310 5B                         POP     BX                      ; GET COUNT TESTED
0311 58                         POP     AX                      ; RECOVER COUNT OF BASE MEMORY BLOCKS
0312 50                         PUSH    AX                      ; SAVE BASE COUNT
0313 53                         PUSH    BX                      ; SAVE TESTED COUNT
0314 51                         PUSH    CX                      ; SAVE TOTAL COUNT
0315 26: 38 06 0064             CMP     BYTE PTR ES:(DS_TEMP.BASE_HI_BYTE),AL           ; MAX BASE COUNT
031A 72 06                      JB      NEXT_A                  ; CONTINUE IF NOT DONE WITH BASE MEMORY

                        ;-----  DO ADDITIONAL STORAGE ABOVE 1 MEG

031C                    NEXT_A2:
031C 26: C6 06 0064 10          MOV     BYTE PTR ES:(DS_TEMP.BASE_HI_BYTE),10H
0322                    NEXT_A:
0322 26: A0 0064                MOV     AL,BYTE PTR ES:(DS_TEMP.BASE_HI_BYTE)

                        ;-----  DMA PAGE REGISTERS 3
                                                                ; SAVE THE HIGH BYTE OF SEGMENT
0326 E6 84                      OUT     DMA_PAGE+3,AL           ;  FOR POSSIBLE ERROR

                        ;-----  CHECK FOR TOP OF MEMORY (FE0000) 16 MEG

0328 3C FE                      CMP     AL,0FEH                 ; TOP OF MEMORY?
032A 74 34                      JZ      KB_LOOP_3               ; GO NEXT TEST IF IT IS

                        ;-----  SET DS REGISTER

032C 6A 60                      PUSH    BYTE PTR DS_TEMP
032E 1F                         POP     DS
032F 2B FF                      SUB     DI,DI                   ; POINT TO START OF BLOCK
0331 8B 15                      MOV     DX,DS:[DI]              ; GET THE VALUE OF THIS BLOCK
0333 8B F7                      MOV     SI,DI                   ; SET SI FOR POSSIBLE ERROR
0335 2B C0                      SUB     AX,AX                   ; CLEAR MEMORY LOCATION
0337 89 05                      MOV     [DI],AX

                        ;-----  ALLOW DISPLAY TIME TO DISPLAY MESSAGE AND REFRESH TO RUN

0339 B9 1A69                    MOV     CX,6761                 ; COUNT FOR 102 MS FIXED TIME DELAY
033C E8 0000 E                  CALL    WAITF                   ; ALLOW FIVE DISPLAY REFRESH CYCLES
033F 59                         POP     CX                      ; GET THE LOOP COUNT
0340 58                         POP     AX                      ; RECOVER TESTED MEMORY
0341 50                         PUSH    AX                      ; SAVE TESTED MEMORY
0342 51                         PUSH    CX                      ; SAVE LOOP COUNT
0343 3B C2                      CMP     AX,DX                   ; DOES THE BLOCK ID MATCH
0345 8B C2                      MOV     AX,DX                   ; GET THE BLOCK ID FOR POSSIBLE ERROR
0347 75 1E                      JNZ     E21A                    ; GO PRINT ERROR

                        ;-----  CHECK FOR CHECK PARITY

0349 E4 61                      IN      AL,PORT_B               ; CHECK FOR I/O OR PARITY CHECK
034B 24 C0                      AND     AL,PARITY_ERR           ; STRIP UNWANTED BITS
034D 75 18                      JNZ     E21A                    ; EXIT IF PARITY ERROR

034F 59                         POP     CX                      ; POP CX TO GET AX
0350 58                         POP     AX                      ; RECOVER TESTED MEMORY
0351 05 0040                    ADD     AX,64                   ; 64K INCREMENTS
0354 50                         PUSH    AX                      ; SAVE TESTED MEMORY
0355 51                         PUSH    CX                      ; SAVE LOOP COUNT
0356 E8 09A0 R                  CALL    PRT_OK                  ; DISPLAY OK MESSAGE
0359 59                         POP     CX                      ; RECOVER 64K BLOCK COUNT
035A 49                         DEC     CX                      ; LOOP TILL ALL MEMORY CHECKED
035B E3 03                      JCXZ    KB_LOOP_3               ; CONTINUE

035D 51                         PUSH    CX                      ; SAVE LOOP COUNT
035E EB 9D                      JMP     E21_A                   ; CONTINUE TILL DONE
```

```
                        PAGE
                        ;-----  BACK TO REAL MODE - MEMORY TESTS DONE

                        KB_LOOP_3:
0360                            MOV     AL,34H                  ;       <><><><><><><><><>
0360 B0 34                      OUT     MFG_PORT,AL             ;         <><> CHECKPOINT  34 <><>
0362 E6 80                                                      ; BACK TO REAL MODE
                                JMP     PROC_SHUTDOWN           ; NEXT TEST VIA JUMP TABLE (SHUT2)
0364 E9 0000 E


                        ;-----  PRINT FAILING ADDRESS AND XOR'ED PATTERN IF DATA COMPARE ERROR
                        ;-----  USE DMA PAGE REGISTERS AS TEMPORARY SAVE AREA FOR ERROR
                        ;       SET SHUTDOWN 3

                        E21A:   OUT     DMA_PAGE+1,AL           ; SAVE FAILING BIT PATTERN (LOW BYTE)
0367 E6 82                      MOV     AL,AH                   ; SAVE HIGH BYTE
0369 8A C4                      OUT     DMA_PAGE+2,AL
036B E6 83                      MOV     AX,SI                   ; GET THE FAILING OFFSET
036D 8B C6                      OUT     DMA_PAGE+5,AL
036F E6 86                      XCHG    AH,AL
0371 86 E0                      OUT     DMA_PAGE+4,AL
0373 E6 85

                        ;-----  CLEAR I/O CHANNEL CHECK OR R/W PARITY CHECK

                                SUB     SI,SI                   ; WRITE TO FAILING BLOCK
0376 2B F6                      STOSW
0377 AB                         IN      AL,PORT_B               ; GET PARITY CHECK LATCHES
0378 E4 61                      OUT     DMA_PAGE+7,AL           ; SAVE FOR ERROR HANDLER
037A E6 88                      OR      AL,RAM_PAR_OFF          ; TOGGLE I/O-PARITY CHECK ENABLE
037C 0C 0C                      OUT     PORT_B,AL               ;  TO RESET CHECKS
037E E6 61                      AND     AL,RAM_PAR_ON
0380 24 F3                      OUT     PORT_B,AL
0382 E6 61

                        ;-----  GET THE LAST OF GOOD MEMORY

                                POP     AX                      ; CLEAR BLOCK COUNT
0384 58                         POP     AX                      ; GET THE LAST OF GOOD MEMORY
0385 58                         POP     BX                      ; GET BASE MEMORY COUNTER
0386 5B                         SHL     BX,6                    ; CONVERT TO MEMORY SIZE COUNTS
0387 C1 E3 06                   SUB     AX,BX                   ; COMPARE LAST GOOD MEMORY WITH BASE
038A 2B C3                      JAE     E211                    ; IF ABOVE OR EQUAL, USE REMAINDER IN
038C 73 17                                                      ;  CMOS_U_M_S_(H/L)

                        ;-----  ELSE SET BASE MEMORY SIZE

                                PUSH    BYTE PTR RSDA_PTR       ; SET THE DATA SEGMENT
038E 6A 18                      POP     DS                      ; IN PROTECTED MODE
0390 1F

                                ADD     AX,BX                   ; CONVERT BACK TO LAST WORKING MEMORY
0391 03 C3                      MOV     @MEMORY_SIZE,AX         ; TO INDICATE HOW MUCH MEMORY WORKING
0393 A3 0013 R

                        ;-----  RESET 512K --> 640K OPTION IF SET

                                MOV     AX,X*(CMOS_INFO128+NMI) ; ADDRESS OPTIONS INFORMATION BYTE
0396 B8 B3B3                    CALL    CMOS_READ               ; READ THE MEMORY INFORMATION FLAG
0399 E8 0000 E                  AND     AL,NOT M640K            ; SET 640K OPTION OFF
039C 24 7F                      XCHG    AL,AH                   ; MOVE TO WORK REGISTER
039E 86 C4                      CALL    CMOS_WRITE              ; UPDATE STATUS IF IT WAS ON
03A0 E8 0000 E                  XOR     AX,AX                   ; CLEAR VALUE FOR EXTENSION MEMORY
03A3 33 C0                      MOV     CX,AX                   ; SAVE ADJUSTED MEMORY SIZE
                        E211:   MOV     AL,CMOS_U_M_S_HI+NMI
03A5                            CALL    CMOS_WRITE              ; SAVE THE HIGH BYTE MEMORY SIZE
03A5 8B C8                      MOV     AH,CL                   ; GET THE LOW BYTE
03A7 B0 B1                      MOV     AL,CMOS_U_M_S_LO+NMI    ; DO THE LOW BYTE
03A9 E8 0000 E                  CALL    CMOS_WRITE              ; WRITE IT
03AC 8A E1
03AE B0 B0
03B0 E8 0000 E

                        ;-----  SET SHUTDOWN 3

                                MOV     AX,3*H+CMOS_SHUT_DOWN+NMI       ; ADDRESS FOR SHUTDOWN RETURN
03B3 B8 038F                    CALL    CMOS_WRITE                      ; SET RETURN 3
03B6 E8 0000 E

                        ;-----  SHUTDOWN

                                JMP     PROC_SHUTDOWN
03B9 E9 0000 E
```

```
                        PAGE
                        ;--------------------------------------------------------------------------
                        ; MEMORY ERROR REPORTING          (R/W/ MEMORY OR PARITY ERRORS)          :
                        ;                                                                        :
                        ; DESCRIPTION FOR ERRORS 201 (CMP ERROR OR PARITY)                       :
                        ;                        OR 202 (ADDRESS LINE 0-15 ERROR)                :
                        ;                                                                        :
                        ;       "AABBCC DDEE 201" (OR 202)                                       :
                        ;                        AA=HIGH BYTE OF 24 BIT ADDRESS                  :
                        ;                        BB=MIDDLE BYTE OF 24 BIT ADDRESS                :
                        ;                        CC=LOW BYTE OF 24 BIT ADDRESS                   :
                        ;                        DD=HIGH BYTE OF XOR FAILING BIT PATTERN         :
                        ;                        EE=LOW BYTE OF XOR FAILING BIT PATTERN          :
                        ;                                                                        :
                        ; DESCRIPTION FOR ERROR 202 (ADDRESS LINE 00-15)                         :
                        ;       A WORD OF FFFF IS WRITTEN AT THE FIRST WORD AND LAST WORD        :
                        ;       OF EACH 64K BLOCK WITH ZEROS AT ALL OTHER LOCATIONS OF THE       :
                        ;       BLOCK.  A SCAN OF THE BLOCK IS MADE TO INSURE ADDRESS LINE       :
                        ;       0-15 ARE FUNCTIONING.                                            :
                        ;                                                                        :
                        ; DESCRIPTION FOR ERROR 203 (ADDRESS LINE 16-23)                         :
                        ;       AT THE LAST PASS OF THE STORAGE TEST, FOR EACH BLOCK OF           :
                        ;       64K, THE CURRENT STORAGE SIZE (ID) IS WRITTEN AT THE FIRST       :
                        ;       WORD OF EACH BLOCK. IT IS USED TO FIND ADDRESSING FAILURES.      :
                        ;                                                                        :
                        ;       "AABBCC DDEE 203"                                                :
                        ;                        SAME AS ABOVE EXCEPT FOR DDEE                   :
                        ;                                                                        :
                        ; GENERAL DESCRIPTION FOR BLOCK ID (DDEE WILL NOW CONTAINED THE ID)      :
                        ;       DD=HIGH BYTE OF BLOCK ID                                         :
                        ;       EE=LOW BYTE OF BLOCK ID                                          :
                        ;                                                                        :
                        ;       BLOCK ID        ADDRESS RANGE                                    :
                        ;       0000            000000 --> 00FFFF                                :
                        ;       0040            010000 --> 01FFFF                                :
                        ;       //                                                               :
                        ;       0200            090000 --> 09FFFF (512->576K) IF 640K BASE       :
                        ;                       100000 --> 10FFFF (1024->1088K) IF 512K BASE     :
                        ;                                                                        :
                        ; EXAMPLE (640K BASE MEMORY + 512K I/O MEMORY = 1152K TOTAL)            :
                        ;       NOTE: THE CORRECT BLOCK ID FOR THIS FAILURE IS 0280 HEX.         :
                        ;             DUE TO AN ADDRESS FAILURE THE BLOCK ID+128K OVERLAYED      :
                        ;             THE CORRECT BLOCK ID.                                      :
                        ;                                                                        :
                        ;       00640K OK           <-- LAST OK MEMORY                           :
                        ;       10000 0300 202      <-- ERROR DUE TO ADDRESS FAILURE             :
                        ;                                                                        :
                        ; IF A PARITY LATCH WAS SET THE CORRESPONDING MESSAGE WILL DISPLAY.      :
                        ;                                                                        :
                        ;       "PARITY CHECK 1" (OR 2)                                          :
                        ;                                                                        :
                        ; DMA PAGE REGISTERS ARE USED AS TEMPORARY SAVE AREAS FOR SEGMENT        :
                        ; DESCRIPTOR VALUES.                                                     :
                        ;--------------------------------------------------------------------------

 03BC                   SHUT3:                                  ;       ENTRY FROM PROCESSOR SHUTDOWN 3
 03BC E8 0000 E                 CALL    DDS                     ; SET REAL MODE DATA SEGMENT
                                                                ;
                                                                ;       <*> MEMORY FAILED  <*>
 03BF C6 06 0016 R 01           MOV     @MFG_ERR_FLAG+1,MEM_FAIL; CLEAR AND SET MANUFACTURING ERROR FLAG
 03C4 B0 0D                     MOV     AL,CR                   ; CARRIAGE RETURN
 03C6 E8 0000 E                 CALL    PRT_HEX
 03C9 B0 0A                     MOV     AL,LF                   ; LINE FEED
 03CB E8 0000 E                 CALL    PRT_HEX
 03CE E4 84                     IN      AL,DMA_PAGE+3           ; GET THE HIGH BYTE OF 24 BIT ADDRESS
 03D0 E8 0000 E                 CALL    XPC_BYTE                ; CONVERT AND PRINT CODE
 03D3 E4 85                     IN      AL,DMA_PAGE+4           ; GET THE MIDDLE BYTE OF 24 BIT ADDRESS
 03D5 E8 0000 E                 CALL    XPC_BYTE
 03D8 E4 86                     IN      AL,DMA_PAGE+5           ; GET THE LOW BYTE OF 24 BIT ADDRESS
 03DA E8 0000 E                 CALL    XPC_BYTE
 03DD B0 20                     MOV     AL,' '                  ; SPACE TO MESSAGE
 03DF E8 0000 E                 CALL    PRT_HEX
 03E2 E4 83                     IN      AL,DMA_PAGE+2           ; GET HIGH BYTE FAILING BIT PATTERN
 03E4 E8 0000 E                 CALL    XPC_BYTE                ; CONVERT AND PRINT CODE
 03E7 E4 82                     IN      AL,DMA_PAGE+1           ; GET LOW BYTE FAILING BIT PATTERN
 03E9 E8 0000 E                 CALL    XPC_BYTE                ; CONVERT AND PRINT CODE

                        ;-----  CHECK FOR ADDRESS ERROR

 03EC E4 80                     IN      AL,MFG_PORT             ; GET THE CHECKPOINT
 03EE 3C 33                     CMP     AL,33H                  ; IS IT AN ADDRESS FAILURE?
 03F0 BE 0000 E                 MOV     SI,OFFSET E203          ; LOAD ADDRESS ERROR 16->23
 03F3 74 0A                     JZ      ERR2                    ; GO IF YES

 03F5 BE 0000 E                 MOV     SI,OFFSET E202          ; LOAD ADDRESS ERROR 00->15
 03F8 3C 32                     CMP     AL,32H                  ; GO IF YES
 03FA 74 03                     JZ      ERR2

 03FC BE 0000 E                 MOV     SI,OFFSET E201          ; SETUP ADDRESS OF ERROR MESSAGE
 03FF                   ERR2:
 03FF E8 0000 E                 CALL    E_MSG                   ; PRINT ERROR MESSAGE
 0402 E4 88                     IN      AL,DMA_PAGE+7           ; GET THE PORT_B VALUE

                        ;-----  DISPLAY "PARITY CHECK ?" ERROR MESSAGES

 0404 A8 80                     TEST    AL,PARITY_CHECK         ; CHECK FOR PLANAR ERROR
 0406 74 0B                     JZ      NMI_M1                  ; SKIP IF NOT

 0408 50                        PUSH    AX                      ; SAVE STATUS
 0409 E8 0990 R                 CALL    PADING                  ; INSERT BLANKS
 040C BE 0000 E                 MOV     SI,OFFSET D1            ; PLANAR ERROR, ADDRESS "PARITY CHECK 1"
 040F E8 0000 E                 CALL    P_MSG                   ; DISPLAY "PARITY CHECK 1" MESSAGE
 0412 58                        POP     AX                      ; AND RECOVER STATUS
 0413                   NMI_M1:
 0413 A8 40                     TEST    AL,IO_CHECK             ; I/O PARITY CHECK ?
 0415 74 09                     JZ      NMI_M2                  ; SKIP IF CORRECT ERROR DISPLAYED

 0417 E8 0990 R                 CALL    PADING                  ; INSERT BLANKS
 041A BE 0000 E                 MOV     SI,OFFSET D2            ; ADDRESS OF "PARITY CHECK 2" MESSAGE
 041D E8 0000 E                 CALL    P_MSG                   ; DISPLAY "PARITY CHECK 2" ERROR
 0420                   NMI_M2:
                                                                ; CONTINUE TESTING SYSTEM ....
```

```
                        PAGE
                        ;-----  ENTRY FROM SHUTDOWN

0420                    SHUT2:

                        ;--------------------------------------------------------
                        ; TEST.20                                               :
                        ;       ADDITIONAL PROTECTED (VIRTUAL MODE) TEST        :
                        ; DESCRIPTION                                          :
                        ;       THE PROCESSOR IS PUT IN PROTECTED MODE AND      :
                        ;       THE FOLLOWING FUNCTIONS ARE VERIFIED            :
                        ;                                                       :
                        ;       1. VERIFY PROTECTED MODE                        :
                        ;          THE MACHINE STATUS IS CHECK FOR VIRTUAL MODE :
                        ;       2. PROGRAMMED INTERRUPT TEST                    :
                        ;          AN PROGRAMMED INTERRUPT 32 IS ISSUED AND     :
                        ;          AND VERIFIED                                 :
                        ;       3. EXCEPTION INTERRUPT 13 TEST                  :
                        ;          A DESCRIPTOR SEGMENT LIMIT IS SET TO ZERO    :
                        ;          AND A WRITE TO THAT SEGMENT IS ATTEMPTED     :
                        ;          AN EXCEPTION 13 IS EXPECTED AND VERIFIED     :
                        ;       4. LDT/SDT LTR/STR TEST                         :
                        ;          LOAD LDT REGISTER AND VERIFY CORRECT         :
                        ;          LOAD TASK REGISTER AND VERIFY CORRECT        :
                        ;          THEY ARE VERIFIED VIA THE STORE INSTRUCTION  :
                        ;       5. THE CONTROL FLAGS OF THE 286 FOR DIRECTION   :
                        ;          ARE VERIFIED VIA THE STD AND CLD COMMANDS     :
                        ;          IN PROTECTED MODE                            :
                        ;       6. BOUND INSTRUCTION TEST (EXCEPTION INT 5)     :
                        ;          CREATE A SIGNED ARRAY INDEX WITHIN AND       :
                        ;          OUTSIDE THE LIMITS.  CHECK THAT NO EXC INT   :
                        ;          IF WITHIN LIMIT AND THAT AN EXC INT 5        :
                        ;          OCCURS IF OUTSIDE THE LIMITS.                :
                        ;       7. PUSH ALL POP ALL TEST                        :
                        ;          SET GENERAL PURPOSE REGISTERS TO DIFFERENT   :
                        ;          VALUES, ISSUE A PUSH ALL, CLEAR THE REGISTERS:
                        ;          THEN ISSUE A POP ALL AND VERIFY CORRECT.     :
                        ;       8. CHECK THE VERR/VERW INSTRUCTIONS             :
                        ;          THE ACCESS BYTE IS SET TO READ ONLY THEN TO  :
                        ;          A WRITE ONLY AND THE VERR/VERW INSTRUCTIONS   :
                        ;          ARE VERIFIED.                                :
                        ;       9. CAUSE AN INTERRUPT 13 VIA A WRITE TO A       :
                        ;          READ ONLY SEGMENT                            :
                        ;      10. VERIFY THE ARPL INSTRUCTION FUNCTIONS        :
                        ;          SET THE RPL FIELD OF A SELECTOR AND          :
                        ;          VERIFY THAT CURRENT SELECTOR RPL IS SET      :
                        ;          CORRECTLY.                                   :
                        ;      11. VERIFY THE LAR INSTRUCTION FUNCTIONS         :
                        ;      12. VERIFY THE LSL INSTRUCTION FUNCTIONS         :
                        ;      13. LOW MEG CHIP SELECT TEST                     :
                        ;--------------------------------------------------------

0420 E9 0000 E                  JMP     POST3                   ; GO TEST THE 286 PROTECTED MODE

                        ;-----  FAILURE ENTRY FROM A SHUTDOWN

0423 E8 0000 E          SHUT7:  CALL    DDS                     ; ESTABLISH THE DATA SEGMENT
0426 E4 80                      IN      AL,MFG_PORT             ; CHECK FOR CHIP SELECT ERROR
0428 3C 35                      CMP     AL,35H
042A BE 0000 E                  MOV     SI,OFFSET E109          ; PRINT ERROR 109
042D 74 0D                      JZ      SHUT7B                  ; GO IF NOT
042F BE 0000 E          SHUT7A: MOV     SI,OFFSET E104          ; PROTECTED MODE FAILED

0432 80 0E 0016 R 02            OR      @MFG_ERR_FLAG+1,PRO_FAIL;       <><><><><><><><><><><><><><><><>
                                                                <>< VIRTUAL MODE FAILED  ><>

0437 E8 0000 E                  CALL    E_MSG                   ; PRINT MESSAGE
043A EB 08                      JMP     SHORT SHUT6
043C E8 0000 E          SHUT7B: CALL    E_MSG                   ; PRINT MESSAGE

043F 80 0E 0016 R 04            OR      @MFG_ERR_FLAG+1,LMCS_FAIL;      <><><><><><><><><><><><><><><><>
                                                                <>< LOW MEG CHIP SELECT  ><>

                        ;-----  PROTECTED MODE TEST PASSED ENTRY FROM A SHUTDOWN

0444 E8 0000 E          SHUT6:  CALL    DDS                     ; PROTECTED MODE TEST PASSED
0447 2B C0                      SUB     AX,AX                   ; CLEAR KEYBOARD STATE FLAGS
0449 A3 0017 R                  MOV     WORD PTR @KB_FLAG,AX
044C B9 000E                    MOV     CX,0EH                  ; CLEAR PAGE REGISTERS
044F BA 0082                    MOV     DX,DMA_PAGE+1
0452                    CLR_LOOP:
0452 2A C0                      SUB     AL,AL
0454 EE                         OUT     DX,AL
0455 42                         INC     DX
0456 E2 FA                      LOOP    CLR_LOOP

                        ;--------------------------------------------
                        ; TEST.21                                   :
                        ;       KEYBOARD TEST                       :
                        ; DESCRIPTION                              :
                        ;       RESET THE KEYBOARD AND CHECK THAT SCAN :
                        ;       CODE "AA" IS RETURNED TO THE PROCESSOR. :
                        ;       CHECK FOR STUCK KEYS.               :
                        ;--------------------------------------------

0458 B0 35                      MOV     AL,35H                  ;       <><><><><><><><><>
045A E6 80                      OUT     MFG_PORT,AL             ;       <>< CHECKPOINT  35 ><>

045C F6 06 0012 R 20            TEST    @MFG_TST,MFG_LOOP       ; MANUFACTURING BURN IN TEST MODE?
0461 75 03                      JNZ     F7_A
0463 E9 0516 R                  JMP     F7                      ; YES - SKIP KEYBOARD TEST
0466 80 3E 0072 R 64    F7_A:   CMP     BYTE PTR @RESET_FLAG,064H ; MANUFACTURING RUN IN MODE?
046B 75 03                      JNZ     F7_B
046D E9 0516 R                  JMP     F7                      ; YES - SKIP KEYBOARD TEST
0470 B0 36              F7_B:   MOV     AL,36H                  ;       <><><><><><><><><>
0472 E6 80                      OUT     MFG_PORT,AL             ;       <>< CHECKPOINT  36 ><>
0474 FA                         CLI
0475 81 3E 0072 R 1234          CMP     @RESET_FLAG,1234H       ; SOFT RESET?
047B 74 17                      JZ      G10
047D 80 3E 0072 R AA            CMP     BYTE PTR @RESET_FLAG,KB_OK ; CHECK FOR AA ALREADY RECEIVED
0482 74 10                      JZ      G10                     ; GO IF YES
0484 B0 AE                      MOV     AL,ENA_KBD
0486 E8 0000 E                  CALL    C8042                   ; ENABLE KEYBOARD
0489 B7 04                      MOV     BH,4                    ; TRY 4 TIMES
```

SECTION 5

```
                        LOOP1:  CALL    OBF_42                  ; CHECK FOR OUTPUT BUFFER FULL
                                JNZ     G10                     ; GO IF BUFFER FULL
                                DEC     BH
                                JNZ     LOOP1
                        G10:    MOV     AL,DIS_KBD              ; DISABLE KEYBOARD
                                CALL    C8042
                                IN      AL,PORT_A               ; FLUSH
                                MOV     AL,KYBD_CLK_DATA        ; GET THE CLOCK AND DATA LINES
                                CALL    C8042
                                CALL    OBF_42                  ; WAIT FOR OUTPUT BUFFER FULL
                                IN      AL,PORT_A               ; GET THE RESULTS
                                TEST    AL,KYBD_CLK             ; KEYBOARD CLOCK MUST BE LOW
                                JZ      G11

 04A9 80 0E 0016 R 08           OR      @MFG_ERR_FLAG+1,KYCLK_FAIL ;    <><><><><><><><><><><><><>
                                                                ;       <> KEYBOARD CLOCK HIGH  <>
 04AE BE 0000 E                 MOV     SI,OFFSET E304          ; DISPLAY 304 ERROR
 04B1 EB 60                     JMP     SHORT F6D               ; REPORT ERROR
 04B3 E8 0000 E                 CALL    KBD_RESET               ; ISSUE RESET TO KEYBOARD
 04B6 E3 29                     JCXZ    F6                      ; PRINT ERROR MESSAGE IF NO INTERRUPT
 04B8 B0 37                     MOV     AL,37H                  ;       <><><><><><><><><><><><><>
 04BA E6 80                     OUT     MFG_PORT,AL             ;       <> CHECKPOINT  37 <>
 04BC 80 FB AA                  CMP     BL,KB_OK                ; SCAN CODE AS EXPECTED?
 04BF 75 20                     JNE     F6                      ; NO - DISPLAY ERROR MESSAGE

                        ;-----  CHECK FOR STUCK KEYS

 04C1 B0 38                     MOV     AL,38H                  ;       <><><><><><><><><><><><><>
 04C3 E6 80                     OUT     MFG_PORT,AL             ;       <> CHECKPOINT  38 <>

 04C5 B0 AE                     MOV     AL,ENA_KBD              ; ASSURE KEYBOARD ENABLED
 04C7 E8 0000 E                 CALL    C8042                   ; ISSUE THE COMMAND
 04CA B9 19E4                   MOV     CX,6628                 ; COUNT FOR 100 MILLISECONDS
 04CD E8 0000 E                 CALL    WAITF                   ; DELAY FOR A WHILE
 04D0 E4 64                     IN      AL,STATUS_PORT          ; CHECK FOR STUCK KEYS
 04D2 A8 01                     TEST    AL,OUT_BUF_FULL         ; OUT BUFFER FULL?
 04D4 74 40                     JE      F7                      ; YES - CONTINUE TESTING

 04D6 B0 39                     MOV     AL,39H                  ;       <><><><><><><><><><><><><>
 04D8 E6 80                     OUT     MFG_PORT,AL             ;       <> CHECKPOINT  39 <>

 04DA E4 60                     IN      AL,PORT_A               ; GET THE SCAN CODE
 04DC E8 0000 E                 CALL    XPC_BYTE                ; CONVERT AND PRINT
 04DF EB 2A                     JMP     SHORT F6C               ; CONTINUE

                        ;-----  KEYBOARD ERROR TRY TO DETERMINE IF 8042 INTERFACE IS WORKING

 04E1 FA                F6:     CLI
 04E2 B0 AB                     MOV     AL,INTR_FACE_CK         ; COMMAND TO 8042
 04E4 E6 64                     OUT     STATUS_PORT,AL
 04E6 2B C9                     SUB     CX,CX
 04E8 B7 05                     MOV     BH,05                   ; WAIT FOR OUTPUT BUFFER FULL
 04EA E4 64             F6A:    IN      AL,STATUS_PORT
 04EC A8 01                     TEST    AL,OUT_BUF_FULL         ; 8042 FINISHED TEST?
 04EE E1 FA                     LOOPZ   F6A
 04F0 75 09                     JNZ     F6B                     ; GO CHECK RESULTS
 04F2 FE CF                     DEC     BH
 04F4 75 F4                     JNZ     F6A                     ; TRY AGAIN
 04F6 BE 0000 E                 MOV     SI,OFFSET E303          ; INDICATE PLANAR FAILURE
 04F9 EB 18                     JMP     SHORT F6D               ;    (REMOVE KEYBOARD TRY AGAIN)
 04FB E4 60             F6B:    IN      AL,PORT_A               ; GET THE RESULTS OF INTERFACE TEST
 04FD 3C 00                     CMP     AL,0                    ; IS THE INTERFACE OK?
 04FF 74 0A                     JZ      F6C
 0501 80 0E 0016 R 10           OR      @MFG_ERR_FLAG+1,KY_SYS_FAIL ;   <><><><><><><><><><><><><>
                                                                ;       <> KEYBOARD/SYSTEM  <>
 0506 BE 0000 E                 MOV     SI,OFFSET E303          ; PLANAR FAILURE
 0509 EB 08                     JMP     SHORT F6D               ; GO IF YES
 050B BE 0000 E         F6C:    MOV     SI,OFFSET E301          ; GET MESSAGE ADDRESS

 050E 80 0E 0016 R 20           OR      @MFG_ERR_FLAG+1,KYBD_FAIL;      <><><><><><><><><><><><><>
                                                                ;       <> KEYBOARD FAILED  <>

 0513 E8 0000 E         F6D:    CALL    E_MSG                   ; PRINT MESSAGE ON SCREEN

                        ;-----  INITIALIZE 8042 TO HONOR KEY LOCK

 0516 B0 3A             F7:     MOV     AL,3AH                  ;       <><><><><><><><><><><><><>
 0518 E6 80                     OUT     MFG_PORT,AL             ;       <> CHECKPOINT  3A <>

 051A B0 FF                     MOV     AL,0FFH                 ; DISABLE INTERRUPTS
 051C E6 21                     OUT     INTA01,AL
 051E FA                        CLI
 051F B0 60                     MOV     AL,WRITE_8042_LOC       ; WRITE 8042 MEMORY COMMAND
 0521 E8 0000 E                 CALL    C8042                   ; ISSUE THE COMMAND
 0524 B0 45                     MOV     AL,45H                  ; SET SYSTEM FLAG - OUTBUF INTERRUPT -
 0526 E6 60                     OUT     PORT_A,AL               ; PC I COMPATIBILITY
                                                                ; RESET INHIBIT OVER RIDE
                        ;-----  DEGATE ADDRESS LINE 20

 0528 B4 DD                     MOV     AH,DISABLE_BIT20        ; SET COMMAND IN AH
 052A E8 0000 E                 CALL    GATE_A20                ; ISSUE THE COMMAND

                        ;-----  SETUP HARDWARE INTERRUPT VECTOR TABLE LEVEL 0-7

 052D 2B C0                     SUB     AX,AX
 052F 8E C0                     MOV     ES,AX
 0531 B9 0008                   MOV     CX,08                   ; GET VECTOR COUNT
 0534 0E                        PUSH    CS                      ; SETUP DS SEGMENT REGISTER
 0535 1F                        POP     DS
 0536 BE 0000 E                 MOV     SI,OFFSET VECTOR_TABLE
 0539 BF 0020 R                 MOV     DI,OFFSET @INT_PTR
 053C A5                F7A:    MOVSW
 053D 47                        INC     DI                      ; SKIP OVER SEGMENT
 053E 47                        INC     DI
 053F E2 FB                     LOOP    F7A

                        ;-----  SETUP HARDWARE INTERRUPT VECTORS LEVEL 8-15 (VECTORS START AT INT 70H)

                                ASSUME  ES:ABS0
 0541 2B C0                     SUB     AX,AX
 0543 8E C0                     MOV     ES,AX
 0545 B9 0008                   MOV     CX,08                   ; GET VECTOR COUNT
 0548 0E                        PUSH    CS                      ; SETUP DS SEGMENT REGISTER
 0549 1F                        POP     DS
```

```
1093 054A BE 0000 E                     MOV     SI,OFFSET SLAVE_VECTOR_TABLE
1094 054D BF 01C0 R                     MOV     DI,OFFSET @SLAVE_INT_PTR
1095 0550 A5                    F7A1:   MOVSW
1096 0551 47                            INC     DI                      ; SKIP OVER SEGMENT
1097 0552 47                            INC     DI
1098 0553 E2 FB                         LOOP    F7A1
1099
1100                            ;-----  SET UP OTHER INTERRUPTS AS NECESSARY
1101
1102                                    ASSUME  DS:ABS0
1103 0555 2B C0                         SUB     AX,AX                   ; DS=0
1104 0557 8E D8                         MOV     DS,AX
1105 0559 C7 06 0008 R 0000 E           MOV     WORD PTR @NMI_PTR,OFFSET NMI_INT ; NMI INTERRUPT
1106 055F C7 06 0014 R 0000 E           MOV     WORD PTR @INT5_PTR,OFFSET PRINT_SCREEN  ; PRINT SCREEN
1107 0565 C7 06 0062 R F600             MOV     WORD PTR @BASIC_PTR+2,0F600H    ; SEGMENT FOR CASSETTE BASIC
1108
1109                            ;-----  ZERO RESERVED VECTORS
1110
1111 056B BF 0180                       MOV     DI,60H*4                ; FILL INTERRUPT 60 THRU 67 WITH ZERO
1112 056E B9 0010                       MOV     CX,16                   ; CLEAR 16 WORDS
1113 0571 C7 05 0000            F7A2:   MOV     WORD PTR DS:[DI],0
1114 0575 83 C7 02                      ADD     DI,2                    ; POINT TO NEXT LOCATION
1115 0578 E2 F7                         LOOP    F7A2
1116
1117                            ;-----  SETUP TIMER 0 TO BLINK LED IF MANUFACTURING TEST MODE
1118
1119                                    ASSUME  DS:DATA
1120 057A E8 0000 E                     CALL    DDS                     ; ESTABLISH DATA SEGMENT
1121
1122 057D F6 06 0012 R 20               TEST    @MFG_TST,MFG_LOOP       ; MFG. TEST MODE?
1123 0582 75 0B                         JNZ     F9
1124 0584 26: C7 06 0020 R 0000 E       MOV     WORD PTR ES:@INT_PTR,OFFSET BLINK_INT ; SETUP TIMER TO BLINK LED
1125 058B B0 FE                         MOV     AL,0FEH                 ; ENABLE TIMER INTERRUPT
1126 058D E6 21                         OUT     INTA01,AL
1127 058F FB                    F9:     STI                             ; ALLOW INTERRUPTS
1128
1129                            ;-----  ISSUE A RESET TO THE HARD FILE IF SOFT RESET
1130
1131 0590 81 3E 0072 R 1234             CMP     @RESET_FLAG,1234H       ; SOFT RESET?
1132 0596 75 0E                         JNZ     F9A                     ; CONTINUE IF NOT
1133 0598 B9 00FF                       MOV     CX,0FFH
1134 059B BA 03F6                       MOV     DX,03F6H
1135 059E B0 04                         MOV     AL,04H                  ; RESET
1136 05A0 EE                            OUT     DX,AL
1137 05A1 E2 FE                 F9_A:   LOOP    F9_A                    ; HOLD RESET
1138 05A3 2A C0                         SUB     AL,AL
1139 05A5 EE                            OUT     DX,AL                   ; REMOVE RESET
1140
1141                            ;--------------------------------------------------------
1142                            ; TEST.23                                              :
1143                            ;       DISKETTE ATTACHMENT TEST                       :
1144                            ; DESCRIPTION                                          :
1145                            ;       CHECK IF IPL DISKETTE DRIVE IS ATTACHED TO SYSTEM.  IF  :
1146                            ;       ATTACHED, VERIFY STATUS OF NEC FDC AFTER A RESET. ISSUE :
1147                            ;       A RECALIBRATE AND SEEK COMMAND TO FDC AND CHECK STATUS. :
1148                            ;       COMPLETE SYSTEM INITIALIZATION THEN PASS CONTROL TO THE :
1149                            ;       BOOT LOADER PROGRAM.                           :
1150                            ;--------------------------------------------------------
1151
1152 05A6 B0 3C                 F9A:    MOV     AL,3CH                  ;       <><><><><><><><><><>
1153 05A8 E6 80                         OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  3C <><>
1154
1155 05AA B0 02                         MOV     AL,02H                  ; SET DATA RATE TO 250 K BITS PER SECOND
1156 05AC BA 03F7                       MOV     DX,3F7H
1157 05AF EE                            OUT     DX,AL
1158 05B0 F6 06 0010 R 01               TEST    BYTE PTR @EQUIP_FLAG,1H ; DISKETTE PRESENT?
1159 05B5 74 55                         JZ      F15
1160 05B7 F6 06 0012 R 20               TEST    @MFG_TST,MFG_LOOP       ; MFG JUMPER INSTALLED?
1161 05BC 74 4E                         JZ      F15                     ; GO IF YES
1162 05BE                       F10:                                    ; DISK_TEST:
1163 05BE E4 21                         IN      AL,INTA01
1164 05C0 EB 00                         JMP     $+2                     ; I/O DELAY
1165 05C2 24 BF                         AND     AL,0BFH                 ; ENABLE DISKETTE INTERRUPTS
1166 05C4 E6 21                         OUT     INTA01,AL
1167 05C6 B4 00                         MOV     AH,0                    ; RESET NEC FDC
1168 05C8 8A D4                         MOV     DL,AH                   ; SET FOR DRIVE 0
1169 05CA CD 13                         INT     13H                     ; VERIFY STATUS AFTER RESET
1170 05CC F6 C4 FF                      TEST    AH,0FFH                 ; STATUS OK?
1171 05CF 75 25                         JNZ     F13                     ; NO - FDC FAILED
1172
1173                            ;-----  TURN DRIVE 0 MOTOR ON
1174
1175 05D1 BA 03F2                       MOV     DX,03F2H                ; GET ADDRESS OF FDC CARD
1176 05D4 B0 1C                         MOV     AL,1CH                  ; TURN MOTOR ON, ENABLE DMA, INTERRUPTS
1177 05D6 EE                            OUT     DX,AL                   ; WRITE FDC CONTROL REGISTER
1178 05D7 2B C9                         SUB     CX,CX                   ; WAITF COUNT FOR 0.988 SECONDS
1179 05D9 E8 0000 E                     CALL    WAITF                   ; WAIT 1 SECOND FOR MOTOR
1180
1181 05DC 33 FF                         XOR     DI,DI                   ; SELECT DRIVE 0
1182 05DE B5 01                         MOV     CH,1                    ; SELECT TRACK 1
1183 05E0 C6 06 003E R 00               MOV     @SEEK_STATUS,0          ; INSURE RECALIBRATE
1184 05E5 80 0E 00A0 R 01               OR      @RTC_WAIT_FLAG,01       ; NO REAL TIME CLOCK, USE WAIT LOOP
1185 05EA E8 0000 E                     CALL    SEEK                    ; RECALIBRATE DISKETTE
1186 05ED 72 07                         JC      F13                     ; GO TO ERR SUBROUTINE IF ERR
1187 05EF B5 22                         MOV     CH,34                   ; SELECT TRACK 34
1188 05F1 E8 0000 E                     CALL    SEEK                    ; SEEK TO TRACK 34
1189 05F4 73 0B                         JNC     F14                     ; OK, TURN MOTOR OFF
1190 05F6                       F13:                                    ; DSK_ERR:
1191 05F6 80 0E 0016 R 40               OR      @MFG_ERR_FLAG+1,DSK_FAIL;       <><><><><><><><><><><><><>
1192                                                                    ;       <><> DISKETTE FAILED  <><>
1193 05FB BE 0000 E                     MOV     SI,OFFSET E601          ; GET ADDRESS OF MESSAGE
1194 05FE E8 0000 E                     CALL    E_MSG                   ; GO PRINT ERROR MESSAGE
1195
1196                            ;-----  TURN DRIVE 0 MOTOR OFF
1197
1198 0601                       F14:                                    ; DRO_OFF:
1199 0601 80 26 00A0 R FE               AND     @RTC_WAIT_FLAG,0FEH     ; ALLOW FOR RTC WAIT
1200 0606 B0 0C                         MOV     AL,0CH                  ; TURN DRIVE 0 MOTOR OFF
1201 0608 BA 03F2                       MOV     DX,03F2H                ; FDC CONTROLLER ADDRESS
1202 060B EE                            OUT     DX,AL
```

SECTION 5

```
                        PAGE
                        ;-----  SETUP KEYBOARD PARAMETERS

060C C6 06 006B R 00    F15:    MOV     @INTR_FLAG,00H          ; SET STRAY INTERRUPT FLAG = 00
0611 BE 001E R                  MOV     SI,OFFSET @KB_BUFFER    ; SETUP KEYBOARD PARAMETERS
0614 89 36 001A R               MOV     @BUFFER_HEAD,SI
0618 89 36 001C R               MOV     @BUFFER_TAIL,SI
061C 89 36 0080 R               MOV     @BUFFER_START,SI
0620 83 C6 20                   ADD     SI,32                   ; DEFAULT BUFFER OF 32 BYTES
0623 89 36 0082 R               MOV     @BUFFER_END,SI

                        ;-----  SET PRINTER TIMEOUT DEFAULT

0627 BF 0078 R                  MOV     DI,OFFSET @PRINT_TIM_OUT; SET DEFAULT PRINTER TIMEOUT
062A 1E                         PUSH    DS
062B 07                         POP     ES
062C B8 1414                    MOV     AX,1414H                ; DEFAULT=20
062F AB                         STOSW
0630 AB                         STOSW

                        ;-----  SET RS232 DEFAULT

0631 B8 0101                    MOV     AX,0101H                ; RS232 DEFAULT=01
0634 AB                         STOSW
0635 AB                         STOSW

                        ;-----  ENABLE TIMER INTERRUPTS

0636 E4 21                      IN      AL,INTA01
0638 24 FE                      AND     AL,0FEH                 ; ENABLE TIMER INTERRUPTS
063A EB 00                      JMP     $+2                     ; I/O DELAY
063C E6 21                      OUT     INTA01,AL

                        ;-----  CHECK CMOS BATTERY AND CHECKSUM

063E F6 06 0012 R 20            TEST    @MFG_TST,MFG_LOOP       ; MFG JUMPER?
0643 75 03                      JNZ     B1_OK                   ; GO IF NOT
0645 E9 072E R                  JMP     F15C                    ; BYPASS IF YES
0648                    B1_OK:
0648 B0 8E                      MOV     AL,CMOS_DIAG+NMI        ; ADDRESS DIAGNOSTIC STATUS BYTE
064A E8 0000 E                  CALL    CMOS_READ               ; READ IT FROM CMOS

064D BE 0000 E                  MOV     SI,OFFSET E161          ; LOAD BAD BATTERY MESSAGE 161
0650 A8 80                      TEST    AL,BAD_BAT              ; BATTERY BAD?
0652 75 07                      JNZ     B1_ER                   ; DISPLAY ERROR IF BAD

0654 BE 0000 E                  MOV     SI,OFFSET E162          ; LOAD CHECKSUM BAD MESSAGE 162
0657 A8 60                      TEST    AL,BAD_CKSUM+BAD_CONFIG ; CHECK FOR CHECKSUM OR NO DISKETTE
0659 74 09                      JZ      C_OK                    ; SKIP AND CONTINUE TESTING CMOS CLOCK
065B                    B1_ER:
065B E8 0000 E                  CALL    E_MSG                   ; ELSE DISPLAY ERROR MESSAGE
065E 81 CD 8000                 OR      BP,08000H               ; FLAG "SET SYSTEM OPTIONS" DISPLAYED
0662 EB 45                      JMP     SHORT H_OK1A            ; SKIP CLOCK TESTING IF ERROR

                        ;-----  TEST CLOCK UPDATING

0664 B3 04              C_OK:   MOV     BL,04H                  ; OUTER LOOP COUNT
0666 2B C9              D_OK:   SUB     CX,CX                   ; INNER LOOP COUNT
0668 B0 8A              E_OK:   MOV     AL,CMOS_REG_A+NMI       ; GET THE CLOCK UPDATE BYTE
066A E8 0000 E                  CALL    CMOS_READ
066D A8 80                      TEST    AL,80H                  ; CHECK FOR UPDATE IN PROGRESS
066F 75 1B                      JNZ     G_OK                    ; GO IF YES
0671 E2 F5                      LOOP    E_OK                    ; TRY AGAIN
0673 FE CB                      DEC     BL                      ; DEC OUTER LOOP
0675 75 EF                      JNZ     D_OK                    ; TRY AGAIN
0677 BE 0000 E          F_OK:   MOV     SI,OFFSET E163          ; PRINT MESSAGE
067A E8 0000 E                  CALL    E_MSG

                        ;-----  SET CMOS DIAGNOSTIC STATUS TO 04 (CLOCK ERROR)

067D B8 0E8E                    MOV     AX,X*CMOS_DIAG+NMI      ; SET CLOCK ERROR
0680 E8 0000 E                  CALL    CMOS_READ               ; GET THE CURRENT STATUS
0683 0C 04                      OR      AL,CMOS_CLK_FAIL        ; SET NEW STATUS
0685 86 C4                      XCHG    AL,AH                   ; GET STATUS ADDRESS AND SAVE NEW STATUS
0687 E8 0000 E                  CALL    CMOS_WRITE              ; MOVE NEW DIAGNOSTIC STATUS TO CMOS
068A EB 0E                      JMP     SHORT H_OK              ; CONTINUE

                        ;-----  CHECK CLOCK UPDATE

068C B9 0320            G_OK:   MOV     CX,800                  ; LOOP COUNT
068F B0 8A              I_OK:   MOV     AL,CMOS_REG_A+NMI       ; CHECK FOR OPPOSITE STATE
0691 E8 0000 E                  CALL    CMOS_READ
0694 A8 80                      TEST    AL,80H
0696 E0 F7                      LOOPNZ  I_OK                    ; TRY AGAIN
0698 E3 DD                      JCXZ    F_OK                    ; PRINT ERROR IF TIMEOUT

                        ;-----  CHECK MEMORY SIZE DETERMINED = CONFIGURATION

069A                    H_OK:
069A B0 8E                      MOV     AL,CMOS_DIAG+NMI        ; GET THE STATUS BYTE
069C E8 0000 E                  CALL    CMOS_READ
069F A8 10                      TEST    AL,W_MEM_SIZE           ; WAS THE CONFIG= MEM_SIZE_DETERMINED?
06A1 74 06                      JZ      H_OK1A                  ; GO IF YES

                        ;-----  MEMORY SIZE ERROR

06A3 BE 0000 E                  MOV     SI,OFFSET E164          ; PRINT SIZE ERROR
06A6 E8 0000 E                  CALL    E_MSG                   ; DISPLAY ERROR

                        ;-----  CHECK FOR CRT ADAPTER ERROR

06A9 80 3E 0015 R 0C    H_OK1A: CMP     @MFG_ERR_FLAG,0CH       ; CHECK FOR MONOCHROME CRT ERROR
06AE BE 0000 E                  MOV     SI,OFFSET E401          ; LOAD MONOCHROME CRT ERROR
06B1 74 0A                      JZ      H_OK1B                  ; GO IF YES

06B3 80 3E 0015 R 0D            CMP     @MFG_ERR_FLAG,0DH       ; CHECK FOR COLOR CRT ADAPTER ERROR
06B8 75 06                      JNZ     J_OK                    ; CONTINUE IF NOT
06BA BE 0000 E                  MOV     SI,OFFSET E501          ; CRT ADAPTER ERROR MESSAGE
06BD                    H_OK1B:
06BD E8 0000 E                  CALL    E_MSG
```

```
                        PAGE
                        ;-----  CHECK FOR MULTIPLE DATA RATE CAPABILITY

0316 06C0               J_OK:
0317 06C0 BA 03F1               MOV     DX,03F1H                ; D/S/P DIAGNOSTIC REGISTER
0318 06C3 EC                    IN      AL,DX                   ; READ D/S/P TYPE CODE
0319 06C4 24 F8                 AND     AL,11111000B            ; KEEP ONLY UNIQUE CODE FOR D/S/P
0320 06C6 3C 50                 CMP     AL,01010000B            ; D/S/P CARD - MULTIPLE DATA RATE ?
0321 06C8 74 46                 JZ      J_OK3                   ; IF SO JUMP
0322
0323 06CA BA 05F7               MOV     DX,05F7H                ; FIXED DISK DIAGNOSTIC REGISTER
0324 06CD EC                    IN      AL,DX                   ; READ FIXED DISK TYPE CODE
0325 06CE 24 F0                 AND     AL,11110000B            ; KEEP ONLY UNIQUE CODE FOR F/D
0326 06D0 3C A0                 CMP     AL,10100000B            ; FIXED DISK ADAPTER ?
0327 06D2 74 2F                 JZ      J_FAIL                  ; MUST BE COMBO ELSE ERROR
0328
0329 06D4 B3 0F                 MOV     BL,0FH                  ; OUTER LOOP COUNT WAIT FOR BUSY OFF
0330 06D6 2B C9                 SUB     CX,CX
0331 06D8 BA 01F7               MOV     DX,01F7H                ; HARD FILE STATUS PORT
0332 06DB               J_OK1:
0333 06DB EC                    IN      AL,DX                   ; GET THE STATUS
0334 06DC A8 80                 TEST    AL,080H                 ; IS THE CONTROLLER BUSY?
0335 06DE 74 0C                 JZ      J_OK2                   ; CONTINUE IF NOT
0336 06E0 E2 F9                 LOOP    J_OK1                   ; TRY AGAIN
0337 06E2 FE CB                 DEC     BL                      ; DECREMENT OUTER LOOP
0338 06E4 75 F5                 JNZ     J_OK1                   ; TRY AGAIN IF NOT ZERO
0339 06E6 24 0C                 AND     AL,0CH                  ; BITS 2 & 3 = 0 IF MULTI DATA CAPABLE
0340 06E8 74 26                 JZ      J_OK3                   ; GO IF YES
0341 06EA EB 17                 JMP     SHORT J_FAIL            ; NO MULTIPLE DATA RATE CAPABILITY
0342 06EC               J_OK2:
0343 06EC BA 01F4               MOV     DX,1F4H                 ; VERIFY MULTIPLE DATA RATE CAPABLE
0344 06EF B0 55                 MOV     AL,055H                 ;  WRITE TO THE CYLINDER BYTE
0345 06F1 EE                    OUT     DX,AL
0346 06F2 EB 00                 JMP     $+2                     ; I/O DELAY
0347 06F4 EC                    IN      AL,DX                   ; CHECK DATA WRITTEN = DATA READ
0348 06F5 3C 55                 CMP     AL,055H
0349 06F7 75 0A                 JNZ     J_FAIL                  ; GO IF NOT
0350 06F9 B0 AA                 MOV     AL,0AAH                 ; WRITE ANOTHER PATTERN
0351 06FB EE                    OUT     DX,AL
0352 06FC EB 00                 JMP     $+2                     ; I/O DELAY
0353 06FE EC                    IN      AL,DX
0354 06FF 3C AA                 CMP     AL,0AAH                 ; IS DATA PATTERN THE SAME?
0355 0701 74 0D                 JZ      J_OK3                   ; GO IF SO
0356
0357 0703               J_FAIL: OR      @MFG_ERR_FLAG+1,DSK_FAIL; <><><><><><><><><><><><><>
                                                                ;       <><> DISKETTE FAILED  <><>
0358 0703 80 0E 0016 R 40
0359
0360 0708 BE 0000 E             MOV     SI,OFFSET E601          ; GET ADDRESS OF MESSAGE
0361 070B E8 0000 E             CALL    E_MSG                   ; GO PRINT ERROR MESSAGE
0362 070E EB 1E                 JMP     SHORT F15C              ; SKIP SETUP IF ERROR
0363
0364 0710               J_OK3:  OR      @LASTRATE,DUAL          ; TURN ON DSP/COMBO FLAG
0365 0710 80 0E 008B R 01
0366
0367                    ;-----  INITIALIZE FLOPPY FOR DRIVE TYPE
0368
0369 0715 B0 3D                 MOV     AL,3DH                  ;       <><><><><><><><><><><><><>
0370 0717 E6 80                 OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  3D <><>
0371 0719 E8 0000 E             CALL    DSKETTE_SETUP           ; INITIALIZE FLOPPY
0372
0373                    ;-----  CHECK FOR 2ND DISKETTE DRIVE
0374
0375 071C E8 0000 E             CALL    DDS                     ; INSURE DATA SEGMENT
0376 071F 8A 26 0091 R          MOV     AH,@DSK_STATE+1         ; GET STATE OF SECOND DRIVE
0377 0723 0A E4                 OR      AH,AH                   ; IS THERE A DRIVE 2 ATTACHED?
0378 0725 74 07                 JZ      F15C                    ; SKIP IF NOT
0379 0727 80 0E 0010 R 40       OR      BYTE PTR @EQUIP_FLAG,40H; ELSE SET SECOND DRIVE INSTALLED
0380 072C B4 FF                 MOV     AH,0FFH                 ; SET TEST MASK FOR DRIVE PRESENT
0381 072E               F15C:
0382 072E B0 0E                 MOV     AL,CMOS_DIAG+NMI        ; GET THE CMOS DIAGNOSTIC STATUS
0383 0730 E8 0000 E             CALL    CMOS_READ
0384 0733 A8 C0                 TEST    AL,BAD_BAT+BAD_CKSUM    ; BATTERY/CHECKSUM OK
0385 0735 75 22                 JNZ     ROM_SCAN1               ; BYPASS DISK SETUP IF NOT
0386
0387 0737 B0 90                 MOV     AL,CMOS_DISKETTE+NMI    ; ADDRESS DISKETTE TYPE BYTE
0388 0739 E8 0000 E             CALL    CMOS_READ               ; GET DISKETTE TYPES
0389 073C 24 0F                 AND     AL,00FH                 ; LOOK AT SECOND DRIVE TYPE DEFINED
0390 073E 3A C4                 CMP     AL,AH                   ; ARE BOTH INDICATORS ZERO
0391 0740 74 07                 JE      F15D                    ; SKIP IF NO SECOND DRIVE
0392
0393 0742 22 C4                 AND     AL,AH                   ; ARE BOTH INDICATORS NON-ZERO
0394 0744 75 03                 JNZ     F15D                    ; SKIP IF DRIVE FOUND AND DEFINED
0395
0396 0746 E8 0000 E             CALL    CONFIG_BAD              ; SET BAD_CONFIG IN CMOS_DIAG
0397
0398                    ;-----  INITIALIZE HARD FILE
0399
0400 0749               F15D:
0401 0749 B0 3E                 MOV     AL,3EH                  ;       <><><><><><><><><><><><><>
0402 074B E6 80                 OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  3E <><>
0403
0404 074D B0 92                 MOV     AL,CMOS_DISK+NMI        ; INSURE CMOS DEFINES TYPE OF FIXED DISK
0405 074F E8 0000 E             CALL    CMOS_READ
0406 0752 3C 00                 CMP     AL,0H                   ; INSURE TYPE IS DEFINED
0407 0754 74 03                 JZ      ROM_SCAN1               ; BYPASS DISK SETUP IF NOT
0408
0409 0756 E8 0000 E             CALL    DISK_SETUP              ; INITIALIZE HARD FILE(S)
0410
0411                    ;--------------------------------------------------------
0412                    ; TEST.22                                               :
0413                    ; CHECK FOR OPTIONAL ROM FROM C800->E000 IN 2K BLOCKS    :
0414                    ;       (A VALID MODULE HAS '55AA' IN THE FIRST 2 LOCATIONS:
0415                    ;       LENGTH INDICATOR (LENGTH/512) IN THE 3RD LOCATION  :
0416                    ;       AND TEST/INIT. CODE STARTING IN THE 4TH LOCATION)  :
0417                    ;--------------------------------------------------------
0418
0419 0759               ROM_SCAN1:
0420 0759 FB                    STI                             ; ALLOW INTERRUPTS
0421 075A B0 3B                 MOV     AL,3BH                  ;       <><><><><><><><><><><><><>
0422 075C E6 80                 OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  3B <><>
0423 075E E8 0000 E             CALL    DDS                     ; SET REAL MODE DATA SEGMENT
0424 0761 B0 0A                 MOV     AL,10                   ; LINE FEED ON DISPLAY
0425 0763 E8 0000 E             CALL    PRT_HEX
0426
```

SECTION 5

```
                        ROM_SCAN:
                        ;-----  SET DMA MASK AND REQUEST REGISTERS
0766 2A C0                      SUB     AL,AL
0768 E6 D2                      OUT     DMA18+2,AL              ; SEND ZERO TO MASK REGISTER
076A EB 00                      JMP     $+2
076C E6 D4                      OUT     DMA18+4,AL              ; SEND ZERO TO REQUEST REGISTER
076E BA C800                    MOV     DX,0C800H               ; SET BEGINNING ADDRESS
0771                    ROM_SCAN2:
0771 8E DA                      MOV     DS,DX
0773 57                         PUSH    DI                      ; SAVE WORK REGISTER
0774 BF AA55                    MOV     DI,0AA55H               ; GET TEST PATTERN
0777 2B DB                      SUB     BX,BX                   ; SET BX=0000
0779 8B 07                      MOV     AX,[BX]                 ; GET 1ST WORD FROM MODULE
077B 3B C7                      CMP     AX,DI                   ; = TO ID WORD?
077D 5F                         POP     DI                      ; RECOVER WORK REGISTER
077E 75 05                      JNZ     NEXT_ROM                ; PROCEED TO NEXT ROM IF NOT
0780 E8 0000 E                  CALL    ROM_CHECK               ; GO CHECK OUT MODULE
0783 EB 04                      JMP     SHORT ARE_WE_DONE       ; CHECK FOR END OF ROM SPACE
0785                    NEXT_ROM:
0785 81 C2 0080                 ADD     DX,0080H                ; POINT TO NEXT 2K ADDRESS
0789                    ARE_WE_DONE:
0789 81 FA E000                 CMP     DX,0E000H               ; AT E0000 YET?
078D 7C E2                      JL      ROM_SCAN2               ; GO CHECK ANOTHER ADD. IF NOT

                        ;-----  TEST FOR KEYBOARD LOCKED

078F E8 0000 E                  CALL    DDS                     ; SET DATA SEGMENT
0792 E4 64                      IN      AL,STATUS_PORT          ; IS KEYBOARD UNLOCKED?
0794 24 10                      AND     AL,KYBD_INH
0796 74 02                      JZ      KEY1                    ; NO - SET ERROR FLAGS AND PRINT MESSAGE
0798 EB 0B                      JMP     SHORT KEY10             ; GO IF OFF
079A                    KEY1:
079A 80 0E 0016 R 80            OR      @MFG_ERR_FLAG+1,KEY_FAIL;       <><><><><><><><><><><><><>
                                                                ;       <><> KEYBOARD IS LOCKED <><>
                                ASSUME  DS:DATA
079F BE 0000 E                  MOV     SI,OFFSET E302          ; PRINT LOCKED MESSAGE  (302)
07A2 E8 0000 E                  CALL    E_MSG
07A5                    KEY10:
                        ;====================
                        ;-----  SETUP @PRINTER_BASE
                        ;====================

07A5 BF 09D6 R                  MOV     DI,OFFSET F4            ; OFFSET OF PRINTER ADDRESS TABLE
07A8 BE 0000                    MOV     SI,0
07AB                    F16:
07AB 2E: 8B 15                  MOV     DX,CS:[DI]              ; GET PRINTER BASE ADDRESS
07AE B0 AA                      MOV     AL,0AAH                 ; WRITE DATA TO PORT A
07B0 EE                         OUT     DX,AL
07B1 EB 00                      JMP     $+2                     ; I/O DELAY
07B3 1E                         PUSH    DS                      ; BUS SETTLING
07B4 EC                         IN      AL,DX                   ; READ PORT A
07B5 1F                         POP     DS
07B6 3C AA                      CMP     AL,0AAH                 ; DATA PATTERN SAME
07B8 75 06                      JNE     F17                     ; NO - CHECK NEXT PRINTER CARD
07BA 89 94 0008 R               MOV     @PRINTER_BASE[SI],DX    ; YES - STORE PRINTER BASE ADDRESS
07BE 46                         INC     SI                      ; INCREMENT TO NEXT WORD
07BF 46                         INC     SI
07C0                    F17:
07C0 47                         INC     DI                      ; POINT TO NEXT BASE ADDRESS
07C1 47                         INC     DI
07C2 81 FF 09DC R               CMP     DI,OFFSET F4E           ; ALL POSSIBLE ADDRESSES CHECKED?
07C6 75 E3                      JNE     F16                     ; PRT_BASE
                        ;=============
                        ;-----  SETUP RS232
                        ;=============

07C8 BB 0000                    MOV     BX,0                    ; POINTER TO RS232 TABLE
07CB BA 03FA                    MOV     DX,3FAH                 ; CHECK IF RS232 CARD 1 ATTACHED ?
07CE EC                         IN      AL,DX                   ; READ INTERRUPT ID REGISTER
07CF A8 F8                      TEST    AL,0F8H
07D1 75 08                      JNZ     F18
07D3 C7 87 0000 R 03F8          MOV     @RS232_BASE[BX],3F8H    ; SETUP RS232 CARD #1 ADDRESS
07D9 43                         INC     BX
07DA 43                         INC     BX
07DB BA 02FA                F18:MOV     DX,2FAH                 ; CHECK IF RS232 CARD 2 ATTACHED
07DE EC                         IN      AL,DX                   ; READ INTERRUPT ID REGISTER
07DF A8 F8                      TEST    AL,0F8H
07E1 75 08                      JNZ     F19                     ; BASE_END
07E3 C7 87 0000 R 02F8          MOV     @RS232_BASE[BX],2F8H    ; SETUP RS232 CARD #2
07E9 43                         INC     BX
07EA 43                         INC     BX
                                ;--------------------------------------------------------------
                        ;-----  SET UP @EQUIP_FLAG TO INDICATE NUMBER OF PRINTERS AND RS232 CARDS
                                ;--------------------------------------------------------------
07EB                    F19:                                    ; BASE_END:
07EB 8B C6                      MOV     AX,SI                   ; SI HAS 2* NUMBER OF RS232
07ED B1 03                      MOV     CL,3                    ; SHIFT COUNT
07EF D2 C8                      ROR     AL,CL                   ; ROTATE RIGHT 3 POSITIONS
07F1 0A C3                      OR      AL,BL                   ; OR IN THE PRINTER COUNT
07F3 A2 0011 R                  MOV     BYTE PTR @EQUIP_FLAG+1,AL       ; STORE AS SECOND BYTE

                        ;-----  INSURE CMOS CLOCK HAS VALID HOURS.MINUTES.SECONDS

07F6 E8 0000 E                  CALL    SET_TOD                 ; INSURE CMOS CLOCK IS VALID

                        ;-----  ENABLE HARDWARE INTERRUPT IF MATH PROCESSOR (80287)

07F9 B0 40                      MOV     AL,40H                  ;       <><><><><><><><><><>
07FB E6 80                      OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  40 <><>

07FD BF 0067 R                  MOV     DI,OFFSET @IO_ROM_INIT  ; ADDRESS WORK STORAGE LOCATION
0800 33 C0                      XOR     AX,AX                   ; CLEAR WORK REGISTER  (AH)= 0 (NO 287)
0802 89 05                      MOV     WORD PTR [DI],AX        ; CLEAR THE WORK LOCATION
0804 DB E3                      FNINIT                          ; INITIALIZE THE 80287 WITH NO WAIT
0806 EB 00                      JMP     $+2                     ; DELAY
0808 D9 3D                      FNSTCW  WORD PTR [DI]           ; WRITE THE CURRENT 80287 CONTROL WORD
080A 60                         PUSHA                           ; TIME FOR 80287 TO RESPOND
080B 61                         POPA
080C 81 25 1F3F                 AND     WORD PTR [DI],01F3FH    ; CLEAR UNUSED 80287 BITS
0810 81 3D 033F                 CMP     WORD PTR [DI],0033FH    ; IS THE 80287 INSTALLED?
0814 75 13                      JNE     NO_287                  ; GO IF MATH PROCESSOR IS NOT INSTALLED
```

```
0816 9B DD 3D                   FSTSW   WORD PTR [DI]           ; STORE THE STATUS WORD (WITH WAIT)
0819 60                         PUSHA                           ; TIME FOR 80287 TO RESPOND
081A 61                         POPA
081B F7 05 B8BF                 TEST    WORD PTR [DI],0B8BFH    ; ALL BITS SHOULD BE OFF (OR ERROR)
081F 75 08                      JNZ     NO_287                  ; GO IF NOT INSTALLED

0821 E4 A1                      IN      AL,INTB01               ; GET THE SLAVE INTERRUPT MASK
0823 24 DF                      AND     AL,0DFH                 ; ENABLE 80287 INTERRUPTS
0825 B4 02                      MOV     AH,002H                 ; SET WORK REGISTER FOR 80287 FOUND
0827 E6 A1                      OUT     INTB01,AL
0829                    NO_287:
0829 A0 0010 R                  MOV     AL,BYTE PTR @EQUIP_FLAG ; GET LOW EQUIPMENT FLAG
082C 24 02                      AND     AL,002H                 ; STRIP OFF OTHER BITS
082E 3A C4                      CMP     AL,AH                   ; DOES CMOS MATCH HARDWARE ?
0830 74 08                      JE      OK_287                  ; SKIP IF EQUIPMENT FLAG CORRECT

0832 80 36 0010 R 02            XOR     BYTE PTR @EQUIP_FLAG,2H ; ELSE SET 80287 BIT TO CORRECT VALUE
0837 E8 0000 E                  CALL    CONFIG_BAD              ; AND SET THE CONFIGURATION ERROR FLAG
083A                    OK_287:
                        ;-----  SET KEYBOARD STATE FLAGS

083A C7 06 0017 R 0000          MOV     WORD PTR @KB_FLAG,0     ; RESET ALL KEYBOARD STATUS FLAGS

                        ;-----  ENABLE KEYBOARD/TIMER INTERRUPTS

0840 E4 21                      IN      AL,INTA01
0842 24 FC                      AND     AL,0FCH                 ; ENABLE TIMER AND KEYBOARD INTERRUPTS
0844 EB 00                      JMP     $+2                     ; I/O DELAY
0846 E6 21                      OUT     INTA01,AL
0848 C6 06 0015 R 00            MOV     @MFG_ERR_FLAG,0         ; CLEAR MFG ERROR FLAG

                        ;-----  READ KEYBOARD ID TO INITIALIZE KEYBOARD TYPE AND NUM LOCK STATE

084D C6 06 0096 R A0            MOV     @KB_FLAG_3,RD_ID+SET_NUM_LK     ; SET READ ID COMMAND FOR KBX
0852 B0 F2                      MOV     AL,KB_READ_ID           ; GET THIS SYSTEMS KEYBOARD ID REQUEST
0854 E8 0000 E                  CALL    SND_DATA                ; USE KEYBOARD TRANSMISSION ROUTINE
0857 B9 067A                    MOV     CX,T658                 ; SET DELAY COUNT TO 26 MILLISECONDS
085A E8 0000 E                  CALL    WAITF                   ; WAIT FOR READ ID RESPONSE (20 MS)
085D 80 26 0096 R 1F            AND     @KB_FLAG_3,NOT RD_ID+LC_AB+SET_NUM_LK   ; RESET READ ID COMMAND

                        ;-----  CHECK FOR SECOND FIXED DISK PRESENT BUT NOT DEFINED

0862 80 3E 0075 R 02            CMP     @HF_NUM,2               ; CHECK FOR TWO DRIVES DEFINED BY CMOS
0867 74 13                      JE      F15G                    ; SKIP TEST IF TWO DRIVES DEFINED

0869 B4 10                      MOV     AH,010H                 ; GET TEST DRIVE READY COMMAND
086B B2 81                      MOV     DL,081H                 ; POINT TO SECOND FIXED DISK
086D FE 06 0075 R               INC     @HF_NUM                 ; TELL BIOS IT HAS TWO DRIVES
0871 CD 13                      INT     13H                     ; CHECK READY THROUGH BIOS
0873 FE 0E 0075 R               DEC     @HF_NUM                 ; RESTORE CORRECT COUNT (RETAIN CY)
0877 72 03                      JC      F15G                    ; SKIP IF SECOND DRIVE NOT READY
                                                                ;       SECOND DRIVE NOT DEFINED
0879 E8 0000 E                  CALL    CONFIG_BAD              ; SET CONFIGURATION BAD
087C                    F15G:
                        ;---------------------------------------
                        ;   TEST FOR ANY ERRORS (BP NOT ZERO)   :
                        ;---------------------------------------

087C 0B ED                      OR      BP,BP                   ; CHECK (BP)= NON-ZERO  (ERROR HAPPENED)
087E 74 55                      JE      F15A_0                  ; SKIP PAUSE IF NO ERROR

0880 80 3E 0072 R 64            CMP     BYTE PTR @RESET_FLAG,64H; MFG RUN IN MODE?
0885 BA 0002                    MOV     DX,2                    ; 2 SHORT BEEP COUNT FOR ERROR(S)
0888 75 0E                      JNZ     ERR_WAIT                ; GO IF NOT

                        ;-----  MFG RUN IN MODE -> SET ERROR FLAG

088A C6 06 0015 R AA            MOV     @MFG_ERR_FLAG,0AAH      ; INDICATE ERROR
088F E4 64                      IN      AL,STATUS_PORT          ; CHECK KEY LOCK STATUS
0891 24 10                      AND     AL,KYBD_INH             ; IS THE KEYBOARD LOCKED
0893 75 40                      JNZ     F15A_0                  ; CONTINUE MFG MODE IF NOT LOCKED
                                                                ;  ELSE
0895 BA 0005                    MOV     DX,5                    ; 5 SHORT BEEPS FOR MFG SETUP ERROR
0898                    ERR_WAIT:
0898 E8 0000 E                  CALL    ERR_BEEP                ; BEEPS FOR ERROR(S)
089B B0 0E                      MOV     AL,CMOS_DIAG            ; ADDRESS CMOS
089D E8 0000 E                  CALL    CMOS_READ               ; GET THE DIAGNOSTIC STATUS BYTE
08A0 A8 20                      TEST    AL,BAD_CONFIG           ; CHECK FOR BAD HARDWARE CONFIGURATION
08A2 74 0C                      JZ      ERR_WKEY                ; SKIP IF NOT SET

08A4 F7 C5 8000                 TEST    BP,08000H               ; ELSE CHECK FOR E161/E162 POSTED
08A8 75 06                      JNZ     ERR_WKEY                ; SKIP IF DISPLAYED BEFORE NOW

08AA BE 0000 E                  MOV     SI,OFFSET E162          ; ELSE DISPLAY "OPTIONS NOT SET"
08AD E8 0000 E                  CALL    P_MSG                   ; WITH NON HALTING ROUTINE

                        ;-----  CHECK FOR "UNLOCK SYSTEM UNIT KEYLOCK" MESSAGE REQUIRED

08B0                    ERR_WKEY:
08B0 E4 64                      IN      AL,STATUS_PORT          ; CHECK IF RESUME MESSAGE NEEDED
08B2 24 10                      AND     AL,KYBD_INH             ; IS THE KEYBOARD LOCKED
08B4 75 06                      JNZ     ERR_WAIT2               ; SKIP LOCK MESSAGE IF NOT

08B6 BE 0000 E                  MOV     SI,OFFSET F3D1          ; ERROR MESSAGE FOR KEYBOARD LOCKED
08B9 E8 0000 E                  CALL    P_MSG

                        ;-----  DISPLAY '(RESUME = "F1" KEY)' FOR ERRORS

08BC                    ERR_WAIT2:
08BC BE 0000 E                  MOV     SI,OFFSET F3D           ; RESUME ERROR MESSAGE
08BF E8 0000 E                  CALL    P_MSG

                        ;-----  INITIALIZE PRINTER  (ALTERNATE DISPLAY DEVICE)

08C2 B4 01                      MOV     AH,1                    ;
08C4 2B D2                      SUB     DX,DX                   ; FIRST PRINTER
08C6 CD 17                      INT     17H                     ;
08C8                    ERR_WAIT1:
08C8 B0 3F                      MOV     AL,3FH                  ;       <><><><><><><><>
08CA E6 80                      OUT     MFG_PORT,AL             ;       <> CHECKPOINT  3F <>
08CC B4 00                      MOV     AH,00
08CE CD 16                      INT     16H                     ; WAIT FOR 'F1' KEY
08D0 80 FC 3B                   CMP     AH,3BH
08D3 75 F3                      JNE     ERR_WAIT1
```

```
                        F15A_0:
08D5 F6 06 0012 R 20            TEST    @MFG_TST,MFG_LOOP       ; MFG BURN IN MODE
08DA 75 03                      JNZ     F15A                    ; GO IF NOT
08DC E9 0000 E                  JMP     START_1                 ; GO LOOP POST
08DF 80 3E 0072 R 64    F15A:   CMP     BYTE PTR @RESET_FLAG,64H; MFG RUN IN?
08E4 74 06                      JZ      F15B                    ; BYPASS BEEP IF YES

08E6 BA 0001                    MOV     DX,1                    ; 1 SHORT BEEP (NO ERRORS)
08E9 E8 0000 E                  CALL    ERR_BEEP
                                ;===============
                        ;-----  SET TIME OF DAY
                                ;===============

08EC E8 0000 E          F15B:   CALL    SET_TOD

                        ;-----  CLEAR DISPLAY SCREEN

08EF 2A E4                      SUB     AH,AH                   ; CLEAR FLAGS
08F1 A0 0049 R                  MOV     AL,@CRT_MODE
08F4 CD 10                      INT     10H                     ; CLEAR SCREEN

                        ;-----  CLEAR DESCRIPTOR TABLES

08F6 B9 01F4            F20:    MOV     CX,0500                 ; CLEAR 1K
08F9 BF 00A0                    MOV     DI,SYS_IDT_LOC          ; POINT ES TO START OF DESCRIPTORS
08FC 2B C0                      SUB     AX,AX
08FE 8E C0                      MOV     ES,AX
0900 26: 89 05          F20_A:  MOV     ES:[DI],AX              ; CLEAR
0903 83 C7 02                   ADD     DI,2                    ; POINT TO NEXT LOCATION
0906 E2 F8                      LOOP    F20_A                   ; CONTINUE TILL DONE

                        ;-----  SET POST SYSTEM STACK

0908 B8 ---- R                  MOV     AX,AB50                 ; GET THE POST STACK SEGMENT
090B 8E D0                      MOV     SS,AX
090D BC 0400 R                  MOV     SP,OFFSET @TOS

                        ;-----  ENSURE THAT MASTER LEVEL 2 ENABLED

0910 E4 21                      IN      AL,INTA01               ; GET THE CURRENT MASK
0912 24 FB                      AND     AL,0FBH
0914 EB 00                      JMP     $+2                     ; I/O DELAY
0916 E6 21                      OUT     INTA01,AL

                        ;-----  TEST FOR MFG RUN-IN TEST

0918 80 3E 0072 R 64            CMP     BYTE PTR @RESET_FLAG,64H; IS THE THE MFG RUN-IN TEST?
091D 75 02                      JNZ     END_287                 ; GO IF NOT
091F EB 5C                      JMP     SHORT SHUT4             ; BOOT LOAD IF YES

                        ;-----  UNMASK SLAVE HARDWARE INTERRUPT 9 (LEVEL 71)
0921                    END_287:
0921 E4 A1                      IN      AL,INTB01               ; GET THE CURRENT MASK
0923 24 FD                      AND     AL,0FDH
0925 EB 00                      JMP     $+2                     ; I/O DELAY
0927 E6 A1                      OUT     INTB01,AL               ; SET NEW MASK

                        ;--------------------------------------------------------------------
                        ; TEST FOR SYSTEM CODE AT SEGMENT E000:0
                        ;   FIRST WORD = AA55H
                        ;   LAST BYTE = CHECKSUM
                        ;   ENTRY POINT = FIRST BYTE + 3
                        ; IF TEST IS SUCCESSFUL A CALL FAR TO THE ENTRY POINT IS EXECUTED
                        ;--------------------------------------------------------------------
0929 B0 41                      MOV     AL,41H                  ;       <><><><><><><><><>
092B E6 80                      OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  41 <><>

092D B0 8F                      MOV     AL,CMOS_SHUT_DOWN+NMI   ; INSURE NMI OFF AND CMOS AT DEFAULT
092F E6 70                      OUT     CMOS_PORT,AL

                        ENDIF:
0931 C6 06 0072 R 00            MOV     BYTE PTR @RESET_FLAG,0  ; CLEAR FLAG
0936 B8 E000                    MOV     AX,0E000H               ; SEGMENT OF SYSTEM CODE
0939 8E C0                      MOV     ES,AX
093B 2B FF                      SUB     DI,DI
093D 26: 8B 05                  MOV     AX,ES:[DI]              ; CHECK FOR AA55
0940 53                         PUSH    BX                      ; BUS SETTLE
0941 5B                         POP     BX
0942 3D AA55                    CMP     AX,0AA55H
0945 9C                         PUSHF                           ; SAVE FLAGS
0946 26: 89 05                  MOV     ES:[DI],AX              ; CLEAR POSSIBLE PARITY CHECK
0949 E4 61                      IN      AL,PORT_B
094B 0C 0C                      OR      AL,RAM_PAR_OFF          ; TOGGLE I/O-PARITY CHECK ENABLES
094D E6 61                      OUT     PORT_B,AL
094F 24 F3                      AND     AL,RAM_PAR_ON
0951 E6 61                      OUT     PORT_B,AL
0953 9D                         POPF                            ; RESTORE FLAGS
0954 75 27                      JNZ     SHUT4                   ; CONTINUE

                        ;-----  CHECKSUM SYSTEM CODE

0956 1E                         PUSH    DS
0957 06                         PUSH    ES
0958 1F                         POP     DS                      ; SET SEGMENT TO TEST
0959 2B DB                      SUB     BX,BX                   ; STARTING OFFSET
095B E8 0000 E                  CALL    ROM_CHECKSUM
095E 1F                         POP     DS                      ; RESTORE DATA SEGMENT
095F 75 1C                      JNZ     SHUT4                   ; GO IF CHECKSUM NOT OK

                        ;-----  ENABLE NMI AND I/O-MEMORY PARITY CHECKS

0961 B0 0F                      MOV     AL,CMOS_SHUT_DOWN       ; ENABLE NMI AND SET DEFAULT ADDRESS
0963 E6 70                      OUT     CMOS_PORT,AL

0965 E4 61                      IN      AL,PORT_B               ; ENABLE PARITY
0967 24 F3                      AND     AL,RAM_PAR_ON           ; ENABLE MEMORY PARITY CHECK / I/O CHECK
0969 E6 61                      OUT     PORT_B,AL

096B C7 06 0067 R 0003          MOV     @IO_ROM_INIT,0003H      ; SET THE OFFSET
0971 8C 06 0069 R               MOV     @IO_ROM_SEG,ES          ; SET THE SEGMENT

0975 B0 42                      MOV     AL,42H                  ;       <><><><><><><><><>
0977 E6 80                      OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  42 <><>
```

```
                        ;-----  EXIT TO SYSTEM CODE

                                CALL    DWORD PTR @IO_ROM_INIT  ; GO TO SYSTEM CODE
                                                                ; VIA CALL THROUGH DATA AREA LOCATION

                        ;-----  ENABLE NMI INTERRUPTS + ENTRY FROM SHUTDOWN WITH BOOT REQUEST

 097D B0 0F             SHUT4:  MOV     AL,CMOS_SHUT_DOWN       ; ENABLE NMI AND READ DEFAULT ADDRESS
 097F E8 0000 E                 CALL    CMOS_READ               ; OPEN STANDBY LATCH
 0982 E4 61                     IN      AL,PORT_B               ; ENABLE PARITY
 0984 24 F3                     AND     AL,RAM_PAR_ON           ; ENABLE MEMORY PARITY CHECK / I/O CHECK
 0986 E6 61                     OUT     PORT_B,AL

                                MOV     AL,43H                  ;       <><><><><><><><><>
 0988 B0 43                     OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  43 <><>
 098A E6 80                     STI                             ; ENABLE INTERRUPTS IF DISABLED
 098C FB

                                INT     19H                     ; GO TO BOOT LOADER
 098D CD 19

                                HLT
 098F F4


                        PADING  PROC    NEAR                    ;       INSERT PADDING
 0990                           MOV     CX,15                   ; GET BLANK CHARACTER COUNT
 0990 B9 000F           PAD1:
 0993                           MOV     AL,' '                  ; GET FILL SPACE
 0993 B0 20                     CALL    PRT_HEX                 ; WRITE A SPACE
 0995 E8 0000 E                 LOOP    PADT                    ; LOOP TILL INSERT DONE
 0998 E2 F9                     MOV     AL,'-'                  ; GET DASH CHARACTER
 099A B0 2D                     CALL    PRT_HEX                 ; WRITE TO DISPLAY
 099C E8 0000 E                 RET
 099F C3                PADING  ENDP
 09A0


 09A0                   PRT_OK  PROC    NEAR                    ;       PRINT "00000 KB OK"
 09A0 50                        PUSH    AX                      ; SAVE WORK REGISTER
 09A1 BB 000A                   MOV     BX,10                   ; SET DECIMAL CONVERT

                        ;-----  CONVERT AND SAVE

                                MOV     CX,5                    ; OF 5 NIBBLES XX,XXX KB
 09A4 B9 0005                   SUB     DI,DI                   ; DISPLAY REGEN BUFFER POSITION
 09A7 2B FF             PRT_DIV:
 09A9                           XOR     DX,DX
 09A9 33 D2                     DIV     BX
 09AB F7 F3                     OR      DL,30H                  ; MAKE INTO ASCII
 09AD 80 CA 30                  PUSH    DX                      ; SAVE
 09B0 52                        LOOP    PRT_DIV
 09B1 E2 F6

                        ;-----  DISPLAY LAST OK MEMORY

 09B3                           MOV     CX,6
 09B3 B9 0005           PRT_DEC:
 09B6                           POP     AX                      ; RECOVER A NUMBER
 09B6 58                        CALL    PROT_PRT_HEX
 09B7 E8 0000 E                 INC     DI                      ; POINT TO DISPLAY REGEN BUFFER
 09BA 47                        LOOP    PRT_DEC
 09BB E2 F9                     MOV     CX,OFFSET F3B_PAD-OFFSET F3B    ; LOAD MESSAGE LENGTH
 09BD B9 0007                   MOV     SI,OFFSET F3B           ; POINT TO PRINT ' KB OK','    ' MESSAGE
 09C0 BE 09CF R         PRT_LOOP:
 09C3                           MOV     AL,CS:[SI]
 09C3 2E: 8A 04                 INC     SI
 09C6 46                        CALL    PROT_PRT_HEX
 09C7 E8 0000 E                 INC     DI                      ; INCREMENT BUFF PTR
 09CA 47                        LOOP    PRT_LOOP
 09CB E2 F6                     POP     AX                      ; RECOVER WORK REGISTERS
 09CD 58                        RET
 09CE C3

 09CF 20 4B 42 20 4F 4B F3B     DB      ' KB OK'                ; OK MESSAGE
 09D5 20                F3B_OK  DB      ' '                     ; PAD A SPACE
 = 09D6                 F3B_PAD EQU     $
                        .LIST
 09D6                   PRT_OK  ENDP

                        ;-------------------------
                        ;       PRINTER TABLE   :
                        ;-------------------------

 09D6 03BC              F4      DW      03BCH                   ; ADDRESS OF MONOCHROME PARALLEL ADAPTER
 09D8 0378                      DW      0378H                   ; BASE ADDRESS STANDARD PARALLEL ADAPTER
 09DA 0278                      DW      0278H                   ; ADDRESS OF ALTERNATE PARALLEL ADAPTER
 09DC                   F4E     LABEL   WORD

 09DC                   POST2   ENDP
 09DC                   CODE    ENDS
                                END
```

```
                        PAGE 118,121
                        TITLE TEST3 ---- 06/10/85  POST EXCEPTION INTERRUPT TESTS
                        .286C
                        .LIST
                        ;--------------------------------------------------------
                        ; TEST.20                                               :
                        ;       ADDITIONAL PROTECTED (VIRTUAL MODE) TEST        :
                        ; DESCRIPTION                                           :
                        ;       THE PROCESSOR IS PUT IN PROTECTED MODE AND      :
                        ;       THE FOLLOWING FUNCTIONS ARE VERIFIED            :
                        ;                                                       :
                        ;       1. VERIFY PROTECTED MODE                        :
                        ;          THE MACHINE STATUS IS CHECK FOR VIRTUAL MODE :
                        ;       2. PROGRAMMED INTERRUPT TEST                    :
                        ;          AN PROGRAMMED INTERRUPT 32 IS ISSUED AND     :
                        ;          AND VERIFIED                                 :
                        ;       3. EXCEPTION INTERRUPT 13 TEST                  :
                        ;          A DESCRIPTOR SEGMENT LIMIT IS SET TO ZERO    :
                        ;          AND A WRITE TO THAT SEGMENT IS ATTEMPTED     :
                        ;          AN EXCEPTION 13 IS EXPECTED AND VERIFIED     :
                        ;       4. LDT/SDT LTR/STR TEST                         :
                        ;          LOAD LDT REGISTER AND VERIFY CORRECT         :
                        ;          LOAD TASK REGISTER AND VERIFY CORRECT        :
                        ;          THEY ARE VERIFIED VIA THE STORE INSTRUCTION  :
                        ;       5. THE CONTROL FLAGS OF THE 286 FOR DIRECTION   :
                        ;          ARE VERIFIED VIA THE STD AND CLD COMMANDS    :
                        ;          IN PROTECTED MODE                            :
                        ;       6. BOUND INSTRUCTION TEST (EXCEPTION INT 5)     :
                        ;          CREATE A SIGNED ARRAY INDEX WITHIN AND       :
                        ;          OUTSIDE THE LIMITS.  CHECK THAT NO EXC INT   :
                        ;          IF WITHIN LIMIT AND THAT AN EXC INT 5        :
                        ;          OCCURS IF OUTSIDE THE LIMITS.                :
                        ;       7. PUSH ALL POP ALL TEST                        :
                        ;          SET GENERAL PURPOSE REGISTERS TO DIFFERENT   :
                        ;          VALUES ISSUE A PUSH ALL, CLEAR THE REGISTERS :
                        ;          ISSUE A POP ALL AND VERIFY CORRECT.          :
                        ;       8. CHECK THE VERR/VERW INSTRUCTIONS             :
                        ;          THE ACCESS BYTE IS SET TO READ ONLY THEN TO  :
                        ;          A WRITE ONLY AND THE VERR/VERW INSTRUCTIONS   :
                        ;          ARE VERIFIED.                                :
                        ;       9. CAUSE AN INTERRUPT 13 VIA A WRITE TO A       :
                        ;          READ ONLY SEGMENT                            :
                        ;      10. VERIFY THE ARPL INSTRUCTION FUNCTIONS        :
                        ;          SET THE RPL FIELD OF A SELECTOR AND          :
                        ;          VERIFY THAT CURRENT SELECTOR RPL IS SET      :
                        ;          CORRECTLY.                                   :
                        ;      11. VERIFY THE LAR INSTRUCTION FUNCTIONS         :
                        ;      12. VERIFY THE LSL INSTRUCTION FUNCTIONS         :
                        ;      13. LOW MEG CHIP SELECT TEST                     :
                        ;--------------------------------------------------------

0000                    CODE    SEGMENT BYTE PUBLIC

                                PUBLIC  POST3

                                EXTRN   CMOS_WRITE:NEAR
                                EXTRN   DDS:NEAR
                                EXTRN   PROC_SHUTDOWN:NEAR
                                EXTRN   SYSINIT1:NEAR

                                ASSUME  CS:CODE
0000                    POST3   PROC
0000 E8 0000 E                  CALL    DDS                     ; SET DATA SEGMENT
0003 B0 F0                      MOV     AL,0F0H                 ;       <><><><><><><><>
0005 E6 80                      OUT     MFG_PORT,AL             ;       <> CHECKPOINT  F0 <>

                        ;-----  SET SHUTDOWN RETURN 7

0007 B8 078F                    MOV     AX,7*H+CMOS_SHUT_DOWN+NMI       ; ADDRESS FOR SHUTDOWN BYTE
000A E8 0000 E                  CALL    CMOS_WRITE              ; SET ERROR EXIT (DOUBLE EXCEPTION?)

                        ;-----  ENABLE PROTECTED MODE

000D BC 0000                    MOV     SP,POST_SS              ; SET STACK FOR SYSINIT1
0010 8E D4                      MOV     SS,SP
0012 BC 8000                    MOV     SP,POST_SP
0015 E8 0000 E                  CALL    SYSINIT1                ; GO ENABLE PROTECTED MODE

                        ;-----  SET TEMPORARY STACK

0018 B8 0008                    MOV     AX,GDT_PTR
001B 8E C0                      MOV     ES,AX
001D 8E D8                      MOV     DS,AX
001F 26: C7 06 005A 0000        MOV     ES:SS_TEMP.BASE_LO_WORD,0
0026 26: C6 06 005C 00          MOV     BYTE PTR ES:(SS_TEMP.BASE_HI_BYTE),0
002C BE 0058                    MOV     SI,SS_TEMP
002F 8E D6                      MOV     SS,SI
0031 BC FFFD                    MOV     SP,MAX_SEG_LEN-2

                        ;-----  VERIFY PROTECTED MODE

0034 0F 01 E0           +       SMSW    AX                      ; GET THE MACHINE STATUS WORD
                                DB      00FH,001H,0E0H
0037 A9 0001                    TEST    AX,VIRTUAL_ENABLE       ; ARE WE IN PROTECTED MODE
003A 75 03                      JNZ     T7_1
003C E9 02CD R                  JMP     ERROR_EXIT              ; ERROR IF NOT

003F B0 F1              T7_1:   MOV     AL,0F1H                 ;       <><><><><><><><>
0041 E6 80                      OUT     MFG_PORT,AL             ;       <> CHECKPOINT  F1 <>

                        ;-----  INTERRUPT TEST (PROGRAMMED INTERRUPT 32)

0043 B0 B0                      MOV     AL,0B0H                 ; SET EXCEPTION FLAG
0045 E6 8B                      OUT     DMA_PAGE+0AH,AL         ;  FOR INTERRUPT 10
0047 CD 20                      INT     32                      ; INTERRUPT
0049 2B C9                      SUB     CX,CX                   ; WAIT FOR INTERRUPT
004B E4 8B              LOOP1:  IN      AL,DMA_PAGE+0AH
004D 22 C0                      AND     AL,AL                   ; DID THE INTERRUPT OCCUR?
004F E0 FA                      LOOPNZ  LOOP1
0051 74 03                      JZ      T7_2
0053 E9 02CD R                  JMP     ERROR_EXIT              ; MISSING INTERRUPT

                        ;-----  CAUSE AN EXCEPTION INTERRUPT (GENERAL PROTECTION INTERRUPT 13D)
```

```
0056 B0 F2              T7_2:   MOV     AL,0F2H                 ;       <><><><><><><><><>
0058 E6 80                      OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  F2 <><>
005A B0 9D                      MOV     AL,9DH                  ; SET INTERRUPT 13 FLAG
005C E6 8B                      OUT     DMA_PAGE+0AH,AL         ; FOR THE INTERRUPT HANDLER

                        ;-----  MODIFY DESCRIPTOR TABLES
                        ;-----   SET TEMPORARY ES DESCRIPTOR TO SEGMENT LIMIT

005E C7 06 0048 0000            MOV     DS:ES_TEMP.SEG_LIMIT,0  ; SET SEGMENT TO 0

                        ;-----  CPL0, DATA ACCESS RIGHTS

0064 C6 06 004D 93              MOV     BYTE PTR DS:(ES_TEMP.DATA_ACC_RIGHTS),CPL0_DATA_ACCESS
0069 C6 06 004C 01              MOV     BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE),01  ; DO ALL TESTS ON 2ND 64K
006E C7 06 004A 0000            MOV     WORD PTR DS:(ES_TEMP.BASE_LO_WORD),0

0074 6A 48                      PUSH    BYTE PTR ES_TEMP        ; LOAD ES REGISTER
0076 07                         POP     ES

                        ;-----  CAUSE AN EXCEPTION 13 INTERRUPT

0077 2B FF                      SUB     DI,DI
0079 26: 8B 05                  MOV     AX,ES:[DI]              ; THIS SHOULD CAUSE AND EXCEPTION
007C 2B C9                      SUB     CX,CX                   ; WAIT FOR INTERRUPT
007E E4 8B              LOOP2:  IN      AL,DMA_PAGE+0AH
0080 22 C0                      AND     AL,AL                   ; DID THE INTERRUPT OCCUR?
0082 E0 FA                      LOOPNZ  LOOP2
0084 74 03                      JZ      T7_3                    ; CONTINUE IF INTERRUPT
0086 E9 02CD R                  JMP     ERROR_EXIT              ; MISSING INTERRUPT

                        ;-----------------------------------------
                        ;                                       :
                        ;       VERIFY 286 LDT/SDT LTR/STR      :
                        ;       INSTRUCTIONS                    :
                        ; DESCRIPTION                           :
                        ;       LOAD LDT  REGISTERS WITH A      :
                        ;       DESCRIPTOR AND VERIFY CORRECT   :
                        ;-----------------------------------------

                        ;-----  WRITE TO 286 LDT REGISTER
0089                    T7_3:
0089 B0 F3                      MOV     AL,0F3H                 ;       <><><><><><><><><>
008B E6 80                      OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  F3 <><>
008D BF 0078                    MOV     DI,POST_LDTR
                                LLDT    DI                      ; REGISTER FROM THIS AREA
0090 0F                 +       DB      00FH
0091                    + ??0000 LABEL  BYTE
0091 8B D7              +       MOV     DX,DI
0093                    + ??0001 LABEL  BYTE
0091                    +       ORG     OFFSET CS:??0000
0091 00                 +       DB      000H
0093                    +       ORG     OFFSET CS:??0001

                        ;-----  READ AND VERIFY 286 LDT SELECTOR

0093 2B C0                      SUB     AX,AX                   ; CLEAR AX
                                SLDT    AX                      ; GET THE LDT SELECTOR
0095 0F                 +       DB      00FH
0096                    + ??0002 LABEL  BYTE
0096 03 C0              +       ADD     AX,AX
0098                    + ??0003 LABEL  BYTE
0096                    +       ORG     OFFSET CS:??0002
0096 00                 +       DB      000H
0098                    +       ORG     OFFSET CS:??0003
0098 25 00F8                    AND     AX,0F8H                 ; STRIP TI/RPL
009B 3D 0078                    CMP     AX,POST_LDTR            ; CORRECT SELECTOR?
009E 75 1B                      JNZ     ERROR                   ; GO IF NOT

                        ;-----  WRITE TO 286 TR

00A0 BF 0068                    MOV     DI,POST_TR
                                LTR     DI                      ; REGISTER FROM THIS AREA
00A3 0F                 +       DB      00FH
00A4                    + ??0004 LABEL  BYTE
00A4 8B DF              +       MOV     BX,DI
00A6                    + ??0005 LABEL  BYTE
00A4                    +       ORG     OFFSET CS:??0004
00A4 00                 +       DB      000H
00A6                    +       ORG     OFFSET CS:??0005

                        ;-----  VERIFY 286 TR REGISTERS

00A6 2B C0                      SUB     AX,AX
                                STR     AX                      ; GET THE TR  REGISTER
00A8 0F                 +       DB      00FH
00A9                    + ??0006 LABEL  BYTE
00A9 8B C8              +       MOV     CX,AX
00AB                    + ??0007 LABEL  BYTE
00A9                    +       ORG     OFFSET CS:??0006
00A9 00                 +       DB      000H
00AB                    +       ORG     OFFSET CS:??0007
00AB 25 00F8                    AND     AX,0F8H
00AE 3D 0068                    CMP     AX,POST_TR              ; CORRECT SELECTOR?
00B1 75 08                      JNZ     ERROR

                        ;-----  TEST 286 CONTROL FLAGS

00B3 FD                         STD                             ; SET DIRECTION FLAG FOR DECREMENT
00B4 9C                         PUSHF                           ; GET THE FLAGS
00B5 58                         POP     AX
00B6 A9 0200                    TEST    AX,0200H                ; INTERRUPT FLAG SHOULD BE OFF
00B9 74 03                      JZ      T7_4                    ; CONTINUE IF OFF
00BB E9 02CD R          ERROR:  JMP     ERROR_EXIT              ; GO IF NOT
00BE                    T7_4:
00BE A9 0400                    TEST    AX,0400H                ; CHECK DIRECTION FLAG
00C1 75 03                      JNZ     T7_5
00C3 E9 02CD R                  JMP     ERROR_EXIT              ; GO IF NOT SET
00C6                    T7_5:
00C6 FC                         CLD                             ; CLEAR DIRECTION FLAG
00C7 9C                         PUSHF                           ; INSURE DIRECTION FLAG IS RESET
00C8 58                         POP     AX
00C9 A9 0400                    TEST    AX,0400H
00CC 74 03                      JZ      T7_6
00CE E9 02CD R                  JMP     ERROR_EXIT              ; GO IF NOT
```

SECTION 5

```
                        ;--------------------------------------------
                        ;       VERIFY 286 BOUND INSTRUCTION       :
                        ; DESCRIPTION                              :
                        ;       CREATE A SIGNED ARRAY INDEX        :
                        ;       WITHIN AND OUTSIDE THE LIMITS      :
                        ;       (EXPECT INT 5)                     :
                        ;--------------------------------------------

00D1 B0 F4              T7_6:   MOV     AL,0F4H                 ;       <><><><><><><><><>
00D3 E6 80                      OUT     MFG_PORT,AL             ;         <><> CHECKPOINT  F4 <><>
00D5 6A 48                      PUSH    BYTE PTR ES_TEMP        ; LOAD ES REGISTER
00D7 07                         POP     ES

                        ;-----  CHECK BOUND FUNCTIONS CORRECTLY

00D8 2B FF                      SUB     DI,DI                   ; POINT BEGINNING OF THE BLOCK
00DA 26: C7 05 0000             MOV     WORD PTR ES:[DI],0      ; SET FIRST WORD TO ZERO
00DF 26: C7 45 02 7FFF          MOV     WORD PTR ES:[DI+2],07FFFH ; SET SECOND TO 07FFFH
00E5 B0 95                      MOV     AL,095H                 ; SET INTERRUPT 5 FLAG
00E7 E6 8B                      OUT     DMA_PAGE+0AH,AL
00E9 B8 1000                    MOV     AX,1000H                ; SET AX WITHIN BOUNDS
00EC 26: 62 05                  BOUND   AX,DWORD PTR ES:[DI]    ; USE THE ES SEGMENT POINTER
00EF 2B C9                      SUB     CX,CX                   ; WAIT FOR POSSIBLE INTERRUPT
00F1 E2 FE              LOOPA:  LOOP    LOOPA
00F3 E4 8B                      IN      AL,DMA_PAGE+0AH         ; GET THE RESULTS
00F5 3C 00                      CMP     AL,0                    ; DID AN INTERRUPT OCCUR?
00F7 75 03                      JNZ     T7_7                    ; CONTINUE IF NOT
00F9 E9 02CD R                  JMP     ERROR_EXIT              ; GO IF YES

                        ;-----  CHECK LOW BOUND WORD CAUSES INTERRUPT 5
00FC                    T7_7:
00FC 2B FF                      SUB     DI,DI                   ; POINT BEGINNING OF THE BLOCK
00FE 26: C7 05 3FF0             MOV     WORD PTR ES:[DI],03FF0H ; SET FIRST WORD TO 03FF0H
0103 B8 1000                    MOV     AX,1000H                ; SET AX OUT OF BOUNDS
0106 26: 62 05                  BOUND   AX,DWORD PTR ES:[DI]
0109 2B C9                      SUB     CX,CX                   ; WAIT FOR POSSIBLE INTERRUPT
010B                    LOOPB:
010B E4 8B                      IN      AL,DMA_PAGE+0AH         ; GET THE RESULTS
010D 3C 00                      CMP     AL,0H                   ; DID AN INTERRUPT OCCUR?
010F E0 FA                      LOOPNZ  LOOPB                   ; TRY AGAIN
0111 74 03                      JZ      T7_8                    ; CONTINUE IF INTERRUPT
0113 E9 02CD R                  JMP     ERROR_EXIT              ; GO IF NO INTERRUPT

                        ;-----  CHECK HIGH BOUND WORD CAUSES INTERRUPT 5

0116 B0 95              T7_8:   MOV     AL,95H                  ; SET FLAG FOR INTERRUPT
0118 E6 8B                      OUT     DMA_PAGE+0AH,AL

011A 2B FF                      SUB     DI,DI                   ; POINT BEGINNING OF THE BLOCK
011C 26: C7 05 0000             MOV     WORD PTR ES:[DI],0      ; SET FIRST WORD TO 0
0121 26: C7 45 02 0FFF          MOV     WORD PTR ES:[DI+2],0FFFH; SET SECOND TO 0FFFH
0127 B8 1000                    MOV     AX,1000H                ; SET AX OUT OF BOUNDS
012A 26: 62 05                  BOUND   AX,DWORD PTR ES:[DI]
012D 2B C9                      SUB     CX,CX                   ; WAIT FOR POSSIBLE INTERRUPT
012F                    LOOPC:
012F E4 8B                      IN      AL,DMA_PAGE+0AH         ; GET THE RESULTS
0131 3C 00                      CMP     AL,0H                   ; DID AN INTERRUPT OCCUR?
0133 E0 FA                      LOOPNZ  LOOPC                   ; TRY AGAIN
0135 74 03                      JZ      T7_9                    ; CONTINUE IF INTERRUPT
0137 E9 02CD R                  JMP     ERROR_EXIT              ; GO IF NO INTERRUPT

                        ;--------------------------------------------
                        ;       VERIFY PUSH ALL AND POP ALL INSTRUCTIONS:
                        ; DESCRIPTION                              :
                        ;       SET REGISTERS TO A KNOWN VALUE AND :
                        ;       PUSH ALL.  RESET THE REGISTERS, POP ALL :
                        ;       AND VERIFY                         :
                        ;--------------------------------------------

013A B0 F5              T7_9:   MOV     AL,0F5H                 ;       <><><><><><><><><>
013C E6 80                      OUT     MFG_PORT,AL             ;         <><> CHECKPOINT  F5 <><>
013E B8 0001                    MOV     AX,01                   ; SET AX=1
0141 8B D8                      MOV     BX,AX                   ; SET BX=2
0143 43                         INC     BX
0144 8B CB                      MOV     CX,BX                   ; SET CX=3
0146 41                         INC     CX
0147 8B D1                      MOV     DX,CX
0149 42                         INC     DX                      ; SET DX=4
014A 8B FA                      MOV     DI,DX
014C 47                         INC     DI                      ; SET DI=5
014D 8B F7                      MOV     SI,DI
014F 46                         INC     SI                      ; SET SI=6
0150 55                         PUSH    BP                      ; SAVE THE (BP) ERROR FLAG REGISTER
0151 8B EE                      MOV     BP,SI                   ; SET BP=7
0153 45                         INC     BP
0154 60                         PUSHA                           ; ISSUE THE PUSH ALL COMMAND
0155 2B C0                      SUB     AX,AX                   ; CLEAR ALL REGISTERS
0157 8B D8                      MOV     BX,AX
0159 8B C8                      MOV     CX,AX
015B 8B D0                      MOV     DX,AX
015D 8B F8                      MOV     DI,AX
015F 8B F0                      MOV     SI,AX
0161 8B E8                      MOV     BP,AX
0163 61                         POPA                            ; GET THE REGISTERS BACK
0164 83 FD 07                   CMP     BP,07                   ; BP SHOULD BE 7
0167 5D                         POP     BP                      ; RESTORE (BP) ERROR FLAG REGISTER
0168 75 1E                      JNZ     ERROR_EXIT1             ; GO IF NOT
016A 3D 0001                    CMP     AX,01                   ; AX SHOULD BE 1
016D 75 19                      JNZ     ERROR_EXIT1             ; GO IF NOT
016F 83 FB 02                   CMP     BX,02                   ; BX SHOULD BE 2
0172 75 14                      JNZ     ERROR_EXIT1             ; GO IF NOT
0174 83 F9 03                   CMP     CX,03                   ; CX SHOULD BE 3
0177 75 0F                      JNZ     ERROR_EXIT1             ; GO IF NOT
0179 83 FA 04                   CMP     DX,04                   ; DX SHOULD BE 4
017C 75 0A                      JNZ     ERROR_EXIT1             ; GO IF NOT
017E 83 FF 05                   CMP     DI,05                   ; DI SHOULD BE 5
0181 75 05                      JNZ     ERROR_EXIT1             ; GO IF NOT
0183 83 FE 06                   CMP     SI,06                   ; SI SHOULD BE 6
0186 74 03                      JZ      T7_10                   ; CONTINUE IF IT IS

                        ;-----  ERROR EXIT

0188                    ERROR_EXIT1:
0188 E9 02CD R                  JMP     ERROR_EXIT
```

```
                        ;--------------------------------------------------
                        ;       VERIFY ACCESS RIGHTS FUNCTION CORRECTLY :
                        ; DESCRIPTION                                   :
                        ;       SET ACCESS RIGHTS OF DESCRIPTOR TO      :
                        ;       READ ONLY.  VERIFY THE VERW/VERR INSTR  :
                        ;       ACCESS A READ ONLY WITH A WRITE AND     :
                        ;       VERIFY AN EXCEPTION INTERRUPT 13        :
                        ;--------------------------------------------------

018B B0 F6              T7_10:  MOV     AL,0F6H                 ;       <><><><><><><><><>
018D E6 80                      OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  F6 <><>
018F C7 06 0048 FFFF            MOV     DS:ES_TEMP.SEG_LIMIT,MAX_SEG_LEN        ; SET SEGMENT TO OFFFFH
0195 C6 06 004C 00              MOV     BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE),0    ; SET THE ADDRESS
019A C7 06 004A F000            MOV     DS:ES_TEMP.BASE_LO_WORD,0F000H
01A0 B8 0048                    MOV     AX,ES_TEMP              ; LOAD ES REGISTER
01A3 8E C0                      MOV     ES,AX                   ; THIS SEGMENT SHOULD BE WRITEABLE

                        ;-----  INSURE ACCESS RIGHTS MAY BE WRITTEN

                                SEGOV   DS                      ; SET SEGMENT OVERRIDE TO START OF TABLE
01A5 3E                 +       DB      03EH
                                VERW    AX                      ; CHECK THE ACCESS RIGHTS OF ES_TEMP
01A6 0F                 +       DB      00FH
01A7                    + ??0009 LABEL  BYTE
01A7 8B E8              +       MOV     BP,AX
01A9                    + ??000A LABEL  BYTE
01A7                    +       ORG     OFFSET CS:??0009
01A7 00                 +       DB      000H
01A9                    +       ORG     OFFSET CS:??000A
01A9 75 DD                      JNZ     ERROR_EXIT1             ; ERROR IF SEGMENT CAN NOT WRITE

                        ;-----  SET ACCESS RIGHTS TO READ ONLY

01AB C6 06 004D 91              MOV     BYTE PTR DS:(ES_TEMP.DATA_ACC_RIGHTS),91H
01B0 B8 0048                    MOV     AX,ES_TEMP              ; LOAD ES REGISTER
01B3 8E C0                      MOV     ES,AX
                                SEGOV   DS                      ; SET SEGMENT OVERRIDE TO START OF TABLE
01B5 3E                 +       DB      03EH
                                VERW    AX                      ; CHECK THE ACCESS RIGHTS OF ES_TEMP
01B6 0F                 +       DB      00FH
01B7                    + ??000C LABEL  BYTE
01B7 8B E8              +       MOV     BP,AX
01B9                    + ??000D LABEL  BYTE
01B7                    +       ORG     OFFSET CS:??000C
01B7 00                 +       DB      000H
01B9                    +       ORG     OFFSET CS:??000D
01B9 74 CD                      JZ      ERROR_EXIT1             ; ERROR IF SEGMENT IS WRITEABLE

01BB B8 0048                    MOV     AX,ES_TEMP              ; INSURE THAT SEGMENT IS READABLE
                                SEGOV   DS
01BE 3E                 +       DB      03EH
                                VERR    AX
01BF 0F                 +       DB      00FH
01C0                    + ??000F LABEL  BYTE
01C0 8B E0              +       MOV     SP,AX
01C2                    + ??0010 LABEL  BYTE
01C0                    +       ORG     OFFSET CS:??000F
01C0 00                 +       DB      000H
01C2                    +       ORG     OFFSET CS:??0010
01C2 75 C4                      JNZ     ERROR_EXIT1             ; GO IF SEGMENT NOT READABLE

                        ;-----  CAUSE AN EXCEPTION 13 INTERRUPT

01C4 B0 9D                      MOV     AL,09DH                 ; SET EXCEPTION FLAG
01C6 E6 8B                      OUT     DMA_PAGE+0AH,AL         ;   FOR INTERRUPT 13
01C8 2B F6                      SUB     SI,SI
01CA 26: C6 04 00               MOV     BYTE PTR ES:[SI],00     ; WRITE A BYTE THAT SHOULD
01CE 2B C9                      SUB     CX,CX                   ; WAIT FOR INTERRUPT
01D0 E4 8B              LOOPD:  IN      AL,DMA_PAGE+0AH
01D2 22 C0                      AND     AL,AL                   ; DID THE INTERRUPT OCCUR?
01D4 E0 FA                      LOOPNZ  LOOPD
01D6 75 B0                      JNZ     ERROR_EXIT1             ; MISSING INTERRUPT

                        ;-----  RESTORE THE ACCESS RIGHTS BYTE

01D8 C6 06 004D 93              MOV     BYTE PTR DS:(ES_TEMP.DATA_ACC_RIGHTS),CPL0_DATA_ACCESS

                        ;--------------------------------------------------
                        ;       VERIFY ADJUST RPL FIELD OF SELECTOR      :
                        ;       INSTRUCTION (ARPL) FUNCTIONS             :
                        ; DESCRIPTION                                    :
                        ;       SET THE RPL FIELD OF A SELECTOR          :
                        ;       AND VERIFY THAT THE ZERO FLAG IS SET     :
                        ;       CORRECTLY AND THAT THE SELECTOR RPL      :
                        ;       FIELD IS SET CORRECTLY                   :
                        ;--------------------------------------------------

01DD B0 F7                      MOV     AL,0F7H                 ;       <><><><><><><><><>
01DF E6 80                      OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  F7 <><>
01E1 B8 0048                    MOV     AX,ES_TEMP              ; PUT A SELECTOR IN AX
01E4 BB 0060                    MOV     BX,DS_TEMP              ; PUT A SELECTOR IN BX
01E7 0D 0003                    OR      AX,03H                  ; MAKE ACCESS OF AX < BX

                        ;-----  NOTE BX = FIRST OPERAND  AX = SECOND OPERAND

                                ARPL    AX,BX                   ; ISSUE THE RPL COMMAND
01EA                    + ??0011 LABEL  BYTE
01EA 8B C3              +       MOV     AX,BX
01EC                    + ??0012 LABEL  BYTE
01EA                    +       ORG     OFFSET CS:??0011
01EA 63                 +       DB      063H
01EC                    +       ORG     OFFSET CS:??0012
01EC 75 9A                      JNZ     ERROR_EXIT1             ; GO IF RPL WAS NOT CHANGED
01EE 80 E3 03                   AND     BL,03H                  ; STRIP UNWANTED BITS
01F1 80 FB 03                   CMP     BL,03H                  ; AS EXPECTED?
01F4 75 92                      JNZ     ERROR_EXIT1             ; GO IF NOT

                        ;-----  CHECK THAT ACCESS RIGHTS DO NOT CHANGE

01F6 BB 0060                    MOV     BX,DS_TEMP              ; PUT A SELECTOR IN BX
01F9 B8 0048                    MOV     AX,ES_TEMP              ; PUT A SELECTOR IN AX
01FC 80 CB 03                   OR      BL,03H                  ; MAKE ACCESS OF BX < AX

                        ;-----  NOTE BX = FIRST OPERAND  AX = SECOND OPERAND
```

```
                                                ARPL    AX,BX                   ; ISSUE THE RPL COMMAND
0201                                    + ??0013 LABEL   BYTE
01FF 8B C3                              +       MOV     AX,BX
0201                                    + ??0014 LABEL   BYTE
01FF                                    +       ORG     OFFSET CS:??0013
01FF 63                                 +       DB      063H
0201                                    +       ORG     OFFSET CS:??0014
0201 74 85                                      JZ      ERROR_EXIT1             ; GO IF RPL WAS NOT CHANGED
0203 80 E3 03                                   AND     BL,03H                  ; STRIP UNWANTED BITS
0206 80 FB 03                                   CMP     BL,03H                  ; AS EXPECTED?
0209 75 2F                                      JNZ     ERROR_EXIT2             ; GO IF NOT

                                        ;-----  VERIFY THE LAR (LOAD ACCESS RIGHTS) INSTRUCTION

020B B0 F8                                      MOV     AL,0F8H                 ;       <><><><><><><><>
020D E6 80                                      OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  F8 <><>

                                        ;-----  SET THE DESCRIPTOR TO LEVEL 3

020F C6 06 004D F3                              MOV     BYTE PTR DS:[ES_TEMP.DATA_ACC_RIGHTS],CPL3_DATA_ACCESS
0214 8B 0048                                    MOV     BX,ES_TEMP
0217 2B C0                                      SUB     AX,AX                   ; CLEAR AX

                                        ;-----  GET THE CURRENT DESCRIPTORS ACCESS RIGHTS

                                                LAR     AX,BX                   ; ISSUE THE LAR COMMAND
0219 0F                                 +       DB      00FH
021A                                    + ??0015 LABEL   BYTE
021A 8B C3                              +       MOV     AX,BX
021C                                    + ??0016 LABEL   BYTE
021A                                    +       ORG     OFFSET CS:??0015
021A 02                                 +       DB      002H
021C                                    +       ORG     OFFSET CS:??0016

                                        ;-----  INSURE THE DESCRIPTOR WAS VISIBLE

021C 75 1C                                      JNZ     ERROR_EXIT2             ; GO IF LAR WAS NOT CHANGED

                                        ;-----  THE DESCRIPTORS ACCESS RIGHTS MUST BE 3

021E 80 FC F3                                   CMP     AH,CPL3_DATA_ACCESS     ; AS EXPECTED?
0221 75 17                                      JNZ     ERROR_EXIT2             ; GO IF NOT

                                        ;-----  VERIFY THE LSL (LOAD SEGMENT LIMITS) INSTRUCTION

0223 B0 F9                                      MOV     AL,0F9H                 ;       <><><><><><><><>
0225 E6 80                                      OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  F9 <><>
0227 C7 06 0048 AAAA                            MOV     DS:ES_TEMP.SEG_LIMIT,0AAAAH     ; SET SEGMENT LIMIT TO 0AAAAH

022D C6 06 004D 93                              MOV     BYTE PTR DS:[ES_TEMP.DATA_ACC_RIGHTS],CPL0_DATA_ACCESS
0232 8B 0048                                    MOV     AX,ES_TEMP              ; LOAD ES REGISTER
                                                LSL     BX,AX                   ; GET THE DESCRIPTOR SEGMENT LIMIT
0235 0F                                 +       DB      00FH
0236                                    + ??0017 LABEL   BYTE
0236 8B D8                              +       MOV     BX,AX
0238                                    + ??0018 LABEL   BYTE
0234                                    +       ORG     OFFSET CS:??0017
0236 03                                 +       DB      003H
0238                                    +       ORG     OFFSET CS:??0018
0238 74 03                                      JZ      R07                     ; GO IF OK

023A                                    ERROR_EXIT2:

023A E9 02CD R                                  JMP     ERROR_EXIT              ; GO IF NOT SUCCESSFUL

023D 81 FB AAAA                 R07:    CMP     BX,0AAAAH               ; INSURE CORRECT SEGMENT LIMIT
0241 C7 06 0048 5555                    MOV     DS:ES_TEMP.SEG_LIMIT,05555H     ;SET THE SEGMENT LIMIT TO 05555H
0247 8B 0048                            MOV     AX,ES_TEMP
                                        LSL     BX,AX                   ; GET THE DESCRIPTOR SEGMENT LIMIT
024A 0F                                 +       DB      00FH
024B                                    + ??0019 LABEL   BYTE
024B 8B D8                              +       MOV     BX,AX
024D                                    + ??001A LABEL   BYTE
024B                                    +       ORG     OFFSET CS:??0019
024B 03                                 +       DB      003H
024D                                    +       ORG     OFFSET CS:??001A
024D 75 EB                                      JNZ     ERROR_EXIT2             ; GO IF NOT SUCCESSFUL

024F 81 FB 5555                                 CMP     BX,05555H               ; INSURE CORRECT SEGMENT LIMIT
0253 75 E5                                      JNZ     ERROR_EXIT2             ; GO IF NOT

                                        ;-------------------------------------------------------
                                        ; LOW MEG CHIP SELECT TEST                             :
                                        ;  TEST THAT A WRITE TO ADDRESS 1B0000 DOES NOT WRITE   :
                                        ;  TO B000:0, OR 1B8000 DOES NOT WRITE TO B800:0        :
                                        ;-------------------------------------------------------

0255 B0 FA                                      MOV     AL,0FAH                 ;       <><><><><><><><>
0257 E6 80                                      OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  FA <><>
0259 6A 08                                      PUSH    BYTE PTR GDT_PTR        ; MODIFY THE DESCRIPTOR TABLE
025B 1F                                         POP     DS

                                        ;-----  SET TEMPORARY ES DESCRIPTOR 64K SEGMENT LIMIT/CPL0 DATA ACCESS

025C C7 06 0048 FFFF                            MOV     DS:ES_TEMP.SEG_LIMIT,MAX_SEG_LEN
0262 C6 06 004D 93                              MOV     BYTE PTR DS:[ES_TEMP.DATA_ACC_RIGHTS],CPL0_DATA_ACCESS

                                        ;-----  START WITH SEGMENT 1B0000

0267 C6 06 004C 1B                              MOV     BYTE PTR DS:[ES_TEMP.BASE_HI_BYTE],1BH
026C C7 06 004A 0000                            MOV     DS:ES_TEMP.BASE_LO_WORD,0
0272 6A 48                                      PUSH    BYTE PTR  ES_TEMP       ; LOAD ES REGISTER
0274 07                                         POP     ES
0275 2B FF                                      SUB     DI,DI                   ; POINT TO FIRST LOCATION
0277 26: C7 05 AA55                             MOV     WORD PTR ES:[DI],0AA55H ; WRITE A TEST PATTERN

                                        ;-----  DO FOR SEGMENT 1B8000

027C C7 06 004A 8000                            MOV     DS:ES_TEMP.BASE_LO_WORD,8000H
0282 6A 48                                      PUSH    BYTE PTR ES_TEMP        ; LOAD ES REGISTER
0284 07                                         POP     ES
0285 26: C7 05 AA55                             MOV     WORD PTR ES:[DI],0AA55H ; WRITE A TEST PATTERN
```

```
                                ;-----  DO FOR SEGMENT 1A0000

                                        MOV     BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE),1AH
028A C6 06 004C 1A              MOV     DS:ES_TEMP.BASE_LO_WORD,0
028F C7 06 004A 0000            PUSH    BYTE PTR ES_TEMP        ; LOAD ES REGISTER
0295 6A 48                      POP     ES
0297 07                         MOV     WORD PTR ES:[DI],0AA55H ; WRITE A TEST PATTERN
0298 26: C7 05 AA55

                                ;-----  B/W VIDEO CARD

                                        PUSH    BYTE PTR  C_BWCRT_PTR
029D 6A 20                      POP     DS                      ; SET DS TO B/W DISPLAY REGEN BUFFER
029F 1F                         MOV     AX,DS:[DI]              ; GET THE WORD FROM B/W VIDEO
02A0 8B 05

                                ;-----  COMPATIBLE COLOR

                                        PUSH    BYTE PTR C_CCRT_PTR     ; SET DS TO COMPATIBLE COLOR MEMORY
02A2 6A 28                      POP     DS
02A4 1F                         MOV     BX,DS:[DI]              ; GET THE WORD FROM COLOR MEMORY
02A5 8B 1D

                                ;-----  EGA COLOR

                                        PUSH    BYTE PTR E_CCRT_PTR             ; EGA COLOR CRT POINTER LOW 64K
02A7 6A 30                      POP     DS
02A9 1F                         MOV     CX,DS:[DI]
02AA 8B 0D

                                ;-----  TEST FOR ERROR

                                        PUSH    AX                      ; SAVE RESULTS
02AC 50                         MOV     AL,35H                  ;       <><><><><><><><><><><>
02AD B0 35                      OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  35 <><>
02AF E6 80                      POP     AX
02B1 58                         CMP     AX,0AA55H
02B2 3D AA55                    JZ      ERROR_EXIT
02B5 74 16                      CMP     BX,0AA55H
02B7 81 FB AA55                 JZ      ERROR_EXIT
02BB 74 10                      CMP     CX,0AA55H
02BD 81 F9 AA55                 JZ      ERROR_EXIT
02C1 74 0A                      MOV     AL,34H                  ; RESTORE CHECKPOINT
02C3 B0 34                      OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  34 <><>
02C5 E6 80

                                ;-----  SHUTDOWN

02C7                            NORMAL_EXIT:
02C7 B8 068F                            MOV     AX,6*H+CMOS_SHUT_DOWN+NMI       ; ADDRESS FOR SHUTDOWN BYTE
02CA E8 0000 E                          CALL    CMOS_WRITE                      ; SET GOOD ENDING
02CD                            ERROR_EXIT:
02CD E9 0000 E                          JMP     PROC_SHUTDOWN

02D0                            POST3   ENDP

02D0                            CODE    ENDS
                                        END
```

```
                        PAGE 118,121
                        TITLE TEST4 ---- 06/10/85  POST AND BIOS UTILITY ROUTINES
                        .286C
                        .LIST
0000                    CODE    SEGMENT BYTE PUBLIC

                                PUBLIC  BEEP
                                PUBLIC  BLINK_INT
                                PUBLIC  CMOS_READ
                                PUBLIC  CMOS_WRITE
                                PUBLIC  CONFIG_BAD
                                PUBLIC  D11
                                PUBLIC  D0S
                                PUBLIC  DUMMY_RETURN_1
                                PUBLIC  ERR_BEEP
                                PUBLIC  E_MSG
                                PUBLIC  INT_287
                                PUBLIC  KBD_RESET
                                PUBLIC  POST4
                                PUBLIC  PROT_PRT_HEX
                                PUBLIC  PROC_SHUTDOWN
                                PUBLIC  PRT_HEX
                                PUBLIC  PRT_SEG
                                PUBLIC  P_MSG
                                PUBLIC  RE_DIRECT
                                PUBLIC  ROM_CHECK
                                PUBLIC  ROM_CHECKSUM
                                PUBLIC  SET_TOD
                                PUBLIC  WAITF
                                PUBLIC  XPC_BYTE

                                EXTRN   E163:NEAR
                                EXTRN   OBF_42:NEAR
                                EXTRN   ROM_ERR:NEAR
                                EXTRN   XMIT_8042:NEAR

                                ASSUME  CS:CODE,DS:DATA
0000                    POST4:
                        ;--- CMOS_READ ------------------------------------------------------------
                        ;                                                                          :
                        ;       READ BYTE FROM CMOS SYSTEM CLOCK CONFIGURATION TABLE               :
                        ;                                                                          :
                        ; INPUT: (AL)=  CMOS TABLE ADDRESS TO BE READ                              :
                        ;               BIT    7 = 0 FOR NMI ENABLED AND 1 FOR NMI DISABLED ON EXIT :
                        ;               BITS 6-0 = ADDRESS OF TABLE LOCATION TO READ               :
                        ;                                                                          :
                        ; OUTPUT: (AL)  VALUE AT LOCATION (AL) MOVED INTO (AL).  IF BIT 7 OF (AL) WAS :
                        ;               ON THEN NMI LEFT DISABLED.  DURING THE CMOS READ BOTH NMI AND :
                        ;               NORMAL INTERRUPTS ARE DISABLED TO PROTECT CMOS DATA INTEGRITY. :
                        ;               THE CMOS ADDRESS REGISTER IS POINTED TO A DEFAULT VALUE AND :
                        ;               THE INTERRUPT FLAG RESTORED TO THE ENTRY STATE ON RETURN.   :
                        ;               ONLY THE (AL) REGISTER AND THE NMI STATE IS CHANGED.        :
                        ;--------------------------------------------------------------------------

0000                    CMOS_READ       PROC    NEAR                    ;       READ LOCATION (AL) INTO (AL)
0000 9C                         PUSHF                                   ; SAVE INTERRUPT ENABLE STATUS AND FLAGS
0001 D0 C0                      ROL     AL,1                            ; MOVE NMI BIT TO LOW POSITION
0003 F9                         STC                                     ; FORCE NMI BIT ON IN CARRY FLAG
0004 D0 D8                      RCR     AL,1                            ; HIGH BIT ON TO DISABLE NMI - OLD IN CY
0006 FA                         CLI                                     ; DISABLE INTERRUPTS
0007 E6 70                      OUT     CMOS_PORT,AL                    ; ADDRESS LOCATION AND DISABLE NMI
0009 90                         NOP                                     ; I/O DELAY
000A E4 71                      IN      AL,CMOS_DATA                    ; READ THE REQUESTED CMOS LOCATION
000C 50                         PUSH    AX                              ; SAVE (AH) REGISTER VALUE AND CMOS BYTE
000D B0 1E                      MOV     AL,CMOS_SHUT_DOWN*2             ; GET ADDRESS OF DEFAULT LOCATION
000F D0 D8                      RCR     AL,1                            ; PUT ORIGINAL NMI MASK BIT INTO ADDRESS
0011 E6 70                      OUT     CMOS_PORT,AL                    ; SET DEFAULT TO READ ONLY REGISTER
0013 90                         NOP                                     ; I/O DELAY
0014 E4 71                      IN      AL,CMOS_DATA                    ; OPEN STANDBY LATCH
0016 58                         POP     AX                              ; RESTORE (AH) AND (AL)= CMOS BYTE
0017 0E                         PUSH    CS                              ; *PLACE CODE SEGMENT IN STACK AND
0018 E8 0039 R                  CALL    CMOS_POPF                       ; *HANDLE POPF FOR B- LEVEL 80286
001B C3                         RET                                     ; RETURN WITH FLAGS RESTORED

001C                    CMOS_READ       ENDP

                        ;--- CMOS_WRITE -----------------------------------------------------------
                        ;                                                                          :
                        ;       WRITE BYTE TO CMOS SYSTEM CLOCK CONFIGURATION TABLE                :
                        ;                                                                          :
                        ; INPUT: (AL)=  CMOS TABLE ADDRESS TO BE WRITTEN TO                        :
                        ;               BIT    7 = 0 FOR NMI ENABLED AND 1 FOR NMI DISABLED ON EXIT :
                        ;               BITS 6-0 = ADDRESS OF TABLE LOCATION TO WRITE              :
                        ;       (AH)=   NEW VALUE TO BE PLACED IN THE ADDRESSED TABLE LOCATION     :
                        ;                                                                          :
                        ; OUTPUT:       VALUE IN (AH) PLACED IN LOCATION (AL) WITH NMI LEFT DISABLED :
                        ;               IF BIT 7 OF (AL) IS ON.  DURING THE CMOS UPDATE BOTH NMI AND :
                        ;               NORMAL INTERRUPTS ARE DISABLED TO PROTECT CMOS DATA INTEGRITY. :
                        ;               THE CMOS ADDRESS REGISTER IS POINTED TO A DEFAULT VALUE AND :
                        ;               THE INTERRUPT FLAG RESTORED TO THE ENTRY STATE ON RETURN.   :
                        ;               ONLY THE CMOS LOCATION AND THE NMI STATE IS CHANGED.        :
                        ;--------------------------------------------------------------------------

001C                    CMOS_WRITE      PROC    NEAR                    ;       WRITE (AH) TO LOCATION (AL)
001C 9C                         PUSHF                                   ; SAVE INTERRUPT ENABLE STATUS AND FLAGS
001D 50                         PUSH    AX                              ; SAVE WORK REGISTER VALUES
001E D0 C0                      ROL     AL,1                            ; MOVE NMI BIT TO LOW POSITION
0020 F9                         STC                                     ; FORCE NMI BIT ON IN CARRY FLAG
0021 D0 D8                      RCR     AL,1                            ; HIGH BIT ON TO DISABLE NMI - OLD IN CY
0023 FA                         CLI                                     ; DISABLE INTERRUPTS
0024 E6 70                      OUT     CMOS_PORT,AL                    ; ADDRESS LOCATION AND DISABLE NMI
0026 8A C4                      MOV     AL,AH                           ; GET THE DATA BYTE TO WRITE
0028 E6 71                      OUT     CMOS_DATA,AL                    ; PLACE IN REQUESTED CMOS LOCATION
002A B0 1E                      MOV     AL,CMOS_SHUT_DOWN*2             ; GET ADDRESS OF DEFAULT LOCATION
002C D0 D8                      RCR     AL,1                            ; PUT ORIGINAL NMI MASK BIT INTO ADDRESS
002E E6 70                      OUT     CMOS_PORT,AL                    ; SET DEFAULT TO READ ONLY REGISTER
0030 90                         NOP                                     ; I/O DELAY
0031 E4 71                      IN      AL,CMOS_DATA                    ; OPEN STANDBY LATCH
0033 58                         POP     AX                              ; RESTORE WORK REGISTERS
0034 0E                         PUSH    CS                              ; *PLACE CODE SEGMENT IN STACK AND
0035 E8 0039 R                  CALL    CMOS_POPF                       ; *HANDLE POPF FOR B- LEVEL 80286
0038 C3                         RET

0039                    CMOS_WRITE      ENDP
```

```
                        PAGE
                CMOS_POPF       PROC    NEAR            ;       POPF FOR LEVEL B- PARTS
 0039 CF                        IRET                    ; RETURN FAR AND RESTORE FLAGS
                CMOS_POPF       ENDP

                DDS     PROC    NEAR                    ;       LOAD (DS) TO DATA AREA
 003A 2E: 8E 1E 0040 R          MOV     DS,CS:DDSDATA   ; PUT SEGMENT VALUE OF DATA AREA INTO DS
 003F C3                        RET                     ; RETURN TO USER WITH (DS)= DATA

 0040 ---- R           DDSDATA DW      DATA            ; SEGMENT SELECTOR VALUE FOR DATA AREA

 0042                   DDS     ENDP

                        ;--- E_MSG -- P_MSG ----------------------------------------------------------
                        ;       THIS SUBROUTINE WILL PRINT A MESSAGE ON THE DISPLAY                     :
                        ;                                                                               :
                        ; ENTRY REQUIREMENTS:                                                          :
                        ;       SI = OFFSET(ADDRESS) OF MESSAGE BUFFER                                  :
                        ;       CX = MESSAGE BYTE COUNT                                                 :
                        ;            MAXIMUM MESSAGE LENGTH IS 36 CHARACTERS                            :
                        ;       BP = BIT 0=E161/E162, BIT 1=CONFIG_BAD, 2-15= FIRST MSG OFFSET          :
                        ;-----------------------------------------------------------------------------

 0042                   E_MSG   PROC    NEAR
 0042 F7 C5 3FFF                TEST    BP,03FFFH       ; CHECK FOR NOT FIRST ERROR MESSAGE
 0046 75 08                     JNZ     E_MSG1          ; SKIP IF NOT FIRST ERROR MESSAGE

 0048 56                        PUSH    SI              ; SAVE MESSAGE POINTER
 0049 81 E6 3FFF                AND     SI,03FFFH       ; USE LOW 14 BITS OF MESSAGE OFFSET
 004D 0B EE                     OR      BP,SI           ;  AS FIRST ERROR MESSAGE FLAG
 004F 5E                        POP     SI              ; (BIT 0 = E161/E162, BIT 1 = BAD_CONFIG
 0050                   E_MSG1:
 0050 E8 0069 R                 CALL    P_MSG           ; PRINT MESSAGE
 0053 1E                        PUSH    DS              ; SAVE CALLERS (DS)
 0054 E8 003A R                 CALL    DDS             ; POINT TO POST/BIOS DATA SEGMENT
 0057 F6 06 0010 R 01           TEST    BYTE PTR @EQUIP_FLAG,01H; LOOP/HALT ON ERROR SWITCH ON ?
 005C 74 02                     JZ      MFG_HALT        ; YES - THEN GO TO MANUFACTURING HALT

 005E 1F                        POP     DS              ; RESTORE CALLERS (DS)
 005F C3                        RET

 0060                   MFG_HALT:                       ; MANUFACTURING LOOP MODE ERROR TRAP
 0060 FA                        CLI                     ; DISABLE INTERRUPTS
 0061 A0 0015 R                 MOV     AL,@MFG_ERR_FLAG ; RECOVER ERROR INDICATOR
 0064 E6 80                     OUT     MFG_PORT,AL     ; SET INTO MANUFACTURING PORT
 0066 F4                        HLT                     ; HALT SYSTEM
 0067 EB F7                     JMP     MFG_HALT        ; HOT NMI TRAP

 0069                   E_MSG   ENDP

 0069                   P_MSG   PROC    NEAR            ;       DISPLAY STRING FROM (CS:)
 0069 2E: 8A 04                 MOV     AL,CS:[SI]      ; PUT CHARACTER IN (AL)
 006C 46                        INC     SI              ; POINT TO NEXT CHARACTER
 006D 50                        PUSH    AX              ; SAVE PRINT CHARACTER
 006E E8 012E R                 CALL    PRT_HEX         ; CALL VIDEO_IO
 0071 58                        POP     AX              ; RECOVER PRINT CHARACTER
 0072 3C 0A                     CMP     AL,LF           ; WAS IT LINE FEED?
 0074 75 F3                     JNE     P_MSG           ; NO, KEEP PRINTING STRING
 0076 C3                        RET

 0077                   P_MSG   ENDP

                        ;--- ERR_BEEP ----------------------------------------------------------------
                        ;       THIS PROCEDURE WILL ISSUE LONG TONES (1-3/4 SECONDS) AND ONE OR          :
                        ;       MORE SHORT TONES (9/32 SECOND) TO INDICATE A FAILURE ON THE              :
                        ;       PLANAR BOARD, A BAD MEMORY MODULE, OR A PROBLEM WITH THE CRT.            :
                        ; ENTRY PARAMETERS:                                                              :
                        ;       DH = NUMBER OF LONG TONES TO BEEP.                                       :
                        ;       DL = NUMBER OF SHORT TONES TO BEEP.                                      :
                        ;-----------------------------------------------------------------------------

 0077                   ERR_BEEP        PROC    NEAR
 0077 9C                        PUSHF                   ; SAVE FLAGS
 0078 FA                        CLI                     ; DISABLE SYSTEM INTERRUPTS
 0079 0A F6                     OR      DH,DH           ; ANY LONG ONES TO BEEP
 007B 74 1E                     JZ      G3              ; NO, DO THE SHORT ONES
 007D                   G1:                             ;       LONG BEEPS
 007D B3 70                     MOV     BL,112          ; COUNTER FOR LONG BEEPS (1-3/4 SECONDS)
 007F B9 0500                   MOV     CX,1280         ; DIVISOR FOR 932 HZ
 0082 E8 00B5 R                 CALL    BEEP            ; DO THE BEEP
 0085 B9 C233                   MOV     CX,49715        ; 2/3 SECOND DELAY AFTER LONG BEEP
 0088 E8 00FB R                 CALL    WAITF           ; DELAY BETWEEN BEEPS
 008B FE CE                     DEC     DH              ; ANY MORE LONG BEEPS TO DO
 008D 75 EE                     JNZ     G1              ; LOOP TILL DONE

 008F 1E                        PUSH    DS              ; SAVE DS REGISTER CONTENTS
 0090 E8 003A R                 CALL    DDS
 0093 80 3E 0012 R 01           CMP     @MFG_TST,01H    ; MANUFACTURING TEST MODE?
 0098 1F                        POP     DS              ; RESTORE ORIGINAL CONTENTS OF (DS)
 0099 74 C5                     JE      MFG_HALT        ; YES - STOP BLINKING LED

 009B                   G3:                             ;       SHORT BEEPS
 009B B3 12                     MOV     BL,18           ; COUNTER FOR A SHORT BEEP (9/32)
 009D B9 04B8                   MOV     CX,1208         ; DIVISOR FOR 987 HZ
 00A0 E8 00B5 R                 CALL    BEEP            ; DO THE SOUND
 00A3 B9 8178                   MOV     CX,33144        ; 1/2 SECOND DELAY AFTER SHORT BEEP
 00A6 E8 00FB R                 CALL    WAITF           ; DELAY BETWEEN BEEPS
 00A9 FE CA                     DEC     DL              ; DONE WITH SHORT BEEPS COUNT
 00AB 75 EE                     JNZ     G3              ; LOOP TILL DONE

 00AD B9 8178                   MOV     CX,33144        ; 1/2 SECOND DELAY AFTER LAST BEEP
 00B0 E8 00FB R                 CALL    WAITF           ; MAKE IT ONE SECOND DELAY BEFORE RETURN
 00B3 9D                        POPF                    ; RESTORE FLAGS TO ORIGINAL SETTINGS
 00B4 C3                        RET                     ; RETURN TO CALLER

 00B5                   ERR_BEEP        ENDP
```

SECTION 5

```
                        PAGE
                        ;--- BEEP ------------------------------------------------------------------
                        ;       ROUTINE TO SOUND THE BEEPER USING TIMER 2 FOR TONE                  :
                        ; ENTRY:                                                                    :
                        ;       (BL) = DURATION COUNTER  ( 1 FOR 1/64 SECOND )                      :
                        ;       (CX) = FREQUENCY DIVISOR (1193180/FREQUENCY) (1331 FOR 886 HZ)      :
                        ; EXIT:                                                                     :
                        ;       (AX),(BL),(CX) MODIFIED.                                            :
                        ;-----------------------------------------------------------------------------

00B5                    BEEP    PROC    NEAR                    ;       SETUP TIMER 2
00B5 9C                         PUSHF                           ; SAVE INTERRUPT STATUS
00B6 FA                         CLI                             ; BLOCK INTERRUPTS DURING UPDATE
00B7 B0 B6                      MOV     AL,10110110B            ; SELECT TIMER 2,LSB,MSB,BINARY
00B9 E6 43                      OUT     TIMER+3,AL              ; WRITE THE TIMER MODE REGISTER
00BB EB 00                      JMP     $+2                     ; I/O DELAY
00BD 8A C1                      MOV     AL,CL                   ; DIVISOR FOR HZ (LOW)
00BF E6 42                      OUT     TIMER+2,AL              ; WRITE TIMER 2 COUNT - LSB
00C1 EB 00                      JMP     $+2                     ; I/O DELAY
00C3 8A C5                      MOV     AL,CH                   ; DIVISOR FOR HZ (HIGH)
00C5 E6 42                      OUT     TIMER+2,AL              ; WRITE TIMER 2 COUNT - MSB
00C7 E4 61                      IN      AL,PORT_B               ; GET CURRENT SETTING OF PORT
00C9 8A E0                      MOV     AH,AL                   ; SAVE THAT SETTING
00CB 0C 03                      OR      AL,GATE2+SPK2           ; GATE TIMER 2 AND TURN SPEAKER ON
00CD E6 61                      OUT     PORT_B,AL               ; AND RESTORE INTERRUPT STATUS
00CF 9D                         POPF
00D0                    G7:                                     ;       1/64 SECOND PER COUNT (BL)
00D0 B9 040B                    MOV     CX,1035                 ; DELAY COUNT FOR 1/64 OF A SECOND
00D3 E8 00FB R                  CALL    WAITF                   ; GO TO BEEP DELAY 1/64 COUNT
00D6 FE CB                      DEC     BL                      ; (BL) LENGTH COUNT EXPIRED?
00D8 75 F6                      JNZ     G7                      ; NO - CONTINUE BEEPING SPEAKER

00DA 9C                         PUSHF                           ; SAVE INTERRUPT STATUS
00DB FA                         CLI                             ; BLOCK INTERRUPTS DURING UPDATE
00DC E4 61                      IN      AL,PORT_B               ; GET CURRENT PORT VALUE
00DE 0C FC                      OR      AL,NOT (GATE2+SPK2)     ; ISOLATE CURRENT SPEAKER BITS IN CASE
00E0 22 E0                      AND     AH,AL                   ;  SOMEONE TURNED THEM OFF DURING BEEP
00E2 8A C4                      MOV     AL,AH                   ; RECOVER VALUE OF PORT
00E4 24 FC                      AND     AL,NOT (GATE2+SPK2)     ; FORCE SPEAKER DATA OFF
00E6 E6 61                      OUT     PORT_B,AL               ; AND STOP SPEAKER TIMER
00E8 9D                         POPF                            ; RESTORE INTERRUPT FLAG STATE
00E9 B9 040B                    MOV     CX,1035                 ; FORCE 1/64 SECOND DELAY (SHORT)
00EC E8 00FB R                  CALL    WAITF                   ; MINIMUM DELAY BETWEEN ALL BEEPS
00EF 9C                         PUSHF                           ; SAVE INTERRUPT STATUS
00F0 FA                         CLI                             ; BLOCK INTERRUPTS DURING UPDATE
00F1 E4 61                      IN      AL,PORT_B               ; GET CURRENT PORT VALUE IN CASE
00F3 24 03                      AND     AL,GATE2+SPK2           ;  SOMEONE TURNED THEM ON
00F5 0A C4                      OR      AL,AH                   ; RECOVER VALUE OF PORT_B
00F7 E6 61                      OUT     PORT_B,AL               ; RESTORE SPEAKER STATUS
00F9 9D                         POPF                            ; RESTORE INTERRUPT FLAG STATE
00FA C3                         RET

00FB                    BEEP    ENDP

                        ;--- WAITF -----------------------------------------------------------------
                        ;       FIXED TIME WAIT ROUTINE (HARDWARE CONTROLLED - NOT PROCESSOR)       :
                        ;                                                                           :
                        ; ENTRY:                                                                    :
                        ;       (CX) = COUNT OF 15.085737 MICROSECOND INTERVALS TO WAIT             :
                        ;              MEMORY REFRESH TIMER 1 OUTPUT USED AS REFERENCE              :
                        ; EXIT:                                                                     :
                        ;              AFTER (CX) TIME COUNT (PLUS OR MINUS 16 MICROSECONDS)        :
                        ;       (CX) = 0                                                            :
                        ;-----------------------------------------------------------------------------

00FB                    WAITF   PROC    NEAR                    ;       DELAY FOR  (CX)*15.085737 US
00FB 50                         PUSH    AX                      ; SAVE WORK REGISTER (AH)

00FC                    WAITF1:                                 ;       USE TIMER 1 OUTPUT BITS
00FC E4 61                      IN      AL,PORT_B               ; READ CURRENT COUNTER OUTPUT STATUS
00FE 24 10                      AND     AL,REFRESH_BIT          ; MASK FOR REFRESH DETERMINE BIT
0100 3A C4                      CMP     AL,AH                   ; DID IT JUST CHANGE
0102 74 F8                      JE      WAITF1                  ; WAIT FOR A CHANGE IN OUTPUT LINE

0104 8A E0                      MOV     AH,AL                   ; SAVE NEW FLAG STATE
0106 E2 F4                      LOOP    WAITF1                  ; DECREMENT HALF CYCLES TILL COUNT END

0108 58                         POP     AX                      ; RESTORE (AH)
0109 C3                         RET                             ; RETURN  (CX)= 0

010A                    WAITF   ENDP

                        ;--- CONFIG_BAD ------------------------------------------------------------
                        ;       SET CMOS_DIAG WITH CONFIG ERROR BIT (WITH NMI DISABLED)             :
                        ;       (BP) BIT 14 SET ON TO INDICATE CONFIGURATION ERROR                  :
                        ;-----------------------------------------------------------------------------

010A                    CONFIG_BAD      PROC    NEAR
010A 50                         PUSH    AX
010B B8 8E8E                    MOV     AX,X*(CMOS_DIAG+NMI)    ; ADDRESS CMOS DIAGNOSTIC STATUS BYTE
010E E8 0000 R                  CALL    CMOS_READ               ; GET CURRENT VALUE
0111 0C 20                      OR      AL,BAD_CONFIG           ; SET BAD CONFIGURATION BIT
0113 86 E0                      XCHG    AH,AL                   ; SETUP FOR WRITE
0115 E8 001C R                  CALL    CMOS_WRITE              ; UPDATE CMOS WITH BAD CONFIGURATION
0118 58                         POP     AX
0119 81 CD 4000                 OR      BP,04000H               ; SET CONFIGURATION BAD FLAG IN (BP)
011D C3                         RET

011E                    CONFIG_BAD      ENDP
```

```
                        PAGE
                        ;--- XPC_BYTE -- XLATE_PR -- PRT_HEX ----------------------------------
                        ;                                                                      :
                        ;       CONVERT AND PRINT ASCII CODE CHARACTERS                        :
                        ;                                                                      :
                        ;       AL CONTAINS NUMBER TO BE CONVERTED.                            :
                        ;       AX AND BX DESTROYED.                                           :
                        ;----------------------------------------------------------------------

 011E                   XPC_BYTE        PROC    NEAR                    ;       DISPLAY TWO HEX DIGITS
 011E 50                        PUSH    AX                              ; SAVE FOR LOW NIBBLE DISPLAY
 011F C0 E8 04                  SHR     AL,4                            ; NIBBLE SWAP
 0122 E8 0128 R                 CALL    XLAT_PR                         ; DO THE HIGH NIBBLE DISPLAY
 0125 58                        POP     AX                              ; RECOVER THE NIBBLE
 0126 24 0F                     AND     AL,0FH                          ; ISOLATE TO LOW NIBBLE
                                                                        ; FALL INTO LOW NIBBLE CONVERSION

 0128                   XLAT_PR PROC    NEAR                    ;       CONVERT 00-0F TO ASCII CHARACTER
 0128 04 90                     ADD     AL,090H                 ; ADD FIRST CONVERSION FACTOR
 012A 27                        DAA                             ; ADJUST FOR NUMERIC AND ALPHA RANGE
 012B 14 40                     ADC     AL,040H                 ; ADD CONVERSION AND ADJUST LOW NIBBLE
 012D 27                        DAA                             ; ADJUST HIGH NIBBLE TO ASCII RANGE

 012E                   PRT_HEX PROC    NEAR
 012E B4 0E                     MOV     AH,0EH                  ; DISPLAY CHARACTER IN (AL) COMMAND
 0130 B7 00                     MOV     BH,0
 0132 CD 10                     INT     10H                     ; CALL VIDEO_IO
 0134 C3                        RET

 0135                   PRT_HEX ENDP
 0135                   XLAT_PR ENDP
 0135                   XPC_BYTE        ENDP

                        ;--- PRT_SEG ----------------------------------------------------------
                        ;       PRINT A SEGMENT VALUE TO LOOK LIKE A 21 BIT ADDRESS            :
                        ;       DX MUST CONTAIN SEGMENT VALUE TO BE PRINTED                    :
                        ;----------------------------------------------------------------------

 0135                   PRT_SEG PROC    NEAR
 0135 8A C6                     MOV     AL,DH                   ; GET MSB
 0137 E8 011E R                 CALL    XPC_BYTE                ; DISPLAY SEGMENT HIGH BYTE
 013A 8A C2                     MOV     AL,DL                   ; LSB
 013C E8 011E R                 CALL    XPC_BYTE                ; DISPLAY SEGMENT LOW BYTE
 013F B0 30                     MOV     AL,'0'                  ; PRINT A '0 '
 0141 E8 012E R                 CALL    PRT_HEX                 ;  TO MAKE LOOK LIKE ADDRESS
 0144 B0 20                     MOV     AL,' '                  ; ADD ENDING SPACE
 0146 E8 012E R                 CALL    PRT_HEX
 0149 C3                        RET

 014A                   PRT_SEG ENDP

                        ;--- PROT_PRT_HEX -----------------------------------------------------
                        ;                                                                      :
                        ;       PUT A CHARACTER TO THE DISPLAY BUFFERS WHEN IN PROTECTED MODE   :
                        ;                                                                      :
                        ;       (AL)= ASCII CHARACTER                                          :
                        ;       (DI)= DISPLAY REGEN BUFFER POSITION                            :
                        ;----------------------------------------------------------------------

 014A                   PROT_PRT_HEX    PROC    NEAR
 014A 06                        PUSH    ES                      ; SAVE CURRENT SEGMENT REGISTERS
 014B 57                        PUSH    DI
 014C D1 E7                     SAL     DI,1                    ; MULTIPLY OFFSET BY TWO

                        ;-----  MONOCHROME VIDEO CARD

 014E 6A 20                     PUSH    BYTE PTR C_BWCRT_PTR    ; GET MONOCHROME BUFFER SEGMENT SELECTOR
 0150 07                        POP     ES                      ; SET (ES) TO B/W DISPLAY BUFFER
 0151 AA                        STOSB                           ; PLACE CHARACTER IN BUFFER
 0152 4F                        DEC     DI                      ; ADJUST POINTER BACK

                        ;-----  ENHANCED GRAPHICS ADAPTER

 0153 6A 30                     PUSH    BYTE PTR E_CCRT_PTR     ; ENHANCED COLOR DISPLAY POINTER LOW 64K
 0155 07                        POP     ES                      ; LOAD SEGMENT SELECTOR
 0156 AA                        STOSB                           ; PLACE CHARACTER IN BUFFER
 0157 4F                        DEC     DI                      ; ADJUST POINTER BACK
 0158 6A 38                     PUSH    BYTE PTR E_CCRT_PTR2    ; ENHANCED COLOR DISPLAY POINTER HI 64K
 015A 07                        POP     ES                      ; LOAD SEGMENT SELECTOR
 015B AA                        STOSB                           ; PLACE CHARACTER IN BUFFER
 015C 4F                        DEC     DI                      ; ADJUST POINTER BACK

                        ;-----  COMPATIBLE COLOR

 015D 6A 28                     PUSH    BYTE PTR C_CCRT_PTR     ; SET (DS) TO COMPATIBLE COLOR MEMORY
 015F 07                        POP     ES
 0160 53                        PUSH    BX                      ; SAVE WORK REGISTERS
 0161 52                        PUSH    DX
 0162 51                        PUSH    CX
 0163 33 C9                     XOR     CX,CX                   ; TIMEOUT LOOP FOR "BAD" HARDWARE
 0165 BA 03DA                   MOV     DX,03DAH                ; STATUS ADDRESS OF COLOR CARD
 0168 93                        XCHG    AX,BX                   ; SAVE IN (BX) REGISTER
 0169                   PROT_S: IN      AL,DX                   ; GET COLOR CARD STATUS
 0169 EC                        TEST    AL,RVRT+RHRZ            ; CHECK FOR VERTICAL RETRACE (OR HORZ)
 016A A8 09                     LOOPZ   PROT_S                  ; TIMEOUT LOOP TILL FOUND
 016C E1 FB                     XCHG    AX,BX                   ; RECOVER CHARACTERS
 016E 93                        STOSB                           ; PLACE CHARACTER IN BUFFER
 016F AA

 0170 59                        POP     CX                      ; RESTORE REGISTERS
 0171 5A                        POP     DX
 0172 5B                        POP     BX
 0173 5F                        POP     DI
 0174 07                        POP     ES
 0175 C3                        RET

 0176                   PROT_PRT_HEX    ENDP
```

SECTION 5

```
                        PAGE
                        ;--------------------------------------
                        ;       ROM CHECKSUM SUBROUTINE        :
                        ;--------------------------------------

0176                    ROM_CHECKSUM    PROC    NEAR
0176 2B C9                      SUB     CX,CX                   ; NUMBER OF BYTES TO ADD IS 64K

0178                    ROM_CHECKSUM_CNT:                       ; ENTRY FOR OPTIONAL ROM TEST
0178 32 C0                      XOR     AL,AL
017A                    ROM_L:
017A 02 07                      ADD     AL,[BX]                 ; GET (DS:BX)
017C 43                         INC     BX                      ; POINT TO NEXT BYTE
017D E2 FB                      LOOP    ROM_L                   ; ADD ALL BYTES IN ROM MODULE

017F 0A C0                      OR      AL,AL                   ; SUM = 0?
0181 C3                         RET

0182                    ROM_CHECKSUM    ENDP

                        ;-----------------------------------------------------------------------
                        ;       THIS ROUTINE CHECKSUMS OPTIONAL ROM MODULES AND                 :
                        ;       IF CHECKSUM IS OK, CALLS INITIALIZATION/TEST CODE IN MODULE     :
                        ;-----------------------------------------------------------------------

0182                    ROM_CHECK       PROC    NEAR
0182 B8 ---- R                  MOV     AX,DATA                 ; POINT ES TO DATA AREA
0185 8E C0                      MOV     ES,AX
0187 2A E4                      SUB     AH,AH                   ; ZERO OUT AH
0189 8A 47 02                   MOV     AL,[BX+2]               ; GET LENGTH INDICATOR
018C C1 E0 09                   SHL     AX,9                    ; MULTIPLY BY 512
018F 8B C8                      MOV     CX,AX                   ; SET COUNT
0191 C1 E8 04                   SHR     AX,4
0194 03 D0                      ADD     DX,AX                   ; SET POINTER TO NEXT MODULE
0196 E8 0178 R                  CALL    ROM_CHECKSUM_CNT        ; DO CHECKSUM
0199 74 05                      JZ      ROM_CHECK_1

019B E8 0000 E                  CALL    ROM_ERR                 ; POST CHECKSUM ERROR
019E EB 13                      JMP     SHORT ROM_CHECK_END     ; AND EXIT

01A0                    ROM_CHECK_1:
01A0 52                         PUSH    DX                      ; SAVE POINTER
01A1 26: C7 06 0067 R 0003      MOV     ES:@IO_ROM_INIT,0003H   ; LOAD OFFSET
01A8 26: 8C 1E 0069 R           MOV     ES:@IO_ROM_SEG,DS       ; LOAD SEGMENT
01AD 26: FF 1E 0067 R           CALL    DWORD PTR ES:@IO_ROM_INIT; CALL INITIALIZE/TEST ROUTINE
01B2 5A                         POP     DX

01B3                    ROM_CHECK_END:
01B3 C3                         RET                             ; RETURN TO CALLER

01B4                    ROM_CHECK       ENDP

                        ;--- KBD_RESET ---------------------------------------------------------
                        ;       THIS PROCEDURE WILL SEND A SOFTWARE RESET TO THE KEYBOARD.      :
                        ;       SCAN CODE  0AAH  SHOULD BE RETURNED TO THE PROCESSOR.          :
                        ;       SCAN CODE  065H  IS DEFINED FOR MANUFACTURING TEST             :
                        ;-----------------------------------------------------------------------

01B4                    KBD_RESET       PROC    NEAR
01B4 B0 FF                      MOV     AL,0FFH                 ; SET KEYBOARD RESET COMMAND
01B6 E8 0000 E                  CALL    XMIT_8042               ; GO ISSUE THE COMMAND
01B9 E3 23                      JCXZ    G13                     ; EXIT IF ERROR

01BB 3C FA                      CMP     AL,KB_ACK
01BD 75 1F                      JNZ     G13

01BF B0 FD                      MOV     AL,0FDH                 ; ENABLE KEYBOARD INTERRUPTS
01C1 E6 21                      OUT     INTA01,AL               ; WRITE 8259 INTERRUPT MASK REGISTER
01C3 C6 06 006B R 00            MOV     @INTR_FLAG,0            ; RESET INTERRUPT INDICATOR
01C8 FB                         STI                             ; ENABLE INTERRUPTS
01C9 B3 0A                      MOV     BL,10                   ; TRY FOR 400 MILLISECONDS
01CB 2B C9                      SUB     CX,CX                   ; SETUP INTERRUPT TIMEOUT COUNT
01CD                    G11:
01CD F6 06 006B R 02            TEST    @INTR_FLAG,02H          ; DID A KEYBOARD INTERRUPT OCCUR ?
01D2 75 06                      JNZ     G12                     ; YES - READ SCAN CODE RETURNED
01D4 E2 F7                      LOOP    G11                     ; NO - LOOP TILL TIMEOUT

01D6 FE CB                      DEC     BL
01D8 75 F3                      JNZ     G11                     ; TRY AGAIN
01DA                    G12:
01DA E4 60                      IN      AL,PORT_A               ; READ KEYBOARD SCAN CODE
01DC 8A D8                      MOV     BL,AL                   ; SAVE SCAN CODE JUST READ
01DE                    G13:
01DE C3                         RET                             ; RETURN TO CALLER

01DF                    KBD_RESET       ENDP

                        ;-------------------------------------------------------
                        ;       BLINK LED PROCEDURE FOR MFG RUN-IN TESTS        :
                        ;       IF LED IS ON, TURN IT OFF. IF OFF, TURN ON.     :
                        ;-------------------------------------------------------

01DF                    BLINK_INT       PROC    NEAR
01DF FB                         STI
01E0 50                         PUSH    AX                      ; SAVE AX REGISTER CONTENTS
01E1 E4 80                      IN      AL,MFG_PORT             ; READ CURRENT VALUE OF MFG_PORT
01E3 34 40                      XOR     AL,01000000B            ; FLIP CONTROL BIT
01E5 E6 80                      OUT     MFG_PORT,AL
01E7 B0 20                      MOV     AL,EOI
01E9 E6 20                      OUT     INTA00,AL
01EB 58                         POP     AX                      ; RESTORE AX REGISTER
01EC CF                         IRET

01ED                    BLINK_INT       ENDP
```

```
                        PAGE
                        ;--------------------------------------------------------------------------------
                        ;       THIS ROUTINE INITIALIZES THE TIMER DATA AREA IN THE ROM BIOS             :
                        ;       DATA AREA.  IT IS CALLED BY THE POWER ON ROUTINES.  IT CONVERTS          :
                        ;       HR:MIN:SEC  FROM CMOS TO TIMER TICS.  IF CMOS IS INVALID, TIMER         :
                        ;       IS SET TO ZERO.                                                         :
                        ;                                                                               :
                        ; INPUT    NONE PASSED TO ROUTINE BY CALLER                                     :
                        ;          CMOS LOCATIONS USED FOR TIME                                         :
                        ;                                                                               :
                        ; OUTPUT   @TIMER_LOW                                                           :
                        ;          @TIMER_HIGH                                                          :
                        ;          @TIMER_OFL                                                           :
                        ;          ALL REGISTERS UNCHANGED                                              :
                        ;--------------------------------------------------------------------------------
= 0012                  COUNTS_SEC      EQU     18              ; TIMER DATA CONVERSION EQUATES
= 0444                  COUNTS_MIN      EQU     1092
= 0007                  COUNTS_HOUR     EQU     7               ; 65543 - 65536
= 0080                  UPDATE_TIMER    EQU     10000000B       ; RTC UPDATE IN PROCESS BIT MASK

01ED                    SET_TOD PROC    NEAR
01ED 60                         PUSHA
01EE 1E                         PUSH    DS
01EF E8 003A R                  CALL    DDS                     ; ESTABLISH SEGMENT
01F2 2B C0                      SUB     AX,AX
01F4 A2 0070 R                  MOV     @TIMER_OFL,AL           ; RESET TIMER ROLL OVER INDICATOR
01F7 A3 006C R                  MOV     @TIMER_LOW,AX           ; AND TIMER COUNT
01FA A3 006E R                  MOV     @TIMER_HIGH,AX
01FD B0 8E                      MOV     AL,CMOS_DIAG+NMI        ; CHECK CMOS VALIDITY
01FF E8 0000 R                  CALL    CMOS_READ               ; READ DIAGNOSTIC LOCATION IN CMOS
0202 24 C4                      AND     AL,BAD_BAT+BAD_CKSUM+CMOS_CLK_FAIL
                                                                ; BAD BATTERY, CHKSUM ERROR, CLOCK ERROR
0204 75 68                      JNZ     POD_DONE                ; CMOS NOT VALID -- TIMER SET TO ZERO
0206 2B C9                      SUB     CX,CX                   ;
0208                    UIP:
0208 B0 8A                      MOV     AL,CMOS_REG_A+NMI       ; ACCESS REGISTER A
020A E8 0000 R                  CALL    CMOS_READ               ; READ CMOS CLOCK REGISTER A
020D A8 80                      TEST    AL,UPDATE_TIMER
020F E1 F7                      LOOPZ   UIP                     ; WAIT TILL UPDATE BIT IS ON

0211 E3 5B                      JCXZ    POD_DONE                ; CMOS CLOCK STUCK IF TIMEOUT
0213 2B C9                      SUB     CX,CX                   ;
0215                    UIPOFF:
0215 B0 8A                      MOV     AL,CMOS_REG_A+NMI       ; ACCESS REGISTER A
0217 E8 0000 R                  CALL    CMOS_READ               ; READ CMOS CLOCK REGISTER A
021A A8 80                      TEST    AL,UPDATE_TIMER
021C E0 F7                      LOOPNZ  UIPOFF                  ; NEXT WAIT TILL END OF UPDATE

021E E3 4E                      JCXZ    POD_DONE                ; CMOS CLOCK STUCK IF TIMEOUT

0220 B0 80                      MOV     AL,CMOS_SECONDS+NMI     ;       TIME JUST UPDATED
0222 E8 0000 R                  CALL    CMOS_READ               ; ACCESS SECONDS VALUE IN CMOS
0225 3C 59                      CMP     AL,59H                  ; ARE THE SECONDS WITHIN LIMITS?
0227 77 48                      JA      TOD_ERROR               ; GO IF NOT

0229 E8 0287 R                  CALL    CVT_BINARY              ; CONVERT IT TO BINARY
022C 8B C8                      MOV     CX,AX                   ; MOVE COUNT TO ACCUMULATION REGISTER
022E C1 E9 02                   SHR     CX,2                    ; ADJUST FOR SYSTEMATIC SECONDS ERROR
0231 B3 12                      MOV     BL,COUNTS_SEC
0233 F6 E3                      MUL     BL                      ; COUNT FOR SECONDS
0235 03 C8                      ADD     CX,AX
0237 B0 82                      MOV     AL,CMOS_MINUTES+NMI
0239 E8 0000 R                  CALL    CMOS_READ               ; ACCESS MINUTES VALUE IN CMOS
023C 3C 59                      CMP     AL,59H                  ; ARE THE MINUTES WITHIN LIMITS?
023E 77 31                      JA      TOD_ERROR               ; GO IF NOT
0240 E8 0287 R                  CALL    CVT_BINARY              ; CONVERT IT TO BINARY
0243 50                         PUSH    AX                      ; SAVE MINUTES COUNT
0244 D1 E8                      SHR     AX,1                    ; ADJUST FOR SYSTEMATIC MINUTES ERROR
0246 03 C8                      ADD     CX,AX                   ; ADD ADJUSTMENT TO COUNT
0248 58                         POP     AX                      ; RECOVER BCD MINUTES VALUE
0249 BB 0444                    MOV     BX,COUNTS_MIN
024C F7 E3                      MUL     BX                      ; COUNT FOR MINUTES
024E 03 C8                      ADD     CX,AX                   ; ADD TO ACCUMULATED VALUE
0250 B0 84                      MOV     AL,CMOS_HOURS+NMI
0252 E8 0000 R                  CALL    CMOS_READ               ; ACCESS HOURS VALUE IN CMOS
0255 3C 23                      CMP     AL,23H                  ; ARE THE HOURS WITHIN LIMITS?
0257 77 18                      JA      TOD_ERROR               ; GO IF NOT

0259 E8 0287 R                  CALL    CVT_BINARY              ; CONVERT IT TO BINARY
025C 8B D0                      MOV     DX,AX
025E B3 07                      MOV     BL,COUNTS_HOUR
0260 F6 E3                      MUL     BL                      ; COUNT FOR HOURS
0262 03 C1                      ADD     AX,CX
0264 83 D2 00                   ADC     DX,0000H
0267 89 16 006E R               MOV     @TIMER_HIGH,DX
026B A3 006C R                  MOV     @TIMER_LOW,AX
026E                    POD_DONE:
026E 1F                         POP     DS
026F 61                         POPA
0270 C3                         RET

0271                    TOD_ERROR:
0271 1F                         POP     DS                      ; RESTORE SEGMENT
0272 61                         POPA                            ; RESTORE REGISTERS
0273 BE 0000 E                  MOV     SI,OFFSET E163          ; DISPLAY CLOCK ERROR
0276 E8 0042 R                  CALL    E_MSG
0279 B8 8E8E                    MOV     AX,X*(CMOS_DIAG+NMI)    ; SET CLOCK ERROR IN STATUS
027C E8 0000 R                  CALL    CMOS_READ               ; READ DIAGNOSTIC CMOS LOCATION
027F 0C 04                      OR      AL,CMOS_CLK_FAIL        ; SET NEW STATUS WITH CMOS CLOCK ERROR
0281 86 C4                      XCHG    AL,AH                   ; MOVE NEW STATUS TO WORK REGISTER
0283 E8 001C R                  CALL    CMOS_WRITE              ; UPDATE STATUS LOCATION
0286 C3                         RET

0287                    SET_TOD ENDP

0287                    CVT_BINARY      PROC    NEAR
0287 8A E0                      MOV     AH,AL                   ; UNPACK 2 BCD DIGITS IN AL
0289 C0 EC 04                   SHR     AH,4
028C 24 0F                      AND     AL,0FH                  ; RESULT IS IN AX
028E D5 0A                      AAD                             ; CONVERT UNPACKED BCD TO BINARY
0290 C3                         RET

0291                    CVT_BINARY      ENDP
```

SECTION 5

```
                        PAGE
                        ;--- D11 -- INT  ?? H -- ( IRQ LEVEL ?? ) ------------------------------------
                        ; TEMPORARY INTERRUPT SERVICE ROUTINE FOR POST                                 :
                        ;                                                                              :
                        ;       THIS ROUTINE IS ALSO LEFT IN PLACE AFTER THE POWER ON DIAGNOSTICS      :
                        ;       TO SERVICE UNUSED INTERRUPT VECTORS.  LOCATION "@INTR_FLAG" WILL       :
                        ;       CONTAIN EITHER:                                                        :
                        ;        1) LEVEL OF HARDWARE INTERRUPT THAT CAUSED CODE TO BE EXECUTED, OR    :
                        ;        2) "FF" FOR A NON-HARDWARE INTERRUPT THAT WAS EXECUTED ACCIDENTALLY.  :
                        ;--------------------------------------------------------------------------------

0291                    D11     PROC    NEAR
0291 50                         PUSH    AX                      ; SAVE REGISTER AX CONTENTS
0292 53                         PUSH    BX
0293 B0 0B                      MOV     AL,0BH                  ; READ IN-SERVICE REGISTER
0295 E6 20                      OUT     INTA00,AL               ; (FIND OUT WHAT LEVEL BEING
0297 EB 00                      JMP     $+2                     ;  SERVICED)
0299 E4 20                      IN      AL,INTA00               ; GET LEVEL
029B 8A E0                      MOV     AH,AL                   ; SAVE IT
029D 0A C4                      OR      AL,AH                   ; 007 (NO HARDWARE ISR ACTIVE)
029F 75 04                      JNZ     HW_INT

02A1 B4 FF                      MOV     AH,0FFH
02A3 EB 2F                      JMP     SHORT SET_INTR_FLAG     ; SET FLAG TO "FF" IF NON-HARDWARE
02A5                    HW_INT:
02A5 B0 0B                      MOV     AL,0BH                  ; READ IN-SERVICE REGISTER FROM
02A7 E6 A0                      OUT     INTB00,AL               ; INTERRUPT CHIP #2
02A9 EB 00                      JMP     $+2                     ; I/O DELAY
02AB E4 A0                      IN      AL,INTB00               ; CHECK THE SECOND INTERRUPT CHIP
02AD 8A F8                      MOV     BH,AL                   ; SAVE IT
02AF 0A FF                      OR      BH,BH
02B1 74 10                      JZ      NOT_SEC                 ; CONTINUE IF NOT

02B3 E4 A1                      IN      AL,INTB01               ; GET SECOND INTERRUPT MASK
02B5 0A C7                      OR      AL,BH                   ; MASK OFF LEVEL BEING SERVICED
02B7 EB 00                      JMP     $+2                     ; I/O DELAY
02B9 E6 A1                      OUT     INTB01,AL
02BB B0 20                      MOV     AL,EOI                  ; SEND EOI TO SECOND CHIP
02BD EB 00                      JMP     $+2                     ; I/O DELAY
02BF E6 A0                      OUT     INTB00,AL
02C1 EB 0D                      JMP     SHORT IS_SEC
02C3                    NOT_SEC:
02C3 E4 21                      IN      AL,INTA01               ; GET CURRENT MASK VALUE
02C5 EB 00                      JMP     $+2                     ; I/O DELAY
02C7 80 E4 FB                   AND     AH,0FBH                 ; DO NOT DISABLE SECOND CONTROLLER
02CA 0A C4                      OR      AL,AH                   ; MASK OFF LEVEL BEING SERVICED
02CC E6 21                      OUT     INTA01,AL               ; SET NEW INTERRUPT MASK
02CE EB 00                      JMP     $+2                     ; I/O DELAY
02D0                    IS_SEC:
02D0 B0 20                      MOV     AL,EOI
02D2 E6 20                      OUT     INTA00,AL
02D4                    SET_INTR_FLAG:
02D4 5B                         POP     BX                      ; RESTORE (BX) FROM STACK
02D5 1E                         PUSH    DS                      ; SAVE ACTIVE (DS)
02D6 E8 003A R                  CALL    DDS                     ; SET DATA SEGMENT
02D9 88 26 006B R               MOV     @INTR_FLAG,AH           ; SET FLAG
02DD 1F                         POP     DS
02DE 58                         POP     AX                      ; RESTORE REGISTER AX CONTENTS
02DF                    DUMMY_RETURN_1:                         ; NEED IRET FOR VECTOR TABLE
02DF CF                         IRET

02E0                    D11     ENDP

                        ;--- HARDWARE INT  71 H -- ( IRQ LEVEL  9 ) -- TO INT  0A H ----------------
                        ; REDIRECT SLAVE INTERRUPT 9 TO INTERRUPT LEVEL 2                              :
                        ;       THIS ROUTINE FIELDS LEVEL 9 INTERRUPTS AND                             :
                        ;       CONTROL IS PASSED TO MASTER INTERRUPT LEVEL 2                          :
                        ;--------------------------------------------------------------------------------

02E0                    RE_DIRECT PROC  NEAR
02E0 50                         PUSH    AX                      ; SAVE (AX)
02E1 B0 20                      MOV     AL,EOI
02E3 E6 A0                      OUT     INTB00,AL               ; EOI TO SLAVE INTERRUPT CONTROLLER
02E5 58                         POP     AX                      ; RESTORE (AX)
02E6 CD 0A                      INT     0AH                     ; GIVE CONTROL TO HARDWARE LEVEL 2

02E8 CF                         IRET                            ; RETURN

02E9                    RE_DIRECT ENDP

                        ;--- HARDWARE INT  75 H -- ( IRQ LEVEL 13 ) ----------------------------------
                        ; SERVICE X287 INTERRUPTS                                                      :
                        ;       THIS ROUTINE FIELDS X287 INTERRUPTS AND CONTROL                        :
                        ;       IS PASSED TO THE NMI INTERRUPT HANDLER FOR                             :
                        ;       COMPATIBILITY.                                                         :
                        ;--------------------------------------------------------------------------------

02E9                    INT_287 PROC    NEAR
02E9 50                         PUSH    AX                      ; SAVE (AX)
02EA 32 C0                      XOR     AL,AL
02EC E6 F0                      OUT     X287,AL                 ; REMOVE THE INTERRUPT REQUEST

02EE B0 20                      MOV     AL,EOI                  ; ENABLE THE INTERRUPT
02F0 E6 A0                      OUT     INTB00,AL               ;  THE SLAVE
02F2 E6 20                      OUT     INTA00,AL               ;  THE MASTER
02F4 58                         POP     AX                      ; RESTORE (AX)
02F5 CD 02                      INT     02H                     ; GIVE CONTROL TO NMI

02F7 CF                         IRET                            ; RETURN

02F8                    INT_287 ENDP

02F8                    PROC_SHUTDOWN   PROC                    ;       COMMON 80286 SHUTDOWN WAIT

02F8 B0 FE                      MOV     AL,SHUT_CMD             ; SHUTDOWN COMMAND
02FA E6 64                      OUT     STATUS_PORT,AL          ; SEND TO KEYBOARD CONTROL PORT
02FC                    PROC_S:
02FC F4                         HLT                             ; WAIT FOR 80286 RESET
02FD EB FD                      JMP     PROC_S                  ; INSURE HALT

02FF                    PROC_SHUTDOWN   ENDP
02FF                    CODE    ENDS
                        END
```

```
                        PAGE 118,121
                        TITLE TEST5 ---- 06/10/85  EXCEPTION INTERRUPT TEST HANDLERS
                        .286C
                        .LIST
0000                    CODE    SEGMENT BYTE PUBLIC

                                PUBLIC  POST5
                                PUBLIC  SYSINIT1


                        ;--------------------------------------
                        ;       EXCEPTION INTERRUPT ROUTINE     :
                        ;--------------------------------------


                                ASSUME  CS:CODE,DS:ABS0
0000                    POST5:
0000                    EXC_00:
0000 B0 90                      MOV     AL,90H                  ;       <<> SET CHECKPOINT <<>
0002 E9 00B2 R                  JMP     TEST_EXC                ; GO TEST IF EXCEPTION WAS EXPECTED
0005                    EXC_01:
0005 B0 91                      MOV     AL,91H                  ;       <<> SET CHECKPOINT <<>
0007 E9 00B2 R                  JMP     TEST_EXC                ; GO TEST IF EXCEPTION WAS EXPECTED
000A                    EXC_02:
000A B0 92                      MOV     AL,92H                  ;       <<> SET CHECKPOINT <<>
000C E9 00B2 R                  JMP     TEST_EXC                ; GO TEST IF EXCEPTION WAS EXPECTED
000F                    EXC_03:
000F B0 93                      MOV     AL,93H                  ;       <<> SET CHECKPOINT <<>
0011 E9 00B2 R                  JMP     TEST_EXC                ; GO TEST IF EXCEPTION WAS EXPECTED
0014                    EXC_04:
0014 B0 94                      MOV     AL,94H                  ;       <<> SET CHECKPOINT <<>
0016 E9 00B2 R                  JMP     TEST_EXC                ; GO TEST IF EXCEPTION WAS EXPECTED
0019                    EXC_05:
0019 06                         PUSH    ES
001A 6A 48                      PUSH    BYTE PTR ES_TEMP        ; LOAD ES REGISTER WITH SELECTOR
001C 07                         POP     ES

                        ;-----  FIX BOUND PARAMETERS

001D 2B FF                      SUB     DI,DI                   ; POINT BEGINNING OF THE BLOCK
001F 26: C7 05 0000             MOV     WORD PTR ES:[DI],0      ; SET FIRST WORD TO ZERO
0024 26: C7 45 02 7FFF          MOV     WORD PTR ES:[DI+2],07FFFH ; SET SECOND TO 07FFFH
002A 07                         POP     ES
002B B0 95                      MOV     AL,95H                  ;       <<> SET CHECKPOINT <<>
002D E9 00B2 R                  JMP     TEST_EXC                ; GO TEST IF EXCEPTION WAS EXPECTED

0030                    EXC_06:
0030 B0 96                      MOV     AL,96H                  ;       <<> SET CHECKPOINT <<>
0032 EB 7E                      JMP     SHORT  TEST_EXC         ; GO TEST IF EXCEPTION WAS EXPECTED
0034                    EXC_07:
0034 B0 97                      MOV     AL,97H                  ;       <<> SET CHECKPOINT <<>
0036 EB 7A                      JMP     SHORT  TEST_EXC         ; GO TEST IF EXCEPTION WAS EXPECTED
0038                    EXC_08:
0038 B0 98                      MOV     AL,98H                  ;       <<> SET CHECKPOINT <<>
003A EB 76                      JMP     SHORT  TEST_EXC         ; GO TEST IF EXCEPTION WAS EXPECTED
003C                    EXC_09:
003C B0 99                      MOV     AL,99H                  ;       <<> SET CHECKPOINT <<>
003E EB 72                      JMP     SHORT  TEST_EXC         ; GO TEST IF EXCEPTION WAS EXPECTED
0040                    EXC_10:
0040 B0 9A                      MOV     AL,9AH                  ;       <<> SET CHECKPOINT <<>
0042 EB 6E                      JMP     SHORT  TEST_EXC         ; GO TEST IF EXCEPTION WAS EXPECTED
0044                    EXC_11:
0044 B0 9B                      MOV     AL,9BH                  ;       <<> SET CHECKPOINT <<>
0046 EB 6A                      JMP     SHORT  TEST_EXC         ; GO TEST IF EXCEPTION WAS EXPECTED
0048                    EXC_12:
0048 B0 9C                      MOV     AL,9CH                  ;       <<> SET CHECKPOINT <<>
004A EB 66                      JMP     SHORT  TEST_EXC         ; GO TEST IF EXCEPTION WAS EXPECTED
004C                    EXC_13:
004C B0 9D                      MOV     AL,9DH                  ;       <<> SET CHECKPOINT <<>
004E EB 62                      JMP     SHORT  TEST_EXC         ; GO TEST IF EXCEPTION WAS EXPECTED
0050                    EXC_14:
0050 B0 9E                      MOV     AL,9EH                  ;       <<> SET CHECKPOINT <<>
0052 EB 5E                      JMP     SHORT  TEST_EXC         ; GO TEST IF EXCEPTION WAS EXPECTED
0054                    EXC_15:
0054 B0 9F                      MOV     AL,9FH                  ;       <<> SET CHECKPOINT <<>
0056 EB 5A                      JMP     SHORT  TEST_EXC         ; GO TEST IF EXCEPTION WAS EXPECTED
0058                    EXC_16:
0058 B0 A0                      MOV     AL,0A0H                 ;       <<> SET CHECKPOINT <<>
005A EB 56                      JMP     SHORT  TEST_EXC         ; GO TEST IF EXCEPTION WAS EXPECTED
005C                    EXC_17:
005C B0 A1                      MOV     AL,0A1H                 ;       <<> SET CHECKPOINT <<>
005E EB 52                      JMP     SHORT  TEST_EXC         ; GO TEST IF EXCEPTION WAS EXPECTED
0060                    EXC_18:
0060 B0 A2                      MOV     AL,0A2H                 ;       <<> SET CHECKPOINT <<>
0062 EB 4E                      JMP     SHORT  TEST_EXC         ; GO TEST IF EXCEPTION WAS EXPECTED
0064                    EXC_19:
0064 B0 A3                      MOV     AL,0A3H                 ;       <<> SET CHECKPOINT <<>
0066 EB 4A                      JMP     SHORT  TEST_EXC         ; GO TEST IF EXCEPTION WAS EXPECTED
0068                    EXC_20:
0068 B0 A4                      MOV     AL,0A4H                 ;       <<> SET CHECKPOINT <<>
006A EB 46                      JMP     SHORT  TEST_EXC         ; GO TEST IF EXCEPTION WAS EXPECTED
006C                    EXC_21:
006C B0 A5                      MOV     AL,0A5H                 ;       <<> SET CHECKPOINT <<>
006E EB 42                      JMP     SHORT  TEST_EXC         ; GO TEST IF EXCEPTION WAS EXPECTED
0070                    EXC_22:
0070 B0 A6                      MOV     AL,0A6H                 ;       <<> SET CHECKPOINT <<>
0072 EB 3E                      JMP     SHORT  TEST_EXC         ; GO TEST IF EXCEPTION WAS EXPECTED
0074                    EXC_23:
0074 B0 A7                      MOV     AL,0A7H                 ;       <<> SET CHECKPOINT <<>
0076 EB 3A                      JMP     SHORT  TEST_EXC         ; GO TEST IF EXCEPTION WAS EXPECTED
0078                    EXC_24:
0078 B0 A8                      MOV     AL,0A8H                 ;       <<> SET CHECKPOINT <<>
007A EB 36                      JMP     SHORT  TEST_EXC         ; GO TEST IF EXCEPTION WAS EXPECTED
007C                    EXC_25:
007C B0 A9                      MOV     AL,0A9H                 ;       <<> SET CHECKPOINT <<>
007E EB 32                      JMP     SHORT  TEST_EXC         ; GO TEST IF EXCEPTION WAS EXPECTED
0080                    EXC_26:
0080 B0 AA                      MOV     AL,0AAH                 ;       <<> SET CHECKPOINT <<>
0082 EB 2E                      JMP     SHORT  TEST_EXC         ; GO TEST IF EXCEPTION WAS EXPECTED
0084                    EXC_27:
0084 B0 AB                      MOV     AL,0ABH                 ;       <<> SET CHECKPOINT <<>
0086 EB 2A                      JMP     SHORT  TEST_EXC         ; GO TEST IF EXCEPTION WAS EXPECTED
0088                    EXC_28:
0088 B0 AC                      MOV     AL,0ACH                 ;       <<> SET CHECKPOINT <<>
008A EB 26                      JMP     SHORT  TEST_EXC         ; GO TEST IF EXCEPTION WAS EXPECTED
```

```
008C                    EXC_29:
008C B0 AD                      MOV     AL,0ADH                 ;       <<> SET CHECKPOINT <<>
008E EB 22                      JMP     SHORT  TEST_EXC         ; GO TEST IF EXCEPTION WAS EXPECTED
0090                    EXC_30:
0090 B0 AE                      MOV     AL,0AEH                 ;       <<> SET CHECKPOINT <<>
0092 EB 1E                      JMP     SHORT  TEST_EXC         ; GO TEST IF EXCEPTION WAS EXPECTED
0094                    EXC_31:
0094 B0 AF                      MOV     AL,0AFH                 ;       <<> SET CHECKPOINT <<>
0096 EB 1A                      JMP     SHORT  TEST_EXC         ; GO TEST IF EXCEPTION WAS EXPECTED
0098                    SYS_32:
0098 B0 B0                      MOV     AL,0B0H                 ;       <<> SET CHECKPOINT <<>
009A EB 16                      JMP     SHORT  TEST_EXC         ; GO TEST IF INTERRUPT WAS EXPECTED
009C                    SYS_33:
009C B0 B1                      MOV     AL,0B1H                 ;       <<> SET CHECKPOINT <<>
009E EB 12                      JMP     SHORT  TEST_EXC         ; GO TEST IF INTERRUPT WAS EXPECTED
00A0                    SYS_34:
00A0 B0 B2                      MOV     AL,0B2H                 ;       <<> SET CHECKPOINT <<>
00A2 EB 0E                      JMP     SHORT  TEST_EXC         ; GO TEST IF INTERRUPT WAS EXPECTED
00A4                    SYS_35:
00A4 B0 B3                      MOV     AL,0B3H                 ;       <<> SET CHECKPOINT <<>
00A6 EB 0A                      JMP     SHORT  TEST_EXC         ; GO TEST IF INTERRUPT WAS EXPECTED
00A8                    SYS_36:
00A8 B0 B4                      MOV     AL,0B4H                 ;       <<> SET CHECKPOINT <<>
00AA EB 06                      JMP     SHORT  TEST_EXC         ; GO TEST IF INTERRUPT WAS EXPECTED
00AC                    SYS_37:
00AC B0 B5                      MOV     AL,0B5H                 ;       <<> SET CHECKPOINT <<>
00AE EB 02                      JMP     SHORT  TEST_EXC         ; GO TEST IF INTERRUPT WAS EXPECTED
00B0                    SYS_38:
00B0 B0 B6                      MOV     AL,0B6H                 ;       <<> SET CHECKPOINT <<>
                                                                ; GO TEST IF INTERRUPT WAS EXPECTED

00B2                    TEST_EXC:
00B2 E6 80                      OUT     MFG_PORT,AL             ; OUTPUT THE CHECKPOINT
00B4 3C AF                      CMP     AL,0AFH                 ; CHECK FOR EXCEPTION
00B6 77 1C                      JA      TEST_EXC0               ; GO IF A SYSTEM INTERRUPT

00B8 1E                         PUSH    DS                      ; SAVE THE CURRENT DATA SEGMENT
00B9 6A 08                      PUSH    BYTE PTR GDT_PTR
00BB 1F                         POP     DS
00BC C7 06 0048 FFFF            MOV     DS:ES_TEMP.SEG_LIMIT,MAX_SEG_LEN
00C2 C6 06 004D 93              MOV     BYTE PTR DS:(ES_TEMP.DATA_ACC_RIGHTS),CPL0_DATA_ACCESS
00C7 6A 48                      PUSH    BYTE PTR ES_TEMP
00C9 07                         POP     ES
00CA 1F                         POP     DS                      ; RESTORE REGISTERS
00CB 5A                         POP     DX                      ; CHECK IF CODE SEGMENT SECOND ON STACK
00CC 59                         POP     CX
00CD 51                         PUSH    CX
00CE 83 F9 40                   CMP     CX,SYS_ROM_CS
00D1 75 01                      JNZ     TEST_EXC0               ; CONTINUE IF ERROR CODE

00D3 52                         PUSH    DX                      ; PUT SEGMENT BACK ON STACK
00D4                    TEST_EXC0:
00D4 86 E0                      XCHG    AH,AL                   ; SAVE THE CHECKPOINT
00D6 E4 8B                      IN      AL,DMA_PAGE+0AH
00D8 3A C4                      CMP     AL,AH                   ; WAS THE EXCEPTION EXPECTED?
00DA 74 0E                      JZ      TEST_EXC3               ; GO IF YES
00DC                    TEST_EXC1:
00DC E4 80                      IN      AL,MFG_PORT             ; CHECK THE CURRENT CHECKPOINT
00DE 3C 3B                      CMP     AL,03BH                 ;  HALT IF CHECKPOINT BELOW 3BH
00E0 72 01                      JB      TEST_EXC2
00E2 CF                         IRET

00E3                    TEST_EXC2:
00E3 86 E0                      XCHG    AH,AL                   ; OUTPUT THE CURRENT CHECKPOINT
00E5 E6 80                      OUT     MFG_PORT,AL             ;   <<> CHECKPOINT  90 THRU  B5 <<>
00E7 F4                         HLT
00E8 EB F9                      JMP     TEST_EXC2               ; INSURE SYSTEM HALT

00EA                    TEST_EXC3:
00EA 2A C0                      SUB     AL,AL                   ; CLEAR DMA PAGE
00EC E6 8B                      OUT     DMA_PAGE+0AH,AL
00EE B8 0100                    MOV     AX,0100H                ; FOR BOUND INSTRUCTION EXPECTED (INT 5)
00F1 CF                         IRETURN

                        ;-----------------------------------------------------------------------
                        ;       THIS BUILDS THE DESCRIPTOR TABLES REQUIRED FOR PROTECTED MODE   ;
                        ;               PROCESSOR MUST BE IN REAL MODE                          ;
                        ;-----------------------------------------------------------------------
                                ASSUME  CS:CODE,DS:NOTHING,ES:NOTHING,SS:NOTHING

00F2                    SYSINIT1        PROC    NEAR
00F2 FA                         CLI                             ; NO INTERRUPTS ALLOWED
00F3 55                         PUSH    BP                      ; SAVE BP
00F4 B0 81                      MOV     AL,81H                  ;       <<<<<<<<<<<<<<<<
00F6 E6 80                      OUT     MFG_PORT,AL             ;       <<> CHECKPOINT  81 <<>
00F8 E8 0149 R                  CALL    SIDT_BLD
00FB 8B EF                      MOV     BP,DT
                                                                ; SAVE THE POINTER TO JUST PAST THE IDT
                                                                ; AS WE HAVE NO SDA, USE THE SIX BYTES
                                                                ;   HERE TO LOAD THE IDTR.  WE WILL SIDT
                                                                ;   WHEN WE GET TO SDA INITIALIZATION.
00FD B8 0800                    MOV     AX,SYS_IDT_LEN          ; SEGMENT LIMIT = LENGTH OF IDT
0100 AB                         STOSW                           ; STORE THAT AS IDT LIMIT
0101 B8 D0A0                    MOV     AX,SYS_IDT_LOC          ; IDT ADDRESS
0104 AB                         STOSW                           ;   AND ACCESS RIGHTS BYTE (UNDEFINED)
0105 B8 0000                    MOV     AX,0
0108 AB                         STOSW
                                SEGOV   ES                      ; LOAD THE IDT
0109 26                 +       DB      026H
                                LIDT    [BP]                    ;   REGISTER FROM THIS AREA
010A 0F                 +       DB      00FH
010B                    + ??0001        LABEL   BYTE
010B 8B 5E 00           +               MOV     BX,WORD PTR [BP]
010E                    + ??0002        LABEL   BYTE
010B                    +               ORG     OFFSET CS:??0001
010B 01                 +               DB      001H
010E                    +               ORG     OFFSET CS:??0002
010E 8B FD                      MOV     DI,BP                   ; ES:DI NOW --> END OF IDT AGAIN

                        ;-----  BUILD THE GDT.

0110 BF D8A0                    MOV     DI,GDT_LOC
0113 E8 0140 R                  CALL    GDT_BLD
0116 8B EF                      MOV     BP,DI                   ; SAVE THE ES:DI POINTER
0118 B8 0088                    MOV     AX,GDT_LEN              ; AX = LENGTH OF THE GDT
```

```
                                        STOSW                   ; PUT THAT IN THE LIMIT FIELD
                                        MOV     AX,GDT_LOC      ; AX = LOW WORD OF GDT ADDRESS
                                        STOSW                   ; PUT THAT IN BASE FIELD - LOW
                                        MOV     AX,0            ; AX = HIGH BYTE OF ADDRESS, AND
                                        STOSW                   ;   ACCESS RIGHTS BYTE IS UNDEFINED
                                        SEGOV   ES              ; LOAD THE GDTR
                                        DB      026H
                                        LGDT    [BX]            ;   FROM THIS AREA
                                        DB      00FH
                                ??0004  LABEL   BYTE
                                        MOV     DX,WORD PTR [BP]
                                ??0005  LABEL   BYTE
                                        ORG     OFFSET CS:??0004
                                        DB      001H
                                        ORG     OFFSET CS:??0005
                                        MOV     DI,BP           ; RESTORE THE ES:DI POINTER
                                        STOSW
                                        STOSW
                                        MOV     DI,BP

                                ;-----  SWITCH TO VIRTUAL MODE

                                        POP     BP              ; RESTORE BP
                                        MOV     AX,VIRTUAL_ENABLE       ; MACHINE STATUS WORD NEEDED TO
                                        LMSW    AX              ;  SWITCH TO VIRTUAL MODE
                                        DB      00FH,001H,0F0H

                                        DB      0EAH            ; JUMP FAR TO PURGE PRE-FETCH QUEUE
                                        DW      OFFSET DONE     ;   TO OFFSET
                                        DW      SYS_ROM_CS      ;   IN SEGMENT
                                DONE:
                                        MOV     AL,85H          ;       <><><><><><><><><><><>
                                        OUT     MFG_PORT,AL     ;       <><> CHECKPOINT  82 <><>
                                        RET                     ; SYSTEM INITIALIZATION

                                SYSINIT1        ENDP


                                GDT_BLD PROC    NEAR
                                        MOV     SI,OFFSET GDT_DATA_START        ; DS:SI --> GDT
                                        MOV     CX,(OFFSET GDT_DATA_END-OFFSET GDT_DATA_START)/2  ; WORD COUNT
                                        REP     MOVSW                           ; COPY GDT INTO MEMORY
                                        RET
                                GDT_BLD ENDP


                                SIDT_BLD        PROC    NEAR

                                ;-----  BUILD THE IDT.  THE IDT WILL CONTAIN VECTORS FOR EXCEPTION HANDLERS

                                        MOV     SI,OFFSET SYS_IDT_OFFSETS  ; MAKE DS:SI POINT TO
                                        MOV     AX,CS           ;   INTERRUPT ENTRY POINTS
                                        MOV     DS,AX
                                        MOV     DI,SYS_IDT_LOC  ; POINT TO SYS_IDT_LOC
                                        SUB     AX,AX
                                        MOV     ES,AX           ;     WHERE THE IDT WILL BE.
                                        MOV     BX,SYS_ROM_CS   ; CS IS THE SAME FOR ALL INTERRUPTS
                                        MOV     DH,TRAP_GATE    ; ACCESS RIGHTS BYTE FOR THE GATE
                                        MOV     DL,0            ; THE WORD COUNT FIELD IS UNUSED
                                        MOV     CX,32           ; THERE ARE 32 RESERVED INTERRUPTS
                                LOW_IDT:                        ; THIS LOOP BUILDS 32 DESCRIPTORS IN THE
                                                                ;   IDT FOR THE RESERVED INTERRUPTS
                                        MOVSW                   ; GET A ROUTINE ENTRY POINT
                                                                ;   AND PUT IT IN THE OFFSET FIELD
                                        MOV     AX,BX           ; GET THE SYSTEM CODE SEGMENT SELECTOR
                                        STOSW                   ;   AND PUT IT IN THE SELECTOR FIELD
                                        MOV     AX,DX           ; GET THE INTERRUPT GATE BYTE
                                        STOSW                   ;   AND PUT IN THE ACCESS RIGHTS FIELD
                                        MOV     AX,0            ; ZERO OUT
                                        STOSW                   ;   THE RESERVED POSITIONS
                                        LOOP    LOW_IDT         ;     AND REPEAT AS DIRECTED
                                        MOV     CX,256-32       ; 256 TOTAL - 32 DONE = WHATEVER IS LEFT
                                        MOV     BP,OFFSET FREE_INTS     ; THERE IS A COPY OF AN UN-INITIALIZED
                                                                ;   INTERRUPT DESCRIPTOR AT FREE_INTS
                                HIGH_IDT:
                                        MOV     SI,BP           ; DS:SI --> FREE DESCRIPTOR
                                                                ; (ES:DI LEFT OFF AT INT 32)
                                        MOVSW                   ; MOVE OFFSET OF THE IRET INSTRUCTION
                                        MOVSW                   ; MOVE THE CS SELECTOR
                                        MOVSW                   ; MOVE THE ACCESS RIGHTS BYTE
                                        STOSW                   ; ZERO OUT THE RESERVED WORD
                                        LOOP    HIGH_IDT        ; FILL THE REMAINDER OF THE TABLE

                                ;-----  INITIALIZE THE ENTRY POINTS FOR POST TEST

                                        MOV     ES:(SYS_IDT_LOC+(032*DESC_LEN)).ENTRY_POINT),OFFSET SYS_32
                                        MOV     ES:(SYS_IDT_LOC+(033*DESC_LEN)).ENTRY_POINT),OFFSET SYS_33
                                        MOV     ES:(SYS_IDT_LOC+(034*DESC_LEN)).ENTRY_POINT),OFFSET SYS_34
                                        MOV     ES:(SYS_IDT_LOC+(035*DESC_LEN)).ENTRY_POINT),OFFSET SYS_35
                                        MOV     ES:(SYS_IDT_LOC+(036*DESC_LEN)).ENTRY_POINT),OFFSET SYS_36
                                        MOV     ES:(SYS_IDT_LOC+(037*DESC_LEN)).ENTRY_POINT),OFFSET SYS_37
                                        MOV     ES:(SYS_IDT_LOC+(038*DESC_LEN)).ENTRY_POINT),OFFSET SYS_38
                                        RET

                                IRET_ADDR       LABEL   WORD    ; FOR UN-INITIALIZED INTERRUPTS
                                        IRET                    ; NULL RETURN
```

```
                        PAGE
                        ;       THE FOLLOWING DATA DEFINES THE PRE-INITIALIZED GDT FOR POST TESTS.
                        ;       THESE MUST BE INITIALIZED IN THE ORDER IN WHICH THEY APPEAR IN THE
                        ;       GDT_DEF STRUCTURE DEFINITION AS IT IS IN "SYSDATA.INC".

= 01AF                  GDT_DATA_START  EQU     $

                        ;-----  FIRST ENTRY UNUSABLE  - (UNUSED_ENTRY)

01AF 0000                       DW      0                       ; SEGMENT LIMIT
01B1 0000                       DW      0                       ; SEGMENT BASE ADDRESS - LOW WORD
01B3 00                         DB      0                       ; SEGMENT BASE ADDRESS - HIGH BYTE
01B4 00                         DB      0                       ; ACCESS RIGHTS BYTE
01B5 0000                       DW      0                       ; RESERVED - MUST BE ZERO

                        ;-----  THE GDT ITSELF  - (GDT_PTR)

01B7 0088                       DW      GDT_LEN                 ; SEGMENT LIMIT
01B9 D8A0                       DW      GDT_LOC                 ; SEGMENT BASE ADDRESS - LOW WORD
01BB 00                         DB      0                       ; SEGMENT BASE ADDRESS - HIGH BYTE
01BC 93                         DB      CPL0_DATA_ACCESS        ; ACCESS RIGHTS BYTE
01BD 0000                       DW      0                       ; RESERVED - MUST BE ZERO

                        ;-----  THE SYSTEM IDT DESCRIPTOR  - (SYS_IDT_PTR)

01BF 0800                       DW      SYS_IDT_LEN             ; SEGMENT LIMIT
01C1 D0A0                       DW      SYS_IDT_LOC             ; SEGMENT BASE ADDRESS - LOW WORD
01C3 00                         DB      0                       ; SEGMENT BASE ADDRESS - HIGH BYTE
01C4 93                         DB      CPL0_DATA_ACCESS        ; ACCESS RIGHTS BYTE
01C5 0000                       DW      0                       ; RESERVED - MUST BE ZERO

                        ;-----  THE SYSTEM DATA AREA DESCRIPTOR  - (RSDA_PTR)

01C7 0300                       DW      SDA_LEN                 ; SEGMENT LIMIT
01C9 0400                       DW      SDA_LOC                 ; SEGMENT BASE ADDRESS - LOW WORD
01CB 00                         DB      0                       ; SEGMENT BASE ADDRESS - HIGH BYTE
01CC 93                         DB      CPL0_DATA_ACCESS        ; ACCESS RIGHTS BYTE
01CD 0000                       DW      0                       ; RESERVED - MUST BE ZERO

                        ;-----  COMPATIBLE MONOCHROME DISPLAY REGEN BUFFER  - (C_BWCRT_PTR)

01CF 1000                       DW      MCRT_SIZE               ; SEGMENT LIMIT
01D1 0000                       DW      MCRT@_LO                ; SEGMENT BASE ADDRESS - LOW WORD
01D3 0B                         DB      MCRT@_HI                ; SEGMENT BASE ADDRESS - HIGH BYTE
01D4 93                         DB      CPL0_DATA_ACCESS        ; ACCESS RIGHTS BYTE
01D5 0000                       DW      0                       ; RESERVED - MUST BE ZERO

                        ;-----  COMPATIBLE COLOR DISPLAY REGEN BUFFER  - (C_CCRT_PTR)

01D7 4000                       DW      CCRT_SIZE               ; SEGMENT LIMIT
01D9 8000                       DW      CCRT@_LO                ; SEGMENT BASE ADDRESS - LOW WORD
01DB 0B                         DB      CCRT@_HI                ; SEGMENT BASE ADDRESS - HIGH BYTE
01DC 93                         DB      CPL0_DATA_ACCESS        ; ACCESS RIGHTS BYTE
01DD 0000                       DW      0                       ; RESERVED - MUST BE ZERO

                        ;-----  ENHANCED GRAPHIC ADAPTER REGEN BUFFER  - (E_CCRT_PRT)

01DF FFFF                       DW      ECCRT_SIZE              ; SEGMENT LIMIT
01E1 0000                       DW      ECCRT@_LO_LO            ; SEGMENT BASE ADDRESS - LOW WORD
01E3 0A                         DB      ECCRT@_LO_HI            ; SEGMENT BASE ADDRESS - HIGH BYTE
01E4 93                         DB      CPL0_DATA_ACCESS        ; ACCESS RIGHTS BYTE
01E5 0000                       DW      0                       ; RESERVED - MUST BE ZERO

                        ;-----  SECOND PART OF EGA  - (E_CCRT_PTR2)

01E7 FFFF                       DW      ECCRT_SIZE              ; SEGMENT LIMIT
01E9 0000                       DW      ECCRT@_HI_LO            ; SEGMENT BASE ADDRESS - LOW WORD
01EB 0B                         DB      ECCRT@_HI_HI            ; SEGMENT BASE ADDRESS - HIGH BYTE
01EC 93                         DB      CPL0_DATA_ACCESS        ; ACCESS RIGHTS BYTE
01ED 0000                       DW      0                       ; RESERVED - MUST BE ZERO

                        ;-----  CODE SEGMENT FOR POST CODE, SYSTEM IDT  - (SYS_ROM_CS)

01EF FFFF                       DW      MAX_SEG_LEN             ; SEGMENT LIMIT
01F1 0000                       DW      CSEG@_LO                ; SEGMENT BASE ADDRESS - LOW WORD
01F3 0F                         DB      CSEG@_HI                ; SEGMENT BASE ADDRESS - HIGH BYTE
01F4 9B                         DB      CPL0_CODE_ACCESS        ; ACCESS RIGHTS BYTE
01F5 0000                       DW      0                       ; RESERVED - MUST BE ZERO

                        ;-----  TEMPORARY DESCRIPTOR FOR ES  - (ES_TEMP)

01F7 FFFF                       DW      MAX_SEG_LEN             ; SEGMENT LIMIT
01F9 0000                       DW      NSEG@_LO                ; SEGMENT BASE ADDRESS - LOW WORD
01FB 00                         DB      NSEG@_HI                ; SEGMENT BASE ADDRESS - HIGH BYTE
01FC 93                         DB      CPL0_DATA_ACCESS        ; ACCESS RIGHTS BYTE
01FD 0000                       DW      0                       ; RESERVED - MUST BE ZERO

                        ;-----  TEMPORARY DESCRIPTOR FOR CS AS A DATA SEGMENT  - (CS_TEMP)

01FF FFFF                       DW      MAX_SEG_LEN             ; SEGMENT LIMIT
0201 0000                       DW      NSEG@_LO                ; SEGMENT BASE ADDRESS - LOW WORD
0203 00                         DB      NSEG@_HI                ; SEGMENT BASE ADDRESS - HIGH BYTE
0204 93                         DB      CPL0_DATA_ACCESS        ; ACCESS RIGHTS BYTE
0205 0000                       DW      0                       ; RESERVED - MUST BE ZERO

                        ;-----  TEMPORARY DESCRIPTOR FOR SS  - (SS_TEMP)

0207 FFFF                       DW      MAX_SEG_LEN             ; SEGMENT LIMIT
0209 0000                       DW      NSEG@_LO                ; SEGMENT BASE ADDRESS - LOW WORD
020B 00                         DB      NSEG@_HI                ; SEGMENT BASE ADDRESS - HIGH BYTE
020C 93                         DB      CPL0_DATA_ACCESS        ; ACCESS RIGHTS BYTE
020D 0000                       DW      0                       ; RESERVED - MUST BE ZERO

                        ;-----  TEMPORARY DESCRIPTOR FOR DS  - (DS_TEMP)

020F FFFF                       DW      MAX_SEG_LEN             ; SEGMENT LIMIT
0211 0000                       DW      NSEG@_LO                ; SEGMENT BASE ADDRESS - LOW WORD
0213 00                         DB      NSEG@_HI                ; SEGMENT BASE ADDRESS - HIGH BYTE
0214 93                         DB      CPL0_DATA_ACCESS        ; ACCESS RIGHTS BYTE
0215 0000                       DW      0                       ; RESERVED - MUST BE ZERO
```

```
                        PAGE
                        ;-----  (POST_TR)
0217                    TR_LOC:
0217 0800               DW      00800H                  ; SEGMENT LIMIT
0219 C000               DW      0C000H                  ; SEGMENT BASE ADDRESS - LOW WORD
021B 00                 DB      0                       ; SEGMENT BASE ADDRESS - HIGH BYTE
021C 81                 DB      FREE_TSS                ; ACCESS RIGHTS BYTE
021D 0000               DW      0                       ; RESERVED - MUST BE ZERO

                        ;-----  (POST_TSS_PTR)

021F 0800               DW      00800H                  ; SEGMENT LIMIT
0221 0217 R             DW      TR_LOC                  ; SEGMENT BASE ADDRESS - LOW WORD
0223 00                 DB      0                       ; SEGMENT BASE ADDRESS - HIGH BYTE
0224 93                 DB      CPL0_DATA_ACCESS        ; ACCESS RIGHTS BYTE
0225 0000               DW      0                       ; RESERVED - MUST BE ZERO

                        ;-----  (POST_LDTR)
0227                    LDT_LOC:
0227 0088               DW      GDT_LEN                 ; SEGMENT LIMIT
0229 D000               DW      0D000H                  ; SEGMENT BASE ADDRESS - LOW WORD
022B 00                 DB      0                       ; SEGMENT BASE ADDRESS - HIGH BYTE
022C E2                 DB      LDT_DESC                ; ACCESS RIGHTS BYTE
022D 0000               DW      0                       ; RESERVED - MUST BE ZERO

                        ;-----  (POST_LDT_PTR)

022F 0088               DW      GDT_LEN                 ; SEGMENT LIMIT
0231 0227 R             DW      LDT_LOC                 ; SEGMENT BASE ADDRESS - LOW WORD
0233 00                 DB      0                       ; SEGMENT BASE ADDRESS - HIGH BYTE
0234 93                 DB      CPL0_DATA_ACCESS        ; ACCESS RIGHTS BYTE
0235 0000               DW      0                       ; RESERVED - MUST BE ZERO

= 0237                  GDT_DATA_END    EQU     $

                        ;-----  END OF PRE-ALLOCATED GDT


                        ;-----  ENTRY POINTS FOR THE FIRST 32 SYSTEM INTERRUPTS

0237                    SYS_IDT_OFFSETS         LABEL   WORD
                                                        ; INTERRUPTS AS DEFINED
0237 0000 R             DW      OFFSET EXC_00           ; EXCPT 00 - DIVIDE ERROR
0239 0005 R             DW      OFFSET EXC_01           ; EXCPT 01 - SINGLE STEP
023B 000A R             DW      OFFSET EXC_02           ; EXCPT 02 - NMI, SYSTEM REQUEST FOR DI
023D 000F R             DW      OFFSET EXC_03           ; EXCPT 03 - BREAKPOINT
023F 0014 R             DW      OFFSET EXC_04           ; EXCPT 04 - INTO DETECT
0241 0019 R             DW      OFFSET EXC_05           ; EXCPT 05 - BOUND
0243 0030 R             DW      OFFSET EXC_06           ; EXCPT 06 - INVALID OPCODE
0245 0034 R             DW      OFFSET EXC_07           ; EXCPT 07 - PROCESSOR EXT NOT AVAIL
0247 0038 R             DW      OFFSET EXC_08           ; EXCPT 08 - DOUBLE EXCEPTION
0249 003C R             DW      OFFSET EXC_09           ; EXCPT 09 - PROCESSOR EXT SEGMENT ERR
024B 0040 R             DW      OFFSET EXC_10           ; EXCPT 10 - TSS BAD IN GATE TRANSFER
024D 0044 R             DW      OFFSET EXC_11           ; EXCPT 11 - SEGMENT NOT PRESENT
024F 0048 R             DW      OFFSET EXC_12           ; EXCPT 12 - STACK SEGMENT NOT PRESENT
0251 004C R             DW      OFFSET EXC_13           ; EXCPT 13 - GENERAL PROTECTION
0253 0050 R             DW      OFFSET EXC_14
0255 0054 R             DW      OFFSET EXC_15
0257 0058 R             DW      OFFSET EXC_16           ; EXCPT 16 - PROCESSOR EXTENSION ERROR
0259 005C R             DW      OFFSET EXC_17
025B 0060 R             DW      OFFSET EXC_18
025D 0064 R             DW      OFFSET EXC_19
025F 0068 R             DW      OFFSET EXC_20
0261 006C R             DW      OFFSET EXC_21
0263 0070 R             DW      OFFSET EXC_22
0265 0074 R             DW      OFFSET EXC_23
0267 0078 R             DW      OFFSET EXC_24
0269 007C R             DW      OFFSET EXC_25
026B 0080 R             DW      OFFSET EXC_26
026D 0084 R             DW      OFFSET EXC_27
026F 0088 R             DW      OFFSET EXC_28
0271 008C R             DW      OFFSET EXC_29
0273 0090 R             DW      OFFSET EXC_30
0275 0094 R             DW      OFFSET EXC_31

                        ;-----  FORMAT INTERRUPT DESCRIPTORS (GATES) 32 - 255

0277 01AE R             FREE_INTS       DW      OFFSET IRET_ADDR        ; DESTINATION OFFSET
0279 0040                               DW      SYS_ROM_CS              ; DESTINATION SEGMENT
027B 00 86                              DB      0,INT_GATE              ; UNUSED AND ACCESS RIGHTS BYTE
027D                    SIDT_BLD        ENDP

027D                    CODE    ENDS
                                END
```

SECTION 5

```
                        PAGE 118,121
                        TITLE TEST6 ---- 06/10/85  POST TESTS AND SYSTEM BOOT STRAP
                        .286C
                        .LIST
0000                    CODE    SEGMENT BYTE PUBLIC

                                PUBLIC  BOOT_STRAP_1
                                PUBLIC  POST6
                                PUBLIC  STGTST_CNT
                                PUBLIC  ROM_ERR
                                PUBLIC  XMIT_8042

                                EXTRN   CMOS_READ:NEAR
                                EXTRN   DDS:NEAR
                                EXTRN   DISK_BASE:NEAR
                                EXTRN   E602I:NEAR
                                EXTRN   ERR_BEEP:NEAR
                                EXTRN   E_MSG:NEAR
                                EXTRN   F3A:NEAR
                                EXTRN   PRT_SEG:NEAR

                                ASSUME  CS:CODE,DS:DATA

0000                    POST6   PROC    NEAR
                        ;--------------------------------------------------------------------
                        ; THIS SUBROUTINE PERFORMS A READ/WRITE STORAGE TEST ON A BLOCK      :
                        ;       OF STORAGE.                                                 :
                        ; ENTRY REQUIREMENTS:                                               :
                        ;       ES = ADDRESS OF STORAGE SEGMENT BEING TESTED                :
                        ;       DS = ADDRESS OF STORAGE SEGMENT BEING TESTED                :
                        ;       CX = WORD COUNT OF STORAGE BLOCK TO BE TESTED               :
                        ; EXIT PARAMETERS:                                                  :
                        ;       ZERO FLAG = 0 IF STORAGE ERROR (DATA COMPARE OR PARITY      :
                        ;       CHECK). AL=0 DENOTES A PARITY CHECK. ELSE AL=XOR'ED         :
                        ;       BIT PATTERN OF THE EXPECTED DATA PATTERN VS THE ACTUAL      :
                        ;       DATA READ.                                                  :
                        ; AX,BX,CX,DX,DI, AND SI ARE ALL DESTROYED.                         :
                        ;--------------------------------------------------------------------
0000                    STGTST_CNT      PROC    NEAR
0000 8B D9                      MOV     BX,CX                   ; SAVE WORD COUNT OF BLOCK TO TEST
0002 E4 61                      IN      AL,PORT_B
0004 0C 0C                      OR      AL,RAM_PAR_OFF          ; TOGGLE PARITY CHECK LATCHES
0006 E6 61                      OUT     PORT_B,AL               ; TO RESET ANY PENDING ERROR
0008 24 F3                      AND     AL,RAM_PAR_ON
000A E6 61                      OUT     PORT_B,AL

                        ;-----  ROLL A BIT THROUGH THE FIRST WORD

000C 33 D2                      XOR     DX,DX                   ; CLEAR THE INITIAL DATA PATTERN
000E B9 0010                    MOV     CX,16                   ; ROLL 16 BIT POSITIONS
0011 2B FF                      SUB     DI,DI                   ; START AT BEGINNING OF BLOCK
0013 2B F6                      SUB     SI,SI                   ; INITIALIZE DESTINATION POINTER
0015 F9                         STC                             ; SET CARRY FLAG ON FOR FIRST BIT
0016                    C1:
0016 D1 D2                      RCL     DX,1                    ; MOVE BIT OVER LEFT TO NEXT POSITION
0018 89 15                      MOV     [DI],DX                 ; STORE DATA PATTERN
001A 8B 05                      MOV     AX,[DI]                 ; GET THE DATA WRITTEN
001C 33 C2                      XOR     AX,DX                   ; INSURE DATA AS EXPECTED (CLEAR CARRY)
001E E1 F6                      LOOPZ   C1                      ; LOOP TILL DONE OR ERROR

0020 75 66                      JNZ     C13                     ; EXIT IF ERROR

                        ;-----  CHECK CAS LINES FOR HIGH BYTE LOW BYTE

0022 BA FF00                    MOV     DX,0FF00H               ; TEST DATA - AX= 0000H
0025 89 05                      MOV     [DI],AX                 ; STORE DATA PATTERN  = 0000H
0027 88 75 01                   MOV     [DI+1],DH               ; WRITE A BYTE OF FFH AT ODD LOCATION
002A 8B 05                      MOV     AX,[DI]                 ; GET THE DATA - SHOULD BE 0FF00H
002C 33 C2                      XOR     AX,DX                   ; CHECK THE FIRST WRITTEN
002E 75 58                      JNZ     C13                     ; ERROR EXIT IF NOT ZERO

0030 89 05                      MOV     [DI],AX                 ; STORE DATA PATTERN OF 0000H
0032 88 35                      MOV     [DI],DH                 ; WRITE A BYTE OF FFH AT EVEN LOCATION
0034 86 F2                      XCHG    DH,DL                   ; SET DX= 000FFH AND BUS SETTLE
0036 8B 05                      MOV     AX,[DI]                 ; GET THE DATA
0038 33 C2                      XOR     AX,DX                   ; CHECK THE FIRST WRITTEN
003A 75 4C                      JNZ     C13                     ; EXIT IF NOT

                        ;-----  CHECK FOR I/O OR BASE MEMORY ERROR

003C E4 61                      IN      AL,PORT_B               ; CHECK FOR I/O - PARITY CHECK
003E 86 C4                      XCHG    AL,AH                   ; SAVE ERROR
0040 E4 87                      IN      AL,DMA_PAGE+6           ; CHECK FOR R/W OR I/O ERROR
0042 22 E0                      AND     AH,AL                   ; MASK FOR ERROR EXPECTED

                        ;-----  PARITY ERROR EXIT

0044 B8 0000                    MOV     AX,0                    ; RESTORE AX TO 0000
0047 75 3F                      JNZ     C13                     ; EXIT IF PARITY ERROR

0049 BA AA55                    MOV     DX,0AA55H               ; WRITE THE INITIAL DATA PATTERN
004C                    C3:
004C 2B FF                      SUB     DI,DI                   ; START AT BEGINNING OF BLOCK
004E 2B F6                      SUB     SI,SI                   ; INITIALIZE DESTINATION POINTER
0050 8B CB                      MOV     CX,BX                   ; SETUP BYTE COUNT FOR LOOP
0052 8B C2                      MOV     AX,DX                   ; GET THE PATTERN
0054 F3/ AB                     REP     STOSW                   ; STORE 64K BYTES (32K WORDS)
0056 8B CB                      MOV     CX,BX                   ; SET COUNT
0058 2B F6                      SUB     SI,SI                   ; START AT BEGINNING
005A                    C6:
005A AD                         LODSW                           ; GET THE FIRST WRITTEN
005B 33 C2                      XOR     AX,DX                   ; INSURE DATA AS EXPECTED
005D E1 FB                      LOOPZ   C6                      ; LOOP TILL DONE OR ERROR

005F 75 27                      JNZ     C13                     ; EXIT IF NOT EXPECTED (ERROR BITS ON)

                        ;-----  CHECK FOR I/O OR BASE MEMORY ERROR

0061 E4 61                      IN      AL,PORT_B               ; CHECK FOR I/O -PARITY CHECK
0063 86 C4                      XCHG    AL,AH                   ; SAVE ERROR
0065 E4 87                      IN      AL,DMA_PAGE+6           ; CHECK FOR R/W OR I/O ERROR
0067 22 E0                      AND     AH,AL
```

```
                        ;-----  PARITY ERROR EXIT
                                MOV     AX,0                    ; RESTORE AX TO 0000
0069 B8 0000                    JNZ     C13                     ; GO IF YES
006C 75 1A
                        ;-----  CHECK FOR END OF 64K BLOCK
                                AND     DX,DX                   ; ENDING ZERO PATTERN WRITTEN TO MEMORY?
006E 23 D2                      JZ      C13                     ; YES - RETURN TO CALLER WITH AL=0
0070 74 16
                        ;-----  SETUP NEXT PATTERN
                                CMP     DX,055AAH               ; CHECK IF LAST PATTERN =55AA
0072 81 FA 55AA                 JZ      C9                      ; GO IF NOT
0076 74 0B                      CMP     DX,0101H                ; LAST PATTERN 0101?
0078 81 FA 0101                 JZ      C10                     ; GO IF YES
007C 74 0B                      MOV     DX,055AAH               ; WRITE 55AA TO STORAGE
007E BA 55AA                    JMP     C3
0081 EB C9
                        ;-----  INSURE PARITY BITS ARE NOT STUCK ON
                        C9:     MOV     DX,0101H                ; WRITE 0101 TO STORAGE
0083 BA 0101                    JMP     C3
0086 EB C4
                        ;-----  EXIT STORAGE TEST
                        C13:
0088                            RET                             ; ERROR IF ZF NOT SET
0088 C3
                        ;-----  CHECKER BOARD TEST
                        C10:    SUB     DI,DI                   ; POINT TO START OF BLOCK
0089 2B FF                      MOV     CX,BX                   ; GET THE BLOCK COUNT
008B 8B CB                      SHR     CX,1                    ; DIVIDE BY 2
008D D1 E9                      MOV     AX,1010101010101010B    ; SECOND CHECKER PATTERN
008F B8 AAAA                    MOV     SI,0101010101010101B    ; FIRST CHECKER PATTERN
0092 BE 5555
                        C11:
0095                            XCHG    AX,SI                   ; FIRST CHECKER PATTERN TO AX
0095 96                         STOSW                           ; WRITE IT TO MEMORY
0096 AB                         XCHG    AX,SI                   ; SECOND CHECKER PATTERN TO AX
0097 96                         STOSW                           ; WRITE IT TO MEMORY
0098 AB                         LOOP    C11                     ; DO IT FOR CX COUNT
0099 E2 FA
                                SUB     SI,SI                   ; POINT TO START OF BLOCK
009B 2B F6                      MOV     CX,BX                   ; GET THE BLOCK COUNT
009D 8B CB                      SHR     CX,1                    ; DIVIDE BY 2
009F D1 E9                      MOV     DI,0101010101010101B    ; CHECK CORRECT
00A1 BF 5555                    MOV     DX,1010101010101010B
00A4 BA AAAA
00A7                    C12:
00A7 AD                         LODSW                           ; GET THE DATA
00A8 33 C7                      XOR     AX,DI                   ; CHECK CORRECT
00AA 75 DC                      JNZ     C13                     ; EXIT IF NOT

00AC AD                         LODSW                           ; GET NEXT DATA
00AD 33 C2                      XOR     AX,DX                   ; CHECK SECOND PATTERN
00AF E1 F6                      LOOPZ   C12                     ; CONTINUE TILL DONE

00B1 75 D8                      JNZ     C13                     ; ERROR EXIT IF NOT CORRECT
                        ;-----  CHECK FOR I/O OR BASE MEMORY PARITY CHECK
00B3 E4 61                      IN      AL,PORT_B               ; CHECK FOR I/O-PARITY CHECK
00B5 86 C4                      XCHG    AL,AH                   ; SAVE ERROR
00B7 E4 87                      IN      AL,DMA_PAGE+6           ; CHECK FOR R/W OR I/O ERROR
00B9 22 E0                      AND     AH,AL
                        ;-----  CHECKPOINT 32 FOR ADDRESS LINE 0->15 FAILURE
00BB B0 32                      MOV     AL,32H                  ;       <><><><><><><><><><>
00BD E6 80                      OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  32 <><>
00BF B8 0000                    MOV     AX,0                    ; RESTORE AX (SET AX TO ZERO)
00C2 75 C4                      JNZ     C13                     ; EXIT IF PARITY ERROR
                        ;-----  64K ADDRESS TEST AND FILL WITH ZERO
00C4 48                         DEC     AX                      ; WRITE FIRST AND LAST LOCATION=FFFF
00C5 2B FF                      SUB     DI,DI                   ; POINT TO START OF BLOCK
00C7 8B CB                      MOV     CX,BX                   ; GET THE BLOCK COUNT
00C9 83 E9 02                   SUB     CX,2                    ; DO ALL LOCATIONS BUT LAST
00CC AB                         STOSW                           ; WRITE FIRST LOCATION AS FFFFH
00CD 40                         INC     AX                      ; WRITE ZERO
00CE F3/ AB                     REP     STOSW                   ; WRITE IT
00D0 48                         DEC     AX                      ; LAST WORD IS FFFF
00D1 AB                         STOSW
00D2 2B F6                      SUB     SI,SI                   ; POINT TO START OF BLOCK
00D4 8B CB                      MOV     CX,BX                   ; GET THE BLOCK COUNT
00D6 83 E9 02                   SUB     CX,2
00D9 AD                         LODSW                           ; GET THE DATA
00DA 35 FFFF                    XOR     AX,0FFFFH               ; CHECK CORRECT
00DD 75 A9                      JNZ     C13                     ; EXIT IF NOT
00DF                    C12A:
00DF AD                         LODSW                           ; GET NEXT DATA
00E0 0B C0                      OR      AX,AX                   ; ANY BIT ON ?
00E2 E1 FB                      LOOPZ   C12A                    ; CONTINUE TILL LAST WORD
00E4 75 A2                      JNZ     C13                     ; GO IF NOT CORRECT
00E6 AD                         LODSW                           ; GET LAST WORD
00E7 35 FFFF                    XOR     AX,0FFFFH               ; S/B FFFF
00EA 75 9C                      JNZ     C13                     ; EXIT IF NOT

                        ;-----  CLEAR WORD 0 AND FFFE
00EC 2B FF                      SUB     DI,DI                   ; CLEAR FIRST WORD
00EE AB                         STOSW
00EF BF FFFE                    MOV     DI,0FFFEH               ; CLEAR TOP WORD
00F2 AB                         STOSW

                        ;-----  CHECK FOR I/O OR BASE MEMORY
00F3 E4 61                      IN      AL,PORT_B               ; CHECK FOR I/O - PARITY CHECK
00F5 86 C4                      XCHG    AL,AH                   ; SAVE ERROR
00F7 E4 87                      IN      AL,DMA_PAGE+6           ; CHECK FOR R/W OR I/O ERROR
00F9 22 E0                      AND     AH,AL
00FB B8 0000                    MOV     AX,0                    ; SET AX EQUAL ZERO
00FE EB 88                      JMP     C13                     ; ERROR EXIT IF ZF NOT SET
0100                    STGTST_CNT      ENDP
```

```
                        PAGE
                        ;--------------------------------------------------------
                        ;  PRINT ADDRESS AND ERROR MESSAGE FOR ROM CHECKSUM ERRORS      :
                        ;--------------------------------------------------------
0100                    ROM_ERR PROC    NEAR
0100 52                         PUSH    DX                      ; SAVE POINTER
0101 06                         PUSH    ES
0102 50                         PUSH    AX
0103 B8 ---- R                  MOV     AX,DATA                 ; SET ES TO DATA SEGMENT
0106 8E C0                      MOV     ES,AX
0108 58                         POP     AX                      ; RESTORE AX
0109 50                         PUSH    AX
010A 8C DA                      MOV     DX,DS                   ; GET ADDRESS POINTER
010C 26: 88 36 0018 R           MOV     ES:@MFG_ERR_FLAG,DH     ;       <><><><><><><><><><><><><><>
                                                                ;               <><> CHECKPOINTS C0->F4 <><>
0111 81 FA C800                 CMP     DX,0C800H               ; DISPLAY CARD IN ERROR?
0115 7C 0D                      JL      ROM_ERR_BEEP            ; GIVE DISPLAY CARD FAIL BEEP
0117 E8 0000 E                  CALL    PRT_SEG                 ; PRINT SEGMENT IN ERROR
011A BE 0000 E                  MOV     SI,OFFSET F3A           ; DISPLAY ERROR MESSAGE
011D E8 0000 E                  CALL    E_MSG
0120                    ROM_ERR_END:
0120 58                         POP     AX
0121 07                         POP     ES
0122 5A                         POP     DX
0123 C3                         RET
0124                    ROM_ERR_BEEP:
0124 BA 0102                    MOV     DX,0102H                ; BEEP 1 LONG, 2 SHORT
0127 E8 0000 E                  CALL    ERR_BEEP
012A EB F4                      JMP     SHORT ROM_ERR_END
012C                    ROM_ERR ENDP
                        ;--------------------------------------------------------------------
                        ; THIS SUBROUTINE SENDS AN OUTPUT COMMAND TO THE KEYBOARD AND       :
                        ;       RECEIVES THE KEYBOARD RESPONSE.                            :
                        ; ENTRY REQUIREMENTS:                                              :
                        ;       AL = COMMAND/DATA TO BE SENT                               :
                        ; EXIT PARAMETERS:                                                 :
                        ;       ZERO FLAG = 1 IF ACK RECEIVED FROM THE KEY BOARD           :
                        ;       AL = RESPONSE                                              :
                        ;--------------------------------------------------------------------
012C                    XMIT_8042 PROC  NEAR

                        ;-----  CHECK INPUT BUFFER FULL

012C 86 E0                      XCHG    AH,AL                   ; SAVE COMMAND
012E 2B C9                      SUB     CX,CX                   ; SET LOOP TIME-OUT
0130                    XMITLOOP:
0130 E4 64                      IN      AL,STATUS_PORT
0132 A8 02                      TEST    AL,INPT_BUF_FULL        ; CHECK INPUT BUFFER FULL
0134 E0 FA                      LOOPNZ  XMITLOOP
0136 E3 34                      JCXZ    SHORT XMIT_EXIT
0138 86 E0                      XCHG    AH,AL                   ; RESTORE COMMAND

                        ;-----  ISSUE THE COMMAND

013A E6 60                      OUT     PORT_A,AL               ; SEND THE COMMAND
013C 2B C9                      SUB     CX,CX                   ; SET LOOP COUNT

                        ;-----  CHECK OUTPUT BUFFER FULL

013E E4 64              XMIT_1: IN      AL,STATUS_PORT
0140 8A E0                      MOV     AH,AL                   ; SAVE STATUS
0142 A8 01                      TEST    AL,OUT_BUF_FULL         ; CHECK IF 8042 HAS DATA
0144 74 02                      JZ      XMIT_2                  ; GO IF NOT
0146 E4 60                      IN      AL,PORT_A               ; FLUSH DATA
0148 F6 C4 02           XMIT_2: TEST    AH,INPT_BUF_FULL        ; CHECK COMMAND ACCEPTED
014B E0 F1                      LOOPNZ  XMIT_1
014D 75 1D                      JNZ     SHORT XMIT_EXIT         ; NO FLUSH OR COMMAND NOT ACCEPTED

                        ;-----  CHECK OUTPUT BUFFER FULL

014F B3 06                      MOV     BL,6                    ; SET COUNT
0151 2B C9                      SUB     CX,CX                   ; SET LOOP COUNT
0153 E4 64              XMIT_3: IN      AL,STATUS_PORT
0155 A8 01                      TEST    AL,OUT_BUF_FULL         ; CHECK IF HAS DATA
0157 E1 FA                      LOOPZ   XMIT_3                  ; WAIT TILL DONE
0159 75 08                      JNZ     XMIT_4
015B FE CB                      DEC     BL                      ; DECREMENT OUTER LOOP
015D 75 F4                      JNZ     SHORT XMIT_3            ; TRY AGAIN
015F FE C3                      INC     BL                      ; SET ERROR FLAG
0161 EB 09                      JMP     SHORT XMIT_EXIT         ; 8042 STUCK BUSY

                        ;-----  GET THE DATA

0163 2B C9              XMIT_4: SUB     CX,CX                   ; ALLOW TIME FOR POSSIBLE
                                                                ; ERROR -> SYSTEM UNIT OR KEYBOARD
0165 E2 FE              XMIT_5: LOOP    XMIT_5
0167 E4 60                      IN      AL,PORT_A
0169 83 E9 01                   SUB     CX,01H                  ; SET CX OTHER THAN ZERO
016C                    XMIT_EXIT:
016C C3                         RET
016D                    XMIT_8042 ENDP

                        ;--- BOOT_STRAP -- INT  19 H ----------------------------
                        ; BOOT STRAP LOADER                                        :
                        ;       TRACK 0, SECTOR 1 IS READ INTO THE                 :
                        ;       BOOT LOCATION (SEGMENT 0 OFFSET 7C00)              :
                        ;       AND CONTROL IS TRANSFERRED THERE.                  :
                        ;                                                          :
                        ;       IF THERE IS A HARDWARE ERROR CONTROL IS            :
                        ;       TRANSFERRED TO THE ROM BASIC ENTRY POINT           :
                        ;----------------------------------------------------------
                                ASSUME CS:CODE,DS:ABS0,ES:ABS0

016D                    BOOT_STRAP_1    PROC    NEAR

016D B8 ---- R                  MOV     AX,ABS0                 ; ESTABLISH ADDRESSING
0170 8E D8                      MOV     DS,AX
0172 8E C0                      MOV     ES,AX

                        ;-----  RESET THE DISK PARAMETER TABLE VECTOR

0174 C7 06 0078 R 0000 E        MOV     WORD PTR @DISK_POINTER, OFFSET DISK_BASE
017A 8C 0E 007A R               MOV     WORD PTR @DISK_POINTER+2,CS
```

```
                        ;-----  CLEAR @BOOT_LOCN

                                XOR     AX,AX
                                MOV     CX,256                  ; CLEAR 256 WORDS
                                MOV     DI,OFFSET @BOOT_LOCN
                                REP     STOSW

                        ;-----  LOAD SYSTEM FROM DISKETTE -- CX HAS RETRY COUNT

                                STI
                                MOV     CX,4                    ; SET RETRY COUNT
                        H1:     PUSH    CX                      ; IPL SYSTEM
                                MOV     AH,0                    ; RESET THE DISKETTE SYSTEM
                                INT     13H                     ; DISKETTE_IO
                                JC      H2                      ; IF ERROR, TRY AGAIN

                                MOV     AX,201H                 ; READ IN THE SINGLE SECTOR
                                SUB     DX,DX                   ; TO THE BOOT LOCATION
                                MOV     ES,DX
                                MOV     BX,OFFSET @BOOT_LOCN    ; DRIVE 0, HEAD 0
                                MOV     CX,1                    ; SECTOR 1, TRACK 0
                                INT     13H                     ; DISKETTE IO
                        H2:     POP     CX                      ; RECOVER RETRY COUNT
                                JNC     H4                      ; CARRY FLAG SET BY UNSUCCESSFUL READ
                                CMP     AH,80H                  ; IF TIME OUT, NO RETRY
                                JZ      H5                      ; TRY FIXED DISK
                                LOOP    H1                      ; DO IT FOR RETRY TIMES
                                JMP     SHORT H5                ; TRY FIXED DISK

                        ;-----  BOOT RECORD READ SUCCESSFUL
                        ;-----  INSURE FIRST BYTE OF LOADED BOOT RECORD IS VALID (NOT ZERO)

01AE 80 3E 7C00 R 06    H4:     CMP     BYTE PTR @BOOT_LOCN,06H ; CHECK FOR FIRST INSTRUCTION INVALID
01B3 72 T1                      JB      H10                     ; IF BOOT NOT VALID PRINT MESSAGE HALT

                        ;-----  INSURE DATA PATTERN FIRST 8 WORDS NOT ALL EQUAL

01B5 BF 7C00 R                  MOV     DI,OFFSET @BOOT_LOCN    ; CHECK DATA PATTERN
01B8 B9 0008                    MOV     CX,8                    ; CHECK THE NEXT 8 WORDS
01BB A1 7C00 R                  MOV     AX,WORD PTR @BOOT_LOCN

01BE 83 C7 02           H4A:    ADD     DI,2                    ; POINT TO NEXT LOCATION
01C1 3B 05                      CMP     AX,[DI]                 ; CHECK DATA PATTERN FOR A FILL PATTERN
01C3 E1 F9                      LOOPZ   H4A
01C5 74 5F                      JZ      H10                     ; BOOT NOT VALID PRINT MESSAGE HALT

01C7 EA 7C00 ---- R     H4_A:   JMP     @BOOT_LOCN

                        ;-----  ATTEMPT BOOTSTRAP FROM FIXED DISK

01CC B0 44              H5:     MOV     AL,044H                 ;       <><><><><><><><><>
01CE E6 80                      OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  44 <><>
                                ASSUME  DS:DATA
01D0 E8 0000 E                  CALL    DDS
01D3 F6 06 008B R 01            TEST    @LASTRATE,DUAL          ; FLOPPY/FIXED DISK CARD INSTALLED
                                ASSUME  DS:ABS0
01D8 B8 ---- R                  MOV     AX,ABS0                 ; ESTABLISH ADDRESSING
01DB 8E D8                      MOV     DS,AX
01DD 74 3D                      JZ      H9                      ; GO IF NOT

                        ;-----  CHECK FOR FIXED DISK INITIALIZATION ERROR

01DF B0 0E                      MOV     AL,CMOS_DIAG            ; GET POST POWER ON STATUS (NMI ENABLED)
01E1 E8 0000 E                  CALL    CMOS_READ               ; FROM DIAGNOSTIC STATUS BYTE
01E4 A8 08                      TEST    AL,HF_FAIL              ; DID WE HAVE A FIXED DISK FAILURE?
01E6 75 34                      JNZ     H9                      ; GO IF YES

01E8 2B C0                      SUB     AX,AX                   ; RESET DISKETTE
01EA 2B D2                      SUB     DX,DX
01EC CD 13                      INT     13H
01EE B9 0003                    MOV     CX,3                    ; RETRY COUNT
01F1                    H6:
01F1 51                         PUSH    CX                      ; SAVE RETRY COUNT
01F2 BA 0080                    MOV     DX,0080H                ; FIXED DISK ZERO
01F5 B8 0201                    MOV     AX,0201H                ; READ IN A SINGLE SECTOR
01F8 2B DB                      SUB     BX,BX
01FA 8E C3                      MOV     ES,BX
01FC BB 7C00 R                  MOV     BX,OFFSET @BOOT_LOCN    ; TO THE BOOT LOCATION
01FF B9 0001                    MOV     CX,1                    ; SECTOR 1, TRACK 0
0202 CD 13                      INT     13H                     ; FILE I/O CALL
0204 59                         POP     CX                      ; RECOVER RETRY COUNT
0205 72 08                      JC      H8
0207 81 3E 7DFE R AA55          CMP     WORD PTR @BOOT_LOCN+510D,0AA55H ; TEST FOR GENERIC BOOT BLOCK
020D 74 B8                      JZ      H4_A

020F 51                 H8:     PUSH    CX
0210 BA 0080                    MOV     DX,0080H                ; FIXED DISK ZERO
0213 2B C0                      SUB     AX,AX                   ; RESET THE FIXED DISK
0215 CD 13                      INT     13H                     ; FILE I/O CALL
0217 59                         POP     CX                      ; RESTORE LOOP COUNT
0218 72 08                      JC      H10A                    ; IF ERROR, TRY AGAIN
021A E2 D5                      LOOP    H6                      ; DO IT FOR RETRY TIMES

                        ;-----  UNABLE TO IPL FROM THE DISKETTE OR FIXED DISK

021C B0 45              H9:     MOV     AL,045H                 ;       <><><><><><><><><>
021E E6 80                      OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  45 <><>

0220 CD 18                      INT     18H                     ; GO TO RESIDENT BASIC

                        ;-----  HARD FILE RESET FAILURE

0222 E2 EB              H10A:   LOOP    H8                      ; TRY RESET AGAIN
0224 EB F6                      JMP     H9                      ; GO TO RESIDENT BASIC

                        ;-----  IF DISKETTE READ OK BUT BOOT RECORD IS NOT STOP SYSTEM ALLOW SOFT RESET

0226 BE 0000 E          H10:    MOV     SI,OFFSET E602          ; PRINT DISKETTE BOOT
0229 E8 0000 E                  CALL    E_MSG                   ; PRINT MESSAGE
022C EB FE              H11:    JMP     H11
                        BOOT_STRAP      ENDP
022E                    POST6   ENDP
022E                    CODE    ENDS
022E                            END
```

SECTION 5

```
                        PAGE 118,121
                        TITLE DSKETTE -- 04/21/86  DISKETTE BIOS
                        .286C
                        .LIST
                        SUBTTL (DSK1.ASM)
                        .LIST
                        ;-- INT 13 H ----------------------------------------------------------------------
                        ; DISKETTE I/O
                        ;       THIS INTERFACE PROVIDES DISK ACCESS TO THE 5.25 INCH 360 KB,
                        ;       1.2 MB, 720 KB, AND 1.44 MB DISKETTE DRIVES.
                        ; INPUT
                        ;       (AH)=00H  RESET DISKETTE SYSTEM
                        ;                 HARD RESET TO NEC, PREPARE COMMAND, RECALIBRATE REQUIRED
                        ;                 ON ALL DRIVES
                        ;-------------------------------------------------------------------------------
                        ;       (AH)=01H  READ THE STATUS OF THE SYSTEM INTO (AH)
                        ;                 @DISKETTE_STATUS FROM LAST OPERATION IS USED
                        ;-------------------------------------------------------------------------------
                        ;       REGISTERS FOR READ/WRITE/VERIFY/FORMAT
                        ;       (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)
                        ;       (DH) - HEAD NUMBER (0-1 ALLOWED, NOT VALUE CHECKED)
                        ;       (CH) - TRACK NUMBER (NOT VALUE CHECKED)
                        ;                 MEDIA       DRIVE           TRACK NUMBER
                        ;               320/360     320/360             0-39
                        ;               320/360     1.2M                0-39
                        ;                1.2M       1.2M                0-79
                        ;               720K        720K                0-79
                        ;               1.44M       1.44M               0-79
                        ;       (CL) - SECTOR NUMBER (NOT VALUE CHECKED, NOT USED FOR FORMAT)
                        ;                 MEDIA       DRIVE           SECTOR NUMBER
                        ;               320/360     320/360             1-8/9
                        ;               320/360     1.2M                1-8/9
                        ;                1.2M       1.2M                1-15
                        ;               720K        720K                1-9
                        ;               1.44M       1.44M               1-18
                        ;       (AL) - NUMBER OF SECTORS (NOT VALUE CHECKED, NOT USED FOR FORMAT)
                        ;                 MEDIA       DRIVE           MAX NUMBER OF SECTORS
                        ;               320/360     320/360             8/9
                        ;               320/360     1.2M                8/9
                        ;                1.2M       1.2M                15
                        ;               720K        720K                9
                        ;               1.44M       1.44M               18
                        ;
                        ;       (ES:BX) - ADDRESS OF BUFFER (NOT REQUIRED FOR VERIFY)
                        ;
                        ;-------------------------------------------------------------------------------
                        ;       (AH)=02H  READ THE DESIRED SECTORS INTO MEMORY
                        ;-------------------------------------------------------------------------------
                        ;       (AH)=03H  WRITE THE DESIRED SECTORS FROM MEMORY
                        ;-------------------------------------------------------------------------------
                        ;       (AH)=04H  VERIFY THE DESIRED SECTORS
                        ;-------------------------------------------------------------------------------
                        ;       (AH)=05H  FORMAT THE DESIRED TRACK
                        ;                 (ES:BX) MUST POINT TO THE COLLECTION OF DESIRED ADDRESS FIELDS
                        ;                 FOR THE TRACK.  EACH FIELD IS COMPOSED OF 4 BYTES, (C,H,R,N),
                        ;                 WHERE C = TRACK NUMBER, H=HEAD NUMBER, R = SECTOR NUMBER,
                        ;                 N= NUMBER OF BYTES PER SECTOR (00=128,01=256,02=512,03=1024).
                        ;                 THERE MUST BE ONE ENTRY FOR EVERY SECTOR ON THE TRACK.
                        ;                 THIS INFORMATION IS USED TO FIND THE REQUESTED SECTOR DURING
                        ;                 READ/WRITE ACCESS.
                        ;
                        ;                 PRIOR TO FORMATTING A DISKETTE, IF THERE EXISTS MORE THAN
                        ;                 ONE SUPPORTED MEDIA FORMAT TYPE WITHIN THE DRIVE IN QUESTION,
                        ;                 THEN "SET DASD TYPE" (INT 13H, AH = 17H) OR "SET MEDIA TYPE"
                        ;                 (INT 13H, AH = 18H) MUST BE CALLED TO SET THE DISKETTE TYPE
                        ;                 THAT IS TO BE FORMATED.  IF "SET DASD TYPE" OR "SET MEDIA TYPE"
                        ;                 IS NOT CALLED, THE FORMAT ROUTINE WILL ASSUME THE MEDIA FORMAT
                        ;                 TO BE THE MAXIMUM CAPACITY OF THE DRIVE.
                        ;
                        ;                 THESE PARAMETERS OF DISK_BASE MUST BE CHANGED IN ORDER TO
                        ;                 FORMAT THE FOLLOWING MEDIAS:
                        ;                 ---------------------------------------------------
                        ;                 : MEDIA  :      DRIVE       : PARM 1 : PARM 2 :
                        ;                 ---------------------------------------------------
                        ;                 : 320K   : 320K/360K/1.2M : 50H    :   8    :
                        ;                 : 360K   : 320K/360K/1.2M : 50H    :   9    :
                        ;                 : 1.2M   : 1.2M           : 54H    :  15    :
                        ;                 : 720K   : 720K/1.44M     : 50H    :   9    :
                        ;                 : 1.44M  : 1.44M          : 6CH    :  18    :
                        ;                 ---------------------------------------------------
                        ;                 NOTES: - PARM 1 = GAP LENGTH FOR FORMAT
                        ;                        - PARM 2 = EOT (LAST SECTOR ON TRACK)
                        ;                        - DISK_BASE IS POINTED TO BY DISK POINTER LOCATED
                        ;                          AT ABSOLUTE ADDRESS 0:78.
                        ;                        - WHEN FORMAT OPERATIONS ARE COMPLETE, THE PARAMETERS
                        ;                          SHOULD BE RESTORED TO THEIR RESPECTIVE INITIAL VALUES
                        ;
                        ;-------------------------------------------------------------------------------
                        ;       (AH)=08H  READ DRIVE PARAMETERS
                        ;       REGISTERS
                        ;         INPUT
                        ;           (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)
                        ;         OUTPUT
                        ;           (ES:DI) POINTS TO DRIVE PARAMETERS TABLE
                        ;           (CH) - LOW ORDER 8 OF 10 BITS MAXIMUM NUMBER OF TRACKS
                        ;           (CL) - BITS 7 & 6 - HIGH ORDER TWO BITS OF MAXIMUM TRACKS
                        ;                - BITS 5 THRU 0 - MAXIMUM SECTORS PER TRACK
                        ;           (DH) - MAXIMUM HEAD NUMBER
                        ;           (DL) - NUMBER OF DISKETTE DRIVES INSTALLED
                        ;           (BH) - 0
                        ;           (BL) - BITS 7 THRU 4 - 0
                        ;                  BITS 3 THRU 0 - VALID DRIVE TYPE VALUE IN CMOS
                        ;           (AX) - 0
                        ;          UNDER THE FOLLOWING CIRCUMSTANCES:
                        ;           (1) THE DRIVE NUMBER IS INVALID,
                        ;           (2) THE DRIVE TYPE IS UNKNOWN AND CMOS IS NOT PRESENT,
                        ;           (3) THE DRIVE TYPE IS UNKNOWN AND CMOS IS BAD,
                        ;           (4) OR THE DRIVE TYPE IS UNKNOWN AND THE CMOS DRIVE TYPE IS INVALID
                        ;          THEN ES,AX,BX,CX,DH,DI=0 ; DL=NUMBER OF DRIVES.
                        ;          IF NO DRIVES ARE PRESENT THEN: ES,AX,BX,CX,DX,DI=0.
                        ;          @DISKETTE_STATUS = 0 AND CY IS RESET.
                        ;-------------------------------------------------------------------------------
                        ;       (AH)=15H READ DASD TYPE
                        ;       OUTPUT REGISTERS
```

```
;       (AH) - ON RETURN IF CARRY FLAG NOT SET, OTHERWISE ERROR
;               00 - DRIVE NOT PRESENT
;               01 - DISKETTE, NO CHANGE LINE AVAILABLE
;               02 - DISKETTE, CHANGE LINE AVAILABLE
;               03 - RESERVED
;       (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)
;
;-----------------------------------------------------------------------
;       (AH)=16H DISK CHANGE LINE STATUS
;       OUTPUT REGISTERS
;       (AH) - 00 - DISK CHANGE LINE NOT ACTIVE
;              06 - DISK CHANGE LINE ACTIVE & CARRY BIT ON
;       (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)
;
;-----------------------------------------------------------------------
;       (AH)=17H SET DASD TYPE FOR FORMAT
;       INPUT REGISTERS
;       (AL) -  00 - NOT USED
;               01 - DISKETTE 320/360K IN 360K DRIVE
;               02 - DISKETTE 360K IN 1.2M DRIVE
;               03 - DISKETTE 1.2M IN 1.2M DRIVE
;               04 - DISKETTE 720K IN 720K DRIVE
;       (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED;
;               DO NOT USE WHEN DISKETTE ATTACH CARD USED)
;-----------------------------------------------------------------------
;       (AH)=18H SET MEDIA TYPE FOR FORMAT
;       INPUT REGISTERS
;       (CH) - LOW ORDER 8 OF 10 BITS MAXIMUM NUMBER OF TRACKS
;       (CL) - BITS 7 & 6 - HIGH ORDER TWO BITS OF MAXIMUM TRACKS
;            - BITS 5 THRU 0 - MAXIMUM SECTORS PER TRACK
;       (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)
;       OUTPUT REGISTERS
;       (ES:DI) - POINTER TO DRIVE PARAMETERS TABLE FOR THIS MEDIA TYPE,
;                 UNCHANGED IF (AH) IS NON-ZERO
;       (AH) - 00H, CY = 0, TRACK AND SECTORS/TRACK COMBINATION IS SUPPORTED
;            - 01H, CY = 1, FUNCTION IS NOT AVAILABLE
;            - 0CH, CY = 1, TRACK AND SECTORS/TRACK COMBINATION IS NOT SUPPORTED
;                           OR DRIVE TYPE UNKNOWN
;            - 80H, CY = 1, TIME OUT (DISKETTE NOT PRESENT)
;-----------------------------------------------------------------------
;       DISK CHANGE STATUS IS ONLY CHECKED WHEN A MEDIA SPECIFIED IS OTHER
;       THAN 360 KB DRIVE.  IF THE DISK CHANGE LINE IS FOUND TO BE
;       ACTIVE THE FOLLOWING ACTIONS TAKE PLACE:
;               ATTEMPT TO RESET DISK CHANGE LINE TO INACTIVE STATE.
;               IF ATTEMPT SUCCEEDS SET DASD TYPE FOR FORMAT AND RETURN DISK
;               CHANGE ERROR CODE
;               IF ATTEMPT FAILS RETURN TIMEOUT ERROR CODE AND SET DASD TYPE
;               TO A PREDETERMINED STATE INDICATING MEDIA TYPE UNKNOWN.
;       IF THE DISK CHANGE LINE IN INACTIVE PERFORM SET DASD TYPE FOR FORMAT.
;
; DATA VARIABLE -- @DISK_POINTER
;       DOUBLE WORD POINTER TO THE CURRENT SET OF DISKETTE PARAMETERS
;-----------------------------------------------------------------------
; OUTPUT FOR ALL FUNCTIONS
;       AH = STATUS OF OPERATION
;               STATUS BITS ARE DEFINED IN THE EQUATES FOR @DISKETTE_STATUS
;               VARIABLE IN THE DATA SEGMENT OF THIS MODULE
;       CY = 0  SUCCESSFUL OPERATION (AH=0 ON RETURN, EXCEPT FOR READ DASD
;               TYPE AH=(15)).
;       CY = 1  FAILED OPERATION (AH HAS ERROR REASON)
;       FOR READ/WRITE/VERIFY
;               AL = COUNT OF SECTORS TRANSFERRED
;               DS,BX,DX,CX PRESERVED
;       NOTE: IF AN ERROR IS REPORTED BY THE DISKETTE CODE, THE APPROPRIATE
;               ACTION IS TO RESET THE DISKETTE, THEN RETRY THE OPERATION.
;               ON READ ACCESSES, NO MOTOR START DELAY IS TAKEN, SO THAT
;               THREE RETRIES ARE REQUIRED ON READS TO ENSURE THAT THE
;               PROBLEM IS NOT DUE TO MOTOR START-UP.
;-----------------------------------------------------------------------
.LIST
; DISKETTE STATE MACHINE - ABSOLUTE ADDRESS 40:90 (DRIVE A) & 91 (DRIVE B)
.LIST
;     -------------------------------------------------
;     I     I     I     I     I     I     I     I     I
;     I  7  I  6  I  5  I  4  I  3  I  2  I  1  I  0  I
;     I     I     I     I     I     I     I     I     I
;     -------------------------------------------------
;        I     I     I     I     I     I     I     I
;        I     I     I     I     I     -------------
;        I     I     I     I     I           I
;        I     I     I     I  RESERVED       I
;        I     I     I     I              PRESENT STATE
;        I     I     I     I        000: 360K IN 360K DRIVE UNESTABLISHED
;        I     I     I     I        001: 360K IN 1.2M DRIVE UNESTABLISHED
;        I     I     I     I        010: 1.2M IN 1.2M DRIVE UNESTABLISHED
;        I     I     I     I        011: 360K IN 360K DRIVE ESTABLISHED
;        I     I     I     I        100: 360K IN 1.2M DRIVE ESTABLISHED
;        I     I     I     I        101: 1.2M IN 1.2M DRIVE ESTABLISHED
;        I     I     I     I        110: RESERVED
;        I     I     I     I        111: NONE OF THE ABOVE
;        I     I     I     I
;        I     I     I     ------> MEDIA/DRIVE ESTABLISHED
;        I     I     I
;        I     I     ------------> DOUBLE STEPPING REQUIRED (360K IN 1.2M
;        I     I                   DRIVE)
;        I     I
;        ------------------------> DATA TRANSFER RATE FOR THIS DRIVE:
;
;                                        00: 500 KBS
;                                        01: 300 KBS
;                                        10: 250 KBS
;                                        11: RESERVED
;
.LIST
;-----------------------------------------------------------------------
; STATE OPERATION STARTED - ABSOLUTE ADDRESS 40:92 (DRIVE A) & 93 (DRIVE B)
;-----------------------------------------------------------------------
; PRESENT CYLINDER NUMBER - ABSOLUTE ADDRESS 40:94 (DRIVE A) & 95 (DRIVE B)
;-----------------------------------------------------------------------
SUBTTL (DSK2.ASM)
```

SECTION 5

```
                        PAGE

                        MD_STRUC        STRUC
0000 ??                 MD_SPEC1        DB      ?       ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
0001 ??                 MD_SPEC2        DB      ?       ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
0002 ??                 MD_OFF_TIM      DB      ?       ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
0003 ??                 MD_BYT_SEC      DB      ?       ; 512 BYTES/SECTOR
0004 ??                 MD_SEC_TRK      DB      ?       ; EOT ( LAST SECTOR ON TRACK)
0005 ??                 MD_GAP          DB      ?       ; GAP LENGTH
0006 ??                 MD_DTL          DB      ?       ; DTL
0007 ??                 MD_GAP3         DB      ?       ; GAP LENGTH FOR FORMAT
0008 ??                 MD_FIL_BYT      DB      ?       ; FILL BYTE FOR FORMAT
0009 ??                 MD_HD_TIM       DB      ?       ; HEAD SETTLE TIME (MILLISECONDS)
000A ??                 MD_STR_TIM      DB      ?       ; MOTOR START TIME (1/8 SECONDS)
000B ??                 MD_MAX_TRK      DB      ?       ; MAX. TRACK NUMBER
000C ??                 MD_RATE         DB      ?       ; DATA TRANSFER RATE
000D                    MD_STRUC        ENDS

= 007F                  BIT7OFF         EQU     7FH
= 0080                  BIT7ON          EQU     80H

                                PUBLIC  DISK_INT_1
                                PUBLIC  SEEK
                                PUBLIC  DSKETTE_SETUP
                                PUBLIC  DISKETTE_IO_1

                                EXTRN   CMOS_READ:NEAR
                                EXTRN   DDS:NEAR
                                EXTRN   DISK_BASE:NEAR
                                EXTRN   WAITF:NEAR

0000                    CODE    SEGMENT BYTE PUBLIC

                                ASSUME  CS:CODE,DS:DATA,ES:DATA

                        ;--------------------------------------------------------
                        ;       DRIVE TYPE TABLE                                :
                        ;--------------------------------------------------------
0000 01                 DR_TYPE         DB      01              ; DRIVE TYPE, MEDIA TABLE
0001 0012 R                             DW      OFFSET MD_TBL1
0003 82                                 DB      02+BIT7ON
0004 001F R                             DW      OFFSET MD_TBL2
0006 02                 DR_DEFAULT      DB      02
0007 002C R                             DW      OFFSET MD_TBL3
0009 03                                 DB      03
000A 0039 R                             DW      OFFSET MD_TBL4
000C 84                                 DB      04+BIT7ON
000D 0046 R                             DW      OFFSET MD_TBL5
000F 04                                 DB      04
0010 0053 R                             DW      OFFSET MD_TBL6
= 0012                  DR_TYPE_E       =$                      ; END OF TABLE
= 0006                  DR_CNT          EQU     (DR_TYPE_E-DR_TYPE)/3

                        ;--------------------------------------------------------
                        ;       MEDIA/DRIVE PARAMETER TABLES                    :
                        ;--------------------------------------------------------

                        ;--------------------------------------------------------
                        ;       360 KB MEDIA IN 360 KB DRIVE                    :
                        ;--------------------------------------------------------

0012                    MD_TBL1         LABEL BYTE
0012 DF                         DB      11011111B       ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
0013 02                         DB      2               ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
0014 25                         DB      MOTOR_WAIT      ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
0015 02                         DB      2               ; 512 BYTES/SECTOR
0016 09                         DB      09              ; EOT ( LAST SECTOR ON TRACK)
0017 2A                         DB      02AH            ; GAP LENGTH
0018 FF                         DB      0FFH            ; DTL
0019 50                         DB      050H            ; GAP LENGTH FOR FORMAT
001A F6                         DB      0F6H            ; FILL BYTE FOR FORMAT
001B 0F                         DB      15              ; HEAD SETTLE TIME (MILLISECONDS)
001C 08                         DB      8               ; MOTOR START TIME (1/8 SECONDS)
001D 27                         DB      39              ; MAX. TRACK NUMBER
001E 80                         DB      RATE_250        ; DATA TRANSFER RATE

                        ;--------------------------------------------------------
                        ;       360 KB MEDIA IN 1.2 MB DRIVE                    :
                        ;--------------------------------------------------------

001F                    MD_TBL2         LABEL BYTE
001F DF                         DB      11011111B       ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
0020 02                         DB      2               ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
0021 25                         DB      MOTOR_WAIT      ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
0022 02                         DB      2               ; 512 BYTES/SECTOR
0023 09                         DB      09              ; EOT ( LAST SECTOR ON TRACK)
0024 2A                         DB      02AH            ; GAP LENGTH
0025 FF                         DB      0FFH            ; DTL
0026 50                         DB      050H            ; GAP LENGTH FOR FORMAT
0027 F6                         DB      0F6H            ; FILL BYTE FOR FORMAT
0028 0F                         DB      15              ; HEAD SETTLE TIME (MILLISECONDS)
0029 08                         DB      8               ; MOTOR START TIME (1/8 SECONDS)
002A 27                         DB      39              ; MAX. TRACK NUMBER
002B 40                         DB      RATE_300        ; DATA TRANSFER RATE

                        ;--------------------------------------------------------
                        ;       1.2 MB MEDIA IN 1.2 MB DRIVE                    :
                        ;--------------------------------------------------------

002C                    MD_TBL3         LABEL BYTE
002C DF                         DB      11011111B       ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
002D 02                         DB      2               ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
002E 25                         DB      MOTOR_WAIT      ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
002F 02                         DB      2               ; 512 BYTES/SECTOR
0030 0F                         DB      15              ; EOT ( LAST SECTOR ON TRACK)
0031 1B                         DB      01BH            ; GAP LENGTH
0032 FF                         DB      0FFH            ; DTL
0033 54                         DB      054H            ; GAP LENGTH FOR FORMAT
0034 F6                         DB      0F6H            ; FILL BYTE FOR FORMAT
0035 0F                         DB      15              ; HEAD SETTLE TIME (MILLISECONDS)
0036 08                         DB      8               ; MOTOR START TIME (1/8 SECONDS)
0037 4F                         DB      79              ; MAX. TRACK NUMBER
0038 00                         DB      RATE_500        ; DATA TRANSFER RATE
```

```
                        ;--------------------------------------------------------
                        ;       720 KB MEDIA IN 720 KB DRIVE                    :
                        ;--------------------------------------------------------

0039                    MD_TBL4         LABEL BYTE
0039 DF                         DB      11011111B       ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
003A 02                         DB      2               ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
003B 25                         DB      MOTOR_WAIT      ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
003C 02                         DB      2               ; 512 BYTES/SECTOR
003D 09                         DB      09              ; EOT ( LAST SECTOR ON TRACK)
003E 2A                         DB      02AH            ; GAP LENGTH
003F FF                         DB      0FFH            ; DTL
0040 50                         DB      050H            ; GAP LENGTH FOR FORMAT
0041 F6                         DB      0F6H            ; FILL BYTE FOR FORMAT
0042 0F                         DB      15              ; HEAD SETTLE TIME (MILLISECONDS)
0043 08                         DB      8               ; MOTOR START TIME (1/8 SECONDS)
0044 4F                         DB      79              ; MAX. TRACK NUMBER
0045 80                         DB      RATE_250        ; DATA TRANSFER RATE

                        ;--------------------------------------------------------
                        ;       720 KB MEDIA IN 1.44 MB DRIVE                   :
                        ;--------------------------------------------------------

0046                    MD_TBL5         LABEL BYTE
0046 DF                         DB      11011111B       ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
0047 02                         DB      2               ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
0048 25                         DB      MOTOR_WAIT      ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
0049 02                         DB      2               ; 512 BYTES/SECTOR
004A 09                         DB      09              ; EOT ( LAST SECTOR ON TRACK)
004B 2A                         DB      02AH            ; GAP LENGTH
004C FF                         DB      0FFH            ; DTL
004D 50                         DB      050H            ; GAP LENGTH FOR FORMAT
004E F6                         DB      0F6H            ; FILL BYTE FOR FORMAT
004F 0F                         DB      15              ; HEAD SETTLE TIME (MILLISECONDS)
0050 08                         DB      8               ; MOTOR START TIME (1/8 SECONDS)
0051 4F                         DB      79              ; MAX. TRACK NUMBER
0052 80                         DB      RATE_250        ; DATA TRANSFER RATE

                        ;--------------------------------------------------------
                        ;       1.44 MB MEDIA IN 1.44 MB DRIVE                  :
                        ;--------------------------------------------------------

0053                    MD_TBL6         LABEL BYTE
0053 AF                         DB      10101111B       ; SRT=A, HD UNLOAD=0F - 1ST SPECIFY BYTE
0054 02                         DB      2               ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
0055 25                         DB      MOTOR_WAIT      ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
0056 02                         DB      2               ; 512 BYTES/SECTOR
0057 12                         DB      18              ; EOT ( LAST SECTOR ON TRACK)
0058 1B                         DB      01BH            ; GAP LENGTH
0059 FF                         DB      0FFH            ; DTL
005A 6C                         DB      06CH            ; GAP LENGTH FOR FORMAT
005B F6                         DB      0F6H            ; FILL BYTE FOR FORMAT
005C 0F                         DB      15              ; HEAD SETTLE TIME (MILLISECONDS)
005D 08                         DB      8               ; MOTOR START TIME (1/8 SECONDS)
005E 4F                         DB      79              ; MAX. TRACK NUMBER
005F 00                         DB      RATE_500        ; DATA TRANSFER RATE


0060                    DISKETTE_IO_1   PROC    FAR             ;>>> ENTRY POINT FOR ORG 0EC59H

0060 FB                         STI                             ; INTERRUPTS BACK ON
0061 55                         PUSH    BP                      ; USER REGISTER
0062 57                         PUSH    DI                      ; USER REGISTER
0063 52                         PUSH    DX                      ; HEAD #, DRIVE # OR USER REGISTER
0064 53                         PUSH    BX                      ; BUFFER OFFSET PARAMETER OR REGISTER
0065 51                         PUSH    CX                      ; TRACK #-SECTOR # OR USER REGISTER
0066 8B EC                      MOV     BP,SP                   ; BP    => PARAMETER LIST DEP. ON AH
                                                                ; [BP]  = SECTOR #
                                                                ; [BP+1]  = TRACK #
                                                                ; [BP+2]  = BUFFER OFFSET
                                                                ; FOR RETURN OF DRIVE PARAMETERS:
                                                                ; CL/[BP] = BITS 7&6 HI BITS OF MAX CYL
                                                                ;           BITS 0-5 MAX SECTORS/TRACK
                                                                ; CH/[BP+1] = LOW 8 BITS OF MAX CYL.
                                                                ; BL/[BP+2] = BITS 7-4 = 0
                                                                ;           BITS 3-0 = VALID CMOS TYPE
                                                                ; BH/[BP+3] = 0
                                                                ; DL/[BP+4] = # DRIVES INSTALLED
                                                                ; DH/[BP+5] = MAX HEAD #
                                                                ; DI/[BP+6] = OFFSET TO DISK BASE
0068 1E                         PUSH    DS                      ; BUFFER SEGMENT PARM OR USER REGISTER
0069 56                         PUSH    SI                      ; USER REGISTERS
006A E8 0000 E                  CALL    DDS                     ; SEGMENT OF BIOS DATA AREA TO DS
006D 80 FC 19                   CMP     AH,(FNC_TAE-FNC_TAB)/2  ; CHECK FOR > LARGEST FUNCTION
0070 72 02                      JB      OK_FUNC                 ; FUNCTION OK
0072 B4 14                      MOV     AH,14H                  ; REPLACE WITH KNOWN INVALID FUNCTION

0074                    OK_FUNC:
0074 80 FC 01                   CMP     AH,1                    ; RESET OR STATUS ?
0077 76 0C                      JBE     OK_DRV                  ; IF RESET OR STATUS DRIVE ALWAYS OK
0079 80 FC 08                   CMP     AH,8                    ; READ DRIVE PARMS ?
007C 74 07                      JZ      OK_DRV                  ; IF SO DRIVE CHECKED LATER
007E 80 FA 01                   CMP     DL,1                    ; DRIVES 0 AND 1 OK
0081 76 02                      JBE     OK_DRV                  ; IF 0 OR 1 THEN JUMP
0083 B4 14                      MOV     AH,14H                  ; REPLACE WITH KNOWN INVALID FUNCTION

0085                    OK_DRV:
0085 8A CC                      MOV     CL,AH                   ; CL = FUNCTION
0087 32 ED                      XOR     CH,CH                   ; CX = FUNCTION
0089 D0 E1                      SHL     CL,1                    ; FUNCTION TIMES 2
008B BB 00B5 R                  MOV     BX,OFFSET FNC_TAB       ; LOAD START OF FUNCTION TABLE
008E 03 D9                      ADD     BX,CX                   ; ADD OFFSET INTO TABLE => ROUTINE
0090 8A E6                      MOV     AH,DH                   ; AX = HEAD #,# OF SECTORS OR DASD TYPE
0092 32 F6                      XOR     DH,DH                   ; DX = DRIVE #
0094 8B F0                      MOV     SI,AX                   ; SI = HEAD #,# OF SECTORS OR DASD TYPE
0096 8B FA                      MOV     DI,DX                   ; DI = DRIVE #
0098 8A 26 0041 R               MOV     AH,@DSKETTE_STATUS      ; LOAD STATUS TO AH FOR STATUS FUNCTION
009C C6 06 0041 R 00            MOV     @DSKETTE_STATUS,0       ; INITIALIZE FOR ALL OTHERS

                        ;       THROUGHOUT THE DISKETTE BIOS, THE FOLLOWING INFORMATION IS CONTAINED IN
                        ;       THE FOLLOWING MEMORY LOCATIONS AND REGISTERS. NOT ALL DISKETTE BIOS
                        ;       FUNCTIONS REQUIRE ALL OF THESE PARAMETERS.
                        ;
```

SECTION 5

```
                        ;               DI      ; DRIVE #
                        ;               SI-HI   ; HEAD #
                        ;               SI-LOW  ; # OF SECTORS OR DASD TYPE FOR FORMAT
                        ;               ES      ; BUFFER SEGMENT
                        ;               [BP]    ; SECTOR #
                        ;               [BP+1]  ; TRACK #
                        ;               [BP+2]  ; BUFFER OFFSET
                        ;
                        ;       ACROSS CALLS TO SUBROUTINES THE CARRY FLAG (CY=1), WHERE INDICATED IN
                        ;       SUBROUTINE PROLOGUES, REPRESENTS AN EXCEPTION RETURN (NORMALLY AN ERROR
                        ;       CONDITION).  IN MOST CASES, WHEN CY = 1, @DSKETTE_STATUS CONTAINS THE
                        ;       SPECIFIC ERROR CODE.
                        ;
00A1 2E: FF 17                  CALL    WORD PTR CS:[BX]        ; (AH) = @DSKETTE_STATUS
                                                                ; CALL THE REQUESTED FUNCTION
00A4 5E                         POP     SI                      ; RESTORE ALL REGISTERS
00A5 1F                         POP     DS
00A6 59                         POP     CX
00A7 5B                         POP     BX
00A8 5A                         POP     DX
00A9 5F                         POP     DI
00AA 8B EC                      MOV     BP,SP
00AC 50                         PUSH    AX
00AD 9C                         PUSHF
00AE 58                         POP     AX
00AF 89 46 06                   MOV     [BP+6],AX
00B2 58                         POP     AX
00B3 5D                         POP     BP
00B4 CF                         IRET
                        ;--------------------------------------------------------
00B5 00E7 R             FNC_TAB DW      DISK_RESET              ; AH = 00H; RESET
00B7 013C R                     DW      DISK_STATUS             ; AH = 01H; STATUS
00B9 0148 R                     DW      DISK_READ               ; AH = 02H; READ
00BB 0154 R                     DW      DISK_WRITE              ; AH = 03H; WRITE
00BD 0160 R                     DW      DISK_VERF               ; AH = 04H; VERIFY
00BF 016C R                     DW      DISK_FORMAT             ; AH = 05H; FORMAT
00C1 01C6 R                     DW      FNC_ERR                 ; AH = 06H; INVALID
00C3 01C6 R                     DW      FNC_ERR                 ; AH = 07H; INVALID
00C5 01D0 R                     DW      DISK_PARMS              ; AH = 08H; READ DRIVE PARAMETERS
00C7 01C6 R                     DW      FNC_ERR                 ; AH = 09H; INVALID
00C9 01C6 R                     DW      FNC_ERR                 ; AH = 0AH; INVALID
00CB 01C6 R                     DW      FNC_ERR                 ; AH = 0BH; INVALID
00CD 01C6 R                     DW      FNC_ERR                 ; AH = 0CH; INVALID
00CF 01C6 R                     DW      FNC_ERR                 ; AH = 0DH; INVALID
00D1 01C6 R                     DW      FNC_ERR                 ; AH = 0EH; INVALID
00D3 01C6 R                     DW      FNC_ERR                 ; AH = 0FH; INVALID
00D5 01C6 R                     DW      FNC_ERR                 ; AH = 10H; INVALID
00D7 01C6 R                     DW      FNC_ERR                 ; AH = 11H; INVALID
00D9 01C6 R                     DW      FNC_ERR                 ; AH = 12H; INVALID
00DB 01C6 R                     DW      FNC_ERR                 ; AH = 13H; INVALID
00DD 01C6 R                     DW      FNC_ERR                 ; AH = 14H; INVALID
00DF 0289 R                     DW      DISK_TYPE               ; AH = 15H; READ DASD TYPE
00E1 02AB R                     DW      DISK_CHANGE             ; AH = 16H; CHANGE STATUS
00E3 02D6 R                     DW      FORMAT_SET              ; AH = 17H; SET DASD TYPE
00E5 0337 R                     DW      SET_MEDIA               ; AH = 18H; SET MEDIA TYPE
= 00E7                  FNC_TAE EQU     $                       ; END
00E7                    DISKETTE_IO_1   ENDP
                        ;--------------------------------------------------------
                        ; DISK_RESET  (AH = 00H)                                :
                        ;       RESET THE DISKETTE SYSTEM.                      :
                        ;                                                       :
                        ; ON EXIT:     @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION :
                        ;--------------------------------------------------------
00E7                    DISK_RESET      PROC    NEAR
00E7 BA 03F2                    MOV     DX,03F2H                ; ADAPTER CONTROL PORT
00EA FA                         CLI                             ; NO INTERRUPTS
00EB A0 003F R                  MOV     AL,@MOTOR_STATUS        ; GET DIGITAL OUTPUT REGISTER REFLECTION
00EE 24 3F                      AND     AL,00111111B            ; KEEP SELECTED AND MOTOR ON BITS
00F0 C0 C0 04                   ROL     AL,4                    ; MOTOR VALUE TO HIGH NIBBLE
                                                                ; DRIVE SELECT TO LOW NIBBLE
00F3 0C 08                      OR      AL,00001000B            ; TURN ON INTERRUPT ENABLE
00F5 EE                         OUT     DX,AL                   ; RESET THE ADAPTER
00F6 C6 06 003E R 00            MOV     @SEEK_STATUS,0          ; SET RECALIBRATE REQUIRED ON ALL DRIVES
00FB EB 00                      JMP     $+2                     ; WAIT FOR I/O
00FD EB 00                      JMP     $+2                     ; WAIT FOR I/O (TO INSURE MINIMUM
                                                                ;   PULSE WIDTH)
00FF 0C 04                      OR      AL,00000100B            ; TURN OFF RESET BIT
0101 EE                         OUT     DX,AL                   ; RESET THE ADAPTER
0102 FB                         STI                             ; ENABLE THE INTERRUPTS
0103 E8 0AC1 R                  CALL    WAIT_INT                ; WAIT FOR THE INTERRUPT
0106 72 2D                      JC      DR_ERR                  ; IF ERROR, RETURN IT
0108 B9 00C0                    MOV     CX,11000000B            ; CL = EXPECTED @NEC_STATUS
                        ;
010B                    NXT_DRV:
010B 51                         PUSH    CX                      ; SAVE FOR CALL
010C B8 0134 R                  MOV     AX,OFFSET DR_POP_ERR    ; LOAD NEC_OUTPUT ERROR ADDRESS
010F 50                         PUSH    AX                      ;  *
0110 B4 08                      MOV     AH,08H                  ; SENSE INTERRUPT STATUS COMMAND
0112 E8 09F8 R                  CALL    NEC_OUTPUT
0115 58                         POP     AX                      ; THROW AWAY ERROR RETURN
0116 E8 0AE9 R                  CALL    RESULTS                 ; READ IN THE RESULTS
0119 59                         POP     CX                      ; RESTORE AFTER CALL
011A 72 19                      JC      DR_ERR                  ; ERROR RETURN
011C 3A 0E 0042 R               CMP     CL,@NEC_STATUS          ; TEST FOR DRIVE READY TRANSITION
0120 75 13                      JNZ     DR_ERR                  ; EVERYTHING OK
0122 FE C1                      INC     CL                      ; NEXT EXPECTED @NEC_STATUS
0124 80 F9 C3                   CMP     CL,11000011B            ; ALL POSSIBLE DRIVES CLEARED
0127 76 E2                      JBE     NXT_DRV                 ; FALL THRU IF 11000100B OR >
                        ;
0129 E8 03D8 R                  CALL    SEND_SPEC               ; SEND SPECIFY COMMAND TO NEC
                        ;
012C                    RESBAC:
012C E8 0864 R                  CALL    SETUP_END               ; VARIOUS CLEANUPS
012F 8B DE                      MOV     BX,SI                   ; GET SAVED AL TO BL
0131 8A C3                      MOV     AL,BL                   ; PUT BACK FOR RETURN
0133 C3                         RET
                        ;
0134                    DR_POP_ERR:
0134 59                         POP     CX                      ; CLEAR STACK
                        ;
0135                    DR_ERR:
0135 80 0E 0041 R 20            OR      @DSKETTE_STATUS,BAD_NEC ; SET ERROR CODE
013A EB F0                      JMP     SHORT RESBAC            ; RETURN FROM RESET
013C                    DISK_RESET      ENDP
```

```
                        ;--------------------------------------------------------------------------
                        ; DISK_STATUS  (AH = 01H)                                                 :
                        ;     DISKETTE STATUS.                                                   :
                        ; ON ENTRY:     AH = STATUS OF PREVIOUS OPERATION                        :
                        ;                                                                        :
                        ; ON EXIT:      @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION          :
                        ;--------------------------------------------------------------------------
013C                    DISK_STATUS     PROC    NEAR
013C 88 26 0041 R               MOV     @DSKETTE_STATUS,AH      ; PUT BACK FOR SETUP_END
0140 E8 0854 R                  CALL    SETUP_END               ; VARIOUS CLEANUPS
0143 8B DE                      MOV     BX,SI                   ; GET SAVED AL TO BL
0145 8A C3                      MOV     AL,BL                   ; PUT BACK FOR RETURN
0147 C3                         RET
0148                    DISK_STATUS     ENDP
                        ;--------------------------------------------------------------------------
                        ; DISK_READ  (AH = 02H)                                                  :
                        ;     DISKETTE READ.                                                     :
                        ; ON ENTRY:     DI      = DRIVE #                                        :
                        ;               SI-HI   = HEAD #                                         :
                        ;               SI-LOW  = # OF SECTORS                                   :
                        ;               ES      = BUFFER SEGMENT                                 :
                        ;               [BP]    = SECTOR #                                       :
                        ;               [BP+1]  = TRACK #                                        :
                        ;               [BP+2]  = BUFFER OFFSET                                  :
                        ;                                                                        :
                        ; ON EXIT:      @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION          :
                        ;--------------------------------------------------------------------------
0148                    DISK_READ       PROC    NEAR
0148 80 26 003F R 7F            AND     @MOTOR_STATUS,01111111B ; INDICATE A READ OPERATION
014D B8 E646                    MOV     AX,0E646H               ; AX = NEC COMMAND, DMA COMMAND
0150 E8 04AE R                  CALL    RD_WR_VF                ; COMMON READ/WRITE/VERIFY
0153 C3                         RET
0154                    DISK_READ       ENDP
                        ;--------------------------------------------------------------------------
                        ; DISK_WRITE  (AH = 03H)                                                 :
                        ;     DISKETTE WRITE.                                                    :
                        ; ON ENTRY:     DI      = DRIVE #                                        :
                        ;               SI-HI   = HEAD #                                         :
                        ;               SI-LOW  = # OF SECTORS                                   :
                        ;               ES      = BUFFER SEGMENT                                 :
                        ;               [BP]    = SECTOR #                                       :
                        ;               [BP+1]  = TRACK #                                        :
                        ;               [BP+2]  = BUFFER OFFSET                                  :
                        ;                                                                        :
                        ; ON EXIT:      @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION          :
                        ;--------------------------------------------------------------------------
0154                    DISK_WRITE      PROC    NEAR
0154 B8 C54A                    MOV     AX,0C54AH               ; AX = NEC COMMAND, DMA COMMAND
0157 80 0E 003F R 80            OR      @MOTOR_STATUS,10000000B ; INDICATE WRITE OPERATION
015C E8 04AE R                  CALL    RD_WR_VF                ; COMMON READ/WRITE/VERIFY
015F C3                         RET
0160                    DISK_WRITE      ENDP
                        ;--------------------------------------------------------------------------
                        ; DISK_VERF  (AH = 04H)                                                  :
                        ;     DISKETTE VERIFY.                                                   :
                        ; ON ENTRY:     DI      = DRIVE #                                        :
                        ;               SI-HI   = HEAD #                                         :
                        ;               SI-LOW  = # OF SECTORS                                   :
                        ;               ES      = BUFFER SEGMENT                                 :
                        ;               [BP]    = SECTOR #                                       :
                        ;               [BP+1]  = TRACK #                                        :
                        ;               [BP+2]  = BUFFER OFFSET                                  :
                        ;                                                                        :
                        ; ON EXIT:      @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION          :
                        ;--------------------------------------------------------------------------
0160                    DISK_VERF       PROC    NEAR
0160 80 26 003F R 7F            AND     @MOTOR_STATUS,01111111B ; INDICATE A READ OPERATION
0165 B8 E642                    MOV     AX,0E642H               ; AX = NEC COMMAND, DMA COMMAND
0168 E8 04AE R                  CALL    RD_WR_VF                ; COMMON READ/WRITE/VERIFY
016B C3                         RET
016C                    DISK_VERF       ENDP
                        ;--------------------------------------------------------------------------
                        ; DISK_FORMAT  (AH = 05H)                                                :
                        ;     DISKETTE FORMAT.                                                   :
                        ; ON ENTRY:     DI      = DRIVE #                                        :
                        ;               SI-HI   = HEAD #                                         :
                        ;               SI-LOW  = # OF SECTORS                                   :
                        ;               ES      = BUFFER SEGMENT                                 :
                        ;               [BP]    = SECTOR #                                       :
                        ;               [BP+1]  = TRACK #                                        :
                        ;               [BP+2]  = BUFFER OFFSET                                  :
                        ;               @DISK_POINTER POINTS TO THE PARAMETER TABLE OF           :
                        ;                       THIS DRIVE                                       :
                        ;                                                                        :
                        ; ON EXIT:      @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION          :
                        ;--------------------------------------------------------------------------
016C                    DISK_FORMAT     PROC    NEAR
016C E8 040F R                  CALL    XLAT_NEW                ; TRANSLATE STATE TO PRESENT ARCH.
016F E8 059B R                  CALL    FMT_INIT                ; ESTABLISH STATE IF UNESTABLISHED
0172 80 0E 003F R 80            OR      @MOTOR_STATUS,10000000B ; INDICATE WRITE OPERATION
0177 E8 05E9 R                  CALL    MED_CHANGE              ; CHECK MEDIA CHANGE AND RESET IF SO
017A 72 3F                      JC      FM_DON                  ; MEDIA CHANGED, SKIP
017C E8 03D8 R                  CALL    SEND_SPEC               ; SEND SPECIFY COMMAND TO NEC
017F E8 063D R                  CALL    CHK_LASTRATE            ; ZF=1 ATTEMPT RATE IS SAME AS LAST RATE
0182 74 03                      JZ      FM_WR                   ; YES, SKIP SPECIFY COMMAND
0184 E8 0624 R                  CALL    SEND_RATE               ; SEND DATA RATE TO CONTROLLER
0187                    FM_WR:
0187 E8 06AD R                  CALL    FMTDMA_SET              ; SET UP THE DMA FOR FORMAT
018A 72 2F                      JC      FM_DON                  ; RETURN WITH ERROR
018C B4 4D                      MOV     AH,04DH                 ; ESTABLISH THE FORMAT COMMAND
018E E8 0700 R                  CALL    NEC_INIT                ; INITIALIZE THE NEC
0191 72 28                      JC      FM_DON                  ; ERROR - EXIT
0193 B8 01BB R                  MOV     AX,OFFSET FM_DON        ; LOAD ERROR ADDRESS
0196 50                         PUSH    AX                      ; PUSH NEC_OUT ERROR RETURN
0197 B2 03                      MOV     DL,3                    ; BYTES/SECTOR VALUE TO NEC
0199 E8 0905 R                  CALL    GET_PARM
019C E8 09F8 R                  CALL    NEC_OUTPUT
019F B2 04                      MOV     DL,4                    ; SECTORS/TRACK VALUE TO NEC
01A1 E8 0905 R                  CALL    GET_PARM
01A4 E8 09F8 R                  CALL    NEC_OUTPUT
01A7 B2 07                      MOV     DL,7                    ; GAP LENGTH VALUE TO NEC
01A9 E8 0905 R                  CALL    GET_PARM
01AC E8 09F8 R                  CALL    NEC_OUTPUT
```

SECTION 5

```
01AF B2 08                      MOV     DL,8                    ; FILLER BYTE TO NEC
01B1 E8 0905 R                  CALL    GET_PARM
01B4 E8 09F8 R                  CALL    NEC_OUTPUT
01B7 58                         POP     AX                      ; THROW AWAY ERROR
01B8 E8 075B R                  CALL    NEC_TERM                ; TERMINATE, RECEIVE STATUS, ETC.
                        FM_DON:
01BB E8 0435 R                  CALL    XLAT_OLD                ; TRANSLATE STATE TO COMPATIBLE MODE
01BE E8 0854 R                  CALL    SETUP_END               ; VARIOUS CLEANUPS
01C1 8B DE                      MOV     BX,SI                   ; GET SAVED AL TO BL
01C3 8A C3                      MOV     AL,BL                   ; PUT BACK FOR RETURN
01C5 C3                         RET
01C6                    DISK_FORMAT     ENDP
                        ;--------------------------------------------------------------
                        ; FNC_ERR                                                     :
                        ;       INVALID FUNCTION REQUESTED OR INVALID DRIVE;          :
                        ;       SET BAD COMMAND IN STATUS.                            :
                        ;                                                             :
                        ; ON EXIT:      @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION :
                        ;--------------------------------------------------------------
01C6                    FNC_ERR PROC    NEAR                    ; INVALID FUNCTION REQUEST
01C6 8B C6                      MOV     AX,SI                   ; RESTORE AL
01C8 B4 01                      MOV     AH,BAD_CMD              ; SET BAD COMMAND ERROR
01CA 88 26 0041 R               MOV     @DSKETTE_STATUS,AH      ; STORE IN DATA AREA
01CE F9                         STC                             ; SET CARRY INDICATING ERROR
01CF C3                         RET
01D0                    FNC_ERR ENDP
                        ;--------------------------------------------------------------
                        ; DISK_PARMS  (AH = 08H)                                       :
                        ;       READ DRIVE PARAMETERS.                                :
                        ; ON ENTRY:                                                   :
                        ;       DI = DRIVE #                                          :
                        ; ON EXIT:                                                    :
                        ;       CL/[BP]   = BITS 7 & 6 HIGH 2 BITS OF MAX CYLINDER    :
                        ;                   BITS 0-5 MAX SECTORS/TRACK                :
                        ;       CH/[BP+1] = LOW 8 BITS OF MAX CYLINDER                :
                        ;       BL/[BP+2] = BITS 7-4 = 0                              :
                        ;                   BITS 3-0 = VALID CMOS DRIVE TYPE          :
                        ;       BH/[BP+3] = 0                                         :
                        ;       DL/[BP+4] = # DRIVES INSTALLED                        :
                        ;       DH/[BP+5] = MAX HEAD #                                :
                        ;       DI/[BP+6] = OFFSET OF MEDIA/DRIVE PARAMETER TABLE     :
                        ;       ES        = SEGMENT OF MEDIA/DRIVE PARAMETER TABLE    :
                        ;       AX        = 0                                         :
                        ;                                                             :
                        ; NOTE : THE ABOVE INFORMATION IS STORED IN THE USERS STACK AT :
                        ;        THE LOCATIONS WHERE THE MAIN ROUTINE WILL POP THEM    :
                        ;        INTO THE APPROPRIATE REGISTERS BEFORE RETURNING TO THE :
                        ;        CALLER.                                              :
                        ;--------------------------------------------------------------
01D0                    DISK_PARMS      PROC    NEAR
01D0 E8 040F R                  CALL    XLAT_NEW                ; TRANSLATE STATE TO PRESENT ARCH.
01D3 C7 46 02 0000              MOV     WORD PTR [BP+2],0       ; DRIVE TYPE = 0
01D8 A1 0010 R                  MOV     AX,@EQUIP_FLAG          ; LOAD EQUIPMENT FLAG FOR # DISKETTES
01DB 24 C1                      AND     AL,11000001B            ; KEEP DISKETTE DRIVE BITS
01DD B2 02                      MOV     DL,2                    ; DISKETTE DRIVES = 2
01DF 3C 41                      CMP     AL,01000001B            ; 2 DRIVES INSTALLED ?
01E1 74 06                      JZ      STO_DL                  ; IF YES JUMP

01E3 FE CA                      DEC     DL                      ; DISKETTE DRIVES = 1
01E5 3C 01                      CMP     AL,00000001B            ; 1 DRIVE INSTALLED ?
01E7 75 6A                      JNZ     NON_DRV                 ; IF NO JUMP

01E9 88 56 04           STO_DL: MOV     [BP+4],DL               ; STORE NUMBER OF DRIVES
01EC 83 FF 01                   CMP     DI,1                    ; CHECK FOR VALID DRIVE
01EF 77 66                      JA      NON_DRV1                ; DRIVE INVALID
01F1 C6 46 05 01                MOV     BYTE PTR[BP+5],1        ; MAXIMUM HEAD NUMBER = 1
01F5 E8 08EC R                  CALL    CMOS_TYPE               ; RETURN DRIVE TYPE IN AL
01F8 72 16                      JC      CHK_EST                 ; ON CMOS BAD CHECK ESTABLISHED
01FA 0A C0                      OR      AL,AL                   ; TEST FOR NO DRIVE TYPE
01FC 74 12                      JZ      CHK_EST                 ; JUMP IF SO
01FE E8 03B8 R                  CALL    DR_TYPE_CHECK           ; RTN CS:@X = MEDIA/DRIVE PARAM TBL
0201 72 0D                      JC      CHK_EST                 ; TYPE NOT IN TABLE (POSSIBLE BAD CMOS)
0203 88 46 02                   MOV     [BP+2],AL               ; STORE VALID CMOS DRIVE TYPE
0206 2E: 8A 4F 04               MOV     CL,CS:[BX].MD_SEC_TRK   ; GET SECTOR/TRACK
020A 2E: 8A 6F 0B               MOV     CH,CS:[BX].MD_MAX_TRK   ; GET MAX. TRACK NUMBER
020E EB 32                      JMP     SHORT STO_CX            ; CMOS GOOD, USE CMOS

0210                    CHK_EST:
0210 8A A5 0090 R               MOV     AH,@DSK_STATE[DI]       ; LOAD STATE FOR THIS DRIVE
0214 F6 C4 10                   TEST    AH,MED_DET              ; CHECK FOR ESTABLISHED STATE
0217 74 3E                      JZ      NON_DRV1                ; CMOS BAD/INVALID AND UNESTABLISHED

0219                    USE_EST:
0219 80 E4 C0                   AND     AH,RATE_MSK             ; ISOLATE STATE
021C 80 FC 80                   CMP     AH,RATE_250             ; RATE 250 ?
021F 75 54                      JNE     USE_EST2                ; NO, GO CHECK OTHER RATE

                        ;--- DATA RATE IS 250 KBS, TRY 360 KB TABLE FIRST

0221 B0 01                      MOV     AL,01                   ; DRIVE TYPE 1 (360KB)
0223 E8 03B8 R                  CALL    DR_TYPE_CHECK           ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
0226 2E: 8A 4F 04               MOV     CL,CS:[BX].MD_SEC_TRK   ; GET SECTOR/TRACK
022A 2E: 8A 6F 0B               MOV     CH,CS:[BX].MD_MAX_TRK   ; GET MAX. TRACK NUMBER
022E F6 85 0090 R 01            TEST    @DSK_STATE[DI],TRK_CAPA ; 80 TRACK ?
0233 74 0D                      JZ      STO_CX                  ; MUST BE 360KB DRIVE

                        ;--- IT IS 1.44 MB DRIVE

0235                    PARM144:
0235 B0 04                      MOV     AL,04                   ; DRIVE TYPE 4 (1.44MB)
0237 E8 03B8 R                  CALL    DR_TYPE_CHECK           ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
023A 2E: 8A 4F 04               MOV     CL,CS:[BX].MD_SEC_TRK   ; GET SECTOR/TRACK
023E 2E: 8A 6F 0B               MOV     CH,CS:[BX].MD_MAX_TRK   ; GET MAX. TRACK NUMBER

0242                    STO_CX:
0242 89 4E 00                   MOV     [BP],CX                 ; SAVE IN STACK FOR RETURN
0245 89 5E 06                   MOV     [BP+6],BX               ; ADDRESS OF MEDIA/DRIVE PARM TABLE
0248 8C C8                      MOV     AX,CS                   ; SEGMENT MEDIA/DRIVE PARAMETER TABLE
024A 8E C0                      MOV     ES,AX                   ; ES IS SEGMENT OF TABLE

024C E8 0435 R          DP_OUT: CALL    XLAT_OLD                ; TRANSLATE STATE TO COMPATIBLE MODE
024F 33 C0                      XOR     AX,AX                   ; CLEAR
0251 F8                         CLC
0252 C3                         RET
```

```
                        ;-----  NO DRIVE PRESENT HANDLER

0253                    NON_DRV:
0253 C6 46 04 00                MOV     BYTE PTR [BP+4],0       ; CLEAR NUMBER OF DRIVES

0257                    NON_DRV1:
0257 81 FF 0080                 CMP     DI,80H                  ; CHECK FOR FIXED MEDIA TYPE REQUEST
025B 72 09                      JB      NON_DRV2                ; CONTINUE IF NOT REQUEST FALL THROUGH

                        ;-----  FIXED DISK REQUEST FALL THROUGH ERROR

025D E8 0435 R                  CALL    XLAT_OLD                ; ELSE TRANSLATE TO COMPATIBLE MODE
0260 8B C6                      MOV     AX,SI                   ; RESTORE AL
0262 B4 01                      MOV     AH,BAD_CMD              ; SET BAD COMMAND ERROR
0264 F9                         STC                             ; SET ERROR RETURN CODE
0265 C3                         RET

0266                    NON_DRV2:
0266 33 C0                      XOR     AX,AX                   ; CLEAR PARMS IF NO DRIVES OR CMOS BAD
0268 89 46 00                   MOV     [BP],AX                 ; TRACKS, SECTORS/TRACK = 0
026B 88 66 05                   MOV     [BP+5],AH               ; HEAD = 0
026E 89 46 06                   MOV     [BP+6],AX               ; OFFSET TO DISK_BASE = 0
0271 8E C0                      MOV     ES,AX                   ; ES IS SEGMENT OF TABLE
0273 EB D7                      JMP     SHORT DP_OUT

                        ;--- DATA RATE IS EITHER 300 KBS OR 500 KBS, TRY 1.2 MB TABLE FIRST

0275                    USE_EST2:
0275 B0 02                      MOV     AL,02                   ; DRIVE TYPE 2 (1.2MB)
0277 E8 03B8 R                  CALL    DR_TYPE_CHECK           ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
027A 2E: 8A 4F 04               MOV     CL,CS:[BX].MD_SEC_TRK   ; GET SECTOR/TRACK
027E 2E: 8A 6F 0B               MOV     CH,CS:[BX].MD_MAX_TRK   ; GET MAX. TRACK NUMBER
0282 80 FC 40                   CMP     AH,RATE_300             ; RATE 300 ?
0285 74 BB                      JE      STO_CX                  ; MUST BE 1.2MB DRIVE
0287 EB AC                      JMP     SHORT PARM144           ; ELSE, IT IS 1.44MB DRIVE

0289                    DISK_PARMS      ENDP

                        ;--------------------------------------------------------------------------
                        ; DISK_TYPE  (AH = 15H)                                                  :
                        ;       THIS ROUTINE RETURNS THE TYPE OF MEDIA INSTALLED.               :
                        ;  ON ENTRY:    DI = DRIVE #                                            :
                        ;                                                                       :
                        ;  ON EXIT:     AH = DRIVE TYPE, CY=0                                   :
                        ;--------------------------------------------------------------------------
0289                    DISK_TYPE       PROC    NEAR
0289 E8 040F R                  CALL    XLAT_NEW                ; TRANSLATE STATE TO PRESENT ARCH.
028C 8A 85 0090 R               MOV     AL,@DSK_STATE[DI]       ; GET PRESENT STATE INFORMATION
0290 0A C0                      OR      AL,AL                   ; CHECK FOR NO DRIVE
0292 74 13                      JZ      NO_DRV                  ;
0294 B4 01                      MOV     AH,NOCHGLN              ; NO CHANGE LINE FOR 40 TRACK DRIVE
0296 A8 01                      TEST    AL,TRK_CAPA             ; IS THIS DRIVE AN 80 TRACK DRIVE?
0298 74 02                      JZ      DT_BACK                 ; IF NO JUMP
029A B4 02                      MOV     AH,CHGLN                ; CHANGE LINE FOR 80 TRACK DRIVE

029C                    DT_BACK:
029C 50                         PUSH    AX                      ; SAVE RETURN VALUE
029D E8 0435 R                  CALL    XLAT_OLD                ; TRANSLATE STATE TO COMPATIBLE MODE
02A0 58                         POP     AX                      ; RESTORE RETURN VALUE
02A1 F8                         CLC                             ; NO ERROR
02A2 8B DE                      MOV     BX,SI                   ; GET SAVED AL TO BL
02A4 8A C3                      MOV     AL,BL                   ; PUT BACK FOR RETURN
02A6 C3                         RET
02A7                    NO_DRV:
02A7 32 E4                      XOR     AH,AH                   ; NO DRIVE PRESENT OR UNKNOWN
02A9 EB F1                      JMP     SHORT DT_BACK
02AB                    DISK_TYPE ENDP

                        ;--------------------------------------------------------------------------
                        ; DISK_CHANGE  (AH = 16H)                                                :
                        ;       THIS ROUTINE RETURNS THE STATE OF THE DISK CHANGE LINE.          :
                        ;                                                                       :
                        ; ON ENTRY:    DI = DRIVE #                                            :
                        ;                                                                       :
                        ; ON EXIT:     AH = @DSKETTE_STATUS                                    :
                        ;                       00 - DISK CHANGE LINE INACTIVE, CY = 0          :
                        ;                       06 - DISK CHANGE LINE ACTIVE, CY = 1            :
                        ;--------------------------------------------------------------------------
02AB                    DISK_CHANGE     PROC    NEAR
02AB E8 040F R                  CALL    XLAT_NEW                ; TRANSLATE STATE TO PRESENT ARCH.
02AE 8A 85 0090 R               MOV     AL,@DSK_STATE[DI]       ; GET MEDIA STATE INFORMATION
02B2 0A C0                      OR      AL,AL                   ; DRIVE PRESENT ?
02B4 74 19                      JZ      DC_NON                  ; JUMP IF NO DRIVE
02B6 A8 01                      TEST    AL,TRK_CAPA             ; 80 TRACK DRIVE ?
02B8 74 05                      JZ      SETIT                   ; IF SO , CHECK CHANGE LINE

02BA E8 0B28 R          DC0:    CALL    READ_DSKCHNG            ; GO CHECK STATE OF DISK CHANGE LINE
02BD 74 05                      JZ      FINIS                   ; CHANGE LINE NOT ACTIVE

02BF C6 06 0041 R 06    SETIT:  MOV     @DSKETTE_STATUS,MEDIA_CHANGE    ; INDICATE MEDIA REMOVED

02C4 E8 0435 R          FINIS:  CALL    XLAT_OLD                ; TRANSLATE STATE TO COMPATIBLE MODE
02C7 E8 0B54 R                  CALL    SETUP_END               ; VARIOUS CLEANUPS
02CA 8B DE                      MOV     BX,SI                   ; GET SAVED AL TO BL
02CC 8A C3                      MOV     AL,BL                   ; PUT BACK FOR RETURN
02CE C3                         RET

02CF                    DC_NON:
02CF 80 0E 0041 R 80            OR      @DSKETTE_STATUS,TIME_OUT        ; SET TIMEOUT, NO DRIVE
02D4 EB EE                      JMP     SHORT FINIS
02D6                    DISK_CHANGE     ENDP

                        ;--------------------------------------------------------------------------
                        ; FORMAT_SET  (AH = 17H)                                                 :
                        ;       THIS ROUTINE IS USED TO ESTABLISH THE TYPE OF                   :
                        ;       MEDIA TO BE USED FOR THE FOLLOWING FORMAT OPERATION.            :
                        ;                                                                       :
                        ; ON ENTRY:    SI LOW = DASD TYPE FOR FORMAT                           :
                        ;               DI     = DRIVE #                                        :
                        ;                                                                       :
                        ; ON EXIT:     @DSKETTE_STATUS REFLECTS STATUS                         :
                        ;               AH = @DSKETTE_STATUS                                    :
                        ;               CY = 1 IF ERROR                                         :
                        ;--------------------------------------------------------------------------
```

```
                        FORMAT_SET      PROC    NEAR
02D6 E8 040F R                  CALL    XLAT_NEW                ; TRANSLATE STATE TO PRESENT ARCH.
02D9 56                         PUSH    SI                      ; SAVE DASD TYPE
02DA 8B C6                      MOV     AX,SI                   ; AH = ? , AL = DASD TYPE
02DC 32 E4                      XOR     AH,AH                   ; AH = 0 , AL = DASD TYPE
02DE 8B F0                      MOV     SI,AX                   ; SI = DASD TYPE
02E0 80 A6 0090 R 0F            AND     @DSK_STATE[DI],NOT MED_DET+DBL_STEP+RATE_MSK    ; CLEAR STATE
02E5 4E                         DEC     SI                      ; CHECK FOR 320/360K MEDIA & DRIVE
02E6 75 07                      JNZ     NOT_320                 ; BYPASS IF NOT
02E8 80 8D 0090 R 90            OR      @DSK_STATE[DI],MED_DET+RATE_250 ; SET TO 320/360
02ED EB 37                      JMP     SHORT S0
                        NOT_320:
02EF E8 05E9 R                  CALL    MED_CHANGE              ; CHECK FOR TIME_OUT
02F2 80 3E 0041 R 80            CMP     @DSKETTE_STATUS,TIME_OUT
02F7 74 2D                      JZ      S0                      ; IF TIME OUT TELL CALLER
02F9 4E                 S3:     DEC     SI                      ; CHECK FOR 320/360K IN 1.2M DRIVE
02FA 75 07                      JNZ     NOT_320_12              ; BYPASS IF NOT
02FC 80 8D 0090 R 70            OR      @DSK_STATE[DI],MED_DET+DBL_STEP+RATE_300 ; SET STATE
0301 EB 23                      JMP     SHORT S0
                        NOT_320_12:
0303 4E                         DEC     SI                      ; CHECK FOR 1.2M MEDIA IN 1.2M DRIVE
0304 75 07                      JNZ     NOT_12                  ; BYPASS IF NOT
0306 80 8D 0090 R 10            OR      @DSK_STATE[DI],MED_DET+RATE_500  ; SET STATE VARIABLE
030B EB 19                      JMP     SHORT S0                ; RETURN TO CALLER
                        NOT_12:
030D 4E                         DEC     SI                      ; CHECK FOR SET DASD TYPE 04
030E 75 20                      JNZ     FS_ERR                  ; BAD COMMAND EXIT IF NOT VALID TYPE
0310 F6 85 0090 R 04            TEST    @DSK_STATE[DI],DRV_DET  ; DRIVE DETERMINED ?
0315 74 09                      JZ      ASSUME                  ; IF STILL NOT DETERMINED ASSUME
0317 B0 50                      MOV     AL,MED_DET+RATE_300
0319 F6 85 0090 R 02            TEST    @DSK_STATE[DI],FMT_CAPA ; MULTIPLE FORMAT CAPABILITY ?
031E 75 02                      JNZ     OR_IT_IN                ; IF 1.2 M THEN DATA RATE 300
                        ASSUME:
0320 B0 90                      MOV     AL,MED_DET+RATE_250     ; SET UP
                        OR_IT_IN:
0322 08 85 0090 R               OR      @DSK_STATE[DI],AL       ; OR IN THE CORRECT STATE
0326 E8 0435 R          S0:     CALL    XLAT_OLD                ; TRANSLATE STATE TO COMPATIBLE MODE
0329 E8 0854 R                  CALL    SETUP_END               ; VARIOUS CLEANUPS
032C 5B                         POP     BX                      ; GET SAVED AL TO BL
032D 8A C3                      MOV     AL,BL                   ; PUT BACK FOR RETURN
032F C3                         RET
                        FS_ERR:
0330 C6 06 0041 R 01            MOV     @DSKETTE_STATUS,BAD_CMD ; UNKNOWN STATE,BAD COMMAND
0335 EB EF                      JMP     SHORT S0
0337                    FORMAT_SET      ENDP

                        ;--------------------------------------------------------------
                        ; SET_MEDIA  (AH = 18H)                                       :
                        ;       THIS ROUTINE SETS THE TYPE OF MEDIA AND DATA RATE     :
                        ;       TO BE USED FOR THE FOLLOWING FORMAT OPERATION.        :
                        ; ON ENTRY:                                                   :
                        ;       [BP]    = SECTOR PER TRACK                            :
                        ;       [BP+1]  = TRACK #                                     :
                        ;       DI      = DRIVE #                                     :
                        ; ON EXIT:                                                    :
                        ;       @DSKETTE_STATUS REFLECTS STATUS                       :
                        ;       IF NO ERROR:                                          :
                        ;           AH = 0                                            :
                        ;           CY = 0                                            :
                        ;           ES = SEGMENT OF MEDIA/DRIVE PARAMETER TABLE       :
                        ;           DI/[BP+6] = OFFSET OF MEDIA/DRIVE PARAMETER TABLE :
                        ;       IF ERROR:                                             :
                        ;           AH = @DSKETTE_STATUS                              :
                        ;           CY = 1                                            :
                        ;--------------------------------------------------------------
0337                    SET_MEDIA       PROC    NEAR
0337 E8 040F R                  CALL    XLAT_NEW                ; TRANSLATE STATE TO PRESENT ARCH.
033A F6 85 0090 R 01            TEST    @DSK_STATE[DI],TRK_CAPA ; CHECK FOR CHANGE LINE AVAILABLE
033F 74 0F                      JZ      SM_CMOS                 ; JUMP IF 40 TRACK DRIVE
0341 E8 05E9 R                  CALL    MED_CHANGE              ; RESET CHANGE LINE
0344 80 3E 0041 R 80            CMP     @DSKETTE_STATUS,TIME_OUT ; IF TIME OUT TELL CALLER
0349 74 66                      JE      SM_RTN
034B C6 06 0041 R 00            MOV     @DSKETTE_STATUS,0       ; CLEAR STATUS
0350                    SM_CMOS:
0350 E8 08EC R                  CALL    CMOS_TYPE               ; RETURN DRIVE TYPE IN (AL)
0353 72 58                      JC      MD_NOT_FND              ; ERROR IN CMOS
0355 0A C0                      OR      AL,AL                   ; TEST FOR NO DRIVE
0357 74 58                      JZ      SM_RTN                  ; RETURN IF SO
0359 E8 03B8 R                  CALL    DR_TYPE_CHECK           ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
035C 72 2F                      JC      MD_NOT_FND              ; TYPE NOT IN TABLE (BAD CMOS)
035E 57                         PUSH    DI                      ; SAVE REG.
035F 33 DB                      XOR     BX,BX                   ; BX = INDEX TO DR_TYPE TABLE
0361 B9 0006                    MOV     CX,DR_CNT               ; CX = LOOP COUNT
0364                    DR_SEARCH:
0364 2E: 8A A7 0000 R           MOV     AH,CS:DR_TYPE[BX]       ; GET DRIVE TYPE
0369 80 E4 7F                   AND     AH,@BIT7OFF             ; MASK OUT MSB
036C 3A C4                      CMP     AL,AH                   ; DRIVE TYPE MATCH ?
036E 75 17                      JNE     NXT_MD                  ; NO, CHECK NEXT DRIVE TYPE
0370 2E: 8B BF 0001 R           MOV     DI,CS:WORD PTR DR_TYPE[BX+1] ; DI = MEDIA/DRIVE PARAMETER TABLE
0375 2E: 8A 65 04               MOV     AH,CS:[DI].MD_SEC_TRK   ; GET SECTOR/TRACK
0379 38 66 00                   CMP     [BP],AH                 ; MATCH ?
037C 75 09                      JNE     NXT_MD                  ; NO, CHECK NEXT MEDIA
037E 2E: 8A 65 0B               MOV     AH,CS:[DI].MD_MAX_TRK   ; GET MAX. TRACK #
0382 38 66 01                   CMP     [BP+1],AH               ; MATCH ?
0385 74 0D                      JE      MD_FND                  ; YES, GO GET RATE
0387                    NXT_MD:
0387 83 C3 03                   ADD     BX,3                    ; CHECK NEXT DRIVE TYPE
038A E2 D8                      LOOP    DR_SEARCH
038C 5F                         POP     DI                      ; RESTORE REG.
038D                    MD_NOT_FND:
038D C6 06 0041 R 0C            MOV     @DSKETTE_STATUS,MED_NOT_FND     ; ERROR, MEDIA TYPE NOT FOUND
0392 EB 1D                      JMP     SHORT SM_RTN            ; RETURN
```

```
                        MD_FND:
03A4 2E: 8A 45 0C               MOV     AL,CS:[DI].MD_RATE      ; GET RATE
03A8 3C 40                      CMP     AL,RATE_300             ; DOUBLE STEP REQUIRED FOR RATE 300
03AA 75 02                      JNE     MD_SET
03AC 0C 20                      OR      AL,DBL_STEP
03AE                    MD_SET:
03AE 89 7E 06                   MOV     [BP+6],DI               ; SAVE TABLE POINTER IN STACK
03A1 0C 10                      OR      AL,MED_DET              ; SET MEDIA ESTABLISHED
03A3 5F                         POP     DI                      ; RESTORE REG.
03A4 80 A5 0090 R 0F            AND     @DSK_STATE[DI],NOT MED_DET+DBL_STEP+RATE_MSK    ; CLEAR STATE
03A9 08 85 0090 R               OR      @DSK_STATE[DI],AL       ; SET STATE
03AD 8C C8                      MOV     AX,CS                   ; SEGMENT MEDIA/DRIVE PARAMETER TABLE
03AF 8E C0                      MOV     ES,AX                   ; ES IS SEGMENT OF TABLE
03B1                    SM_RTN:
03B1 E8 0435 R                  CALL    XLAT_OLD                ; TRANSLATE STATE TO COMPATIBLE MODE
03B4 E8 0854 R                  CALL    SETUP_END               ; VARIOUS CLEANUPS
03B7 C3                         RET
03B8                    SET_MEDIA       ENDP

                        ;--------------------------------------------------------------------
                        ; DR_TYPE_CHECK                                                     :
                        ;       CHECK IF THE GIVEN DRIVE TYPE IN REGISTER (AL)              :
                        ;       IS SUPPORTED IN BIOS DRIVE TYPE TABLE                       :
                        ; ON ENTRY:                                                         :
                        ;       AL = DRIVE TYPE                                             :
                        ; ON EXIT:                                                          :
                        ;       CS = SEGMENT OF MEDIA/DRIVE PARAMETER TABLE (CODE)          :
                        ;       CY = 0   DRIVE TYPE SUPPORTED                               :
                        ;            BX = OFFSET TO MEDIA/DRIVE PARAMETER TABLE             :
                        ;       CY = 1   DRIVE TYPE NOT SUPPORTED                           :
                        ; REGISTERS ALTERED:  BX                                            :
                        ;--------------------------------------------------------------------
03B8                    DR_TYPE_CHECK   PROC    NEAR
03B8 50                         PUSH    AX
03B9 51                         PUSH    CX
03BA 33 DB                      XOR     BX,BX                   ; BX = INDEX TO DR_TYPE TABLE
03BC B9 0006                    MOV     CX,DR_CNT               ; CX = LOOP COUNT
03BF                    TYPE_CHK:
03BF 2E: 8A A7 0000 R           MOV     AH,CS:DR_TYPE[BX]       ; GET DRIVE TYPE
03C4 3A C4                      CMP     AL,AH                   ; DRIVE TYPE MATCH ?
03C6 74 08                      JE      DR_TYPE_VALID           ; YES, RETURN WITH CARRY RESET
03C8 83 C3 03                   ADD     BX,3                    ; CHECK NEXT DRIVE TYPE
03CB E2 F2                      LOOP    TYPE_CHK
03CD F9                         STC                             ; DRIVE TYPE NOT FOUND IN TABLE
03CE EB 05                      JMP     SHORT TYPE_RTN
03D0                    DR_TYPE_VALID:
03D0 2E: 8B 9F 0001 R           MOV     BX,CS:WORD PTR DR_TYPE[BX+1]    ; BX = MEDIA TABLE
03D5                    TYPE_RTN:
03D5 59                         POP     CX
03D6 58                         POP     AX
03D7 C3                         RET
03D8                    DR_TYPE_CHECK   ENDP

                        ;--------------------------------------------------------------------
                        ; SEND_SPEC                                                         :
                        ;       SEND THE SPECIFY COMMAND TO CONTROLLER USING DATA FROM      :
                        ;       THE DRIVE PARAMETER TABLE POINTED BY @DISK_POINTER          :
                        ; ON ENTRY:     @DISK_POINTER = DRIVE PARAMETER TABLE               :
                        ; ON EXIT :     NONE                                                :
                        ; REGISTERS ALTERED: CX, DX                                         :
                        ;--------------------------------------------------------------------
03D8                    SEND_SPEC       PROC    NEAR
03D8 50                         PUSH    AX                      ; SAVE AX
03D9 B8 03F3 R                  MOV     AX,OFFSET SPECBAC       ; LOAD ERROR ADDRESS
03DC 50                         PUSH    AX                      ; PUSH NEC_OUT ERROR RETURN
03DD B4 03                      MOV     AH,03H                  ; SPECIFY COMMAND
03DF E8 09F8 R                  CALL    NEC_OUTPUT              ; OUTPUT THE COMMAND
03E2 2A D2                      SUB     DL,DL                   ; FIRST SPECIFY BYTE
03E4 E8 0905 R                  CALL    GET_PARM                ; GET PARAMETER TO AH
03E7 E8 09F8 R                  CALL    NEC_OUTPUT              ; OUTPUT THE COMMAND
03EA B2 01                      MOV     DL,1                    ; SECOND SPECIFY BYTE
03EC E8 0905 R                  CALL    GET_PARM                ; GET PARAMETER TO AH
03EF E8 09F8 R                  CALL    NEC_OUTPUT              ; OUTPUT THE COMMAND
03F2 58                         POP     AX                      ; POP ERROR RETURN
03F3                    SPECBAC:
03F3 58                         POP     AX                      ; RESTORE ORIGINAL AX VALUE
03F4 C3                         RET
03F5                    SEND_SPEC       ENDP

                        ;--------------------------------------------------------------------
                        ; SEND_SPEC_MD                                                      :
                        ;       SEND THE SPECIFY COMMAND TO CONTROLLER USING DATA FROM      :
                        ;       THE MEDIA/DRIVE PARAMETER TABLE POINTED BY (CS:BX)          :
                        ; ON ENTRY:     CS:BX = MEDIA/DRIVE PARAMETER TABLE                 :
                        ; ON EXIT :     NONE                                                :
                        ; REGISTERS ALTERED: AX                                             :
                        ;--------------------------------------------------------------------
03F5                    SEND_SPEC_MD    PROC    NEAR
03F5 50                         PUSH    AX                      ; SAVE RATE DATA
03F6 B8 040D R                  MOV     AX,OFFSET SPEC_ESBAC    ; LOAD ERROR ADDRESS
03F9 50                         PUSH    AX                      ; PUSH NEC_OUT ERROR RETURN
03FA B4 03                      MOV     AH,03H                  ; SPECIFY COMMAND
03FC E8 09F8 R                  CALL    NEC_OUTPUT              ; OUTPUT THE COMMAND
03FF 2E: 8A 27                  MOV     AH,CS:[BX].MD_SPEC1     ; GET 1ST SPECIFY BYTE
0402 E8 09F8 R                  CALL    NEC_OUTPUT              ; OUTPUT THE COMMAND
0405 2E: 8A 67 01               MOV     AH,CS:[BX].MD_SPEC2     ; GET SECOND SPECIFY BYTE
0409 E8 09F8 R                  CALL    NEC_OUTPUT              ; OUTPUT THE COMMAND
040C 58                         POP     AX                      ; POP ERROR RETURN
040D                    SPEC_ESBAC:
040D 58                         POP     AX                      ; RESTORE RATE
040E C3                         RET
040F                    SEND_SPEC_MD    ENDP
```

```
                        PAGE
                        ;--------------------------------------------------------------------
                        ; XLAT_NEW                                                          :
                        ;       TRANSLATES DISKETTE STATE LOCATIONS FROM COMPATIBLE         :
                        ;       MODE TO NEW ARCHITECTURE.                                  :
                        ;                                                                  :
                        ; ON ENTRY:     DI : DRIVE                                         :
                        ;--------------------------------------------------------------------
0340                    XLAT_NEW        PROC    NEAR                    ;
0340 83 FF 01                   CMP     DI,1                    ; VALID DRIVE ?
0343 77 1C                      JA      XN_OUT                  ; IF INVALID BACK
0345 80 BD 0090 R 00            CMP     @DSK_STATE[DI],0        ; NO DRIVE ?
034A 74 16                      JZ      DO_DET                  ; IF NO DRIVE ATTEMPT DETERMINE
034C 8B CF                      MOV     CX,DI                   ; CX = DRIVE NUMBER
034E C0 E1 02                   SHL     CL,2                    ; CL = SHIFT COUNT, A=0, B=4
0351 A0 008F R                  MOV     AL,@HF_CNTRL            ; DRIVE INFORMATION
0354 D2 C8                      ROR     AL,CL                   ; TO LOW NIBBLE
0356 24 07                      AND     AL,DRV_DET+FMT_CAPA+TRK_CAPA    ; KEEP DRIVE BITS
0358 80 A5 0090 R F8            AND     @DSK_STATE[DI],NOT DRV_DET+FMT_CAPA+TRK_CAPA
035D 08 85 0090 R               OR      @DSK_STATE[DI],AL       ; UPDATE DRIVE STATE
0361                    XN_OUT:
0361 C3                         RET                             ;

0362                    DO_DET:
0362 E8 0B32 R                  CALL    DRIVE_DET               ; TRY TO DETERMINE
0365 C3                         RET

0366                    XLAT_NEW        ENDP
                        ;--------------------------------------------------------------------
                        ; XLAT_OLD                                                          :
                        ;       TRANSLATES DISKETTE STATE LOCATIONS FROM NEW                :
                        ;       ARCHITECTURE TO COMPATIBLE MODE.                           :
                        ;                                                                  :
                        ; ON ENTRY:     DI : DRIVE                                         :
                        ;--------------------------------------------------------------------
0366                    XLAT_OLD        PROC    NEAR
0366 83 FF 01                   CMP     DI,1                    ; VALID DRIVE ?
0369 77 73                      JA      XO_OUT                  ; IF INVALID BACK
036B 80 BD 0090 R 00            CMP     @DSK_STATE[DI],0        ; NO DRIVE ?
0370 74 6C                      JZ      XO_OUT                  ; IF NO DRIVE TRANSLATE DONE

                        ;-----  TEST FOR SAVED DRIVE INFORMATION ALREADY SET

0372 8B CF                      MOV     CX,DI                   ; CX = DRIVE NUMBER
0374 C0 E1 02                   SHL     CL,2                    ; CL = SHIFT COUNT, A=0, B=4
0377 B4 07                      MOV     AH,FMT_CAPA             ; LOAD MULTI DATA RATE BIT MASK
0379 D2 CC                      ROR     AH,CL                   ; ROTATE BY MASK
037B 84 26 008F R               TEST    @HF_CNTRL,AH            ; MULTI-DATA RATE DETERMINED ?
037F 75 16                      JNZ     SAVE_SET                ; IF SO, NO NEED TO RE-SAVE

                        ;-----  ERASE DRIVE BITS IN @HF_CNTRL FOR THIS DRIVE

0381 B4 07                      MOV     AH,DRV_DET+FMT_CAPA+TRK_CAPA    ; MASK TO KEEP
0383 D2 CC                      ROR     AH,CL                   ; FIX MASK TO KEEP
0385 F6 D4                      NOT     AH                      ; TRANSLATE MASK
0387 20 26 008F R               AND     @HF_CNTRL,AH            ; KEEP BITS FROM OTHER DRIVE INTACT

                        ;-----  ACCESS CURRENT DRIVE BITS AND STORE IN @HF_CNTRL

038B 8A 85 0090 R               MOV     AL,@DSK_STATE[DI]       ; ACCESS STATE
038F 24 07                      AND     AL,DRV_DET+FMT_CAPA+TRK_CAPA    ; KEEP DRIVE BITS
0391 D2 C8                      ROR     AL,CL                   ; FIX FOR THIS DRIVE
0393 08 06 008F R               OR      @HF_CNTRL,AL            ; UPDATE SAVED DRIVE STATE

                        ;-----  TRANSLATE TO COMPATIBILITY MODE

0397                    SAVE_SET:
0397 8A A5 0090 R               MOV     AH,@DSK_STATE[DI]       ; ACCESS STATE
039B 8A FC                      MOV     BH,AH                   ; TO BH FOR LATER
039D 80 E4 C0                   AND     AH,RATE_MSK             ; KEEP ONLY RATE
03A0 80 FC 00                   CMP     AH,RATE_500             ; RATE 500 ?
03A3 74 10                      JZ      CHK_144                 ; YES, 1.2/1.2 OR 1.44/1.44
03A5 B0 01                      MOV     AL,M3D1U                ; AL = 360 IN 1.2 UNESTABLISHED
03A7 80 FC 40                   CMP     AH,RATE_300             ; RATE 300 ?
03AA 75 16                      JNZ     CHK_250                 ; NO, 360/360 ,720/720 OR 720/1.44
03AC F6 C7 20                   TEST    BH,DBL_STEP             ; YES, DOUBLE STEP ?
03AF 75 1D                      JNZ     TST_DET                 ; YES, MUST BE 360 IN 1.2

03B1                    UNKNO:
03B1 B0 07                      MOV     AL,MED_UNK              ; 'NONE OF THE ABOVE'
03B3 EB 20                      JMP     SHORT AL_SET            ; PROCESS COMPLETE

03B5                    CHK_144:
03B5 E8 08EC R                  CALL    CMOS_TYPE               ; RETURN DRIVE TYPE IN (AL)
03B8 72 F7                      JC      UNKNO                   ; ERROR, SET 'NONE OF THE ABOVE'
03BA 3C 02                      CMP     AL,02                   ; 1.2MB DRIVE ?
03BC 75 F3                      JNE     UNKNO                   ; NO, GO SET 'NONE OF THE ABOVE'
03BE B0 02                      MOV     AL,M1D1U                ; AL = 1.2 IN 1.2 UNESTABLISHED
03C0 EB 0C                      JMP     SHORT TST_DET

03C2                    CHK_250:
03C2 B0 00                      MOV     AL,M3D3U                ; AL = 360 IN 360 UNESTABLISHED
03C4 80 FC 80                   CMP     AH,RATE_250             ; RATE 250 ?
03C7 75 E8                      JNZ     UNKNO                   ; IF SO FALL THRU
03C9 F6 C7 01                   TEST    BH,TRK_CAPA             ; 80 TRACK CAPABILITY ?
03CC 75 E3                      JNZ     UNKNO                   ; IF SO JUMP, FALL THRU TEST DET

03CE                    TST_DET:
03CE F6 C7 10                   TEST    BH,MED_DET              ; DETERMINED ?
03D1 74 02                      JZ      AL_SET                  ; IF NOT THEN SET
03D3 04 03                      ADD     AL,3                    ; MAKE DETERMINED/ESTABLISHED

03D5                    AL_SET:
03D5 80 A5 0090 R F8            AND     @DSK_STATE[DI],NOT DRV_DET+FMT_CAPA+TRK_CAPA  ; CLEAR DRIVE
03DA 08 85 0090 R               OR      @DSK_STATE[DI],AL       ; REPLACE WITH COMPATIBLE MODE
03DE                    XO_OUT:
03DE C3                         RET
03DF                    XLAT_OLD        ENDP
```

```
                        PAGE
                        ;--------------------------------------------------------
                        ; RD_WR_VF                                              :
                        ;       COMMON READ, WRITE AND VERIFY;                  :
                        ;       MAIN LOOP FOR STATE RETRIES.                    :
                        ;                                                       :
                        ; ON ENTRY:     AH : READ/WRITE/VERIFY NEC PARAMETER    :
                        ;               AL : READ/WRITE/VERIFY DMA PARAMETER    :
                        ;                                                       :
                        ; ON EXIT:      @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION :
                        ;--------------------------------------------------------
04AE                    RD_WR_VF        PROC    NEAR
04AE 50                         PUSH    AX                      ; SAVE DMA, NEC PARAMETERS
04AF E8 040F R                  CALL    XLAT_NEW                ; TRANSLATE STATE TO PRESENT ARCH.
04B2 E8 056A R                  CALL    SETUP_STATE             ; INITIALIZE START AND END RATE
04B5 58                         POP     AX                      ; RESTORE READ/WRITE/VERIFY
                        DO_AGAIN:
04B6 50                         PUSH    AX                      ; SAVE READ/WRITE/VERIFY PARAMETER
04B7 E8 05E9 R                  CALL    MED_CHANGE              ; MEDIA CHANGE AND RESET IF CHANGED
04BA 58                         POP     AX                      ; RESTORE READ/WRITE/VERIFY
                        ;       JC      RWV_END                 ; MEDIA CHANGE ERROR OR TIME-OUT
04BB 73 03                      JNC     RWV
04BD E9 055B R                  JMP     RWV_END
04C0                    RWV:
04C0 50                         PUSH    AX                      ; SAVE READ/WRITE/VERIFY PARAMETER
                                MOV     DH,@DSK_STATE[DI]       ; GET RATE STATE OF THIS DRIVE
04C1 8A B5 0090 R               AND     DH,RATE_MSK             ; KEEP ONLY RATE
04C5 80 E6 C0                   CALL    CMOS_TYPE               ; RETURN DRIVE TYPE IN (AL)
04C8 E8 08EC R                  JC      RWV_ASSUME              ; ERROR IN CMOS
04CB 72 46                      CMP     AL,7                    ; 40 TRACK DRIVE?
04CD 3C 01                      JNE     RWV_1                   ; NO, BYPASS CMOS VALIDITY CHECK
04CF 75 0B                      TEST    @DSK_STATE[DI],TRK_CAPA ; CHECK FOR 40 TRACK DRIVE
04D1 F6 85 0090 R 01            JZ      RWV_2                   ; YES, CMOS IS CORRECT
04D6 74 0F                      MOV     AL,2                    ; CHANGE TO 1.2 M
04D8 B0 02                      JMP     SHORT RWV_2             ; CONTINUE
04DA EB 0B                  RWV_1:
04DC                            JB      RWV_2                   ; NO DRIVE SPECIFIED, CONTINUE
04DC 72 09                      TEST    @DSK_STATE[DI],TRK_CAPA ; IS IT REALLY 40 TRACK?
04DE F6 85 0090 R 01            JNZ     RWV_2                   ; NO, 80 TRACK
04E3 75 02                      MOV     AL,7                    ; IT'S 40 TRACK, FIX CMOS VALUE
04E5 B0 01                  RWV_2:
04E7
04E7 0A C0                      OR      AL,AL                   ; TEST FOR NO DRIVE
04E9 74 28                      JZ      RWV_ASSUME              ; ASSUME TYPE, USE MAX TRACK
04EB E8 03B8 R                  CALL    DR_TYPE_CHECK           ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
04EE 72 23                      JC      RWV_ASSUME              ; TYPE NOT IN TABLE (BAD CMOS)

                        ;--- SEARCH FOR MEDIA/DRIVE PARAMETER TABLE

                                PUSH    DI                      ; SAVE DRIVE #
04F0 57                         XOR     BX,BX                   ; BX = INDEX TO DR_TYPE TABLE
04F1 33 DB                      MOV     CX,DR_CNT               ; CX = LOOP COUNT
04F3 B9 0006                RWV_DR_SEARCH:
04F6                            MOV     AH,CS:DR_TYPE[BX]       ; GET DRIVE TYPE
04F6 2E: 8A A7 0000 R           AND     AH,BIT7OFF              ; MASK OUT MSB
04FB 80 E4 7F                   CMP     AL,AH                   ; DRIVE TYPE MATCH ?
04FE 3A C4                      JNE     RWV_NXT_MD              ; NO, CHECK NEXT DRIVE TYPE
0500 75 0B                  RWV_DR_FND:
0502                            MOV     DI,WORD PTR CS:DR_TYPE[BX+1]  ; DI = MEDIA/DRIVE PARAMETER TABLE
0502 2E: 8B BF 0001 R       RWV_MD_SEARCH:
0507                            CMP     DH,CS:[DI].MD_RATE      ; MATCH ?
0507 2E: 3A 75 0C               JE      RWV_MD_FND              ; YES, GO GET 1ST SPECIFY BYTE
050B 74 16                  RWV_NXT_MD:
050D                            ADD     BX,3                    ; CHECK NEXT DRIVE TYPE
050D 83 C3 03                   LOOP    RWV_DR_SEARCH
0510 E2 E4                      POP     DI                      ; RESTORE DRIVE #
0512 5F

                        ;--- ASSUME PRIMARY DRIVE IS INSTALLED AS SHIPPED

0513                    RWV_ASSUME:
0513 BB 0012 R                  MOV     BX,OFFSET MD_TBL1       ; POINT TO 40 TK 250 KBS
0516 F6 85 0090 R 01            TEST    @DSK_STATE[DI],TRK_CAPA ; TEST FOR 80 TRACK
051B 74 09                      JZ      RWV_MD_FND1             ; MUST BE 40 TRACK
051D BB 002C R                  MOV     BX,OFFSET MD_TBL3       ; POINT TO 80 TK 500 KBS
0520 EB 04 90                   JMP     RWV_MD_FND1             ; GO SET SPECIFY PARAMETERS

                        ;--- CS:BX POINTS TO MEDIA/DRIVE PARAMETER TABLE

0523                    RWV_MD_FND:
0523 8B DF                      MOV     BX,DI                   ; BX = MEDIA/DRIVE PARAMETER TABLE
0525 5F                         POP     DI                      ; RESTORE DRIVE #
0526                    RWV_MD_FND1:

                        ;--- SEND THE SPECIFY COMMAND TO THE CONTROLLER

0526 E8 03F5 R                  CALL    SEND_SPEC_MD
0529 E8 063D R                  CALL    CHK_LASTRATE            ; ZF=1 ATTEMPT RATE IS SAME AS LAST RATE
052C 74 03                      JZ      RWV_DBL                 ; YES, SKIP SEND RATE COMMAND
052E E8 0624 R                  CALL    SEND_RATE               ; SEND DATA RATE TO NEC


0531                    RWV_DBL:
0531 53                         PUSH    BX                      ; SAVE MEDIA/DRIVE PARAM ADDRESS
0532 E8 066E R                  CALL    SETUP_DBL               ; CHECK FOR DOUBLE STEP
0535 5B                         POP     BX                      ; RESTORE ADDRESS
0536 72 1A                      JC      CHK_RET                 ; ERROR FROM READ ID, POSSIBLE RETRY
0538 58                         POP     AX                      ; RESTORE NEC,DMA COMMAND
0539 50                         PUSH    AX                      ; SAVE NEC COMMAND
053A 53                         PUSH    BX                      ; SAVE MEDIA/DRIVE PARAM ADDRESS
053B E8 064D R                  CALL    DMA_SETUP               ; SET UP THE DMA
053E 5B                         POP     BX                      ; RESTORE ADDRESS
053F 58                         POP     AX                      ; RESTORE NEC COMMAND
0540 72 1F                      JC      RWV_BAC                 ; CHECK FOR DMA BOUNDARY ERROR
0542 50                         PUSH    AX                      ; SAVE NEC COMMAND
0543 53                         PUSH    BX                      ; SAVE MEDIA/DRIVE PARAM ADDRESS
0544 E8 0700 R                  CALL    NEC_INIT                ; INITIALIZE NEC
0547 5B                         POP     BX                      ; RESTORE ADDRESS
0548 72 08                      JC      CHK_RET                 ; ERROR - EXIT
054A E8 0725 R                  CALL    RWV_COM                 ; OP CODE COMMON TO READ/WRITE/VERIFY
054D 72 03                      JC      CHK_RET                 ; ERROR - EXIT
054F E8 075B R                  CALL    NEC_TERM                ; TERMINATE, GET STATUS, ETC.
```

SECTION 5

```
                        CHK_RET:
0552 E8 07E5 R                  CALL    RETRY                   ; CHECK FOR, SETUP RETRY
0555 58                         POP     AX                      ; RESTORE READ/WRITE/VERIFY PARAMETER
0556 73 03                      JNC     RWV_END                 ; CY = 0 NO RETRY
0558 E9 04B6 R                  JMP     DO_AGAIN                ; CY = 1 MEANS RETRY

                        RWV_END:
055B E8 07AD R                  CALL    DSTATE                  ; ESTABLISH STATE IF SUCCESSFUL
055E E8 0827 R                  CALL    NUM_TRANS               ; AL = NUMBER TRANSFERRED

                        RWV_BAC:                                ; BAD DMA ERROR ENTRY
0561 50                         PUSH    AX                      ; SAVE NUMBER TRANSFERRED
0562 E8 0435 R                  CALL    XLAT_OLD                ; TRANSLATE STATE TO COMPATIBLE MODE
0565 58                         POP     AX                      ; RESTORE NUMBER TRANSFERRED
0566 E8 0854 R                  CALL    SETUP_END               ; VARIOUS CLEANUPS
0569 C3                         RET
056A                    RD_WR_VF        ENDP
                        ;--------------------------------------------------------
                        ; SETUP_STATE:  INITIALIZES START AND END RATES.       :
                        ;--------------------------------------------------------
056A                    SETUP_STATE     PROC    NEAR            ;
056A F6 85 0090 R 10            TEST    @DSK_STATE[DI],MED_DET  ; MEDIA DETERMINED ?
056F 75 29                      JNZ     JIC                     ; NO STATES IF DETERMINED
0571 B8 0040                    MOV     AX,RATE_500*H+RATE_300  ; AH = START RATE,  AL = END RATE
0574 F6 85 0090 R 04            TEST    @DSK_STATE[DI],DRV_DET  ; DRIVE ?
0579 74 0A                      JZ      AX_SET                  ; DO NOT KNOW DRIVE
057B F6 85 0090 R 02            TEST    @DSK_STATE[DI],FMT_CAPA ; MULTI-RATE ?
0580 75 03                      JNZ     AX_SET                  ; JUMP IF YES
0582 B8 8080                    MOV     AX,RATE_250*X           ; START & END RATE = 250 FOR 360 DRIVE

0585                    AX_SET:
0585 80 A5 0090 R 1F            AND     @DSK_STATE[DI],NOT RATE_MSK+DBL_STEP ; TURN OFF THE RATE
058A 08 A5 0090 R               OR      @DSK_STATE[DI],AH       ; RATE FIRST TO TRY
058E 80 26 008B R F3            AND     @LASTRATE,NOT STRT_MSK  ; ERASE LAST TO TRY RATE BITS
0593 C0 C8 04                   ROR     AL,4                    ; TO OPERATION LAST RATE LOCATION
0596 08 06 008B R               OR      @LASTRATE,AL            ; LAST RATE
059A                    JIC:
059A C3                         RET
059B                    SETUP_STATE     ENDP
                        ;--------------------------------------------------------
                        ;  FMT_INIT: ESTABLISH STATE IF UNESTABLISHED AT FORMAT TIME.  :
                        ;--------------------------------------------------------
059B                    FMT_INIT        PROC    NEAR
059B F6 85 0090 R 10            TEST    @DSK_STATE[DI],MED_DET  ; IS MEDIA ESTABLISHED
05A0 75 42                      JNZ     FI_OUT                  ; IF SO RETURN
05A2 E8 08EC R                  CALL    CMOS_TYPE               ; RETURN DRIVE TYPE IN AL
05A5 72 3E                      JC      CL_DRV                  ; ERROR IN CMOS ASSUME NO DRIVE
05A7 FE C8                      DEC     AL                      ; MAKE ZERO ORIGIN
05A9 78 3A                      JS      CL_DRV                  ; NO DRIVE IF AL 0
05AB 8A A5 0090 R               MOV     AH,@DSK_STATE[DI]       ; AH = CURRENT STATE
05AF 80 E4 0F                   AND     AH,NOT MED_DET+DBL_STEP+RATE_MSK  ; CLEAR
05B2 0A C0                      OR      AL,AL                   ; CHECK FOR 360
05B4 75 05                      JNZ     N_360                   ; IF 360 WILL BE 0
05B6 80 CC 90                   OR      AH,MED_DET+RATE_250     ; ESTABLISH MEDIA
05B9 EB 25                      JMP     SHORT SKP_STATE         ; SKIP OTHER STATE PROCESSING

05BB                    N_360:
05BB FE C8                      DEC     AL                      ; 1.2 M DRIVE
05BD 75 05                      JNZ     N_12                    ; JUMP IF NOT
05BF 80 CC 10           FI_RATE:OR      AH,MED_DET+RATE_500     ; SET FORMAT RATE
05C2 EB 1C                      JMP     SHORT SKP_STATE         ; SKIP OTHER STATE PROCESSING

05C4                    N_12:
05C4 FE C8                      DEC     AL                      ; CHECK FOR TYPE 3
05C6 75 0F                      JNZ     N_720                   ; JUMP IF NOT
05C8 F6 C4 04                   TEST    AH,DRV_DET              ; IS DRIVE DETERMINED
05CB 74 10                      JZ      ISNT_12                 ; TREAT AS NON 1.2 DRIVE
05CD F6 C4 02                   TEST    AH,FMT_CAPA             ; IS 1.2M
05D0 74 0B                      JZ      ISNT_12                 ; JUMP IF NOT
05D2 80 CC 50                   OR      AH,MED_DET+RATE_300     ; RATE 300
05D5 EB 09                      JMP     SHORT SKP_STATE         ; CONTINUE

05D7                    N_720:
05D7 FE C8                      DEC     AL                      ; CHECK FOR TYPE 4
05D9 75 0A                      JNZ     CL_DRV                  ; NO DRIVE, CMOS BAD
05DB EB E2                      JMP     SHORT FI_RATE

05DD                    ISNT_12:
05DD 80 CC 90                   OR      AH,MED_DET+RATE_250     ; MUST BE RATE 250

05E0                    SKP_STATE:
05E0 88 A5 0090 R               MOV     @DSK_STATE[DI],AH       ; STORE AWAY

05E4                    FI_OUT:
05E4 C3                         RET

05E5                    CL_DRV:
05E5 32 E4                      XOR     AH,AH                   ; CLEAR STATE
05E7 EB F7                      JMP     SHORT SKP_STATE         ; SAVE IT
05E9                    FMT_INIT        ENDP
                        ;--------------------------------------------------------
                        ; MED_CHANGE                                           :
                        ;       CHECKS FOR MEDIA CHANGE, RESETS MEDIA CHANGE,  :
                        ;       CHECKS MEDIA CHANGE AGAIN.                     :
                        ;                                                      :
                        ; ON EXIT:      CY = 1 MEANS MEDIA CHANGE OR TIMEOUT   :
                        ;               @DSKETTE_STATUS = ERROR CODE           :
                        ;--------------------------------------------------------
05E9                    MED_CHANGE      PROC    NEAR
05E9 E8 0B28 R                  CALL    READ_DSKCHNG            ; READ DISK CHANGE LINE STATE
05EC 74 34                      JZ      MC_OUT                  ; BYPASS HANDLING DISK CHANGE LINE
05EE 80 A5 0090 R EF            AND     @DSK_STATE[DI],NOT MED_DET      ; CLEAR STATE FOR THIS DRIVE

                        ;       THIS SEQUENCE ENSURES WHENEVER A DISKETTE IS CHANGED THAT
                        ;       ON THE NEXT OPERATION THE REQUIRED MOTOR START UP TIME WILL
                        ;       BE WAITED. (DRIVE MOTOR MAY GO OFF UPON DOOR OPENING).

05F3 8B CF                      MOV     CX,DI                   ; CL = DRIVE #
05F5 B0 01                      MOV     AL,1                    ; MOTOR ON BIT MASK
05F7 D2 E0                      SHL     AL,CL                   ; TO APPROPRIATE POSITION
05F9 F6 D0                      NOT     AL                      ; KEEP ALL BUT MOTOR ON
05FB FA                         CLI                             ; NO INTERRUPTS
05FC 20 06 003F R               AND     @MOTOR_STATUS,AL        ; TURN MOTOR OFF INDICATOR
```

```
                        STI                             ; INTERRUPTS ENABLED
                        CALL    MOTOR_ON                ; TURN MOTOR ON

                ;-----  THIS SEQUENCE OF SEEKS IS USED TO RESET DISKETTE CHANGE SIGNAL

                        CALL    DISK_RESET              ; RESET NEC
                        MOV     CH,0TH                  ; MOVE TO CYLINDER 1
                        CALL    SEEK                    ; ISSUE SEEK
                        XOR     CH,CH                   ; MOVE TO CYLINDER 0
                        CALL    SEEK                    ; ISSUE SEEK
                        MOV     @DSKETTE_STATUS,MEDIA_CHANGE    ; STORE IN STATUS

                OK1:    CALL    READ_DSKCHNG            ; CHECK MEDIA CHANGED AGAIN
                        JZ      OK2                     ; IF ACTIVE, NO DISKETTE, TIMEOUT

                OK4:    MOV     @DSKETTE_STATUS,TIME_OUT; TIMEOUT IF DRIVE EMPTY

                OK2:    STC                             ; MEDIA CHANGED, SET CY
                        RET
                MC_OUT:
                        CLC                             ; NO MEDIA CHANGED, CLEAR CY
                        RET
                MED_CHANGE      ENDP
                ;--------------------------------------------------------------------
                ; SEND_RATE                                                         :
                ;       SENDS DATA RATE COMMAND TO NEC                              :
                ; ON ENTRY:     DI = DRIVE #                                        :
                ; ON EXIT:      NONE                                                :
                ; REGISTERS ALTERED: DX                                            :
                ;--------------------------------------------------------------------
                SEND_RATE       PROC    NEAR            ;

                        PUSH    AX                      ; SAVE REG.
                        AND     @LASTRATE,NOT SEND_MSK  ; ELSE CLEAR LAST RATE ATTEMPTED
                        MOV     AL,@DSK_STATE[DI]       ; GET RATE STATE OF THIS DRIVE
                        AND     AL,SEND_MSK             ; KEEP ONLY RATE BITS
                        OR      @LASTRATE,AL            ; SAVE NEW RATE FOR NEXT CHECK
                        ROL     AL,2                    ; MOVE TO BIT OUTPUT POSITIONS
                        MOV     DX,03F7H                ; OUTPUT NEW DATA RATE
                        OUT     DX,AL

                        POP     AX                      ; RESTORE REG.
                        RET
                SEND_RATE       ENDP

                ;--------------------------------------------------------------------
                ; CHK_LASTRATE                                                      :
                ;       CHECK PREVIOUS DATA RATE SENT TO THE CONTROLLER.             :
                ; ON ENTRY:                                                        :
                ;       DI = DRIVE #                                               :
                ; ON EXIT:                                                         :
                ;       ZF = 1   DATA RATE IS THE SAME AS LAST RATE SENT TO NEC    :
                ;       ZF = 0   DATA RATE IS DIFFERENT FROM LAST RATE             :
                ; REGISTERS ALTERED: NONE                                          :
                ;--------------------------------------------------------------------
                CHK_LASTRATE    PROC    NEAR
                        PUSH    AX                      ; SAVE REG
                        MOV     AH,@LASTRATE            ; GET LAST DATA RATE SELECTED
                        MOV     AL,@DSK_STATE[DI]       ; GET RATE STATE OF THIS DRIVE
                        AND     AX,SEND_MSK*X           ; KEEP ONLY RATE BITS OF BOTH
                        CMP     AL,AH                   ; COMPARE TO PREVIOUSLY TRIED
                                                        ; ZF = 1  RATE IS THE SAME
                        POP     AX                      ; RESTORE REG.
                        RET
                CHK_LASTRATE    ENDP

                SUBTTL (DSK3.ASM)
```

```
                        PAGE
                        ;--------------------------------------------------------
                        ; DMA_SETUP                                             :
                        ;       THIS ROUTINE SETS UP THE DMA FOR READ/WRITE/VERIFY
                        ;       OPERATIONS.                                     :
                        ;                                                       :
                        ; ON ENTRY:     AL = DMA COMMAND                        :
                        ;                                                       :
                        ; ON EXIT:      @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION :
                        ;--------------------------------------------------------
064D                    DMA_SETUP       PROC    NEAR
064D FA                         CLI                             ; DISABLE INTERRUPTS DURING DMA SET-UP
064E E6 0C                      OUT     DMA+12,AL               ; SET THE FIRST/LAST F/F
0650 EB 00                      JMP     $+2                     ; WAIT FOR I/O
0652 E6 0B                      OUT     DMA+11,AL               ; OUTPUT THE MODE BYTE
0654 3C 42                      CMP     AL,42H                  ; DMA VERIFY COMMAND
0656 75 04                      JNE     NOT_VERF                ; NO
0658 33 C0                      XOR     AX,AX                   ; START ADDRESS
065A EB 10                      JMP     SHORT J33
065C                    NOT_VERF:
065C 8C C0                      MOV     AX,ES                   ; GET THE ES VALUE
065E C1 C0 04                   ROL     AX,4                    ; ROTATE LEFT
0661 8A E8                      MOV     CH,AL                   ; GET HIGHEST NIBBLE OF ES TO CH
0663 24 F0                      AND     AL,11110000B            ; ZERO THE LOW NIBBLE FROM SEGMENT
0665 03 46 02                   ADD     AX,[BP+2]               ; TEST FOR CARRY FROM ADDITION
0668 73 02                      JNC     J33
066A FE C5                      INC     CH                      ; CARRY MEANS HIGH 4 BITS MUST BE INC
066C                    J33:
066C 50                         PUSH    AX                      ; SAVE START ADDRESS
066D E6 04                      OUT     DMA+4,AL                ; OUTPUT LOW ADDRESS
066F EB 00                      JMP     $+2                     ; WAIT FOR I/O
0671 8A C4                      MOV     AL,AH
0673 E6 04                      OUT     DMA+4,AL                ; OUTPUT HIGH ADDRESS
0675 8A C5                      MOV     AL,CH                   ; GET HIGH 4 BITS
0677 EB 00                      JMP     $+2                     ; I/O WAIT STATE
0679 24 0F                      AND     AL,00001111B
067B E6 81                      OUT     081H,AL                 ; OUTPUT HIGH 4 BITS TO PAGE REGISTER
                        ;
                        ;-----  DETERMINE COUNT
                        ;
067D 8B C6                      MOV     AX,SI                   ; AL = # OF SECTORS
067F 86 C4                      XCHG    AL,AH                   ; AH = # OF SECTORS
0681 2A C0                      SUB     AL,AL                   ; AL = 0, AX = # OF SECTORS * 256
0683 D1 E8                      SHR     AX,1                    ; AX = # SECTORS * 128
0685 50                         PUSH    AX                      ; SAVE # OF SECTORS * 128
0686 B2 03                      MOV     DL,3                    ; GET BYTES/SECTOR PARAMETER
0688 E8 0905 R                  CALL    GET_PARM                ;
068B 8A CC                      MOV     CL,AH                   ; SHIFT COUNT (0=128, 1=256 ETC)
068D 58                         POP     AX                      ; AX = # OF SECTORS * 128
068E D3 E0                      SHL     AX,CL                   ; SHIFT BY PARAMETER VALUE
0690 48                         DEC     AX                      ;-1 FOR DMA VALUE
0691 50                         PUSH    AX                      ; SAVE COUNT VALUE
0692 E6 05                      OUT     DMA+5,AL                ; LOW BYTE OF COUNT
0694 EB 00                      JMP     $+2                     ; WAIT FOR I/O
0696 8A C4                      MOV     AL,AH
0698 E6 05                      OUT     DMA+5,AL                ; HIGH BYTE OF COUNT
069A FB                         STI                             ; RE-ENABLE INTERRUPTS
069B 59                         POP     CX                      ; RECOVER COUNT VALUE
069C 58                         POP     AX                      ; RECOVER ADDRESS VALUE
069D 03 C1                      ADD     AX,CX                   ; ADD, TEST FOR 64K OVERFLOW
069F B0 02                      MOV     AL,2                    ; MODE FOR 8237
06A1 EB 00                      JMP     $+2                     ; WAIT FOR I/O
06A3 E6 0A                      OUT     DMA+10,AL               ; INITIALIZE THE DISKETTE CHANNEL
                        ;
06A5 73 05                      JNC     NO_BAD                  ; CHECK FOR ERROR
06A7 C6 06 0041 R 09            MOV     @DSKETTE_STATUS,DMA_BOUNDARY    ; SET ERROR
                        ;
06AC                    NO_BAD:
06AC C3                         RET                             ; CY SET BY ABOVE IF ERROR
06AD                    DMA_SETUP       ENDP
```

```
                        PAGE
                        ;--------------------------------------------------------
                        ; FMTDMA_SET                                            :
                        ;       THIS ROUTINE SETS UP THE DMA CONTROLLER FOR A FORMAT :
                        ;       OPERATION.                                      :
                        ;                                                       :
                        ; ON ENTRY:    NOTHING REQUIRED                        :
                        ;                                                       :
                        ; ON EXIT:     @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION :
                        ;--------------------------------------------------------
1608 06AD               FMTDMA_SET      PROC    NEAR
1609 06AD B0 4A                 MOV     AL,04AH                 ; WILL WRITE TO THE DISKETTE
1610 06AF FA                    CLI                             ; DISABLE INTERRUPTS DURING DMA SET-UP
1611 06B0 E6 0C                 OUT     DMA+12,AL               ; SET THE FIRST/LAST F/F
1612 06B2 EB 00                 JMP     $+2                     ; WAIT FOR I/O
1613 06B4 E6 0B                 OUT     DMA+11,AL               ; OUTPUT THE MODE BYTE
1614
1615 06B6 8C C0                 MOV     AX,ES                   ; GET THE ES VALUE
1616 06B8 C1 C0 04              ROL     AX,4                    ; ROTATE LEFT
1617 06BB 8A E8                 MOV     CH,AL                   ; GET HIGHEST NIBBLE OF ES TO CH
1618 06BD 24 F0                 AND     AL,11110000B            ; ZERO THE LOW NIBBLE FROM SEGMENT
1619 06BF 03 46 02              ADD     AX,[BP+2]               ; TEST FOR CARRY FROM ADDITION
1620 06C2 73 02                 JNC     J33A
1621 06C4 FE C5                 INC     CH                      ; CARRY MEANS HIGH 4 BITS MUST BE INC
1622 06C6                       J33A:
1623 06C6 50                    PUSH    AX                      ; SAVE START ADDRESS
1624 06C7 E6 04                 OUT     DMA+4,AL                ; OUTPUT LOW ADDRESS
1625 06C9 EB 00                 JMP     $+2                     ; WAIT FOR I/O
1626 06CB 8A C4                 MOV     AL,AH
1627 06CD E6 04                 OUT     DMA+4,AL                ; OUTPUT HIGH ADDRESS
1628 06CF 8A C5                 MOV     AL,CH                   ; GET HIGH 4 BITS
1629 06D1 EB 00                 JMP     $+2                     ; I/O WAIT STATE
1630 06D3 24 0F                 AND     AL,00001111B
1631 06D5 E6 81                 OUT     081H,AL                 ; OUTPUT HIGH 4 BITS TO PAGE REGISTER
1632
1633                    ;-----  DETERMINE COUNT
1634
1635 06D7 B2 04                 MOV     DL,4                    ; SECTORS/TRACK VALUE IN PARM TABLE
1636 06D9 E8 0905 R             CALL    GET_PARM                ;
1637 06DC 86 C4                 XCHG    AL,AH                   ; AL = SECTORS/TRACK VALUE
1638 06DE 2A E4                 SUB     AH,AH                   ; AX = SECTORS/TRACK VALUE
1639 06E0 C1 E0 02              SHL     AX,2                    ; AX = SEC/TRK * 4 (OFFSET FOR C,H,R,N)
1640 06E3 48                    DEC     AX                      ;-1 FOR DMA VALUE
1641 06E4 50                    PUSH    AX                      ; SAVE # OF BYTES TO BE TRANSFERED
1642 06E5 E6 05                 OUT     DMA+5,AL                ; LOW BYTE OF COUNT
1643 06E7 EB 00                 JMP     $+2                     ; WAIT FOR I/O
1644 06E9 8A C4                 MOV     AL,AH
1645 06EB E6 05                 OUT     DMA+5,AL                ; HIGH BYTE OF COUNT
1646 06ED FB                    STI                             ; RE-ENABLE INTERRUPTS
1647 06EE 59                    POP     CX                      ; RECOVER COUNT VALUE
1648 06EF 58                    POP     AX                      ; RECOVER ADDRESS VALUE
1649 06F0 03 C1                 ADD     AX,CX                   ; ADD, TEST FOR 64K OVERFLOW
1650 06F2 B0 02                 MOV     AL,2                    ; MODE FOR 8237
1651 06F4 EB 00                 JMP     $+2                     ; WAIT FOR I/O
1652 06F6 E6 0A                 OUT     DMA+10,AL               ; INITIALIZE THE DISKETTE CHANNEL
1653
1654 06F8 73 05                 JNC     FMTDMA_OK               ; CHECK FOR ERROR
1655 06FA C6 06 0041 R 09       MOV     @DSKETTE_STATUS,DMA_BOUNDARY    ; SET ERROR
1656
1657 06FF                       FMTDMA_OK:
1658 06FF C3                    RET                             ; CY SET BY ABOVE IF ERROR
1659 0700                       FMTDMA_SET      ENDP
                        ;--------------------------------------------------------
                        ; NEC_INIT                                              :
                        ;       THIS ROUTINE SEEKS TO THE REQUESTED TRACK AND   :
                        ;       INITIALIZES THE NEC FOR THE READ/WRITE/VERIFY/FORMAT :
                        ;       OPERATION.                                      :
                        ;                                                       :
                        ; ON ENTRY:    AH : NEC COMMAND TO BE PERFORMED        :
                        ;                                                       :
                        ; ON EXIT:     @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION :
                        ;--------------------------------------------------------
1660                    NEC_INIT        PROC    NEAR
1661 0700 50                    PUSH    AX                      ; SAVE NEC COMMAND
1662 0701 E8 091A R             CALL    MOTOR_ON                ; TURN MOTOR ON FOR SPECIFIC DRIVE
1663
1664                    ;-----  DO THE SEEK OPERATION
1665
1666 0704 8A 6E 01              MOV     CH,[BP+1]               ; CH = TRACK #
1667 0707 E8 0A24 R             CALL    SEEK                    ; MOVE TO CORRECT TRACK
1668 070A 58                    POP     AX                      ; RECOVER COMMAND
1669 070B 72 1T                 JC      ER_1                    ; ERROR ON SEEK
1670 070D BB 0724 R             MOV     BX,OFFSET ER_1          ; LOAD ERROR ADDRESS
1671 0710 53                    PUSH    BX                      ; PUSH NEC_OUT ERROR RETURN
1672
1673                    ;-----  SEND OUT THE PARAMETERS TO THE CONTROLLER
1674
1675 0711 E8 09F8 R             CALL    NEC_OUTPUT              ; OUTPUT THE OPERATION COMMAND
1676 0714 8B C6                 MOV     AX,SI                   ; AH = HEAD #
1677 0716 8B DF                 MOV     BX,DI                   ; BL = DRIVE #
1678 0718 C0 E4 02              SAL     AH,2                    ; MOVE IT TO BIT 2
1679 071B 80 E4 04              AND     AH,00000100B            ; ISOLATE THAT BIT
1680 071E 0A E3                 OR      AH,BL                   ; OR IN THE DRIVE NUMBER
1681 0720 E8 09F8 R             CALL    NEC_OUTPUT              ; FALL THRU CY SET IF ERROR
1682 0723 5B                    POP     BX                      ; THROW AWAY ERROR RETURN
1683 0724                       ER_1:
1684 0724 C3                    RET
1685 0725                       NEC_INIT        ENDP
                        ;--------------------------------------------------------
                        ; RWV_COM                                               :
                        ;       THIS ROUTINE SENDS PARAMETERS TO THE NEC SPECIFIC :
                        ;       TO THE READ/WRITE/VERIFY OPERATIONS.            :
                        ;                                                       :
                        ; ON ENTRY:    CS:BX = ADDRESS OF MEDIA/DRIVE PARAMETER TABLE :
                        ; ON EXIT :    @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION :
                        ;--------------------------------------------------------
1686                    RWV_COM PROC    NEAR
1687 0725 B8 075A R             MOV     AX,OFFSET ER_2          ; LOAD ERROR ADDRESS
1688 0728 50                    PUSH    AX                      ; PUSH NEC_OUT ERROR RETURN
1689 0729 8A 66 01              MOV     AH,[BP+1]               ; OUTPUT TRACK #
1690 072C E8 09F8 R             CALL    NEC_OUTPUT
1691 072F 8B C6                 MOV     AX,SI                   ; OUTPUT HEAD #
1692 0731 E8 09F8 R             CALL    NEC_OUTPUT
1693 0734 8A 66 00              MOV     AH,[BP]                 ; OUTPUT SECTOR #
```

```
0737 E8 09F8 R                  CALL    NEC_OUTPUT
073A B2 03                      MOV     DL,3
073C E8 0905 R                  CALL    GET_PARM                ; BYTES/SECTOR PARAMETER FROM BLOCK
073F E8 09F8 R                  CALL    NEC_OUTPUT              ; . TO THE NEC
0742 B2 04                      MOV     DL,4                    ; OUTPUT TO CONTROLLER
0744 E8 0905 R                  CALL    GET_PARM                ; EOT PARAMETER FROM BLOCK
0747 E8 09F8 R                  CALL    NEC_OUTPUT              ; . TO THE NEC
                                                                ; OUTPUT TO CONTROLLER

074A 2E: 8A 67 08               MOV     AH,CS:[BX].MD_GAP       ; GET GAP LENGTH
074E E8 09F8 R          R15:    CALL    NEC_OUTPUT
0751 B2 06                      MOV     DL,6                    ; DTL PARAMETER FROM BLOCK
0753 E8 0905 R                  CALL    GET_PARM                ;  TO THE NEC
0756 E8 09F8 R                  CALL    NEC_OUTPUT              ; OUTPUT TO CONTROLLER
0759 58                         POP     AX                      ; THROW AWAY ERROR EXIT
075A                    ER_2:
075A C3                         RET
075B                    RWV_COM ENDP
                        ;--------------------------------------------------------------
                        ; NEC_TERM                                                    :
                        ;       THIS ROUTINE WAITS FOR THE OPERATION THEN ACCEPTS     :
                        ;       THE STATUS FROM THE NEC FOR THE READ/WRITE/VERIFY/    :
                        ;       FORMAT OPERATION.                                    :
                        ;                                                            :
                        ; ON EXIT:     @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION :
                        ;--------------------------------------------------------------
075B                    NEC_TERM        PROC    NEAR

                        ;-----  LET THE OPERATION HAPPEN

075B 56                         PUSH    SI                      ; SAVE HEAD #, # OF SECTORS
075C E8 0AC1 R                  CALL    WAIT_INT                ; WAIT FOR THE INTERRUPT
075F 9C                         PUSHF
0760 E8 0AE9 R                  CALL    RESULTS                 ; GET THE NEC STATUS
0763 72 45                      JC      SET_END_POP
0765 9D                         POPF
0766 72 3A                      JC      SET_END                 ; LOOK FOR ERROR

                        ;-----  CHECK THE RESULTS RETURNED BY THE CONTROLLER

0768 FC                         CLD                             ; SET THE CORRECT DIRECTION
0769 BE 0042 R                  MOV     SI,OFFSET @NEC_STATUS   ; POINT TO STATUS FIELD
076C AC                         LODS    @NEC_STATUS             ; GET STO
076D 24 C0                      AND     AL,11000000B            ; TEST FOR NORMAL TERMINATION
076F 74 31                      JZ      SET_END                 ;
0771 3C 40                      CMP     AL,01000000B            ; TEST FOR ABNORMAL TERMINATION
0773 75 27                      JNZ     J18                     ; NOT ABNORMAL, BAD NEC

                        ;-----  ABNORMAL TERMINATION, FIND OUT WHY

0775 AC                         LODS    @NEC_STATUS             ; GET ST1
0776 D0 E0                      SAL     AL,1                    ; TEST FOR EOT FOUND
0778 B4 04                      MOV     AH,RECORD_NOT_FND
077A 72 22                      JC      J19
077C C0 E0 02                   SAL     AL,2
077F B4 10                      MOV     AH,BAD_CRC
0781 72 1B                      JC      J19
0783 D0 E0                      SAL     AL,1                    ; TEST FOR DMA OVERRUN
0785 B4 08                      MOV     AH,BAD_DMA
0787 72 15                      JC      J19
0789 C0 E0 02                   SAL     AL,2                    ; TEST FOR RECORD NOT FOUND
078C B4 04                      MOV     AH,RECORD_NOT_FND
078E 72 0E                      JC      J19
0790 D0 E0                      SAL     AL,1
0792 B4 03                      MOV     AH,WRITE_PROTECT        ; TEST FOR WRITE_PROTECT
0794 72 08                      JC      J19
0796 D0 E0                      SAL     AL,1                    ; TEST MISSING ADDRESS MARK
0798 B4 02                      MOV     AH,BAD_ADDR_MARK
079A 72 02                      JC      J19

                        ;-----  NEC MUST HAVE FAILED
079C                    J18:
079C B4 20                      MOV     AH,BAD_NEC
079E                    J19:
079E 08 26 0041 R               OR      @DSKETTE_STATUS,AH
07A2                    SET_END:
07A2 80 3E 0041 R 01            CMP     @DSKETTE_STATUS,1       ; SET ERROR CONDITION
07A7 F5                         CMC                             ;
07A8 5E                         POP     SI                      ; RESTORE HEAD #, # OF SECTORS
07A9 C3                         RET

07AA                    SET_END_POP:
07AA 9D                         POPF
07AB EB F5                      JMP     SHORT SET_END
07AD                    NEC_TERM        ENDP
                        ;--------------------------------------------------------------
                        ; DSTATE:      ESTABLISH STATE UPON SUCCESSFUL OPERATION.     :
                        ;--------------------------------------------------------------
07AD                    DSTATE  PROC    NEAR
07AD 80 3E 0041 R 00            CMP     @DSKETTE_STATUS,0       ; CHECK FOR ERROR
07B2 75 30                      JNZ     SETBAC                  ; IF ERROR JUMP
07B4 80 8D 0090 R 10            OR      @DSK_STATE[DI],MED_DET  ; NO ERROR, MARK MEDIA AS DETERMINED
07B9 F6 85 0090 R 04            TEST    @DSK_STATE[DI],DRV_DET  ; DRIVE DETERMINED ?
07BE 75 24                      JNZ     SETBAC                  ; IF DETERMINED NO TRY TO DETERMINE
07C0 8A 85 0090 R               MOV     AL,@DSK_STATE[DI]       ; LOAD STATE
07C4 24 C0                      AND     AL,RATE_MSK             ; KEEP ONLY RATE
07C6 3C 80                      CMP     AL,RATE_250             ; RATE 250 ?
07C8 75 15                      JNE     M_12                    ; NO, MUST BE 1.2M OR 1.44M DRV

                        ;--- CHECK IF IT IS 1.44M

07CA E8 08EC R                  CALL    CMOS_TYPE               ; RETURN DRIVE TYPE IN (AL)
07CD 72 10                      JC      M_12                    ; CMOS BAD
07CF 3C 04                      CMP     AL,04                   ; 1.44MB DRIVE ?
07D1 74 0C                      JE      M_12                    ; YES
07D3                    M_720:
07D3 80 A5 0090 R FD            AND     @DSK_STATE[DI],NOT FMT_CAPA ; TURN OFF FORMAT CAPA
07D8 80 8D 0090 R 04            OR      @DSK_STATE[DI],DRV_DET  ; MARK DRIVE DETERMINED
07DD EB 05                      JMP     SHORT SETBAC            ; BACK

07DF 80 8D 0090 R 06    M_12:   OR      @DSK_STATE[DI],DRV_DET+FMT_CAPA  ; TURN ON DETERMINED & FMT CAPA

07E4                    SETBAC:
07E4 C3                         RET
07E5                    DSTATE  ENDP
```

```
                        ;--------------------------------------------------------------------
                        ; RETRY                                                             :
                        ;       DETERMINES WHETHER A RETRY IS NECESSARY. IF RETRY IS        :
                        ;       REQUIRED THEN STATE INFORMATION IS UPDATED FOR RETRY.       :
                        ;                                                                   :
                        ; ON EXIT:      CY = 1 FOR RETRY, CY = 0 FOR NO RETRY               :
                        ;--------------------------------------------------------------------
07E5                    RETRY   PROC    NEAR
07E5 80 3E 0041 R 00            CMP     @DSKETTE_STATUS,0       ; GET STATUS OF OPERATION
07EA 74 39                      JZ      NO_RETRY                ; SUCCESSFUL OPERATION
07EC 80 3E 0041 R 80            CMP     @DSKETTE_STATUS,TIME_OUT        ; IF TIME OUT NO RETRY
07F1 74 32                      JZ      NO_RETRY                ;
07F3 8A A5 0090 R               MOV     AH,@DSK_STATE[DI]       ; GET MEDIA STATE OF DRIVE
07F7 F6 C4 10                   TEST    AH,MED_DET              ; ESTABLISHED/DETERMINED ?
07FA 75 29                      JNZ     NO_RETRY                ; IF ESTABLISHED STATE THEN TRUE ERROR
07FC 80 E4 C0                   AND     AH,RATE_MSK             ; ISOLATE RATE
07FF 8A 2E 008B R               MOV     CH,@LASTRATE            ; GET START OPERATION STATE
0803 C0 C5 04                   ROL     CH,4                    ; TO CORRESPONDING BITS
0806 80 E5 C0                   AND     CH,RATE_MSK             ; ISOLATE RATE BITS
0809 3A EC                      CMP     CH,AH                   ; ALL RATES TRIED
080B 74 18                      JE      NO_RETRY                ; IF YES, THEN TRUE ERROR
                        ;       SETUP STATE INDICATOR FOR RETRY ATTEMPT TO NEXT RATE
                        ;         00000000B (500) -> 10000000B (250)
                        ;         10000000B (250) -> 01000000B (300)
                        ;         01000000B (300) -> 00000000B (500)
080D 80 FC 01                   CMP     AH,RATE_500+1           ; SET CY FOR RATE 500
0810 D0 DC                      RCR     AH,1                    ; TO NEXT STATE
0812 80 E4 C0                   AND     AH,RATE_MSK             ; KEEP ONLY RATE BITS
0815 80 A5 0090 R 1F            AND     @DSK_STATE[DI],NOT RATE_MSK+DBL_STEP    ; RATE, DBL STEP OFF
081A 08 A5 0090 R               OR      @DSK_STATE[DI],AH       ; TURN ON NEW RATE
081E C6 06 0041 R 00            MOV     @DSKETTE_STATUS,0       ; RESET STATUS FOR RETRY
0823 F9                         STC                             ; SET CARRY FOR RETRY
0824 C3                         RET                             ; RETRY RETURN
                        NO_RETRY:
0825                            CLC                             ; CLEAR CARRY NO RETRY
0826 C3                         RET                             ; NO RETRY RETURN
0827                    RETRY   ENDP
                        ;--------------------------------------------------------------------
                        ; NUM_TRANS                                                         :
                        ;       THIS ROUTINE CALCULATES THE NUMBER OF SECTORS THAT          :
                        ;       WERE ACTUALLY TRANSFERRED TO/FROM THE DISKETTE.             :
                        ;                                                                   :
                        ; ON ENTRY:     [BP+1] = TRACK                                      :
                        ;               SI-HI  = HEAD                                       :
                        ;               [BP]   = START SECTOR                               :
                        ;                                                                   :
                        ; ON EXIT:      AL = NUMBER ACTUALLY TRANSFERRED                    :
                        ;--------------------------------------------------------------------
0827                    NUM_TRANS       PROC    NEAR
0827 32 C0                      XOR     AL,AL                   ; CLEAR FOR ERROR
0829 80 3E 0041 R 00            CMP     @DSKETTE_STATUS,0       ; CHECK FOR ERROR
082E 75 23                      JNZ     NT_OUT                  ; IF ERROR 0 TRANSFERRED
0830 B2 04                      MOV     DL,4                    ; SECTORS/TRACK OFFSET TO DL
0832 E8 0905 R                  CALL    GET_PARM                ; AH = SECTORS/TRACK
0835 8A 1E 0047 R               MOV     BL,@NEC_STATUS+5        ; GET ENDING SECTOR
0839 8B CE                      MOV     CX,SI                   ; CH = HEAD # STARTED
083B 3A 2E 0046 R               CMP     CH,@NEC_STATUS+4        ; GET HEAD ENDED UP ON
083F 75 0B                      JNZ     DIF_HD                  ; IF ON SAME HEAD, THEN NO ADJUST
0841 8A 2E 0045 R               MOV     CH,@NEC_STATUS+3        ; GET TRACK ENDED UP ON
0845 3A 6E 01                   CMP     CH,[BP+1]               ; IS IT ASKED FOR TRACK
0848 74 04                      JZ      SAME_TRK                ; IF SAME TRACK NO INCREASE
084A 02 DC                      ADD     BL,AH                   ; ADD SECTORS/TRACK
084C                    DIF_HD:
084C 02 DC                      ADD     BL,AH                   ; ADD SECTORS/TRACK
084E                    SAME_TRK:
084E 2A 5E 00                   SUB     BL,[BP]                 ; SUBTRACT START FROM END
0851 8A C3                      MOV     AL,BL                   ; TO AL
                        NT_OUT:
0853                            RET
0854                    NUM_TRANS       ENDP
                        ;--------------------------------------------------------------------
                        ; SETUP_END                                                         :
                        ;       RESTORES @MOTOR_COUNT TO PARAMETER PROVIDED IN TABLE        :
                        ;       AND LOADS @DSKETTE_STATUS TO AH, AND SETS CY.               :
                        ; ON EXIT:                                                          :
                        ;       AH, @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION         :
                        ;--------------------------------------------------------------------
0854                    SETUP_END       PROC    NEAR
0854 B2 02                      MOV     DL,2                    ; GET THE MOTOR WAIT PARAMETER
0856 50                         PUSH    AX                      ; SAVE NUMBER TRANSFERRED
0857 E8 0905 R                  CALL    GET_PARM
085A 88 26 0040 R               MOV     @MOTOR_COUNT,AH         ; STORE UPON RETURN
085E 58                         POP     AX                      ; RESTORE NUMBER TRANSFERRED
085F 8A 26 0041 R               MOV     AH,@DSKETTE_STATUS      ; GET STATUS OF OPERATION
0863 0A E4                      OR      AH,AH                   ; CHECK FOR ERROR
0865 74 02                      JZ      NUN_ERR                 ; NO ERROR
0867 32 C0                      XOR     AL,AL                   ; CLEAR NUMBER RETURNED
0869                    NUN_ERR:
0869 80 FC 01                   CMP     AH,1                    ; SET THE CARRY FLAG TO INDICATE
086C F5                         CMC                             ; SUCCESS OR FAILURE
086D C3                         RET
086E                    SETUP_END       ENDP
```

SECTION 5

```
                        PAGE
                        ;--------------------------------------------------------
                        ; SETUP_DBL                                             :
                        ;       CHECK DOUBLE STEP.                              :
                        ; ON ENTRY:                                             :
                        ;       DI = DRIVE                                      :
                        ; ON EXIT :     CY = 1 MEANS ERROR                      :
                        ;--------------------------------------------------------
0866                    SETUP_DBL       PROC    NEAR
0866 8A A5 0090 R               MOV     AH,@DSK_STATE[DI]       ; ACCESS STATE
0872 F6 C4 10                   TEST    AH,MED_DET              ; ESTABLISHED STATE ?
0875 75 5E                      JNZ     NO_DBL                  ; IF ESTABLISHED THEN DOUBLE DONE

                        ;-----  CHECK FOR TRACK 0 TO SPEED UP ACKNOWLEDGE OF UNFORMATTED DISKETTE

0877 C6 06 003E R 00            MOV     @SEEK_STATUS,0          ; SET RECALIBRATE REQUIRED ON ALL DRIVES
087C E8 091A R                  CALL    MOTOR_ON                ; ENSURE MOTOR STAY ON
087F B5 00                      MOV     CH,0                    ; LOAD TRACK 0
0881 E8 0A24 R                  CALL    SEEK                    ; SEEK TO TRACK 0
0884 E8 08D7 R                  CALL    READ_ID                 ; READ ID FUNCTION
0887 72 37                      JC      SD_ERR                  ; IF ERROR NO TRACK 0

                        ;-----  INITIALIZE START AND MAX TRACKS (TIMES 2 FOR BOTH HEADS)

0889 B9 0450                    MOV     CX,0450H                ; START, MAX TRACKS
088C F6 85 0090 R 01            TEST    @DSK_STATE[DI],TRK_CAPA ; TEST FOR 80 TRACK CAPABILITY
0891 74 02                      JZ      CNT_OK                  ; IF NOT COUNT IS SETUP
0893 B1 A0                      MOV     CL,0A0H                 ; MAXIMUM TRACK 1.2 MB

                        ;       ATTEMPT READ ID OF ALL TRACKS, ALL HEADS UNTIL SUCCESS; UPON SUCCESS,
                        ;       MUST SEE IF ASKED FOR TRACK IN SINGLE STEP MODE = TRACK ID READ; IF NOT
                        ;       THEN SET DOUBLE STEP ON.

0895 C6 06 0040 R FF    CNT_OK: MOV     @MOTOR_COUNT,0FFH       ; ENSURE MOTOR STAYS ON FOR OPERATION
089A 51                         PUSH    CX                      ; SAVE TRACK, COUNT
089B C6 06 0041 R 00            MOV     @DSKETTE_STATUS,0       ; CLEAR STATUS, EXPECT ERRORS
08A0 33 C0                      XOR     AX,AX                   ; CLEAR AX
08A2 D0 ED                      SHR     CH,1                    ; HALVE TRACK, CY = HEAD
08A4 D0 D0 03                   RCL     AL,3                    ; AX = HEAD IN CORRECT BIT
08A7 50                         PUSH    AX                      ; SAVE HEAD
08A8 E8 0A24 R                  CALL    SEEK                    ; SEEK TO TRACK
08AB 58                         POP     AX                      ; RESTORE HEAD
08AC 0B F8                      OR      DI,AX                   ; DI = HEAD OR'ED DRIVE
08AE E8 08D7 R                  CALL    READ_ID                 ; READ ID HEAD 0
08B1 9C                         PUSHF                           ; SAVE RETURN FROM READ_ID
08B2 81 E7 00FB                 AND     DI,11111011B            ; TURN OFF HEAD 1 BIT
08B6 9D                         POPF                            ; RESTORE ERROR RETURN
08B7 59                         POP     CX                      ; RESTORE COUNT
08B8 73 08                      JNC     DO_CHK                  ; IF OK, ASKED = RETURNED TRACK ?
08BA FE C5                      INC     CH                      ; INC FOR NEXT TRACK
08BC 3A E9                      CMP     CH,CL                   ; REACHED MAXIMUM YET
08BE 75 D5                      JNZ     CNT_OK                  ; CONTINUE TILL ALL TRIED

                        ;-----  FALL THRU, READ ID FAILED FOR ALL TRACKS

08C0 F9                 SD_ERR: STC                             ; SET CARRY FOR ERROR
08C1 C3                         RET                             ; SETUP_DBL ERROR EXIT

08C2 8A 0E 0045 R       DO_CHK: MOV     CL,@NEC_STATUS+3        ; LOAD RETURNED TRACK
08C6 88 8D 0094 R               MOV     @DSK_TRK[DI],CL         ; STORE TRACK NUMBER
08CA D0 ED                      SHR     CH,1                    ; HALVE TRACK
08CC 3A E9                      CMP     CH,CL                   ; IS IT THE SAME AS ASKED FOR TRACK
08CE 74 05                      JZ      NO_DBL                  ; IF SAME THEN NO DOUBLE STEP
08D0 80 8D 0090 R 20            OR      @DSK_STATE[DI],DBL_STEP ; TURN ON DOUBLE STEP REQUIRED

08D5 F8                 NO_DBL: CLC                             ; CLEAR ERROR FLAG
08D6 C3                         RET
08D7                    SETUP_DBL       ENDP
                        ;--------------------------------------------------------
                        ; READ_ID                                               :
                        ;       READ ID FUNCTION.                               :
                        ; ON ENTRY:     DI = BIT 2 = HEAD; BITS 1,0 = DRIVE     :
                        ;                                                       :
                        ; ON EXIT:      DI = BIT 2 IS RESET, BITS 1,0 = DRIVE   :
                        ;               @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION :
                        ;--------------------------------------------------------
08D7                    READ_ID PROC    NEAR
08D7 B8 08EB R                  MOV     AX,OFFSET ER_9          ; MOVE NEC OUTPUT ERROR ADDRESS
08DA 50                         PUSH    AX
08DB B4 4A                      MOV     AH,4AH                  ; READ ID COMMAND
08DD E8 09F8 R                  CALL    NEC_OUTPUT              ; TO CONTROLLER
08E0 8B C7                      MOV     AX,DI                   ; DRIVE # TO AH, HEAD 0
08E2 8A E0                      MOV     AH,AL
08E4 E8 09F8 R                  CALL    NEC_OUTPUT              ; TO CONTROLLER
08E7 E8 075B R                  CALL    NEC_TERM                ; WAIT FOR OPERATION, GET STATUS
08EA 58                         POP     AX                      ; THROW AWAY ERROR ADDRESS
08EB                    ER_9:
08EB C3                         RET
08EC                    READ_ID ENDP
                        ;--------------------------------------------------------
                        ; CMOS_TYPE                                             :
                        ;       RETURNS DISKETTE TYPE FROM CMOS                 :
                        ;                                                       :
                        ; ON ENTRY:     DI = DRIVE #                            :
                        ;                                                       :
                        ; ON EXIT:      AL = TYPE; CY REFLECTS STATUS           :
                        ;--------------------------------------------------------
08EC                    CMOS_TYPE       PROC    NEAR
08EC B0 0E                      MOV     AL,CMOS_DIAG            ; CMOS DIAGNOSTIC STATUS BYTE ADDRESS
08EE E8 0000 E                  CALL    CMOS_READ               ; GET CMOS STATUS
08F1 A8 C0                      TEST    AL,BAD_BAT+BAD_CKSUM    ; BATTERY GOOD AND CHECKSUM VALID ?
08F3 F9                         STC                             ; SET CY = 1 INDICATING ERROR FOR RETURN
08F4 75 0E                      JNZ     BAD_CM                  ; ERROR IF EITHER BIT ON

08F6 B0 10                      MOV     AL,CMOS_DISKETTE        ; ADDRESS OF DISKETTE BYTE IN CMOS
08F8 E8 0000 E                  CALL    CMOS_READ               ; GET DISKETTE BYTE
08FB 0B FF                      OR      DI,DI                   ; SEE WHICH DRIVE IN QUESTION
08FD 75 03                      JNZ     TB                      ; IF DRIVE 1, DATA IN LOW NIBBLE
08FF C0 C8 04                   ROR     AL,4                    ; EXCHANGE NIBBLES IF SECOND DRIVE
0902                    TB:
0902 24 0F                      AND     AL,00FH                 ; KEEP ONLY DRIVE DATA, RESET CY = 0
0904                    BAD_CM:
0904 C3                         RET
0905                    CMOS_TYPE       ENDP
```

```
                        ;--------------------------------------------------------
                        ; GET_PARM                                              :
                        ;       THIS ROUTINE FETCHES THE INDEXED POINTER FROM THE :
                        ;       DISK_BASE BLOCK POINTED TO BY THE DATA VARIABLE   :
                        ;       @DISK_POINTER. A BYTE FROM THAT TABLE IS THEN MOVED:
                        ;       INTO AH, THE INDEX OF THAT BYTE BEING THE PARAMETER:
                        ;       IN DL.                                           :
                        ;                                                        :
                        ; ON ENTRY:     DL = INDEX OF BYTE TO BE FETCHED         :
                        ;                                                        :
                        ; ON EXIT:      AH = THAT BYTE FROM BLOCK                :
                        ;               AL,DH DESTROYED                          :
                        ;--------------------------------------------------------
0905                    GET_PARM        PROC    NEAR
0905 1E                         PUSH    DS
0906 56                         PUSH    SI
0907 2B C0                      SUB     AX,AX                   ; DS = 0 , BIOS DATA AREA
0909 8E D8                      MOV     DS,AX
090B 87 D3                      XCHG    DX,BX                   ; BL = INDEX
090D 2A FF                      SUB     BH,BH                   ; BX = INDEX
                                ASSUME  DS:ABS0
090F C5 36 0078 R               LDS     SI,@DISK_POINTER        ; POINT TO BLOCK
0913 8A 20                      MOV     AH,[SI+BX]              ; GET THE WORD
0915 87 D3                      XCHG    DX,BX                   ; RESTORE BX
0917 5E                         POP     SI
0918 1F                         POP     DS
0919 C3                         RET
                                ASSUME  DS:DATA
091A                    GET_PARM        ENDP
                        ;--------------------------------------------------------
                        ; MOTOR_ON                                              :
                        ;       TURN MOTOR ON AND WAIT FOR MOTOR START UP TIME. THE @MOTOR_COUNT:
                        ;       IS REPLACED WITH A SUFFICIENTLY HIGH NUMBER (0FFH) TO ENSURE     :
                        ;       THAT THE MOTOR DOES NOT GO OFF DURING THE OPERATION. IF THE      :
                        ;       MOTOR NEEDED TO BE TURNED ON, THE MULTITASKING HOOK FUNCTION     :
                        ;       (AX=90FDH, INT 15H) IS CALLED TELLING THE OPERATING SYSTEM       :
                        ;       THAT THE BIOS IS ABOUT TO WAIT FOR MOTOR START UP. IF THIS       :
                        ;       FUNCTION RETURNS WITH CY = 1, IT MEANS THAT THE MINIMUM WAIT     :
                        ;       HAS BEEN COMPLETED. AT THIS POINT A CHECK IS MADE TO ENSURE      :
                        ;       THAT THE MOTOR WASN'T TURNED OFF BY THE TIMER. IF THE HOOK DID   :
                        ;       NOT WAIT, THE WAIT FUNCTION (AH=086H) IS CALLED TO WAIT THE      :
                        ;       PRESCRIBED AMOUNT OF TIME. IF THE CARRY FLAG IS SET ON RETURN,   :
                        ;       IT MEANS THAT THE FUNCTION IS IN USE AND DID NOT PERFORM THE     :
                        ;       WAIT. A TIMER 1 WAIT LOOP WILL THEN DO THE WAIT.                 :
                        ;                                                        :
                        ; ON ENTRY:     DI = DRIVE #                             :
                        ;                                                        :
                        ; ON EXIT:      AX,CX,DX DESTROYED                       :
                        ;--------------------------------------------------------
091A                    MOTOR_ON        PROC    NEAR
091A 53                         PUSH    BX                      ; SAVE REG.
091B E8 0965 R                  CALL    TURN_ON                 ; TURN ON MOTOR
091E 72 43                      JC      MOT_TS_ON               ; IF CY=1 NO WAIT
0920 E8 0435 R                  CALL    XLAT_OLD                ; TRANSLATE STATE TO COMPATIBLE MODE
0923 B8 90FD                    MOV     AX,090FDH               ; LOAD WAIT CODE & TYPE
0926 CD 15                      INT     15H                     ; TELL OPERATING SYSTEM ABOUT TO DO WAIT
0928 9C                         PUSHF                           ; SAVE CY FOR TEST
0929 E8 040F R                  CALL    XLAT_NEW                ; TRANSLATE STATE TO PRESENT ARCH.
092C 9D                         POPF                            ; RESTORE CY FOR TEST
092D 73 05                      JNC     M_WAIT                  ; BYPASS LOOP IF OP SYSTEM HANDLED WAIT
092F E8 0965 R                  CALL    TURN_ON                 ; CHECK AGAIN IF MOTOR ON
0932 72 2F                      JC      MOT_TS_ON               ; IF NO WAIT MEANS IT IS ON

0934                    M_WAIT:
0934 B2 0A                      MOV     DL,10                   ; GET THE MOTOR WAIT PARAMETER
0936 E8 0905 R                  CALL    GET_PARM
0939 8A C4                      MOV     AL,AH                   ; AL = MOTOR WAIT PARAMETER
093B 32 E4                      XOR     AH,AH                   ; AX = MOTOR WAIT PARAMETER
093D 3C 08                      CMP     AL,8                    ; SEE IF AT LEAST A SECOND IS SPECIFIED
093F 73 02                      JAE     GP2                     ; IF YES, CONTINUE
0941 B0 08                      MOV     AL,8                    ; ONE SECOND WAIT FOR MOTOR START UP

                        ;-----  AX CONTAINS NUMBER OF 1/8 SECONDS (125000 MICROSECONDS) TO WAIT

0943 50                 GP2:    PUSH    AX                      ; SAVE WAIT PARAMETER
0944 BA F424                    MOV     DX,62500                ; LOAD LARGEST POSSIBLE MULTIPLIER
0947 F7 E2                      MUL     DX                      ; MULTIPLY BY HALF OF WHAT'S NECESSARY
0949 8B CA                      MOV     CX,DX                   ; CX = HIGH WORD
094B 8B D0                      MOV     DX,AX                   ; CX,DX = 1/2 * (# OF MICROSECONDS)
094D F8                         CLC                             ; CLEAR CARRY FOR ROTATE
094E D1 D2                      RCL     DX,1                    ; DOUBLE LOW WORD, CY CONTAINS OVERFLOW
0950 D1 D1                      RCL     CX,1                    ; DOUBLE HI, INCLUDING LOW WORD OVERFLOW
0952 B4 86                      MOV     AH,86H                  ; LOAD WAIT CODE
0954 CD 15                      INT     15H                     ; PERFORM WAIT
0956 58                         POP     AX                      ; RESTORE WAIT PARAMETER
0957 73 0A                      JNC     MOT_IS_ON               ; CY MEANS WAIT COULD NOT BE DONE

                        ;-----  FOLLOWING LOOPS REQUIRED WHEN RTC WAIT FUNCTION IS ALREADY IN USE

0959                    J13:                                    ;       WAIT FOR 1/8 SECOND PER (AL)
0959 B9 205E                    MOV     CX,8286                 ; COUNT FOR 1/8 SECOND AT 15.085737 US
095C E8 0000 E                  CALL    WAITF                   ; GO TO FIXED WAIT ROUTINE
095F FE C8                      DEC     AL                      ; DECREMENT TIME VALUE
0961 75 F6                      JNZ     J13                     ; ARE WE DONE YET

0963                    MOT_IS_ON:
0963 5B                         POP     BX                      ; RESTORE REG.
0964 C3                         RET
0965                    MOTOR_ON        ENDP
                        ;--------------------------------------------------------
                        ; TURN_ON                                               :
                        ;       TURN MOTOR ON AND RETURN WAIT STATE.             :
                        ;                                                        :
                        ; ON ENTRY:     DI = DRIVE #                             :
                        ;                                                        :
                        ; ON EXIT:      CY = 0 MEANS WAIT REQUIRED               :
                        ;               CY = 1 MEANS NO WAIT REQUIRED            :
                        ;               AX,BX,CX,DX DESTROYED                    :
                        ;--------------------------------------------------------
0965                    TURN_ON PROC    NEAR
0965 8B DF                      MOV     BX,DI                   ; BX = DRIVE #
0967 8A CB                      MOV     CL,BL                   ; CL = DRIVE #
0969 C0 C3 04                   ROL     BL,4                    ; BL = DRIVE SELECT
096C FA                         CLI                             ; NO INTERRUPTS WHILE DETERMINING STATUS
```

```
                        MOV     @MOTOR_COUNT,0FFH       ; ENSURE MOTOR STAYS ON FOR OPERATION
                        MOV     AL,@MOTOR_STATUS        ; GET DIGITAL OUTPUT REGISTER REFLECTION
                        AND     AL,00110000B            ; KEEP ONLY DRIVE SELECT BITS
                        MOV     AH,1                    ; MASK FOR DETERMINING MOTOR BIT
                        SHL     AH,CL                   ; AH = MOTOR ON, A=00000001, B=00000010

                ;  AL = DRIVE SELECT FROM @MOTOR_STATUS
                ;  BL = DRIVE SELECT DESIRED
                ;  AH = MOTOR ON MASK DESIRED

                        CMP     AL,BL                   ; REQUESTED DRIVE ALREADY SELECTED ?
                        JNZ     TURN_IT_ON              ; IF NOT SELECTED JUMP
                        TEST    AH,@MOTOR_STATUS        ; TEST MOTOR ON BIT
                        JNZ     NO_MOT_WAIT             ; JUMP IF MOTOR ON AND SELECTED

                TURN_IT_ON:
                        OR      AH,BL                   ; AH = DRIVE SELECT AND MOTOR ON
                        MOV     BH,@MOTOR_STATUS        ; SAVE COPY OF @MOTOR_STATUS BEFORE
                        AND     BH,00001111B            ; KEEP ONLY MOTOR BITS
                        AND     @MOTOR_STATUS,11001111B ; CLEAR OUT DRIVE SELECT
                        OR      @MOTOR_STATUS,AH        ; OR IN DRIVE SELECTED AND MOTOR ON
                        MOV     AL,@MOTOR_STATUS        ; GET DIGITAL OUTPUT REGISTER REFLECTION
                        MOV     BL,AL                   ; BL=@MOTOR_STATUS AFTER, BH=BEFORE
                        AND     BL,00001111B            ; KEEP ONLY MOTOR BITS
                        STI                             ; ENABLE INTERRUPTS AGAIN
                        AND     AL,00111111B            ; STRIP AWAY UNWANTED BITS
                        ROL     AL,4                    ; PUT BITS IN DESIRED POSITIONS
                        OR      AL,00001100B            ; NO RESET, ENABLE DMA/INTERRUPT
                        MOV     DX,03F2H                ; SELECT DRIVE AND TURN ON MOTOR
                        OUT     DX,AL                   ;  "
                        CMP     BL,BH                   ; NEW MOTOR TURNED ON ?
                        JZ      NO_MOT_WAIT             ; NO WAIT REQUIRED IF JUST SELECT
                        CLC                             ; SET CARRY MEANING WAIT
                        RET

                NO_MOT_WAIT:
                        STC                             ; SET NO WAIT REQUIRED
                        STI                             ; INTERRUPTS BACK ON
                        RET
                TURN_ON ENDP
                ;--------------------------------------------------------------
                ; HD_WAIT                                                     :
                ;       WAIT FOR HEAD SETTLE TIME.                          :
                ;                                                           :
                ; ON ENTRY:     DI : DRIVE #                                :
                ;                                                           :
                ; ON EXIT:      AX,BX,CX,DX DESTROYED                       :
                ;--------------------------------------------------------------
                HD_WAIT         PROC    NEAR
                        MOV     DL,9                    ; GET HEAD SETTLE PARAMETER
                        CALL    GET_PARM                ;  "
                        TEST    @MOTOR_STATUS,10000000B ; SEE IF A WRITE OPERATION
                        JZ      ISNT_WRITE              ; IF NOT, DO NOT ENFORCE ANY VALUES
                        OR      AH,AH                   ; CHECK FOR ANY WAIT?
                        JNZ     DO_WAT                  ; IF THERE DO NOT ENFORCE
                        MOV     AH,HD12_SETTLE          ; LOAD 1.2M HEAD SETTLE MINIMUM
                        MOV     AL,@DSK_STATE[DI]       ; LOAD STATE
                        AND     AL,RATE_MSK             ; KEEP ONLY RATE
                        CMP     AL,RATE_250             ; 1.2 M DRIVE ?
                        JNZ     DO_WAT                  ; DEFAULT HEAD SETTLE LOADED

                GP3:    MOV     AH,HD320_SETTLE         ; USE 320/360 HEAD SETTLE
                        JMP     SHORT DO_WAT            ;  "

                ISNT_WRITE:
                        OR      AH,AH                   ; CHECK FOR NO WAIT
                        JZ      HW_DONE                 ; IF NOT WRITE AND 0 ITS OK

                ;-----  AH CONTAINS NUMBER OF MILLISECONDS TO WAIT

                DO_WAT: MOV     AL,AH                   ; AL = # MILLISECONDS
                        XOR     AH,AH                   ; AX = # MILLISECONDS
                        PUSH    AX                      ; SAVE HEAD SETTLE PARAMETER
                        MOV     DX,1000                 ; SET UP FOR MULTIPLY TO MICROSECONDS
                        MUL     DX                      ; DX,AX = # MICROSECONDS
                        MOV     CX,DX                   ; CX,AX = # MICROSECONDS
                        MOV     DX,AX                   ; CX,DX = # MICROSECONDS
                        MOV     AH,86H                  ; LOAD WAIT CODE
                        INT     15H                     ; PERFORM WAIT
                        POP     AX                      ; RESTORE HEAD SETTLE PARAMETER
                        JNC     HW_DONE                 ; CHECK FOR EVENT WAIT ACTIVE

                J29:                                    ;       1 MILLISECOND LOOP
                        MOV     CX,66                   ; COUNT AT 15.085737 US PER COUNT
                        CALL    WAITF                   ; DELAY FOR 1 MILLISECOND
                        DEC     AL                      ; DECREMENT THE COUNT
                        JNZ     J29                     ; DO AL MILLISECOND # OF TIMES
                HW_DONE:
                        RET
                HD_WAIT         ENDP
                ;--------------------------------------------------------------
                ; NEC_OUTPUT                                                  :
                ;       THIS ROUTINE SENDS A BYTE TO THE NEC CONTROLLER AFTER :
                ;       TESTING FOR CORRECT DIRECTION AND CONTROLLER READY THIS:
                ;       ROUTINE WILL TIME OUT IF THE BYTE IS NOT ACCEPTED WITHIN:
                ;       A REASONABLE AMOUNT OF TIME, SETTING THE DISKETTE STATUS:
                ;       ON COMPLETION.                                      :
                ;                                                           :
                ; ON ENTRY:                                                 :
                ;       AH = BYTE TO BE OUTPUT                              :
                ; ON EXIT:                                                  :
                ;       CY = 0  SUCCESS                                     :
                ;       CY = 1  FAILURE -- DISKETTE STATUS UPDATED          :
                ;               IF A FAILURE HAS OCCURRED, THE RETURN IS MADE :
                ;               ONE LEVEL HIGHER THAN THE CALLER OF NEC_OUTPUT. :
                ;               THIS REMOVES THE REQUIREMENT OF TESTING AFTER :
                ;               EVERY CALL OF NEC_OUTPUT.                   :
                ;       AX,CX,DX DESTROYED                                  :
                ;--------------------------------------------------------------
                NEC_OUTPUT      PROC    NEAR
                        PUSH    BX                      ; SAVE REG.
                        MOV     DX,03F4H                ; STATUS PORT
                        MOV     BL,2                    ; HIGH ORDER COUNTER
                        XOR     CX,CX                   ; COUNT FOR TIME OUT
```

```
                                ;
0A00 EC                 J23:    IN      AL,DX                   ; GET STATUS
0A01 24 C0                      AND     AL,11000000B            ; KEEP STATUS AND DIRECTION
0A03 3C 80                      CMP     AL,10000000B            ; STATUS 1 AND DIRECTION 0 ?
0A05 74 0F                      JZ      J27                     ; STATUS AND DIRECTION OK
0A07 E2 F7                      LOOP    J23                     ; CONTINUE TILL CX EXHAUSTED
                                ;
0A09 FE CB                      DEC     BL                      ; DECREMENT COUNTER
0A0B 75 F3                      JNZ     J23                     ; REPEAT TILL DELAY FINISHED, CX = 0
                                ;
                        ;-----  FALL THRU TO ERROR RETURN
                                ;
0A0D 80 0E 0041 R 80            OR      @DSKETTE_STATUS,TIME_OUT
0A12 5B                         POP     BX                      ; RESTORE REG.
0A13 58                         POP     AX                      ; DISCARD THE RETURN ADDRESS
0A14 F9                         STC                             ; INDICATE ERROR TO CALLER
0A15 C3                         RET
                                ;
                        ;-----  DIRECTION AND STATUS OK; OUTPUT BYTE
                                ;
0A16                    J27:    ;
0A16 8A C4                      MOV     AL,AH                   ; GET BYTE TO OUTPUT
0A18 42                         INC     DX                      ; DATA PORT = STATUS PORT + 1
0A19 EE                         OUT     DX,AL                   ; OUTPUT THE BYTE
                                ;
0A1A 9C                         PUSHF                           ; SAVE FLAGS
0A1B B9 0003                    MOV     CX,3                    ; 30 TO 45 MICROSECOND WAIT FOR
0A1E E8 0000 E                  CALL    WAITF                   ; NEC FLAGS UPDATE CYCLE
0A21 9D                         POPF                            ; RESTORE FLAGS FOR EXIT
0A22 5B                         POP     BX                      ; RESTORE REG.
0A23 C3                         RET                             ; CY = 0 FROM TEST INSTRUCTION
0A24                    NEC_OUTPUT      ENDP
                        ;--------------------------------------------------------
                        ; SEEK                                                  :
                        ;       THIS ROUTINE WILL MOVE THE HEAD ON THE NAMED DRIVE :
                        ;       TO THE NAMED TRACK.  IF THE DRIVE HAS NOT BEEN ACCESSED :
                        ;       SINCE THE DRIVE RESET COMMAND WAS ISSUED, THE DRIVE :
                        ;       WILL BE RECALIBRATED.                           :
                        ;                                                       :
                        ; ON ENTRY:     DI = DRIVE #                           :
                        ;               CH = TRACK #                           :
                        ;                                                       :
                        ; ON EXIT:      @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION. :
                        ;               AX,BX,CX,DX DESTROYED                  :
                        ;--------------------------------------------------------
0A24                    SEEK    PROC    NEAR
0A24 8B DF                      MOV     BX,DI                   ; BX = DRIVE #
0A26 B0 01                      MOV     AL,1                    ; ESTABLISH MASK FOR RECALIBRATE TEST
0A28 86 CB                      XCHG    CL,BL                   ; GET DRIVE VALUE INTO CL
0A2A D2 C0                      ROL     AL,CL                   ; SHIFT MASK BY THE DRIVE VALUE
0A2C 86 CB                      XCHG    CL,BL                   ; RECOVER TRACK VALUE
0A2E 84 06 003E R               TEST    AL,@SEEK_STATUS         ; TEST FOR RECALIBRATE REQUIRED
0A32 75 1C                      JNZ     J28A                    ; JUMP IF RECALIBRATE NOT REQUIRED
                                ;
0A34 08 06 003E R               OR      @SEEK_STATUS,AL         ; TURN ON THE NO RECALIBRATE BIT IN FLAG
0A38 E8 0A83 R                  CALL    RECAL                   ; RECALIBRATE DRIVE
0A3B 73 0A                      JNC     AFT_RECAL               ; RECALIBRATE DONE
                                ;
                        ;-----  ISSUE RECALIBRATE FOR 80 TRACK DISKETTES
                                ;
0A3D C6 06 0041 R 00            MOV     @DSKETTE_STATUS,0       ; CLEAR OUT INVALID STATUS
0A42 E8 0A83 R                  CALL    RECAL                   ; RECALIBRATE DRIVE
0A45 72 3B                      JC      RB                      ; IF RECALIBRATE FAILS TWICE THEN ERROR
                                ;
0A47                    AFT_RECAL:
0A47 C6 85 0094 R 00            MOV     @DSK_TRK[DI],0          ; SAVE NEW CYLINDER AS PRESENT POSITION
0A4C 0A ED                      OR      CH,CH                   ; CHECK FOR SEEK TO TRACK 0
0A4E 74 2D                      JZ      DO_WAIT                 ; HEAD SETTLE, CY = 0 IF JUMP
                                ;
                        ;-----  DRIVE IS IN SYNCHRONIZATION WITH CONTROLLER, SEEK TO TRACK
                                ;
0A50 F6 85 0090 R 20    J28A:   TEST    @DSK_STATE[DI],DBL_STEP ; CHECK FOR DOUBLE STEP REQUIRED
0A55 74 02                      JZ      R7                      ; SINGLE STEP REQUIRED BYPASS DOUBLE
0A57 D0 E5                      SHL     CH,1                    ; DOUBLE NUMBER OF STEP TO TAKE
                                ;
0A59 3A AD 0094 R       R7:     CMP     CH,@DSK_TRK[DI]         ; SEE IF ALREADY AT THE DESIRED TRACK
0A5D 74 23                      JE      RB                      ; IF YES, DO NOT NEED TO SEEK
                                ;
0A5F BA 0A82 R                  MOV     DX,OFFSET NEC_ERR       ; LOAD RETURN ADDRESS
0A62 52                         PUSH    DX                      ; ON STACK FOR NEC_OUTPUT ERROR
0A63 88 AD 0094 R               MOV     @DSK_TRK[DI],CH         ; SAVE NEW CYLINDER AS PRESENT POSITION
0A67 B4 0F                      MOV     AH,0FH                  ; SEEK COMMAND TO NEC
0A69 E8 09F8 R                  CALL    NEC_OUTPUT
0A6C 8B DF                      MOV     BX,DI                   ; BX = DRIVE #
0A6E 8A E3                      MOV     AH,BL                   ; OUTPUT DRIVE NUMBER
0A70 E8 09F8 R                  CALL    NEC_OUTPUT
0A73 8A A5 0094 R               MOV     AH,@DSK_TRK[DI]         ; GET CYLINDER NUMBER
0A77 E8 09F8 R                  CALL    NEC_OUTPUT
0A7A E8 0A9A R                  CALL    CHK_STAT_2              ; ENDING INTERRUPT AND SENSE STATUS
                                ;
                        ;-----  WAIT FOR HEAD SETTLE
                                ;
0A7D                    DO_WAIT:
0A7D 9C                         PUSHF                           ; SAVE STATUS
0A7E E8 09B4 R                  CALL    HD_WAIT                 ; WAIT FOR HEAD SETTLE TIME
0A81 9D                         POPF                            ; RESTORE STATUS
0A82                    RB:
0A82                    NEC_ERR:
0A82 C3                         RET                             ; RETURN TO CALLER
0A83                    SEEK    ENDP
                        ;--------------------------------------------------------
                        ; RECAL                                                 :
                        ;       RECALIBRATE DRIVE                              :
                        ;                                                       :
                        ; ON ENTRY      DI = DRIVE #                           :
                        ;                                                       :
                        ; ON EXIT:      CY  REFLECTS STATUS OF OPERATION.      :
                        ;--------------------------------------------------------
0A83                    RECAL   PROC    NEAR
0A83 51                         PUSH    CX
0A84 B8 0A98 R                  MOV     AX,OFFSET RC_BACK       ; LOAD NEC_OUTPUT ERROR
0A87 50                         PUSH    AX
0A88 B4 07                      MOV     AH,07H                  ; RECALIBRATE COMMAND
0A8A E8 09F8 R                  CALL    NEC_OUTPUT
```

```
0A8D 8B DF                      MOV     BX,DI                   ; BX = DRIVE #
0A8F 8A E3                      MOV     AH,BL
0A91 E8 09F8 R                  CALL    NEC_OUTPUT              ; OUTPUT THE DRIVE NUMBER
0A94 E8 0A9A R                  CALL    CHK_STAT_2              ; GET THE INTERRUPT AND SENSE INT STATUS
0A97 58                         POP     AX                      ; THROW AWAY ERROR
0A98                    RC_BACK:
0A98 59                         POP     CX
0A99 C3                         RET
0A9A                    RECAL   ENDP
                        ;--------------------------------------------------------
                        ; CHK_STAT_2                                           :
                        ;       THIS ROUTINE HANDLES THE INTERRUPT RECEIVED AFTER :
                        ;       RECALIBRATE OR SEEK TO THE ADAPTER. THE         :
                        ;       INTERRUPT IS WAITED FOR, THE INTERRUPT STATUS SENSED, :
                        ;       AND THE RESULT RETURNED TO THE CALLER.          :
                        ;                                                       :
                        ; ON EXIT:    @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION.:
                        ;--------------------------------------------------------
0A9A                    CHK_STAT_2      PROC    NEAR
0A9A B8 0AB8 R                  MOV     AX,OFFSET CS_BACK       ; LOAD NEC_OUTPUT ERROR ADDRESS
0A9D 50                         PUSH    AX
0A9E E8 0AC1 R                  CALL    WAIT_INT                ; WAIT FOR THE INTERRUPT
0AA1 72 14                      JC      J34                     ; IF ERROR, RETURN IT
0AA3 B4 08                      MOV     AH,08H                  ; SENSE INTERRUPT STATUS COMMAND
0AA5 E8 09F8 R                  CALL    NEC_OUTPUT
0AA8 E8 0AE9 R                  CALL    RESULTS                 ; READ IN THE RESULTS
0AAB 72 0A                      JC      J34
0AAD A0 0042 R                  MOV     AL,@NEC_STATUS          ; GET THE FIRST STATUS BYTE
0AB0 24 60                      AND     AL,01100000B            ; ISOLATE THE BITS
0AB2 3C 60                      CMP     AL,01100000B            ; TEST FOR CORRECT VALUE
0AB4 74 03                      JZ      J36                     ; IF ERROR, GO MARK IT
0AB6 F8                         CLC                             ; GOOD RETURN
0AB7                    J34:
0AB7 58                         POP     AX                      ; THROW AWAY ERROR RETURN
0AB8                    CS_BACK:
0AB8 C3                         RET

0AB9                    J35:
0AB9 80 0E 0041 R 40            OR      @DSKETTE_STATUS,BAD_SEEK
0ABE F9                         STC                             ; ERROR RETURN CODE
0ABF EB F6                      JMP     SHORT J34
0AC1                    CHK_STAT_2      ENDP
                        ;--------------------------------------------------------
                        ; WAIT_INT                                             :
                        ;       THIS ROUTINE WAITS FOR AN INTERRUPT TO OCCUR A TIME OUT :
                        ;       ROUTINE TAKES PLACE DURING THE WAIT, SO THAT AN ERROR :
                        ;       MAY BE RETURNED IF THE DRIVE IS NOT READY.      :
                        ;                                                       :
                        ; ON EXIT:    @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION.:
                        ;--------------------------------------------------------
0AC1                    WAIT_INT        PROC    NEAR
0AC1 FB                         STI                             ; TURN ON INTERRUPTS, JUST IN CASE
0AC2 F8                         CLC                             ; CLEAR TIMEOUT INDICATOR
0AC3 B8 9001                    MOV     AX,09001H               ; LOAD WAIT CODE AND TYPE
0AC6 CD 15                      INT     15H                     ; PERFORM OTHER FUNCTION
0AC8 72 11                      JC      J36A                    ; BYPASS TIMING LOOP IF TIMEOUT DONE
0ACA B3 0A                      MOV     BL,10                   ; CLEAR THE COUNTERS
0ACC 33 C9                      XOR     CX,CX                   ; FOR 2 SECOND WAIT
0ACE                    J36:
0ACE F6 06 003E R 80            TEST    @SEEK_STATUS,INT_FLAG   ; TEST FOR INTERRUPT OCCURRING
0AD3 75 0C                      JNZ     J37
0AD5 E2 F7                      LOOP    J36                     ; COUNT DOWN WHILE WAITING
0AD7 FE CB                      DEC     BL                      ; SECOND LEVEL COUNTER
0AD9 75 F3                      JNZ     J36

0ADB 80 0E 0041 R 80    J36A:   OR      @DSKETTE_STATUS,TIME_OUT        ; NOTHING HAPPENED
0AE0 F9                         STC                             ; ERROR RETURN
0AE1                    J37:
0AE1 9C                         PUSHF                           ; SAVE CURRENT CARRY
0AE2 80 26 003E R 7F            AND     @SEEK_STATUS,NOT INT_FLAG       ; TURN OFF INTERRUPT FLAG
0AE7 9D                         POPF                            ; RECOVER CARRY
0AE8 C3                         RET                             ; GOOD RETURN CODE
0AE9                    WAIT_INT        ENDP
                        ;--------------------------------------------------------
                        ; RESULTS                                              :
                        ;       THIS ROUTINE WILL READ ANYTHING THAT THE NEC CONTROLLER :
                        ;       RETURNS FOLLOWING AN INTERRUPT.                 :
                        ;                                                       :
                        ; ON EXIT:    @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION.:
                        ;       AX,BX,CX,DX DESTROYED                           :
                        ;--------------------------------------------------------
0AE9                    RESULTS PROC    NEAR
0AE9 57                         PUSH    DI
0AEA BF 0042 R                  MOV     DI,OFFSET @NEC_STATUS   ; POINTER TO DATA AREA
0AED B3 07                      MOV     BL,7                    ; MAX STATUS BYTES
0AEF BA 03F4                    MOV     DX,03F4H                ; STATUS PORT

                        ;-----  WAIT FOR REQUEST FOR MASTER

0AF2 B7 02              R10:    MOV     BH,2                    ; HIGH ORDER COUNTER
0AF4 33 C9                      XOR     CX,CX                   ; COUNTER
0AF6                    J39:                                    ; WAIT FOR MASTER
0AF6 EC                         IN      AL,DX                   ; GET STATUS
0AF7 24 C0                      AND     AL,11000000B            ; KEEP ONLY STATUS AND DIRECTION
0AF9 3C C0                      CMP     AL,11000000B            ; STATUS 1 AND DIRECTION 1 ?
0AFB 74 0E                      JZ      J42                     ; STATUS AND DIRECTION OK
0AFD E2 F7                      LOOP    J39                     ; LOOP TILL TIMEOUT

0AFF FE CF                      DEC     BH                      ; DECREMENT HIGH ORDER COUNTER
0B01 75 F3                      JNZ     J39                     ; REPEAT TILL DELAY DONE

0B03 80 0E 0041 R 80            OR      @DSKETTE_STATUS,TIME_OUT
0B08 F9                         STC                             ; SET ERROR RETURN
0B09 EB 1B                      JMP     SHORT POPRES            ; POP REGISTERS AND RETURN

                        ;-----  READ IN THE STATUS

0B0B                    J42:
                        ;       JMP     $+2                     ; I/O DELAY
0B0B 42                         INC     DX                      ; POINT AT DATA PORT
0B0C EC                         IN      AL,DX                   ; GET THE DATA
0B0D 88 05                      MOV     [DI],AL                 ; STORE THE BYTE
0B0F 47                         INC     DI                      ; INCREMENT THE POINTER
```

```
0B10 B9 0003                    MOV     CX,3                    ; MINIMUM 24 MICROSECONDS FOR NEC
0B13 E8 0000 E                  CALL    WAITF                   ; WAIT 30 TO 45 MICROSECONDS
0B16 4A                         DEC     DX                      ; POINT AT STATUS PORT
0B17 EC                         IN      AL,DX                   ; GET STATUS
0B18 A8 10                      TEST    AL,00010000B            ; TEST FOR NEC STILL BUSY
0B1A 74 0A                      JZ      POPRES                  ; RESULTS DONE ?

0B1C FE CB                      DEC     BL                      ; DECREMENT THE STATUS COUNTER
0B1E 75 D2                      JNZ     R10                     ; GO BACK FOR MORE
0B20 80 0E 0041 R 20            OR      @DSKETTE_STATUS,BAD_NEC ; TOO MANY STATUS BYTES
0B25 F9                         STC                             ; SET ERROR FLAG

                        ;-----  RESULT OPERATION IS DONE

                        POPRES:
0B26                            POP     DI
0B26 5F                         RET                             ; RETURN WITH CARRY SET
0B27 C3                 RESULTS ENDP
0B28                    ;--------------------------------------------------------------------
                        ; READ_DSKCHNG                                                      :
                        ;       READS THE STATE OF THE DISK CHANGE LINE.                    :
                        ;                                                                   :
                        ; ON ENTRY:     DI = DRIVE #                                       :
                        ;                                                                   :
                        ; ON EXIT:      DI = DRIVE #                                       :
                        ;               ZF = 0 ; DISK CHANGE LINE INACTIVE                 :
                        ;               ZF = 1 ; DISK CHANGE LINE ACTIVE                   :
                        ;               AX,CX,DX DESTROYED                                 :
                        ;--------------------------------------------------------------------
0B28                    READ_DSKCHNG    PROC    NEAR
0B28 E8 091A R                  CALL    MOTOR_ON                ; TURN ON THE MOTOR IF OFF
0B2B BA 03F7                    MOV     DX,03F7H                ; ADDRESS DIGITAL INPUT REGISTER
0B2E EC                         IN      AL,DX                   ; INPUT DIGITAL INPUT REGISTER
0B2F A8 80                      TEST    AL,DSK_CHG              ; CHECK FOR DISK CHANGE LINE ACTIVE
0B31 C3                         RET                             ; RETURN TO CALLER WITH ZERO FLAG SET
0B32                    READ_DSKCHNG    ENDP
                        ;--------------------------------------------------------------------
                        ; DRIVE_DET                                                         :
                        ;       DETERMINES WHETHER DRIVE IS 80 OR 40 TRACKS AND             :
                        ;       UPDATES STATE INFORMATION ACCORDINGLY.                      :
                        ;                                                                   :
                        ; ON ENTRY:     DI = DRIVE #                                       :
                        ;--------------------------------------------------------------------
0B32                    DRIVE_DET       PROC    NEAR
0B32 E8 091A R                  CALL    MOTOR_ON                ; TURN ON MOTOR IF NOT ALREADY ON
0B35 E8 0A83 R                  CALL    RECAL                   ; RECALIBRATE DRIVE
0B38 72 3C                      JC      DD_BAC                  ; ASSUME NO DRIVE PRESENT
0B3A B5 30                      MOV     CH,TRK_SLAP             ; SEEK TO TRACK 48
0B3C E8 0A24 R                  CALL    SEEK                    ; "
0B3F 72 35                      JC      DD_BAC                  ; ERROR NO DRIVE
0B41 B5 0B                      MOV     CH,QUIET_SEEK+1         ; SEEK TO TRACK 10
0B43                    SK_GIN:
0B43 FE CD                      DEC     CH                      ; DECREMENT TO NEXT TRACK
0B45 51                         PUSH    CX                      ; SAVE TRACK
0B46 E8 0A24 R                  CALL    SEEK                    ; "
0B49 72 2C                      JC      POP_BAC                 ; POP AND RETURN
0B4B B8 0B77 R                  MOV     AX,OFFSET POP_BAC       ; LOAD NEC OUTPUT ERROR ADDRESS
0B4E 50                         PUSH    AX                      ; "
0B4F B4 04                      MOV     AH,SENSE_DRV_ST         ; SENSE DRIVE STATUS COMMAND BYTE
0B51 E8 09F8 R                  CALL    NEC_OUTPUT              ; OUTPUT TO NEC
0B54 8B C7                      MOV     AX,DI                   ; AL = DRIVE
0B56 8A E0                      MOV     AH,AL                   ; AH = DRIVE
0B58 E8 09F8 R                  CALL    NEC_OUTPUT              ; OUTPUT TO NEC
0B5B E8 0AE9 R                  CALL    RESULTS                 ; GO GET STATUS
0B5E 58                         POP     AX                      ; THROW AWAY ERROR ADDRESS
0B5F 59                         POP     CX                      ; RESTORE TRACK
0B60 F6 06 0042 R 10            TEST    @NEC_STATUS,HOME        ; TRACK 0 ?
0B65 74 DC                      JZ      SK_GIN                  ; GO TILL TRACK 0
0B67 0A ED                      OR      CH,CH                   ; IS HOME AT TRACK 0 ?
0B69 74 06                      JZ      IS_80                   ; MUST BE 80 TRACK DRIVE

                        ;       DRIVE IS A 360; SET DRIVE TO DETERMINED;
                        ;       SET MEDIA TO DETERMINED AT RATE 250.

0B6B 80 8D 0090 R 94            OR      @DSK_STATE[DI],DRV_DET+MED_DET+RATE_250
0B70 C3                         RET                             ; ALL INFORMATION SET

0B71                    IS_80:
0B71 80 8D 0090 R 01            OR      @DSK_STATE[DI],TRK_CAPA ; SETUP 80 TRACK CAPABILITY
0B76                    DD_BAC:
0B76 C3                         RET

0B77                    POP_BAC:
0B77 59                         POP     CX                      ; THROW AWAY
0B78 C3                         RET

0B79                    DRIVE_DET       ENDP
                        ;--------------------------------------------------------------------
                        ; DISK_INT                                                          :
                        ;       THIS ROUTINE HANDLES THE DISKETTE INTERRUPT.                :
                        ;                                                                   :
                        ; ON EXIT:      THE INTERRUPT FLAG IS SET IN @SEEK_STATUS.          :
                        ;--------------------------------------------------------------------
0B79                    DISK_INT_1      PROC    FAR             ; ENTRY POINT FOR ORG 0EF57H
0B79 50                         PUSH    AX                      ; SAVE WORK REGISTER
0B7A 1E                         PUSH    DS                      ; SAVE REGISTERS
0B7B E8 0000 E                  CALL    DDS                     ; SETUP DATA ADDRESSING
0B7E 80 0E 003E R 80            OR      @SEEK_STATUS,INT_FLAG   ; TURN ON INTERRUPT OCCURRED
0B83 1F                         POP     DS                      ; RESTORE USER (DS)
0B84 B0 20                      MOV     AL,EOI                  ; END OF INTERRUPT MARKER
0B86 E6 20                      OUT     INTA00,AL               ; INTERRUPT CONTROL PORT
0B88 FB                         STI                             ; RE-ENABLE INTERRUPTS
0B89 B8 9101                    MOV     AX,09101H               ; INTERRUPT POST CODE AND TYPE
0B8C CD 15                      INT     15H                     ; GO PERFORM OTHER TASK
0B8E 58                         POP     AX                      ; RECOVER REGISTER
0B8F CF                         IRET                            ; RETURN FROM INTERRUPT
0B90                    DISK_INT_1      ENDP
```

```
                        PAGE
                        ;--------------------------------------------------------
                        ; DSKETTE_SETUP                                         :
                        ;       THIS ROUTINE DOES A PRELIMINARY CHECK TO SEE WHAT TYPE  :
                        ;       OF DISKETTE DRIVES ARE ATTACH TO THE SYSTEM.    :
                        ;--------------------------------------------------------
0B90                    DSKETTE_SETUP PROC      NEAR
0B90 50                         PUSH    AX                      ; SAVE REGISTERS
0B91 53                         PUSH    BX
0B92 51                         PUSH    CX
0B93 52                         PUSH    DX
0B94 57                         PUSH    DI
0B95 1E                         PUSH    DS
0B96 E8 0000 E                  CALL    DDS                     ; POINT DATA SEGMENT TO BIOS DATA AREA
0B99 80 0E 00A0 R 01            OR      @RTC_WAIT_FLAG,01       ; NO RTC WAIT, FORCE USE OF LOOP
0B9E 33 FF                      XOR     DI,DI                   ; INITIALIZE DRIVE POINTER
0BA0 C7 06 0090 R 0000          MOV     WORD PTR @DSK_STATE,0   ; INITIALIZE STATES
0BA6 80 26 008B R 33            AND     @LASTRATE,NOT_STRT_MSK+SEND_MSK ; CLEAR START & SEND
0BAB 80 0E 008B R C0            OR      @LASTRATE,SEND_MSK      ; INITIALIZE SENT TO IMPOSSIBLE
0BB0 C6 06 003E R 00            MOV     @SEEK_STATUS,0          ; INDICATE RECALIBRATE NEEDED
0BB5 C6 06 0040 R 00            MOV     @MOTOR_COUNT,0          ; INITIALIZE MOTOR COUNT
0BBA C6 06 003F R 00            MOV     @MOTOR_STATUS,0         ; INITIALIZE DRIVES TO OFF STATE
0BBF C6 06 0041 R 00            MOV     @DSKETTE_STATUS,0       ; NO ERRORS
                        ;
0BC4                    SUP0:
0BC4 E8 0B32 R                  CALL    DRIVE_DET               ; DETERMINE DRIVE
0BC7 E8 0435 R                  CALL    XLAT_OLD                ; TRANSLATE STATE TO COMPATIBLE MODE
0BCA 47                         INC     DI                      ; POINT TO NEXT DRIVE
0BCB 83 FF 02                   CMP     DI,MAX_DRV              ; SEE IF DONE
0BCE 75 F4                      JNZ     SUP0                    ; REPEAT FOR EACH DRIVE
0BD0 C6 06 003E R 00            MOV     @SEEK_STATUS,0          ; FORCE RECALIBRATE
0BD5 80 26 00A0 R FE            AND     @RTC_WAIT_FLAG,0FEH     ; ALLOW FOR RTC WAIT
0BDA E8 0854 R                  CALL    SETUP_END               ; VARIOUS CLEANUPS
0BDD 1F                         POP     DS                      ; RESTORE CALLERS RESISTERS
0BDE 5F                         POP     DI
0BDF 5A                         POP     DX
0BE0 59                         POP     CX
0BE1 5B                         POP     BX
0BE2 58                         POP     AX
0BE3 C3                         RET
0BE4                    DSKETTE_SETUP ENDP
0BE4                    CODE    ENDS
                                END
```

```
                        PAGE 118,121
                        TITLE DISK ----- 09/25/85  FIXED DISK BIOS
                        .286C
                        .LIST
0000                    CODE    SEGMENT BYTE PUBLIC

                                PUBLIC  DISK_IO
                                PUBLIC  DISK_SETUP
                                PUBLIC  HD_INT

                                EXTRN   CMOS_READ:NEAR
                                EXTRN   CMOS_WRITE:NEAR
                                EXTRN   DDS:NEAR
                                EXTRN   E_MSG:NEAR
                                EXTRN   F1780:NEAR
                                EXTRN   F1781:NEAR
                                EXTRN   F1782:NEAR
                                EXTRN   F1790:NEAR
                                EXTRN   F1791:NEAR
                                EXTRN   FD_TBL:NEAR

                        ;--- INT 13H ----------------------------------------------------------------
                        ;                                                                           :
                        ; FIXED DISK I/O INTERFACE                                                  :
                        ;                                                                           :
                        ;       THIS INTERFACE PROVIDES ACCESS TO 5 1/4" FIXED DISKS THROUGH        :
                        ;       THE IBM FIXED DISK CONTROLLER.                                      :
                        ;                                                                           :
                        ;       THE  BIOS  ROUTINES  ARE  MEANT  TO  BE  ACCESSED  THROUGH         :
                        ;       SOFTWARE  INTERRUPTS  ONLY.    ANY  ADDRESSES  PRESENT  IN         :
                        ;       THESE  LISTINGS  ARE  INCLUDED  ONLY  FOR  COMPLETENESS,            :
                        ;       NOT  FOR  REFERENCE.  APPLICATIONS  WHICH  REFERENCE  ANY           :
                        ;       ABSOLUTE  ADDRESSES  WITHIN  THE  CODE  SEGMENTS  OF  BIOS          :
                        ;       VIOLATE  THE  STRUCTURE  AND  DESIGN  OF  BIOS.                     :
                        ;                                                                           :
                        ;---------------------------------------------------------------------------:
                        ;                                                                           :
                        ; INPUT  (AH)= HEX COMMAND VALUE                                            :
                        ;                                                                           :
                        ;       (AH)= 00H  RESET DISK (DL = 80H,81H) / DISKETTE                     :
                        ;       (AH)= 01H  READ THE STATUS OF THE LAST DISK OPERATION INTO (AL)     :
                        ;                    NOTE: DL < 80H - DISKETTE                              :
                        ;                          DL > 80H - DISK                                  :
                        ;       (AH)= 02H  READ THE DESIRED SECTORS INTO MEMORY                     :
                        ;       (AH)= 03H  WRITE THE DESIRED SECTORS FROM MEMORY                    :
                        ;       (AH)= 04H  VERIFY THE DESIRED SECTORS                               :
                        ;       (AH)= 05H  FORMAT THE DESIRED TRACK                                 :
                        ;       (AH)= 06H  UNUSED                                                   :
                        ;       (AH)= 07H  UNUSED                                                   :
                        ;       (AH)= 08H  RETURN THE CURRENT DRIVE PARAMETERS                      :
                        ;       (AH)= 09H  INITIALIZE DRIVE CHARACTERISTICS                         :
                        ;                    INTERRUPT 41 POINTS TO DATA BLOCK FOR DRIVE 0          :
                        ;                    INTERRUPT 46 POINTS TO DATA BLOCK FOR DRIVE 1          :
                        ;       (AH)= 0AH  READ LONG                                                :
                        ;       (AH)= 0BH  WRITE LONG  (READ & WRITE LONG ENCOMPASS 512 + 4 BYTES ECC) :
                        ;       (AH)= 0CH  SEEK                                                     :
                        ;       (AH)= 0DH  ALTERNATE DISK RESET (SEE DL)                            :
                        ;       (AH)= 0EH  UNUSED                                                   :
                        ;       (AH)= 0FH  UNUSED                                                   :
                        ;       (AH)= 10H  TEST DRIVE READY                                         :
                        ;       (AH)= 11H  RECALIBRATE                                              :
                        ;       (AH)= 12H  UNUSED                                                   :
                        ;       (AH)= 13H  UNUSED                                                   :
                        ;       (AH)= 14H  CONTROLLER INTERNAL DIAGNOSTIC                           :
                        ;       (AH)= 15H  READ DASD TYPE                                           :
                        ;                                                                           :
                        ;---------------------------------------------------------------------------:
                        ;                                                                           :
                        ;       REGISTERS USED FOR FIXED DISK OPERATIONS                            :
                        ;                                                                           :
                        ;               (DL)    -  DRIVE NUMBER      (80H-81H FOR DISK; VALUE CHECKED) :
                        ;               (DH)    -  HEAD NUMBER       (0-15 ALLOWED, NOT VALUE CHECKED) :
                        ;               (CH)    -  CYLINDER NUMBER   (0-1023, NOT VALUE CHECKED)(SEE CL):
                        ;               (CL)    -  SECTOR NUMBER     (1-17, NOT VALUE CHECKED)      :
                        ;                                                                           :
                        ;                          NOTE: HIGH 2 BITS OF CYLINDER NUMBER ARE PLACED  :
                        ;                                IN THE HIGH 2 BITS OF THE CL REGISTER      :
                        ;                                (10 BITS TOTAL)                            :
                        ;                                                                           :
                        ;               (AL)    -  NUMBER OF SECTORS (MAXIMUM POSSIBLE RANGE 1-80H, :
                        ;                                             FOR READ/WRITE LONG 1-79H)    :
                        ;                                                                           :
                        ;               (ES:BX) -  ADDRESS OF BUFFER FOR READS AND WRITES,          :
                        ;                          (NOT REQUIRED FOR VERIFY)                        :
                        ;                                                                           :
                        ;               FORMAT (AH=5) ES:BX POINTS TO A 512 BYTE BUFFER.  THE FIRST :
                        ;                          2*(SECTORS/TRACK) BYTES CONTAIN F,N FOR EACH SECTOR.:
                        ;                          F = 00H FOR A GOOD SECTOR                        :
                        ;                              80H FOR A BAD SECTOR                         :
                        ;                          N = SECTOR NUMBER                                :
                        ;                          FOR AN INTERLEAVE OF 2 AND 17 SECTORS/TRACK      :
                        ;                          THE TABLE SHOULD BE:                             :
                        ;                                                                           :
                        ;                            DB    00H,01H,00H,0AH,00H,02H,00H,0BH,00H,03H,00H,0CH :
                        ;                            DB    00H,04H,00H,0DH,00H,05H,00H,0EH,00H,06H,00H,0FH :
                        ;                            DB    00H,07H,00H,10H,00H,08H,00H,11H,00H,09H  :
                        ;                                                                           :
                        ;---------------------------------------------------------------------------:
```

```
                        PAGE
                ;--------------------------------------------------------------------------------
                ; OUTPUT                                                                        :
                ;       AH = STATUS OF CURRENT OPERATION                                        :
                ;            STATUS BITS ARE DEFINED IN THE EQUATES BELOW                       :
                ;       CY = 0  SUCCESSFUL OPERATION (AH=0 ON RETURN)                           :
                ;       CY = 1  FAILED OPERATION (AH HAS ERROR REASON)                          :
                ;                                                                               :
                ;       NOTE:   ERROR 11H  INDICATES THAT THE DATA READ HAD A RECOVERABLE       :
                ;               ERROR WHICH WAS CORRECTED BY THE ECC ALGORITHM.  THE DATA       :
                ;               IS PROBABLY GOOD,   HOWEVER THE BIOS ROUTINE INDICATES AN       :
                ;               ERROR TO ALLOW THE CONTROLLING PROGRAM A CHANCE TO DECIDE       :
                ;               FOR  ITSELF.  THE  ERROR  MAY  NOT  RECUR  IF THE DATA IS      :
                ;               REWRITTEN.                                                      :
                ;                                                                               :
                ;       IF DRIVE PARAMETERS WERE REQUESTED (DL >= 80H),                         :
                ;          INPUT:                                                               :
                ;            (DL) = DRIVE NUMBER                                                :
                ;          OUTPUT:                                                              :
                ;            (DL) = NUMBER OF CONSECUTIVE ACKNOWLEDGING DRIVES ATTACHED (1-2)   :
                ;                   (CONTROLLER CARD ZERO TALLY ONLY)                           :
                ;            (DH) = MAXIMUM USEABLE VALUE FOR HEAD NUMBER                       :
                ;            (CH) = MAXIMUM USEABLE VALUE FOR CYLINDER NUMBER                   :
                ;            (CL) = MAXIMUM USEABLE VALUE FOR SECTOR NUMBER                     :
                ;                   AND CYLINDER NUMBER HIGH BITS                               :
                ;                                                                               :
                ;       IF READ DASD TYPE WAS REQUESTED,                                        :
                ;                                                                               :
                ;       AH = 0 - NOT PRESENT                                                    :
                ;            1 - DISKETTE - NO CHANGE LINE AVAILABLE                            :
                ;            2 - DISKETTE - CHANGE LINE AVAILABLE                               :
                ;            3 - FIXED DISK                                                     :
                ;       CX,DX = NUMBER OF 512 BYTE BLOCKS WHEN AH = 3                           :
                ;                                                                               :
                ;       REGISTERS WILL BE PRESERVED EXCEPT WHEN THEY ARE USED TO RETURN         :
                ;       INFORMATION.                                                            :
                ;                                                                               :
                ;       NOTE: IF AN ERROR IS REPORTED BY THE DISK CODE, THE APPROPRIATE         :
                ;             ACTION IS TO RESET THE DISK, THEN RETRY THE OPERATION.            :
                ;                                                                               :
                ;--------------------------------------------------------------------------------

= 00FF          SENSE_FAIL      EQU     0FFH            ; NOT IMPLEMENTED
= 00E0          NO_ERR          EQU     0E0H            ; STATUS ERROR/ERROR REGISTER=0
= 00CC          WRITE_FAULT     EQU     0CCH            ; WRITE FAULT ON SELECTED DRIVE
= 00BB          UNDEF_ERR       EQU     0BBH            ; UNDEFINED ERROR OCCURRED
= 00AA          NOT_RDY         EQU     0AAH            ; DRIVE NOT READY
= 0080          TIME_OUT        EQU     80H             ; ATTACHMENT FAILED TO RESPOND
= 0040          BAD_SEEK        EQU     40H             ; SEEK OPERATION FAILED
= 0020          BAD_CNTLR       EQU     20H             ; CONTROLLER HAS FAILED
= 0011          DATA_CORRECTED  EQU     11H             ; ECC CORRECTED DATA ERROR
= 0010          BAD_ECC         EQU     10H             ; BAD ECC ON DISK READ
= 000B          BAD_TRACK       EQU     0BH             ; NOT IMPLEMENTED
= 000A          BAD_SECTOR      EQU     0AH             ; BAD SECTOR FLAG DETECTED
= 0009          DMA_BOUNDARY    EQU     09H             ; DATA EXTENDS TOO FAR
= 0007          INIT_FAIL       EQU     07H             ; DRIVE PARAMETER ACTIVITY FAILED
= 0006          BAD_RESET       EQU     06H             ; RESET FAILED
= 0004          RECORD_NOT_FND  EQU     04H             ; REQUESTED SECTOR NOT FOUND
= 0002          BAD_ADDR_MARK   EQU     02H             ; ADDRESS MARK NOT FOUND
= 0001          BAD_CMD         EQU     01H             ; BAD COMMAND PASSED TO DISK I/O


                ;--------------------------------------------------
                ;                                                  :
                ; FIXED DISK PARAMETER TABLE                       :
                ;                                                  :
                ;  -  THE TABLE IS COMPOSED OF A BLOCK DEFINED AS: :
                ;                                                  :
                ;  +0   (1 WORD) - MAXIMUM NUMBER OF CYLINDERS     :
                ;  +2   (1 BYTE) - MAXIMUM NUMBER OF HEADS         :
                ;  +3   (1 WORD) - NOT USED/SEE PC-XT              :
                ;  +5   (1 WORD) - STARTING WRITE PRECOMPENSATION CYL :
                ;  +7   (1 BYTE) - MAXIMUM ECC DATA BURST LENGTH   :
                ;  +8   (1 BYTE) - CONTROL BYTE                    :
                ;                  BIT    7 DISABLE RETRIES -OR-   :
                ;                  BIT    6 DISABLE RETRIES        :
                ;                  BIT    3 MORE THAN 8 HEADS      :
                ;  +9   (3 BYTES)- NOT USED/SEE PC-XT              :
                ; +12   (1 WORD) - LANDING ZONE                    :
                ; +14   (1 BYTE) - NUMBER OF SECTORS/TRACK         :
                ; +15   (1 BYTE) - RESERVED FOR FUTURE USE         :
                ;                                                  :
                ;       - TO DYNAMICALLY DEFINE A SET OF PARAMETERS :
                ;         BUILD A TABLE FOR UP TO 15 TYPES AND PLACE :
                ;         THE CORRESPONDING VECTOR INTO INTERRUPT 41 :
                ;         FOR DRIVE 0 AND INTERRUPT 46 FOR DRIVE 1. :
                ;                                                  :
                ;--------------------------------------------------
```

```
                        PAGE
                        ;--------------------------------------------------------
                        ;                                                       :
                        ; HARDWARE SPECIFIC VALUES                             :
                        ;                                                       :
                        ;  -  CONTROLLER I/O PORT                              :
                        ;                                                       :
                        ;     > WHEN READ FROM:                                 :
                        ;       HF_PORT+0 - READ DATA (FROM CONTROLLER TO CPU)  :
                        ;       HF_PORT+1 - GET ERROR REGISTER                  :
                        ;       HF_PORT+2 - GET SECTOR COUNT                    :
                        ;       HF_PORT+3 - GET SECTOR NUMBER                   :
                        ;       HF_PORT+4 - GET CYLINDER LOW                    :
                        ;       HF_PORT+5 - GET CYLINDER HIGH (2 BITS)          :
                        ;       HF_PORT+6 - GET SIZE/DRIVE/HEAD                 :
                        ;       HF_PORT+7 - GET STATUS REGISTER                 :
                        ;                                                       :
                        ;     > WHEN WRITTEN TO:                                :
                        ;       HF_PORT+0 - WRITE DATA (FROM CPU TO CONTROLLER) :
                        ;       HF_PORT+1 - SET PRECOMPENSATION CYLINDER        :
                        ;       HF_PORT+2 - SET SECTOR COUNT                    :
                        ;       HF_PORT+3 - SET SECTOR NUMBER                   :
                        ;       HF_PORT+4 - SET CYLINDER LOW                    :
                        ;       HF_PORT+5 - SET CYLINDER HIGH (2 BITS)          :
                        ;       HF_PORT+6 - SET SIZE/DRIVE/HEAD                 :
                        ;       HF_PORT+7 - SET COMMAND REGISTER                :
                        ;                                                       :
                        ;--------------------------------------------------------

= 01F0                  HF_PORT         EQU     01F0H           ; DISK PORT
= 03F6                  HF_REG_PORT     EQU     03F6H

                        ;-----          STATUS REGISTER

= 0001                  ST_ERROR        EQU     00000001B       ;
= 0002                  ST_INDEX        EQU     00000010B       ;
= 0004                  ST_CORRCTD      EQU     00000100B       ; ECC CORRECTION SUCCESSFUL
= 0008                  ST_DRQ          EQU     00001000B       ;
= 0010                  ST_SEEK_COMPL   EQU     00010000B       ; SEEK COMPLETE
= 0020                  ST_WRT_FLT      EQU     00100000B       ; WRITE FAULT
= 0040                  ST_READY        EQU     01000000B       ;
= 0080                  ST_BUSY         EQU     10000000B       ;

                        ;-----          ERROR REGISTER

= 0001                  ERR_DAM         EQU     00000001B       ; DATA ADDRESS MARK NOT FOUND
= 0002                  ERR_TRK_0       EQU     00000010B       ; TRACK 0 NOT FOUND ON RECAL
= 0004                  ERR_ABORT       EQU     00000100B       ; ABORTED COMMAND
                        ;               EQU     00001000B       ; NOT USED
= 0010                  ERR_ID          EQU     00010000B       ; ID NOT FOUND
                        ;               EQU     00100000B       ; NOT USED
= 0040                  ERR_DATA_ECC    EQU     01000000B
= 0080                  ERR_BAD_BLOCK   EQU     10000000B

= 0010                  RECAL_CMD       EQU     00010000B       ; DRIVE RECAL   (10H)
= 0020                  READ_CMD        EQU     00100000B       ;       READ    (20H)
= 0030                  WRITE_CMD       EQU     00110000B       ;       WRITE   (30H)
= 0040                  VERIFY_CMD      EQU     01000000B       ;       VERIFY  (40H)
= 0050                  FMTTRK_CMD      EQU     01010000B       ; FORMAT TRACK  (50H)
= 0060                  INIT_CMD        EQU     01100000B       ;   INITIALIZE  (60H)
= 0070                  SEEK_CMD        EQU     01110000B       ;       SEEK    (70H)
= 0090                  DIAG_CMD        EQU     10010000B       ; DIAGNOSTIC    (90H)
= 0091                  SET_PARM_CMD    EQU     10010001B       ; DRIVE PARMS   (91H)
= 0001                  NO_RETRIES      EQU     00000001B       ; CMD MODIFIER  (01H)
= 0002                  ECC_MODE        EQU     00000010B       ; CMD MODIFIER  (02H)
= 0008                  BUFFER_MODE     EQU     00001000B       ; CMD MODIFIER  (08H)

= 0002                  MAX_FILE        EQU     2
= 0002                  S_MAX_FILE      EQU     2

= 0025                  DELAY_1         EQU     25H             ; DELAY FOR OPERATION COMPLETE
= 0600                  DELAY_2         EQU     0600H           ; DELAY FOR READY
= 0100                  DELAY_3         EQU     0100H           ; DELAY FOR DATA REQUEST

= 0008                  HF_FAIL         EQU     08H             ; CMOS FLAG IN BYTE 0EH

                        ;-----          COMMAND BLOCK REFERENCE

=                       @CMD_BLOCK      EQU     BYTE PTR [BP]-8 ; @CMD_BLOCK REFERENCES BLOCK HEAD IN SS
                                                                ;  (BP) POINTS TO COMMAND BLOCK TAIL
                                                                ;       AS DEFINED BY THE "ENTER" PARMS
```

SECTION 5

```
                        PAGE
                        ;--------------------------------------------------------
                        ; FIXED DISK I/O SETUP                                  :
                        ;                                                      :
                        ;   -  ESTABLISH TRANSFER VECTORS FOR THE FIXED DISK    :
                        ;   -  PERFORM POWER ON DIAGNOSTICS                     :
                        ;      SHOULD AN ERROR OCCUR A "1701" MESSAGE IS DISPLAYED :
                        ;                                                      :
                        ;--------------------------------------------------------
                                ASSUME  CS:CODE,DS:ABS0                         ; WORK OFF DS REGISTER

0000                    DISK_SETUP      PROC    NEAR
0000 FA                         CLI
0001 B8 ---- R                  MOV     AX,ABS0                                 ; GET ABSOLUTE SEGMENT
0004 8E D8                      MOV     DS,AX                                   ; SET SEGMENT REGISTER
0006 A1 004C R                  MOV     AX,WORD PTR @ORG_VECTOR                 ; GET DISKETTE VECTOR
0009 A3 0100 R                  MOV     WORD PTR @DISK_VECTOR,AX                ;  INTO INT 40H
000C A1 004E R                  MOV     AX,WORD PTR @ORG_VECTOR+2
000F A3 0102 R                  MOV     WORD PTR @DISK_VECTOR+2,AX
0012 C7 06 004C R 01A9 R        MOV     WORD PTR @ORG_VECTOR,OFFSET DISK_IO     ; FIXED DISK HANDLER
0018 8C 0E 004E R               MOV     WORD PTR @ORG_VECTOR+2,CS
001C C7 06 01D8 R 06DA R        MOV     WORD PTR @HDISK_INT,OFFSET HD_INT       ; FIXED DISK INTERRUPT
0022 8C 0E 01DA R               MOV     WORD PTR @HDISK_INT+2,CS
0026 C7 06 0104 R 0000 E        MOV     WORD PTR @HF_TBL_VEC,OFFSET FD_TBL      ; PARM TABLE DRIVE 80
002C 8C 0E 0106 R               MOV     WORD PTR @HF_TBL_VEC+2,CS
0030 C7 06 0118 R 0000 E        MOV     WORD PTR @HFT_TBL_VEC,OFFSET FD_TBL     ; PARM TABLE DRIVE 81
0036 8C 0E 011A R               MOV     WORD PTR @HF1_TBL_VEC+2,CS
003A E4 A1                      IN      AL,INTB01                               ; TURN ON SECOND INTERRUPT CHIP
003C 24 BF                      AND     AL,0BFH
003E EB 00                      JMP     $+2
0040 E6 A1                      OUT     INTB01,AL
0042 E4 21                      IN      AL,INTA01               ; LET INTERRUPTS PASS THRU TO
0044 24 FB                      AND     AL,0FBH                 ;  SECOND CHIP
0046 EB 00                      JMP     $+2
0048 E6 21                      OUT     INTA01,AL

004A FB                         STI
                                ASSUME  DS:DATA,ES:ABS0
004B 1E                         PUSH    DS                      ; MOVE ABS0 POINTER TO
004C 07                         POP     ES                      ; EXTRA SEGMENT POINTER
004D E8 0000 E                  CALL    DDS                     ; ESTABLISH DATA SEGMENT
0050 C6 06 0074 R 00            MOV     @DISK_STATUS1,0         ; RESET THE STATUS INDICATOR
0055 C6 06 0075 R 00            MOV     @HF_NUM,0               ; ZERO NUMBER OF FIXED DISKS
005A C6 06 0076 R 00            MOV     @CONTROL_BYTE,0
005F B0 8E                      MOV     AL,CMOS_DIAG+NMI
0061 E8 0000 E                  CALL    CMOS_READ               ; CHECK CMOS VALIDITY
0064 8A E0                      MOV     AH,AL                   ; SAVE CMOS FLAG
0066 24 C0                      AND     AL,BAD_BAT+BAD_CKSUM    ; CHECK FOR VALID CMOS
0068 74 03                      JZ      L1
006A E9 00F8 R                  JMP     POD_DONE                ; CMOS NOT VALID -- NO FIXED DISKS
006D                    L1:
006D 80 E4 F7                   AND     AH,NOT HF_FAIL          ; ALLOW FIXED DISK IPL
0070 B0 8E                      MOV     AL,CMOS_DIAG+NMI        ; WRITE IT BACK
0072 E8 0000 E                  CALL    CMOS_WRITE
0075 B0 92                      MOV     AL,CMOS_DISK+NMI
0077 E8 0000 E                  CALL    CMOS_READ
007A C6 06 0077 R 00            MOV     @PORT_OFF,0             ; ZERO CARD OFFSET
007F 8A D8                      MOV     BL,AL                   ; SAVE FIXED DISK BYTE
0081 25 00F0                    AND     AX,000F0H               ; GET FIRST DRIVE TYPE AS OFFSET
0084 74 72                      JZ      POD_DONE                ; NO FIXED DISKS

0086 3C F0                      CMP     AL,0F0H                 ; CHECK FOR EXTENDED DRIVE TYPE BYTE USE
0088 75 10                      JNE     L2                      ; USE DRIVE TYPE 1 --> 14 IF NOT IN USE

008A B0 99                      MOV     AL,CMOS_DISK_1+NMI      ; GET EXTENDED TYPE FOR DRIVE C:
008C E8 0000 E                  CALL    CMOS_READ               ; FROM CMOS
008F 3C 00                      CMP     AL,0                    ; IS TYPE SET TO ZERO
0091 74 65                      JE      POD_DONE                ; EXIT IF NOT VALID AND NO FIXED DISKS
0093 3C 2F                      CMP     AL,47                   ; IS TYPE WITHIN VALID RANGE
0095 77 61                      JA      POD_DONE                ; EXIT WITH NO FIXED DISKS IF NOT VALID
0097 C1 E0 04                   SHL     AX,4                    ; ADJUST TYPE TO HIGH NIBBLE
009A                    L2:
009A 05 FFF0 E                  ADD     AX,OFFSET FD_TBL-16D    ; COMPUTE OFFSET OF FIRST DRIVE TABLE
009D 26: A3 0104 R              MOV     WORD PTR @HF_TBL_VEC,AX ; SAVE IN VECTOR POINTER
00A1 C6 06 0075 R 01            MOV     @HF_NUM,1               ; AT LEAST ONE DRIVE
00A6 8A C3                      MOV     AL,BL
00A8 C0 E0 04                   SHL     AL,4                    ; GET SECOND DRIVE TYPE
00AB 74 2A                      JZ      SHORT L4                ; ONLY ONE DRIVE
00AD B4 00                      MOV     AH,0

00AF 3C F0                      CMP     AL,0F0H                 ; CHECK FOR EXTENDED DRIVE TYPE BYTE USE
00B1 75 10                      JNE     L3                      ; USE DRIVE TYPE 1 --> 14 IF NOT IN USE

00B3 B0 9A                      MOV     AL,CMOS_DISK_2+NMI      ; GET EXTENDED TYPE FOR DRIVE D:
00B5 E8 0000 E                  CALL    CMOS_READ               ; FROM CMOS
00B8 3C 00                      CMP     AL,0                    ; IS TYPE SET TO ZERO
00BA 74 1B                      JE      L4                      ; SKIP IF SECOND FIXED DISK NOT VALID
00BC 3C 2F                      CMP     AL,47                   ; IS TYPE WITHIN VALID RANGE
00BE 77 17                      JA      L4                      ; SKIP IF NOT VALID
00C0 C1 E0 04                   SHL     AX,4                    ; ADJUST TYPE TO HIGH NIBBLE
00C3                    L3:
00C3 05 FFF0 E                  ADD     AX,OFFSET FD_TBL-16D    ; COMPUTE OFFSET FOR SECOND FIXED DISK
00C6 8B D8                      MOV     BX,AX
00C8 2E: 83 3F 00               CMP     WORD PTR CS:[BX],0      ; CHECK FOR ZERO CYLINDERS IN TABLE
00CC 74 09                      JE      L4                      ; SKIP DRIVE IF NOT A VALID TABLE ENTRY
00CE 26: A3 0118 R              MOV     WORD PTR @HF1_TBL_VEC,AX
00D2 C6 06 0075 R 02            MOV     @HF_NUM,2               ; TWO DRIVES
00D7                    L4:
00D7 B2 80                      MOV     DL,80H                  ; CHECK THE CONTROLLER
00D9 B4 14                      MOV     AH,14H                  ; USE CONTROLLER DIAGNOSTIC COMMAND
00DB CD 13                      INT     13H                     ; CALL BIOS WITH DIAGNOSTIC COMMAND
00DD 72 1A                      JC      CTL_ERRX                ;  DISPLAY ERROR MESSAGE IF BAD RETURN
00DF A1 006C R                  MOV     AX,@TIMER_LOW           ; GET START TIMER COUNTS
00E2 8B D8                      MOV     BX,AX
00E4 05 0444                    ADD     AX,6*182                ; 60 SECONDS* 18.2
00E7 8B C8                      MOV     CX,AX
00E9 E8 0104 R                  CALL    HD_RESET_1              ; SET UP DRIVE 0
00EC 80 3E 0075 R 01            CMP     @HF_NUM,1               ; WERE THERE TWO DRIVES?
00F1 76 05                      JBE     POD_DONE                ; NO-ALL DONE
00F3 B2 81                      MOV     DL,81H                  ; SET UP DRIVE 1
00F5 E8 0104 R                  CALL    HD_RESET_1
00F8                    POD_DONE:
00F8 C3                         RET
```

```
                        ;----- POD ERROR

00F9                    CTL_ERRX:
00F9 BE 0000 E                  MOV     SI,OFFSET F1782         ; CONTROLLER ERROR
00FC E8 017C R                  CALL    SET_FAIL                ; DO NOT IPL FROM DISK
00FF E8 0000 E                  CALL    E_MSG                   ; DISPLAY ERROR AND SET (BP) ERROR FLAG
0102 EB F4                      JMP     POD_DONE


0104                    HD_RESET_1      PROC    NEAR
0104 53                         PUSH    BX                      ; SAVE TIMER LIMITS
0105 51                         PUSH    CX
0106 B4 09              RES_1:  MOV     AH,09H                  ; SET DRIVE PARAMETERS
0108 CD 13                      INT     13H
010A 72 06                      JC      RES_2
010C B4 11                      MOV     AH,11H                  ; RECALIBRATE DRIVE
010E CD 13                      INT     13H
0110 73 19                      JNC     RES_CK                  ; DRIVE OK
0112 E8 018A R          RES_2:  CALL    POD_TCHK                ; CHECK TIME OUT
0115 73 EF                      JNC     RES_1
0117 BE 0000 E          RES_FL: MOV     SI,OFFSET F1781         ; INDICATE DISK 1 FAILURE
011A F6 C2 01                   TEST    DL,1
011D 75 57                      JNZ     RES_E1
011F BE 0000 E                  MOV     SI,OFFSET F1780         ; INDICATE DISK 0 FAILURE
0122 E8 017C R                  CALL    SET_FAIL                ; DO NOT TRY TO IPL DISK 0
0125 EB 4F                      JMP     SHORT RES_E1
0127 B4 00              RES_RS: MOV     AH,00H                  ; RESET THE DRIVE
0129 CD 13                      INT     13H
012B B4 08              RES_CK: MOV     AH,08H                  ; GET MAX CYLINDER,HEAD,SECTOR
012D 8A DA                      MOV     BL,DL                   ; SAVE DRIVE CODE
012F CD 13                      INT     13H
0131 72 38                      JC      RES_ER
0133 89 0E 0042 R               MOV     WORD PTR @NEC_STATUS,CX ; SAVE MAX CYLINDER, SECTOR
0137 8A D3                      MOV     DL,BL                   ; RESTORE DRIVE CODE
0139 B8 0401            RES_3:  MOV     AX,0401H                ; VERIFY THE LAST SECTOR
013C CD 13                      INT     13H
013E 73 39                      JNC     RES_OK                  ; VERIFY OK
0140 80 FC 0A                   CMP     AH,BAD_SECTOR           ; OK ALSO IF JUST ID READ
0143 74 34                      JE      RES_OK
0145 80 FC 11                   CMP     AH,DATA_CORRECTED
0148 74 2F                      JE      RES_OK
014A 80 FC 10                   CMP     AH,BAD_ECC
014D 74 2A                      JE      RES_OK
014F E8 018A R                  CALL    POD_TCHK                ; CHECK FOR TIME OUT
0152 72 17                      JC      RES_ER                  ; FAILED
0154 8B 0E 0042 R               MOV     CX,WORD PTR @NEC_STATUS ; GET SECTOR ADDRESS, AND CYLINDER
0158 8A C1                      MOV     AL,CL                   ; SEPARATE OUT SECTOR NUMBER
015A 24 3F                      AND     AL,3FH
015C FE C8                      DEC     AL                      ; TRY PREVIOUS ONE
015E 74 C7                      JZ      RES_RS                  ; WE'VE TRIED ALL SECTORS ON TRACK
0160 80 E1 C0                   AND     CL,0C0H                 ; KEEP CYLINDER BITS
0163 0A C8                      OR      CL,AL                   ; MERGE SECTOR WITH CYLINDER BITS
0165 89 0E 0042 R               MOV     WORD PTR @NEC_STATUS,CX ; SAVE CYLINDER, NEW SECTOR NUMBER
0169 EB CE                      JMP     RES_3                   ; TRY AGAIN
016B BE 0000 E          RES_ER: MOV     SI,OFFSET F1791         ; INDICATE DISK 1 ERROR
016E F6 C2 01                   TEST    DL,1
0171 75 03                      JNZ     RES_E1
0173 BE 0000 E                  MOV     SI,OFFSET F1790         ; INDICATE DISK 0 ERROR
0176                    RES_E1:
0176 E8 0000 E                  CALL    E_MSG                   ; DISPLAY ERROR AND SET (BP) ERROR FLAG
0179                    RES_OK:
0179 59                         POP     CX                      ; RESTORE TIMER LIMITS
017A 5B                         POP     BX
017B C3                         RET
017C                    HD_RESET_1      ENDP

017C                    SET_FAIL        PROC    NEAR
017C B8 8E8E                    MOV     AX,X*(CMOS_DIAG+NMI)    ; GET CMOS ERROR BYTE
017F E8 0000 E                  CALL    CMOS_READ
0182 0C 08                      OR      AL,HF_FAIL              ; SET DO_NOT IPL FROM DISK FLAG
0184 86 E0                      XCHG    AH,AL                   ; SAVE IT
0186 E8 0000 E                  CALL    CMOS_WRITE              ; PUT IT OUT
0189 C3                         RET
018A                    SET_FAIL        ENDP

018A                    POD_TCHK        PROC NEAR               ; CHECK FOR 30 SECOND TIME OUT
018A 58                         POP     AX                      ; SAVE RETURN
018B 59                         POP     CX                      ; GET TIME OUT LIMITS
018C 5B                         POP     BX
018D 53                         PUSH    BX                      ; AND SAVE THEM AGAIN
018E 51                         PUSH    CX
018F 50                         PUSH    AX                      ; RESTORE RETURN
0190 A1 006C R                  MOV     AX,@TIMER_LOW           ; AX = CURRENT TIME
                                                                ; BX = START TIME
                                                                ; CX = END TIME

0193 3B D9                      CMP     BX,CX
0195 72 06                      JB      TCHK1                   ; START < END
0197 3B D8                      CMP     BX,AX
0199 72 0C                      JB      TCHKG                   ; END < START < CURRENT
019B EB 04                      JMP     SHORT TCHK2             ; END, CURRENT < START
019D 3B C3              TCHK1:  CMP     AX,BX
019F 72 04                      JB      TCHKNG                  ; CURRENT < START < END
01A1 3B C1              TCHK2:  CMP     AX,CX
01A3 72 02                      JB      TCHKG                   ; START < CURRENT < END
                                                                ; OR CURRENT < END < START
01A5 F9                 TCHKNG: STC                             ; CARRY SET INDICATES TIME OUT
01A6 C3                         RET
01A7 F8                 TCHKG:  CLC                             ; INDICATE STILL TIME
01A8 C3                         RET
01A9                    POD_TCHK        ENDP

01A9                    DISK_SETUP      ENDP
```

SECTION 5

```
                        PAGE
                        ;--------------------------------------
                        ;       FIXED DISK BIOS ENTRY POINT     :
                        ;--------------------------------------

01A9                    DISK_IO PROC    FAR
                                ASSUME  DS:DATA,ES:NOTHING
01A9 80 FA 80                   CMP     DL,80H                  ; TEST FOR FIXED DISK DRIVE
01AC 73 05                      JAE     A1                      ; YES, HANDLE HERE
01AE CD 40                      INT     40H                     ; DISKETTE HANDLER
01B0                    RET_2:
01B0 CA 0002                    RET     2                       ; BACK TO CALLER

01B3                    A1:
01B3 FB                         STI                             ; ENABLE INTERRUPTS
01B4 0A E4                      OR      AH,AH
01B6 75 09                      JNZ     A2
01B8 CD 40                      INT     40H                     ; RESET NEC WHEN AH=0
01BA 2A E4                      SUB     AH,AH
01BC 80 FA 81                   CMP     DL,(80H + S_MAX_FILE - 1)
01BF 77 EF                      JA      RET_2
01C1                    A2:
01C1 80 FC 08                   CMP     AH,08H                  ; GET PARAMETERS IS A SPECIAL CASE
01C4 75 03                      JNZ     A3
01C6 E9 0393 R                  JMP     GET_PARM_N
01C9 80 FC 15           A3:     CMP     AH,15H                  ; READ DASD TYPE IS ALSO
01CC 75 03                      JNZ     A4
01CE E9 0353 R                  JMP     READ_DASD_TYPE

01D1                    A4:
01D1 C8 0008 00                 ENTER   8,0                     ;       SAVE REGISTERS DURING OPERATION
01D5 53                         PUSH    BX                      ; SAVE (BP) AND MAKE ROOM FOR @CMD_BLOCK
01D6 51                         PUSH    CX                      ;  IN THE STACK.  THE COMMAND_BLOCK IS:
01D7 52                         PUSH    DX                      ;   @CMD_BLOCK == BYTE PTR [BP]-8
01D8 1E                         PUSH    DS
01D9 06                         PUSH    ES
01DA 56                         PUSH    SI
01DB 57                         PUSH    DI
01DC 0A E4                      OR      AH,AH                   ; CHECK FOR RESET
01DE 75 02                      JNZ     A5
01E0 B2 80                      MOV     DL,80H                  ; FORCE DRIVE 80 FOR RESET
01E2 E8 0228 R          A5:     CALL    DISK_IO_CONT            ; PERFORM THE OPERATION
01E5 E8 0000 E                  CALL    DDS                     ; ESTABLISH SEGMENT
01E8 8A 26 0074 R               MOV     AH,@DISK_STATUS1        ; GET STATUS FROM OPERATION
01EC 80 FC 01                   CMP     AH,1                    ; SET THE CARRY FLAG TO INDICATE
01EF F5                         CMC                             ;  SUCCESS OR FAILURE
01F0 5F                         POP     DI                      ; RESTORE REGISTERS
01F1 5E                         POP     SI
01F2 07                         POP     ES
01F3 1F                         POP     DS
01F4 5A                         POP     DX
01F5 59                         POP     CX
01F6 5B                         POP     BX
01F7 C9                         LEAVE
01F8 CA 0002                    RET     2                       ; ADJUST (SP) AND RESTORE (BP)
01FB                    DISK_IO ENDP                            ; THROW AWAY SAVED FLAGS

01FB                    MI      LABEL   WORD                    ; FUNCTION TRANSFER TABLE
01FB 02C1 R                     DW      DISK_RESET              ; 000H
01FD 0315 R                     DW      RETURN_STATUS           ; 001H
01FF 031E R                     DW      DISK_READ               ; 002H
0201 0325 R                     DW      DISK_WRITE              ; 003H
0203 032C R                     DW      DISK_VERF               ; 004H
0205 033E R                     DW      FMT_TRK                 ; 005H
0207 02B9 R                     DW      BAD_COMMAND             ; 006H  FORMAT BAD SECTORS
0209 02B9 R                     DW      BAD_COMMAND             ; 007H  FORMAT DRIVE
020B 02B9 R                     DW      BAD_COMMAND             ; 008H  RETURN PARAMETERS
020D 03F1 R                     DW      INIT_DRV                ; 009H
020F 0423 R                     DW      RD_LONG                 ; 00AH
0211 042A R                     DW      WR_LONG                 ; 00BH
0213 0431 R                     DW      DISK_SEEK               ; 00CH
0215 02C1 R                     DW      DISK_RESET              ; 00DH
0217 02B9 R                     DW      BAD_COMMAND             ; 00EH  READ BUFFER
0219 02B9 R                     DW      BAD_COMMAND             ; 00FH  WRITE BUFFER
021B 044F R                     DW      TST_RDY                 ; 010H
021D 0466 R                     DW      HDISK_RECAL             ; 011H
021F 02B9 R                     DW      BAD_COMMAND             ; 012H  MEMORY DIAGNOSTIC
0221 02B9 R                     DW      BAD_COMMAND             ; 013H  DRIVE DIAGNOSTIC
0223 048E R                     DW      CTLR_DIAGNOSTIC         ; 014H  CONTROLLER DIAGNOSTIC
                        = 002A  MIL     EQU     $-MI

0226                    DISK_IO_CONT    PROC    NEAR
0226 E8 0000 E                  CALL    DDS                     ; ESTABLISH SEGMENT
0228 80 FC 01                   CMP     AH,01H                  ; RETURN STATUS
022B 75 03                      JNZ     SU0
022D E9 0315 R                  JMP     RETURN_STATUS
0230                    SU0:
0230 C6 06 0074 R 00            MOV     @DISK_STATUS1,0         ; RESET THE STATUS INDICATOR
0235 53                         PUSH    BX                      ; SAVE DATA ADDRESS
0236 8A 1E 0075 R               MOV     BL,@HF_NUM              ; GET NUMBER OF DRIVES
023A 50                         PUSH    AX
023B 80 E2 7F                   AND     DL,7FH                  ; GET DRIVE AS 0 OR 1
023E 3A DA                      CMP     BL,DL
0240 76 75                      JBE     BAD_COMMAND_POP         ; INVALID DRIVE
0242 06                         PUSH    ES
0243 E8 06C4 R                  CALL    GET_VEC                 ; GET DISK PARAMETERS
0246 26: 8B 47 05               MOV     AX,WORD PTR ES:[BX][5]  ; GET WRITE PRE-COMPENSATION CYLINDER
024A C1 E8 02                   SHR     AX,2
024D 88 46 F8                   MOV     @CMD_BLOCK,AL
0250 26: 8A 47 08               MOV     AL,BYTE PTR ES:[BX][8]  ; GET CONTROL BYTE MODIFIER
0254 52                         PUSH    DX
0255 BA 03F6                    MOV     DX,HF_REG_PORT
0258 EE                         OUT     DX,AL                   ; SET EXTRA HEAD OPTION
0259 5A                         POP     DX
025A 07                         POP     ES
025B 8A 26 0076 R               MOV     AH,@CONTROL_BYTE        ; SET EXTRA HEAD OPTION IN
025F 80 E4 C0                   AND     AH,0C0H                 ; CONTROL BYTE
0262 0A E0                      OR      AH,AL
0264 88 26 0076 R               MOV     @CONTROL_BYTE,AH
0268 58                         POP     AX
0269 88 46 F9                   MOV     @CMD_BLOCK+1,AL         ; SECTOR COUNT
026C 50                         PUSH    AX
026D 8A C1                      MOV     AL,CL                   ; GET SECTOR NUMBER
026F 24 3F                      AND     AL,3FH
```

```
0271 88 46 FA                   MOV     @CMD_BLOCK+2,AL
0274 88 6E FB                   MOV     @CMD_BLOCK+3,CH         ; GET CYLINDER NUMBER
0277 8A C1                      MOV     AL,CL
0279 C0 E8 06                   SHR     AL,6
027C 88 46 FC                   MOV     @CMD_BLOCK+4,AL         ; CYLINDER HIGH ORDER 2 BITS
027F 8A C2                      MOV     AL,DL                   ; DRIVE NUMBER
0281 C0 E0 04                   SHL     AL,4
0284 80 E6 0F                   AND     DH,0FH                  ; HEAD NUMBER
0287 0A C6                      OR      AL,DH
0289 0C A0                      OR      AL,80H OR 20H           ; ECC AND 512 BYTE SECTORS
028B 88 46 FD                   MOV     @CMD_BLOCK+5,AL         ; ECC/SIZE/DRIVE/HEAD
028E 58                         POP     AX
028F 50                         PUSH    AX
0290 8A C4                      MOV     AL,AH                   ; GET INTO LOW BYTE
0292 32 E4                      XOR     AH,AH                   ; ZERO HIGH BYTE
0294 D1 E0                      SAL     AX,1                    ; *2 FOR TABLE LOOKUP
0296 8B F0                      MOV     SI,AX                   ; PUT INTO SI FOR BRANCH
0298 3D 002A                    CMP     AX,MIL                  ; TEST WITHIN RANGE
029B 73 1A                      JNB     BAD_COMMAND_POP
029D 58                         POP     AX                      ; RESTORE AX
029E 5B                         POP     BX                      ; AND DATA ADDRESS
029F 51                         PUSH    CX
02A0 50                         PUSH    AX                      ; ADJUST ES:BX
02A1 8B CB                      MOV     CX,BX                   ; GET 3 HIGH ORDER NIBBLES OF BX
02A3 C1 E9 04                   SHR     CX,4
02A6 8C C0                      MOV     AX,ES
02A8 03 C1                      ADD     AX,CX
02AA 8E C0                      MOV     ES,AX
02AC 81 E3 000F                 AND     BX,000FH                ; ES:BX CHANGED TO ES:000X
02B0 58                         POP     AX
02B1 59                         POP     CX
02B2 2E: FF A4 01FB R           JMP     WORD PTR CS:[SI + OFFSET MI]
02B7                    BAD_COMMAND_POP:
02B7 58                         POP     AX
02B8 5B                         POP     BX
02B9                    BAD_COMMAND:
02B9 C6 06 0074 R 01            MOV     @DISK_STATUS1,BAD_CMD   ; COMMAND ERROR
02BE B0 00                      MOV     AL,0
02C0 C3                         RET
02C1                    DISK_IO_CONT    ENDP


                        ;--------------------------------------
                        ;       RESET THE DISK SYSTEM  (AH=00H) :
                        ;--------------------------------------

02C1                    DISK_RESET      PROC    NEAR
02C1 FA                         CLI
02C2 E4 A1                      IN      AL,INTB01               ; GET THE MASK REGISTER
02C4 EB 00                      JMP     $+2
02C6 24 BF                      AND     AL,0BFH                 ; ENABLE FIXED DISK INTERRUPT
02C8 E6 A1                      OUT     INTB01,AL
02CA FB                         STI                             ; START INTERRUPTS
02CB B0 04                      MOV     AL,04H
02CD BA 03F6                    MOV     DX,HF_REG_PORT
02D0 EE                         OUT     DX,AL                   ; RESET
02D1 B9 000A                    MOV     CX,10                   ; DELAY COUNT
02D4 49                 DRD:    DEC     CX
02D5 75 FD                      JNZ     DRD                     ; WAIT 4.8 MICRO-SEC
02D7 A0 0076 R                  MOV     AL,@CONTROL_BYTE
02DA 24 0F                      AND     AL,0FH                  ; SET HEAD OPTION
02DC EE                         OUT     DX,AL                   ; TURN RESET OFF
02DD E8 05F3 R                  CALL    NOT_BUSY
02E0 75 2D                      JNZ     DRERR                   ; TIME OUT ON RESET
02E2 BA 01F1                    MOV     DX,HF_PORT+1
02E5 EC                         IN      AL,DX                   ; GET RESET STATUS
02E6 3C 01                      CMP     AL,1
02E8 75 25                      JNZ     DRERR                   ; BAD RESET STATUS
02EA 80 66 FD EF                AND     @CMD_BLOCK+5,0EFH       ; SET TO DRIVE 0
02EE 2A D2                      SUB     DL,DL
02F0 E8 03F1 R                  CALL    INIT_DRV                ; SET MAX HEADS
02F3 E8 0466 R                  CALL    HDISK_RECAL             ; RECAL TO RESET SEEK SPEED
02F6 80 3E 0075 R 01            CMP     @HF_NUM,1               ; CHECK FOR DRIVE 1
02FB 76 0C                      JBE     DRE
02FD 80 4E FD 10                OR      @CMD_BLOCK+5,010H       ; SET TO DRIVE 1
0301 B2 01                      MOV     DL,1
0303 E8 03F1 R                  CALL    INIT_DRV                ; SET MAX HEADS
0306 E8 0466 R                  CALL    HDISK_RECAL             ; RECAL TO RESET SEEK SPEED
0309 C6 06 0074 R 00    DRE:    MOV     @DISK_STATUS1,0         ; IGNORE ANY SET UP ERRORS
030E C3                         RET
030F C6 06 0074 R 05    DRERR:  MOV     @DISK_STATUS1,BAD_RESET ; CARD FAILED
0314 C3                         RET
0315                    DISK_RESET      ENDP


                        ;--------------------------------------
                        ;       DISK STATUS ROUTINE  (AH = 01H) :
                        ;--------------------------------------

0315                    RETURN_STATUS   PROC    NEAR
0315 A0 0074 R                  MOV     AL,@DISK_STATUS1        ; OBTAIN PREVIOUS STATUS
0318 C6 06 0074 R 00            MOV     @DISK_STATUS1,0         ; RESET STATUS
031D C3                         RET
031E                    RETURN_STATUS   ENDP
```

```
                        PAGE
                        ;--------------------------------------
                        ;       DISK READ ROUTINE       (AH = 02H) :
                        ;--------------------------------------

031E                    DISK_READ       PROC    NEAR
031E C6 46 FE 20                MOV     @CMD_BLOCK+6,READ_CMD
0322 E9 04C6 R                  JMP     COMMAND1
0325                    DISK_READ       ENDP

                        ;--------------------------------------
                        ;       DISK WRITE ROUTINE      (AH = 03H) :
                        ;--------------------------------------

0325                    DISK_WRITE      PROC    NEAR
0325 C6 46 FE 30                MOV     @CMD_BLOCK+6,WRITE_CMD
0329 E9 0505 R                  JMP     COMMANDO
032C                    DISK_WRITE      ENDP

                        ;--------------------------------------
                        ;       DISK VERIFY             (AH = 04H) :
                        ;--------------------------------------

032C                    DISK_VERF       PROC    NEAR
032C C6 46 FE 40                MOV     @CMD_BLOCK+6,VERIFY_CMD
0330 E8 055C R                  CALL    COMMAND
0333 75 08                      JNZ     VERF_EXIT               ; CONTROLLER STILL BUSY
0335 E8 05C2 R                  CALL    WAIT
0338 75 03                      JNZ     VERF_EXIT               ; TIME OUT
033A E8 0630 R                  CALL    CHECK_STATUS
033D                    VERF_EXIT:
033D C3                         RET
033E                    DISK_VERF       ENDP

                        ;--------------------------------------
                        ;       FORMATTING              (AH = 05H) :
                        ;--------------------------------------

033E                    FMT_TRK PROC    NEAR                    ; FORMAT TRACK  (AH = 005H)
033E C6 46 FE 50                MOV     @CMD_BLOCK+6,FMTTRK_CMD
0342 06                         PUSH    ES
0343 53                         PUSH    BX
0344 E8 06C4 R                  CALL    GET_VEC                 ; GET DISK PARAMETERS ADDRESS
0347 26: 8A 47 0E               MOV     AL,ES:[BX][14]          ; GET SECTORS/TRACK
034B 88 46 F9                   MOV     @CMD_BLOCK+1,AL         ; SET SECTOR COUNT IN COMMAND
034E 5B                         POP     BX
034F 07                         POP     ES
0350 E9 050A R                  JMP     CMD_OF                  ; GO EXECUTE THE COMMAND
0353                    FMT_TRK ENDP


                        ;--------------------------------------
                        ;       READ DASD TYPE          (AH = 15H) :
                        ;--------------------------------------

0353                    READ_DASD_TYPE  LABEL   NEAR
0353                    READ_D_T        PROC    FAR             ; GET DRIVE PARAMETERS
0353 1E                         PUSH    DS                      ; SAVE REGISTERS
0354 06                         PUSH    ES
0355 53                         PUSH    BX
                                ASSUME  DS:DATA
0356 E8 0000 E                  CALL    DDS                     ; ESTABLISH ADDRESSING
0359 C6 06 0074 R 00            MOV     @DISK_STATUS1,0
035E 8A 1E 0075 R               MOV     BL,@HF_NUM              ; GET NUMBER OF DRIVES
0362 80 E2 7F                   AND     DL,7FH                  ; GET DRIVE NUMBER
0365 3A DA                      CMP     BL,DL
0367 76 22                      JBE     RDT_NOT_PRESENT         ; RETURN DRIVE NOT PRESENT
0369 E8 06C4 R                  CALL    GET_VEC                 ; GET DISK PARAMETER ADDRESS
036C 26: 8A 47 02               MOV     AL,ES:[BX][2]           ; HEADS
0370 26: 8A 4F 0E               MOV     CL,ES:[BX][14]
0374 F6 E9                      IMUL    CL                      ; * NUMBER OF SECTORS
0376 26: 8B 0F                  MOV     CX,ES:[BX]              ; MAX NUMBER OF CYLINDERS
0379 49                         DEC     CX                      ; LEAVE ONE FOR DIAGNOSTICS
037A F7 E9                      IMUL    CX                      ; NUMBER OF SECTORS
037C 8B CA                      MOV     CX,DX                   ; HIGH ORDER HALF
037E 8B D0                      MOV     DX,AX                   ; LOW ORDER HALF
0380 2B C0                      SUB     AX,AX
0382 B4 03                      MOV     AH,03H                  ; INDICATE FIXED DISK
0384 5B                 RDT2:   POP     BX                      ; RESTORE REGISTERS
0385 07                         POP     ES
0386 1F                         POP     DS
0387 F8                         CLC                             ; CLEAR CARRY
0388 CA 0002                    RET     2
038B                    RDT_NOT_PRESENT:
038B 2B C0                      SUB     AX,AX                   ; DRIVE NOT PRESENT RETURN
038D 8B C8                      MOV     CX,AX                   ; ZERO BLOCK COUNT
038F 8B D0                      MOV     DX,AX
0391 EB F1                      JMP     RDT2
0393                    READ_D_T        ENDP
```

```
                        PAGE
                        ;--------------------------------------
                        ;       GET PARAMETERS          (AH = 08H) :
                        ;--------------------------------------

0393                    GET_PARM_N      LABEL   NEAR
0393                    GET_PARM        PROC    FAR             ; GET DRIVE PARAMETERS
0393 1E                         PUSH    DS                      ; SAVE REGISTERS
0394 06                         PUSH    ES
0395 53                         PUSH    BX
                                ASSUME  DS:ABS0
0396 B8 ---- R                  MOV     AX,ABS0                 ; ESTABLISH ADDRESSING
0399 8E D8                      MOV     DS,AX
039B F6 C2 01                   TEST    DL,1                    ; CHECK FOR DRIVE 1
039E 74 06                      JZ      G0
03A0 C4 1E 0118 R               LES     BX,@HF1_TBL_VEC
03A4 EB 04                      JMP     SHORT G1
03A6 C4 1E 0104 R       G0:     LES     BX,@HF_TBL_VEC
                                ASSUME  DS:DATA
03AA                    G1:
03AA E8 0000 E                  CALL    DDS                     ; ESTABLISH SEGMENT
03AD 80 EA 80                   SUB     DL,80H
03B0 90 FA 02                   CMP     DL,MAX_FILE             ; TEST WITHIN RANGE
03B3 73 2C                      JAE     G4
03B5 C6 06 0074 R 00            MOV     @DISK_STATUS1,0
03BA 26: 8B 07                  MOV     AX,ES:[BX]              ; MAX NUMBER OF CYLINDERS
03BD 2D 0002                    SUB     AX,2                    ; ADJUST FOR 0-N
03C0 8A E8                      MOV     CH,AL
03C2 25 0300                    AND     AX,0300H                ; HIGH TWO BITS OF CYLINDER
03C5 D1 E8                      SHR     AX,1
03C7 D1 E8                      SHR     AX,1
03C9 26: 0A 47 0E               OR      AL,ES:[BX][14]          ; SECTORS
03CD 8A C8                      MOV     CL,AL
03CF 26: 8A 77 02               MOV     DH,ES:[BX][2]           ; HEADS
03D3 FE CE                      DEC     DH                      ; 0-N RANGE
03D5 8A 16 0075 R               MOV     DL,@HF_NUM              ; DRIVE COUNT
03D9 2B C0                      SUB     AX,AX
03DB                    G5:
03DB 5B                         POP     BX                      ; RESTORE REGISTERS
03DC 07                         POP     ES
03DD 1F                         POP     DS
03DE CA 0002                    RET     2
03E1                    G4:
03E1 C6 06 0074 R 07            MOV     @DISK_STATUS1,INIT_FAIL ; OPERATION FAILED
03E6 B4 07                      MOV     AH,INIT_FAIL
03E8 2A C0                      SUB     AL,AL
03EA 2B D2                      SUB     DX,DX
03EC 2B C9                      SUB     CX,CX
03EE F9                         STC                             ; SET ERROR FLAG
03EF EB EA                      JMP     G5
03F1                    GET_PARM        ENDP

                        ;--------------------------------------
                        ;       INITIALIZE DRIVE        (AH = 09H) :
                        ;--------------------------------------

03F1                    INIT_DRV        PROC    NEAR
03F1 C6 46 FE 91                MOV     @CMD_BLOCK+6,SET_PARM_CMD
03F5 E8 06C4 R                  CALL    GET_VEC                 ; ES:BX -> PARAMETER BLOCK
03F8 26: 8A 47 02               MOV     AL,ES:[BX][2]           ; GET NUMBER OF HEADS
03FC FE C8                      DEC     AL                      ; CONVERT TO 0-INDEX
03FE 8A 66 FD                   MOV     AH,@CMD_BLOCK+5         ; GET SDH REGISTER
0401 80 E4 F0                   AND     AH,0F0H                 ; CHANGE HEAD NUMBER
0404 0A E0                      OR      AH,AL                   ;  TO MAX HEAD
0406 88 66 FD                   MOV     @CMD_BLOCK+5,AH
0409 26: 8A 47 0E               MOV     AL,ES:[BX][14]          ; MAX SECTOR NUMBER
040D 88 46 F9                   MOV     @CMD_BLOCK+1,AL
0410 2B C0                      SUB     AX,AX
0412 88 46 FB                   MOV     @CMD_BLOCK+3,AL         ; ZERO FLAGS
0415 E8 055C R                  CALL    COMMAND                 ; TELL CONTROLLER
0418 75 08                      JNZ     INIT_EXIT               ; CONTROLLER BUSY ERROR
041A E8 05F3 R                  CALL    NOT_BUSY                ; WAIT FOR IT TO BE DONE
041D 75 03                      JNZ     INIT_EXIT               ; TIME OUT
041F E8 0630 R                  CALL    CHECK_STATUS
0422                    INIT_EXIT:
0422 C3                         RET
0423                    INIT_DRV        ENDP

                        ;--------------------------------------
                        ;       READ LONG               (AH = 0AH) :
                        ;--------------------------------------

0423                    RD_LONG         PROC    NEAR
0423 C6 46 FE 22                MOV     @CMD_BLOCK+6,READ_CMD OR ECC_MODE
0427 E9 04C6 R                  JMP     COMMAND1
042A                    RD_LONG         ENDP

                        ;--------------------------------------
                        ;       WRITE LONG              (AH = 0BH) :
                        ;--------------------------------------

042A                    WR_LONG         PROC    NEAR
042A C6 46 FE 32                MOV     @CMD_BLOCK+6,WRITE_CMD OR ECC_MODE
042E E9 0505 R                  JMP     COMMAND0
0431                    WR_LONG         ENDP

                        ;--------------------------------------
                        ;       SEEK                    (AH = 0CH) :
                        ;--------------------------------------

0431                    DISK_SEEK       PROC    NEAR
0431 C6 46 FE 70                MOV     @CMD_BLOCK+6,SEEK_CMD
0435 E8 055C R                  CALL    COMMAND
0438 75 14                      JNZ     DS_EXIT                 ; CONTROLLER BUSY ERROR
043A E8 05C2 R                  CALL    WAIT
043D 75 0F                      JNZ     DS_EXIT                 ; TIME OUT ON SEEK
043F E8 9630 R                  CALL    CHECK_STATUS
0442 80 3E 0074 R 40            CMP     @DISK_STATUS1,BAD_SEEK
0447 75 05                      JNE     DS_EXIT
0449 C6 06 0074 R 00            MOV     @DISK_STATUS1,0
044E                    DS_EXIT:
044E C3                         RET
044F                    DISK_SEEK       ENDP
```

```
                        PAGE
                        ;--------------------------------------
                        ;       TEST DISK READY      (AH = 10H) :
                        ;--------------------------------------

044F                    TST_RDY PROC    NEAR
044F E8 05F3 R                  CALL    NOT_BUSY                ; WAIT FOR CONTROLLER
0452 75 11                      JNZ     TR_EX
0454 8A 46 FD                   MOV     AL,@CMD_BLOCK+5         ; SELECT DRIVE
0457 BA 01F6                    MOV     DX,HF_PORT+6
045A EE                         OUT     DX,AL
045B E8 0642 R                  CALL    CHECK_ST                ; CHECK STATUS ONLY
045E 75 05                      JNZ     TR_EX
0460 C6 06 0074 R 00            MOV     @DISK_STATUS1,0         ; WIPE OUT DATA CORRECTED ERROR
0465 C3                 TR_EX:  RET
0466                    TST_RDY ENDP

                        ;--------------------------------------
                        ;       RECALIBRATE          (AH = 11H) :
                        ;--------------------------------------

0466                    HDISK_RECAL     PROC    NEAR
0466 C6 46 FE 10                MOV     @CMD_BLOCK+6,RECAL_CMD
046A E8 055C R                  CALL    COMMAND                 ; START THE OPERATION
046D 75 19                      JNZ     RECAL_EXIT              ; ERROR
046F E8 05C2 R                  CALL    WAIT                    ; WAIT FOR COMPLETION
0472 74 05                      JZ      RECAL_X                 ; TIME OUT ONE OK ?
0474 E8 05C2 R                  CALL    WAIT                    ; WAIT FOR COMPLETION LONGER
0477 75 0F                      JNZ     RECAL_EXIT              ; TIME OUT TWO TIMES IS ERROR
0479                    RECAL_X:
0479 E8 0630 R                  CALL    CHECK_STATUS
047C 80 3E 0074 R 40            CMP     @DISK_STATUS1,BAD_SEEK  ; SEEK NOT COMPLETE
0481 75 05                      JNE     RECAL_EXIT              ; IS OK
0483 C6 06 0074 R 00            MOV     @DISK_STATUS1,0
0488                    RECAL_EXIT:
0488 80 3E 0074 R 00            CMP     @DISK_STATUS1,0
048D C3                         RET
048E                    HDISK_RECAL     ENDP

                        ;--------------------------------------
                        ;       CONTROLLER DIAGNOSTIC (AH = 14H) :
                        ;--------------------------------------

048E                    CTLR_DIAGNOSTIC PROC    NEAR
048E FA                         CLI                             ; DISABLE INTERRUPTS WHILE CHANGING MASK
048F E4 A1                      IN      AL,INTB01               ; TURN ON SECOND INTERRUPT CHIP
0491 24 BF                      AND     AL,0BFH
0493 EB 00                      JMP     $+2
0495 E6 A1                      OUT     INTB01,AL
0497 E4 21                      IN      AL,INTA01               ; LET INTERRUPTS PASS THRU TO
0499 24 FB                      AND     AL,0FBH                 ;  SECOND CHIP
049B EB 00                      JMP     $+2
049D E6 21                      OUT     INTA01,AL
049F FB                         STI
04A0 E8 05F3 R                  CALL    NOT_BUSY                ; WAIT FOR CARD
04A3 75 1A                      JNZ     CD_ERR                  ; BAD CARD
04A5 BA 01F7                    MOV     DX,HF_PORT+7
04A8 B0 90                      MOV     AL,DIAG_CMD             ; START DIAGNOSE
04AA EE                         OUT     DX,AL
04AB E8 05F3 R                  CALL    NOT_BUSY                ; WAIT FOR IT TO COMPLETE
04AE B4 80                      MOV     AH,TIME_OUT
04B0 75 0F                      JNZ     CD_EXIT                 ; TIME OUT ON DIAGNOSTIC
04B2 BA 01F1                    MOV     DX,HF_PORT+1            ; GET ERROR REGISTER
04B5 EC                         IN      AL,DX
04B6 A2 008D R                  MOV     @HF_ERROR,AL            ; SAVE IT
04B9 B4 00                      MOV     AH,0
04BB 3C 01                      CMP     AL,1                    ; CHECK FOR ALL OK
04BD 74 02                      JE      SHORT CD_EXIT
04BF B4 20              CD_ERR: MOV     AH,BAD_CNTLR
04C1                    CD_EXIT:
04C1 88 26 0074 R               MOV     @DISK_STATUS1,AH
04C5 C3                         RET
04C6                    CTLR_DIAGNOSTIC ENDP

                        ;--------------------------------------
                        ; COMMAND1                             :
                        ;       REPEATEDLY INPUTS DATA TILL    :
                        ;       NSECTOR RETURNS ZERO           :
                        ;--------------------------------------
04C6                    COMMAND1:
04C6 E8 06A1 R                  CALL    CHECK_DMA               ; CHECK 64K BOUNDARY ERROR
04C9 72 39                      JC      CMD_ABORT
04CB 8B FB                      MOV     DI,BX
04CD E8 055C R                  CALL    COMMAND                 ; OUTPUT COMMAND
04D0 75 32                      JNZ     CMD_ABORT
04D2                    CMD_11:
04D2 E8 05C2 R                  CALL    WAIT                    ; WAIT FOR DATA REQUEST INTERRUPT
04D5 75 2D                      JNZ     TM_OUT                  ; TIME OUT
04D7 B9 0100                    MOV     CX,256D                 ; SECTOR SIZE IN WORDS
04DA BA 01F0                    MOV     DX,HF_PORT
04DD FA                         CLI
04DE FC                         CLD
04DF F3/ 6D                     REP     INSW                    ; GET THE SECTOR
04E1 FB                         STI
04E2 F6 46 FE 02                TEST    @CMD_BLOCK+6,ECC_MODE   ; CHECK FOR NORMAL INPUT
04E6 74 12                      JZ      CMD_I3
04E8 E8 061A R                  CALL    WAIT_DRQ                ; WAIT FOR DATA REQUEST
04EB 72 17                      JC      TM_OUT
04ED BA 01F0                    MOV     DX,HF_PORT
04F0 B9 0004                    MOV     CX,4                    ; GET ECC BYTES
04F3 EC                 CMD_12: IN      AL,DX
04F4 26: 88 05                  MOV     ES:BYTE PTR [DI],AL     ; GO SLOW FOR BOARD
04F7 47                         INC     DI
04F8 E2 F9                      LOOP    CMD_12
04FA E8 0630 R          CMD_13: CALL    CHECK_STATUS
04FD 75 05                      JNZ     CMD_ABORT               ; ERROR RETURNED
04FF FE 4E F9                   DEC     @CMD_BLOCK+1            ; CHECK FOR MORE
0502 75 CE                      JNZ     SHORT CMD_11
0504                    CMD_ABORT:
0504                    TM_OUT:
0504 C3                         RET
```

```
                        PAGE
                        ;--------------------------------------
                        ; COMMANDO                            :
                        ;       REPEATEDLY OUTPUTS DATA TILL  :
                        ;       NSECTOR RETURNS ZERO          :
                        ;--------------------------------------
1005                    COMMANDO:
1006 0505 E8 06A1 R             CALL    CHECK_DMA               ; CHECK 64K BOUNDARY ERROR
1007 0508 72 FA                 JC      CMD_ABORT
1008 050A 8B F3         CMD_OF: MOV     SI,BX
1009 050C E8 055C R             CALL    COMMAND                 ; OUTPUT COMMAND
1010 050F 75 F3                 JNZ     CMD_ABORT
1011 0511 E8 061A R             CALL    WAIT_DRQ                ; WAIT FOR DATA REQUEST
1012 0514 72 EE                 JC      TM_OUT                  ; TOO LONG
1013 0516 1E            CMD_01: PUSH    DS
1014 0517 06                    PUSH    ES                      ; MOVE ES TO DS
1015 0518 1F                    POP     DS
1016 0519 B9 0100               MOV     CX,256D                 ; PUT THE DATA OUT TO THE CARD
1017 051C BA 01F0               MOV     DX,HF_PORT
1018 051F FA                    CLI
1019 0520 FC                    CLD
1020 0521 F3/ 6F                REP     OUTSW
1021 0523 FB                    STI
1022 0524 1F                    POP     DS                      ; RESTORE DS
1023 0525 F6 46 FE 02           TEST    @CMD_BLOCK+6,ECC_MODE   ; CHECK FOR NORMAL OUTPUT
1024 0529 74 12                 JZ      CMD_03
1025 052B E8 061A R             CALL    WAIT_DRQ                ; WAIT FOR DATA REQUEST
1026 052E 72 D4                 JC      TM_OUT
1027 0530 BA 01F0               MOV     DX,HF_PORT
1028 0533 B9 0004               MOV     CX,4                    ; OUTPUT THE ECC BYTES
1029 0536 26: 8A 04     CMD_02: MOV     AL,ES:BYTE PTR [SI]
1030 0539 EE                    OUT     DX,AL
1031 053A 46                    INC     SI
1032 053B E2 F9                 LOOP    CMD_02
1033 053D               CMD_03:
1034 053D E8 05C2 R             CALL    WAIT                    ; WAIT FOR SECTOR COMPLETE INTERRUPT
1035 0540 75 C2                 JNZ     TM_OUT                  ; ERROR RETURNED
1036 0542 E8 0630 R             CALL    CHECK_STATUS
1037 0545 75 BD                 JNZ     CMD_ABORT
1038 0547 F6 06 008C R 08       TEST    @HF_STATUS,ST_DRQ       ; CHECK FOR MORE
1039 054C 75 C8                 JNZ     SHORT CMD_01
1040 054E BA 01F2               MOV     DX,HF_PORT+2            ; CHECK RESIDUAL SECTOR COUNT
1041 0551 EC                    IN      AL,DX                   ;
1042 0552 A8 FF                 TEST    AL,0FFH                 ;
1043 0554 74 05                 JZ      CMD_04                  ; COUNT = 0  OK
1044 0556 C6 06 0074 R BB       MOV     @DISK_STATUS1,UNDEF_ERR ; OPERATION ABORTED - PARTIAL TRANSFER
1045 055B               CMD_04:
1046 055B C3                    RET
1047                    ;--------------------------------------------------------
1048                    ; COMMAND                                                :
1049                    ;       THIS ROUTINE OUTPUTS THE COMMAND BLOCK           :
1050                    ; OUTPUT                                                 :
1051                    ;       BL = STATUS                                      :
1052                    ;       BH = ERROR REGISTER                              :
1053                    ;--------------------------------------------------------
1054
1055 055C                COMMAND PROC    NEAR
1056 055C 53                    PUSH    BX                      ; WAIT FOR SEEK COMPLETE AND READY
1057 055D B9 0600               MOV     CX,DELAY_2              ; SET INITIAL DELAY BEFORE TEST
1058 0560               COMMAND1:
1059 0560 51                    PUSH    CX                      ; SAVE LOOP COUNT
1060 0561 E8 044F R             CALL    TST_RDY                 ; CHECK DRIVE READY
1061 0564 59                    POP     CX
1062 0565 74 0B                 JZ      COMMAND2                ; DRIVE IS READY
1063 0567 80 3E 0074 R 80       CMP     @DISK_STATUS1,TIME_OUT  ; TST_RDY TIMED OUT--GIVE UP
1064 056C 74 48                 JZ      CMD_TTMEOUT
1065 056E E2 F0                 LOOP    COMMAND1                ; KEEP TRYING FOR A WHILE
1066 0570 EB 49                 JMP     SHORT COMMAND4          ; ITS NOT GOING TO GET READY
1067 0572               COMMAND2:
1068 0572 5B                    POP     BX
1069 0573 57                    PUSH    DI
1070 0574 C6 06 008E R 00       MOV     @HF_INT_FLAG,0          ; RESET INTERRUPT FLAG
1071 0579 FA                    CLI                             ; INHIBIT INTERRUPTS WHILE CHANGING MASK
1072 057A E4 A1                 IN      AL,INTB01               ; TURN ON SECOND INTERRUPT CHIP
1073 057C 24 BF                 AND     AL,0BFH
1074 057E EB 00                 JMP     $+2
1075 0580 E6 A1                 OUT     INTB01,AL
1076 0582 E4 21                 IN      AL,INTA01               ; LET INTERRUPTS PASS THRU TO
1077 0584 24 FB                 AND     AL,0FBH                 ;  SECOND CHIP
1078 0586 EB 00                 JMP     $+2
1079 0588 E6 21                 OUT     INTA01,AL
1080 058A FB                    STI
1081 058B 33 FF                 XOR     DI,DI                   ; INDEX THE COMMAND TABLE
1082 058D BA 01F1               MOV     DX,HF_PORT+1            ; DISK ADDRESS
1083 0590 F6 06 0076 R C0       TEST    @CONTROL_BYTE,0C0H      ; CHECK FOR RETRY SUPPRESSION
1084 0595 74 11                 JZ      COMMAND3
1085 0597 8A 46 FE              MOV     AL,@CMD_BLOCK+6         ; YES-GET OPERATION CODE
1086 059A 24 F0                 AND     AL,0F0H                 ; GET RID OF MODIFIERS
1087 059C 3C 20                 CMP     AL,20H                  ; 20H-40H IS READ, WRITE, VERIFY
1088 059E 72 08                 JB      COMMAND3
1089 05A0 3C 40                 CMP     AL,40H
1090 05A2 77 04                 JA      COMMAND3
1091 05A4 80 4E FE 01           OR      @CMD_BLOCK+6,NO_RETRIES ; VALID OPERATION FOR RETRY SUPPRESS
1092 05A8               COMMAND3:
1093 05A8 8A 43 F8              MOV     AL,[@CMD_BLOCK+DI]      ; GET THE COMMAND STRING BYTE
1094 05AB EE                    OUT     DX,AL                   ; GIVE IT TO CONTROLLER
1095 05AC 47                    INC     DI                      ; NEXT BYTE IN COMMAND BLOCK
1096 05AD 42                    INC     DX                      ; NEXT DISK ADAPTER REGISTER
1097 05AE 81 FA 01F8            CMP     DX,HF_PORT+8            ; ALL DONE?
1098 05B2 75 F4                 JNZ     COMMAND3                ; NO--GO DO NEXT ONE
1099 05B4 5F                    POP     DI
1100 05B5 C3                    RET                             ; ZERO FLAG IS SET
1101 05B6               CMD_TIMEOUT:
1102 05B6 C6 06 0074 R 20       MOV     @DISK_STATUS1,BAD_CNTLR
1103 05BB               COMMAND4:
1104 05BB 5B                    POP     BX
1105 05BC 80 3E 0074 R 00       CMP     @DISK_STATUS1,0         ; SET CONDITION CODE FOR CALLER
1106 05C1 C3                    RET
1107 05C2               COMMAND ENDP
```

```
                        PAGE
                        ;--------------------------------------
                        ;       WAIT FOR INTERRUPT             :
                        ;--------------------------------------
05C2                    WAIT    PROC    NEAR
05C2 FB                         STI                             ; MAKE SURE INTERRUPTS ARE ON
05C3 2B C9                      SUB     CX,CX                   ; SET INITIAL DELAY BEFORE TEST
05C5 F8                         CLC
05C6 B8 9000                    MOV     AX,9000H                ; DEVICE WAIT INTERRUPT
05C9 CD 15                      INT     15H
05CB 72 0F                      JC      WT2                     ; DEVICE TIMED OUT
                        
05CD B3 25                      MOV     BL,DELAY_1              ; SET DELAY COUNT
                        
                        ;-----  WAIT LOOP
                        
05CF F6 06 008E R 80    WT1:    TEST    @HF_INT_FLAG,80H        ; TEST FOR INTERRUPT
05D4 E1 F9                      LOOPZ   WT1
05D6 75 0B                      JNZ     WT3                     ; INTERRUPT--LETS GO
05D8 FE CB                      DEC     BL
05DA 75 F3                      JNZ     WT1                     ; KEEP TRYING FOR A WHILE
                        
05DC C6 06 0074 R 80    WT2:    MOV     @DISK_STATUS1,TIME_OUT  ; REPORT TIME OUT ERROR
05E1 EB 0A                      JMP     SHORT WT4
05E3 C6 06 0074 R 00    WT3:    MOV     @DISK_STATUS1,0
05E8 C6 06 008E R 00            MOV     @HF_INT_FLAG,0
05ED 80 3E 0074 R 00    WT4:    CMP     @DISK_STATUS1,0         ; SET CONDITION CODE FOR CALLER
05F2 C3                         RET
05F3                    WAIT    ENDP
                        
                        ;--------------------------------------
                        ;       WAIT FOR CONTROLLER NOT BUSY   :
                        ;--------------------------------------
05F3                    NOT_BUSY        PROC    NEAR
05F3 FB                         STI                             ; MAKE SURE INTERRUPTS ARE ON
05F4 53                         PUSH    BX
05F5 2B C9                      SUB     CX,CX                   ; SET INITIAL DELAY BEFORE TEST
05F7 BA 01F7                    MOV     DX,HF_PORT+7
05FA B3 25                      MOV     BL,DELAY_1
05FC EC                 NB1:    IN      AL,DX                   ; CHECK STATUS
05FD A8 80                      TEST    AL,ST_BUSY
05FF E0 FB                      LOOPNZ  NB1
0601 74 0B                      JZ      NB2                     ; NOT BUSY--LETS GO
0603 FE CB                      DEC     BL
0605 75 F5                      JNZ     NB1                     ; KEEP TRYING FOR A WHILE
0607 C6 06 0074 R 80            MOV     @DISK_STATUS1,TIME_OUT  ; REPORT TIME OUT ERROR
060C EB 05                      JMP     SHORT NB3
060E C6 06 0074 R 00    NB2:    MOV     @DISK_STATUS1,0
0613 5B                 NB3:    POP     BX
0614 80 3E 0074 R 00            CMP     @DISK_STATUS1,0         ; SET CONDITION CODE FOR CALLER
0619 C3                         RET
061A                    NOT_BUSY        ENDP
                        
                        ;--------------------------------------
                        ;       WAIT FOR DATA REQUEST          :
                        ;--------------------------------------
061A                    WAIT_DRQ        PROC    NEAR
061A B9 0100                    MOV     CX,DELAY_3
061D BA 01F7                    MOV     DX,HF_PORT+7
0620 EC                 WQ_1:   IN      AL,DX                   ; GET STATUS
0621 A8 08                      TEST    AL,ST_DRQ               ; WAIT FOR DRQ
0623 75 09                      JNZ     WQ_OK
0625 E2 F9                      LOOP    WQ_1                    ; KEEP TRYING FOR A SHORT WHILE
0627 C6 06 0074 R 80            MOV     @DISK_STATUS1,TIME_OUT  ; ERROR
062C F9                         STC
062D C3                         RET
062E F8                 WQ_OK:  CLC
062F C3                         RET
0630                    WAIT_DRQ        ENDP
                        ;--------------------------------------
                        ;       CHECK FIXED DISK STATUS        :
                        ;--------------------------------------
0630                    CHECK_STATUS    PROC    NEAR
0630 E8 0642 R                  CALL    CHECK_ST                ; CHECK THE STATUS BYTE
0633 75 07                      JNZ     CHECK_S1                ; AN ERROR WAS FOUND
0635 A8 01                      TEST    AL,ST_ERROR             ; WERE THERE ANY OTHER ERRORS
0637 74 03                      JZ      CHECK_S1                ; NO ERROR REPORTED
0639 E8 0676 R                  CALL    CHECK_ER                ; ERROR REPORTED
063C                    CHECK_S1:
063C 80 3E 0074 R 00            CMP     @DISK_STATUS1,0         ; SET STATUS FOR CALLER
0641 C3                         RET
0642                    CHECK_STATUS    ENDP
                        ;--------------------------------------
                        ;       CHECK FIXED DISK STATUS BYTE   :
                        ;--------------------------------------
0642                    CHECK_ST        PROC    NEAR
0642 BA 01F7                    MOV     DX,HF_PORT+7            ; GET THE STATUS
0645 EC                         IN      AL,DX
0646 A2 008C R                  MOV     @HF_STATUS,AL
0649 B4 00                      MOV     AH,0
064B A8 80                      TEST    AL,ST_BUSY              ; IF STILL BUSY
064D 75 1A                      JNZ     CKST_EXIT               ;  REPORT OK
064F B4 CC                      MOV     AH,WRITE_FAULT
0651 A8 20                      TEST    AL,ST_WRT_FLT           ; CHECK FOR WRITE FAULT
0653 75 14                      JNZ     CKST_EXIT
0655 B4 AA                      MOV     AH,NOT_RDY
0657 A8 40                      TEST    AL,ST_READY             ; CHECK FOR NOT READY
0659 74 0E                      JZ      CKST_EXIT
065B B4 40                      MOV     AH,BAD_SEEK
065D A8 10                      TEST    AL,ST_SEEK_COMPL        ; CHECK FOR SEEK NOT COMPLETE
065F 74 08                      JZ      CKST_EXIT
0661 B4 11                      MOV     AH,DATA_CORRECTED
0663 A8 04                      TEST    AL,ST_CORRCTD           ; CHECK FOR CORRECTED ECC
0665 75 02                      JNZ     CKST_EXIT
0667 B4 00                      MOV     AH,0
0669                    CKST_EXIT:
0669 88 26 0074 R               MOV     @DISK_STATUS1,AH        ; SET ERROR FLAG
066D 80 FC 11                   CMP     AH,DATA_CORRECTED       ; KEEP GOING WITH DATA CORRECTED
0670 74 03                      JZ      CKST_EXT
0672 80 FC 00                   CMP     AH,0
0675                    CKST_EXT:
0675 C3                         RET
0676                    CHECK_ST        ENDP
```

```
                        PAGE
                        ;--------------------------------------
                        ;       CHECK FIXED DISK ERROR REGISTER :
                        ;--------------------------------------
0676                    CHECK_ER        PROC    NEAR
0676 BA 01F1                    MOV     DX,HF_PORT+1            ; GET THE ERROR REGISTER
0679 EC                         IN      AL,DX
067A A2 008D R                  MOV     @HF_ERROR,AL
067D 53                         PUSH    BX
067E B9 0008                    MOV     CX,8                    ; TEST ALL 8 BITS
0681 D0 E0              CK1:    SHL     AL,1                    ; MOVE NEXT ERROR BIT TO CARRY
0683 72 02                      JC      CK2                     ; FOUND THE ERROR
0685 E2 FA                      LOOP    CK1                     ; KEEP TRYING
0687 BB 0698 R          CK2:    MOV     BX,OFFSET ERR_TBL       ; COMPUTE ADDRESS OF
068A 03 D9                      ADD     BX,CX                   ; ERROR CODE'
068C 2E: 8A 27                  MOV     AH,BYTE PTR CS:[BX]     ; GET ERROR CODE
068F 88 26 0074 R       CKEX:   MOV     @DISK_STATUS1,AH        ; SAVE ERROR CODE
0693 5B                         POP     BX
0694 80 FC 00                   CMP     AH,0
0697 C3                         RET
0698 E0                 ERR_TBL DB      NO_ERR
0699 02 40 01 BB                DB      BAD_ADDR_MARK,BAD_SEEK,BAD_CMD,UNDEF_ERR
069D 04 BB 10 0A                DB      RECORD_NOT_FND,UNDEF_ERR,BAD_ECC,BAD_SECTOR
06A1                    CHECK_ER        ENDP

                        ;--------------------------------------------------------
                        ; CHECK_DMA                                             :
                        ;  -CHECK ES:BX AND # SECTORS TO MAKE SURE THAT IT WILL :
                        ;   FIT WITHOUT SEGMENT OVERFLOW.                       :
                        ;  -ES:BX HAS BEEN REVISED TO THE FORMAT SSSS:000X      :
                        ;  -OK IF # SECTORS < 80H (7FH IF LONG READ OR WRITE)   :
                        ;  -OK IF # SECTORS = 80H (7FH) AND BX <= 00H (04H)     :
                        ;  -ERROR OTHERWISE                                     :
                        ;--------------------------------------------------------
06A1                    CHECK_DMA       PROC    NEAR
06A1 50                         PUSH    AX                      ; SAVE REGISTERS
06A2 B8 8000                    MOV     AX,8000H                ; AH = MAX # SECTORS   AL = MAX OFFSET
06A5 F6 46 FE 02                TEST    @CMD_BLOCK+6,ECC_MODE
06A9 74 03                      JZ      CKD1
06AB B8 7F04                    MOV     AX,7F04H                ; ECC IS 4 MORE BYTES
06AE 3A 66 F9           CKD1:   CMP     AH,@CMD_BLOCK+1         ; NUMBER OF SECTORS
06B1 77 06                      JA      CKDOK                   ; IT WILL FIT
06B3 72 07                      JB      CKDERR                  ; TOO MANY
06B5 3A C3                      CMP     AL,BL                   ; CHECK OFFSET ON MAX SECTORS
06B7 72 03                      JB      CKDERR                  ; ERROR
06B9 F8                 CKDOK:  CLC                             ; CLEAR CARRY
06BA 58                         POP     AX
06BB C3                         RET                             ; NORMAL RETURN
06BC F9                 CKDERR: STC                             ; INDICATE ERROR
06BD C6 06 0074 R 09            MOV     @DISK_STATUS1,DMA_BOUNDARY
06C2 58                         POP     AX
06C3 C3                         RET
06C4                    CHECK_DMA       ENDP

                        ;--------------------------------------
                        ;       SET UP  ES:BX-> DISK PARMS      :
                        ;--------------------------------------
06C4                    GET_VEC PROC    NEAR
06C4 2B C0                      SUB     AX,AX                   ; GET DISK PARAMETER ADDRESS
06C6 8E C0                      MOV     ES,AX
                                ASSUME  ES:ABS0
06C8 F6 C2 01                   TEST    DL,1
06CB 74 07                      JZ      GV_0
06CD 26: C4 1E 0118 R           LES     BX,@HF1_TBL_VEC         ; ES:BX -> DRIVE PARAMETERS
06D2 EB 05                      JMP     SHORT GV_EXIT
06D4                    GV_0:
06D4 26: C4 1E 0104 R           LES     BX,@HF_TBL_VEC          ; ES:BX -> DRIVE PARAMETERS
06D9                    GV_EXIT:
06D9 C3                         RET
06DA                    GET_VEC ENDP

                        ;--- HARDWARE INT  76H -- ( IRQ LEVEL  14 ) ---------------------------------
                        ;                                                                            :
                        ;       FIXED DISK INTERRUPT ROUTINE                                         :
                        ;                                                                            :
                        ;--------------------------------------------------------------------------

06DA                    HD_INT  PROC    NEAR
06DA 50                         PUSH    AX
06DB 1E                         PUSH    DS
06DC E8 0000 E                  CALL    DDS
06DF C6 06 008E R FF            MOV     @HF_INT_FLAG,0FFH       ; ALL DONE
06E4 B0 20                      MOV     AL,EOI                  ; NON-SPECIFIC END OF INTERRUPT
06E6 E6 A0                      OUT     INTB00,AL               ; FOR CONTROLLER #2
06E8 EB 00                      JMP     $+2                     ; WAIT
06EA E6 20                      OUT     INTA00,AL               ; FOR CONTROLLER #1
06EC 1F                         POP     DS
06ED FB                         STI                             ; RE-ENABLE INTERRUPTS
06EE B8 9100                    MOV     AX,9100H                ; DEVICE POST
06F1 CD 15                      INT     15H                     ;  INTERRUPT
06F3 58                         POP     AX
06F4 CF                         IRET                            ; RETURN FROM INTERRUPT
06F5                    HD_INT  ENDP
06F5 31 31 2F 31 35 2F  
        38 35           
06FD                            DB      '11/15/85'              ; RELEASE MARKER
06FD                    CODE    ENDS
                                END
```

```
                        PAGE 118,121
                        TITLE KYBD ----- 03/06/86  KEYBOARD BIOS
                        .LIST
0000                    CODE    SEGMENT BYTE PUBLIC

                                PUBLIC  K16
                                PUBLIC  KEYBOARD_IO_1
                                PUBLIC  KB_INT_1
                                PUBLIC  SND_DATA

                                EXTRN   BEEP:NEAR
                                EXTRN   DDS:NEAR
                                EXTRN   START_1:NEAR
                                EXTRN   K6:BYTE
                                EXTRN   K6L:ABS
                                EXTRN   K7:BYTE
                                EXTRN   K8:BYTE
                                EXTRN   K10:BYTE
                                EXTRN   K11:BYTE
                                EXTRN   K12:BYTE
                                EXTRN   K14:BYTE
                                EXTRN   K15:BYTE

                        ;--- INT 16 H ----------------------------------------------------------------
                        ; KEYBOARD I/O                                                              :
                        ;       THESE ROUTINES PROVIDE READ KEYBOARD SUPPORT                        :
                        ; INPUT                                                                     :
                        ;       (AH)= 00H  READ THE NEXT ASCII CHARACTER ENTERED FROM THE KEYBOARD, :
                        ;                  RETURN THE RESULT IN (AL), SCAN CODE IN (AH).            :
                        ;                  THIS IS THE COMPATIBLE READ INTERFACE, EQUIVALENT TO THE  :
                        ;                  STANDARD PC OR PCAT KEYBOARD                             :
                        ;-------------------------------------------------------------------------- :
                        ;       (AH)= 01H  SET THE Z FLAG TO INDICATE IF AN ASCII CHARACTER IS       :
                        ;                  AVAILABLE TO BE READ.                                    :
                        ;                  (ZF)= 1 -- NO CODE AVAILABLE                             :
                        ;                  (ZF)= 0 -- CODE IS AVAILABLE  (AX)= CHARACTER            :
                        ;                  IF (ZF)= 0, THE NEXT CHARACTER IN THE BUFFER TO BE READ IS:
                        ;                  IN (AX), AND THE ENTRY REMAINS IN THE BUFFER.            :
                        ;                  THIS WILL RETURN ONLY PC/PCAT KEYBOARD COMPATIBLE CODES   :
                        ;-------------------------------------------------------------------------- :
                        ;       (AH)= 02H  RETURN THE CURRENT SHIFT STATUS IN AL REGISTER            :
                        ;                  THE BIT SETTINGS FOR THIS CODE ARE INDICATED IN THE      :
                        ;                  THE EQUATES FOR @KB_FLAG                                 :
                        ;-------------------------------------------------------------------------- :
                        ;       (AH)= 03H  SET TYPAMATIC RATE AND DELAY                             :
                        ;             (AL) = 05H                                                    :
                        ;             (BL) = TYPAMATIC RATE  (BITS 5 - 7 MUST BE RESET TO 0)        :
                        ;                                                                           :
                        ;                  REGISTER    RATE      REGISTER    RATE                   :
                        ;                   VALUE     SELECTED    VALUE     SELECTED                :
                        ;                  ----------------------------------------                 :
                        ;                     00H       30.0      10H       7.5                     :
                        ;                     01H       26.7      11H       6.7                     :
                        ;                     02H       24.0      12H       6.0                     :
                        ;                     03H       21.8      13H       5.5                     :
                        ;                     04H       20.0      14H       5.0                     :
                        ;                     05H       18.5      15H       4.6                     :
                        ;                     06H       17.1      16H       4.3                     :
                        ;                     07H       16.0      17H       4.0                     :
                        ;                     08H       15.0      18H       3.7                     :
                        ;                     09H       13.3      19H       3.3                     :
                        ;                     0AH       12.0      1AH       3.0                     :
                        ;                     0BH       10.9      1BH       2.7                     :
                        ;                     0CH       10.0      1CH       2.5                     :
                        ;                     0DH        9.2      1DH       2.3                     :
                        ;                     0EH        8.6      1EH       2.1                     :
                        ;                     0FH        8.0      1FH       2.0                     :
                        ;                                                                           :
                        ;             (BH) = TYPAMATIC DELAY (BITS 2 - 7 MUST BE RESET TO 0)        :
                        ;                                                                           :
                        ;                  REGISTER    DELAY                                        :
                        ;                   VALUE      VALUE                                        :
                        ;                  ----------------                                         :
                        ;                     00H      250 ms                                       :
                        ;                     01H      500 ms                                       :
                        ;                     02H      750 ms                                       :
                        ;                     03H     1000 ms                                       :
                        ;-------------------------------------------------------------------------- :
                        ;       (AH)= 05H  PLACE ASCII CHARACTER/SCAN CODE COMBINATION IN KEYBOARD   :
                        ;                  BUFFER AS IF STRUCK FROM KEYBOARD                        :
                        ;                  ENTRY:  (CL) = ASCII CHARACTER                           :
                        ;                          (CH) = SCAN CODE                                 :
                        ;                  EXIT:   (AL) = 00H = SUCCESSFUL OPERATION                :
                        ;                          (AL) = 01H = UNSUCCESSFUL - BUFFER FULL          :
                        ;                  FLAGS:  CARRY IF ERROR                                   :
                        ;-------------------------------------------------------------------------- :
                        ;       (AH)= 10H  EXTENDED READ INTERFACE FOR THE ENHANCED KEYBOARD,       :
                        ;                  OTHERWISE SAME AS FUNCTION AH=0                          :
                        ;-------------------------------------------------------------------------- :
                        ;       (AH)= 11H  EXTENDED ASCII STATUS FOR THE ENHANCED KEYBOARD,         :
                        ;                  OTHERWISE SAME AS FUNCTION AH=1                          :
                        ;-------------------------------------------------------------------------- :
                        ;       (AH)= 12H  RETURN THE EXTENDED SHIFT STATUS IN AX REGISTER           :
                        ;                  AL = BITS FROM KB_FLAG, AH = BITS FOR LEFT AND RIGHT     :
                        ;                  CTL AND ALT KEYS FROM KB_FLAG_1 AND KB_FLAG_3            :
                        ;                                                                           :
                        ; OUTPUT                                                                    :
                        ;       AS NOTED ABOVE, ONLY (AX) AND FLAGS CHANGED                         :
                        ;       ALL REGISTERS RETAINED                                              :
                        ;-------------------------------------------------------------------------- :
                                ASSUME  CS:CODE,DS:DATA

0000                    KEYBOARD_IO_1   PROC    FAR             ; >>> ENTRY POINT FOR ORG 0E82EH
0000 FB                         STI                             ; INTERRUPTS BACK ON
0001 1E                         PUSH    DS                      ; SAVE CURRENT DS
0002 53                         PUSH    BX                      ; SAVE BX TEMPORARILY
0003 51                         PUSH    CX                      ; SAVE CX TEMPORARILY
0004 E8 0000 E                  CALL    DDS                     ; ESTABLISH POINTER TO DATA REGION
0007 0A E4                      OR      AH,AH                   ; CHECK FOR (AH)= 00H
0009 74 2D                      JZ      K1                      ; ASCII_READ
000B FE CC                      DEC     AH                      ; CHECK FOR (AH)= 01H
000D 74 3E                      JZ      K2                      ; ASCII_STATUS
000F FE CC                      DEC     AH                      ; CHECK FOR (AH)= 02H
0011 74 6B                      JZ      K3                      ; SHIFT_STATUS
```

```
0013 FE CC                      DEC     AH                      ; CHECK FOR (AH)= 03H
0015 74 6C                      JZ      K300                    ; SET TYPAMATIC RATE/DELAY
0017 80 EC 02                   SUB     AH,2                    ; CHECK FOR (AH)= 05H
001A 75 03                      JNZ     K101                    ;
001C E9 00A4 R                  JMP     K500                    ; KEYBOARD WRITE
001F 80 EC 0B           K101:   SUB     AH,11                   ; AH = 10
0022 74 0C                      JZ      K1E                     ; EXTENDED_ASCII_READ
0024 FE CC                      DEC     AH                      ; CHECK FOR (AH)= 11H
0026 74 1A                      JZ      K2E                     ; EXTENDED_ASCII_STATUS
0028 FE CC                      DEC     AH                      ; CHECK FOR (AH)= 12H
002A 74 39                      JZ      K3E                     ; EXTENDED_SHIFT_STATUS
002C                    KIO_EXIT:
002C 59                         POP     CX                      ; RECOVER REGISTER
002D 5B                         POP     BX                      ; RECOVER REGISTER
002E 1F                         POP     DS                      ; RECOVER SEGMENT
002F CF                         IRET                            ; INVALID COMMAND

                        ;------ ASCII CHARACTER

0030 E8 00C7 R          K1E:    CALL    K1S             ; GET A CHARACTER FROM THE BUFFER (EXTENDED)
0033 E8 0125 R                  CALL    K10_E_XLAT      ; ROUTINE TO XLATE FOR EXTENDED CALLS
0036 EB F4                      JMP     K10_EXIT        ; GIVE IT TO THE CALLER

0038 E8 00C7 R          K1:     CALL    K1S             ; GET A CHARACTER FROM THE BUFFER
003B E8 0130 R                  CALL    K10_S_XLAT      ; ROUTINE TO XLATE FOR STANDARD CALLS
003E 72 F8                      JC      K1              ; CARRY SET MEANS THROW CODE AWAY
0040 EB EA              K1A:    JMP     K10_EXIT        ; RETURN TO CALLER

                        ;------ ASCII STATUS

0042 E8 0103 R          K2E:    CALL    K2S             ; TEST FOR CHARACTER IN BUFFER (EXTENDED)
0045 74 18                      JZ      K2B             ; RETURN IF BUFFER EMPTY
0047 9C                         PUSHF                   ; SAVE ZF FROM TEST
0048 E8 0125 R                  CALL    K10_E_XLAT      ; ROUTINE TO XLATE FOR EXTENDED CALLS
004B EB 11                      JMP     SHORT K2A       ; GIVE IT TO THE CALLER

004D E8 0103 R          K2:     CALL    K2S             ; TEST FOR CHARACTER IN BUFFER
0050 74 0D                      JZ      K2B             ; RETURN IF BUFFER EMPTY
0052 9C                         PUSHF                   ; SAVE ZF FROM TEST
0053 E8 0130 R                  CALL    K10_S_XLAT      ; ROUTINE TO XLATE FOR STANDARD CALLS
0056 73 06                      JNC     K2A             ; CARRY CLEAR MEANS PASS VALID CODE
0058 9D                         POPF                    ;  INVALID CODE FOR THIS TYPE OF CALL
0059 E8 00C7 R                  CALL    K1S             ;  THROW THE CHARACTER AWAY
005C EB EF                      JMP     K2              ; GO LOOK FOR NEXT CHAR, IF ANY

005E 9D                 K2A:    POPF                    ; RESTORE ZF FROM TEST
005F 59                 K2B:    POP     CX              ; RECOVER REGISTER
0060 5B                         POP     BX              ; RECOVER REGISTER
0061 1F                         POP     DS              ; RECOVER SEGMENT
0062 CA 0002                    RET     2               ; THROW AWAY FLAGS

                        ;------ SHIFT STATUS

0065                    K3E:                            ; GET THE EXTENDED SHIFT STATUS FLAGS
0065 8A 26 0018 R               MOV     AH,@KB_FLAG_1   ; GET SYSTEM SHIFT KEY STATUS
0069 80 E4 04                   AND     AH,SYS_SHIFT    ; MASK ALL BUT SYS KEY BIT
006C B1 05                      MOV     CL,5            ; SHIFT THE SYSTEM KEY BIT OVER TO
006E D2 E4                      SHL     AH,CL           ;  BIT 7 POSITION
0070 A0 0018 R                  MOV     AL,@KB_FLAG_1   ; GET SHIFT STATES BACK
0073 24 73                      AND     AL,01110011B    ; ELIMINATE SYS_SHIFT, HOLD_STATE, AND INS_SHIFT
0075 0A E0                      OR      AH,AL           ; MERGE THE REMAINING BITS INTO AH
0077 A0 0096 R                  MOV     AL,@KB_FLAG_3   ; GET RIGHT CTL AND ALT
007A 24 0C                      AND     AL,00001100B    ; ELIMINATE LC_E0 AND LC_E1
007C 0A E0                      OR      AH,AL           ; OR THE SHIFT_FLAGS TOGETHER
007E A0 0017 R          K3:     MOV     AL,@KB_FLAG     ; GET THE SHIFT STATUS FLAGS
0081 EB A9                      JMP     K10_EXIT        ; RETURN TO CALLER

                        ;------ SET TYPAMATIC RATE AND DELAY

0083 3C 05              K300:   CMP     AL,5            ; CORRECT FUNCTION CALL?
0085 75 A5                      JNE     K10_EXIT        ;  NO, RETURN
0087 F6 C3 E0                   TEST    BL,0E0H         ; TEST FOR OUT-OF-RANGE RATE
008A 75 A0                      JNZ     K10_EXIT        ; RETURN IF SO
008C F6 C7 FC                   TEST    BH,0FCH         ; TEST FOR OUT-OF-RANGE DELAY
008F 75 9B                      JNZ     K10_EXIT        ; RETURN IF SO
0091 B0 F3                      MOV     AL,KB_TYPA_RD   ; COMMAND FOR TYPAMATIC RATE/DELAY
0093 E8 064B R                  CALL    SND_DATA        ; SEND TO KEYBOARD
0096 B9 0005                    MOV     CX,5            ; SHIFT COUNT
0099 D2 E7                      SHL     BH,CL           ; SHIFT DELAY OVER
009B 8A C3                      MOV     AL,BL           ; PUT IN RATE
009D 0A C7                      OR      AL,BH           ; AND DELAY
009F E8 064B R                  CALL    SND_DATA        ; SEND TO KEYBOARD
00A2 EB 88                      JMP     K10_EXIT        ; RETURN TO CALLER

                        ;------ WRITE TO KEYBOARD BUFFER

00A4 56                 K500:   PUSH    SI              ; SAVE SI
00A5 FA                         CLI
00A6 8B 1E 001C R               MOV     BX,@BUFFER_TAIL ; GET THE "IN TO" POINTER TO THE BUFFER
00AA 8B F3                      MOV     SI,BX           ; SAVE A COPY IN CASE BUFFER NOT FULL
00AC E8 0168 R                  CALL    K4              ; BUMP THE POINTER TO SEE IF BUFFER IS FULL
00AF 3B 1E 001A R               CMP     BX,@BUFFER_HEAD ; WILL THE BUFFER OVERRUN IF WE STORE THIS?
00B3 74 0B                      JE      K502            ;  YES - INFORM CALLER OF ERROR
00B5 89 0C                      MOV     [SI],CX         ;  NO  - PUT THE ASCII/SCAN CODE INTO BUFFER
00B7 89 1E 001C R               MOV     @BUFFER_TAIL,BX ; ADJUST "IN TO" POINTER TO REFLECT CHANGE
00BB 2A C0                      SUB     AL,AL           ; TELL CALLER THAT OPERATION WAS SUCCESSFUL
00BD EB 03 90                   JMP     K504            ; SUB INSTRUCTION ALSO RESETS CARRY FLAG
00C0                    K502:
00C0 B0 01                      MOV     AL,01H          ; BUFFER FULL INDICATION
00C2                    K504:
00C2 FB                         STI
00C3 5E                         POP     SI              ; RECOVER SI
00C4 E9 002C R                  JMP     K10_EXIT        ; RETURN TO CALLER WITH STATUS IN AL

00C7                    KEYBOARD_IO_1   ENDP
```

SECTION 5

```
                        PAGE
                        ;------ READ THE KEY TO FIGURE OUT WHAT TO DO ------------------------

00C7                    K1S     PROC    NEAR
00C7 8B 1E 001A R               MOV     BX,@BUFFER_HEAD ; GET POINTER TO HEAD OF BUFFER
00CB 3B 1E 001C R               CMP     BX,@BUFFER_TAIL ; TEST END OF BUFFER
00CF 75 07                      JNE     K1U             ; IF ANYTHING IN BUFFER DONT DO INTERRUPT

00D1 B8 9002                    MOV     AX,09002H       ; MOVE IN WAIT CODE & TYPE
00D4 CD 15                      INT     15H             ; PERFORM OTHER FUNCTION
00D6                    K1T:                            ; ASCII READ
00D6 FB                         STI                     ; INTERRUPTS BACK ON DURING LOOP
00D7 90                         NOP                     ; ALLOW AN INTERRUPT TO OCCUR
00D8 FA                 K1U:    CLI                     ; INTERRUPTS BACK OFF
00D9 8B 1E 001A R               MOV     BX,@BUFFER_HEAD ; GET POINTER TO HEAD OF BUFFER
00DD 3B 1E 001C R               CMP     BX,@BUFFER_TAIL ; TEST END OF BUFFER
00E1 53                         PUSH    BX              ; SAVE ADDRESS
00E2 9C                         PUSHF                   ; SAVE FLAG
00E3 E8 06D8 R                  CALL    MAKE_LED        ; GO GET MODE INDICATOR DATA BYTE
00E6 8A 1E 0097 R               MOV     BL,@KB_FLAG_2   ; GET PREVIOUS BITS
00EA 32 D8                      XOR     BL,AL           ; SEE IF ANY DIFFERENT
00EC 80 E3 07                   AND     BL,07H          ; ISOLATE INDICATOR BITS
00EF 74 04                      JZ      K1V             ; IF NO CHANGE BYPASS UPDATE

00F1 E8 069A R                  CALL    SND_LED1        ; GO TURN ON MODE INDICATORS
00F4 FA                         CLI                     ; DISABLE INTERRUPTS
00F5 9D                 K1V:    POPF                    ; RESTORE FLAGS
00F6 5B                         POP     BX              ; RESTORE ADDRESS
00F7 74 DD                      JZ      K1T             ; LOOP UNTIL SOMETHING IN BUFFER

00F9 8B 07                      MOV     AX,[BX]         ; GET SCAN CODE AND ASCII CODE
00FB E8 0168 R                  CALL    K4              ; MOVE POINTER TO NEXT POSITION
00FE 89 1E 001A R               MOV     @BUFFER_HEAD,BX ; STORE VALUE IN VARIABLE
0102 C3                         RET                     ; RETURN
0103                    K1S     ENDP


                        ;------ READ THE KEY TO SEE IF ONE IS PRESENT ------------------------

0103                    K2S     PROC    NEAR
0103 FA                         CLI                     ; INTERRUPTS OFF
0104 8B 1E 001A R               MOV     BX,@BUFFER_HEAD ; GET HEAD POINTER
0108 3B 1E 001C R               CMP     BX,@BUFFER_TAIL ; IF EQUAL (Z=1) THEN NOTHING THERE
010C 8B 07                      MOV     AX,[BX]
010E 9C                         PUSHF                   ; SAVE FLAGS

010F 50                         PUSH    AX              ; SAVE CODE
0110 E8 06D8 R                  CALL    MAKE_LED        ; GO GET MODE INDICATOR DATA BYTE
0113 8A 1E 0097 R               MOV     BL,@KB_FLAG_2   ; GET PREVIOUS BITS
0117 32 D8                      XOR     BL,AL           ; SEE IF ANY DIFFERENT
0119 80 E3 07                   AND     BL,07H          ; ISOLATE INDICATOR BITS
011C 74 03                      JZ      K2T             ; IF NO CHANGE BYPASS UPDATE

011E E8 0687 R                  CALL    SND_LED         ; GO TURN ON MODE INDICATORS
0121 58                 K2T:    POP     AX              ; RESTORE CODE
0122 9D                         POPF                    ; RESTORE FLAGS
0123 FB                         STI                     ; INTERRUPTS BACK ON
0124 C3                         RET                     ; RETURN
0125                    K2S     ENDP


                        ;------ ROUTINE TO TRANSLATE SCAN CODE PAIRS FOR EXTENDED CALLS

0125                    K10_E_XLAT:
0125 3C F0                      CMP     AL,0F0h         ; IS IT ONE OF THE FILL-INs?
0127 75 06                      JNE     K10_E_RET       ;  NO, PASS IT ON
0129 0A E4                      OR      AH,AH           ; AH = 0 IS SPECIAL CASE
012B 74 02                      JZ      K10_E_RET       ;  PASS THIS ON UNCHANGED
012D 32 C0                      XOR     AL,AL           ;  OTHERWISE SET AL = 0
012F                    K10_E_RET:
012F C3                         RET                     ; GO BACK


                        ;------ ROUTINE TO TRANSLATE SCAN CODE PAIRS FOR STANDARD CALLS

0130                    K10_S_XLAT:
0130 80 FC E0                   CMP     AH,0E0h         ; IS IT KEYPAD ENTER OR / ?
0133 75 12                      JNE     K10_S2          ;  NO, CONTINUE
0135 3C 0D                      CMP     AL,0Dh          ; KEYPAD ENTER CODE?
0137 74 09                      JE      K10_S1          ;  YES, MASSAGE A BIT
0139 3C 0A                      CMP     AL,0Ah          ; CTRL KEYPAD ENTER CODE?
013B 74 05                      JE      K10_S1          ;  YES, MASSAGE THE SAME
013D B4 35                      MOV     AH,35h          ;  NO, MUST BE KEYPAD /
013F EB 23 90                   JMP     K10_USE         ; GIVE TO CALLER
0142 B4 1C              K10_S1: MOV     AH,1Ch          ; CONVERT TO COMPATIBLE OUTPUT
0144 EB 1E 90                   JMP     K10_USE         ; GIVE TO CALLER

0147 80 FC 84           K10_S2: CMP     AH,84h          ; IS IT ONE OF THE EXTENDED ONES?
014A 77 1A                      JA      K10_DIS         ;  YES, THROW AWAY AND GET ANOTHER CHAR

014C 3C F0                      CMP     AL,0F0h         ; IS IT ONE OF THE FILL-INs?
014E 75 07                      JNE     K10_S3          ;  NO, TRY LAST TEST
0150 0A E4                      OR      AH,AH           ; AH = 0 IS SPECIAL CASE
0152 74 10                      JZ      K10_USE         ;  PASS THIS ON UNCHANGED
0154 EB 10 90                   JMP     K10_DIS         ;  THROW AWAY THE REST

0157 3C E0              K10_S3: CMP     AL,0E0h         ; IS IT AN EXTENSION OF A PREVIOUS ONE?
0159 75 09                      JNE     K10_USE         ;  NO, MUST BE A STANDARD CODE
015B 0A E4                      OR      AH,AH           ; AH = 0 IS SPECIAL CASE
015D 74 05                      JZ      K10_USE         ; JUMP IF AH = 0
015F 32 C0                      XOR     AL,AL           ; CONVERT TO COMPATIBLE OUTPUT
0161 EB 01 90                   JMP     K10_USE         ; PASS IT ON TO CALLER

0164                    K10_USE:
0164 F8                         CLC                     ; CLEAR CARRY TO INDICATE GOOD CODE
0165 C3                         RET                     ; RETURN
0166                    K10_DIS:
0166 F9                         STC                     ; SET CARRY TO INDICATE DISCARD CODE
0167 C3                         RET                     ; RETURN
```

```
                        PAGE
                        ;--------------------------------------------------------
                        ;       INCREMENT BUFFER POINTER ROUTINE                 -
                        ;--------------------------------------------------------

0168                    K4      PROC    NEAR
0168 43                         INC     BX                      ; MOVE TO NEXT WORD IN LIST
0169 43                         INC     BX
                        ;
016A 3B 1E 0082 R               CMP     BX,@BUFFER_END          ; AT END OF BUFFER?
016E 75 04                      JNE     K5                      ; NO, CONTINUE
0170 8B 1E 0080 R               MOV     BX,@BUFFER_START        ; YES, RESET TO BUFFER BEGINNING
0174 C3                 K5:     RET
0175                    K4      ENDP

                        ;--- HARDWARE INT  09 H -- ( IRQ LEVEL  1 ) ---------------------------------
                        ;                                                                            :
                        ;       KEYBOARD INTERRUPT ROUTINE                                           :
                        ;                                                                            :
                        ;--------------------------------------------------------------------------

0175                    KB_INT_1 PROC   FAR
0175 FB                         STI                             ; ENABLE INTERRUPTS
0176 55                         PUSH    BP
0177 50                         PUSH    AX
0178 53                         PUSH    BX
0179 51                         PUSH    CX
017A 52                         PUSH    DX
017B 56                         PUSH    SI
017C 57                         PUSH    DI
017D 1E                         PUSH    DS
017E 06                         PUSH    ES
017F FC                         CLD                             ; FORWARD DIRECTION
0180 E8 0000 E                  CALL    DDS                     ; SET UP ADDRESSING

                        ;-----  WAIT FOR KEYBOARD DISABLE COMMAND TO BE ACCEPTED

0183 B0 AD                      MOV     AL,DIS_KBD              ; DISABLE THE KEYBOARD COMMAND
0185 E8 063C R                  CALL    SHIP_IT                 ; EXECUTE DISABLE
0188 FA                         CLI                             ; DISABLE INTERRUPTS
0189 2B C9                      SUB     CX,CX                   ; SET MAXIMUM TIMEOUT
018B                    KB_INT_01:
018B E4 64                      IN      AL,STATUS_PORT          ; READ ADAPTER STATUS
018D A8 02                      TEST    AL,INPT_BUF_FULL        ; CHECK INPUT BUFFER FULL STATUS BIT
018F E0 FA                      LOOPNZ  KB_INT_01               ; WAIT FOR COMMAND TO BE ACCEPTED

                        ;-----  READ CHARACTER FROM KEYBOARD INTERFACE

0191 E4 60                      IN      AL,PORT_A               ; READ IN THE CHARACTER

                        ;-----  SYSTEM HOOK  INT 15H - FUNCTION 4FH  (ON HARDWARE INTERRUPT LEVEL 9H)

0193 B4 4F                      MOV     AH,04FH                 ; SYSTEM INTERCEPT - KEY CODE FUNCTION
0195 F9                         STC                             ; SET CY= 1 (IN CASE OF IRET)
0196 CD 15                      INT     15H                     ; CASSETTE CALL   (AL)= KEY SCAN CODE
                                                                ;  RETURNS CY= 1 FOR INVALID FUNCTION
0198 72 03                      JC      KB_INT_02               ; CONTINUE IF CARRY FLAG SET ((AL)=CODE)
019A E9 03A0 R                  JMP     K26                     ; EXIT IF SYSTEM HANDLED SCAN CODE
                                                                ;  EXIT HANDLES HARDWARE EOI AND ENABLE

                        ;------ CHECK FOR A RESEND COMMAND TO KEYBOARD

019D                    KB_INT_02:                              ;       (AL)= SCAN CODE
019D FB                         STI                             ; ENABLE INTERRUPTS AGAIN
019E 3C FE                      CMP     AL,KB_RESEND            ; IS THE INPUT A RESEND
01A0 74 0D                      JE      KB_INT_4                ; GO IF RESEND

                        ;------ CHECK FOR RESPONSE TO A COMMAND TO KEYBOARD

01A2 3C FA                      CMP     AL,KB_ACK               ; IS THE INPUT AN ACKNOWLEDGE
01A4 75 12                      JNZ     KB_INT_2                ; GO IF NOT

                        ;------ A COMMAND TO THE KEYBOARD WAS ISSUED

01A6 FA                         CLI                             ; DISABLE INTERRUPTS
01A7 80 0E 0097 R 10            OR      @KB_FLAG_2,KB_FA        ; INDICATE ACK RECEIVED
01AC E9 03A0 R                  JMP     K26                     ; RETURN IF NOT (ACK RETURNED FOR DATA)

                        ;------ RESEND THE LAST BYTE

01AF                    KB_INT_4:
01AF FA                         CLI                             ; DISABLE INTERRUPTS
01B0 80 0E 0097 R 20            OR      @KB_FLAG_2,KB_FE        ; INDICATE RESEND RECEIVED
01B5 E9 03A0 R                  JMP     K26                     ; RETURN IF NOT (ACK RETURNED FOR DATA)


                        ;------ UPDATE MODE INDICATORS IF CHANGE IN STATE

01B8                    KB_INT_2:
01B8 50                         PUSH    AX                      ; SAVE DATA IN
01B9 E8 06D8 R                  CALL    MAKE_LED                ; GO GET MODE INDICATOR DATA BYTE
01BC 8A 1E 0097 R               MOV     BL,@KB_FLAG_2           ; GET PREVIOUS BITS
01C0 32 D8                      XOR     BL,AL                   ; SEE IF ANY DIFFERENT
01C2 80 E3 07                   AND     BL,KB_LEDS              ; ISOLATE INDICATOR BITS
01C5 74 03                      JZ      UP0                     ; IF NO CHANGE BYPASS UPDATE
01C7 E8 0687 R                  CALL    SND_LED                 ; GO TURN ON MODE INDICATORS
01CA 58                 UP0:    POP     AX                      ; RESTORE DATA IN
```

```
                        PAGE
                        ;--------------------------------------------------------------------------
                        ;                START OF KEY PROCESSING                                   -
                        ;--------------------------------------------------------------------------

01CB 8A E0                      MOV     AH,AL                   ; SAVE SCAN CODE IN AH ALSO

                        ;------ TEST FOR OVERRUN SCAN CODE FROM KEYBOARD

01CD 3C FF                      CMP     AL,KB_OVER_RUN          ; IS THIS AN OVERRUN CHAR?
01CF 75 03                      JNZ     K16                     ; NO, TEST FOR SHIFT KEY
01D1 E9 062D R                  JMP     K62                     ; BUFFER_FULL_BEEP

01D4 0E                 K16:    PUSH    CS
01D5 07                         POP     ES                      ; ESTABLISH ADDRESS OF TABLES
01D6 8A 3E 0096 R               MOV     BH,@KB_FLAG_3           ; LOAD FLAGS FOR TESTING

                        ;------ TEST TO SEE IF A READ_ID IS IN PROGRESS

01DA F6 C7 C0                   TEST    BH,RD_ID+LC_AB          ; ARE WE DOING A READ ID?
01DD 74 34                      JZ      NOT_ID                  ; CONTINUE IF NOT
01DF 79 10                      JNS     TST_ID_2                ; IS THE RD_ID FLAG ON?
01E1 3C AB                      CMP     AL,TD_1                 ; IS THIS THE 1ST ID CHARACTER?
01E3 75 05                      JNE     RST_RD_ID
01E5 80 0E 0096 R 40            OR      @KB_FLAG_3,LC_AB        ; INDICATE 1ST ID WAS OK
01EA                    RST_RD_ID:
01EA 80 26 0096 R 7F            AND     @KB_FLAG_3,NOT RD_ID    ; RESET THE READ ID FLAG
01EF EB 1F                      JMP     SHORT ID_EX             ; AND EXIT

01F1                    TST_ID_2:
01F1 80 26 0096 R BF            AND     @KB_FLAG_3,NOT LC_AB    ; RESET FLAG
01F6 3C 54                      CMP     AL,TD_2A                ; IS THIS THE 2ND ID CHARACTER?
01F8 74 11                      JE      KX_BIT                  ;  JUMP IF SO
01FA 3C 41                      CMP     AL,ID_2                 ; IS THIS THE 2ND ID CHARACTER?
01FC 75 12                      JNE     ID_EX                   ;  LEAVE IF NOT

                        ;------ A READ ID SAID THAT IT WAS ENHANCED KEYBOARD

01FE F6 C7 20                   TEST    BH,SET_NUM_LK           ; SHOULD WE SET NUM LOCK?
0201 74 08                      JZ      KX_BIT                  ; EXIT IF NOT
0203 80 0E 0017 R 20            OR      @KB_FLAG,NUM_STATE      ; FORCE NUM LOCK ON
0208 E8 0687 R                  CALL    SND_LED                 ; GO SET THE NUM LOCK INDICATOR
020B 80 0E 0096 R 10    KX_BIT: OR      @KB_FLAG_3,KBX          ; INDICATE ENHANCED KEYBOARD WAS FOUND
0210 E9 03A0 R          ID_EX:  JMP     K26                     ; EXIT

                        NOT_ID:
0213 3C E0                      CMP     AL,MC_E0                ; IS THIS THE GENERAL MARKER CODE?
0215 75 07                      JNE     TEST_E1                 ;
0217 80 0E 0096 R 12            OR      @KB_FLAG_3,LC_E0+KBX    ; SET FLAG BIT, SET KBX, AND
021C EB 09                      JMP     SHORT EXIT              ; THROW AWAY THIS CODE

021E                    TEST_E1:
021E 3C E1                      CMP     AL,MC_E1                ; IS THIS THE PAUSE KEY?
0220 75 08                      JNE     NOT_HC                  ;
0222 80 0E 0096 R 11            OR      @KB_FLAG_3,LC_E1+KBX    ; SET FLAG, PAUSE KEY MARKER CODE
0227 E9 03A5 R          EXIT:   JMP     K26A                    ; THROW AWAY THIS CODE

022A                    NOT_HC:
022A 24 7F                      AND     AL,07FH                 ; TURN OFF THE BREAK BIT
022C F6 C7 02                   TEST    BH,LC_E0                ; LAST CODE THE E0 MARKER CODE?
022F 74 0C                      JZ      NOT_LC_E0               ; JUMP IF NOT

0231 B9 0002                    MOV     CX,2                    ; LENGTH OF SEARCH
0234 BF 0006 E                  MOV     DI,OFFSET K6+6          ; IS THIS A SHIFT KEY?
0237 F2/ AE                     REPNE   SCASB                   ; CHECK IT
0239 75 65                      JNE     K16A                    ; NO, CONTINUE KEY PROCESSING
023B EB 49                      JMP     SHORT K16B              ; YES, THROW AWAY & RESET FLAG

023D                    NOT_LC_E0:
023D F6 C7 01                   TEST    BH,LC_E1                ; LAST CODE THE E1 MARKER CODE?
0240 74 1D                      JZ      T_SYS_KEY               ; JUMP IF NOT

0242 B9 0004                    MOV     CX,4                    ; LENGTH OF SEARCH
0245 BF 0004 E                  MOV     DI,OFFSET K6+4          ; IS THIS AN ALT, CTL, OR SHIFT?
0248 F2/ AE                     REPNE   SCASB                   ; CHECK IT
024A 74 DB                      JE      EXIT                    ; THROW AWAY IF SO

024C 3C 45                      CMP     AL,NUM_KEY              ; IS IT THE PAUSE KEY?
024E 75 36                      JNE     K16B                    ; NO, THROW AWAY & RESET FLAG
0250 F6 C4 80                   TEST    AH,80H                  ; YES, IS IT THE BREAK OF THE KEY?
0253 75 31                      JNZ     K16B                    ;  YES, THROW THIS AWAY, TOO
0255 F6 06 0018 R 08            TEST    @KB_FLAG_1,HOLD_STATE   ;  NO, ARE WE PAUSED ALREADY?
025A 75 2A                      JNZ     K16B                    ;   YES, THROW AWAY
025C E9 04DB R                  JMP     K39P                    ;  NO, THIS IS THE REAL PAUSE STATE

                        ;------ TEST FOR SYSTEM KEY

025F                    T_SYS_KEY:
025F 3C 54                      CMP     AL,SYS_KEY              ; IS IT THE SYSTEM KEY?
0261 75 3D                      JNE     K16A                    ; CONTINUE IF NOT

0263 F6 C4 80                   TEST    AH,080H                 ; CHECK IF THIS A BREAK CODE
0266 75 21                      JNZ     K16C                    ; DON'T TOUCH SYSTEM INDICATOR IF TRUE

0268 F6 06 0018 R 04            TEST    @KB_FLAG_1,SYS_SHIFT    ; SEE IF IN SYSTEM KEY HELD DOWN
026D 75 17                      JNZ     K16B                    ; IF YES, DON'T PROCESS SYSTEM INDICATOR

026F 80 0E 0018 R 04            OR      @KB_FLAG_1,SYS_SHIFT    ; INDICATE SYSTEM KEY DEPRESSED
0274 B0 20                      MOV     AL,EOI                  ; END OF INTERRUPT COMMAND
0276 E6 20                      OUT     020H,AL                 ; SEND COMMAND TO INTERRUPT CONTROL PORT
                                                                ; INTERRUPT-RETURN-NO-EOI
0278 B0 AE                      MOV     AL,ENA_KBD              ; INSURE KEYBOARD IS ENABLED
027A E8 063C R                  CALL    SHIP_IT                 ; EXECUTE ENABLE
027D BB 8500                    MOV     AX,08500H               ; FUNCTION VALUE FOR MAKE OF SYSTEM KEY
0280 FB                         STI                             ; MAKE SURE INTERRUPTS ENABLED
0281 CD 15                      INT     15H                     ; USER INTERRUPT
0283 E9 03AF R                  JMP     K27A                    ; END PROCESSING

0286 E9 03A0 R          K16B:   JMP     K26                     ; IGNORE SYSTEM KEY

0289 80 26 0018 R FB    K16C:   AND     @KB_FLAG_1,NOT SYS_SHIFT ;TURN OFF SHIFT KEY HELD DOWN
028E B0 20                      MOV     AL,EOI                  ; END OF INTERRUPT COMMAND
0290 E6 20                      OUT     020H,AL                 ; SEND COMMAND TO INTERRUPT CONTROL PORT
```

```
                                ; INTERRUPT-RETURN-NO-EOI
0292 B0 AE                      MOV     AL,ENA_KBD              ; INSURE KEYBOARD IS ENABLED
0294 E8 063C R                  CALL    SHIP_IT                 ; EXECUTE ENABLE
0297 B8 8501                    MOV     AX,08501H               ; FUNCTION VALUE FOR BREAK OF SYSTEM KEY
029A FB                         STI                             ; MAKE SURE INTERRUPTS ENABLED
029B CD 15                      INT     15H                     ; USER INTERRUPT
029D E9 03AF R                  JMP     K27A                    ; IGNORE SYSTEM KEY

                        ;------ TEST FOR SHIFT KEYS

02A0 8A 1E 0017 R       K16A:   MOV     BL,@KB_FLAG             ; PUT STATE FLAGS IN BL
02A4 BF 0000 E                  MOV     DI,OFFSET K6            ; SHIFT KEY TABLE
02A7 B9 0000 E                  MOV     CX,OFFSET K6L           ; LENGTH
02AA F2/ AE                     REPNE   SCASB                   ; LOOK THROUGH THE TABLE FOR A MATCH
02AC 8A C4                      MOV     AL,AH                   ; RECOVER SCAN CODE
02AE 74 03                      JE      K17                     ; JUMP IF MATCH FOUND
02B0 E9 038C R                  JMP     K25                     ; IF NO MATCH, THEN SHIFT NOT FOUND

                        ;------ SHIFT KEY FOUND

02B3 81 EF 0001 E       K17:    SUB     DI,OFFSET K6+1          ; ADJUST PTR TO SCAN CODE MTCH
02B7 2E: 8A A5 0000 E           MOV     AH,CS:K7[DI]            ; GET MASK INTO AH
02BC B1 02                      MOV     CL,2                    ; SET UP COUNT FOR FLAG SHIFTS
02BE A8 80                      TEST    AL,80H                  ; TEST FOR BREAK KEY
02C0 74 03                      JZ      K17C
02C2 EB 78 90                   JMP     K23                     ; JUMP IF BREAK

                        ;------ SHIFT MAKE FOUND, DETERMINE SET OR TOGGLE

02C5 80 FC 10           K17C:   CMP     AH,SCROLL_SHIFT
02C8 73 21                      JAE     K18                     ; IF SCROLL SHIFT OR ABOVE, TOGGLE KEY

                        ;------ PLAIN SHIFT KEY, SET SHIFT ON

02CA 08 26 0017 R               OR      @KB_FLAG,AH             ; TURN ON SHIFT BIT
02CE F6 C4 0C                   TEST    AH,CTL_SHIFT+ALT_SHIFT  ; IS IT ALT OR CTRL?
02D1 75 03                      JNZ     K17D                    ;  YES, MORE FLAGS TO SET
02D3 E9 03A0 R                  JMP     K26                     ;  NO, INTERRUPT_RETURN
02D6 F6 C7 02           K17D:   TEST    BH,LC_E0                ; IS THIS ONE OF THE NEW KEYS?
02D9 74 07                      JZ      K17E                    ;  NO, JUMP
02DB 08 26 0096 R               OR      @KB_FLAG_3,AH           ; SET BITS FOR RIGHT CTRL, ALT
02DF E9 03A0 R                  JMP     K26                     ; INTERRUPT_RETURN
02E2 D2 EC              K17E:   SHR     AH,CL                   ; MOVE FLAG BITS TWO POSITIONS
02E4 08 26 0018 R               OR      @KB_FLAG_1,AH           ; SET BITS FOR LEFT CTRL, ALT
02E8 E9 03A0 R                  JMP     K26                     ; INTERRUPT_RETURN

                        ;------ TOGGLED SHIFT KEY, TEST FOR 1ST MAKE OR NOT

02EB                    K18:                                    ; SHIFT-TOGGLE
02EB F6 C3 04                   TEST    BL,CTL_SHIFT            ; CHECK CTL SHIFT STATE
02EE 74 03                      JZ      K18A                    ; JUMP IF NOT CTL STATE
02F0 E9 038C R                  JMP     K25                     ; JUMP IF CTL STATE
02F3 3C 52              K18A:   CMP     AL,INS_KEY              ; CHECK FOR INSERT KEY
02F5 75 21                      JNE     K22                     ; JUMP IF NOT INSERT KEY
02F7 F6 C3 08                   TEST    BL,ALT_SHIFT            ; CHECK FOR ALTERNATE SHIFT
02FA 74 03                      JZ      K18B                    ; JUMP IF NOT ALTERNATE SHIFT
02FC E9 038C R                  JMP     K25                     ; JUMP IF ALTERNATE SHIFT
02FF F6 C7 02           K18B:   TEST    BH,LC_E0                ; IS THIS THE NEW INSERT KEY?
0302 75 14                      JNZ     K22                     ; YES, THIS ONE'S NEVER A "0"
0304 F6 C3 20           K19:    TEST    BL,NUM_STATE            ; CHECK FOR BASE STATE
0307 75 0A                      JNZ     K21                     ; JUMP IF NUM LOCK IS ON
0309 F6 C3 03                   TEST    BL,LEFT_SHIFT+RIGHT_SHIFT  ; TEST FOR SHIFT STATE
030C 74 0A                      JZ      K22                     ; JUMP IF BASE STATE
030E 8A E0              K20:    MOV     AH,AL                   ; PUT SCAN CODE BACK IN AH
0310 EB 7A 90                   JMP     K25                     ; NUMERAL "0", STNDRD. PROCESSING

0313 F6 C3 03           K21:    TEST    BL,LEFT_SHIFT+RIGHT_SHIFT  ; MIGHT BE NUMERIC
0316 74 F6                      JZ      K20                     ; IS NUMERIC, STD. PROC.

0318                    K22:                                    ; SHIFT TOGGLE KEY HIT; PROCESS IT
0318 84 26 0018 R               TEST    AH,@KB_FLAG_1           ; IS KEY ALREADY DEPRESSED?
031C 74 03                      JZ      K22A
031E E9 03A0 R                  JMP     K26                     ; JUMP IF KEY ALREADY DEPRESSED
0321 08 26 0018 R       K22A:   OR      @KB_FLAG_1,AH           ; INDICATE THAT THE KEY IS DEPRESSED
0325 30 26 0017 R               XOR     @KB_FLAG,AH             ; TOGGLE THE SHIFT STATE

                        ;------ TOGGLE LED IF CAPS, NUM, OR SCROLL KEY DEPRESSED

0329 F6 C4 70                   TEST    AH,CAPS_SHIFT+NUM_SHIFT+SCROLL_SHIFT ; SHIFT TOGGLE?
032C 74 05                      JZ      K22B                    ; GO IF NOT
032E 50                         PUSH    AX                      ; SAVE SCAN CODE AND SHIFT MASK
032F E8 0687 R                  CALL    SND_LED                 ; GO TURN MODE INDICATORS ON
0332 58                         POP     AX                      ; RESTORE SCAN CODE

0333 3C 52              K22B:   CMP     AL,INS_KEY              ; TEST FOR 1ST MAKE OF INSERT KEY
0335 75 69                      JNE     K26                     ; JUMP IF NOT INSERT KEY
0337 8A E0                      MOV     AH,AL                   ; SCAN CODE IN BOTH HALVES OF AX
0339 EB 7F 90                   JMP     K26                     ; FLAGS UPDATED, PROC. FOR BUFFER

                        ;------ BREAK SHIFT FOUND

033C                    K23:                                    ; BREAK-SHIFT-FOUND
033C 80 FC 10                   CMP     AH,SCROLL_SHIFT         ; IS THIS A TOGGLE KEY?
033F F6 D4                      NOT     AH                      ; INVERT MASK
0341 73 43                      JAE     K24                     ; YES, HANDLE BREAK TOGGLE
0343 20 26 0017 R               AND     @KB_FLAG,AH             ; TURN OFF SHIFT BIT
0347 80 FC FB                   CMP     AH,NOT CTL_SHIFT        ; IS THIS ALT OR CTL?
034A 77 26                      JA      K23D                    ;  NO, ALL DONE

034C F6 C7 02                   TEST    BH,LC_E0                ; 2ND ALT OR CTL?
034F 74 06                      JZ      K23A                    ; NO, HANDLE NORMALLY
0351 20 26 0096 R               AND     @KB_FLAG_3,AH           ; RESET BIT FOR RIGHT ALT OR CTL
0355 EB 06                      JMP     SHORT K23B              ; CONTINUE
0357 D2 FC              K23A:   SAR     AH,CL                   ; MOVE THE MASK BIT TWO POSITIONS
0359 20 26 0018 R               AND     @KB_FLAG_1,AH           ; RESET BIT FOR LEFT ALT OR CTL
035D 8A E0              K23B:   MOV     AH,AL                   ; SAVE SCAN CODE
035F A0 0096 R                  MOV     AL,@KB_FLAG_3           ; GET RIGHT ALT & CTRL FLAGS
0362 D2 E8                      SHR     AL,CL                   ; MOVE TO BITS 1 & 0
0364 0A 06 0018 R               OR      AL,@KB_FLAG_1           ; PUT IN LEFT ALT & CTL FLAGS
0368 D2 E0                      SHL     AL,CL                   ; MOVE BACK TO BITS 3 & 2
036A 24 0C                      AND     AL,ALT_SHIFT+CTL_SHIFT  ; FILTER OUT OTHER GARBAGE
036C 08 06 0017 R               OR      @KB_FLAG,AL             ; PUT RESULT IN THE REAL FLAGS
0370 8A C4                      MOV     AL,AH                   ; RECOVER SAVED SCAN CODE
```

```
0372 3C B8              K23D:   CMP     AL,ALT_KEY+80H          ; IS THIS ALTERNATE SHIFT RELEASE
0374 75 2A                      JNE     K26                     ; INTERRUPT_RETURN
                        ;------ ALTERNATE SHIFT KEY RELEASED, GET THE VALUE INTO BUFFER
0376 A0 0019 R                  MOV     AL,@ALT_INPUT
0379 B4 00                      MOV     AH,0                    ; SCAN CODE OF 0
037B 88 26 0019 R               MOV     @ALT_INPUT,AH           ; ZERO OUT THE FIELD
037F 3C 00                      CMP     AL,0                    ; WAS THE INPUT = 0?
0381 74 1D                      JE      K26                     ; INTERRUPT_RETURN
0383 E9 0601 R                  JMP     K61                     ; IT WASN'T, SO PUT IN BUFFER
0386                    K24:                                    ; BREAK-TOGGLE
0386 20 26 0018 R               AND     @KB_FLAG_1,AH           ; INDICATE NO LONGER DEPRESSED
038A EB 14                      JMP     SHORT K26               ; INTERRUPT_RETURN
                        ;------ TEST FOR HOLD STATE
038C                    K25:                                    ; AL, AH = SCAN CODE
                                                                ; NO-SHIFT-FOUND
038C 3C 80                      CMP     AL,80H                  ; TEST FOR BREAK KEY
038E 73 10                      JAE     K26                     ; NOTHING FOR BREAK CHARS FROM HERE ON
0390 F6 06 0018 R 08            TEST    @KB_FLAG_1,HOLD_STATE   ; ARE WE IN HOLD STATE
0395 74 23                      JZ      K28                     ; BRANCH AROUND TEST IF NOT
0397 3C 45                      CMP     AL,NUM_KEY
0399 74 05                      JE      K26                     ; CAN'T END HOLD ON NUM_LOCK
039B 80 26 0018 R F7            AND     @KB_FLAG_1,NOT HOLD_STATE  ; TURN OFF THE HOLD STATE BIT
03A0                    K26:
03A0 80 26 0096 R FC            AND     @KB_FLAG_3,NOT LC_E0+LC_E1 ; RESET LAST CHAR H.C. FLAG
03A5                    K26A:                                   ; INTERRUPT-RETURN
03A5 FA                         CLI                             ; TURN OFF INTERRUPTS
03A6 B0 20                      MOV     AL,EOI                  ; END OF INTERRUPT COMMAND
03A8 E6 20                      OUT     020H,AL                 ; SEND COMMAND TO INTERRUPT CONTROL PORT
03AA                    K27:                                    ; INTERRUPT-RETURN-NO-EOI
03AA B0 AE                      MOV     AL,ENA_KBD              ; INSURE KEYBOARD IS ENABLED
03AC E8 063C R                  CALL    SHIP_IT                 ; EXECUTE ENABLE
03AF FA                 K27A:   CLI                             ; DISABLE INTERRUPTS
03B0 07                         POP     ES                      ; RESTORE REGISTERS
03B1 1F                         POP     DS
03B2 5F                         POP     DI
03B3 5E                         POP     SI
03B4 5A                         POP     DX
03B5 59                         POP     CX
03B6 5B                         POP     BX
03B7 58                         POP     AX
03B8 5D                         POP     BP
03B9 CF                         IRET                            ; RETURN
```

```
                        PAGE
                        ;------ NOT IN HOLD STATE
                                                        ; AL, AH = SCAN CODE (ALL MAKES)
03BA                    K28:                            ; NO-HOLD-STATE
03BA 3C 58                      CMP     AL,88           ; TEST FOR OUT-OF-RANGE SCAN CODES
03BC 77 E2                      JA      K26             ; IGNORE IF OUT-OF-RANGE
                        ;
03BE F6 C3 08                   TEST    BL,ALT_SHIFT    ; ARE WE IN ALTERNATE SHIFT?
03C1 74 0C                      JZ      K28A            ; JUMP IF NOT ALTERNATE
                        ;
03C3 F6 C7 10                   TEST    BH,KBX          ; IS THIS THE ENHANCED KEYBOARD?
03C6 74 0A                      JZ      K29             ; NO, ALT STATE IS REAL
                        ;
03C8 F6 06 0018 R 04            TEST    @KB_FLAG_1,SYS_SHIFT    ; YES, IS SYSREQ KEY DOWN?
03CD 74 03                      JZ      K29             ;  NO, ALT STATE IS REAL
03CF E9 04A3 R          K28A:   JMP     K38             ;  YES, THIS IS PHONY ALT STATE
                        ;                               ;       DUE TO PRESSING SYSREQ
                        ;------ TEST FOR RESET KEY SEQUENCE (CTL ALT DEL)
03D2                    K29:                            ; TEST-RESET
03D2 F6 C3 04                   TEST    BL,CTL_SHIFT    ; ARE WE IN CONTROL SHIFT ALSO?
03D5 74 31                      JZ      K31             ; NO_RESET
03D7 3C 53                      CMP     AL,DEL_KEY      ; SHIFT STATE IS THERE, TEST KEY
03D9 75 2D                      JNE     K31             ; NO_RESET,  IGNORE
                        ;
                        ;------ CTL-ALT-DEL HAS BEEN FOUND, DO I/O CLEANUP
03DB C7 06 0072 R 1234          MOV     @RESET_FLAG,1234H       ; SET FLAG FOR RESET FUNCTION
03E1 E9 0000 E                  JMP     START_T         ; JUMP TO POWER ON DIAGNOSTICS
                        ;
                        ;---------------------  TABLES FOR ALT CASE  -------------------------
                        ;------ ALT-INPUT-TABLE
03E4                    K30     LABEL   BYTE
03E4 52 4F 50 51 4B             DB      82,79,80,81,75
03E9 4C 4D 47 48 49             DB      76,77,71,72,73  ; 10 NUMBERS ON KEYPAD
                        ;------ SUPER-SHIFT-TABLE
03EE 10 11 12 13 14 15          DB      16,17,18,19,20,21       ; A-Z TYPEWRITER CHARS
03F4 16 17 18 19 1E 1F          DB      22,23,24,25,30,31
03FA 20 21 22 23 24 25          DB      32,33,34,35,36,37
0400 26 2C 2D 2E 2F 30          DB      38,44,45,46,47,48
0406 31 32                      DB      49,50
                        ;
                        ;------ IN ALTERNATE SHIFT, RESET NOT FOUND
0408                    K31:                            ; NO-RESET
0408 3C 39                      CMP     AL,57           ; TEST FOR SPACE KEY
040A 75 05                      JNE     K311            ; NOT THERE
040C B0 20                      MOV     AL,' '          ; SET SPACE CHAR
040E E9 05F5 R                  JMP     K57             ; BUFFER_FILL
0411                    K311:
0411 3C 0F                      CMP     AL,15           ; TEST FOR TAB KEY
0413 75 06                      JNE     K312            ; NOT THERE
0415 B8 A500                    MOV     AX,0A500h       ; SET SPECIAL CODE FOR ALT-TAB
0418 E9 05F5 R                  JMP     K57             ; BUFFER_FILL
041B                    K312:
041B 3C 4A                      CMP     AL,74           ; TEST FOR KEYPAD -
041D 74 79                      JE      K37B            ; GO PROCESS
041F 3C 4E                      CMP     AL,78           ; TEST FOR KEYPAD +
0421 74 75                      JE      K37B            ; GO PROCESS
                        ;
                        ;------ LOOK FOR KEY PAD ENTRY
0423                    K32:                            ; ALT-KEY-PAD
0423 BF 03E4 R                  MOV     DI,OFFSET K30   ; ALT-INPUT-TABLE
0426 B9 000A                    MOV     CX,10           ; LOOK FOR ENTRY USING KEYPAD
0429 F2/ AE                     REPNE   SCASB           ; LOOK FOR MATCH
042B 75 18                      JNE     K33             ; NO_ALT_KEYPAD
042D F6 C7 02                   TEST    BH,LC_E0        ; IS THIS ONE OF THE NEW KEYS?
0430 75 6B                      JNZ     K37C            ;  YES, JUMP, NOT NUMPAD KEY
0432 81 EF 03E5 R               SUB     DI,OFFSET K30+1 ; DI NOW HAS ENTRY VALUE
0436 A0 0019 R                  MOV     AL,@ALT_INPUT   ; GET THE CURRENT BYTE
0439 B4 0A                      MOV     AH,10           ; MULTIPLY BY 10
043B F6 E4                      MUL     AH
043D 03 C7                      ADD     AX,DI           ; ADD IN THE LATEST ENTRY
043F A2 0019 R                  MOV     @ALT_INPUT,AL   ; STORE IT AWAY
0442 E9 03A0 R          K32A:   JMP     K26             ; THROW AWAY THAT KEYSTROKE
                        ;
                        ;------ LOOK FOR SUPERSHIFT ENTRY
0445                    K33:                            ; NO-ALT-KEYPAD
0445 C6 06 0019 R 00            MOV     @ALT_INPUT,0    ; ZERO ANY PREVIOUS ENTRY INTO INPUT
044A B9 001A                    MOV     CX,26           ; DI,ES ALREADY POINTING
044D F2/ AE                     REPNE   SCASB           ; LOOK FOR MATCH IN ALPHABET
044F 74 42                      JE      K37A            ; MATCH FOUND, GO FILL THE BUFFER
                        ;
                        ;------ LOOK FOR TOP ROW OF ALTERNATE SHIFT
0451                    K34:                            ; ALT-TOP-ROW
0451 3C 02                      CMP     AL,2            ; KEY WITH '1' ON IT
0453 72 43                      JB      K37B            ; MUST BE ESCAPE
0455 3C 0D                      CMP     AL,13           ; IS IT IN THE REGION
0457 77 05                      JA      K35             ;  NO, ALT-SOMETHING ELSE
0459 80 C4 76                   ADD     AH,118          ; CONVERT PSEUDO SCAN CODE TO RANGE
045C EB 36                      JMP     SHORT K37A      ; GO FILL THE BUFFER
                        ;
                        ;------ TRANSLATE ALTERNATE SHIFT PSEUDO SCAN CODES
045E                    K35:                            ; ALT-FUNCTION
045E 3C 57                      CMP     AL,F11_M        ; IS IT F11?
0460 72 09                      JB      K35A            ;  NO, BRANCH
0462 3C 58                      CMP     AL,F12_M        ; IS IT F12?
0464 77 05                      JA      K35A            ;  NO, BRANCH
0466 80 C4 34                   ADD     AH,52           ; CONVERT TO PSEUDO SCAN CODE
0469 EB 28                      JMP     SHORT K37A      ; GO FILL THE BUFFER
                        ;
046B F6 C7 02           K35A:   TEST    BH,LC_E0        ; DO WE HAVE ONE OF THE NEW KEYS?
046E 74 18                      JZ      K37             ;  NO, JUMP
0470 3C 1C                      CMP     AL,28           ; TEST FOR KEYPAD ENTER
0472 75 06                      JNE     K35B            ; NOT THERE
0474 B8 A600                    MOV     AX,0A600h       ; SPECIAL CODE
0477 E9 05F5 R                  JMP     K57             ; BUFFER FILL
047A 3C 53              K35B:   CMP     AL,83           ; TEST FOR DELETE KEY
047C 74 1F                      JE      K37C            ; HANDLE WITH OTHER EDIT KEYS
```

```
047E 3C 35                      CMP     AL,53                   ; TEST FOR KEYPAD /
0480 75 C0                      JNE     K32A                    ; NOT THERE, NO OTHER E0 SPECIALS
0482 B8 A400                    MOV     AX,0A400h               ; SPECIAL CODE
0485 E9 05F5 R                  JMP     K57                     ; BUFFER FILL

0488 3C 3B              K37:    CMP     AL,59                   ; TEST FOR FUNCTION KEYS (F1)
048A 72 0C                      JB      K37B                    ;  NO FN, HANDLE W/OTHER EXTENDED
048C 3C 44                      CMP     AL,68                   ; IN KEYPAD REGION?
                                                                ; OR NUM_LOCK, SCROLLOCK?
048E 77 B2                      JA      K32A                    ; IF SO, IGNORE
0490 80 C4 2D                   ADD     AH,45                   ; CONVERT TO PSEUDO SCAN CODE

0493 B0 00              K37A:   MOV     AL,0                    ; ASCII CODE OF ZERO
0495 E9 05F5 R                  JMP     K57                     ; PUT IT IN THE BUFFER

0498 B0 F0              K37B:   MOV     AL,0F0h                 ; USE SPECIAL ASCII CODE
049A E9 05F5 R                  JMP     K57                     ; PUT IT IN THE BUFFER

049D 04 50              K37C:   ADD     AL,80                   ; CONVERT SCAN CODE (EDIT KEYS)
049F 8A E0                      MOV     AH,AL                   ; (SCAN CODE NOT IN AH FOR INSERT)
04A1 EB F0                      JMP     K37A                    ; PUT IT IN THE BUFFER

                        ;------ NOT IN ALTERNATE SHIFT

04A3                    K38:                                    ; NOT-ALT-SHIFT
                                                                ; BL STILL HAS SHIFT FLAGS
04A3 F6 C3 04                   TEST    BL,CTL_SHIFT            ; ARE WE IN CONTROL SHIFT?
04A6 75 03                      JNZ     K38A                    ;  YES, START PROCESSING
04A8 E9 0535 R                  JMP     K44                     ; NOT-CTL-SHIFT

                        ;------ CONTROL SHIFT, TEST SPECIAL CHARACTERS

                        ;------ TEST FOR BREAK

04AB 3C 46              K38A:   CMP     AL,SCROLL_KEY           ; TEST FOR BREAK
04AD 75 23                      JNE     K39                     ; JUMP, NO-BREAK
04AF F6 C7 10                   TEST    BH,KBX                  ; IS THIS THE ENHANCED KEYBOARD?
04B2 74 05                      JZ      K38B                    ;  NO, BREAK IS VALID
04B4 F6 C7 02                   TEST    BH,LC_E0                ;  YES, WAS LAST CODE AN E0?
04B7 74 19                      JZ      K39                     ;   NO-BREAK, TEST FOR PAUSE

04B9 8B 1E 001A R       K38B:   MOV     BX,@BUFFER_HEAD         ; RESET BUFFER TO EMPTY
04BD 89 1E 001C R               MOV     @BUFFER_TAIL,BX         ;
04C1 C6 06 0071 R 80            MOV     @BIOS_BREAK,80H         ; TURN ON BIOS_BREAK BIT

                        ;-------- ENABLE KEYBOARD

04C6 B0 AE                      MOV     AL,ENA_KBD              ; ENABLE KEYBOARD
04C8 E8 063C R                  CALL    SHIP_IT                 ; EXECUTE ENABLE
04CB CD 1B                      INT     1BH                     ; BREAK INTERRUPT VECTOR
04CD 2B C0                      SUB     AX,AX                   ; PUT OUT DUMMY CHARACTER
04CF E9 05F5 R                  JMP     K57                     ; BUFFER_FILL

                        ;-------- TEST FOR PAUSE

04D2                    K39:                                    ; NO-BREAK
04D2 F6 C7 10                   TEST    BH,KBX                  ; IS THIS THE ENHANCED KEYBOARD?
04D5 75 2A                      JNZ     K41                     ; YES, THEN THIS CAN'T BE PAUSE
04D7 3C 45                      CMP     AL,NUM_KEY              ; LOOK FOR PAUSE KEY
04D9 75 26                      JNE     K41                     ; NO-PAUSE
04DB 80 0E 0018 R 08    K39P:   OR      @KB_FLAG_1,HOLD_STATE   ; TURN ON THE HOLD FLAG

                        ;-------- ENABLE KEYBOARD

04E0 B0 AE                      MOV     AL,ENA_KBD              ; ENABLE KEYBOARD
04E2 E8 063C R                  CALL    SHIP_IT                 ; EXECUTE ENABLE
04E5 B0 20              K39A:   MOV     AL,EOI                  ; END OF INTERRUPT TO CONTROL PORT
04E7 E6 20                      OUT     020H,AL                 ; ALLOW FURTHER KEYSTROKE INTS

                        ;------ DURING PAUSE INTERVAL, TURN CRT BACK ON

04E9 80 3E 0049 R 07            CMP     @CRT_MODE,7             ; IS THIS BLACK AND WHITE CARD
04EE 74 07                      JE      K40                     ; YES, NOTHING TO DO
04F0 BA 03D8                    MOV     DX,03D8H                ; PORT FOR COLOR CARD
04F3 A0 0065 R                  MOV     AL,@CRT_MODE_SET        ; GET THE VALUE OF THE CURRENT MODE
04F6 EE                         OUT     DX,AL                   ; SET THE CRT MODE, SO THAT CRT IS ON
04F7                    K40:                                    ; PAUSE-LOOP
04F7 F6 06 0018 R 08            TEST    @KB_FLAG_1,HOLD_STATE
04FC 75 F9                      JNZ     K40                     ; LOOP UNTIL FLAG TURNED OFF
04FE E9 03AA R                  JMP     K27                     ; INTERRUPT_RETURN_NO_EOI

                        ;------ TEST SPECIAL CASE KEY 55

0501                    K41:                                    ; NO-PAUSE
0501 3C 37                      CMP     AL,55                   ; TEST FOR */PRTSC KEY
0503 75 10                      JNE     K42                     ; NOT-KEY-55
0505 F6 C7 10                   TEST    BH,KBX                  ; IS THIS THE ENHANCED KEYBOARD?
0508 74 05                      JZ      K41A                    ;  NO, CTL-PRTSC IS VALID
050A F6 C7 02                   TEST    BH,LC_E0                ;  YES, WAS LAST CODE AN E0?
050D 74 20                      JZ      K42B                    ;   NO, TRANSLATE TO A FUNCTION
050F B8 7200            K41A:   MOV     AX,114*256              ; START/STOP PRINTING SWITCH
0512 E9 05F5 R                  JMP     K57                     ; BUFFER_FILL

                        ;------ SET UP TO TRANSLATE CONTROL SHIFT

0515                    K42:                                    ; NOT-KEY-55
0515 3C 0F                      CMP     AL,15                   ; IS IT THE TAB KEY?
0517 74 16                      JE      K42B                    ;  YES, XLATE TO FUNCTION CODE
0519 3C 35                      CMP     AL,53                   ; IS IT THE / KEY?
051B 75 0B                      JNE     K42A                    ;  NO, NO MORE SPECIAL CASES
051D F6 C7 02                   TEST    BH,LC_E0                ;  YES, IS IT FROM THE KEYPAD?
0520 74 06                      JZ      K42A                    ;   NO, JUST TRANSLATE
0522 B8 9500                    MOV     AX,9500h                ;   YES, SPECIAL CODE FOR THIS ONE
0525 E9 05F5 R                  JMP     K57                     ;   BUFFER FILL

0528 BB 0000 E          K42A:   MOV     BX,OFFSET K8            ; SET UP TO TRANSLATE CTL
052B 3C 3B                      CMP     AL,59                   ; IS IT IN CHARACTER TABLE?
052D 72 5E                      JB      K45F                    ;  YES, GO TRANSLATE CHAR
052F BB 0000 E          K42B:   MOV     BX,OFFSET K8            ; SET UP TO TRANSLATE CTL
0532 E9 05E4 R                  JMP     K64                     ;  NO, GO TRANSLATE_SCAN

                        ;------ NOT IN CONTROL SHIFT

0535 3C 37              K44:    CMP     AL,55                   ; PRINT SCREEN KEY?
```

```
0537 75 26                      JNE     K45                     ; NOT-PRINT-SCREEN
0539 F6 C7 10                   TEST    BH,KBX                  ; IS THIS ENHANCED KEYBOARD?
053C 74 07                      JZ      K44A                    ; NO, TEST FOR SHIFT STATE
053E F6 C7 02                   TEST    BH,LC_E0                ; YES, LAST CODE A MARKER?
0541 75 07                      JNZ     K44B                    ;  YES, IS PRINT SCREEN
0543 EB 3B                      JMP     SHORT K45C              ;  NO, XLATE TO "*" CHARACTER
0545 F6 C3 03           K44A:   TEST    BL,LEFT_SHIFT+RIGHT_SHIFT ;NOT 101 KBD, SHIFT KEY DOWN?
0548 74 36                      JZ      K45C                    ; NO, XLATE TO "*" CHARACTER

                        ;------ ISSUE INTERRUPT TO PERFORM PRINT SCREEN FUNCTION
054A B0 AE              K44B:   MOV     AL,ENA_KBD              ; INSURE KEYBOARD IS ENABLED
054C E8 063C R                  CALL    SHIP_IT                 ; EXECUTE ENABLE
054F B0 20                      MOV     AL,EOI                  ; END OF CURRENT INTERRUPT
0551 E6 20                      OUT     020H,AL                 ;  SO FURTHER THINGS CAN HAPPEN
0553 55                         PUSH    BP                      ; SAVE POINTER
0554 CD 05                      INT     5H                      ; ISSUE PRINT SCREEN INTERRUPT
0556 5D                         POP     BP                      ; RESTORE POINTER
0557 80 26 0096 R FC            AND     @KB_FLAG_3,NOT LC_E0+LC_E1 ;ZERO OUT THESE FLAGS
055C E9 03AA R                  JMP     K27                     ; GO BACK WITHOUT EOI OCCURRING


                        ;------ HANDLE THE IN-CORE KEYS
055F                    K45:                                    ; NOT-PRINT-SCREEN
055F 3C 3A                      CMP     AL,58                   ; TEST FOR IN-CORE AREA
0561 77 2C                      JA      K46                     ; JUMP IF NOT

0563 3C 35                      CMP     AL,53                   ; IS THIS THE "/" KEY?
0565 75 05                      JNE     K45A                    ;  NO, JUMP
0567 F6 C7 02                   TEST    BH,LC_E0                ; WAS LAST CODE THE MARKER?
056A 75 14                      JNZ     K45C                    ;  YES, TRANSLATE TO CHARACTER

056C B9 001A            K45A:   MOV     CX,26                   ; LENGTH OF SEARCH
056F BF 03EE R                  MOV     DI,OFFSET K30+10        ; POINT TO TABLE OF A-Z CHARS
0572 F2/ AE                     REPNE   SCASB                   ; IS THIS A LETTER KEY?
0574 75 05                      JNE     K45B                    ;  NO, SYMBOL KEY

0576 F6 C3 40                   TEST    BL,CAPS_STATE           ; ARE WE IN CAPS_LOCK?
0579 75 0A                      JNZ     K45D                    ; TEST FOR SURE
057B F6 C3 03           K45B:   TEST    BL,LEFT_SHIFT+RIGHT_SHIFT ; ARE WE IN SHIFT STATE?
057E 75 0A                      JNZ     K45E                    ;  YES, UPPERCASE
                                                                ;  NO, LOWERCASE
0580 BB 0000 E          K45C:   MOV     BX,OFFSET K10           ; TRANSLATE TO LOWERCASE LETTERS
0583 EB 50                      JMP     SHORT K56
0585                    K45D:                                   ; ALMOST-CAPS-STATE
0585 F6 C3 03                   TEST    BL,LEFT_SHIFT+RIGHT_SHIFT ; CL ON. IS SHIFT ON, TOO?
0588 75 F6                      JNZ     K45C                    ; SHIFTED TEMP OUT OF CAPS STATE
058A BB 0000 E          K45E:   MOV     BX,OFFSET K11           ; TRANSLATE TO UPPERCASE LETTERS
058D EB 46              K45F:   JMP     SHORT K56


                        ;------ TEST FOR KEYS F1 - F10
058F                    K46:                                    ; NOT IN-CORE AREA
058F 3C 44                      CMP     AL,68                   ; TEST FOR F1 - F10
0591 77 02                      JA      K47                     ; JUMP IF NOT
0593 EB 36                      JMP     SHORT K53               ; YES, GO DO FN KEY PROCESS


                        ;------ HANDLE THE NUMERIC PAD KEYS
0595                    K47:                                    ; NOT F1 - F10
0595 3C 53                      CMP     AL,83                   ; TEST FOR NUMPAD KEYS
0597 77 2C                      JA      K52                     ; JUMP IF NOT

                        ;------ KEYPAD KEYS, MUST TEST NUM LOCK FOR DETERMINATION
0599 3C 4A              K48:    CMP     AL,74                   ; SPECIAL CASE FOR MINUS
059B 74 ED                      JE      K45E                    ; GO TRANSLATE
059D 3C 4E                      CMP     AL,78                   ; SPECIAL CASE FOR PLUS
059F 74 E9                      JE      K45E                    ; GO TRANSLATE
05A1 F6 C7 02                   TEST    BH,LC_E0                ; IS THIS ONE OF THE NEW KEYS?
05A4 75 0A                      JNZ     K49                     ;  YES, TRANSLATE TO BASE STATE

05A6 F6 C3 20                   TEST    BL,NUM_STATE            ; ARE WE IN NUM_LOCK?
05A9 75 13                      JNZ     K50                     ; TEST FOR SURE
05AB F6 C3 03                   TEST    BL,LEFT_SHIFT+RIGHT_SHIFT  ; ARE WE IN SHIFT STATE?
05AE 75 13                      JNZ     K51                     ;  IF SHIFTED, REALLY NUM STATE

                        ;------ BASE CASE FOR KEYPAD
05B0 3C 4C              K49:    CMP     AL,76                   ; SPECIAL CASE FOR BASE STATE 5
05B2 75 05                      JNE     K49A                    ; CONTINUE IF NOT KEYPAD 5
05B4 B0 F0                      MOV     AL,0F0h                 ; SPECIAL ASCII CODE
05B6 EB 3D 90                   JMP     K57                     ; BUFFER FILL
05B9 BB 0000 E          K49A:   MOV     BX,OFFSET K10           ; BASE CASE TABLE
05BC EB 26                      JMP     SHORT K64               ; CONVERT TO PSEUDO SCAN


                        ;------ MIGHT BE NUM LOCK, TEST SHIFT STATUS
05BE F6 C3 03           K50:    TEST    BL,LEFT_SHIFT+RIGHT_SHIFT       ;ALMOST-NUM-STATE
05C1 75 ED                      JNZ     K49                     ; SHIFTED TEMP OUT OF NUM STATE
05C3 EB C5              K51:    JMP     SHORT K45E              ; REALLY_NUM_STATE


                        ;------ TEST FOR THE NEW KEY ON WT KEYBOARDS

05C5                    K52:                                    ; NOT A NUMPAD KEY
05C5 3C 56                      CMP     AL,86                   ; IS IT THE NEW WT KEY?
05C7 75 02                      JNE     K53                     ; JUMP IF NOT
05C9 EB B0                      JMP     SHORT K45B              ; HANDLE WITH REST OF LETTER KEYS


                        ;------ MUST BE F11 OR F12
                                                                ; F1 - F10 COME HERE, TOO
05CB F6 C3 03           K53:    TEST    BL,LEFT_SHIFT+RIGHT_SHIFT ;TEST SHIFT STATE
05CE 74 E0                      JZ      K49                     ; JUMP, LOWERCASE PSEUDO SC'S

05D0 BB 0000 E                  MOV     BX,OFFSET K11           ; UPPER CASE PSEUDO SCAN CODES
05D3 EB 0F                      JMP     SHORT K64               ; TRANSLATE_SCAN


                        ;------ TRANSLATE THE CHARACTER

05D5                    K56:                                    ; TRANSLATE-CHAR
05D5 FE C8                      DEC     AL                      ; CONVERT ORIGIN
05D7 2E: D7                     XLAT    CS:K11                  ; CONVERT THE SCAN CODE TO ASCII
05D9 F6 06 0096 R 02            TEST    @KB_FLAG_3,LC_E0        ; IS THIS A NEW KEY?
05DE 74 15                      JZ      K57                     ;  NO, GO FILL BUFFER
05E0 B4 E0                      MOV     AH,MC_E0                ;  YES, PUT SPECIAL MARKER IN AH
05E2 EB 11                      JMP     SHORT K57               ; PUT IT INTO THE BUFFER
```

```
                        ;------ TRANSLATE SCAN FOR PSEUDO SCAN CODES

                        K64:
05E4                    DEC     AL                      ; TRANSLATE-SCAN-ORGD
05E4 FE C8              XLAT    CS:K8                   ; CONVERT ORIGIN
05E6 2E: D7             MOV     AH,AL                   ; CTL TABLE SCAN
05E8 8A E0              MOV     AL,0                    ; PUT VALUE INTO AH
05EA B0 00              TEST    @KB_FLAG_3,LC_E0        ; ZERO ASCII CODE
05EC F6 06 0096 R 02    JZ      K57                     ; IS THIS A NEW KEY?
05F1 74 02              MOV     AL,MC_E0                ;  NO, GO FILL BUFFER
05F3 B0 E0              ;                               ;  YES, PUT SPECIAL MARKER IN AL

                        ;------ PUT CHARACTER INTO BUFFER

05F5                    K57:                            ; BUFFER-FILL
05F5 3C FF              CMP     AL,-1                   ; IS THIS AN IGNORE CHAR
05F7 74 05              JE      K59                     ; YES, DO NOTHING WITH IT
05F9 80 FC FF           CMP     AH,-1                   ; LOOK FOR -1 PSEUDO SCAN
05FC 75 03              JNE     K61                     ; NEAR_INTERRUPT_RETURN
05FE                    ;
05FE E9 03A0 R          K59:                            ; NEAR-INTERRUPT-RETURN
                        JMP     K26                     ; INTERRUPT_RETURN
0601                    ;
0601 8B 1E 001C R       K61:
0605 8B F3              MOV     BX,@BUFFER_TAIL         ; GET THE END POINTER TO THE BUFFER
0607 E8 0168 R          MOV     SI,BX                   ; SAVE THE VALUE
060A 3B 1E 001A R       CALL    K4                      ; ADVANCE THE TAIL
060E 74 1D              CMP     BX,@BUFFER_HEAD         ; HAS THE BUFFER WRAPPED AROUND
0610 89 04              JE      K62                     ; BUFFER_FULL_BEEP
0612 89 1E 001C R       MOV     [SI],AX                 ; STORE THE VALUE
0616 FA                 MOV     @BUFFER_TAIL,BX         ; MOVE THE POINTER UP
0617 B0 20              CLI                             ; TURN OFF INTERRUPTS
0619 E6 20              MOV     AL,EOI                  ; END OF INTERRUPT COMMAND
061B B0 AE              OUT     INTA00,AL               ; SEND COMMAND TO INTERRUPT CONTROL PORT
061D E8 063C R          MOV     AL,ENA_KBD              ; INSURE KEYBOARD IS ENABLED
0620 B8 9102            CALL    SHIP_IT                 ; EXECUTE ENABLE
0623 CD 15              MOV     AX,9102H                ; MOVE IN POST CODE & TYPE
0625 80 26 0096 R FC    INT     15H                     ; PERFORM OTHER FUNCTION
062A E9 03AF R          AND     @KB_FLAG_3,NOT LC_E0+LC_E1 ; RESET LAST CHAR H.C. FLAG
                        JMP     K27A                    ; INTERRUPT_RETURN

                        ;------ BUFFER IS FULL SOUND THE BEEPER

062D                    K62:
062D B0 20              MOV     AL,EOI                  ; ENABLE INTERRUPT CONTROLLER CHIP
062F E6 20              OUT     INTA00,AL               ;
0631 B9 02A6            MOV     CX,678                  ; DIVISOR FOR 1760 HZ
0634 B3 04              MOV     BL,4                    ; SHORT BEEP COUNT (1/16 + 1/64 DELAY)
0636 E8 0000 E          CALL    BEEP                    ; GO TO COMMON BEEP HANDLER
0639 E9 03AA R          JMP     K27                     ; EXIT
                        ;
063C                    KB_INT_1        ENDP
```

```
                        PAGE
                        ;--------------------------------------------------------------------------
                        ;                                                                         -
                        ;       SHIP_IT                                                           -
                        ;                                                                         -
                        ;       THIS ROUTINE HANDLES TRANSMISSION OF COMMAND AND DATA BYTES       -
                        ;       TO THE KEYBOARD CONTROLLER.                                       -
                        ;                                                                         -
                        ;--------------------------------------------------------------------------
1107 063C               SHIP_IT PROC    NEAR
1108 063C 50                    PUSH    AX                      ; SAVE DATA TO SEND
1109
1110                    ;------- WAIT FOR COMMAND TO BE ACCEPTED
1111 063D FA                    CLI                             ; DISABLE INTERRUPTS TILL DATA SENT
1112 063E 2B C9                 SUB     CX,CX                   ; CLEAR TIMEOUT COUNTER
1113 0640               SI0:
1114 0640 E4 64                 IN      AL,STATUS_PORT          ; READ KEYBOARD CONTROLLER STATUS
1115 0642 A8 02                 TEST    AL,INPT_BUF_FULL        ; CHECK FOR ITS INPUT BUFFER BUSY
1116 0644 E0 FA                 LOOPNZ  SI0                     ; WAIT FOR COMMAND TO BE ACCEPTED
1117
1118 0646 58                    POP     AX                      ; GET DATA TO SEND
1119 0647 E6 64                 OUT     STATUS_PORT,AL          ; SEND TO KEYBOARD CONTROLLER
1120 0649 FB                    STI                             ; ENABLE INTERRUPTS AGAIN
1121 064A C3                    RET                             ; RETURN TO CALLER
1122 064B               SHIP_IT ENDP
1123
                        ;--------------------------------------------------------------------------
                        ;                                                                         -
                        ;       SND_DATA                                                          -
                        ;                                                                         -
                        ;       THIS ROUTINE HANDLES TRANSMISSION OF COMMAND AND DATA BYTES       -
                        ;       TO THE KEYBOARD AND RECEIPT OF ACKNOWLEDGEMENTS.  IT ALSO         -
                        ;       HANDLES ANY RETRIES IF REQUIRED                                   -
                        ;                                                                         -
                        ;--------------------------------------------------------------------------
1133
1134 064B               SND_DATA PROC   NEAR
1135 064B 50                    PUSH    AX                      ; SAVE REGISTERS
1136 064C 53                    PUSH    BX                      ; *
1137 064D 51                    PUSH    CX
1138 064E 8A F8                 MOV     BH,AL                   ; SAVE TRANSMITTED BYTE FOR RETRIES
1139 0650 B3 03                 MOV     BL,3                    ; LOAD RETRY COUNT
1140 0652 FA            SD0:    CLI                             ; DISABLE INTERRUPTS
1141 0653 80 26 0097 R CF       AND     @KB_FLAG_2,NOT (KB_FE+KB_FA) ; CLEAR ACK AND RESEND FLAGS
1142
1143                    ;------- WAIT FOR COMMAND TO BE ACCEPTED
1144
1145 0658 2B C9                 SUB     CX,CX
1146 065A E4 64         SD5:    IN      AL,STATUS_PORT
1147 065C A8 02                 TEST    AL,INPT_BUF_FULL
1148 065E E0 FA                 LOOPNZ  SD5                     ; WAIT FOR COMMAND TO BE ACCEPTED
1149
1150 0660 8A C7                 MOV     AL,BH                   ; REESTABLISH BYTE TO TRANSMIT
1151 0662 E6 60                 OUT     PORT_A,AL               ; SEND BYTE
1152 0664 FB                    STI                             ; ENABLE INTERRUPTS
1153 0665 B9 1A00               MOV     CX,01A00H               ; LOAD COUNT FOR 10mS+
1154 0668 F6 06 0097 R 30 SD1:  TEST    @KB_FLAG_2,KB_FE+KB_FA  ; SEE IF EITHER BIT SET
1155 066D 75 0D                 JNZ     SD3                     ; IF SET, SOMETHING RECEIVED GO PROCESS
1156 066F E2 F7                 LOOP    SD1                     ; OTHERWISE WAIT
1157
1158 0671 FE CB         SD2:    DEC     BL                      ; DECREMENT RETRY COUNT
1159 0673 75 DD                 JNZ     SD0                     ; RETRY TRANSMISSION
1160 0675 80 0E 0097 R 80       OR      @KB_FLAG_2,KB_ERR       ; TURN ON TRANSMIT ERROR FLAG
1161 067A EB 07                 JMP     SHORT SD4               ; RETRIES EXHAUSTED FORGET TRANSMISSION
1162
1163 067C F6 06 0097 R 10 SD3:  TEST    @KB_FLAG_2,KB_FA        ; SEE IF THIS IS AN ACKNOWLEDGE
1164 0681 74 EE                 JZ      SD2                     ; IF NOT, GO RESEND
1165
1166 0683 59            SD4:    POP     CX                      ; RESTORE REGISTERS
1167 0684 5B                    POP     BX                      ; *
1168 0685 58                    POP     AX                      ; *
1169 0686 C3                    RET                             ; RETURN, GOOD TRANSMISSION
1170 0687               SND_DATA ENDP
```

SECTION 5

```
                        PAGE
                        ;--------------------------------------------------------------------------
                        ;                                                                         -
                        ;       SND_LED                                                           -
                        ;                                                                         -
                        ;       THIS ROUTINE TURNS ON THE MODE INDICATORS.                        -
                        ;                                                                         -
                        ;--------------------------------------------------------------------------
0687                    SND_LED PROC    NEAR
0687 FA                         CLI                             ; TURN OFF INTERRUPTS
0688 F6 06 0097 R 40            TEST    @KB_FLAG_2,KB_PR_LED    ; CHECK FOR MODE INDICATOR UPDATE
068D 75 47                      JNZ     SL1                     ; DONT UPDATE AGAIN IF UPDATE UNDERWAY
                        ;
068F 80 0E 0097 R 40            OR      @KB_FLAG_2,KB_PR_LED    ; TURN ON UPDATE IN PROCESS
0694 B0 20                      MOV     AL,EOI                  ; END OF INTERRUPT COMMAND
0696 E6 20                      OUT     020H,AL                 ; SEND COMMAND TO INTERRUPT CONTROL PORT
0698 EB 0D                      JMP     SHORT SL0               ; GO SEND MODE INDICATOR COMMAND
                        ;
069A                    SND_LED1:
069A FA                         CLI                             ; TURN OFF INTERRUPTS
069B F6 06 0097 R 40            TEST    @KB_FLAG_2,KB_PR_LED    ; CHECK FOR MODE INDICATOR UPDATE
06A0 75 34                      JNZ     SL1                     ; DONT UPDATE AGAIN IF UPDATE UNDERWAY
                        ;
06A2 80 0E 0097 R 40            OR      @KB_FLAG_2,KB_PR_LED    ; TURN ON UPDATE IN PROCESS
06A7 B0 ED              SL0:    MOV     AL,LED_CMD              ; LED CMD BYTE
06A9 E8 064B R                  CALL    SND_DATA                ; SEND DATA TO KEYBOARD
06AC FA                         CLI                             ;
06AD E8 06D8 R                  CALL    MAKE_LED                ; GO FORM INDICATOR DATA BYTE
06B0 80 26 0097 R F8            AND     @KB_FLAG_2,0F8H         ; CLEAR MODE INDICATOR BITS
06B5 08 06 0097 R               OR      @KB_FLAG_2,AL           ; SAVE PRESENT INDICATORS FOR NEXT TIME
06B9 F6 06 0097 R 80            TEST    @KB_FLAG_2,KB_ERR       ; TRANSMIT ERROR DETECTED
06BE 75 0B                      JNZ     SL2                     ;  YES, BYPASS SECOND BYTE TRANSMISSION
                        ;
06C0 E8 064B R                  CALL    SND_DATA                ; SEND DATA TO KEYBOARD
06C3 FA                         CLI                             ; TURN OFF INTERRUPTS
06C4 F6 06 0097 R 80            TEST    @KB_FLAG_2,KB_ERR       ; TRANSMIT ERROR DETECTED
06C9 74 06                      JZ      SL3                     ; IF NOT, DONT SEND AN ENABLE COMMAND
                        ;
06CB B0 F4              SL2:    MOV     AL,KB_ENABLE            ; GET KEYBOARD CSA ENABLE COMMAND
06CD E8 064B R                  CALL    SND_DATA                ; SEND DATA TO KEYBOARD
06D0 FA                         CLI                             ; TURN OFF INTERRUPTS
06D1 80 26 0097 R 3F    SL3:    AND     @KB_FLAG_2,NOT(KB_PR_LED+KB_ERR) ; TURN OFF MODE INDICATOR
                                                                ; UPDATE AND TRANSMIT ERROR FLAG
06D6 FB                 SL1:    STI                             ; ENABLE INTERRUPTS
06D7 C3                         RET                             ; RETURN TO CALLER
06D8                    SND_LED ENDP
                        ;--------------------------------------------------------------------------
                        ;                                                                         -
                        ;       MAKE_LED                                                          -
                        ;                                                                         -
                        ;       THIS ROUTINE FORMS THE DATA BYTE NECESSARY TO TURN ON/OFF          -
                        ;         THE MODE INDICATORS                                             -
                        ;                                                                         -
                        ;--------------------------------------------------------------------------
06D8                    MAKE_LED_PROC   NEAR
06D8 51                         PUSH    CX                      ; SAVE CX
06D9 A0 0017 R                  MOV     AL,@KB_FLAG             ; GET CAPS & NUM LOCK INDICATORS
06DC 24 70                      AND     AL,CAPS_STATE+NUM_STATE+SCROLL_STATE ; ISOLATE INDICATORS
06DE B1 04                      MOV     CL,4                    ; SHIFT COUNT
06E0 D2 C0                      ROL     AL,CL                   ; SHIFT BITS OVER TO TURN ON INDICATORS
06E2 24 07                      AND     AL,07H                  ; MAKE SURE ONLY MODE BITS ON
06E4 59                         POP     CX
06E5 C3                         RET                             ; RETURN TO CALLER
06E6                    MAKE_LED ENDP
                        
06E6                    CODE    ENDS
                                END
```

```
                        PAGE 118,121
                        TITLE PRT ------ 06/10/85  PRINTER ADAPTER BIOS
                        .286C
                        .LIST
0000                    CODE    SEGMENT BYTE PUBLIC

                                PUBLIC  PRINTER_IO_1
                                EXTRN   DDS:NEAR

                        ;--- INT  17 H ----------------------------------------------------------------
                        ; PRINTER_IO                                                                  :
                        ;       THIS ROUTINE PROVIDES COMMUNICATION WITH THE PRINTER                  :
                        ; INPUT                                                                       :
                        ;       (AH)= 00H  PRINT THE CHARACTER IN (AL)                                :
                        ;                  ON RETURN, (AH)= 1 IF CHARACTER NOT BE PRINTED (TIME OUT)  :
                        ;                  OTHER BIT5 SET AS ON NORMAL STATUS CALL                    :
                        ;       (AH)= 01H  INITIALIZE THE PRINTER PORT                                :
                        ;                  RETURNS WITH (AH) SET WITH PRINTER STATUS                  :
                        ;       (AH)= 02H  READ THE PRINTER STATUS INTO (AH)                          :
                        ;                  7       6       5       4       3       2-1     0          :
                        ;                  |       |       |       |       |       |       |_ TIME OUT:
                        ;                  |       |       |       |       |       |_ UNUSED          :
                        ;                  |       |       |       |       |_ 1 = I/O ERROR           :
                        ;                  |       |       |       |_ 1 = SELECTED                    :
                        ;                  |       |       |_ 1 = OUT OF PAPER                        :
                        ;                  |       |_ 1 = ACKNOWLEDGE                                 :
                        ;                  |_ 1 = NOT BUSY                                          :
                        ;       (DX) = PRINTER TO BE USED (0,1,2) CORRESPONDING TO ACTUAL VALUES      :
                        ;              IN @PRINTER_BASE AREA                                          :
                        ; DATA AREA @PRINTER_BASE CONTAINS THE BASE ADDRESS OF THE PRINTER CARD(S)    :
                        ; AVAILABLE (LOCATED AT BEGINNING OF DATA SEGMENT, 408H ABSOLUTE, 3 WORDS)    :
                        ;                                                                             :
                        ; DATA AREA @PRINT_TIM_OUT (BYTE) MAY BE CHANGE TO CAUSE DIFFERENT            :
                        ; TIME OUT WAITS. DEFAULT=20 * 4                                              :
                        ;                                                                             :
                        ; REGISTERS     (AH) IS MODIFIED WITH STATUS INFORMATION                      :
                        ;               ALL OTHERS UNCHANGED                                          :
                        ;-------------------------------------------------------------------------------
                                ASSUME  CS:CODE,DS:DATA

0000                    PRINTER_IO_1    PROC    FAR             ; ENTRY POINT FOR ORG 0EFD2H
0000 FB                         STI                             ; INTERRUPTS BACK ON
0001 1E                         PUSH    DS                      ; SAVE SEGMENT
0002 56                         PUSH    SI
0003 52                         PUSH    DX
0004 51                         PUSH    CX
0005 53                         PUSH    BX
0006 E8 0000 E                  CALL    DDS                     ; ADDRESS DATA SEGMENT
0009 8B F2                      MOV     SI,DX                   ; GET PRINTER PARAMETER
000B C1 EA 02                   SHR     DX,2                    ; TEST PARAMETER
000E 75 1A                      JNZ     B10                     ; RETURN IF NOT IN RANGE
0010 8A 9C 0078 R               MOV     BL,@PRINT_TIM_OUT[SI]   ; LOAD TIMEOUT VALUE
0014 D1 E6                      SHL     SI,1                    ; WORD OFFSET INTO TABLE INTO (SI)
0016 8B 94 0008 R               MOV     DX,@PRINTER_BASE[SI]    ; GET BASE ADDRESS FOR PRINTER CARD
001A 0B D2                      OR      DX,DX                   ; TEST DX = ZERO, INDICATING NO PRINTER
001C 74 0C                      JZ      B10                     ; EXIT,  NO PRINTER ADAPTER AT OFFSET
001E 0A E4                      OR      AH,AH                   ; TEST FOR (AH)= 00H
0020 74 0E                      JZ      B20                     ;  PRINT CHARACTER IN (AL)
0022 FE CC                      DEC     AH                      ; TEST FOR (AH)= 01H
0024 74 58                      JZ      B50                     ;  INITIALIZE PRINTER
0026 FE CC                      DEC     AH                      ; TEST FOR (AH)= 02H
0028 74 3F                      JZ      B50                     ;  GET PRINTER STATUS
002A                    B10:
002A 5B                         POP     BX                      ; RETURN
002B 59                         POP     CX
002C 5A                         POP     DX
002D 5E                         POP     SI                      ; RECOVER REGISTERS
002E 1F                         POP     DS
002F CF                         IRET                            ; RETURN TO CALLING PROGRAM

                        ;-----  PRINT THE CHARACTER IN (AL)

0030 50                 B20:    PUSH    AX                      ; SAVE VALUE TO PRINT
0031 EE                         OUT     DX,AL                   ; OUTPUT CHARACTER TO DATA PORT
0032 42                         INC     DX                      ; POINT TO STATUS PORT

                        ;-----  CHECK FOR PRINTER BUSY

0033 53                         PUSH    BX                      ; SAVE TIMEOUT BASE COUNT
0034 EC                         IN      AL,DX                   ; GET STATUS PORT VALUE
0035 A8 80                      TEST    AL,80H                  ; IS THE PRINTER CURRENTLY BUSY
0037 75 05                      JNZ     B25                     ; SKIP SYSTEM DEVICE BUSY CALL IF NOT

                        ;-----  INT 15 H -- DEVICE BUSY

0039 B8 90FE                    MOV     AX,90FEH                ; FUNCTION 90 PRINTER ID
003C CD 15                      INT     15H                     ; SYSTEM CALL

                        ;-----  WAIT BUSY

                                                                ; ADJUST OUTER LOOP COUNT
003E 2A FF              B25:    SUB     BH,BH                   ; CLEAR (BH)
0040 C1 D3 02                   RCL     BX,2                    ; MULTIPLY BY 4
0043                    B30:
0043 2B C9                      SUB     CX,CX                   ; INNER LOOP (64K)
0045                    B35:
0045 EC                         IN      AL,DX                   ; GET STATUS
0046 8A E0                      MOV     AH,AL                   ; STATUS TO (AH) ALSO
0048 A8 80                      TEST    AL,80H                  ; IS THE PRINTER CURRENTLY BUSY
004A 75 0E                      JNZ     B40                     ; GO TO OUTPUT STROBE
004C E2 F7                      LOOP    B35                     ; LOOP IF NOT
004E 4B                         DEC     BX                      ; DECREMENT OUTER LOOP COUNT
004F 75 F2                      JNZ     B30                     ; MAKE ANOTHER PASS IF NOT ZERO

0051 5B                         POP     BX                      ; CLEAR (BX) FROM STACK
0052 80 CC 01                   OR      AH,1                    ; SET ERROR FLAG
0055 80 E4 F9                   AND     AH,0F9H                 ; TURN OFF THE UNUSED BITS
0058 EB 1C                      JMP     SHORT B70               ; RETURN WITH ERROR FLAG SET
```

```
005A                                    B40:
005A 5B                                         POP     BX                      ;       SEND STROBE PULSE
005B B0 0D                                      MOV     AL,0DH                  ; RESTORE (BX) WITH TIMEOUT COUNT
005D 42                                         INC     DX                      ; SET THE STROBE LOW (BIT ON)
005E FA                                         CLI                             ; OUTPUT STROBE TO CONTROL PORT
005F EE                                         OUT     DX,AL                   ; PREVENT INTERRUPT PULSE STRETCHING
0060 EB 00                                      JMP     $+2                     ; OUTPUT STROBE BIT   > 1us  < 5us
0062 EB 00                                      JMP     $+2                     ; I/O DELAY TO ALLOW FOR LINE LOADING
0064 B0 0C                                      MOV     AL,0CH                  ;  AND FOR CORRECT PULSE WIDTH
0066 EE                                         OUT     DX,AL                   ; SET THE -STROBE HIGH
0067 FB                                         STI                             ; INTERRUPTS BACK ON
0068 58                                         POP     AX                      ; RECOVER THE OUTPUT CHAR
                                        ;-----  PRINTER STATUS
0069                                    B50:
0069 50                                         PUSH    AX                      ; SAVE (AL) REGISTER
006A                                    B60:
006A 8B 94 0008 R                               MOV     DX,@PRINTER_BASE[SI]    ; GET PRINTER ATTACHMENT BASE ADDRESS
006E 42                                         INC     DX                      ; POINT TO CONTROL PORT
006F EC                                         IN      AL,DX                   ; PRE-CHARGE +BUSY LINE IF FLOATING
0070 EC                                         IN      AL,DX                   ; GET PRINTER STATUS HARDWARE BITS
0071 8A E0                                      MOV     AH,AL                   ; SAVE
0073 80 E4 F8                                   AND     AH,0F8H                 ; TURN OFF UNUSED BITS
0076                                    B70:
0076 5A                                         POP     DX                      ; RECOVER (AL) REGISTER
0077 8A C2                                      MOV     AL,DL                   ; MOVE CHARACTER INTO (AL)
0079 80 F4 48                                   XOR     AH,48H                  ; FLIP A COUPLE OF BITS
007C EB AC                                      JMP     B10                     ; RETURN FROM ROUTINE WITH STATUS IN AH
                                        ;-----  INITIALIZE THE PRINTER PORT
007E                                    B80:
007E 50                                         PUSH    AX                      ; SAVE (AL)
007F 42                                         INC     DX                      ; POINT TO OUTPUT PORT
0080 42                                         INC     DX
0081 B0 08                                      MOV     AL,8                    ; SET INIT LINE LOW
0083 EE                                         OUT     DX,AL
0084 B8 0FA0                                    MOV     AX,1000*4               ; ADJUST FOR INITIALIZATION DELAY LOOP
0087                                    B90:                                    ; INIT LOOP
0087 48                                         DEC     AX                      ; LOOP FOR RESET TO TAKE
0088 75 FD                                      JNZ     B90                     ; INIT LOOP
008A B0 0C                                      MOV     AL,0CH                  ; NO INTERRUPTS, NON AUTO LF, INIT HIGH
008C EE                                         OUT     DX,AL
008D EB DB                                      JMP     B60                     ; EXIT THROUGH STATUS ROUTINE
                                        PRINTER_IO_1    ENDP
008F                                    CODE    ENDS
                                                END
```

```
                        PAGE 118,121
                        TITLE RS232 ---- 06/10/85  COMMUNICATIONS BIOS (RS232)
                        .286C
                        .LIST
0000                    CODE    SEGMENT BYTE PUBLIC
                                PUBLIC  RS232_IO_1
                                EXTRN   A1:NEAR
                                EXTRN   DDS:NEAR


                        ;--- INT 14 H ----------------------------------------------------------------
                        ;RS232_IO                                                                     :
                        ;       THIS ROUTINE PROVIDES BYTE STREAM I/O TO THE COMMUNICATIONS            :
                        ;       PORT ACCORDING TO THE PARAMETERS:                                      :
                        ;                                                                              :
                        ;       (AH)= 00H  INITIALIZE THE COMMUNICATIONS PORT                          :
                        ;                  (AL) HAS PARAMETERS FOR INITIALIZATION                      :
                        ;                                                                              :
                        ;                  7      6      5      4      3      2      1      0          :
                        ;                  ----- BAUD RATE --      -PARITY--     STOPBIT   --WORD LENGTH--:
                        ;                                                                              :
                        ;                  000 - 110              X0 - NONE      0 - 1    10 - 7 BITS  :
                        ;                  001 - 150              01 - ODD       1 - 2    11 - 8 BITS  :
                        ;                  010 - 300              11 - EVEN                           :
                        ;                  011 - 600                                                  :
                        ;                  100 - 1200                                                 :
                        ;                  101 - 2400                                                 :
                        ;                  110 - 4800                                                 :
                        ;                  111 - 9600                                                 :
                        ;                  ON RETURN, CONDITIONS SET AS IN CALL TO COMMO STATUS (AH=03H):
                        ;                                                                              :
                        ;       (AH)= 01H  SEND THE CHARACTER IN (AL) OVER THE COMMO LINE              :
                        ;                  (AL) REGISTER IS PRESERVED                                  :
                        ;                  ON EXIT, BIT 7 OF AH IS SET IF THE ROUTINE WAS UNABLE TO    :
                        ;                       TO TRANSMIT THE BYTE OF DATA OVER THE LINE.            :
                        ;                       IF BIT 7 OF AH IS NOT SET, THE                         :
                        ;                       REMAINDER OF (AH) IS SET AS IN A STATUS REQUEST,       :
                        ;                       REFLECTING THE CURRENT STATUS OF THE LINE.             :
                        ;       (AH)= 02H  RECEIVE A CHARACTER IN (AL) FROM COMMO LINE BEFORE          :
                        ;                       RETURNING TO CALLER                                    :
                        ;                  ON EXIT, (AH) HAS THE CURRENT LINE STATUS, AS SET BY THE    :
                        ;                       THE STATUS ROUTINE, EXCEPT THAT THE ONLY BITS          :
                        ;                       LEFT ON ARE THE ERROR BITS (7,4,3,2,1)                 :
                        ;                       IF (AH) HAS BIT 7 ON (TIME OUT) THE REMAINING          :
                        ;                       BITS ARE NOT PREDICTABLE.                              :
                        ;                       THUS, (AH) IS NON ZERO ONLY WHEN AN ERROR OCCURRED.    :
                        ;       (AH)= 03H  RETURN THE COMMO PORT STATUS IN (AX)                        :
                        ;                   (AH) CONTAINS THE LINE CONTROL STATUS                      :
                        ;                        BIT 7 = TIME OUT                                      :
                        ;                        BIT 6 = TRANSMIT SHIFT REGISTER EMPTY                 :
                        ;                        BIT 5 = TRANSMIT HOLDING REGISTER EMPTY               :
                        ;                        BIT 4 = BREAK DETECT                                  :
                        ;                        BIT 3 = FRAMING ERROR                                 :
                        ;                        BIT 2 = PARITY ERROR                                  :
                        ;                        BIT 1 = OVERRUN ERROR                                 :
                        ;                        BIT 0 = DATA READY                                    :
                        ;                   (AL) CONTAINS THE MODEM STATUS                             :
                        ;                        BIT 7 = RECEIVE LINE SIGNAL DETECT                    :
                        ;                        BIT 6 = RING INDICATOR                                :
                        ;                        BIT 5 = DATA SET READY                                :
                        ;                        BIT 4 = CLEAR TO SEND                                 :
                        ;                        BIT 3 = DELTA RECEIVE LINE SIGNAL DETECT              :
                        ;                        BIT 2 = TRAILING EDGE RING DETECTOR                   :
                        ;                        BIT 1 = DELTA DATA SET READY                          :
                        ;                        BIT 0 = DELTA CLEAR TO SEND                           :
                        ;                                                                              :
                        ;       (DX) = PARAMETER INDICATING WHICH RS232 CARD (0 - 3 ALLOWED)           :
                        ;                                                                              :
                        ; DATA AREA @RS232_BASE CONTAINS THE BASE ADDRESS OF THE 8250 ON THE CARD      :
                        ;       LOCATION 400H CONTAINS UP TO 4 RS232 ADDRESSES POSSIBLE                :
                        ;       DATA AREA LABEL @RS232_TIM_OUT (BYTE) CONTAINS OUTER LOOP COUNT        :
                        ;       VALUE FOR TIMEOUT (DEFAULT=1)                                          :
                        ;OUTPUT                                                                        :
                        ;               AX MODIFIED ACCORDING TO PARAMETERS OF CALL                    :
                        ;               ALL OTHERS UNCHANGED                                           :
                        ;----------------------------------------------------------------------------
                                ASSUME  CS:CODE,DS:DATA
0000                    RS232_IO_1      PROC    FAR

                        ;-----  VECTOR TO APPROPRIATE ROUTINE

0000 FB                         STI                             ; INTERRUPTS BACK ON
0001 1E                         PUSH    DS                      ; SAVE SEGMENT
0002 52                         PUSH    DX
0003 56                         PUSH    SI
0004 57                         PUSH    DI
0005 51                         PUSH    CX
0006 53                         PUSH    BX
0007 8B F2                      MOV     SI,DX                   ; RS232 VALUE TO (SI)
0009 8B FA                      MOV     DI,DX                   ; AND TO (DI) (FOR TIMEOUTS)
000B C1 EA 02                   SHR     DX,2                    ; TEST PARAMETER
000E 75 20                      JNZ     A3                      ; RETURN IF NOT IN RANGE
0010 D1 E6                      SHL     SI,1                    ; WORD OFFSET
0012 E8 0000 E                  CALL    DDS
0015 8B 94 0000 R               MOV     DX,@RS232_BASE[SI]      ; GET BASE ADDRESS
0019 0B D2                      OR      DX,DX                   ; TEST FOR 0 BASE ADDRESS
001B 74 13                      JZ      A3                      ; RETURN
001D 0A E4                      OR      AH,AH                   ; TEST FOR (AH)= 00H
001F 74 16                      JZ      A4                      ; COMMO INITIALIZATION
0021 FE CC                      DEC     AH                      ; TEST FOR (AH)= 01H
0023 74 4B                      JZ      A6                      ; SEND (AL)
0025 FE CC                      DEC     AH                      ; TEST FOR (AH)= 02H
0027 74 70                      JZ      A12                     ; RECEIVE INTO (AL)
0029                    A2:
0029 FE CC                      DEC     AH                      ; TEST FOR (AH)= 03H
002B 75 03                      JNZ     A3
002D E9 00BB R                  JMP     A18                     ; COMMUNICATION STATUS
0030                    A3:                                     ; RETURN FROM RS232
0030 5B                         POP     BX
0031 59                         POP     CX
0032 5F                         POP     DI
0033 5E                         POP     SI
0034 5A                         POP     DX
0035 1F                         POP     DS
0036 CF                         IRET                            ; RETURN TO CALLER, NO ACTION
```

```
                        PAGE
                        ;-----  INITIALIZE THE COMMUNICATIONS PORT

                        A4:
                                MOV     AH,AL                   ; SAVE INITIALIZATION PARAMETERS IN (AH)
                                ADD     DX,3                    ; POINT TO 8250 CONTROL REGISTER
                                MOV     AL,80H
                                OUT     DX,AL                   ; SET DLAB=1

                        ;-----  DETERMINE BAUD RATE DIVISOR

                                MOV     DL,AH                   ; GET PARAMETERS TO (DL)
                                MOV     CL,4
                                ROL     DL,CL
                                AND     DX,0EH                  ; ISOLATE THEM
                                MOV     DI,OFFSET A1            ; BASE OF TABLE
                                ADD     DI,DX                   ; PUT INTO INDEX REGISTER
                                MOV     DX,@RS232_BASE[SI]      ; POINT TO HIGH ORDER OF DIVISOR
                                INC     DX
                                MOV     AL,CS:[DI]+1            ; GET HIGH ORDER OF DIVISOR
                                OUT     DX,AL                   ; SET ms OF DIVISOR TO 0
                                DEC     DX
                                JMP     $+2                     ; I/O DELAY
                                MOV     AL,CS:[DI]              ; GET LOW ORDER OF DIVISOR
                                OUT     DX,AL                   ; SET LOW OF DIVISOR
                                ADD     DX,3
                                MOV     AL,AH                   ; GET PARAMETERS BACK
                                AND     AL,01FH                 ; STRIP OFF THE BAUD BITS
                                OUT     DX,AL                   ; LINE CONTROL TO 8 BITS
                                DEC     DX
                                DEC     DX
                                JMP     $+2                     ; I/O DELAY
                                MOV     AL,0
                                OUT     DX,AL                   ; INTERRUPT ENABLES ALL OFF
                                JMP     SHORT A18               ; COM_STATUS

                        ;-----  SEND CHARACTER IN (AL) OVER COMMO LINE

                        A5:
                                PUSH    AX                      ; SAVE CHAR TO SEND
                                ADD     DX,4                    ; MODEM CONTROL REGISTER
                                MOV     AL,3                    ; DTR AND RTS
                                OUT     DX,AL                   ; DATA TERMINAL READY, REQUEST TO SEND
                                INC     DX                      ; MODEM STATUS REGISTER
                                INC     DX
                                MOV     BH,30H                  ; DATA SET READY & CLEAR TO SEND
                                CALL    WAIT_FOR_STATUS         ; ARE BOTH TRUE
                                JE      A9                      ; YES, READY TO TRANSMIT CHAR
                        A7:
                                POP     CX
                                MOV     AL,CL                   ; RELOAD DATA BYTE
                        A8:
                                OR      AH,80H                  ; INDICATE TIME OUT
                                JMP     A3                      ; RETURN

                        A9:                                     ; CLEAR_TO_SEND
                                DEC     DX                      ; LINE STATUS REGISTER
                        A10:                                    ; WAIT_SEND
                                MOV     BH,20H                  ; IS TRANSMITTER READY
                                CALL    WAIT_FOR_STATUS         ; TEST FOR TRANSMITTER READY
                                JNZ     A7                      ; RETURN WITH TIME OUT SET
                        A11:                                    ; OUT_CHAR
                                SUB     DX,5                    ; DATA PORT
                                POP     CX                      ; RECOVER IN CX TEMPORARILY
                                MOV     AL,CL                   ; MOVE CHAR TO AL FOR OUT, STATUS IN AH
                                OUT     DX,AL                   ; OUTPUT CHARACTER
                                JMP     A3                      ; RETURN

                        ;-----  RECEIVE CHARACTER FROM COMMO LINE

                        A12:
                                ADD     DX,4                    ; MODEM CONTROL REGISTER
                                MOV     AL,1                    ; DATA TERMINAL READY
                                OUT     DX,AL
                                INC     DX                      ; MODEM STATUS REGISTER
                                INC     DX
                        A13:                                    ; WAIT_DSR
                                MOV     BH,20H                  ; DATA SET READY
                                CALL    WAIT_FOR_STATUS         ; TEST FOR DSR
                                JNZ     A8                      ; RETURN WITH ERROR
                        A15:                                    ; WAIT_DSR_END
                                DEC     DX                      ; LINE STATUS REGISTER
                        A16:                                    ; WAIT_RECV
                                MOV     BH,1                    ; RECEIVE BUFFER FULL
                                CALL    WAIT_FOR_STATUS         ; TEST FOR RECEIVE BUFFER FULL
                                JNZ     A8                      ; SET TIME OUT ERROR
                        A17:                                    ; GET_CHAR
                                AND     AH,00011110B            ; TEST FOR ERROR CONDITIONS ON RECEIVE

                                MOV     DX,@RS232_BASE[SI]      ; DATA PORT
                                IN      AL,DX                   ; GET CHARACTER FROM LINE
                                JMP     A3                      ; RETURN

                        ;-----  COMMO PORT STATUS ROUTINE

                        A18:
                                MOV     DX,@RS232_BASE[SI]
                                ADD     DX,5                    ; CONTROL PORT
                                IN      AL,DX                   ; GET LINE CONTROL STATUS
                                MOV     AH,AL                   ; PUT IN (AH) FOR RETURN
                                INC     DX                      ; POINT TO MODEM STATUS REGISTER
                                IN      AL,DX                   ; GET MODEM CONTROL STATUS
                                JMP     A3                      ; RETURN
```

```
                        PAGE
                        ;--------------------------------------
                        ;       WAIT FOR STATUS ROUTINE        :
                        ;ENTRY: (BH)= STATUS BIT(S) TO LOOK FOR :
                        ;       (DX)= ADDRESS OF STATUS REG    :
                        ;EXIT:  ZERO FLAG ON = STATUS FOUND    :
                        ;       ZERO FLAG OFF = TIMEOUT.       :
                        ;       (AH)= LAST STATUS READ         :
                        ;--------------------------------------

00CA                    WAIT_FOR_STATUS PROC    NEAR
00CA 8A 9D 007C R               MOV     BL,@RS232_TIM_OUT[DI]   ; LOAD OUTER LOOP COUNT

                        ;-----  ADJUST OUTER LOOP COUNT

00CE 55                         PUSH    BP                      ; SAVE (BP)
00CF 53                         PUSH    BX                      ; SAVE (BX)
00D0 5D                         POP     BP                      ; USE BP FOR OUTER LOOP COUNT
00D1 81 E5 00FF                 AND     BP,00FFH                ; STRIP HIGH BITS
00D5 D1 D5                      RCL     BP,1                    ; MULTIPLY OUTER COUNT BY 4
00D7 D1 D5                      RCL     BP,1
00D9                    WFS0:
00D9 2B C9                      SUB     CX,CX
00DB                    WFS1:
00DB EC                         IN      AL,DX                   ; GET STATUS
00DC 8A E0                      MOV     AH,AL                   ; MOVE TO (AH)
00DE 22 C7                      AND     AL,BH                   ; ISOLATE BITS TO TEST
00E0 3A C7                      CMP     AL,BH                   ; EXACTLY = TO MASK
00E2 74 07                      JE      WFS_END                 ; RETURN WITH ZERO FLAG ON

00E4 E2 F5                      LOOP    WFS1                    ; TRY AGAIN

00E6 4D                         DEC     BP
00E7 75 F0                      JNZ     WFS0

00E9 0A FF                      OR      BH,BH                   ; SET ZERO FLAG OFF
00EB                    WFS_END:
00EB 5D                         POP     BP                      ; RESTORE (BP)
00EC C3                         RET

00ED                    WAIT_FOR_STATUS ENDP

00ED                    RS232_IO_1      ENDP

00ED                    CODE    ENDS
                        END
```

```
                        PAGE 118,121
                        TITLE VIDEO1 --- 03/24/86  VIDEO DISPLAY BIOS
                        .LIST
0000                    CODE    SEGMENT BYTE PUBLIC

                                PUBLIC  ACT_DISP_PAGE
                                PUBLIC  READ_AC_CURRENT
                                PUBLIC  READ_CURSOR
                                PUBLIC  READ_DOT
                                PUBLIC  READ_LPEN
                                PUBLIC  SCROLL_DOWN
                                PUBLIC  SCROLL_UP
                                PUBLIC  SET_COLOR
                                PUBLIC  SET_CPOS
                                PUBLIC  SET_CTYPE
                                PUBLIC  SET_MODE
                                PUBLIC  WRITE_AC_CURRENT
                                PUBLIC  WRITE_C_CURRENT
                                PUBLIC  WRITE_DOT
                                PUBLIC  WRITE_TTY
                                PUBLIC  VIDEO_IO_1
                                PUBLIC  VIDEO_STATE

                                EXTRN   BEEP:NEAR               ; SPEAKER BEEP ROUTINE
                                EXTRN   CRT_CHAR_GEN:NEAR       ; CHARACTER GENERATOR GRAPHICS TABLE
                                EXTRN   DDS:NEAR                ; LOAD (DS) WITH DATA SEGMENT SELECTOR
                                EXTRN   M5:WORD                 ; REGEN BUFFER LENGTH TABLE
                                EXTRN   M6:BYTE                 ; COLUMNS PER MODE TABLE
                                EXTRN   M7:BYTE                 ; MODE SET VALUE PER MODE TABLE

                        ;--- INT 10 H ----------------------------------------------------------------
                        ;                                                                           :
                        ; VIDEO_IO                                                                  :
                        ;       THESE ROUTINES PROVIDE THE CRT DISPLAY INTERFACE                    :
                        ;       THE FOLLOWING FUNCTIONS ARE PROVIDED:                               :
                        ;                                                                           :
                        ;     (AH)= 00H  SET MODE (AL) CONTAINS MODE VALUE                          :
                        ;                  (AL) = 00H  40X25 BW MODE (POWER ON DEFAULT)             :
                        ;                  (AL) = 01H  40X25 COLOR                                  :
                        ;                  (AL) = 02H  80X25 BW                                     :
                        ;                  (AL) = 03H  80X25 COLOR                                  :
                        ;                              GRAPHICS MODES                               :
                        ;                  (AL) = 04H  320X200 COLOR                                :
                        ;                  (AL) = 05H  320X200 BW MODE                              :
                        ;                  (AL) = 06H  640X200 BW MODE                              :
                        ;                  (AL) = 07H  80X25 MONOCHROME (USED INTERNAL TO VIDEO ONLY) :
                        ;                  *** NOTES -BW MODES OPERATE SAME AS COLOR MODES, BUT COLOR :
                        ;                             BURST IS NOT ENABLED                          :
                        ;                            -CURSOR IS NOT DISPLAYED IN GRAPHICS MODE      :
                        ;     (AH)= 01H  SET CURSOR TYPE                                            :
                        ;                  (CH) =  BITS 4-0 = START LINE FOR CURSOR                 :
                        ;                          ** HARDWARE WILL ALWAYS CAUSE BLINK              :
                        ;                          ** SETTING BIT 5 OR 6 WILL CAUSE ERRATIC BLINKING :
                        ;                             OR NO CURSOR AT ALL                           :
                        ;                  (CL) =  BITS 4-0 = END LINE FOR CURSOR                   :
                        ;     (AH)= 02H  SET CURSOR POSITION                                        :
                        ;                  (DH,DL) = ROW,COLUMN  (00H,00H) IS UPPER LEFT            :
                        ;                  (BH) = PAGE NUMBER (MUST BE 00H FOR GRAPHICS MODES)      :
                        ;     (AH)= 03H  READ CURSOR POSITION                                       :
                        ;                  (BH) = PAGE NUMBER (MUST BE 00H FOR GRAPHICS MODES)      :
                        ;                  ON EXIT (DH,DL) = ROW,COLUMN OF CURRENT CURSOR           :
                        ;                          (CH,CL) = CURSOR MODE CURRENTLY SET              :
                        ;     (AH)= 04H  READ LIGHT PEN POSITION                                    :
                        ;                  ON EXIT:                                                 :
                        ;                   (AH) = 00H -- LIGHT PEN SWITCH NOT DOWN/NOT TRIGGERED   :
                        ;                   (AH) = 01H -- VALID LIGHT PEN VALUE IN REGISTERS        :
                        ;                          (DH,DL) = ROW,COLUMN OF CHARACTER LP POSITION    :
                        ;                          (CH) = RASTER LINE (0-199)                       :
                        ;                          (BX) = PIXEL COLUMN (0-319,639)                  :
                        ;     (AH)= 05H  SELECT ACTIVE DISPLAY PAGE (VALID ONLY FOR ALPHA MODES)    :
                        ;                  (AL) = NEW PAGE VALUE (0-7 FOR MODES 0&1, 0-3 FOR MODES 2&3) :
                        ;     (AH)= 06H  SCROLL ACTIVE PAGE UP                                      :
                        ;                  (AL) = NUMBER OF LINES, ( LINES BLANKED AT BOTTOM OF WINDOW ) :
                        ;                         (AL) = 00H MEANS BLANK ENTIRE WINDOW              :
                        ;                  (CH,CL) = ROW,COLUMN OF UPPER LEFT CORNER OF SCROLL      :
                        ;                  (DH,DL) = ROW,COLUMN OF LOWER RIGHT CORNER OF SCROLL     :
                        ;                  (BH) = ATTRIBUTE TO BE USED ON BLANK LINE                :
                        ;     (AH)= 07H  SCROLL ACTIVE PAGE DOWN                                    :
                        ;                  (AL) = NUMBER OF LINES, INPUT LINES BLANKED AT TOP OF WINDOW :
                        ;                         (AL) = 00H MEANS BLANK ENTIRE WINDOW              :
                        ;                  (CH,CL) = ROW,COLUMN OF UPPER LEFT CORNER OF SCROLL      :
                        ;                  (DH,DL) = ROW,COLUMN OF LOWER RIGHT CORNER OF SCROLL     :
                        ;                  (BH) = ATTRIBUTE TO BE USED ON BLANK LINE                :
                        ;                                                                           :
                        ;   CHARACTER HANDLING ROUTINES                                             :
                        ;                                                                           :
                        ;     (AH)= 08H  READ ATTRIBUTE/CHARACTER AT CURRENT CURSOR POSITION        :
                        ;                  (BH) = DISPLAY PAGE (VALID FOR ALPHA MODES ONLY)         :
                        ;                  ON EXIT:                                                 :
                        ;                  (AL) = CHAR READ                                         :
                        ;                  (AH) = ATTRIBUTE OF CHARACTER READ (ALPHA MODES ONLY)    :
                        ;     (AH)= 09H  WRITE ATTRIBUTE/CHARACTER AT CURRENT CURSOR POSITION       :
                        ;                  (BH) = DISPLAY PAGE (VALID FOR ALPHA MODES ONLY)         :
                        ;                  (CX) = COUNT OF CHARACTERS TO WRITE                      :
                        ;                  (AL) = CHAR TO WRITE                                     :
                        ;                  (BL) = ATTRIBUTE OF CHARACTER (ALPHA)/COLOR OF CHAR (GRAPHICS) :
                        ;                             SEE NOTE ON WRITE DOT FOR BIT 7 OF BL = 1.    :
                        ;     (AH)= 0AH  WRITE CHARACTER ONLY AT CURRENT CURSOR POSITION            :
                        ;                  (BH) = DISPLAY PAGE (VALID FOR ALPHA MODES ONLY)         :
                        ;                  (CX) = COUNT OF CHARACTERS TO WRITE                      :
                        ;                  (AL) = CHAR TO WRITE                                     :
                        ;                         NOTE: USE FUNCTION (AH)= 09H IN GRAPHICS MODES    :
                        ;       FOR READ/WRITE CHARACTER INTERFACE WHILE IN GRAPHICS MODE, THE       :
                        ;               CHARACTERS ARE FORMED FROM A CHARACTER GENERATOR IMAGE       :
                        ;               MAINTAINED IN THE SYSTEM ROM.  ONLY THE 1ST 128 CHARS       :
                        ;               ARE CONTAINED THERE.  TO READ/WRITE THE SECOND 128 CHARS,   :
                        ;               THE USER MUST INITIALIZE THE POINTER AT INTERRUPT 1FH       :
                        ;               (LOCATION 0007CH) TO POINT TO THE 1K BYTE TABLE CONTAINING  :
                        ;               THE CODE POINTS FOR THE SECOND 128 CHARS (128-255).         :
                        ;       FOR WRITE CHARACTER INTERFACE IN GRAPHICS MODE, THE REPLICATION FACTOR :
                        ;               CONTAINED IN (CX) ON ENTRY WILL PRODUCE VALID RESULTS ONLY  :
                        ;               FOR CHARACTERS CONTAINED ON THE SAME ROW.  CONTINUATION TO   :
                        ;               SUCCEEDING LINES WILL NOT PRODUCE CORRECTLY.                 :
                        ;                                                                           :
```

```
                        ;    GRAPHICS INTERFACE                                                          :
                        ;                                                                                :
                        ;    (AH)= 0BH  SET COLOR PALETTE                                                :
                        ;               (BH) = PALETTE COLOR ID BEING SET (0-127)                        :
                        ;               (BL) = COLOR VALUE TO BE USED WITH THAT COLOR ID                 :
                        ;                      NOTE: FOR THE CURRENT COLOR CARD, THIS ENTRY POINT HAS     :
                        ;                            MEANING ONLY FOR 320X200 GRAPHICS.                  :
                        ;                      COLOR ID = 0 SELECTS THE BACKGROUND COLOR (0-15)           :
                        ;                      COLOR ID = 1 SELECTS THE PALETTE TO BE USED:               :
                        ;                                 0 = GREEN(1)/RED(2)/YELLOW(3)                  :
                        ;                                 1 = CYAN(1)/MAGENTA(2)/WHITE(3)                :
                        ;                      IN 40X25 OR 80X25 ALPHA MODES, THE VALUE SET FOR          :
                        ;                            PALETTE COLOR 0 INDICATES THE BORDER COLOR          :
                        ;                            TO BE USED (VALUES 0-31, WHERE 16-31 SELECT         :
                        ;                            THE HIGH INTENSITY BACKGROUND SET.                  :
                        ;    (AH)= 0CH  WRITE DOT                                                        :
                        ;               (DX) = ROW NUMBER                                                :
                        ;               (CX) = COLUMN NUMBER                                             :
                        ;               (AL) = COLOR VALUE                                               :
                        ;                      IF BIT 7 OF AL = 1, THEN THE COLOR VALUE IS EXCLUSIVE     :
                        ;                      ORed WITH THE CURRENT CONTENTS OF THE DOT                  :
                        ;    (AH)= 0DH  READ DOT                                                         :
                        ;               (DX) = ROW NUMBER                                                :
                        ;               (CX) = COLUMN NUMBER                                             :
                        ;               (AL) RETURNS THE DOT READ                                        :
                        ;                                                                                :
                        ;    ASCII TELETYPE ROUTINE FOR OUTPUT                                           :
                        ;                                                                                :
                        ;    (AH)= 0EH  WRITE TELETYPE TO ACTIVE PAGE                                    :
                        ;               (AL) = CHAR TO WRITE                                             :
                        ;               (BL) = FOREGROUND COLOR IN GRAPHICS MODE                         :
                        ;               NOTE -- SCREEN WIDTH IS CONTROLLED BY PREVIOUS MODE SET          :
                        ;    (AH)= 0FH  CURRENT VIDEO STATE                                              :
                        ;               RETURNS THE CURRENT VIDEO STATE                                  :
                        ;               (AL) = MODE CURRENTLY SET ( SEE (AH)= 00H FOR EXPLANATION)       :
                        ;               (AH) = NUMBER OF CHARACTER COLUMNS ON SCREEN                     :
                        ;               (BH) = CURRENT ACTIVE DISPLAY PAGE                               :
                        ;    (AH)= 10H  RESERVED                                                         :
                        ;    (AH)= 11H  RESERVED                                                         :
                        ;    (AH)= 12H  RESERVED                                                         :
                        ;    (AH)= 13H  WRITE STRING                                                     :
                        ;                          ES:BP  -  POINTER TO STRING TO BE WRITTEN             :
                        ;                          CX     -  LENGTH OF CHARACTER STRING TO WRITTEN       :
                        ;                          DX     -  CURSOR POSITION FOR STRING TO BE WRITTEN    :
                        ;                          BH     -  PAGE NUMBER                                 :
                        ;               (AL)= 00H  WRITE CHARACTER STRING                                :
                        ;                          BL     -  ATTRIBUTE                                   :
                        ;                          STRING IS  <CHAR,CHAR, ... ,CHAR>                     :
                        ;                          CURSOR NOT MOVED                                      :
                        ;               (AL)= 01H  WRITE CHARACTER STRING AND MOVE CURSOR                :
                        ;                          BL     -  ATTRIBUTE                                   :
                        ;                          STRING IS  <CHAR,CHAR, ... ,CHAR>                     :
                        ;                          CURSOR IS MOVED                                       :
                        ;               (AL)= 02H  WRITE CHARACTER AND ATTRIBUTE STRING                  :
                        ;                                (VALID FOR ALPHA MODES ONLY)                    :
                        ;                          STRING IS  <CHAR,ATTR,CHAR,ATTR ..  ,CHAR,ATTR>       :
                        ;                          CURSOR IS NOT MOVED                                   :
                        ;               (AL)= 03H  WRITE CHARACTER AND ATTRIBUTE STRING AND MOVE CURSOR  :
                        ;                                (VALID FOR ALPHA MODES ONLY)                    :
                        ;                          STRING IS  <CHAR,ATTR,CHAR,ATTR ..  ,CHAR,ATTR>       :
                        ;                          CURSOR IS MOVED                                       :
                        ;                 NOTE:  CARRIAGE RETURN, LINE FEED, BACKSPACE, AND BELL ARE     :
                        ;                        TREATED AS COMMANDS RATHER THAN PRINTABLE CHARACTERS.   :
                        ;                                                                                :
                        ;       BX,CX,DX,SI,DI,BP,SP,DS,ES,SS PRESERVED DURING CALLS EXCEPT FOR          :
                        ;       BX,CX,DX RETURN VALUES ON FUNCTIONS 03H,04H,0DH AND 0DH. ON ALL CALLS    :
                        ;       AX IS MODIFIED.                                                          :
                        ;--------------------------------------------------------------------------------

                                ASSUME  CS:CODE,DS:DATA,ES:NOTHING

0000 005F R             MI      DW      OFFSET  SET_MODE          ;  TABLE OF ROUTINES WITHIN VIDEO I/O
0002 0146 R                     DW      OFFSET  SET_CTYPE
0004 016B R                     DW      OFFSET  SET_CPOS
0006 0193 R                     DW      OFFSET  READ_CURSOR
0008 0789 R                     DW      OFFSET  READ_LPEN
000A 01AA R                     DW      OFFSET  ACT_DISP_PAGE
000C 0213 R                     DW      OFFSET  SCROLL_UP
000E 02B1 R                     DW      OFFSET  SCROLL_DOWN
0010 0303 R                     DW      OFFSET  READ_AC_CURRENT
0012 0360 R                     DW      OFFSET  WRITE_AC_CURRENT
0014 0392 R                     DW      OFFSET  WRITE_C_CURRENT
0016 01CC R                     DW      OFFSET  SET_COLOR
0018 0454 R                     DW      OFFSET  WRITE_DOT
001A 0443 R                     DW      OFFSET  READ_DOT
001C 0702 R                     DW      OFFSET  WRITE_TTY
001E 01F2 R                     DW      OFFSET  VIDEO_STATE
0020 013D R                     DW      OFFSET  VIDEO_RETURN    ; RESERVED
0022 013D R                     DW      OFFSET  VIDEO_RETURN    ; RESERVED
0024 013D R                     DW      OFFSET  VIDEO_RETURN    ; RESERVED
0026 03BF R                     DW      OFFSET  WRITE_STRING    ; CASE 13H, WRITE STRING
= 0028                  MIL     EQU     $-MI
                        ;
0028                    VIDEO_IO_1      PROC    NEAR                    ;       ENTRY POINT FOR ORG 0F065H
0028 FB                         STI                                     ; INTERRUPTS BACK ON
0029 FC                         CLD                                     ; SET DIRECTION FORWARD
002A 80 FC 14                   CMP     AH,MIL/2                        ; TEST FOR WITHIN TABLE RANGE
002D 73 2F                      JNB     M4                              ; BRANCH TO EXIT IF NOT A VALID COMMAND
                        ;
002F 06                         PUSH    ES
0030 1E                         PUSH    DS                              ; SAVE WORK AND PARAMETER REGISTERS
0031 52                         PUSH    DX
0032 51                         PUSH    CX
0033 53                         PUSH    BX
0034 56                         PUSH    SI
0035 57                         PUSH    DI
0036 55                         PUSH    BP
0037 BE ---- R                  MOV     SI,DATA                         ; POINT DS: TO DATA SEGMENT
003A 8E DE                      MOV     DS,SI
003C 8B F0                      MOV     SI,AX                           ; SAVE COMMAND/DATA INTO (SI) REGISTER
003E A0 0010 R                  MOV     AL,BYTE PTR @EQUIP_FLAG         ; GET THE EQUIPMENT FLAG VIDEO BITS
0041 24 30                      AND     AL,30H                          ; ISOLATE CRT SWITCHES
0043 3C 30                      CMP     AL,30H                          ; IS SETTING FOR MONOCHROME CARD?
0045 BF B800                    MOV     DI,0B800H                       ; GET SEGMENT FOR COLOR CARD
```

```
                        JNE     M2                      ; SKIP IF NOT MONOCHROME CARD
                        MOV     DI,0B000H               ; ELSE GET SEGMENT FOR MONOCHROME CARD
                M2:
                        MOV     ES,DI                   ; SET UP TO POINT AT VIDEO MEMORY AREAS
                        MOV     AL,AH                   ; PLACE COMMAND IN LOW BYTE OF (AX)
                        CBW                             ; AND FORM BYTE OFFSET WITH COMMAND
                        SAL     AX,1                    ; TIMES 2 FOR WORD TABLE LOOKUP
                        XCHG    SI,AX                   ; MOVE OFFSET INTO LOOK UP REGISTER (SI)
                                                        ;  AND RESTORE COMMAND/DATA INTO (AX)
                        MOV     AH,@CRT_MODE            ; MOVE CURRENT MODE INTO (AH) REGISTER

                        JMP     WORD PTR CS:[SI+OFFSET M1]      ; GO TO SELECTED FUNCTION

                M4:                                     ;       COMMAND NOT VALID
                        IRET                            ; DO NOTHING IF NOT IN VALID RANGE
                VIDEO_IO_1      ENDP
                ;--------------------------------------------------------
                ; SET_MODE                                              :
                ;       THIS ROUTINE INITIALIZES THE ATTACHMENT TO      :
                ;       THE SELECTED MODE.  THE SCREEN IS BLANKED.      :
                ; INPUT                                                 :
                ;       @EQUIP_FLAG BITS 5-4 = MODE/WIDTH               :
                ;           11 = MONOCHROME (FORCES MODE 7)             :
                ;           01 = COLOR ADAPTER 40x25 (MODE 0 DEFAULT)   :
                ;           10 = COLOR ADAPTER 80x25 (MODE 2 DEFAULT)   :
                ;         (AL) = COLOR MODE REQUESTED ( RANGE  0 - 6 )  :
                ; OUTPUT                                                :
                ;       NONE                                            :
                ;--------------------------------------------------------
                SET_MODE        PROC    NEAR
                        MOV     DX,03D4H                ; ADDRESS OF COLOR CARD
                        MOV     DI,@EQUIP_FLAG          ; GET EQUIPMENT FLAGS SETTING
                        AND     DI,30H                  ; ISOLATE CRT SWITCHES
                        CMP     DI,30H                  ; IS BW CARD INSTALLED AS PRIMARY
                        JNE     M8C                     ; SKIP AND CHECK IF COLOR
                        MOV     AL,7                    ; ELSE INDICATE INTERNAL BW CARD MODE
                        MOV     DL,0B4H                 ; SET ADDRESS OF BW (MONOCHROME) CARD
                        JMP     SHORT M8                ; CONTINUE WITH FORCED MODE 7
                M8C:
                        CMP     AL,7                    ; CHECK FOR VALID COLOR MODES 0-6
                        JB      M8                      ; CONTINUE IF BELOW MODE 7
                        MOV     AL,0                    ; FORCE DEFAULT 40x25 BW MODE
                        CMP     DI,10H                  ; CHECK FOR @EQUIP_FLAG AT 40x25 COLOR
                        JE      M8                      ; CONTINUE WITH MODE 0 IF NOT
                        MOV     AL,2                    ; ELSE FORCE MODE 2
                M8:
                        MOV     @CRT_MODE,AL            ; SAVE MODE IN GLOBAL VARIABLE
                        MOV     @ADDR_6845,DX           ; SAVE ADDRESS OF BASE
                        MOV     @ROWS,25-1              ; INITIALIZE DEFAULT ROW COUNT OF 25
                        PUSH    DS                      ; SAVE POINTER TO DATA SEGMENT
                        PUSH    AX                      ; SAVE MODE NUMBER (AL)
                        CBW                             ; CLEAR HIGH BYTE OF MODE
                        MOV     SI,AX                   ; SET TABLE POINTER, INDEXED BY MODE
                        MOV     AL,CS:[SI + OFFSET M7]  ; GET THE MODE SET VALUE FROM TABLE
                        MOV     @CRT_MODE_SET,AL        ; SAVE THE MODE SET VALUE
                        AND     AL,037H                 ; VIDEO OFF, SAVE HIGH RESOLUTION BIT
                        PUSH    DX                      ; SAVE OUTPUT PORT VALUE
                        ADD     DX,4                    ; POINT TO CONTROL REGISTER
                        OUT     DX,AL                   ; RESET VIDEO TO OFF TO SUPPRESS ROLLING
                        POP     DX                      ; BACK TO BASE REGISTER

                        ASSUME  DS:ABS0
                        SUB     BX,BX                   ; SET UP FOR ABS0 SEGMENT
                        MOV     DS,BX                   ; ESTABLISH VECTOR TABLE ADDRESSING
                        LDS     BX,@PARM_PTR            ; GET POINTER TO VIDEO PARMS
                        ASSUME  DS:CODE
                        POP     AX                      ; RECOVER MODE NUMBER IN (AL)
                        MOV     CX,16                   ; LENGTH OF EACH ROW OF TABLE
                        CMP     AL,2                    ; DETERMINE WHICH ONE TO USE
                        JC      M9                      ; MODE IS 0 OR 1
                        ADD     BX,CX                   ; NEXT ROW OF INITIALIZATION TABLE
                        CMP     AL,4
                        JC      M9                      ; MODE IS 2 OR 3
                        ADD     BX,CX                   ; MOVE TO GRAPHICS ROW OF INIT_TABLE
                        CMP     AL,7
                        JC      M9                      ; MODE IS 4,5, OR 6
                        ADD     BX,CX                   ; MOVE TO BW CARD ROW OF INIT_TABLE

                ;-----  BX POINTS TO CORRECT ROW OF INITIALIZATION TABLE

                M9:                                     ; OUT_INIT
                        PUSH    AX                      ; SAVE MODE IN (AL)
                        MOV     AX,[BX+10]              ; GET THE CURSOR MODE FROM THE TABLE
                        XCHG    AH,AL                   ; PUT CURSOR MODE IN CORRECT POSITION
                        PUSH    DS                      ; SAVE TABLE SEGMENT POINTER
                        ASSUME  DS:DATA
                        CALL    DDS                     ; POINT DS TO DATA SEGMENT
                        MOV     @CURSOR_MODE,AX         ; PLACE INTO BIOS DATA SAVE AREA
                        ASSUME  DS:CODE
                        POP     DS                      ; RESTORE THE TABLE SEGMENT POINTER
                        XOR     AH,AH                   ; AH IS REGISTER NUMBER DURING LOOP

                ;-----  LOOP THROUGH TABLE, OUTPUTTING REGISTER ADDRESS, THEN VALUE FROM TABLE

                M10:                                    ;  INITIALIZATION LOOP
                        MOV     AL,AH                   ; GET 6845 REGISTER NUMBER
                        OUT     DX,AL
                        INC     DX                      ; POINT TO DATA PORT
                        INC     AH                      ; NEXT REGISTER VALUE
                        MOV     AL,[BX]                 ; GET TABLE VALUE
                        OUT     DX,AL                   ; OUT TO CHIP
                        INC     BX                      ; NEXT IN TABLE
                        DEC     DX                      ; BACK TO POINTER REGISTER
                        LOOP    M10                     ; DO THE WHOLE TABLE
                        POP     AX                      ; GET MODE BACK INTO (AL)
                        POP     DS                      ; RECOVER SEGMENT VALUE
                        ASSUME  DS:DATA

                ;-----  FILL REGEN AREA WITH BLANK

                        XOR     DI,DI                   ; SET UP POINTER FOR REGEN
                        MOV     @CRT_START,DI           ; START ADDRESS SAVED IN GLOBAL
                        MOV     @ACTIVE_PAGE,0          ; SET PAGE VALUE
                        MOV     CX,8192                 ; NUMBER OF WORDS IN COLOR CARD
                        CMP     AL,4                    ; TEST FOR GRAPHICS
```

```
00F0 72 0A                      JC      M12                     ; NO GRAPHICS_INIT
00F2 3C 07                      CMP     AL,7                    ; TEST FOR BW CARD
00F4 74 04                      JE      M11                     ; BW CARD_INIT
00F6 33 C0                      XOR     AX,AX                   ; FILL FOR GRAPHICS MODE
00F8 EB 05                      JMP     SHORT M13               ; CLEAR_BUFFER
00FA                    M11:                                    ; BW CARD_INIT
00FA B5 08                      MOV     CH,08H                  ; BUFFER SIZE ON BW CARD (2048)
00FC                    M12:                                    ; NO_GRAPHICS_INIT
00FC B8 0720                    MOV     AX,' '+7*H              ; FILL CHAR FOR ALPHA + ATTRIBUTE
00FF                    M13:                                    ; CLEAR_BUFFER
00FF F3/ AB                     REP     STOSW                   ; FILL THE REGEN BUFFER WITH BLANKS

                        ;-----  ENABLE VIDEO AND CORRECT PORT SETTING

0101 8B 16 0063 R               MOV     DX,@ADDR_6845           ; PREPARE TO OUTPUT TO VIDEO ENABLE PORT
0105 83 C2 04                   ADD     DX,4                    ; POINT TO THE MODE CONTROL REGISTER
0108 A0 0065 R                  MOV     AL,@CRT_MODE_SET        ; GET THE MODE SET VALUE
010B EE                         OUT     DX,AL                   ; SET VIDEO ENABLE PORT

                        ;-----  DETERMINE NUMBER OF COLUMNS, BOTH FOR ENTIRE DISPLAY
                        ;-----  AND THE NUMBER TO BE USED FOR TTY INTERFACE

010C 2E: 8A 84 0000 E           MOV     AL,CS:[SI + OFFSET M6]  ; GET NUMBER OF COLUMNS ON THIS SCREEN
0111 98                         CBW                             ; CLEAR HIGH BYTE
0112 A3 004A R                  MOV     @CRT_COLS,AX            ; INITIALIZE NUMBER OF COLUMNS COUNT

                        ;-----  SET CURSOR POSITIONS

0115 81 E6 000E                 AND     SI,000EH                ; WORD OFFSET INTO CLEAR LENGTH TABLE
0119 2E: 8B 84 0000 E           MOV     AX,CS:[SI + OFFSET M5]  ; LENGTH TO CLEAR
011E A3 004C R                  MOV     @CRT_LEN,AX             ; SAVE LENGTH OF CRT -- NOT USED FOR BW
0121 B9 0008                    MOV     CX,8                    ; CLEAR ALL CURSOR POSITIONS
0124 BF 0050 R                  MOV     DI,OFFSET @CURSOR_POSN
0127 1E                         PUSH    DS                      ; ESTABLISH SEGMENT
0128 07                         POP     ES                      ;   ADDRESSING
0129 33 C0                      XOR     AX,AX
012B F3/ AB                     REP     STOSW                   ; FILL WITH ZEROES

                        ;-----  SET UP OVERSCAN REGISTER

012D 42                         INC     DX                      ; SET OVERSCAN PORT TO A DEFAULT
012E B0 30                      MOV     AL,30H                  ; 30H VALUE FOR ALL MODES EXCEPT 640X200
0130 80 3E 0049 R 06            CMP     @CRT_MODE,6             ; SEE IF THE MODE IS 640X200 BW
0135 75 02                      JNZ     M14                     ; IF NOT 640X200, THEN GO TO REGULAR
0137 B0 3F                      MOV     AL,3FH                  ; IF IT IS 640X200, THEN PUT IN 3FH
0139                    M14:
0139 EE                         OUT     DX,AL                   ; OUTPUT THE CORRECT VALUE TO 3D9 PORT
013A A2 0066 R                  MOV     @CRT_PALETTE,AL         ; SAVE THE VALUE FOR FUTURE USE

                        ;-----  NORMAL RETURN FROM ALL VIDEO RETURNS

013D                    VIDEO_RETURN:
013D 5D                         POP     BP
013E 5F                         POP     DI
013F 5E                         POP     SI
0140 5B                         POP     BX
0141                    M15:                                    ; VIDEO_RETURN_C
0141 59                         POP     CX
0142 5A                         POP     DX
0143 1F                         POP     DS
0144 07                         POP     ES                      ; RECOVER SEGMENTS
0145 CF                         IRET                            ; ALL DONE
0146                    SET_MODE        ENDP
                        ;--------------------------------------------------
                        ; SET_CTYPE
                        ;       THIS ROUTINE SETS THE CURSOR VALUE
                        ; INPUT
                        ;       (CX) HAS CURSOR VALUE CH-START LINE, CL-STOP LINE
                        ; OUTPUT
                        ;       NONE
                        ;--------------------------------------------------
0146                    SET_CTYPE       PROC    NEAR
0146 B4 0A                      MOV     AH,10                   ; 6845 REGISTER FOR CURSOR SET
0148 89 0E 0060 R               MOV     @CURSOR_MODE,CX         ; SAVE IN DATA AREA
014C E8 0151 R                  CALL    M16                     ; OUTPUT CX REGISTER
014F EB EC                      JMP     VIDEO_RETURN

                        ;-----  THIS ROUTINE OUTPUTS THE CX REGISTER TO THE 6845 REGISTERS NAMED IN (AH)

0151                    M16:
0151 8B 16 0063 R               MOV     DX,@ADDR_6845           ; ADDRESS REGISTER
0155 8A C4                      MOV     AL,AH                   ; GET VALUE
0157 EE                         OUT     DX,AL                   ; REGISTER SET
0158 42                         INC     DX                      ; DATA REGISTER
0159 EB 00                      JMP     $+2                     ; I/O DELAY
015B 8A C5                      MOV     AL,CH                   ; DATA
015D EE                         OUT     DX,AL
015E 4A                         DEC     DX
015F 8A C4                      MOV     AL,AH
0161 FE C0                      INC     AL                      ; POINT TO OTHER DATA REGISTER
0163 EE                         OUT     DX,AL                   ; SET FOR SECOND REGISTER
0164 42                         INC     DX
0165 EB 00                      JMP     $+2                     ; I/O DELAY
0167 8A C1                      MOV     AL,CL                   ; SECOND DATA VALUE
0169 EE                         OUT     DX,AL
016A C3                         RET                             ; ALL DONE
016B                    SET_CTYPE       ENDP

                        ;--------------------------------------------
                        ; SET_CPOS
                        ;       THIS ROUTINE SETS THE CURRENT CURSOR POSITION TO THE
                        ;       NEW X-Y VALUES PASSED
                        ; INPUT
                        ;       DX - ROW,COLUMN OF NEW CURSOR
                        ;       BH - DISPLAY PAGE OF CURSOR
                        ; OUTPUT
                        ;       CURSOR IS SET AT 6845 IF DISPLAY PAGE IS CURRENT DISPLAY
                        ;--------------------------------------------
016B                    SET_CPOS        PROC    NEAR
016B 8A C7                      MOV     AL,BH                   ; MOVE PAGE NUMBER TO WORK REGISTER
016D 98                         CBW                             ; CONVERT PAGE TO WORD VALUE
016E D1 E0                      SAL     AX,1                    ; WORD OFFSET
0170 96                         XCHG    AX,SI                   ; USE INDEX REGISTER
```

```
0171 89 94 0050 R               MOV     [SI+OFFSET @CURSOR_POSN],DX     ; SAVE THE POINTER
0175 38 3E 0062 R               CMP     @ACTIVE_PAGE,BH
0179 75 05                      JNZ     M17                     ; SET_CPOS_RETURN
017B 8B C2                      MOV     AX,DX                   ; GET ROW/COLUMN TO AX
017D E8 0182 R                  CALL    M18                     ; CURSOR_SET
0180                    M17:                                    ; SET_CPOS_RETURN
0180 EB B8                      JMP     VIDEO_RETURN
0182                    SET_CPOS        ENDP

                        ;-----  SET CURSOR POSITION, AX HAS ROW/COLUMN FOR CURSOR

0182                    M18     PROC    NEAR
0182 E8 0204 R                  CALL    POSITION                ; DETERMINE LOCATION IN REGEN BUFFER
0185 8B C8                      MOV     CX,AX
0187 03 0E 004E R               ADD     CX,@CRT_START           ; ADD IN THE START ADDRESS FOR THIS PAGE
018B D1 F9                      SAR     CX,1                    ; DIVIDE BY 2 FOR CHAR ONLY COUNT
018D B4 0E                      MOV     AH,14                   ; REGISTER NUMBER FOR CURSOR
018F E8 0151 R                  CALL    M16                     ; OUTPUT THE VALUE TO THE 6845
0192 C3                         RET
0193                    M18     ENDP
                        ;--------------------------------------------------------
                        ; READ_CURSOR
                        ;       THIS ROUTINE READS THE CURRENT CURSOR VALUE FROM THE
                        ;       6845, FORMATS IT, AND SENDS IT BACK TO THE CALLER
                        ; INPUT
                        ;       BH - PAGE OF CURSOR
                        ; OUTPUT
                        ;       DX - ROW, COLUMN OF THE CURRENT CURSOR POSITION
                        ;       CX - CURRENT CURSOR MODE
                        ;--------------------------------------------
0193                    READ_CURSOR     PROC    NEAR
0193 8A DF                      MOV     BL,BH
0195 32 FF                      XOR     BH,BH
0197 D1 E3                      SAL     BX,1                    ; WORD OFFSET
0199 8B 97 0050 R               MOV     DX,[BX+OFFSET @CURSOR_POSN]
019D 8B 0E 0060 R               MOV     CX,@CURSOR_MODE
01A1 5D                         POP     BP
01A2 5F                         POP     DI
01A3 5E                         POP     SI
01A4 5B                         POP     BX
01A5 58                         POP     AX                      ; DISCARD SAVED CX AND DX
01A6 58                         POP     AX
01A7 1F                         POP     DS
01A8 07                         POP     ES
01A9 CF                         IRET
01AA                    READ_CURSOR     ENDP
                        ;--------------------------------------------------
                        ; ACT_DISP_PAGE
                        ;       THIS ROUTINE SETS THE ACTIVE DISPLAY PAGE, ALLOWING
                        ;       THE FULL USE OF THE MEMORY SET ASIDE FOR THE VIDEO ATTACHMENT
                        ; INPUT
                        ;       AL HAS THE NEW ACTIVE DISPLAY PAGE
                        ; OUTPUT
                        ;       THE 6845 IS RESET TO DISPLAY THAT PAGE
                        ;--------------------------------------------
01AA                    ACT_DISP_PAGE   PROC    NEAR
01AA A2 0062 R                  MOV     @ACTIVE_PAGE,AL         ; SAVE ACTIVE PAGE VALUE
01AD 98                         CBW                             ; CONVERT (AL) TO WORD
01AE 50                         PUSH    AX                      ; SAVE PAGE VALUE
01AF F7 26 004C R               MUL     WORD PTR @CRT_LEN       ; DISPLAY PAGE TIMES REGEN LENGTH
01B3 A3 004E R                  MOV     @CRT_START,AX           ; SAVE START ADDRESS FOR LATER
01B6 8B C8                      MOV     CX,AX                   ; START ADDRESS TO CX
01B8 D1 F9                      SAR     CX,1                    ; DIVIDE BY 2 FOR 6845 HANDLING
01BA B4 0C                      MOV     AH,12                   ; 6845 REGISTER FOR START ADDRESS
01BC E8 0151 R                  CALL    M16
01BF 5B                         POP     BX                      ; RECOVER PAGE VALUE
01C0 D1 E3                      SAL     BX,1                    ; *2 FOR WORD OFFSET
01C2 8B 87 0050 R               MOV     AX,[BX + OFFSET @CURSOR_POSN]   ; GET CURSOR FOR THIS PAGE
01C6 E8 0182 R                  CALL    M18                     ; SET THE CURSOR POSITION
01C9 E9 013D R                  JMP     VIDEO_RETURN
01CC                    ACT_DISP_PAGE   ENDP
                        ;--------------------------------------------
                        ; SET_COLOR
                        ;       THIS ROUTINE WILL ESTABLISH THE BACKGROUND COLOR, THE OVERSCAN COLOR,
                        ;       AND THE FOREGROUND COLOR SET FOR MEDIUM RESOLUTION GRAPHICS
                        ; INPUT
                        ;       (BH) HAS COLOR ID
                        ;               IF BH=0, THE BACKGROUND COLOR VALUE IS SET
                        ;                       FROM THE LOW BITS OF BL (0-31)
                        ;               IF BH=1, THE PALETTE SELECTION IS MADE
                        ;                       BASED ON THE LOW BIT OF BL:
                        ;                               0 = GREEN, RED, YELLOW FOR COLORS 1,2,3
                        ;                               1 = BLUE, CYAN, MAGENTA FOR COLORS 1,2,3
                        ;       (BL) HAS THE COLOR VALUE TO BE USED
                        ; OUTPUT
                        ;       THE COLOR SELECTION IS UPDATED
                        ;--------------------------------------------
01CC                    SET_COLOR       PROC    NEAR
01CC 8B 16 0063 R               MOV     DX,@ADDR_6845           ; I/O PORT FOR PALETTE
01D0 83 C2 05                   ADD     DX,5                    ; OVERSCAN PORT
01D3 A0 0066 R                  MOV     AL,@CRT_PALETTE         ; GET THE CURRENT PALETTE VALUE
01D6 0A FF                      OR      BH,BH                   ; IS THIS COLOR 0?
01D8 75 0E                      JNZ     M20                     ; OUTPUT COLOR 1

                        ;-----  HANDLE COLOR 0 BY SETTING THE BACKGROUND COLOR

01DA 24 E0                      AND     AL,0E0H                 ; TURN OFF LOW 5 BITS OF CURRENT
01DC 80 E3 1F                   AND     BL,01FH                 ; TURN OFF HIGH 3 BITS OF INPUT VALUE
01DF 0A C3                      OR      AL,BL                   ; PUT VALUE INTO REGISTER
01E1                    M19:                                    ; OUTPUT THE PALETTE
01E1 EE                         OUT     DX,AL                   ; OUTPUT COLOR SELECTION TO 3D9 PORT
01E2 A2 0066 R                  MOV     @CRT_PALETTE,AL         ; SAVE THE COLOR VALUE
01E5 E9 013D R                  JMP     VIDEO_RETURN

                        ;-----  HANDLE COLOR 1 BY SELECTING THE PALETTE TO BE USED

01E8                    M20:
01E8 24 DF                      AND     AL,0DFH                 ; TURN OFF PALETTE SELECT BIT
01EA D0 EB                      SHR     BL,1                    ; TEST THE LOW ORDER BIT OF BL
01EC 73 F3                      JNC     M19                     ; ALREADY DONE
01EE 0C 20                      OR      AL,20H                  ; TURN ON PALETTE SELECT BIT
01F0 EB EF                      JMP     M19                     ; GO DO IT
01F2                    SET_COLOR       ENDP
```

```
                        ;--------------------------------------------------
                        ; VIDEO STATE                                     :
                        ;  RETURNS THE CURRENT VIDEO STATE IN AX          :
                        ;  AH = NUMBER OF COLUMNS ON THE SCREEN           :
                        ;  AL = CURRENT VIDEO MODE                        :
                        ;  BH = CURRENT ACTIVE PAGE                       :
                        ;--------------------------------------------------
01F2                    VIDEO_STATE     PROC    NEAR
01F2 8A 26 004A R               MOV     AH,BYTE PTR @CRT_COLS   ; GET NUMBER OF COLUMNS
01F6 A0 0049 R                  MOV     AL,@CRT_MODE            ; CURRENT MODE
01F9 8A 3E 0062 R               MOV     BH,@ACTIVE_PAGE         ; GET CURRENT ACTIVE PAGE
01FD 5D                         POP     BP                      ; RECOVER REGISTERS
01FE 5F                         POP     DI
01FF 5E                         POP     SI
0200 59                         POP     CX                      ; DISCARD SAVED BX
0201 E9 0141 R                  JMP     M15                     ; RETURN TO CALLER
0204                    VIDEO_STATE     ENDP
                        ;--------------------------------------------
                        ; POSITION                                  :
                        ;       THIS SERVICE ROUTINE CALCULATES THE REGEN BUFFER ADDRESS
                        ;       OF A CHARACTER IN THE ALPHA MODE
                        ; INPUT                                     :
                        ;       AX = ROW, COLUMN POSITION
                        ; OUTPUT                                    :
                        ;       AX = OFFSET OF CHAR POSITION IN REGEN BUFFER
                        ;--------------------------------------------
0204                    POSITION        PROC    NEAR
0204 53                         PUSH    BX                      ; SAVE REGISTER
0205 93                         XCHG    BX,AX                   ; SAVE ROW/COLUMN POSITION IN (BX)
0206 A0 004A R                  MOV     AL,BYTE PTR @CRT_COLS   ; GET COLUMNS PER ROW COUNT
0209 F6 E7                      MUL     BH                      ; DETERMINE BYTES TO ROW
020B 32 FF                      XOR     BH,BH
020D 03 C3                      ADD     AX,BX                   ; ADD IN COLUMN VALUE
020F D1 E0                      SAL     AX,1                    ; * 2 FOR ATTRIBUTE BYTES
0211 5B                         POP     BX
0212 C3                         RET
0213                    POSITION        ENDP
                        ;--------------------------------------------
                        ; SCROLL UP                                 :
                        ;       THIS ROUTINE MOVES A BLOCK OF CHARACTERS UP
                        ;       ON THE SCREEN
                        ; INPUT                                     :
                        ;       (AH) = CURRENT CRT MODE
                        ;       (AL) = NUMBER OF ROWS TO SCROLL
                        ;       (CX) = ROW/COLUMN OF UPPER LEFT CORNER
                        ;       (DX) = ROW/COLUMN OF LOWER RIGHT CORNER
                        ;       (BH) = ATTRIBUTE TO BE USED ON BLANKED LINE
                        ;       (DS) = DATA SEGMENT
                        ;       (ES) = REGEN BUFFER SEGMENT
                        ; OUTPUT                                    :
                        ;       NONE -- THE REGEN BUFFER IS MODIFIED
                        ;--------------------------------------------
                                ASSUME  DS:DATA,ES:DATA
0213                    SCROLL_UP       PROC    NEAR

0213 E8 02EE R                  CALL    TEST_LINE_COUNT
0216 80 FC 04                   CMP     AH,4                    ; TEST FOR GRAPHICS MODE
0219 72 08                      JC      N1                      ; HANDLE SEPARATELY
021B 80 FC 07                   CMP     AH,7                    ; TEST FOR BW CARD
021E 74 03                      JE      N1
0220 E9 04B0 R                  JMP     GRAPHICS_UP
0223                    N1:                                     ; UP_CONTINUE
0223 53                         PUSH    BX                      ; SAVE FILL ATTRIBUTE IN BH
0224 8B C1                      MOV     AX,CX                   ; UPPER LEFT POSITION
0226 E8 0260 R                  CALL    SCROLL_POSITION         ; DO SETUP FOR SCROLL
0229 74 31                      JZ      N7                      ; BLANK_FIELD
022B 03 F0                      ADD     SI,AX                   ; FROM ADDRESS
022D 8A E6                      MOV     AH,DH                   ; # ROWS IN BLOCK
022F 2A E3                      SUB     AH,BL                   ; # ROWS TO BE MOVED
0231                    N2:                                     ; ROW_LOOP
0231 E8 02A1 R                  CALL    N10                     ; MOVE ONE ROW
0234 03 F5                      ADD     SI,BP
0236 03 FD                      ADD     DI,BP                   ; POINT TO NEXT LINE IN BLOCK
0238 FE CC                      DEC     AH                      ; COUNT OF LINES TO MOVE
023A 75 F5                      JNZ     N2                      ; ROW_LOOP
023C                    N3:                                     ; CLEAR_ENTRY
023C 58                         POP     AX                      ; RECOVER ATTRIBUTE IN AH
023D B0 20                      MOV     AL,' '                  ; FILL WITH BLANKS
023F                    N4:                                     ; CLEAR_LOOP
023F E8 02AA R                  CALL    N11                     ; CLEAR THE ROW
0242 03 FD                      ADD     DI,BP                   ; POINT TO NEXT LINE
0244 FE CB                      DEC     BL                      ; COUNTER OF LINES TO SCROLL
0246 75 F7                      JNZ     N4                      ; CLEAR_LOOP
0248                    N5:                                     ; SCROLL_END
0248 E8 0000 E                  CALL    DDS
024B 80 3E 0049 R 07            CMP     @CRT_MODE,7             ; IS THIS THE BLACK AND WHITE CARD
0250 74 07                      JE      N6                      ; IF SO, SKIP THE MODE RESET
0252 A0 0065 R                  MOV     AL,@CRT_MODE_SET        ; GET THE VALUE OF THE MODE SET
0255 BA 03D8                    MOV     DX,03D8H                ; ALWAYS SET COLOR CARD PORT
0258 EE                         OUT     DX,AL
0259                    N6:                                     ; VIDEO_RET_HERE
0259 E9 013D R                  JMP     VIDEO_RETURN
025C                    N7:                                     ; BLANK_FIELD
025C 8A DE                      MOV     BL,DH                   ; GET ROW COUNT
025E EB DC                      JMP     N3                      ; GO CLEAR THAT AREA
0260                    SCROLL_UP       ENDP

                        ;-----  HANDLE COMMON SCROLL SET UP HERE

0260                    SCROLL_POSITION PROC    NEAR
0260 E8 0204 R                  CALL    POSITION                ; CONVERT TO REGEN POINTER
0263 03 06 004E R               ADD     AX,@CRT_START           ; OFFSET OF ACTIVE PAGE
0267 8B F8                      MOV     DI,AX                   ; TO ADDRESS FOR SCROLL
0269 8B F0                      MOV     SI,AX                   ; FROM ADDRESS FOR SCROLL
026B 2B D1                      SUB     DX,CX                   ; DX = #ROWS, #COLS IN BLOCK
026D FE C6                      INC     DH
026F FE C2                      INC     DL                      ; INCREMENT FOR 0 ORIGIN
0271 32 ED                      XOR     CH,CH                   ; SET HIGH BYTE OF COUNT TO ZERO
0273 8B 2E 004A R               MOV     BP,@CRT_COLS            ; GET NUMBER OF COLUMNS IN DISPLAY
0277 03 ED                      ADD     BP,BP                   ; TIMES 2 FOR ATTRIBUTE BYTE
0279 A0 004A R                  MOV     AL,BYTE PTR @CRT_COLS   ; GET CHARACTERS PER LINE COUNT
027C F6 E3                      MUL     BL                      ; DETERMINE OFFSET TO FROM ADDRESS
027E 03 C0                      ADD     AX,AX                   ; *2 FOR ATTRIBUTE BYTE
0280 50                         PUSH    AX                      ; SAVE LINE COUNT
```

```
0281 A0 0049 R                  MOV     AL,@CRT_MODE            ; GET CURRENT MODE
0284 06                         PUSH    ES                      ; ESTABLISH ADDRESSING TO REGEN BUFFER
0285 1F                         POP     DS                      ;  FOR BOTH POINTERS
0286 3C 02                      CMP     AL,2                    ; TEST FOR COLOR CARD SPECIAL CASES HERE
0288 72 13                      JB      N9                      ; HAVE TO HANDLE 80X25 SEPARATELY
028A 3C 03                      CMP     AL,3
028C 77 0F                      JA      N9
                        ;-----                                  ; 80X25 COLOR CARD SCROLL
028E 52                         PUSH    DX
028F BA 03DA                    MOV     DX,3DAH                 ; GUARANTEED TO BE COLOR CARD HERE
                        N8:                                     ; WAIT_DISP_ENABLE
0292 EC                         IN      AL,DX                   ; GET PORT
0293 A8 08                      TEST    AL,RVRT                 ; WAIT FOR VERTICAL RETRACE
0295 74 FB                      JZ      N8                      ; WAIT_DISP_ENABLE
0297 B0 25                      MOV     AL,25H
0299 B2 D8                      MOV     DL,0D8H                 ; ADDRESS CONTROL PORT
029B EE                         OUT     DX,AL                   ; TURN OFF VIDEO DURING VERTICAL RETRACE
029C 5A                         POP     DX
                        N9:
029D 58                         POP     AX                      ; RESTORE LINE COUNT
029E 0A DB                      OR      BL,BL                   ; 0 SCROLL MEANS BLANK FIELD
02A0 C3                         RET                             ; RETURN WITH FLAGS SET
                        SCROLL_POSITION ENDP
02A1
                        ;-----  MOVE_ROW
02A1                    N10     PROC    NEAR
02A1 8A CA                      MOV     CL,DL                   ; GET # OF COLS TO MOVE
02A3 56                         PUSH    SI
02A4 57                         PUSH    DI                      ; SAVE START ADDRESS
02A5 F3/ A5                     REP     MOVSW                   ; MOVE THAT LINE ON SCREEN
02A7 5F                         POP     DI
02A8 5E                         POP     SI                      ; RECOVER ADDRESSES
02A9 C3                         RET
02AA                    N10     ENDP

                        ;-----  CLEAR_ROW
02AA                    N11     PROC    NEAR
02AA 8A CA                      MOV     CL,DL                   ; GET # COLUMNS TO CLEAR
02AC 57                         PUSH    DI
02AD F3/ AB                     REP     STOSW                   ; STORE THE FILL CHARACTER
02AF 5F                         POP     DI
02B0 C3                         RET
02B1                    N11     ENDP
                        ;--------------------------------------
                        ; SCROLL_DOWN
                        ;       THIS ROUTINE MOVES THE CHARACTERS WITHIN A DEFINED
                        ;       BLOCK DOWN ON THE SCREEN, FILLING THE TOP LINES
                        ;       WITH A DEFINED CHARACTER
                        ; INPUT
                        ;       (AH) = CURRENT CRT MODE
                        ;       (AL) = NUMBER OF LINES TO SCROLL
                        ;       (CX) = UPPER LEFT CORNER OF REGION
                        ;       (DX) = LOWER RIGHT CORNER OF REGION
                        ;       (BH) = FILL CHARACTER
                        ;       (DS) = DATA SEGMENT
                        ;       (ES) = REGEN SEGMENT
                        ; OUTPUT
                        ;       NONE -- SCREEN IS SCROLLED
                        ;--------------------------------------
02B1                    SCROLL_DOWN     PROC    NEAR
02B1 FD                         STD                             ; DIRECTION FOR SCROLL DOWN
02B2 E8 02EE R                  CALL    TEST_LINE_COUNT         ;
02B5 80 FC 04                   CMP     AH,4                    ; TEST FOR GRAPHICS
02B8 72 08                      JC      N12
02BA 80 FC 07                   CMP     AH,7                    ; TEST FOR BW CARD
02BD 74 03                      JE      N12
02BF E9 0507 R                  JMP     GRAPHICS_DOWN
02C2                    N12:                                    ; CONTINUE_DOWN
02C2 53                         PUSH    BX                      ; SAVE ATTRIBUTE IN BH
02C3 8B C2                      MOV     AX,DX                   ; LOWER RIGHT CORNER
02C5 E8 0260 R                  CALL    SCROLL_POSITION         ; GET REGEN LOCATION
02C8 74 20                      JZ      N16
02CA 2B F0                      SUB     SI,AX                   ; SI IS FROM ADDRESS
02CC 8A E6                      MOV     AH,DH                   ; GET TOTAL # ROWS
02CE 2A E3                      SUB     AH,BL                   ; COUNT TO MOVE IN SCROLL
02D0                    N13:
02D0 E8 02A1 R                  CALL    N10                     ; MOVE ONE ROW
02D3 2B F5                      SUB     SI,BP
02D5 2B FD                      SUB     DI,BP
02D7 FE CC                      DEC     AH
02D9 75 F5                      JNZ     N13
02DB                    N14:
02DB 58                         POP     AX                      ; RECOVER ATTRIBUTE IN AH
02DC B0 20                      MOV     AL,' '
02DE                    N15:
02DE E8 02AA R                  CALL    N11                     ; CLEAR ONE ROW
02E1 2B FD                      SUB     DI,BP                   ; GO TO NEXT ROW
02E3 FE CB                      DEC     BL
02E5 75 F7                      JNZ     N15
02E7 E9 0248 R                  JMP     N6                      ; SCROLL_END
02EA                    N16:
02EA 8A DE                      MOV     BL,DH
02EC EB ED                      JMP     N14
02EE                    SCROLL_DOWN     ENDP

                        ;-----  IF  AMOUNT OF LINES TO BE SCROLLED = AMOUNT OF LINES IN WINDOW
                        ;-----   THEN  ADJUST AL; ELSE  RETURN;

02EE                    TEST_LINE_COUNT PROC    NEAR
02EE 8A D8                      MOV     BL,AL                   ; SAVE LINE COUNT IN BL
02F0 0A C0                      OR      AL,AL                   ; TEST IF AL IS ALREADY ZERO
02F2 74 0E                      JZ      BL_SET                  ; IF IT IS THEN RETURN...
02F4 50                         PUSH    AX                      ; SAVE AX
02F5 8A C6                      MOV     AL,DH                   ; SUBTRACT LOWER ROW FROM UPPER ROW
02F7 2A C5                      SUB     AL,CH
02F9 FE C0                      INC     AL                      ; ADJUST DIFFERENCE BY 1
02FB 3A C3                      CMP     AL,BL                   ; LINE COUNT = AMOUNT OF ROWS IN WINDOW?
02FD 58                         POP     AX                      ; RESTORE AX
02FE 75 02                      JNE     BL_SET                  ; IF NOT THEN WE'RE ALL SET
0300 2A DB                      SUB     BL,BL                   ; OTHERWISE SET BL TO ZERO
0302                    BL_SET:
0302 C3                         RET                             ; RETURN
0303                    TEST_LINE_COUNT ENDP
```

```
                        PAGE
                        ;--------------------------------------------------------------------------------
                        ; READ_AC_CURRENT                                                              :
                        ;       THIS ROUTINE READS THE ATTRIBUTE AND CHARACTER AT THE CURRENT          :
                        ;       CURSOR POSITION AND RETURNS THEM TO THE CALLER                        :
                        ; INPUT                                                                       :
                        ;       (AH) = CURRENT CRT MODE                                               :
                        ;       (BH) = DISPLAY PAGE ( ALPHA MODES ONLY )                              :
                        ;       (DS) = DATA SEGMENT                                                   :
                        ;       (ES) = REGEN SEGMENT                                                  :
                        ; OUTPUT                                                                      :
                        ;       (AL) = CHARACTER READ                                                 :
                        ;       (AH) = ATTRIBUTE READ                                                 :
                        ;--------------------------------------------------------------------------------
                                ASSUME  DS:DATA,ES:DATA

0303                    READ_AC_CURRENT PROC    NEAR
0303 80 FC 04                   CMP     AH,4                    ; IS THIS GRAPHICS
0306 72 08                      JC      P10

0308 80 FC 07                   CMP     AH,7                    ; IS THIS BW CARD
030B 74 03                      JE      P10

030D E9 0642 R                  JMP     GRAPHICS_READ
0310                    P10:                                    ;  READ_AC_CONTINUE
0310 E8 032C R                  CALL    FIND_POSITION           ; GET REGEN LOCATION AND PORT ADDRESS
0313 8B F7                      MOV     SI,DI                   ; ESTABLISH ADDRESSING IN SI
0315 06                         PUSH    ES                      ; GET REGEN SEGMENT FOR QUICK ACCESS
0316 1F                         POP     DS

                        ;-----  WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80

0317 0A DB                      OR      BL,BL                   ; CHECK MODE FLAG FOR COLOR CARD IN 80
0319 75 0D                      JNZ     P13                     ; ELSE SKIP RETRACE WAIT - DO FAST READ
031B                    P11:                                    ;  WAIT FOR HORZ RETRACE LOW OR VERTICAL
031B FB                         STI                             ; ENABLE INTERRUPTS FIRST
031C 90                         NOP                             ; ALLOW FOR SMALL INTERRUPT WINDOW
031D FA                         CLI                             ; BLOCK INTERRUPTS FOR SINGLE LOOP
031E EC                         IN      AL,DX                   ; GET STATUS FROM THE ADAPTER
031F A8 01                      TEST    AL,RHRZ                 ; IS HORIZONTAL RETRACE LOW
0321 75 F8                      JNZ     P11                     ; WAIT UNTIL IT IS
0323                    P12:                                    ;  NOW WAIT FOR EITHER RETRACE HIGH
0323 EC                         IN      AL,DX                   ; GET STATUS
0324 A8 09                      TEST    AL,RVRT+RHRZ            ; IS HORIZONTAL OR VERTICAL RETRACE HIGH
0326 74 FB                      JZ      P12                     ; WAIT UNTIL EITHER IS ACTIVE
0328                    P13:
0328 AD                         LODSW                           ; GET THE CHARACTER AND ATTRIBUTE
0329 E9 013D R                  JMP     VIDEO_RETURN            ;  EXIT WITH (AX)

032C                    READ_AC_CURRENT ENDP


032C                    FIND_POSITION   PROC    NEAR            ;       SETUP FOR BUFFER READ OR WRITE
032C 86 E3                      XCHG    AH,BL                   ; SWAP MODE TYPE WITH ATTRIBUTE
032E 8B E8                      MOV     BP,AX                   ; SAVE CHARACTER/ATTR IN (BP) REGISTER
0330 80 EB 02                   SUB     BL,2                    ; CONVERT DISPLAY MODE TYPE TO A
0333 D0 EB                      SHR     BL,1                    ;  ZERO VALUE FOR COLOR IN 80 COLUMN
0335 8A C7                      MOV     AL,BH                   ; MOVE DISPLAY PAGE TO LOW BYTE
0337 98                         CBW                             ; CLEAR HIGH BYTE FOR BYTE OFFSET
0338 8B F8                      MOV     DI,AX                   ; MOVE DISPLAY PAGE (COUNT) TO WORK REG
033A D1 E7                      SAL     DI,1                    ; TIMES 2 FOR WORD OFFSET
033C 8B 95 0050 R               MOV     DX,[DI+OFFSET @CURSOR_POSN]     ; GET ROW/COLUMN OF THAT PAGE
0340 74 09                      JZ      P21                     ; SKIP BUFFER ADJUSTMENT IF PAGE ZERO

0342 33 FF                      XOR     DI,DI                   ; ELSE SET BUFFER START ADDRESS TO ZERO
0344                    P20:
0344 03 3E 004C R               ADD     DI,@CRT_LEN             ; ADD LENGTH OF BUFFER FOR ONE PAGE
0348 48                         DEC     AX                      ; DECREMENT PAGE COUNT
0349 75 F9                      JNZ     P20                     ; LOOP TILL PAGE COUNT EXHAUSTED

034B                    P21:                                    ; DETERMINE LOCATION IN REGEN IN PAGE
034B A0 004A R                  MOV     AL,BYTE PTR @CRT_COLS   ; GET COLUMNS PER ROW COUNT
034E F6 E6                      MUL     DH                      ; DETERMINE BYTES TO ROW
0350 32 F6                      XOR     DH,DH
0352 03 C2                      ADD     AX,DX                   ; ADD IN COLUMN VALUE
0354 D1 E0                      SAL     AX,1                    ; * 2 FOR ATTRIBUTE BYTES
0356 03 F8                      ADD     DI,AX                   ; ADD LOCATION TO START OF REGEN PAGE
0358 8B 16 0063 R               MOV     DX,@ADDR_6845           ; GET BASE ADDRESS OF ACTIVE DISPLAY
035C 83 C2 06                   ADD     DX,6                    ;  DX= STATUS PORT ADDRESS OF ADAPTER
035F C3                         RET                             ;  BP= ATTRIBUTE/CHARACTER (FROM BL/AL)
                                                                ;  DI= POSITION (OFFSET IN REGEN BUFFER)
0360                    FIND_POSITION   ENDP                    ;  BL= MODE FLAG (ZERO FOR 80X25 COLOR)
```

SECTION 5

```
                        PAGE
                        ;--------------------------------------------------------------------------------
                        ; WRITE_AC_CURRENT                                                              :
                        ;       THIS ROUTINE WRITES THE ATTRIBUTE AND CHARACTER                         :
                        ;       AT THE CURRENT CURSOR POSITION                                         :
                        ; INPUT                                                                        :
                        ;       (AH) = CURRENT CRT MODE                                                :
                        ;       (BH) = DISPLAY PAGE                                                    :
                        ;       (CX) = COUNT OF CHARACTERS TO WRITE                                    :
                        ;       (AL) = CHAR TO WRITE                                                   :
                        ;       (BL) = ATTRIBUTE OF CHAR TO WRITE                                      :
                        ;       (DS) = DATA SEGMENT                                                    :
                        ;       (ES) = REGEN SEGMENT                                                   :
                        ; OUTPUT                                                                       :
                        ;       DISPLAY REGEN BUFFER UPDATED                                           :
                        ;--------------------------------------------------------------------------------

0360                    WRITE_AC_CURRENT        PROC    NEAR
0360 80 FC 04                   CMP     AH,4                    ; IS THIS GRAPHICS
0363 72 08                      JC      P30
0365 80 FC 07                   CMP     AH,7                    ; IS THIS BW CARD
0368 74 03                      JE      P30
036A E9 058E R                  JMP     GRAPHICS_WRITE
036D                    P30:                                    ;  WRITE_AC_CONTINUE
036D E8 032C R                  CALL    FIND_POSITION           ; GET REGEN LOCATION AND PORT ADDRESS
                                                                ; ADDRESS IN (DI) REGISTER
0370 0A DB                      OR      BL,BL                   ; CHECK MODE FLAG FOR COLOR CARD AT 80
0372 74 06                      JZ      P32                     ; SKIP TO RETRACE WAIT IF COLOR AT 80

0374 95                         XCHG    AX,BP                   ; GET THE ATTR/CHAR SAVED FOR FAST WRITE
0375 F3/ AB                     REP     STOSW                   ; STRING WRITE THE ATTRIBUTE & CHARACTER
0377 EB 16                      JMP     SHORT  P35              ; EXIT FAST WRITE ROUTINE

                        ;-----  WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80

0379                    P31:                                    ;  LOOP FOR EACH ATTR/CHAR WRITE
0379 95                         XCHG    BP,AX                   ; PLACE ATTR/CHAR BACK IN SAVE REGISTER
037A                    P32:                                    ;  WAIT FOR HORZ RETRACE LOW OR VERTICAL
037A FB                         STI                             ; ENABLE INTERRUPTS FIRST
037B 90                         NOP                             ; ALLOW FOR INTERRUPT WINDOW
037C FA                         CLI                             ; BLOCK INTERRUPTS FOR SINGLE LOOP
037D EC                         IN      AL,DX                   ; GET STATUS FROM THE ADAPTER
037E A8 08                      TEST    AL,RVRT                 ; CHECK FOR VERTICAL RETRACE FIRST
0380 75 09                      JNZ     P34                     ; DO FAST WRITE NOW IF VERTICAL RETRACE
0382 A8 01                      TEST    AL,RHRZ                 ; IS HORIZONTAL RETRACE LOW THEN
0384 75 F4                      JNZ     P32                     ; WAIT UNTIL IT IS
0386                    P33:                                    ;  WAIT FOR EITHER RETRACE HIGH
0386 EC                         IN      AL,DX                   ; GET STATUS AGAIN
0387 A8 09                      TEST    AL,RVRT+RHRZ            ; IS HORIZONTAL OR VERTICAL RETRACE HIGH
0389 74 FB                      JZ      P33                     ; WAIT UNTIL EITHER IS ACTIVE
038B                    P34:
038B 95                         XCHG    AX,BP                   ; GET THE ATTR/CHAR SAVED IN (BP)
038C AB                         STOSW                           ; WRITE THE ATTRIBUTE AND CHARACTER
038D E2 EA                      LOOP    P31                     ; AS MANY TIMES AS REQUESTED - TILL CX=0
038F                    P35:
038F E9 013D R                  JMP     VIDEO_RETURN            ; EXIT

0392                    WRITE_AC_CURRENT        ENDP

                        ;--------------------------------------------------------------------------------
                        ; WRITE_C_CURRENT                                                               :
                        ;       THIS ROUTINE WRITES THE CHARACTER AT                                   :
                        ;       THE CURRENT CURSOR POSITION, ATTRIBUTE UNCHANGED                       :
                        ; INPUT                                                                        :
                        ;       (AH) = CURRENT CRT MODE                                                :
                        ;       (BH) = DISPLAY PAGE                                                    :
                        ;       (CX) = COUNT OF CHARACTERS TO WRITE                                    :
                        ;       (AL) = CHAR TO WRITE                                                   :
                        ;       (DS) = DATA SEGMENT                                                    :
                        ;       (ES) = REGEN SEGMENT                                                   :
                        ; OUTPUT                                                                       :
                        ;       DISPLAY REGEN BUFFER UPDATED                                           :
                        ;--------------------------------------------------------------------------------

0392                    WRITE_C_CURRENT PROC    NEAR
0392 80 FC 04                   CMP     AH,4                    ; IS THIS GRAPHICS
0395 72 08                      JC      P40
0397 80 FC 07                   CMP     AH,7                    ; IS THIS BW CARD
039A 74 03                      JE      P40
039C E9 058E R                  JMP     GRAPHICS_WRITE
039F                    P40:
039F E8 032C R                  CALL    FIND_POSITION           ; GET REGEN LOCATION AND PORT ADDRESS
                                                                ; ADDRESS OF LOCATION IN (DI)

                        ;-----  WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80

03A2                    P41:                                    ;  WAIT FOR HORZ RETRACE LOW OR VERTICAL
03A2 FB                         STI                             ; ENABLE INTERRUPTS FIRST
03A3 0A DB                      OR      BL,BL                   ; CHECK MODE FLAG FOR COLOR CARD IN 80
03A5 75 0F                      JNZ     P43                     ; ELSE SKIP RETRACE WAIT - DO FAST WRITE
03A7 FA                         CLI                             ; BLOCK INTERRUPTS FOR SINGLE LOOP
03A8 EC                         IN      AL,DX                   ; GET STATUS FROM THE ADAPTER
03A9 A8 08                      TEST    AL,RVRT                 ; CHECK FOR VERTICAL RETRACE FIRST
03AB 75 09                      JNZ     P43                     ; DO FAST WRITE NOW IF VERTICAL RETRACE
03AD A8 01                      TEST    AL,RHRZ                 ; IS HORIZONTAL RETRACE LOW THEN
03AF 75 F1                      JNZ     P41                     ; WAIT UNTIL IT IS
03B1                    P42:                                    ;  WAIT FOR EITHER RETRACE HIGH
03B1 EC                         IN      AL,DX                   ; GET STATUS AGAIN
03B2 A8 09                      TEST    AL,RVRT+RHRZ            ; IS HORIZONTAL OR VERTICAL RETRACE HIGH
03B4 74 FB                      JZ      P42                     ; WAIT UNTIL EITHER RETRACE ACTIVE
03B6                    P43:
03B6 8B C5                      MOV     AX,BP                   ; GET THE CHARACTER SAVE IN (BP)
03B8 AA                         STOSB                           ; PUT THE CHARACTER INTO REGEN BUFFER
03B9 47                         INC     DI                      ; BUMP POINTER PAST ATTRIBUTE
03BA E2 E6                      LOOP    P41                     ;  AS MANY TIMES AS REQUESTED

03BC E9 013D R                  JMP     VIDEO_RETURN

03BF                    WRITE_C_CURRENT ENDP
```

```
                        PAGE
                        ;--------------------------------------------------------------------------------
                        ; WRITE_STRING                                                                  :
                        ;       THIS ROUTINE WRITES A STRING OF CHARACTERS TO THE CRT.                  :
                        ; INPUT                                                                        :
                        ;       (AL) = WRITE STRING COMMAND  0 - 3                                     :
                        ;       (BH) = DISPLAY PAGE (ACTIVE PAGE)                                      :
                        ;       (CX) = COUNT OF CHARACTERS TO WRITE, IF (CX) = 0 THEN RETURN            :
                        ;       (DX) = CURSOR POSITION FOR START OF STRING WRITE                        :
                        ;       (BL) = ATTRIBUTE OF CHARACTER TO WRITE IF (AL) = 0  OR  (AL) = 1        :
                        ;       (BP) = SOURCE STRING OFFSET                                            :
                        ;       [0E] = SOURCE STRING SEGMENT (FOR USE IN (ES) IN STACK +14)            :
                        ; OUTPUT                                                                       :
                        ;       NONE                                                                   :
                        ;--------------------------------------------------------------------------------
1006 03BF               WRITE_STRING    PROC    NEAR
1007 03BF 55                    PUSH    BP                      ; SAVE BUFFER OFFSET (BP) IN STACK
1008 03C0 8B EC                 MOV     BP,SP                   ; GET POINTER TO STACKED REGISTERS
1009 03C2 8E 46 10              MOV     ES,[BP]+14+2            ; RECOVER ENTRY (ES) SEGMENT REGISTER
1010 03C5 5D                    POP     BP                      ; RESTORE BUFFER OFFSET
1011 03C6 98                    CBW                             ; CLEAR (AH) REGISTER
1012 03C7 8B F8                 MOV     DI,AX                   ; SAVE (AL) COMMAND IN (DI) REGISTER
1013 03C9 3C 04                 CMP     AL,04                   ; TEST FOR INVALID WRITE STRING OPTION
1014 03CB 73 73                 JNB     P59                     ; IF OPTION INVALID THEN RETURN
1015
1016 03CD E3 71                 JCXZ    P59                     ; IF ZERO LENGTH STRING THEN RETURN
1017
1018 03CF 8B F3                 MOV     SI,BX                   ; SAVE CURRENT CURSOR PAGE
1019 03D1 8A DF                 MOV     BL,BH                   ; MOVE PAGE TO LOW BYTE
1020 03D3 32 FF                 XOR     BH,BH                   ; CLEAR HIGH BYTE
1021 03D5 87 F3                 XCHG    SI,BX                   ; MOVE OFFSET AND RESTORE PAGE REGISTER
1022 03D7 D1 E6                 SAL     SI,1                    ; CONVERT TO PAGE OFFSET  (SI= PAGE)
1023 03D9 FF B4 0050 R          PUSH    [SI+OFFSET @CURSOR_POSN]; SAVE CURRENT CURSOR POSITION IN STACK
1024 03DD B8 0200               MOV     AX,0200H                ; SET NEW CURSOR POSITION
1025 03E0 CD 10                 INT     10H
1026 03E2               P50:
1027 03E2 26: 8A 46 00          MOV     AL,ES:[BP]              ; GET CHARACTER FROM INPUT STRING
1028 03E6 45                    INC     BP                      ; BUMP POINTER TO CHARACTER
1029
1030                    ;-----  TEST FOR SPECIAL CHARACTER'S
1031
1032 03E7 3C 08                 CMP     AL,08H                  ; IS IT A BACKSPACE
1033 03E9 74 0C                 JE      P51                     ; BACK_SPACE
1034 03EB 3C 0D                 CMP     AL,CR                   ; IS IT CARRIAGE RETURN
1035 03ED 74 08                 JE      P51                     ; CAR_RET
1036 03EF 3C 0A                 CMP     AL,LF                   ; IS IT A LINE FEED
1037 03F1 74 04                 JE      P51                     ; LINE_FEED
1038 03F3 3C 07                 CMP     AL,07H                  ; IS IT A BELL
1039 03F5 75 0A                 JNE     P52                     ; IF NOT THEN DO WRITE CHARACTER
1040 03F7               P51:
1041 03F7 B4 0E                 MOV     AH,0EH                  ; TTY_CHARACTER WRITE
1042 03F9 CD 10                 INT     10H                       ;WRITE TTY CHARACTER TO THE CRT
1043 03FB 8B 94 0050 R          MOV     DX,[SI+OFFSET @CURSOR_POSN] ; GET CURRENT CURSOR POSITION
1044 03FF EB 2D                 JMP     SHORT P54                 ; SET CURSOR POSITION AND CONTINUE
1045
1046 0401               P52:
1047 0401 51                    PUSH    CX
1048 0402 53                    PUSH    BX
1049 0403 B9 0001               MOV     CX,1                    ; SET CHARACTER WRITE AMOUNT TO ONE
1050 0406 83 FF 02              CMP     DI,2                    ; IS THE ATTRIBUTE IN THE STRING
1051 0409 72 05                 JB      P53                     ; IF NOT THEN SKIP
1052 040B 26: 8A 5E 00          MOV     BL,ES:[BP]              ; ELSE GET NEW ATTRIBUTE
1053 040F 45                    INC     BP                      ; BUMP STRING POINTER
1054 0410               P53:
1055 0410 B4 09                 MOV     AH,09H                  ;  GOT_CHARACTER
1056 0412 CD 10                 INT     10H                     ; WRITE CHARACTER TO THE CRT
1057 0414 5B                    POP     BX                      ; RESTORE REGISTERS
1058 0415 59                    POP     CX
1059 0416 FE C2                 INC     DL                      ; INCREMENT COLUMN COUNTER
1060 0418 3A 16 004A R          CMP     DL,BYTE PTR @CRT_COLS   ; IF COLS ARE WITHIN RANGE FOR THIS MODE
1061 041C 72 10                 JB      P54                     ;    THEN GO TO COLUMNS SET
1062 041E FE C6                 INC     DH                      ; BUMP ROW COUNTER BY ONE
1063 0420 2A D2                 SUB     DL,DL                   ; SET COLUMN COUNTER TO ZERO
1064 0422 80 FE 19              CMP     DH,25                   ; IF ROWS ARE LESS THAN 25 THEN
1065 0425 72 07                 JB      P54                     ;    GO TO ROWS_COLUMNS_SET
1066
1067 0427 B8 0E0A               MOV     AX,0E0AH                ; ELSE SCROLL SCREEN ONE LINE
1068 042A CD 10                 INT     10H                     ; RESET ROW COUNTER TO 24
1069 042C FE CE                 DEC     DH
1070 042E               P54:                                    ;  ROW_COLUMNS_SET
1071 042E B8 0200               MOV     AX,0200H                ; SET NEW CURSOR POSITION COMMAND
1072 0431 CD 10                 INT     10H                     ; ESTABLISH NEW CURSOR POSITION
1073 0433 E2 AD                 LOOP    P50                     ; DO IT ONCE MORE UNTIL (CX) = ZERO
1074
1075 0435 5A                    POP     DX                      ; RESTORE OLD CURSOR COORDINATES
1076 0436 97                    XCHG    AX,DI                   ; RECOVER WRITE STRING COMMAND
1077 0437 A8 01                 TEST    AL,01H                  ; IF CURSOR WAS NOT TO BE MOVED THEN
1078 0439 75 05                 JNZ     P59                     ;  THEN EXIT WITHOUT RESETTING OLD VALUE
1079 043B B8 0200               MOV     AX,0200H                ; ELSE RESTORE OLD CURSOR POSITION
1080 043E CD 10                 INT     10H
1081 0440               P59:                                    ;  DONE - EXIT WRITE STRING
1082 0440 E9 013D R             JMP     VIDEO_RETURN            ; RETURN TO CALLER
1083
1084 0443               WRITE_STRING    ENDP
```

```
                        PAGE
                        ;--------------------------------------------
                        ; READ DOT  -- WRITE DOT
                        ; THESE ROUTINES WILL WRITE A DOT, OR READ THE
                        ;  DOT AT THE INDICATED LOCATION
                        ; ENTRY --
                        ;   DX = ROW (0-199)     (THE ACTUAL VALUE DEPENDS ON THE MODE)
                        ;   CX = COLUMN ( 0-639) ( THE VALUES ARE NOT RANGE CHECKED )
                        ;   AL = DOT VALUE TO WRITE (1,2 OR 4 BITS DEPENDING ON MODE,
                        ;        REQUIRED FOR WRITE DOT ONLY, RIGHT JUSTIFIED)
                        ;        BIT 7 OF AL = 1 INDICATES XOR THE VALUE INTO THE LOCATION
                        ;   DS = DATA SEGMENT
                        ;   ES = REGEN SEGMENT
                        ;
                        ; EXIT
                        ;       AL = DOT VALUE READ, RIGHT JUSTIFIED, READ ONLY
                        ;--------------------------------------------
                                ASSUME  DS:DATA,ES:DATA
0443                    READ_DOT        PROC    NEAR
0443 E8 0477 R                  CALL    R3                      ; DETERMINE BYTE POSITION OF DOT
0446 26: 8A 04                  MOV     AL,ES:[SI]              ; GET THE BYTE
0449 22 C4                      AND     AL,AH                   ; MASK OFF THE OTHER BITS IN THE BYTE
044B D2 E0                      SHL     AL,CL                   ; LEFT JUSTIFY THE VALUE
044D 8A CE                      MOV     CL,DH                   ; GET NUMBER OF BITS IN RESULT
044F D2 C0                      ROL     AL,CL                   ; RIGHT JUSTIFY THE RESULT
0451 E9 013D R                  JMP     VIDEO_RETURN            ; RETURN FROM VIDEO I/O
0454                    READ_DOT        ENDP
                        ;
0454                    WRITE_DOT       PROC    NEAR
0454 50                         PUSH    AX                      ; SAVE DOT VALUE
0455 50                         PUSH    AX                      ;  TWICE
0456 E8 0477 R                  CALL    R3                      ; DETERMINE BYTE POSITION OF THE DOT
0459 D2 E8                      SHR     AL,CL                   ; SHIFT TO SET UP THE BITS FOR OUTPUT
045B 22 C4                      AND     AL,AH                   ; STRIP OFF THE OTHER BITS
045D 26: 8A 0C                  MOV     CL,ES:[SI]              ; GET THE CURRENT BYTE
0460 5B                         POP     BX                      ; RECOVER XOR FLAG
0461 F6 C3 80                   TEST    BL,80H                  ; IS IT ON
0464 75 0D                      JNZ     R2                      ; YES, XOR THE DOT
0466 F6 D4                      NOT     AH                      ; SET MASK TO REMOVE THE INDICATED BITS
0468 22 CC                      AND     CL,AH
046A 0A C1                      OR      AL,CL                   ; OR IN THE NEW VALUE OF THOSE BITS
046C                    R1:                                     ; FINISH_DOT
046C 26: 88 04                  MOV     ES:[SI],AL              ; RESTORE THE BYTE IN MEMORY
046F 58                         POP     AX
0470 E9 013D R                  JMP     VIDEO_RETURN            ; RETURN FROM VIDEO I/O
0473                    R2:                                     ; XOR_DOT
0473 32 C1                      XOR     AL,CL                   ; EXCLUSIVE OR THE DOTS
0475 EB F5                      JMP     R1                      ; FINISH UP THE WRITING
0477                    WRITE_DOT       ENDP
                        ;--------------------------------------------
                        ; THIS SUBROUTINE DETERMINES THE REGEN BYTE LOCATION OF THE
                        ; INDICATED ROW COLUMN VALUE IN GRAPHICS MODE.
                        ; ENTRY --
                        ;  DX = ROW VALUE (0-199)
                        ;  CX = COLUMN VALUE (0-639)
                        ; EXIT --
                        ;  SI = OFFSET INTO REGEN BUFFER FOR BYTE OF INTEREST
                        ;  AH = MASK TO STRIP OFF THE BITS OF INTEREST
                        ;  CL = BITS TO SHIFT TO RIGHT JUSTIFY THE MASK IN AH
                        ;  DH = # BITS IN RESULT
                        ;  BX = MODIFIED
                        ;--------------------------------------------
0477                    R3      PROC    NEAR

                        ;-----  DETERMINE 1ST BYTE IN INDICATED ROW BY MULTIPLYING ROW VALUE BY 40
                        ;-----    ( LOW BIT OF ROW DETERMINES EVEN/ODD, 80 BYTES/ROW )

0477 96                         XCHG    SI,AX                   ; WILL SAVE AL AND AH DURING OPERATION
0478 B0 28                      MOV     AL,40
047A F6 E2                      MUL     DL                      ; AX= ADDRESS OF START OF INDICATED ROW
047C A8 08                      TEST    AL,008H                 ; TEST FOR EVEN/ODD ROW CALCULATED
047E 74 03                      JZ      R4                      ; JUMP IF EVEN ROW
0480 05 1FD8                    ADD     AX,2000H-40             ; OFFSET TO LOCATION OF ODD ROWS ADJUST
0483                    R4:                                     ; EVEN ROW
0483 96                         XCHG    SI,AX                   ; MOVE POINTER TO (SI) AND RECOVER (AX)
0484 8B D1                      MOV     DX,CX                   ; COLUMN VALUE TO DX

                        ;-----  DETERMINE GRAPHICS MODE CURRENTLY IN EFFECT

                        ; SET UP THE REGISTERS ACCORDING TO THE MODE
                        ; CH = MASK FOR LOW OF COLUMN ADDRESS ( 7/3 FOR HIGH/MED RES )
                        ; CL = # OF ADDRESS BITS IN COLUMN VALUE ( 3/2 FOR H/M )
                        ; BL = MASK TO SELECT BITS FROM POINTED BYTE ( 80H/C0H FOR H/M )
                        ; BH = NUMBER OF VALID BITS IN POINTED BYTE ( 1/2 FOR H/M )

0486 BB 02C0                    MOV     BX,2C0H
0489 B9 0302                    MOV     CX,302H                 ; SET PARMS FOR MED RES
048C 80 3E 0049 R 06            CMP     @CRT_MODE,6
0491 72 06                      JC      R5                      ; HANDLE IF MED RES
0493 BB 0180                    MOV     BX,180H
0496 B9 0703                    MOV     CX,703H                 ; SET PARMS FOR HIGH RES

                        ;-----  DETERMINE BIT OFFSET IN BYTE FROM COLUMN MASK
0499                    R5:
0499 22 EA                      AND     CH,DL                   ; ADDRESS OF PEL WITHIN BYTE TO CH

                        ;-----  DETERMINE BYTE OFFSET FOR THIS LOCATION IN COLUMN

049B D3 EA                      SHR     DX,CL                   ; SHIFT BY CORRECT AMOUNT
049D 03 F2                      ADD     SI,DX                   ; INCREMENT THE POINTER
049F 8A F7                      MOV     DH,BH                   ; GET THE # OF BITS IN RESULT TO DH

                        ;-----  MULTIPLY BH (VALID BITS IN BYTE) BY CH (BIT OFFSET)

04A1 2A C9                      SUB     CL,CL                   ; ZERO INTO STORAGE LOCATION
04A3                    R6:
04A3 D0 C8                      ROR     AL,1                    ; LEFT JUSTIFY VALUE IN AL (FOR WRITE)
04A5 02 CD                      ADD     CL,CH                   ; ADD IN THE BIT OFFSET VALUE
04A7 FE CF                      DEC     BH                      ; LOOP CONTROL
04A9 75 F8                      JNZ     R6                      ; ON EXIT, CL HAS COUNT TO RESTORE BITS
04AB 8A E3                      MOV     AH,BL                   ;  GET MASK TO AH
04AD D2 EC                      SHR     AH,CL                   ;  MOVE THE MASK TO CORRECT LOCATION
04AF C3                         RET                             ;  RETURN WITH EVERYTHING SET UP
04B0                    R3      ENDP
```

```
                        ;--------------------------------------------------------
                        ;  SCROLL UP
                        ;    THIS ROUTINE SCROLLS UP THE INFORMATION ON THE CRT
                        ; ENTRY --
                        ;  CH,CL = UPPER LEFT CORNER OF REGION TO SCROLL
                        ;  DH,DL = LOWER RIGHT CORNER OF REGION TO SCROLL
                        ;    BOTH OF THE ABOVE ARE IN CHARACTER POSITIONS
                        ;  BH = FILL VALUE FOR BLANKED LINES
                        ;  AL = # LINES TO SCROLL (AL=0 MEANS BLANK THE ENTIRE FIELD)
                        ;  DS = DATA SEGMENT
                        ;  ES = REGEN SEGMENT
                        ; EXIT --
                        ;  NOTHING, THE SCREEN IS SCROLLED
                        ;--------------------------------------------------------
04B0                    GRAPHICS_UP     PROC    NEAR
04B0 8A D8                      MOV     BL,AL                   ; SAVE LINE COUNT IN BL
04B2 8B C1                      MOV     AX,CX                   ; GET UPPER LEFT POSITION INTO AX REG

                        ;-----  USE CHARACTER SUBROUTINE FOR POSITIONING
                        ;-----  ADDRESS RETURNED IS MULTIPLIED BY 2 FROM CORRECT VALUE

04B4 E8 06F0 R                  CALL    GRAPH_POSN
04B7 8B F8                      MOV     DI,AX                   ; SAVE RESULT AS DESTINATION ADDRESS

                        ;-----  DETERMINE SIZE OF WINDOW

04B9 2B D1                      SUB     DX,CX
04BB 81 C2 0101                 ADD     DX,101H                 ; ADJUST VALUES
04BF D0 E6                      SAL     DH,1                    ; MULTIPLY ROWS BY 4 AT 8 VERT DOTS/CHAR
04C1 D0 E6                      SAL     DH,1                    ;  AND EVEN/ODD ROWS

                        ;-----  DETERMINE CRT MODE

04C3 80 3E 0049 R 06            CMP     @CRT_MODE,6             ; TEST FOR MEDIUM RES
04C8 73 04                      JNC     R7                      ; FIND_SOURCE

                        ;-----  MEDIUM RES UP
04CA D0 E2                      SAL     DL,1                    ; # COLUMNS * 2, SINCE 2 BYTES/CHAR
04CC D1 E7                      SAL     DI,1                    ; OFFSET *2 SINCE 2 BYTES/CHAR

                        ;-----  DETERMINE THE SOURCE ADDRESS IN THE BUFFER
04CE                    R7:                                     ; FIND_SOURCE
04CE 06                         PUSH    ES                      ; GET SEGMENTS BOTH POINTING TO REGEN
04CF 1F                         POP     DS
04D0 2A ED                      SUB     CH,CH                   ; ZERO TO HIGH OF COUNT REGISTER
04D2 D0 E3                      SAL     BL,1                    ; MULTIPLY NUMBER OF LINES BY 4
04D4 D0 E3                      SAL     BL,1
04D6 74 2B                      JZ      R11                     ; IF ZERO, THEN BLANK ENTIRE FIELD
04D8 B0 50                      MOV     AL,80                   ; 80 BYTES/ROW
04DA F6 E3                      MUL     BL                      ; DETERMINE OFFSET TO SOURCE
04DC 8B F7                      MOV     SI,DI                   ; SET UP SOURCE
04DE 03 F0                      ADD     SI,AX                   ;  ADD IN OFFSET TO IT
04E0 8A E6                      MOV     AH,DH                   ; NUMBER OF ROWS IN FIELD
04E2 2A E3                      SUB     AH,BL                   ; DETERMINE NUMBER TO MOVE

                        ;-----  LOOP THROUGH, MOVING ONE ROW AT A TIME, BOTH EVEN AND ODD FIELDS
04E4                    R8:                                     ; ROW_LOOP
04E4 E8 0564 R                  CALL    R17                     ; MOVE ONE ROW
04E7 81 EE 1FB0                 SUB     SI,2000H-80             ; MOVE TO NEXT ROW
04EB 81 EF 1FB0                 SUB     DI,2000H-80
04EF FE CC                      DEC     AH                      ; NUMBER OF ROWS TO MOVE
04F1 75 F1                      JNZ     R8                      ; CONTINUE TILL ALL MOVED

                        ;-----  FILL IN THE VACATED LINE(S)
04F3                    R9:                                     ; CLEAR_ENTRY
04F3 8A C7                      MOV     AL,BH                   ; ATTRIBUTE TO FILL WITH
04F5                    R10:
04F5 E8 057D R                  CALL    R18                     ; CLEAR THAT ROW
04F8 81 EF 1FB0                 SUB     DI,2000H-80             ; POINT TO NEXT LINE
04FC FE CB                      DEC     BL                      ; NUMBER OF LINES TO FILL
04FE 75 F5                      JNZ     R10                     ; CLEAR_LOOP
0500 E9 013D R                  JMP     VIDEO_RETURN            ; EVERYTHING DONE

0503                    R11:                                    ; BLANK_FIELD
0503 8A DE                      MOV     BL,DH                   ; SET BLANK COUNT TO EVERYTHING IN FIELD
0505 EB EC                      JMP     R9                      ; CLEAR THE FIELD
0507                    GRAPHICS_UP     ENDP
                        ;--------------------------------------------------------
                        ; SCROLL DOWN
                        ;  THIS ROUTINE SCROLLS DOWN THE INFORMATION ON THE CRT
                        ; ENTRY --
                        ;  CH,CL = UPPER LEFT CORNER OF REGION TO SCROLL
                        ;  DH,DL = LOWER RIGHT CORNER OF REGION TO SCROLL
                        ;    BOTH OF THE ABOVE ARE IN CHARACTER POSITIONS
                        ;  BH = FILL VALUE FOR BLANKED LINES
                        ;  AL = # LINES TO SCROLL (AL=0 MEANS BLANK THE ENTIRE FIELD)
                        ;  DS = DATA SEGMENT
                        ;  ES = REGEN SEGMENT
                        ; EXIT --
                        ;  NOTHING, THE SCREEN IS SCROLLED
                        ;--------------------------------------------------------

0507                    GRAPHICS_DOWN   PROC    NEAR
0507 FD                         STD                             ; SET DIRECTION
0508 8A D8                      MOV     BL,AL                   ; SAVE LINE COUNT IN BL
050A 8B C2                      MOV     AX,DX                   ; GET LOWER RIGHT POSITION INTO AX REG

                        ;-----  USE CHARACTER SUBROUTINE FOR POSITIONING
                        ;-----  ADDRESS RETURNED IS MULTIPLIED BY 2 FROM CORRECT VALUE

050C E8 06F0 R                  CALL    GRAPH_POSN
050F 8B F8                      MOV     DI,AX                   ; SAVE RESULT AS DESTINATION ADDRESS

                        ;-----  DETERMINE SIZE OF WINDOW

0511 2B D1                      SUB     DX,CX
0513 81 C2 0101                 ADD     DX,101H                 ; ADJUST VALUES
0517 D0 E6                      SAL     DH,1                    ; MULTIPLY ROWS BY 4 AT 8 VERT DOTS/CHAR
0519 D0 E6                      SAL     DH,1                    ;  AND EVEN/ODD ROWS

                        ;-----  DETERMINE CRT MODE

051B 80 3E 0049 R 06            CMP     @CRT_MODE,6             ; TEST FOR MEDIUM RES
0520 73 05                      JNC     R12                     ; FIND_SOURCE_DOWN
```

SECTION 5

```
                        ;-----  MEDIUM RES DOWN
0522 D0 E2                      SAL     DL,1                    ; # COLUMNS * 2, SINCE 2 BYTES/CHAR
0524 D1 E7                      SAL     DI,1                    ; OFFSET *2 SINCE 2 BYTES/CHAR
0526 47                         INC     DI                      ; POINT TO LAST BYTE
                        ;-----  DETERMINE THE SOURCE ADDRESS IN THE BUFFER
0527                    R12:                                    ; FIND_SOURCE_DOWN
0527 06                         PUSH    ES                      ; BOTH SEGMENTS TO REGEN
0528 1F                         POP     DS
0529 2A ED                      SUB     CH,CH                   ; ZERO TO HIGH OF COUNT REGISTER
052B 81 C7 00F0                 ADD     DI,240                  ; POINT TO LAST ROW OF PIXELS
052F D0 E3                      SAL     BL,1                    ; MULTIPLY NUMBER OF LINES BY 4
0531 D0 E3                      SAL     BL,1
0533 74 2B                      JZ      R16                     ; IF ZERO, THEN BLANK ENTIRE FIELD
0535 B0 50                      MOV     AL,80                   ; 80 BYTES/ROW
0537 F6 E3                      MUL     BL                      ; DETERMINE OFFSET TO SOURCE
0539 8B F7                      MOV     SI,DI                   ; SET UP SOURCE
053B 2B F0                      SUB     SI,AX                   ;  SUBTRACT THE OFFSET
053D 8A E6                      MOV     AH,DH                   ; NUMBER OF ROWS IN FIELD
053F 2A E3                      SUB     AH,BL                   ; DETERMINE NUMBER TO MOVE
                        ;-----  LOOP THROUGH, MOVING ONE ROW AT A TIME, BOTH EVEN AND ODD FIELDS
0541                    R13:                                    ; ROW_LOOP_DOWN
0541 E8 0564 R                  CALL    R17                     ; MOVE ONE ROW
0544 81 EE 2050                 SUB     SI,2000H+80             ; MOVE TO NEXT ROW
0548 81 EF 2050                 SUB     DI,2000H+80
054C FE CC                      DEC     AH                      ; NUMBER OF ROWS TO MOVE
054E 75 F1                      JNZ     R13                     ; CONTINUE TILL ALL MOVED
                        ;-----  FILL IN THE VACATED LINE(S)
0550                    R14:                                    ; CLEAR_ENTRY_DOWN
0550 8A C7                      MOV     AL,BH                   ; ATTRIBUTE TO FILL WITH
0552                    R15:                                    ; CLEAR_LOOP_DOWN
0552 E8 057D R                  CALL    R18                     ; CLEAR A ROW
0555 81 EF 2050                 SUB     DI,2000H+80             ; POINT TO NEXT LINE
0559 FE CB                      DEC     BL                      ; NUMBER OF LINES TO FILL
055B 75 F5                      JNZ     R15                     ; CLEAR_LOOP_DOWN
055D E9 013D R                  JMP     VIDEO_RETURN            ; EVERYTHING DONE
0560                    R16:                                    ; BLANK_FIELD_DOWN
0560 8A DE                      MOV     BL,DH                   ; SET BLANK COUNT TO EVERYTHING IN FIELD
0562 EB EC                      JMP     R14                     ; CLEAR THE FIELD
0564                    GRAPHICS_DOWN   ENDP
                        ;-----  ROUTINE TO MOVE ONE ROW OF INFORMATION
0564                    R17     PROC    NEAR
0564 8A CA                      MOV     CL,DL                   ; NUMBER OF BYTES IN THE ROW
0566 56                         PUSH    SI
0567 57                         PUSH    DI                      ; SAVE POINTERS
0568 F3/ A4                     REP     MOVSB                   ; MOVE THE EVEN FIELD
056A 5F                         POP     DI
056B 5E                         POP     SI
056C 81 C6 2000                 ADD     SI,2000H
0570 81 C7 2000                 ADD     DI,2000H                ; POINT TO THE ODD FIELD
0574 56                         PUSH    SI
0575 57                         PUSH    DI                      ; SAVE THE POINTERS
0576 8A CA                      MOV     CL,DL                   ; COUNT BACK
0578 F3/ A4                     REP     MOVSB                   ; MOVE THE ODD FIELD
057A 5F                         POP     DI
057B 5E                         POP     SI                      ; POINTERS BACK
057C C3                         RET                             ; RETURN TO CALLER
057D                    R17     ENDP
                        ;-----  CLEAR A SINGLE ROW
057D                    R18     PROC    NEAR
057D 8A CA                      MOV     CL,DL                   ; NUMBER OF BYTES IN FIELD
057F 57                         PUSH    DI                      ; SAVE POINTER
0580 F3/ AA                     REP     STOSB                   ; STORE THE NEW VALUE
0582 5F                         POP     DI                      ; POINTER BACK
0583 81 C7 2000                 ADD     DI,2000H                ; POINT TO ODD FIELD
0587 57                         PUSH    DI
0588 8A CA                      MOV     CL,DL
058A F3/ AA                     REP     STOSB                   ; FILL THE ODD FIELD
058C 5F                         POP     DI
058D C3                         RET                             ; RETURN TO CALLER
058E                    R18     ENDP

                        ;--------------------------------------------------------
                        ; GRAPHICS WRITE
                        ;  THIS ROUTINE WRITES THE ASCII CHARACTER TO THE CURRENT
                        ;  POSITION ON THE SCREEN.
                        ; ENTRY --
                        ;  AL = CHARACTER TO WRITE
                        ;  BL = COLOR ATTRIBUTE TO BE USED FOR FOREGROUND COLOR
                        ;       IF BIT 7 IS SET, THE CHAR IS XOR'D INTO THE REGEN BUFFER
                        ;       (0 IS USED FOR THE BACKGROUND COLOR)
                        ;  CX = NUMBER OF CHARS TO WRITE
                        ;  DS = DATA SEGMENT
                        ;  ES = REGEN SEGMENT
                        ; EXIT --
                        ;  NOTHING IS RETURNED
                        ;
                        ; GRAPHICS READ
                        ;   THIS ROUTINE READS THE ASCII CHARACTER AT THE CURRENT CURSOR
                        ;   POSITION ON THE SCREEN BY MATCHING THE DOTS ON THE SCREEN TO THE
                        ;   CHARACTER GENERATOR CODE POINTS
                        ; ENTRY --
                        ;  NONE  (0 IS ASSUMED AS THE BACKGROUND COLOR)
                        ; EXIT --
                        ;  AL = CHARACTER READ AT THAT POSITION (0 RETURNED IF NONE FOUND)
                        ;
                        ; FOR BOTH ROUTINES, THE IMAGES USED TO FORM CHARS ARE CONTAINED IN ROM
                        ;  FOR THE 1ST 128 CHARS.  TO ACCESS CHARS IN THE SECOND HALF, THE USER
                        ;  MUST INITIALIZE THE VECTOR AT INTERRUPT 1FH (LOCATION 0007CH) TO
                        ;  POINT TO THE USER SUPPLIED TABLE OF GRAPHIC IMAGES (8X8 BOXES).
                        ;  FAILURE TO DO SO WILL CAUSE IN STRANGE RESULTS
                        ;--------------------------------------------------------
```

```
                        ASSUME  DS:DATA,ES:DATA
058E                    GRAPHICS_WRITE  PROC    NEAR
058E B4 00                      MOV     AH,0                    ; ZERO TO HIGH OF CODE POINT
0590 50                         PUSH    AX                      ; SAVE CODE POINT VALUE
                        ;-----  DETERMINE POSITION IN REGEN BUFFER TO PUT CODE POINTS
0591 E8 06ED R                  CALL    S26                     ; FIND LOCATION IN REGEN BUFFER
0594 8B F8                      MOV     DI,AX                   ; REGEN POINTER IN DI
                        ;-----  DETERMINE REGION TO GET CODE POINTS FROM
0596 58                         POP     AX                      ; RECOVER CODE POINT
0597 3C 80                      CMP     AL,80H                  ; IS IT IN SECOND HALF
0599 73 06                      JAE     S1                      ; YES
                        ;-----  IMAGE IS IN FIRST HALF, CONTAINED IN ROM
059B BE 0000 E                  MOV     SI,OFFSET CRT_CHAR_GEN  ; OFFSET OF IMAGES
059E 0E                         PUSH    CS                      ; SAVE SEGMENT ON STACK
059F EB 18                      JMP     SHORT S2                ; DETERMINE_MODE
                        ;-----  IMAGE IS IN SECOND HALF, IN USER MEMORY
05A1                    S1:                                     ; EXTEND_CHAR
05A1 2C 80                      SUB     AL,80H                  ; ZERO ORIGIN FOR SECOND HALF
05A3 1E                         PUSH    DS                      ; SAVE DATA POINTER
05A4 2B F6                      SUB     SI,SI
05A6 8E DE                      MOV     DS,SI                   ; ESTABLISH VECTOR ADDRESSING
                                ASSUME  DS:ABS0
05A8 C5 36 007C R               LDS     SI,@EXT_PTR             ; GET THE OFFSET OF THE TABLE
05AC 8C DA                      MOV     DX,DS                   ; GET THE SEGMENT OF THE TABLE
                                ASSUME  DS:DATA
05AE 1F                         POP     DS                      ; RECOVER DATA SEGMENT
05AF 52                         PUSH    DX                      ; SAVE TABLE SEGMENT ON STACK
05B0 0B D6                      OR      DX,SI                   ; CHECK FOR VALID TABLE DEFINED
05B2 75 05                      JNZ     S2                      ; CONTINUE IF DS:SI NOT 0000:0000
05B4 58                         POP     AX                      ; ELSE SET (AX)= 0000 FOR "NULL"
05B5 BE 0000 E                  MOV     SI,OFFSET CRT_CHAR_GEN  ; POINT TO DEFAULT TABLE OFFSET
05B8 0E                         PUSH    CS                      ;  IN THE CODE SEGMENT
                        ;-----  DETERMINE GRAPHICS MODE IN OPERATION
05B9                    S2:                                     ;       DETERMINE_MODE
05B9 D1 E0                      SAL     AX,1                    ; MULTIPLY CODE POINT VALUE BY 8
05BB D1 E0                      SAL     AX,1
05BD D1 E0                      SAL     AX,1
05BF 03 F0                      ADD     SI,AX                   ; SI HAS OFFSET OF DESIRED CODES
05C1 80 3E 0049 R 06            CMP     @CRT_MODE,6
05C6 1F                         POP     DS                      ; RECOVER TABLE POINTER SEGMENT
05C7 72 2C                      JC      S7                      ; TEST FOR MEDIUM RESOLUTION MODE
                        ;-----  HIGH RESOLUTION MODE
05C9                    S3:                                     ; HIGH_CHAR
05C9 57                         PUSH    DI                      ; SAVE REGEN POINTER
05CA 56                         PUSH    SI                      ; SAVE CODE POINTER
05CB B6 04                      MOV     DH,4                    ; NUMBER OF TIMES THROUGH LOOP
05CD                    S4:
05CD AC                         LODSB                           ; GET BYTE FROM CODE POINTS
05CE F6 C3 80                   TEST    BL,80H                  ; SHOULD WE USE THE FUNCTION
05D1 75 16                      JNZ     S6                      ;  TO PUT CHAR IN
05D3 AA                         STOSB                           ; STORE IN REGEN BUFFER
05D4 AC                         LODSB
05D5                    S5:
05D5 26: 88 85 1FFF             MOV     ES:[DI+2000H-1],AL      ; STORE IN SECOND HALF
05DA 83 C7 4F                   ADD     DI,79                   ; MOVE TO NEXT ROW IN REGEN
05DD FE CE                      DEC     DH                      ; DONE WITH LOOP
05DF 75 EC                      JNZ     S4
05E1 5E                         POP     SI
05E2 5F                         POP     DI                      ; RECOVER REGEN POINTER
05E3 47                         INC     DI                      ; POINT TO NEXT CHAR POSITION
05E4 E2 E3                      LOOP    S3                      ; MORE CHARS TO WRITE
05E6 E9 013D R                  JMP     VIDEO_RETURN
05E9                    S6:
05E9 26: 32 05                  XOR     AL,ES:[DI]              ; EXCLUSIVE OR WITH CURRENT
05EC AA                         STOSB                           ; STORE THE CODE POINT
05ED AC                         LODSB                           ; AGAIN FOR ODD FIELD
05EE 26: 32 85 1FFF             XOR     AL,ES:[DI+2000H-1]
05F3 EB E0                      JMP     S5                      ; BACK TO MAINSTREAM
                        ;-----  MEDIUM RESOLUTION WRITE
05F5                    S7:                                     ;  MED_RES_WRITE
05F5 8A D3                      MOV     DL,BL                   ; SAVE HIGH COLOR BIT
05F7 D1 E7                      SAL     DI,1                    ; OFFSET*2 SINCE 2 BYTES/CHAR
                                                                ; EXPAND BL TO FULL WORD OF COLOR
05F9 80 E3 03                   AND     BL,3                    ; ISOLATE THE COLOR BITS ( LOW 2 BITS )
05FC B0 55                      MOV     AL,055H                 ; GET BIT CONVERSION MULTIPLIER
05FE F6 E3                      MUL     BL                      ; EXPAND 2 COLOR BITS TO 4 REPLICATIONS
0600 8A D8                      MOV     BL,AL                   ; PLACE BACK IN WORK REGISTER
0602 8A F8                      MOV     BH,AL                   ; EXPAND TO 8 REPLICATIONS OF COLOR BITS
0604                    S8:                                     ;  MED_CHAR
0604 57                         PUSH    DI                      ; SAVE REGEN POINTER
0605 56                         PUSH    SI                      ; SAVE THE CODE POINTER
0606 B6 04                      MOV     DH,4                    ; NUMBER OF LOOPS
0608                    S9:
0608 AC                         LODSB                           ; GET CODE POINT
0609 E8 06C4 R                  CALL    S2I                     ; DOUBLE UP ALL THE BITS
060C 23 C3                      AND     AX,BX                   ; CONVERT TO FOREGROUND COLOR ( 0 BACK )
060E 86 E0                      XCHG    AH,AL                   ; SWAP HIGH/LOW BYTES FOR WORD MOVE
0610 F6 C2 80                   TEST    DL,80H                  ; IS THIS XOR FUNCTION
0613 74 03                      JZ      S10                     ; NO, STORE IT IN AS IT IS
0615 26: 33 05                  XOR     AX,ES:[DI]              ; DO FUNCTION WITH LOW/HIGH
0618                    S10:
0618 26: 89 05                  MOV     ES:[DI],AX              ; STORE FIRST BYTE HIGH, SECOND LOW
061B AC                         LODSB                           ; GET CODE POINT
061C E8 06C4 R                  CALL    S2I
061F 23 C3                      AND     AX,BX                   ; CONVERT TO COLOR
0621 86 E0                      XCHG    AH,AL                   ; SWAP HIGH/LOW BYTES FOR WORD MOVE
0623 F6 C2 80                   TEST    DL,80H                  ; AGAIN, IS THIS XOR FUNCTION
0626 74 05                      JZ      S11                     ; NO, JUST STORE THE VALUES
0628 26: 33 85 2000             XOR     AX,ES:[DI+2000H]        ; FUNCTION WITH FIRST HALF LOW
```

```
                        S11:
                                MOV     ES:[DI+2000H],AX        ; STORE SECOND PORTION HIGH
                                ADD     DI,80                   ; POINT TO NEXT LOCATION
                                DEC     DH
                                JNZ     S9                      ; KEEP GOING
                                POP     SI                      ; RECOVER CODE POINTER
                                POP     DI                      ; RECOVER REGEN POINTER
                                INC     DI                      ; POINT TO NEXT CHAR POSITION
                                INC     DI
                                LOOP    S8                      ; MORE TO WRITE
                                JMP     VIDEO_RETURN
                        GRAPHICS_WRITE  ENDP
                        ;--------------------------------------
                        ; GRAPHICS READ
                        ;--------------------------------------
                        GRAPHICS_READ   PROC    NEAR
                                CALL    S26                     ; CONVERTED TO OFFSET IN REGEN
                                MOV     SI,AX                   ; SAVE IN SI
                                SUB     SP,8                    ; ALLOCATE SPACE FOR THE READ CODE POINT
                                MOV     BP,SP                   ; POINTER TO SAVE AREA

                        ;-----  DETERMINE GRAPHICS MODES
                                CMP     @CRT_MODE,6
                                PUSH    ES
                                POP     DS                      ; POINT TO REGEN SEGMENT
                                JC      S13                     ; MEDIUM RESOLUTION

                        ;-----  HIGH RESOLUTION READ
                        ;-----  GET VALUES FROM REGEN BUFFER AND CONVERT TO CODE POINT
                                MOV     DH,4                    ; NUMBER OF PASSES
                        S12:
                                MOV     AL,[SI]                 ; GET FIRST BYTE
                                MOV     [BP],AL                 ; SAVE IN STORAGE AREA
                                INC     BP                      ; NEXT LOCATION
                                MOV     AL,[SI+2000H]           ; GET LOWER REGION BYTE
                                MOV     [BP],AL                 ; ADJUST AND STORE
                                INC     BP
                                ADD     SI,80                   ; POINTER INTO REGEN
                                DEC     DH                      ; LOOP CONTROL
                                JNZ     S12                     ; DO IT SOME MORE
                                JMP     SHORT S15               ; GO MATCH THE SAVED CODE POINTS

                        ;-----  MEDIUM RESOLUTION READ
                        S13:    SAL     SI,1                    ; OFFSET*2 SINCE 2 BYTES/CHAR
                                MOV     DH,4                    ; NUMBER OF PASSES
                        S14:
                                CALL    S23                     ; GET BYTES FROM REGEN INTO SINGLE SAVE
                                ADD     SI,2000H-2              ; GO TO LOWER REGION
                                CALL    S23                     ; GET THIS PAIR INTO SAVE
                                SUB     SI,2000H-80+2           ; ADJUST POINTER BACK INTO UPPER
                                DEC     DH
                                JNZ     S14                     ; KEEP GOING UNTIL ALL 8 DONE

                        ;-----  SAVE AREA HAS CHARACTER IN IT, MATCH IT
                        S15:                                    ; FIND_CHAR
                                MOV     DI,OFFSET CRT_CHAR_GEN  ; ESTABLISH ADDRESSING
                                PUSH    CS
                                POP     ES                      ; CODE POINTS IN CS
                                SUB     BP,8                    ; ADJUST POINTER TO START OF SAVE AREA
                                MOV     SI,BP
                                MOV     AL,0                    ; CURRENT CODE POINT BEING MATCHED
                        S16:
                                PUSH    SS                      ; ESTABLISH ADDRESSING TO STACK
                                POP     DS                      ; FOR THE STRING COMPARE
                                MOV     DX,128                  ; NUMBER TO TEST AGAINST
                        S17:
                                PUSH    SI                      ; SAVE SAVE AREA POINTER
                                PUSH    DI                      ; SAVE CODE POINTER
                                MOV     CX,4                    ; NUMBER OF WORDS TO MATCH
                                REPE    CMPSW                   ; COMPARE THE 8 BYTES AS WORDS
                                POP     DI                      ; RECOVER THE POINTERS
                                POP     SI
                                JZ      S18                     ; IF ZERO FLAG SET, THEN MATCH OCCURRED
                                INC     AL                      ; NO MATCH, MOVE ON TO NEXT
                                ADD     DI,8                    ; NEXT CODE POINT
                                DEC     DX                      ; LOOP CONTROL
                                JNZ     S17                     ; DO ALL OF THEM

                        ;-----  CHAR NOT MATCHED, MIGHT BE IN USER SUPPLIED SECOND HALF
                                CMP     AL,0                    ;  AL<> 0 IF ONLY 1ST HALF SCANNED
                                JE      S18                     ;  IF = 0, THEN ALL HAS BEEN SCANNED
                                SUB     AX,AX
                                MOV     DS,AX                   ;  ESTABLISH ADDRESSING TO VECTOR
                                ASSUME  DS:ABS0
                                LES     DI,@EXT_PTR             ;  GET POINTER
                                MOV     AX,ES                   ;  SEE IF THE POINTER REALLY EXISTS
                                OR      AX,DI                   ; IF ALL 0, THEN DOESN'T EXIST
                                JZ      S18                     ; NO SENSE LOOKING
                                MOV     AL,128                  ; ORIGIN FOR SECOND HALF
                                JMP     S16                     ; GO BACK AND TRY FOR IT
                                ASSUME  DS:DATA

                        ;-----  CHARACTER IS FOUND ( AL=0 IF NOT FOUND )
                        S18:    ADD     SP,8                    ; READJUST THE STACK, THROW AWAY SAVE
                                JMP     VIDEO_RETURN            ; ALL DONE
                        GRAPHICS_READ   ENDP
                        ;--------------------------------------------
                        ; EXPAND_BYTE
                        ;  THIS ROUTINE TAKES THE BYTE IN AL AND DOUBLES ALL
                        ;  OF THE BITS, TURNING THE 8 BITS INTO 16 BITS.
                        ;  THE RESULT IS LEFT IN AX
                        ;--------------------------------------------
                        S21     PROC    NEAR
                                PUSH    CX                      ; SAVE REGISTER
                                MOV     CX,8                    ; SHIFT COUNT REGISTER FOR ONE BYTE
                        S22:
                                ROR     AL,1                    ; SHIFT BITS, LOW BIT INTO CARRY FLAG
                                RCR     BP,1                    ; MOVE CARRY FLAG (LOW BIT) INTO RESULTS
                                SAR     BP,1                    ; SIGN EXTEND HIGH BIT (DOUBLE IT)
                                LOOP    S22                     ; REPEAT FOR ALL 8 BITS
                                XCHG    AX,BP                   ; MOVE RESULTS TO PARAMETER REGISTER
                                POP     CX                      ; RECOVER REGISTER
                                RET                             ; ALL DONE
                        S21     ENDP
```

```
                        ;--------------------------------------------------
                        ; MED_READ_BYTE
                        ; THIS ROUTINE WILL TAKE 2 BYTES FROM THE REGEN BUFFER,
                        ;  COMPARE AGAINST THE CURRENT FOREGROUND COLOR, AND PLACE
                        ;  THE CORRESPONDING ON/OFF BIT PATTERN INTO THE CURRENT
                        ;  POSITION IN THE SAVE AREA
                        ; ENTRY --
                        ;  SI,DS = POINTER TO REGEN AREA OF INTEREST
                        ;  BX = EXPANDED FOREGROUND COLOR
                        ;  BP = POINTER TO SAVE AREA
                        ; EXIT --
                        ;  SI AND BP ARE INCREMENTED
                        ;--------------------------------------------------
06D3                    S23     PROC    NEAR
06D3 AD                         LODSW                   ; GET FIRST BYTE AND SECOND BYTES
06D4 86 C4                      XCHG    AL,AH           ; SWAP FOR COMPARE
06D6 B9 C000                    MOV     CX,0C000H       ; 2 BIT MASK TO TEST THE ENTRIES
06D9 B2 00                      MOV     DL,0            ; RESULT REGISTER
06DB                    S24:
06DB 85 C1                      TEST    AX,CX           ; IS THIS SECTION BACKGROUND?
06DD 74 01                      JZ      S25             ; IF ZERO, IT IS BACKGROUND (CARRY=0)
06DF F9                         STC                     ; WASN'T, SO SET CARRY
06E0                    S25:
06E0 D0 D2                      RCL     DL,1            ; MOVE THAT BIT INTO THE RESULT
06E2 D1 E9                      SHR     CX,1
06E4 D1 E9                      SHR     CX,1            ; MOVE THE MASK TO THE RIGHT BY 2 BITS
06E6 73 F3                      JNC     S24             ; DO IT AGAIN IF MASK DIDN'T FALL OUT
06E8 88 56 00                   MOV     [BP],DL         ; STORE RESULT IN SAVE AREA
06EB 45                         INC     BP              ; ADJUST POINTER
06EC C3                         RET                     ; ALL DONE
06ED                    S23     ENDP
                        ;--------------------------------------
                        ; V4_POSITION
                        ;  THIS ROUTINE TAKES THE CURSOR POSITION CONTAINED IN
                        ;  THE MEMORY LOCATION, AND CONVERTS IT INTO AN OFFSET
                        ;  INTO THE REGEN BUFFER, ASSUMING ONE BYTE/CHAR.
                        ;  FOR MEDIUM RESOLUTION GRAPHICS, THE NUMBER MUST
                        ;  BE DOUBLED.
                        ; ENTRY -- NO REGISTERS,MEMORY LOCATION @CURSOR_POSN IS USED
                        ; EXIT--
                        ;  AX CONTAINS OFFSET INTO REGEN BUFFER
                        ;--------------------------------------
06ED                    S26     PROC    NEAR
06ED A1 0050 R                  MOV     AX,@CURSOR_POSN ; GET CURRENT CURSOR
06F0                    GRAPH_POSN      LABEL   NEAR
06F0 53                         PUSH    BX              ; SAVE REGISTER
06F1 8B D8                      MOV     BX,AX           ; SAVE A COPY OF CURRENT CURSOR
06F3 A0 004A R                  MOV     AL,BYTE PTR @CRT_COLS   ; GET BYTES PER COLUMN
06F6 F6 E4                      MUL     AH              ; MULTIPLY BY ROWS
06F8 D1 E0                      SHL     AX,1
06FA D1 E0                      SHL     AX,1            ; MULTIPLY * 4 SINCE 4 ROWS/BYTE
06FC 2A FF                      SUB     BH,BH           ; ISOLATE COLUMN VALUE
06FE 03 C3                      ADD     AX,BX           ; DETERMINE OFFSET
0700 5B                         POP     BX              ; RECOVER POINTER
0701 C3                         RET                     ; ALL DONE
0702                    S26     ENDP
                        ;--- WRITE_TTY ------------------------------------------------------------------------------
                        ;                                                                                       :
                        ;   THIS INTERFACE PROVIDES A TELETYPE LIKE INTERFACE TO THE                            :
                        ;     VIDEO CARDS.  THE INPUT CHARACTER IS WRITTEN TO THE CURRENT                       :
                        ;     CURSOR POSITION, AND THE CURSOR IS MOVED TO THE NEXT POSITION.                    :
                        ;     IF THE CURSOR LEAVES THE LAST COLUMN OF THE FIELD, THE COLUMN                     :
                        ;     IS SET TO ZERO, AND THE ROW VALUE IS INCREMENTED.  IF THE ROW                     :
                        ;     ROW VALUE LEAVES THE FIELD, THE CURSOR IS PLACED ON THE LAST ROW,                 :
                        ;     FIRST COLUMN, AND THE ENTIRE SCREEN IS SCROLLED UP ONE LINE.                      :
                        ;     WHEN THE SCREEN IS SCROLLED UP, THE ATTRIBUTE FOR FILLING THE                     :
                        ;     NEWLY BLANKED LINE IS READ FROM THE CURSOR POSITION ON THE PREVIOUS               :
                        ;     LINE BEFORE THE SCROLL, IN CHARACTER MODE.  IN GRAPHICS MODE,                     :
                        ;     THE 0 COLOR IS USED.                                                              :
                        ;   ENTRY --                                                                            :
                        ;     (AH) = CURRENT CRT MODE                                                           :
                        ;     (AL) = CHARACTER TO BE WRITTEN                                                    :
                        ;             NOTE THAT BACK SPACE, CARRIAGE RETURN, BELL AND LINE FEED ARE             :
                        ;             HANDLED AS COMMANDS RATHER THAN AS DISPLAY GRAPHICS CHARACTERS            :
                        ;     (BL) = FOREGROUND COLOR FOR CHAR WRITE IF CURRENTLY IN A GRAPHICS MODE             :
                        ;   EXIT --                                                                             :
                        ;     ALL REGISTERS SAVED THROUGH VIDEO_EXIT (INCLUDING (AX))                           :
                        ;-----------------------------------------------------------------------------------------
                                ASSUME  DS:DATA
0702                    WRITE_TTY       PROC    NEAR
0702 97                         XCHG    DI,AX           ; SAVE (AX) REGISTER IN (DI) FOR EXIT
0703 B4 03                      MOV     AH,03H          ; READ CURSOR POSITION
0705 8A 3E 0062 R               MOV     BH,@ACTIVE_PAGE ; GET CURRENT PAGE SETTING
0709 CD 10                      INT     10H             ; READ THE CURRENT CURSOR POSITION
070B 8B C7                      MOV     AX,DI           ; RECOVER CHARACTER FROM (DI) REGISTER
                        ;-----  DX NOW HAS THE CURRENT CURSOR POSITION
070D 3C 0D                      CMP     AL,CR           ; IS IT CARRIAGE RETURN OR CONTROL
070F 76 46                      JBE     U8              ; GO TO CONTROL CHECKS IF IT IS
                        ;-----  WRITE THE CHAR TO THE SCREEN
0711 B4 0A              U0:     MOV     AH,0AH          ; WRITE CHARACTER ONLY COMMAND
0713 B9 0001                    MOV     CX,1            ; ONLY ONE CHARACTER
0716 CD 10                      INT     10H             ; WRITE THE CHARACTER
                        ;-----  POSITION THE CURSOR FOR NEXT CHAR
0718 FE C2                      INC     DL
071A 3A 16 004A R               CMP     DL,BYTE PTR @CRT_COLS   ; TEST FOR COLUMN OVERFLOW
071E 75 33                      JNZ     U7              ;  SET_CURSOR
0720 B2 00                      MOV     DL,0            ; COLUMN FOR CURSOR
0722 80 FE 18                   CMP     DH,25-1         ; CHECK FOR LAST ROW
0725 75 2A                      JNZ     U6              ;  SET_CURSOR_INC
                        ;-----  SCROLL REQUIRED
0727 B4 02              U1:     MOV     AH,02H
0729 CD 10                      INT     10H             ; SET THE CURSOR
                        ;-----  DETERMINE VALUE TO FILL WITH DURING SCROLL
072B A0 0049 R                  MOV     AL,@CRT_MODE    ; GET THE CURRENT MODE
072E 3C 04                      CMP     AL,4
0730 72 06                      JC      U2              ; READ-CURSOR
```

SECTION 5

```
                        CMP     AL,7
                        MOV     BH,0                    ; FILL WITH BACKGROUND
                        JNE     U3                      ; SCROLL-UP
                U2:                                     ; READ-CURSOR
                        MOV     AH,08H                  ; GET READ CURSOR COMMAND
                        INT     10H                     ; READ CHAR/ATTR AT CURRENT CURSOR
                        MOV     BH,AH                   ; STORE IN BH
                U3:                                     ; SCROLL-UP
                        MOV     AX,0601H                ; SCROLL ONE LINE
                        SUB     CX,CX                   ; UPPER LEFT CORNER
                        MOV     DH,25-1                 ; LOWER RIGHT ROW
                        MOV     DL,BYTE PTR @CRT_COLS   ; LOWER RIGHT COLUMN
                        DEC     DL
                U4:                                     ; VIDEO-CALL-RETURN
                        INT     10H                     ; SCROLL UP THE SCREEN
                U5:                                     ; TTY-RETURN
                        XCHG    AX,DI                   ; RESTORE THE ENTRY CHARACTER FROM (DI)
                        JMP     VIDEO_RETURN            ; RETURN TO CALLER

                U6:                                     ; SET-CURSOR-INC
                        INC     DH                      ; NEXT ROW
                U7:                                     ; SET-CURSOR
                        MOV     AH,02H
                        JMP     U4                      ; ESTABLISH THE NEW CURSOR

                ;-----  CHECK FOR CONTROL CHARACTERS
                U8:
                        JE      U9                      ; WAS IT A CARRIAGE RETURN
                        CMP     AL,LF                   ; IS IT A LINE FEED
                        JE      U10                     ; GO TO LINE FEED
                        CMP     AL,07H                  ; IS IT A BELL
                        JE      U11                     ; GO TO BELL
                        CMP     AL,08H                  ; IS IT A BACKSPACE
                        JNE     U0                      ; IF NOT A CONTROL, DISPLAY IT

                ;-----  BACK SPACE FOUND

                        OR      DL,DL                   ; IS IT ALREADY AT START OF LINE
                        JE      U7                      ; SET_CURSOR
                        DEC     DX                      ; NO -- JUST MOVE IT BACK
                        JMP     U7                      ; SET_CURSOR

                ;-----  CARRIAGE RETURN FOUND
                U9:     MOV     DL,0                    ; MOVE TO FIRST COLUMN
                        JMP     U7                      ; SET_CURSOR

                ;-----  LINE FEED FOUND
                U10:    CMP     DH,25-1                 ; BOTTOM OF SCREEN
                        JNE     U6                      ; YES, SCROLL THE SCREEN
                        JMP     U1                      ; NO, JUST SET THE CURSOR

                ;-----  BELL FOUND
                U11:    MOV     CX,1331                 ; DIVISOR FOR 896 HZ TONE
                        MOV     BL,31                   ; SET COUNT FOR 31/64 SECOND FOR BEEP
                        CALL    BEEP                    ; SOUND THE POD BELL
                        JMP     U5                      ; TTY_RETURN
                WRITE_TTY       ENDP
                ;--------------------------------------------------------------------------
                ; LIGHT PEN                                                               :
                ;       THIS ROUTINE TESTS THE LIGHT PEN SWITCH AND THE LIGHT           :
                ;       PEN TRIGGER. IF BOTH ARE SET, THE LOCATION OF THE LIGHT         :
                ;       PEN IS DETERMINED. OTHERWISE, A RETURN WITH NO INFORMATION      :
                ;       IS MADE.                                                        :
                ; ON EXIT:                                                              :
                ;       (AH) = 0 IF NO LIGHT PEN INFORMATION IS AVAILABLE               :
                ;               BX,CX,DX ARE DESTROYED                                  :
                ;       (AH) = 1 IF LIGHT PEN IS AVAILABLE                              :
                ;               (DH,DL) = ROW,COLUMN OF CURRENT LIGHT PEN POSITION      :
                ;               (CH) = RASTER POSITION                                  :
                ;               (BX) = BEST GUESS AT PIXEL HORIZONTAL POSITION          :
                ;--------------------------------------------------------------------------
                        ASSUME  DS:DATA
0781 03 03 05 05 03 03  V1      DB      3,3,5,5,3,3,3,4         ; SUBTRACT_TABLE
     03 04

                ;-----  WAIT FOR LIGHT PEN TO BE DEPRESSED

0789                    READ_LPEN       PROC    NEAR
0789 B4 00                      MOV     AH,0                    ; SET NO LIGHT PEN RETURN CODE
078B 8B 16 0063 R               MOV     DX,@ADDR_6845           ; GET BASE ADDRESS OF 6845
078F 83 C2 06                   ADD     DX,6                    ; POINT TO STATUS REGISTER
0792 EC                         IN      AL,DX                   ; GET STATUS REGISTER
0793 A8 04                      TEST    AL,004H                 ; TEST LIGHT PEN SWITCH
0795 74 03                      JZ      V6_A                    ; GO IF YES
0797 E9 081C R                  JMP     V6                      ; NOT SET, RETURN

                ;-----  NOW TEST FOR LIGHT PEN TRIGGER

079A A8 02              V6_A:   TEST    AL,2                    ; TEST LIGHT PEN TRIGGER
079C 75 03                      JNZ     V7A                     ; RETURN WITHOUT RESETTING TRIGGER
079E E9 0826 R                  JMP     V7

                ;-----  TRIGGER HAS BEEN SET, READ THE VALUE IN

07A1 B4 10              V7A:    MOV     AH,16                   ; LIGHT PEN REGISTERS ON 6845

                ;-----  INPUT REGISTERS POINTED TO BY AH, AND CONVERT TO ROW COLUMN IN (DX)

07A3 8B 16 0063 R               MOV     DX,@ADDR_6845           ; ADDRESS REGISTER FOR 6845
07A7 8A C4                      MOV     AL,AH                   ; REGISTER TO READ
07A9 EE                         OUT     DX,AL                   ; SET IT UP
07AA EB 00                      JMP     $+2                     ; I/O DELAY
07AC 42                         INC     DX                      ; DATA REGISTER
07AD EC                         IN      AL,DX                   ; GET THE VALUE
07AE 8A E8                      MOV     CH,AL                   ; SAVE IN CX
07B0 4A                         DEC     DX                      ; ADDRESS REGISTER
07B1 FE C4                      INC     AH
07B3 8A C4                      MOV     AL,AH                   ; SECOND DATA REGISTER
07B5 EE                         OUT     DX,AL
07B6 42                         INC     DX                      ; POINT TO DATA REGISTER
07B7 EB 00                      JMP     $+2                     ; I/O DELAY
07B9 EC                         IN      AL,DX                   ; GET SECOND DATA VALUE
07BA 8A E5                      MOV     AH,CH                   ; AX HAS INPUT VALUE
```

```
                        PAGE
                        ;-----  AX HAS THE VALUE READ IN FROM THE 6845
                        
07BC 8A 1E 0049 R               MOV     BL,@CRT_MODE
07C0 2A FF                      SUB     BH,BH                   ; MODE VALUE TO BX
07C2 2E: 8A 9F 0781 R           MOV     BL,CS:V1[BX]            ; DETERMINE AMOUNT TO SUBTRACT
07C7 2B C3                      SUB     AX,BX                   ; TAKE IT AWAY
07C9 8B 1E 004E R               MOV     BX,@CRT_START
07CD D1 EB                      SHR     BX,1
07CF 2B C3                      SUB     AX,BX                   ; CONVERT TO CORRECT PAGE ORIGIN
07D1 79 02                      JNS     V2                      ; IF POSITIVE, DETERMINE MODE
07D3 2B C0                      SUB     AX,AX                   ; <0 PLAYS AS 0
                        
                        ;-----  DETERMINE MODE OF OPERATION
                        
07D5                    V2:                                     ; DETERMINE_MODE
07D5 B1 03                      MOV     CL,3                    ; SET #8 SHIFT COUNT
07D7 80 3E 0049 R 04            CMP     @CRT_MODE,4             ; DETERMINE IF GRAPHICS OR ALPHA
07DC 72 2A                      JB      V4                      ; ALPHA_PEN
07DE 80 3E 0049 R 07            CMP     @CRT_MODE,7
07E3 74 23                      JE      V4                      ; ALPHA_PEN
                        
                        ;-----  GRAPHICS MODE
                        
07E5 B2 28                      MOV     DL,40                   ; DIVISOR FOR GRAPHICS
07E7 F6 F2                      DIV     DL                      ; DETERMINE ROW(AL) AND COLUMN(AH)
                                                                ;  AL RANGE 0-99, AH RANGE 0-39
                        ;-----  DETERMINE GRAPHIC ROW POSITION
                        
07E9 8A E8                      MOV     CH,AL                   ; SAVE ROW VALUE IN CH
07EB 02 ED                      ADD     CH,CH                   ; *2 FOR EVEN/ODD FIELD
07ED 8A DC                      MOV     BL,AH                   ; COLUMN VALUE TO BX
07EF 2A FF                      SUB     BH,BH                   ; MULTIPLY BY 8 FOR MEDIUM RES
07F1 80 3E 0049 R 06            CMP     @CRT_MODE,6             ; DETERMINE MEDIUM OR HIGH RES
07F6 75 04                      JNE     V3                      ; NOT_HIGH_RES
07F8 B1 04                      MOV     CL,4                    ; SHIFT VALUE FOR HIGH RES
07FA D0 E4                      SAL     AH,1                    ; COLUMN VALUE TIMES 2 FOR HIGH RES
07FC                    V3:                                     ; NOT_HIGH_RES
07FC D3 E3                      SHL     BX,CL                   ; MULTIPLY *16 FOR HIGH RES
                        
                        ;-----  DETERMINE ALPHA CHAR POSITION
                        
07FE 8A D4                      MOV     DL,AH                   ; COLUMN VALUE FOR RETURN
0800 8A F0                      MOV     DH,AL                   ; ROW VALUE
0802 D0 EE                      SHR     DH,1                    ; DIVIDE BY 4
0804 D0 EE                      SHR     DH,1                    ;    FOR VALUE IN 0-24 RANGE
0806 EB 12                      JMP     SHORT V5                ; LIGHT_PEN_RETURN_SET
                        
                        ;-----  ALPHA MODE ON LIGHT PEN
                        
0808                    V4:                                     ; ALPHA_PEN
0808 F6 36 004A R               DIV     BYTE PTR @CRT_COLS      ; DETERMINE ROW,COLUMN VALUE
080C 8A F0                      MOV     DH,AL                   ; ROWS TO DH
080E 8A D4                      MOV     DL,AH                   ; COLS TO DL
0810 D2 E0                      SAL     AL,CL                   ; MULTIPLY ROWS * 8
0812 8A E8                      MOV     CH,AL                   ; GET RASTER VALUE TO RETURN REGISTER
0814 8A DC                      MOV     BL,AH                   ; COLUMN VALUE
0816 32 FF                      XOR     BH,BH                   ;  TO BX
0818 D3 E3                      SAL     BX,CL
081A                    V5:                                     ; LIGHT_PEN_RETURN_SET
081A B4 01                      MOV     AH,1                    ; INDICATE EVERY THING SET
081C                    V6:                                     ; LIGHT_PEN_RETURN
081C 52                         PUSH    DX                      ; SAVE RETURN VALUE (IN CASE)
081D 8B 16 0063 R               MOV     DX,@ADDR_6845           ;  GET BASE ADDRESS
0821 83 C2 07                   ADD     DX,7                    ; POINT TO RESET PARM
0824 EE                         OUT     DX,AL                   ; ADDRESS, NOT DATA, IS IMPORTANT
0825 5A                         POP     DX                      ; RECOVER VALUE
0826                    V7:                                     ; RETURN_NO_RESET
0826 5D                         POP     BP
0827 5F                         POP     DI
0828 5E                         POP     SI
0829 1F                         POP     DS                      ; DISCARD SAVED BX,CX,DX
082A 1F                         POP     DS
082B 1F                         POP     DS
082C 1F                         POP     DS
082D 07                         POP     ES
082E CF                         IRET
082F                    READ_LPEN       ENDP
082F                    CODE    ENDS
                                END
```

```
                        PAGE 118,121
                        TITLE BIOS ----- 06/10/85  BIOS ROUTINES
                        .286C
                        .LIST
0000                    CODE    SEGMENT BYTE PUBLIC

                                PUBLIC  EQUIPMENT_1
                                PUBLIC  MEMORY_SIZE_DET_1
                                PUBLIC  NMI_INT_1

                                EXTRN   C8042:NEAR              ; POST SEND 8042 COMMAND ROUTINE
                                EXTRN   CMOS_READ:NEAR          ; READ CMOS LOCATION ROUTINE
                                EXTRN   D1:NEAR                 ; "PARITY CHECK 1" MESSAGE
                                EXTRN   D2:NEAR                 ; "PARITY CHECK 2" MESSAGE
                                EXTRN   D2A:NEAR                ; "?????" UNKNOWN ADDRESS MESSAGE
                                EXTRN   DDS:NEAR                ; LOAD (DS) WITH DATA SEGMENT SELECTOR
                                EXTRN   OBF_42:NEAR             ; POST WAIT 8042 RESPONSE ROUTINE
                                EXTRN   PRT_HEX:NEAR            ; DISPLAY CHARACTER ROUTINE
                                EXTRN   PRT_SEG:NEAR            ; DISPLAY FIVE CHARACTER ADDRESS ROUTINE
                                EXTRN   P_MSG:NEAR              ; DISPLAY MESSAGE STRING ROUTINE

                        ;--- INT  12 H ----------------------------------------------------------------
                        ; MEMORY_SIZE_DETERMINE                                                      :
                        ;       THIS ROUTINE RETURNS THE AMOUNT OF MEMORY IN THE SYSTEM AS          :
                        ;       DETERMINED BY THE POST ROUTINES.  (UP TO 640K)                      :
                        ;       NOTE THAT THE SYSTEM MAY NOT BE ABLE TO USE I/O MEMORY UNLESS       :
                        ;       THERE IS A FULL COMPLEMENT OF 512K BYTES ON THE PLANAR.             :
                        ; INPUT                                                                     :
                        ;       NO REGISTERS                                                        :
                        ;       THE @MEMORY_SIZE VARIABLE IS SET DURING POWER ON DIAGNOSTICS        :
                        ;       ACCORDING TO THE FOLLOWING ASSUMPTIONS:                             :
                        ;                                                                           :
                        ;       1. CONFIGURATION RECORD IN NON-VOLATILE MEMORY EQUALS THE ACTUAL     :
                        ;          MEMORY SIZE INSTALLED.                                           :
                        ;                                                                           :
                        ;       2. ALL INSTALLED MEMORY IS FUNCTIONAL.  IF THE MEMORY TEST DURING   :
                        ;          POST INDICATES LESS, THEN THIS VALUE BECOMES THE DEFAULT.        :
                        ;          IF NON-VOLATILE MEMORY IS NOT VALID (NOT INITIALIZED OR BATTERY  :
                        ;          FAILURE) THEN ACTUAL MEMORY DETERMINED BECOMES THE DEFAULT.      :
                        ;                                                                           :
                        ;       3. ALL MEMORY FROM 0 TO 640K MUST BE CONTIGUOUS.                    :
                        ;                                                                           :
                        ; OUTPUT                                                                    :
                        ;       (AX) = NUMBER OF CONTIGUOUS 1K BLOCKS OF MEMORY                     :
                        ;-------------------------------------------------------------------------
                                ASSUME  CS:CODE,DS:DATA

0000                    MEMORY_SIZE_DET_1  PROC FAR
0000 FB                         STI                             ; INTERRUPTS BACK ON
0001 1E                         PUSH    DS                      ; SAVE SEGMENT
0002 E8 0000 E                  CALL    DDS                     ; ESTABLISH ADDRESSING
0005 A1 0013 R                  MOV     AX,@MEMORY_SIZE         ; GET VALUE
0008 1F                         POP     DS                      ; RECOVER SEGMENT
0009 CF                         IRET                            ; RETURN TO CALLER
000A                    MEMORY_SIZE_DET_1 ENDP

                        ;--- INT  11 H ----------------------------------------------------------------
                        ; EQUIPMENT DETERMINATION                                                   :
                        ;       THIS ROUTINE ATTEMPTS TO DETERMINE WHAT OPTIONAL                    :
                        ;       DEVICES ARE ATTACHED TO THE SYSTEM.                                 :
                        ; INPUT                                                                     :
                        ;       NO REGISTERS                                                        :
                        ;       THE @EQUIP_FLAG VARIABLE IS SET DURING THE POWER ON                 :
                        ;       DIAGNOSTICS USING THE FOLLOWING HARDWARE ASSUMPTIONS:               :
                        ;       PORT 03FA = INTERRUPT ID REGISTER OF 8250 (PRIMARY)                 :
                        ;            02FA = INTERRUPT ID REGISTER OF 8250 (SECONDARY)               :
                        ;              BITS 7-3 ARE ALWAYS 0                                        :
                        ;       PORT 0378 = OUTPUT PORT OF PRINTER (PRIMARY)                        :
                        ;            0278 = OUTPUT PORT OF PRINTER (SECONDARY)                      :
                        ;            03BC = OUTPUT PORT OF PRINTER (MONOCHROME-PRINTER)             :
                        ;  OUTPUT                                                                   :
                        ;       (AX) IS SET, BIT SIGNIFICANT, TO INDICATE ATTACHED I/O              :
                        ;       BIT 15,14 = NUMBER OF PRINTERS ATTACHED                             :
                        ;       BIT 13 = INTERNAL MODEM INSTALLED                                   :
                        ;       BIT 12 NOT USED                                                     :
                        ;       BIT 11,10,9 = NUMBER OF RS232 CARDS ATTACHED                        :
                        ;       BIT 8 = NOT USED                                                    :
                        ;       BIT 7,6 = NUMBER OF DISKETTE DRIVES                                 :
                        ;             00=1, 01=2 ONLY IF BIT 0 = 1                                  :
                        ;       BIT 5,4 = INITIAL VIDEO MODE                                        :
                        ;                 00 - UNUSED                                               :
                        ;                 01 - 40X25 BW USING COLOR CARD                            :
                        ;                 10 - 80X25 BW USING COLOR CARD                            :
                        ;                 11 - 80X25 BW USING BW CARD                               :
                        ;                                                                           :
                        ;       BIT 3 = NOT USED                                                    :
                        ;       BIT 2 = NOT USED                                                    :
                        ;       BIT 1 = MATH COPROCESSOR                                            :
                        ;       BIT 0 = 1 (IPL DISKETTE INSTALLED)                                  :
                        ;       NO OTHER REGISTERS AFFECTED                                         :
                        ;-------------------------------------------------------------------------

000A                    EQUIPMENT_1  PROC  FAR                  ; ENTRY POINT FOR ORG 0F84DH
000A FB                         STI                             ; INTERRUPTS BACK ON
000B 1E                         PUSH    DS                      ; SAVE SEGMENT REGISTER
000C E8 0000 E                  CALL    DDS                     ; ESTABLISH ADDRESSING
000F A1 0010 R                  MOV     AX,@EQUIP_FLAG          ; GET THE CURRENT SETTINGS
0012 1F                         POP     DS                      ; RECOVER SEGMENT
0013 CF                         IRET                            ; RETURN TO CALLER
0014                    EQUIPMENT_1     ENDP
```

```
                        PAGE
                        ;-- HARDWARE INT  02 H -- ( NMI LEVEL ) ------------------------------------
                        ; NON-MASKABLE INTERRUPT ROUTINE (REAL MODE)                                 :
                        ;       THIS ROUTINE WILL PRINT A "PARITY CHECK 1 OR 2" MESSAGE AND ATTEMPT  :
                        ;       TO FIND THE STORAGE LOCATION IN BASE 640K CONTAINING THE BAD PARITY. :
                        ;       IF FOUND, THE SEGMENT ADDRESS WILL BE PRINTED.  IF NO PARITY ERROR   :
                        ;       CAN BE FOUND (INTERMITTENT READ PROBLEM)  ?????  WILL BE DISPLAYED   :
                        ;       WHERE THE ADDRESS WOULD NORMALLY GO.                                 :
                        ;                                                                            :
                        ;       PARITY CHECK 1 = PLANAR BOARD MEMORY FAILURE.                        :
                        ;       PARITY CHECK 2 = OFF PLANAR BOARD MEMORY FAILURE.                    :
                        ;--------------------------------------------------------------------------

0014                    NMI_INT_1 PROC    NEAR
0014 50                         PUSH    AX                      ; SAVE ORIGINAL CONTENTS OF (AX)

0015 E4 61                      IN      AL,PORT_B               ; READ STATUS PORT
0017 A8 C0                      TEST    AL,PARITY_ERR           ; PARITY CHECK OR I/O CHECK ?
0019 75 07                      JNZ     NMI_1                   ; GO TO ERROR HALTS IF HARDWARE ERROR

001B B0 0F                      MOV     AL,CMOS_SHUT_DOWN       ; ELSE ?? - LEAVE NMI ON
001D E8 0000 E                  CALL    CMOS_READ               ; TOGGLE NMI USING COMMON READ ROUTINE
0020 58                         POP     AX                      ; RESTORE ORIGINAL CONTENTS OF (AX)
0021 CF                         IRET                            ; EXIT NMI HANDLER BACK TO PROGRAM


0022                    NMI_1:                                  ;       HARDWARE ERROR
0022 50                         PUSH    AX                      ; SAVE INITIAL CHECK MASK IN (AL)
0023 B0 8F                      MOV     AL,CMOS_SHUT_DOWN+NMI   ; MASK TRAP (NMI) INTERRUPTS OFF
0025 E8 0000 E                  CALL    CMOS_READ               ; OPEN STANDBY LATCH BEFORE POWER DOWN
0028 B0 AD                      MOV     AL,DIS_KBD              ; DISABLE THE KEYBOARD
002A E8 0000 E                  CALL    C8042                   ; SEND COMMAND TO ADAPTER
002D E8 0000 E                  CALL    DDS                     ; ADDRESS DATA SEGMENT
0030 B4 00                      MOV     AH,0                    ; INITIALIZE AND SET MODE FOR VIDEO
0032 A0 0049 R                  MOV     AL,@CRT_MODE            ; GET CURRENT MODE
0035 CD 10                      INT     10H                     ; CALL VIDEO_IO TO CLEAR SCREEN

                        ;-----  DISPLAY "PARITY CHECK ?" ERROR MESSAGES

0037 58                         POP     AX                      ; RECOVER INITIAL CHECK STATUS
0038 BE 0000 E                  MOV     SI,OFFSET D1            ; PLANAR ERROR, ADDRESS "PARITY CHECK 1"
003B A8 80                      TEST    AL,PARITY_CHECK         ; CHECK FOR PLANAR ERROR
003D 74 05                      JZ      NMI_2                   ; SKIP IF NOT

003F 50                         PUSH    AX                      ; SAVE STATUS
0040 E8 0000 E                  CALL    P_MSG                   ; DISPLAY "PARITY CHECK 1" MESSAGE
0043 58                         POP     AX                      ; AND RECOVER STATUS
0044                    NMI_2:
0044 BE 0000 E                  MOV     SI,OFFSET D2            ; ADDRESS OF "PARITY CHECK 2" MESSAGE
0047 A8 40                      TEST    AL,IO_CHECK             ; I/O PARITY CHECK ?
0049 74 03                      JZ      NMI_3                   ; SKIP IF CORRECT ERROR DISPLAYED
004B E8 0000 E                  CALL    P_MSG                   ; DISPLAY "PARITY CHECK 2" ERROR

                        ;-----  TEST FOR HOT NMI ON PLANAR PARITY LINE

004E                    NMI_3:
004E E4 61                      IN      AL,PORT_B
0050 0C 0C                      OR      AL,RAM_PAR_OFF          ; TOGGLE PARITY CHECK ENABLES
0052 E6 61                      OUT     PORT_B,AL
0054 24 F3                      AND     AL,RAM_PAR_ON           ; TO CLEAR THE PENDING CHECK
0056 E6 61                      OUT     PORT_B,AL

0058 FC                         CLD                             ; SET DIRECTION FLAG TO INCREMENT
0059 2B D2                      SUB     DX,DX                   ; POINT (DX) AT START OF REAL MEMORY
005B 2B F6                      SUB     SI,SI                   ; SET (SI) TO START OF (DS:)
005D E4 61                      IN      AL,PORT_B               ; READ CURRENT PARITY CHECK LATCH
005F A8 C0                      TEST    AL,PARITY_ERR           ; CHECK FOR HOT NMI SOURCE
0061 75 19                      JNZ     NMI_5                   ; SKIP IF ERROR NOT RESET (DISPLAY ???)

                        ;-----  SEE IF LOCATION THAT CAUSED PARITY CHECK CAN BE FOUND IN BASE MEMORY

0063 8B 1E 0013 R               MOV     BX,@MEMORY_SIZE         ; GET BASE MEMORY SIZE WORD
0067                    NMI_4:
0067 8E DA                      MOV     DS,DX                   ; POINT TO 64K SEGMENT
0069 B9 8000                    MOV     CX,4000H*2              ; SET WORD COUNT FOR 64 KB SCAN
006C F3/ AD                     REP     LODSW                   ; READ 64 KB OF MEMORY
006E E4 61                      IN      AL,PORT_B               ; READ PARITY CHECK LATCHES
0070 A8 C0                      TEST    AL,PARITY_ERR           ; CHECK FOR ANY PARITY ERROR PENDING
0072 75 10                      JNZ     NMI_6                   ; GO PRINT SEGMENT ADDRESS IF ERROR

0074 80 C6 10                   ADD     DH,010H                 ; POINT TO NEXT 64K BLOCK
0077 83 EB 40                   SUB     BX,16D*4                ; DECREMENT COUNT OF 1024 BYTE SEGMENTS
007A 77 EB                      JA      NMI_4                   ; LOOP TILL ALL 64K SEGMENTS DONE
007C                    NMI_5:
007C BE 0000 E                  MOV     SI,OFFSET D2A           ; PRINT ROW OF ????? IF PARITY
007F E8 0000 E                  CALL    P_MSG                   ; CHECK COULD NOT BE RE-CREATED
0082 FA                         CLI
0083 F4                         HLT                             ; HALT SYSTEM

0084                    NMI_6:
0084 E8 0000 E                  CALL    PRT_SEG                 ; PRINT SEGMENT VALUE (IN DX)
0087 B0 28                      MOV     AL,'('                  ; PRINT (S)
0089 E8 0000 E                  CALL    PRT_HEX
008C B0 53                      MOV     AL,'S'
008E E8 0000 E                  CALL    PRT_HEX
0091 B0 29                      MOV     AL,')'
0093 E8 0000 E                  CALL    PRT_HEX
0096 FA                         CLI                             ; HALT SYSTEM
0097 F4                         HLT

0098                    NMI_INT_1 ENDP

0098                    CODE    ENDS
                        END
```

SECTION 5

```
                        PAGE 118,121
                        TITLE BIOS1 ---- 06/10/85  INTERRUPT 15H BIOS ROUTINES
                        .286C
                        .LIST
0000                    CODE    SEGMENT BYTE PUBLIC

                                PUBLIC  CASSETTE_IO_1
                                PUBLIC  GATE_A20
                                PUBLIC  SHUT9

                                EXTRN   CMOS_READ:NEAR          ; READ CMOS LOCATION ROUTINE
                                EXTRN   CMOS_WRITE:NEAR         ; WRITE CMOS LOCATION ROUTINE
                                EXTRN   CONF_TBL:NEAR           ; SYSTEM/BIOS CONFIGURATION TABLE
                                EXTRN   DDS:NEAR                ; LOAD (DS) WITH DATA SEGMENT SELECTOR
                                EXTRN   PROC_SHUTDOWN:NEAR      ; 80286 HARDWARE RESET ROUTINE

                        ;--- INT 15 H ----------------------------------------------------------------
                        ;    INPUT - CASSETTE I/O FUNCTIONS                                         :
                        ;                                                                           :
                        ;         (AH) = 00H                                                        :
                        ;         (AH) = 01H                                                        :
                        ;         (AH) = 02H                                                        :
                        ;         (AH) = 03H                                                        :
                        ;       RETURNS FOR THESE FUNCTIONS ALWAYS (AH) = 86H, CY = 1)              :
                        ;       IF CASSETTE PORT NOT PRESENT                                        :
                        ;-------------------------------------------------------------------------- :
                        ;    INPUT - UNUSED FUNCTIONS                                               :
                        ;        (AH) = 04H THROUGH 7FH                                             :
                        ;      RETURNS FOR THESE FUNCTIONS ALWAYS (AH) = 86H, CY = 1)               :
                        ;              (UNLESS INTERCEPTED BY SYSTEM HANDLERS)                      :
                        ;             NOTE: THE KEYBOARD INTERRUPT HANDLER INTERRUPTS WITH AH=4FH    :
                        ;-------------------------------------------------------------------------- :
                        ; EXTENSIONS                                                                :
                        ;         (AH) = 80H   DEVICE OPEN                                          :
                        ;                        (BX) = DEVICE ID                                   :
                        ;                        (CX) = PROCESS ID                                  :
                        ;                                                                           :
                        ;         (AH) = 81H   DEVICE CLOSE                                         :
                        ;                        (BX) = DEVICE ID                                   :
                        ;                        (CX) = PROCESS ID                                  :
                        ;                                                                           :
                        ;         (AH) = 82H   PROGRAM TERMINATION                                  :
                        ;                        (BX) = DEVICE ID                                   :
                        ;                                                                           :
                        ;         (AH) = 83H   EVENT WAIT                                           :
                        ;                                                                           :
                        ;                          (AL) = 00H SET INTERVAL                          :
                        ;                          (ES:BX) POINTER TO A BYTE IN CALLERS MEMORY      :
                        ;                                  THAT WILL HAVE THE HIGH ORDER BIT SET    :
                        ;                                  AS SOON AS POSSIBLE AFTER THE INTERVAL   :
                        ;                                  EXPIRES.                                 :
                        ;                           (CX,DX) NUMBER OF MICROSECONDS TO ELAPSE BEFORE :
                        ;                                  POSTING.                                 :
                        ;                          (AL) = 01H CANCEL                                :
                        ;                                                                           :
                        ;                   RETURNS: CARRY IF AL NOT = 00H OR 01H                   :
                        ;                             OR IF FUNCTION AL=0 ALREADY BUSY              :
                        ;                                                                           :
                        ;         (AH) = 84H   JOYSTICK SUPPORT                                     :
                        ;                        (DX) = 00H - READ THE CURRENT SWITCH SETTINGS       :
                        ;                                  RETURNS AL = SWITCH SETTINGS (BITS 7-4)  :
                        ;                        (DX) = 01H - READ THE RESISTIVE INPUTS              :
                        ;                                  RETURNS AX = A(x) VALUE                  :
                        ;                                          BX = A(y) VALUE                  :
                        ;                                          CX = B(x) VALUE                  :
                        ;                                          DX = B(y) VALUE                  :
                        ;                                                                           :
                        ;         (AH) = 85H   SYSTEM REQUEST KEY PRESSED                           :
                        ;                        (AL) = 00H MAKE OF KEY                             :
                        ;                        (AL) = 01H BREAK OF KEY                            :
                        ;                                                                           :
                        ;         (AH) = 86H   WAIT                                                 :
                        ;                        (CX,DX) NUMBER OF MICROSECONDS TO ELAPSE BEFORE    :
                        ;                                RETURN TO CALLER                           :
                        ;                                                                           :
                        ;         (AH) = 87H   MOVE BLOCK                                           :
                        ;                        (CX)    NUMBER OF WORDS TO MOVE                    :
                        ;                        (ES:SI)  POINTER TO DESCRIPTOR TABLE               :
                        ;                                                                           :
                        ;         (AH) = 88H   EXTENDED MEMORY SIZE DETERMINE                       :
                        ;                                                                           :
                        ;         (AH) = 89H   PROCESSOR TO VIRTUAL MODE                            :
                        ;                                                                           :
                        ;         (AH) = 90H    DEVICE BUSY LOOP                                    :
                        ;                        (AL)   SEE TYPE CODE                               :
                        ;                                                                           :
                        ;         (AH) = 91H   INTERRUPT COMPLETE FLAG SET                          :
                        ;                        (AL)   TYPE CODE                                   :
                        ;                        00H -> 7FH                                         :
                        ;                               SERIALLY REUSABLE DEVICES                   :
                        ;                               OPERATING SYSTEM MUST SERIALIZE ACCESS      :
                        ;                        80H -> BFH                                         :
                        ;                               REENTRANT DEVICES; ES:BX IS USED TO         :
                        ;                               DISTINGUISH DIFFERENT CALLS (MULTIPLE I/O   :
                        ;                               CALLS ARE ALLOWED SIMULTANEOUSLY)           :
                        ;                        C0H -> FFH                                         :
                        ;                               WAIT ONLY CALLS --  THERE IS NO             :
                        ;                               COMPLEMENTARY 'POST' FOR THESE WAITS.       :
                        ;                               THESE ARE TIMEOUT ONLY.  TIMES ARE          :
                        ;                               FUNCTION NUMBER DEPENDENT.                  :
                        ;                                                                           :
                        ;                        TYPE  DESCRIPTION            TIMEOUT               :
                        ;                                                                           :
                        ;                        00H = DISK                   YES                   :
                        ;                        01H = DISKETTE               YES                   :
                        ;                        02H = KEYBOARD               NO                    :
                        ;                        80H = NETWORK                NO                    :
                        ;                          ES:BX --> NCB                                    :
                        ;                        FDH = DISKETTE MOTOR START   YES                   :
                        ;                        FEH = PRINTER                YES                   :
                        ;                                                                           :
```

```
                        PAGE
                        ;       (AH) = C0H   RETURN CONFIGURATION PARAMETERS POINTER                    ;
                        ;                      RETURNS                                                  ;
                        ;                        (AH) = 00H AND CY= 0 (IF PRESENT ELSE 86 AND CY= 1)    ;
                        ;                        (ES:BX) = PARAMETER TABLE ADDRESS POINTER              ;
                        ;                                  WHERE:                                       ;
                        ;                                                                               ;
                        ;                   DW   8          LENGTH OF FOLLOWING TABLE                   ;
                        ;                   DB   MODEL_BYTE  SYSTEM MODEL BYTE                          ;
                        ;                   DB   TYPE_BYTE   SYSTEM MODEL TYPE BYTE                     ;
                        ;                   DB   BIOS_LEVEL  BIOS REVISION LEVEL                        ;
                        ;                   DB   ?           10000000 = DMA CHANNEL 3 USE BY BIOS       ;
                        ;                                    01000000 = CASCADED INTERRUPT LEVEL 2      ;
                        ;                                    00100000 = REAL TIME CLOCK AVAILABLE       ;
                        ;                                    00010000 = KEYBOARD SCAN CODE HOOK 1AH     ;
                        ;                   DB   0           RESERVED                                   ;
                        ;                   DB   0           RESERVED                                   ;
                        ;                   DB   0           RESERVED                                   ;
                        ;                   DB   0           RESERVED                                   ;
                        ;                                                                               ;
                        ;-------------------------------------------------------------------------------;

                                ASSUME  CS:CODE

0000                    CASSETTE_IO_1   PROC    FAR
0000 FB                         STI                             ; ENABLE INTERRUPTS
0001 80 FC 80                   CMP     AH,080H                 ; CHECK FOR RANGE
0004 72 4C                      JB      C1                      ; RETURN IF 00-7FH
0006 80 FC C0                   CMP     AH,0C0H                 ; CHECK FOR CONFIGURATION PARAMETERS
0009 74 4F                      JE      CONF_PARMS
000B 80 EC 80                   SUB     AH,080H                 ; BASE ON 0
000E 74 48                      JZ      DEV_OPEN                ; DEVICE OPEN
0010 FE CC                      DEC     AH
0012 74 44                      JZ      DEV_CLOSE               ; DEVICE CLOSE
0014 FE CC                      DEC     AH
0016 74 40                      JZ      PROG_TERM               ; PROGRAM TERMINATION
0018 FE CC                      DEC     AH
001A 74 47                      JZ      EVENT_WAIT              ; EVENT WAIT
001C FE CC                      DEC     AH
001E 75 03                      JNZ     NOT_JOYSTICK
0020 E9 00DB R                  JMP     JOY_STICK               ; JOYSTICK BIOS
0023                    NOT_JOYSTICK:
0023 FE CC                      DEC     AH
0025 74 31                      JZ      SYS_REQ                 ; SYSTEM REQUEST KEY
0027 FE CC                      DEC     AH
0029 74 07                      JZ      C1_A                    ; WAIT
002B FE CC                      DEC     AH
002D 75 06                      JNZ     C1_B
002F E9 01D6 R                  JMP     BLOCKMOVE               ; MOVE BLOCK

0032 E9 0175 R          C1_A:   JMP     WAIT                    ; WAIT

0035 FE CC              C1_B:   DEC     AH

0037 75 03                      JNZ     C1_C
0039 E9 03FC R                  JMP     EXT_MEMORY              ; GO GET THE EXTENDED MEMORY

003C FE CC              C1_C:   DEC     AH
003E 75 03                      JNZ     C1_D                    ; CHECK FOR FUNCTION 89H
0040 E9 0408 R                  JMP     SET_VMODE               ; SWAP TO VIRTUAL MODE

0043 80 EC 07           C1_D:   SUB     AH,7                    ; CHECK FOR FUNCTION 90H
0046 75 03                      JNZ     C1_E                    ; GO IF NOT
0048 E9 0491 R                  JMP     DEVICE_BUSY

004B FE CC              C1_E:   DEC     AH                      ; CHECK FOR FUNCTION 91H
004D 75 03                      JNZ     C1                      ; GO IF NOT
004F E9 0495 R                  JMP     INT_COMPLETE

0052 B4 86              C1:     MOV     AH,86H                  ; SET BAD COMMAND
0054 F9                         STC                             ; SET CARRY FLAG ON
0055                    C1_F:
0055 CA 0002                    RET     2                       ; FAR RETURN EXIT FROM ROUTINES


0058                    DEV_OPEN:                               ; NULL HANDLERS

0058                    DEV_CLOSE:

0058                    PROG_TERM:

0058                    SYS_REQ:
0058 EB FB                      JMP     C1_F                    ; RETURN
005A                    CASSETTE_IO_1   ENDP

005A                    CONF_PARMS      PROC    NEAR
005A 0E                         PUSH    CS                      ; GET CODE SEGMENT
005B 07                         POP     ES                      ; PLACE IN SELECTOR POINTER
005C BB 0000 E                  MOV     BX,OFFSET CONF_TBL      ; GET OFFSET OF PARAMETER TABLE
005F 32 E4                      XOR     AH,AH                   ; CLEAR AH AND SET CARRY OFF
0061 EB F2                      JMP     C1_F                    ; EXIT THROUGH COMMON RETURN
0063                    CONF_PARMS      ENDP

0063                    EVENT_WAIT      PROC    NEAR
                                ASSUME  DS:DATA
0063 1E                         PUSH    DS                      ; SAVE
0064 E8 0000 E                  CALL    DDS
0067 0A C0                      OR      AL,AL
0069 74 08                      JZ      EVENT_WAIT_2            ; GO IF ZERO
006B FE C8                      DEC     AL                      ; CHECK IF 1
006D 74 48                      JZ      EVENT_WAIT_3
006F 1F                         POP     DS                      ; RESTORE DATA SEGMENT
0070 F9                         STC                             ; SET CARRY
0071 EB E2                      JMP     C1_F                    ; EXIT

0073                    EVENT_WAIT_2:
0073 FA                         CLI                             ; NO INTERRUPTS ALLOWED
0074 F6 06 00A0 R 01            TEST    @RTC_WAIT_FLAG,01       ; CHECK FOR FUNCTION ACTIVE
0079 74 05                      JZ      EVENT_WAIT_1
007B FB                         STI                             ; ENABLE INTERRUPTS
007C 1F                         POP     DS
007D F9                         STC                             ; SET ERROR
007E EB D5                      JMP     C1_F                    ; RETURN
```

SECTION 5

```
0080                    EVENT_WAIT_1:
0080 E4 A1                      IN      AL,INTB01               ; ENSURE INTERRUPT UNMASKED
0082 EB 00                      JMP     $+2
0084 24 FE                      AND     AL,0FEH
0086 E6 A1                      OUT     INTB01,AL
0088 8C 06 009A R               MOV     @USER_FLAG_SEG,ES       ; SET UP TRANSFER TABLE
008C 89 1E 0098 R               MOV     @USER_FLAG,BX
0090 89 0E 009E R               MOV     @RTC_HIGH,CX
0094 89 16 009C R               MOV     @RTC_LOW,DX
0098 C6 06 00A0 R 01            MOV     @RTC_WAIT_FLAG,01       ; SET ON FUNCTION ACTIVE SWITCH
009D B0 0B                      MOV     AL,CMOS_REG_B           ; ENABLE PIE
009F E8 0000 E                  CALL    CMOS_READ               ; READ CMOS LOCATION
00A2 24 7F                      AND     AL,07FH                 ; CLEAR SET
00A4 0C 40                      OR      AL,040H                 ; ENABLE PIE
00A6 50                         PUSH    AX                      ; SAVE AH
00A7 8A E0                      MOV     AH,AL                   ; PLACE DATA INTO DATA REGISTER
00A9 B0 0B                      MOV     AL,CMOS_REG_B           ; ADDRESS ALARM REGISTER
00AB E8 0000 E                  CALL    CMOS_WRITE              ; PLACE DATA IN AH INTO ALARM REGISTER
00AE 58                         POP     AX                      ; RESTORE AH
00AF 1F                         POP     DS
00B0 FB                         STI                             ; ENABLE INTERRUPTS
00B1 F8                         CLC                             ; CLEAR CARRY
00B2 EB A1                      JMP     C1_F

                        ;-----  CANCEL

00B4                    EVENT_WAIT_3:
00B4 FA                         CLI                             ; DISABLE INTERRUPTS
00B5 F6 06 00A0 R 02            TEST    @RTC_WAIT_FLAG,02H      ; CHECK FOR "WAIT" IN PROGRESS
00BA 74 05                      JZ      EVENT_WAIT_4            ; SKIP TO CANCEL CURRENT "EVENT_WAIT"

00BC FB                         STI                             ; ENABLE INTERRUPTS
00BD 1F                         POP     DS                      ; CLEAR STACK
00BE F9                         STC                             ; AND SET CARRY FLAG FOR ERROR REQUEST
00BF EB 94                      JMP     C1_F                    ; EXIT

00C1                    EVENT_WAIT_4:
00C1 50                         PUSH    AX                      ; SAVE (WITH INTERRUPTS DISABLED)
00C2 B8 0B0B                    MOV     AX,X*CMOS_REG_B         ; TURN OFF PIE
00C5 E8 0000 E                  CALL    CMOS_READ               ; GET ALARM REGISTER
00C8 24 BF                      AND     AL,0BFH                 ; CLEAR PIE
00CA 86 E0                      XCHG    AH,AL                   ; PLACE INTO WRITE REGISTER
00CC E8 0000 E                  CALL    CMOS_WRITE              ; WRITE BACK TO ALARM REGISTER
00CF 58                         POP     AX                      ; RESTORE AH
00D0 C6 06 00A0 R 00            MOV     @RTC_WAIT_FLAG,0        ; SET FUNCTION ACTIVE FLAG OFF
00D5 FB                         STI                             ; ENABLE INTERRUPTS
00D6 1F                         POP     DS                      ; RESTORE DATA SEGMENT
00D7 F8                         CLC                             ; SET CARRY OFF
00D8 E9 0056 R                  JMP     C1_F                    ; RETURN

00DB                    EVENT_WAIT      ENDP
                        ;--- JOY_STICK ------------------------------------------------
                        ;       THIS ROUTINE WILL READ THE JOYSTICK PORT                 :
                        ;                                                               :
                        ;       INPUT                                                   :
                        ;       (DX)=0 READ THE CURRENT SWITCHES                        :
                        ;              RETURNS (AL)= SWITCH SETTINGS IN BITS 7-4        :
                        ;                                                               :
                        ;       (DX)=1  READ THE RESISTIVE INPUTS                       :
                        ;               RETURNS (AX)=A(x) VALUE                         :
                        ;                       (BX)=A(y) VALUE                         :
                        ;                       (CX)=B(x) VALUE                         :
                        ;                       (DX)=B(y) VALUE                         :
                        ;                                                               :
                        ;       CY FLAG ON IF NO ADAPTER CARD OR INVALID CALL           :
                        ;---------------------------------------------------------------

00DB                    JOY_STICK       PROC    NEAR
00DB FB                         STI                             ; INTERRUPTS BACK ON
00DC 8B C2                      MOV     AX,DX                   ; GET SUB FUNCTION CODE
00DE BA 0201                    MOV     DX,201H                 ; ADDRESS OF PORT
00E1 0A C0                      OR      AL,AL
00E3 74 0B                      JZ      JOY_2                   ; READ SWITCHES
00E5 FE C8                      DEC     AL
00E7 74 0C                      JZ      JOY_3                   ; READ RESISTIVE INPUTS
00E9 E9 0052 R                  JMP     C1                      ; GO TO ERROR RETURN
00EC                    JOY_1:
00EC FB                         STI
00ED E9 0056 R                  JMP     C1_F                    ; GO TO COMMON RETURN

00F0                    JOY_2:
00F0 EC                         IN      AL,DX
00F1 24 F0                      AND     AL,0F0H                 ; STRIP UNWANTED BITS OFF
00F3 EB F7                      JMP     JOY_1                   ; FINISHED

00F5                    JOY_3:
00F5 B3 01                      MOV     BL,1
00F7 E8 0113 R                  CALL    TEST_CORD
00FA 51                         PUSH    CX                      ; SAVE A(X) VALUE
00FB B3 02                      MOV     BL,2
00FD E8 0113 R                  CALL    TEST_CORD
0100 51                         PUSH    CX                      ; SAVE A(Y) VALUE
0101 B3 04                      MOV     BL,4
0103 E8 0113 R                  CALL    TEST_CORD
0106 51                         PUSH    CX                      ; SAVE B(X) VALUE
0107 B3 08                      MOV     BL,8
0109 E8 0113 R                  CALL    TEST_CORD
010C 8B D1                      MOV     DX,CX                   ; SAVE B(Y) VALUE
010E 59                         POP     CX                      ; GET B(X) VALUE
010F 5B                         POP     BX                      ; GET A(Y) VALUE
0110 58                         POP     AX                      ; GET A(X) VALUE
0111 EB D9                      JMP     JOY_1                   ; FINISHED - RETURN

0113                    TEST_CORD       PROC  NEAR
0113 52                         PUSH    DX                      ; SAVE
0114 FA                         CLI                             ; BLOCK INTERRUPTS WHILE READING
0115 B0 00                      MOV     AL,0                    ; SET UP TO LATCH TIMER 0
0117 E6 43                      OUT     TIMER+3,AL
0119 EB 00                      JMP     $+2
011B E4 40                      IN      AL,TIMER                ; READ LOW BYTE OF TIMER 0
011D EB 00                      JMP     $+2
011F 8A E0                      MOV     AH,AL
0121 E4 40                      IN      AL,TIMER                ; READ HIGH BYTE OF TIMER 0
0123 86 E0                      XCHG    AH,AL                   ; REARRANGE TO HIGH,LOW
```

```
0125 50                         PUSH    AX                      ; SAVE
0126 B9 04FF                    MOV     CX,4FFH                 ; SET COUNT
0129 EE                         OUT     DX,AL                   ; FIRE TIMER
012A EB 00                      JMP     $+2
012C                    TEST_CORD_1:
012C EC                         IN      AL,DX                   ; READ VALUES
012D 84 C3                      TEST    AL,BL                   ; HAS PULSE ENDED?
012F E0 FB                      LOOPNZ  TEST_CORD_1
0131 83 F9 00                   CMP     CX,0
0134 59                         POP     CX                      ; ORIGINAL COUNT
0135 75 04                      JNZ     SHORT TEST_CORD_2
0137 2B C9                      SUB     CX,CX                   ; SET 0 COUNT FOR RETURN
0139 EB 26                      JMP     SHORT TEST_CORD_3       ; EXIT WITH COUNT = 0
013B                    TEST_CORD_2:
013B B0 00                      MOV     AL,0                    ; SET UP TO LATCH TIMER 0
013D E6 43                      OUT     TIMER+3,AL
013F EB 00                      JMP     $+2
0141 E4 40                      IN      AL,TIMER                ; READ LOW BYTE OF TIMER 0
0143 8A E0                      MOV     AH,AL
0145 EB 00                      JMP     $+2
0147 E4 40                      IN      AL,TIMER                ; READ HIGH BYTE OF TIMER 0
0149 86 E0                      XCHG    AH,AL                   ; REARRANGE TO HIGH,LOW

014B 3B C8                      CMP     CX,AX                   ; CHECK FOR COUNTER WRAP
014D 73 0B                      JAE     TEST_CORD_4             ; GO IF NO
014F 52                         PUSH    DX
0150 BA FFFF                    MOV     DX,-1

0153 2B D0                      SUB     DX,AX                   ; ADJUST FOR WRAP
0155 03 CA                      ADD     CX,DX
0157 5A                         POP     DX
0158 EB 02                      JMP     SHORT TEST_CORD_5

015A                    TEST_CORD_4:
015A 2B C8                      SUB     CX,AX
015C                    TEST_CORD_5:
015C 81 E1 1FF0                 AND     CX,1FF0H                ; ADJUST
0160 C1 E9 04                   SHR     CX,4

0163                    TEST_CORD_3:
0163 FB                         STI                             ; INTERRUPTS BACK ON
0164 BA 0201                    MOV     DX,201H                 ; FLUSH OTHER INPUTS
0167 51                         PUSH    CX
0168 50                         PUSH    AX
0169 B9 04FF                    MOV     CX,4FFH                 ; COUNT
016C                    TEST_CORD_6:
016C EC                         IN      AL,DX
016D A8 0F                      TEST    AL,0FH
016F E0 FB                      LOOPNZ  TEST_CORD_6

0171 58                         POP     AX
0172 59                         POP     CX
0173 5A                         POP     DX                      ; SET COUNT

0174 C3                         RET                             ; RETURN

0175                    TEST_CORD       ENDP
0175                    JOY_STICK       ENDP

0175                    WAIT    PROC    NEAR
0175 1E                         PUSH    DS                      ; SAVE
0176 E8 0000 E                  CALL    DDS
0179 FA                         CLI                             ; NO INTERRUPTS ALLOWED
017A F6 06 00A0 R 01            TEST    @RTC_WAIT_FLAG,01       ; TEST FOR FUNCTION ACTIVE
017F 74 06                      JZ      WAIT_1
0181 FB                         STI                             ; ENABLE INTERRUPTS PRIOR TO RETURN
0182 1F                         POP     DS
0183 F9                         STC                             ; SET ERROR
0184 E9 0055 R                  JMP     C1_F                    ; RETURN
0187                    WAIT_1: IN      AL,INTB01               ; ENSURE INTERRUPT UNMASKED
0187 E4 A1
0189 EB 00                      JMP     $+2
018B 24 FE                      AND     AL,0FEH
018D E6 A1                      OUT     INTB01,AL
018F 8C 1E 009A R               MOV     @USER_FLAG_SEG,DS       ; SET UP TRANSFER TABLE
0193 C7 06 0098 R 00A0 R        MOV     @USER_FLAG;OFFSET @RTC_WAIT_FLAG
0199 89 0E 009E R               MOV     @RTC_HIGH,CX
019D 89 16 009C R               MOV     @RTC_LOW,DX
01A1 C6 06 00A0 R 03            MOV     @RTC_WAIT_FLAG,03       ; SET ON "WAIT" FUNCTION ACTIVE SWITCHES
01A6 50                         PUSH    AX                      ; SAVE (AH)
01A7 B8 0B0B                    MOV     AX,X*CMOS_REG_B         ; ENABLE PIE
01AA E8 0000 E                  CALL    CMOS_READ               ; READ ALARM BYTE
01AD 24 7F                      AND     AL,07FH                 ; CLEAR SIT BIT
01AF 0C 40                      OR      AL,040H                 ; ENABLE PIE BIT
01B1 86 E0                      XCHG    AH,AL                   ; DATA TO WORK REGISTER
01B3 E8 0000 E                  CALL    CMOS_WRITE              ; WRITE NEW ALARM BYTE
01B6 58                         POP     AX                      ; RESTORE (AH)

                        ;-----  WAIT TILL RTC TIMEOUT POSTED    (WITH ERROR TIMEOUT)

01B7 FB                         STI                             ; ENABLE INTERRUPTS
01B8 51                         PUSH    CX
01B9 52                         PUSH    DX                      ; SAVE CALLERS PARAMETERS
01BA 87 D1                      XCHG    DX,CX                   ; SWAP COUNT WORK REGISTERS
01BC                    WAIT_2: TEST    @RTC_WAIT_FLAG,080H     ; CHECK FOR END OF WAIT - CLEAR CARRY
01BC F6 06 00A0 R 80            LOOPZ   WAIT_2                  ; DECREMENT TIMEOUT DELAY TILL WAIT END
01C1 E1 F9                      JNZ     WAIT_9                  ; EXIT IF RTC TIMER WAIT ENDED FLAG SET
01C3 75 03                      SUB     DX,1                    ; DECREMENT ERROR TIMEOUT COUNTER
01C5 83 EA 01                   JNC     WAIT_2                  ; LOOP TILL COUNTERS TIMEOUT
01C8 73 F2                      MOV     @RTC_WAIT_FLAG,0        ; SET FUNCTION INACTIVE
01CA                    WAIT_9: POP     DX
01CA C6 06 00A0 R 00            POP     CX                      ; RESTORE CALLERS PARAMETERS
01CF 5A                         POP     DS
01D0 59                         CLC                             ; CLEAR CARRY FLAG
01D1 1F                         JMP     C1_F
01D2 F8
01D3 E9 0055 R
01D6                    WAIT    ENDP
```

SECTION 5

```
                        PAGE
                        ;--- INT  15 H -- ( FUNCTION 87 H - BLOCK MOVE ) ----------------------------------
                        ;
                        ;       THIS BIOS FUNCTION PROVIDES A MEANS FOR A REAL MODE PROGRAM OR SYSTEM
                        ;       TO TRANSFER A BLOCK OF STORAGE TO AND FROM STORAGE ABOVE THE 1 MEG
                        ;       ADDRESS RANGE IN PROTECTED MODE SPACE BY SWITCHING TO PROTECTED MODE.
                        ;
                        ; ENTRY:
                        ;       (AH) =  87H (FUNCTION CALL) - BLOCK MOVE.
                        ;       (CX) =  WORD COUNT OF STORAGE BLOCK TO BE MOVED.
                        ;               NOTE: MAX COUNT = 8000H FOR 32K WORDS (65K BYTES)
                        ;       ES:SI = LOCATION OF A GDT TABLE BUILT BY ROUTINE USING THIS FUNCTION.
                        ;
                        ;       (ES:SI) POINTS TO A DESCRIPTOR TABLE (GDT) BUILT BEFORE INTERRUPTING
                        ;       TO THIS FUNCTION.  THE DESCRIPTORS ARE USE TO PERFORM THE BLOCK
                        ;       MOVE IN THE PROTECTED MODE.  THE SOURCE AND TARGET DESCRIPTORS
                        ;       BUILT BY THE USER MUST HAVE A SEGMENT LENGTH = 2 * CX-1 OR GREATER,
                        ;       THE DATA ACCESS RIGHTS BYTE MUST BE SET TO CPL0-R/W (93H).  THE
                        ;       24 BIT ADDRESS (BYTE HI, WORD LOW) MUST BE SET TO THE TARGET/SOURCE.
                        ;
                        ;       *** NO INTERRUPTS ARE ALLOWED DURING TRANSFER.  LARGE BLOCK MOVES
                        ;           MAY CAUSE LOST INTERRUPTS.
                        ;
                        ; EXIT:
                        ;       (AH) = 00H  IF SUCCESSFUL
                        ;       (AH) = 01H  IF MEMORY PARITY (PARITY ERROR REGISTERS ARE CLEARED)
                        ;       (AH) = 02H  IF ANY OTHER EXCEPTION INTERRUPT ERROR OCCURRED
                        ;       (AH) = 03H  IF GATE ADDRESS LINE 20 FAILED
                        ;                   ALL REGISTERS ARE RESTORED EXCEPT (AH).
                        ;
                        ;       IF SUCCESSFUL - CARRY FLAG = 0
                        ;       IF ERROR ------ CARRY FLAG = 1
                        ;
                        ; DESCRIPTION:
                        ;
                        ;       1.  SAVE ENTRY REGISTERS AND SETUP FOR SHUTDOWN EXIT.
                        ;       2.  THE REQUIRED ENTRIES ARE BUILT IN THE GDT AT (ES:SI).
                        ;       3.  GATE ADDRESS LINE 20 ACTIVE, CLI AND SET SHUTDOWN CODES.
                        ;       4.  THE IDTR IS LOADED AND POINTS TO A ROM RESIDENT TABLE.
                        ;       5.  THE GDTR IS LOADED FROM THE OFFSET POINTER (ES:SI).
                        ;       6.  THE PROCESSOR IS PUT INTO PROTECTED MODE.
                        ;       7.  LOAD (DS) AND (ES) WITH SELECTORS FOR THE SOURCE AND TARGET.
                        ;       8.  DS:SI (SOURCE) (ES:DI) (TARGET) REP MOVSW IS EXECUTED.
                        ;       9.  CHECK MADE FOR PARITY ERRORS.
                        ;      10.  REAL MODE RESTORED WHEN SHUTDOWN 09H IS EXECUTED.
                        ;      11.  ERRORS ARE CHECKED FOR AND RETURN CODES ARE SET FOR (AH).
                        ;      12.  ADDRESS LINE 20 GATE IS DISABLED.
                        ;      13.  RETURN WITH REGISTERS RESTORED AND STATUS RETURN CODE.
                        ;             (FOR PC-AT COMPATIBILITY ZF=1 IF SUCCESSFUL, ZF=0 IF ERROR.)
                        ;-----------------------------------------------------------------------------
                        ;
                        ;       THE FOLLOWING DIAGRAM DEPICTS THE ORGANIZATION OF A BLOCK MOVE GDT.
                        ;-----------------------------------------------------------------------------
                        ;                 G D T
                        ; (ES:SI)
                        ;    |          .-----------.
                        ;    |          |           |
                        ;    |          v           |
                        ;    +00  .---------------. |
                        ;         |     DUMMY     | |   1. THE FIRST DESCRIPTOR IS THE REQUIRED DUMMY.
                        ;         |               | |      (USER INITIALIZED TO 0)
                        ;    +08  |---------------| |
                        ;         |    GDT LOC    |-'   2. THE SECOND DESCRIPTOR POINTS TO THE GDT
                        ;         |               |        TABLE AS A DATA SEGMENT.
                        ;    +10  |---------------|        (USER INITIALIZED TO 0 - MODIFIED BY BIOS)
                        ;         |    SOURCE     |     3. THE THIRD DESCRIPTOR POINTS TO THE SOURCE
                        ;         |      GDT      |        TO BE MOVED. (FROM)
                        ;    +18  |---------------|        (USER INITIALIZED)
                        ;         |    TARGET     |     4. THE FOURTH DESCRIPTOR POINTS TO THE
                        ;         |      GDT      |        DESTINATION SEGMENT. (TO)
                        ;    +20  |---------------|        (USER INITIALIZED)
                        ;         |     BIOS      |     5. THE FIFTH IS A DESCRIPTOR THAT BIOS USES
                        ;         |     (CS)      |        TO CREATE THE PROTECTED MODE CODE SEGMENT.
                        ;    +28  |---------------|        (USER INITIALIZED TO 0 - MODIFIED BY BIOS)
                        ;         |     (SS)      |     6. THE SIXTH DESCRIPTOR IS USED BY BIOS TO
                        ;         |               |        CREATE A PROTECTED MODE STACK SEGMENT.
                        ;         '---------------'        (USER INITIALIZED TO 0 - MODIFIED BY BIOS)
                        ;                                  (POINTS TO USERS STACK)
                        ;
                        ;
                        ;           SAMPLE  OF  SOURCE  OR  TARGET  DESCRIPTOR
                        ;
                        ;       SOURCE_TARGET_DEF     STRUC
                        ;
                        ;         SEG_LIMIT         DW     ?    ; SEGMENT LIMIT (1-65536 BYTES)
                        ;         LO_WORD           DW     ?    ; 24 BIT SEGMENT PHYSICAL
                        ;         HI_BYTE           DB     ?    ;      ADDRESS (0 TO (16M-1))
                        ;         DATA_ACC_RIGHTS   DB     93H  ; ACCESS RIGHTS BYTE (CPL0-R/W)
                        ;         RESERVED          DW     0    ; RESERVED WORD (MUST BE ZERO)
                        ;
                        ;       SOURCE_TARGET_DEF     ENDS
                        ;
                        ;-----------------------------------------------------------------------------
                        ;
                        ;     THE GLOBAL DESCRIPTOR TABLE (ACTUAL LOCATION POINTED TO BY ES:SI)
                        ;
                        BLOCKMOVE_GDT_DEF       STRUC
0000 ????????????????   ;                       DQ      ?               ; FIRST DESCRIPTOR NOT ACCESSIBLE
0008 ????????????????   CGDT_LOC                 DQ      ?               ; LOCATION OF CALLING ROUTINE GDT
0010 ????????????????   SOURCE                  DQ      ?               ; SOURCE DESCRIPTOR
0018 ????????????????   TARGET                  DQ      ?               ; TARGET DESCRIPTOR
0020 ????????????????   BIOS_CS                 DQ      ?               ; BIOS CODE DESCRIPTOR
0028 ????????????????   TEMP_SS                 DQ      ?               ; STACK DESCRIPTOR
0030                    BLOCKMOVE_GDT_DEF       ENDS
                        ;
01D6                    BLOCKMOVE       PROC    NEAR
                        ;
01D6 FC                         CLD                             ; SET DIRECTION FORWARD
01D7 60                         PUSHA                           ; SAVE GENERAL PURPOSE REGISTERS
01D8 06                         PUSH    ES                      ; SAVE USERS EXTRA SEGMENT
01D9 1E                         PUSH    DS                      ; SAVE USERS DATA SEGMENT
                        ;
                        ;-----  SAVE THE CALLING ROUTINE'S STACK
                        ;
01DA E8 0000 E                  CALL    DDS                     ; SET DS TO DATA AREA
```

```
01DD 8C 16 0069 R                MOV     @IO_ROM_SEG,SS          ; SAVE USERS STACK SEGMENT
01E1 89 26 0067 R                MOV     @IO_ROM_INIT,SP         ; SAVE USERS STACK POINTER

                         ;=====  SET UP THE PROTECTED MODE DEFINITIONS  =====

                         ;-----  MAKE A 24 BIT ADDRESS OUT OF THE ES:SI FOR THE GDT POINTER

                                 ASSUME  DS:NOTHING              ; POINT (DS) TO USERS CONTROL BLOCK
01E5 8C C0                       MOV     AX,ES                   ; GET THE GDT DATA SEGMENT
01E7 8E D8                       MOV     DS,AX                   ; MOVE THE GDT SEGMENT POINTER TO (DS)
01E9 8A F4                       MOV     DH,AH                   ; BUILD HIGH BYTE OF THE 24 BIT ADDRESS
01EB C0 EE 04                    SHR     DH,4                    ; USE ONLY HIGH NIBBLE SHIFT - RIGHT 4
01EE C1 E0 04                    SHL     AX,4                    ; STRIP HIGH NIBBLE FROM (AX)
01F1 03 C6                       ADD     AX,SI                   ; ADD THE GDT OFFSET TO DEVELOP LOW WORD
01F3 80 D6 00                    ADC     DH,0                    ; ADJUST HIGH BYTE IF CARRY FROM LOW

                         ;-----  SET THE GDT_LOC

01F6 C7 44 08 FFFF               MOV     [SI].CGDT_LOC.SEG_LIMIT,MAX_SEG_LEN
01FB 89 44 0A                    MOV     [SI].CGDT_LOC.BASE_LO_WORD,AX   ; SET THE LOW WORD
01FE 88 74 0C                    MOV     [SI].CGDT_LOC.BASE_HI_BYTE,DH   ; SET THE HIGH BYTE
0201 C7 44 0E 0000               MOV     [SI].CGDT_LOC.DATA_RESERVED,0   ; RESERVED

                         ;-----  SET UP THE CODE SEGMENT DESCRIPTOR

0206 C7 44 20 FFFF               MOV     [SI].BIOS_CS.SEG_LIMIT,MAX_SEG_LEN
020B C7 44 22 0000               MOV     [SI].BIOS_CS.BASE_LO_WORD,CSEG@_LO      ; LOW WORD OF (CS)= 0
0210 C6 44 24 0F                 MOV     [SI].BIOS_CS.BASE_HI_BYTE,CSEG@_HI      ; HIGH BYTE OF (CS)= 0FH
0214 C6 44 25 9B                 MOV     [SI].BIOS_CS.DATA_ACC_RIGHTS,CPL0_CODE_ACCESS
0218 C7 44 26 0000               MOV     [SI].BIOS_CS.DATA_RESERVED,0            ; RESERVED

                         ;-----  MAKE A 24 BIT ADDRESS OUT OF THE (SS) - ( (SP) REMAINS USER (SP) )

021D 8C D0                       MOV     AX,SS                   ; GET THE CURRENT STACK SEGMENT
021F 8A F4                       MOV     DH,AH                   ; FORM HIGH BYTE OF 24 BIT ADDRESS
0221 C0 EE 04                    SHR     DH,4                    ; FORM HIGH BYTE - SHIFT RIGHT 4
0224 C1 E0 04                    SHL     AX,4                    ; STRIP HIGH NIBBLE FROM (AX)

                         ;-----  SS IS NOW IN POSITION FOR A 24 BIT ADDRESS --> SETUP THE (SS) DESCRIPTOR

0227 C7 44 28 FFFF               MOV     [SI].TEMP_SS.SEG_LIMIT,MAX_SEG_LEN  ; SET THE SS SEGMENT LIMIT
022C 89 44 2A                    MOV     [SI].TEMP_SS.BASE_LO_WORD,AX        ; SET THE LOW WORD
022F 88 74 2C                    MOV     [SI].TEMP_SS.BASE_HI_BYTE,DH        ; SET THE HIGH BYTE
0232 C6 44 2D 93                 MOV     [SI].TEMP_SS.DATA_ACC_RIGHTS,CPL0_DATA_ACCESS  ; SET CPL 0

                         ;-----  GATE ADDRESS BIT 20 ON  (DISABLE INTERRUPTS)

0236 B4 DF                       MOV     AH,ENABLE_BIT20         ; GET ENABLE MASK
0238 E8 03DA R                   CALL    GATE_A20                ; ENABLE A20 AND CLEAR INTERRUPTS
023B 3C 00                       CMP     AL,0                    ; WAS THE COMMAND ACCEPTED?
023D 74 06                       JZ      BL4                     ; GO IF YES

023F B0 03                       MOV     AL,03H                  ; SET THE ERROR FLAG IF NOT
0241 E6 80                       OUT     MFG_PORT,AL
0243 EB 51                       JMP     SHORT SHUT9             ; EARLY ERROR EXIT

                         ;-----  SET SHUTDOWN RETURN ADDRESS AND DISABLE NMI

                         BL4:
0245                             MOV     AX,9*H+CMOS_SHUT_DOWN+NMI       ; SET THE SHUTDOWN BYTE LOCATION
0245 B8 098F                     CALL    CMOS_WRITE                      ; TO SHUT DOWN 9 AND DISABLE NMI
0248 E8 0000 E

                         ;-----  CLEAR EXCEPTION ERROR FLAG

024B 2A C0                       SUB     AL,AL
024D E6 80                       OUT     MFG_PORT,AL             ; SET ERROR FLAG LOCATION TO 0

                         ;-----  LOAD THE IDT AND GDT

024F BD 02D4 R                   MOV     BP,OFFSET ROM_IDT_LOC
                                 SEGOV   CS                      ; LOAD THE IDT
0252 2E                  +       DB      02EH
                                 LIDT    [BP]                    ;   REGISTER FROM THIS AREA
0253 0F                  +       DB      00FH
0254                     + ??0001 LABEL   BYTE
0254 8B 5E 00            +       MOV     BX,WORD PTR [BP]
0257                     + ??0002 LABEL   BYTE
0254                     +       ORG     OFFSET CS:??0001
0254 01                  +       DB      001H
0257                     +       ORG     OFFSET CS:??0002

                                 LGDT    [SI].CGDT_LOC           ; LOAD GLOBAL DESCRIPTOR TABLE REGISTER
0257 0F                  +       DB      00FH
0258                     + ??0003 LABEL   BYTE
0258 8B 54 08            +       MOV     DX,WORD PTR [SI].CGDT_LOC
025B                     + ??0004 LABEL   BYTE
0258                     +       ORG     OFFSET CS:??0003
0258 01                  +       DB      001H
025B                     +       ORG     OFFSET CS:??0004


                         ;-----  SWITCH TO VIRTUAL MODE

025B B8 0001                     MOV     AX,VIRTUAL_ENABLE       ; MACHINE STATUS WORD NEEDED TO
                                 LMSW    AX                      ;  SWITCH TO VIRTUAL MODE
025E 0F 01 F0            +       DB      00FH,001H,0F0H
0261 EA                           DB      0EAH                    ; PURGE PRE-FETCH QUEUE WITH FAR JUMP
0262 0266 R                       DW      OFFSET VIRT             ;  - TO OFFSET
0264 0020                         DW      BIOS_CS                 ;  - IN SEGMENT -PROTECTED MODE SELECTOR
0266                     VIRT:

                         ;-----  IN PROTECTED MODE - SETUP STACK SELECTOR AND SOURCE/TARGET SELECTORS

0266 B8 0028                     MOV     AX,TEMP_SS              ; USER'S SS+SP IS NOT A DESCRIPTOR
0269 8E D0                       MOV     SS,AX                   ; LOAD STACK SELECTOR
026B B8 0010                     MOV     AX,SOURCE               ; GET THE SOURCE ENTRY
026E 8E D8                       MOV     DS,AX                   ; LOAD SOURCE SELECTOR
0270 B8 0018                     MOV     AX,TARGET               ; GET THE TARGET ENTRY
0273 8E C0                       MOV     ES,AX                   ; LOAD TARGET SELECTOR
0275 2B F6                       SUB     SI,SI                   ; SET SOURCE INDEX REGISTER TO ZERO
0277 2B FF                       SUB     DI,DI                   ; SET TARGET INDEX REGISTER TO ZERO

0279 F3/ A5                      REP     MOVSW                   ; MOVE THE BLOCK COUNT PASSED IN (CX)
```

SECTION 5

```
                        ;-----  CHECK FOR MEMORY PARITY BEFORE SHUTDOWN
027B E4 61                      IN      AL,PORT_B               ; GET THE PARITY LATCHES
027D 24 C0                      AND     AL,PARITY_ERR           ; STRIP UNWANTED BITS
027F 74 12                      JZ      DONE1                   ; GO IF NO PARITY ERROR

                        ;-----  CLEAR PARITY BEFORE SHUTDOWN

0281 8B 05                      MOV     AX,DS:[DI]              ; FETCH CURRENT SOURCE DATA
0283 89 05                      MOV     DS:[DI],AX              ; WRITE IT BACK
0285 B0 01                      MOV     AL,01                   ; SET PARITY CHECK ERROR = 01
0287 E6 80                      OUT     MFG_PORT,AL
0289 E4 61                      IN      AL,PORT_B
028B 0C 0C                      OR      AL,RAM_PAR_OFF          ; TOGGLE PARITY CHECK LATCHES
028D E6 61                      OUT     PORT_B,AL               ;  TO CLEAR THE PENDING ERROR
028F 24 F3                      AND     AL,RAM_PAR_ON           ;  AND ENABLE CHECKING
0291 E6 61                      OUT     PORT_B,AL

                        ;-----  CAUSE A SHUTDOWN

0293                    DONE1:
0293 E9 0000 E                  JMP     PROC_SHUTDOWN           ; GO RESET PROCESSOR AND SHUTDOWN

                                ;=======================
                        ;------   RETURN FROM SHUTDOWN
                                ;=======================

0296                    SHUT9:                                  ;       RESTORE USERS STACK
                                ASSUME  DS:DATA
0296 B8 ---- R                  MOV     AX,DATA                 ; SET DS TO DATA AREA
0299 8E D8                      MOV     DS,AX
029B 8E 16 0069 R               MOV     SS,@IO_ROM_SEG          ; GET USER STACK SEGMENT
029F 8B 26 0067 R               MOV     SP,@IO_ROM_INIT         ; GET USER STACK POINTER

                        ;-----  GATE ADDRESS BIT 20 OFF

02A3 B4 DD                      MOV     AH,DISABLE_BIT20        ; DISABLE MASK
02A5 E8 03DA R                  CALL    GATE_A20                ; GATE ADDRESS 20 LINE OFF
02A8 3C 00                      CMP     AL,0                    ; COMMAND ACCEPTED?
02AA 74 0A                      JZ      DONE3                   ; GO IF YES

02AC E4 80                      IN      AL,MFG_PORT             ; CHECK FOR ANY OTHER ERROR FIRST
02AE 3C 00                      CMP     AL,0                    ; WAS THERE AN ERROR?
02B0 75 04                      JNZ     DONE3                   ; REPORT FIRST ERROR IF YES
02B2 B0 03                      MOV     AL,03H                  ; ELSE SET GATE A20 ERROR FLAG
02B4 E6 80                      OUT     MFG_PORT,AL

                        ;-----  RESTORE THE USERS REGISTERS AND SET RETURN CODES

02B6                    DONE3:
02B6 B8 000F                    MOV     AX,CMOS_SHUT_DOWN       ; CLEAR (AH) TO ZERO AND (AL) TO DEFAULT
02B9 E6 70                      OUT     CMOS_PORT,AL            ; ENABLE NMI INTERRUPTS
02BB 1F                         POP     DS                      ; RESTORE USER DATA SEGMENT
02BC 07                         POP     ES                      ; RESTORE USER EXTRA SEGMENT
02BD E4 71                      IN      AL,CMOS_DATA            ; OPEN CMOS STANDBY LATCH

02BF E4 80                      IN      AL,MFG_PORT             ; GET THE ENDING STATUS RETURN CODE
02C1 8B EC                      MOV     BP,SP                   ; POINT TO REGISTERS IN THE STACK
02C3 88 46 0F                   MOV     [BP+15],AL              ; PLACE ERROR CODE INTO STACK AT (AH)
02C6 3A E0                      CMP     AH,AL                   ; SET THE ZF & CY FLAGS WITH RETURN CODE
02C8 61                         POPA                            ; RESTORE THE GENERAL PURPOSE REGISTERS
02C9 FB                         STI                             ; TURN INTERRUPTS ON
02CA                    DONE4   PROC    FAR
02CA CA 0002                    RET     2                       ; RETURN WITH FLAGS SET --  (AH)= CODE
02CD                    DONE4   ENDP                            ;  (CY=0,ZF=1)= OK   (CY=1,ZF=0)= ERROR

                        ;-----  BLOCK MOVE EXCEPTION INTERRUPT HANDLER

02CD                    EX_INT:
02CD B0 02                      MOV     AL,02H                  ; GET EXCEPTION ERROR CODE
02CF E6 80                      OUT     MFG_PORT,AL             ; SET EXCEPTION INTERRUPT OCCURRED FLAG
02D1 E9 0000 E                  JMP     PROC_SHUTDOWN           ; CAUSE A EARLY SHUTDOWN

                        ;-----  ROM IDT LOCATION

02D4                    ROM_IDT_LOC:
02D4 0100                       DW      ROM_IDT_END-ROM_IDT     ; LENGTH OF ROM IDT TABLE
02D6 02DA R                     DW      ROM_IDT                 ; LOW WORD OF BASE ADDRESS
02D8 0F                         DB      CSEG@_HI                ; HIGH BYTE OF BASE ADDRESS
02D9 00                         DB      0                       ; RESERVED

                        ;-----  THE ROM EXCEPTION INTERRUPT VECTOR GATES FOR BLOCK MOVE

02DA                    ROM_IDT:                                ;       EXCEPTION 00
02DA 02CD R                     DW      EX_INT                  ; DESTINATION OFFSET
02DC 0020                       DW      BIOS_CS                 ; DESTINATION SEGMENT SELECTOR
02DE 00                         DB      0                       ; WORD COPY COUNT
02DF 87                         DB      TRAP_GATE               ; GATE TYPE - ACCESS RIGHTS BYTE
02E0 0000                       DW      0                       ; RESERVED
                                                                ;       EXCEPTION 01
02E2 02CD R                     DW      EX_INT                  ; DESTINATION OFFSET
02E4 0020                       DW      BIOS_CS                 ; DESTINATION SEGMENT SELECTOR
02E6 00                         DB      0                       ; WORD COPY COUNT
02E7 87                         DB      TRAP_GATE               ; GATE TYPE - ACCESS RIGHTS BYTE
02E8 0000                       DW      0                       ; RESERVED
                                                                ;       EXCEPTION 02
02EA 02CD R                     DW      EX_INT                  ; DESTINATION OFFSET
02EC 0020                       DW      BIOS_CS                 ; DESTINATION SEGMENT SELECTOR
02EE 00                         DB      0                       ; WORD COPY COUNT
02EF 87                         DB      TRAP_GATE               ; GATE TYPE - ACCESS RIGHTS BYTE
02F0 0000                       DW      0                       ; RESERVED
                                                                ;       EXCEPTION 03
02F2 02CD R                     DW      EX_INT                  ; DESTINATION OFFSET
02F4 0020                       DW      BIOS_CS                 ; DESTINATION SEGMENT SELECTOR
02F6 00                         DB      0                       ; WORD COPY COUNT
02F7 87                         DB      TRAP_GATE               ; GATE TYPE - ACCESS RIGHTS BYTE
02F8 0000                       DW      0                       ; RESERVED
                                                                ;       EXCEPTION 04
02FA 02CD R                     DW      EX_INT                  ; DESTINATION OFFSET
02FC 0020                       DW      BIOS_CS                 ; DESTINATION SEGMENT SELECTOR
02FE 00                         DB      0                       ; WORD COPY COUNT
02FF 87                         DB      TRAP_GATE               ; GATE TYPE - ACCESS RIGHTS BYTE
0300 0000                       DW      0                       ; RESERVED
                                                                ;       EXCEPTION 05
```

```
0302 02CD R                     DW      EX_INT          ; DESTINATION OFFSET
0304 0020                       DW      BIOS_CS         ; DESTINATION SEGMENT SELECTOR
0306 00                         DB      0               ; WORD COPY COUNT
0307 87                         DB      TRAP_GATE       ; GATE TYPE - ACCESS RIGHTS BYTE
0308 0000                       DW      0               ; RESERVED
                                                        ;       EXCEPTION 06
030A 02CD R                     DW      EX_INT          ; DESTINATION OFFSET
030C 0020                       DW      BIOS_CS         ; DESTINATION SEGMENT SELECTOR
030E 00                         DB      0               ; WORD COPY COUNT
030F 87                         DB      TRAP_GATE       ; GATE TYPE - ACCESS RIGHTS BYTE
0310 0000                       DW      0               ; RESERVED
                                                        ;       EXCEPTION 07
0312 02CD R                     DW      EX_INT          ; DESTINATION OFFSET
0314 0020                       DW      BIOS_CS         ; DESTINATION SEGMENT SELECTOR
0316 00                         DB      0               ; WORD COPY COUNT
0317 87                         DB      TRAP_GATE       ; GATE TYPE - ACCESS RIGHTS BYTE
0318 0000                       DW      0               ; RESERVED
                                                        ;       EXCEPTION 08
031A 02CD R                     DW      EX_INT          ; DESTINATION OFFSET
031C 0020                       DW      BIOS_CS         ; DESTINATION SEGMENT SELECTOR
031E 00                         DB      0               ; WORD COPY COUNT
031F 87                         DB      TRAP_GATE       ; GATE TYPE - ACCESS RIGHTS BYTE
0320 0000                       DW      0               ; RESERVED
                                                        ;       EXCEPTION 09
0322 02CD R                     DW      EX_INT          ; DESTINATION OFFSET
0324 0020                       DW      BIOS_CS         ; DESTINATION SEGMENT SELECTOR
0326 00                         DB      0               ; WORD COPY COUNT
0327 87                         DB      TRAP_GATE       ; GATE TYPE - ACCESS RIGHTS BYTE
0328 0000                       DW      0               ; RESERVED
                                                        ;       EXCEPTION 10
032A 02CD R                     DW      EX_INT          ; DESTINATION OFFSET
032C 0020                       DW      BIOS_CS         ; DESTINATION SEGMENT SELECTOR
032E 00                         DB      0               ; WORD COPY COUNT
032F 87                         DB      TRAP_GATE       ; GATE TYPE - ACCESS RIGHTS BYTE
0330 0000                       DW      0               ; RESERVED
                                                        ;       EXCEPTION 11
0332 02CD R                     DW      EX_INT          ; DESTINATION OFFSET
0334 0020                       DW      BIOS_CS         ; DESTINATION SEGMENT SELECTOR
0336 00                         DB      0               ; WORD COPY COUNT
0337 87                         DB      TRAP_GATE       ; GATE TYPE - ACCESS RIGHTS BYTE
0338 0000                       DW      0               ; RESERVED
                                                        ;       EXCEPTION 12
033A 02CD R                     DW      EX_INT          ; DESTINATION OFFSET
033C 0020                       DW      BIOS_CS         ; DESTINATION SEGMENT SELECTOR
033E 00                         DB      0               ; WORD COPY COUNT
033F 87                         DB      TRAP_GATE       ; GATE TYPE - ACCESS RIGHTS BYTE
0340 0000                       DW      0               ; RESERVED
                                                        ;       EXCEPTION 13
0342 02CD R                     DW      EX_INT          ; DESTINATION OFFSET
0344 0020                       DW      BIOS_CS         ; DESTINATION SEGMENT SELECTOR
0346 00                         DB      0               ; WORD COPY COUNT
0347 87                         DB      TRAP_GATE       ; GATE TYPE - ACCESS RIGHTS BYTE
0348 0000                       DW      0               ; RESERVED
                                                        ;       EXCEPTION 14
034A 02CD R                     DW      EX_INT          ; DESTINATION OFFSET
034C 0020                       DW      BIOS_CS         ; DESTINATION SEGMENT SELECTOR
034E 00                         DB      0               ; WORD COPY COUNT
034F 87                         DB      TRAP_GATE       ; GATE TYPE - ACCESS RIGHTS BYTE
0350 0000                       DW      0               ; RESERVED
                                                        ;       EXCEPTION 15
0352 02CD R                     DW      EX_INT          ; DESTINATION OFFSET
0354 0020                       DW      BIOS_CS         ; DESTINATION SEGMENT SELECTOR
0356 00                         DB      0               ; WORD COPY COUNT
0357 87                         DB      TRAP_GATE       ; GATE TYPE - ACCESS RIGHTS BYTE
0358 0000                       DW      0               ; RESERVED
                                                        ;       EXCEPTION 16
035A 02CD R                     DW      EX_INT          ; DESTINATION OFFSET
035C 0020                       DW      BIOS_CS         ; DESTINATION SEGMENT SELECTOR
035E 00                         DB      0               ; WORD COPY COUNT
035F 87                         DB      TRAP_GATE       ; GATE TYPE - ACCESS RIGHTS BYTE
0360 0000                       DW      0               ; RESERVED
                                                        ;       EXCEPTION 17
0362 02CD R                     DW      EX_INT          ; DESTINATION OFFSET
0364 0020                       DW      BIOS_CS         ; DESTINATION SEGMENT SELECTOR
0366 00                         DB      0               ; WORD COPY COUNT
0367 87                         DB      TRAP_GATE       ; GATE TYPE - ACCESS RIGHTS BYTE
0368 0000                       DW      0               ; RESERVED
                                                        ;       EXCEPTION 18
036A 02CD R                     DW      EX_INT          ; DESTINATION OFFSET
036C 0020                       DW      BIOS_CS         ; DESTINATION SEGMENT SELECTOR
036E 00                         DB      0               ; WORD COPY COUNT
036F 87                         DB      TRAP_GATE       ; GATE TYPE - ACCESS RIGHTS BYTE
0370 0000                       DW      0               ; RESERVED
                                                        ;       EXCEPTION 19
0372 02CD R                     DW      EX_INT          ; DESTINATION OFFSET
0374 0020                       DW      BIOS_CS         ; DESTINATION SEGMENT SELECTOR
0376 00                         DB      0               ; WORD COPY COUNT
0377 87                         DB      TRAP_GATE       ; GATE TYPE - ACCESS RIGHTS BYTE
0378 0000                       DW      0               ; RESERVED
                                                        ;       EXCEPTION 20
037A 02CD R                     DW      EX_INT          ; DESTINATION OFFSET
037C 0020                       DW      BIOS_CS         ; DESTINATION SEGMENT SELECTOR
037E 00                         DB      0               ; WORD COPY COUNT
037F 87                         DB      TRAP_GATE       ; GATE TYPE - ACCESS RIGHTS BYTE
0380 0000                       DW      0               ; RESERVED
                                                        ;       EXCEPTION 21
0382 02CD R                     DW      EX_INT          ; DESTINATION OFFSET
0384 0020                       DW      BIOS_CS         ; DESTINATION SEGMENT SELECTOR
0386 00                         DB      0               ; WORD COPY COUNT
0387 87                         DB      TRAP_GATE       ; GATE TYPE - ACCESS RIGHTS BYTE
0388 0000                       DW      0               ; RESERVED
                                                        ;       EXCEPTION 22
038A 02CD R                     DW      EX_INT          ; DESTINATION OFFSET
038C 0020                       DW      BIOS_CS         ; DESTINATION SEGMENT SELECTOR
038E 00                         DB      0               ; WORD COPY COUNT
038F 87                         DB      TRAP_GATE       ; GATE TYPE - ACCESS RIGHTS BYTE
0390 0000                       DW      0               ; RESERVED
                                                        ;       EXCEPTION 23
0392 02CD R                     DW      EX_INT          ; DESTINATION OFFSET
0394 0020                       DW      BIOS_CS         ; DESTINATION SEGMENT SELECTOR
0396 00                         DB      0               ; WORD COPY COUNT
0397 87                         DB      TRAP_GATE       ; GATE TYPE - ACCESS RIGHTS BYTE
0398 0000                       DW      0               ; RESERVED
                                                        ;       EXCEPTION 24
```

```
03A2 02CD R                     DW      EX_INT          ; DESTINATION OFFSET
03A4 0020                       DW      BIOS_CS         ; DESTINATION SEGMENT SELECTOR
03A6 00                         DB      0               ; WORD COPY COUNT
03A7 87                         DB      TRAP_GATE       ; GATE TYPE - ACCESS RIGHTS BYTE
03A8 0000                       DW      0               ; RESERVED
                                                        ;       EXCEPTION 25
03AA 02CD R                     DW      EX_INT          ; DESTINATION OFFSET
03AC 0020                       DW      BIOS_CS         ; DESTINATION SEGMENT SELECTOR
03AE 00                         DB      0               ; WORD COPY COUNT
03AF 87                         DB      TRAP_GATE       ; GATE TYPE - ACCESS RIGHTS BYTE
03B0 0000                       DW      0               ; RESERVED
                                                        ;       EXCEPTION 26
03B2 02CD R                     DW      EX_INT          ; DESTINATION OFFSET
03B4 0020                       DW      BIOS_CS         ; DESTINATION SEGMENT SELECTOR
03B6 00                         DB      0               ; WORD COPY COUNT
03B7 87                         DB      TRAP_GATE       ; GATE TYPE - ACCESS RIGHTS BYTE
03B8 0000                       DW      0               ; RESERVED
                                                        ;       EXCEPTION 27
03BA 02CD R                     DW      EX_INT          ; DESTINATION OFFSET
03BC 0020                       DW      BIOS_CS         ; DESTINATION SEGMENT SELECTOR
03BE 00                         DB      0               ; WORD COPY COUNT
03BF 87                         DB      TRAP_GATE       ; GATE TYPE - ACCESS RIGHTS BYTE
03C0 0000                       DW      0               ; RESERVED
                                                        ;       EXCEPTION 28
03C2 02CD R                     DW      EX_INT          ; DESTINATION OFFSET
03C4 0020                       DW      BIOS_CS         ; DESTINATION SEGMENT SELECTOR
03C6 00                         DB      0               ; WORD COPY COUNT
03C7 87                         DB      TRAP_GATE       ; GATE TYPE - ACCESS RIGHTS BYTE
03C8 0000                       DW      0               ; RESERVED
                                                        ;       EXCEPTION 29
03CA 02CD R                     DW      EX_INT          ; DESTINATION OFFSET
03CC 0020                       DW      BIOS_CS         ; DESTINATION SEGMENT SELECTOR
03CE 00                         DB      0               ; WORD COPY COUNT
03CF 87                         DB      TRAP_GATE       ; GATE TYPE - ACCESS RIGHTS BYTE
03D0 0000                       DW      0               ; RESERVED
                                                        ;       EXCEPTION 30
03D2 02CD R                     DW      EX_INT          ; DESTINATION OFFSET
03D4 0020                       DW      BIOS_CS         ; DESTINATION SEGMENT SELECTOR
03D6 00                         DB      0               ; WORD COPY COUNT
03D7 87                         DB      TRAP_GATE       ; GATE TYPE - ACCESS RIGHTS BYTE
03D8 0000                       DW      0               ; RESERVED
03DA                    ROM_IDT_END:

03DA                    BLOCKMOVE       ENDP
```

```
                        PAGE
                        ;--------------------------------------------------------------------------------
                        ; GATE_A20                                                                      :
                        ;       THIS ROUTINE CONTROLS A SIGNAL WHICH GATES ADDRESS BIT 20.              :
                        ;       THE GATE A20 SIGNAL IS AN OUTPUT OF THE 8042 SLAVE PROCESSOR.          :
                        ;       ADDRESS BIT 20 SHOULD BE GATED ON BEFORE ENTERING PROTECTED MODE.      :
                        ;       IT SHOULD BE GATED OFF AFTER ENTERING REAL MODE FROM PROTECTED         :
                        ;       MODE.   INTERRUPTS ARE LEFT DISABLED ON EXIT.                          :
                        ; INPUT                                                                        :
                        ;       (AH)= DDH  ADDRESS BIT 20 GATE OFF. (A20 ALWAYS ZERO)                  :
                        ;       (AH)= DFH  ADDRESS BIT 20 GATE ON. (A20 CONTROLLED BY 80286)           :
                        ; OUTPUT                                                                       :
                        ;       (AL)= 00H  OPERATION SUCCESSFUL. 8042 HAS ACCEPTED COMMAND.            :
                        ;       (AL)= 02H  FAILURE--8042 UNABLE TO ACCEPT COMMAND.                     :
                        ;--------------------------------------------------------------------------------
03DA                    GATE_A20        PROC
03DA 51                         PUSH    CX                      ; SAVE USERS (CX)
03DB FA                         CLI                             ; DISABLE INTERRUPTS WHILE USING 8042
03DC E8 03F3 R                  CALL    EMPTY_8042              ; INSURE 8042 INPUT BUFFER EMPTY
03DF 75 10                      JNZ     GATE_A20_RETURN         ; EXIT IF 8042 UNABLE TO ACCEPT COMMAND
03E1 B0 D1                      MOV     AL,0D1H                 ; 8042 COMMAND TO WRITE OUTPUT PORT
03E3 E6 64                      OUT     STATUS_PORT,AL          ; OUTPUT COMMAND TO 8042
03E5 E8 03F3 R                  CALL    EMPTY_8042              ; WAIT FOR 8042 TO ACCEPT COMMAND
03E8 75 07                      JNZ     GATE_A20_RETURN         ; EXIT IF 8042 UNABLE TO ACCEPT COMMAND
03EA 8A C4                      MOV     AL,AH                   ; 8042 PORT DATA
03EC E6 60                      OUT     PORT_A,AL               ; OUTPUT PORT DATA TO 8042
03EE E8 03F3 R                  CALL    EMPTY_8042              ; WAIT FOR 8042 TO ACCEPT PORT DATA

                        ;-----  8042 OUTPUT WILL SWITCH WITHIN 20 MICRO SECONDS OF ACCEPTING PORT DATA

03F1                    GATE_A20_RETURN:
03F1 59                         POP     CX                      ; RESTORE USERS (CX)
03F2 C3                         RET
                        ;--------------------------------------------------------------------------------
                        ; EMPTY_8042                                                                   :
                        ;       THIS ROUTINE WAITS FOR THE 8042 INPUT BUFFER TO EMPTY.                 :
                        ; INPUT                                                                        :
                        ;       NONE                                                                   :
                        ; OUTPUT                                                                       :
                        ;       (AL)= 00H  8042 INPUT BUFFER EMPTY (ZERO FLAG SET)                     :
                        ;       (AL)= 02H  TIME OUT, 8042 INPUT BUFFER FULL (NON-ZERO FLAG SET)        :
                        ;       (CX)       - MODIFIED                                                  :
                        ;--------------------------------------------------------------------------------
03F3                    EMPTY_8042:
03F3 2B C9                      SUB     CX,CX                   ; (CX)=0, WILL BE USED AS TIME OUT VALUE
03F5                    EMPTY_L:
03F5 E4 64                      IN      AL,STATUS_PORT          ; READ 8042 STATUS PORT
03F7 24 02                      AND     AL,INPT_BUF_FULL        ; TEST INPUT BUFFER FULL FLAG (BIT 1)
03F9 E0 FA                      LOOPNZ  EMPTY_L                 ; LOOP UNTIL BUFFER EMPTY OR TIME OUT
03FB C3                         RET
03FC                    GATE_A20        ENDP


                        ;--- INT 15 H -- ( FUNCTION 88 H - I/O MEMORY SIZE DETERMINE ) ------------------
                        ; EXT_MEMORY                                                                   :
                        ;       THIS ROUTINE RETURNS  THE AMOUNT OF MEMORY IN THE SYSTEM THAT IS       :
                        ;       LOCATED STARTING AT THE 1024K ADDRESSING RANGE, AS DETERMINED BY       :
                        ;       THE POST ROUTINES.                                                     :
                        ;       NOTE THAT THE SYSTEM MAY NOT BE ABLE TO USE I/O MEMORY UNLESS THERE    :
                        ;       IS A FULL COMPLEMENT OF 512K OR 640 BYTES ON THE PLANAR.  THIS SIZE    :
                        ;       SIZE IS STORED IN CMOS AT ADDRESS LOCATIONS 30H AND 31H.               :
                        ; INPUT                                                                        :
                        ;       AH = 88H                                                               :
                        ;                                                                              :
                        ;       THE I/O MEMORY SIZE VARIABLE IS SET DURING POWER ON                    :
                        ;       DIAGNOSTICS ACCORDING TO THE FOLLOWING ASSUMPTIONS:                    :
                        ;                                                                              :
                        ;       1. ALL INSTALLED MEMORY IS FUNCTIONAL.                                 :
                        ;       2. ALL MEMORY FROM 0 TO 640K MUST BE CONTIGUOUS.                       :
                        ;                                                                              :
                        ; OUTPUT                                                                       :
                        ;       (AX) = NUMBER OF CONTIGUOUS 1K BLOCKS OF MEMORY A                      :
                        ;              AVAILABLE STARTING AT ADDRESS 1024K.                            :
                        ;                                                                              :
                        ;--------------------------------------------------------------------------------

03FC                    EXT_MEMORY      PROC

03FC B8 3031                    MOV     AX,CMOS_U_M_S_LO*H+CMOS_U_M_S_HI ; ADDRESS HIGH/LOW BYTES
03FF E8 0000 E                  CALL    CMOS_READ               ; GET THE HIGH BYTE OF I/O MEMORY
0402 86 C4                      XCHG    AL,AH                   ; PUT HIGH BYTE IN POSITION (AH)
0404 E8 0000 E                  CALL    CMOS_READ               ; GET THE LOW BYTE OF I/O MEMORY
0407 CF                         IRET                            ; RETURN TO USER

0408                    EXT_MEMORY      ENDP
```

```
                        PAGE
                        ;--- INT 15 H ( FUNCTION 89 H ) ----------------------------------------------
                        ; PURPOSE:                                                                    :
                        ;       THIS BIOS FUNCTION PROVIDES A MEANS TO THE USER TO SWITCH INTO        :
                        ;       VIRTUAL (PROTECTED) MODE.  UPON COMPLETION OF THIS FUNCTION THE       :
                        ;       PROCESSOR WILL BE IN  VIRTUAL (PROTECTED) MODE AND CONTROL WILL       :
                        ;       BE TRANSFERRED TO THE CODE SEGMENT THAT WAS SPECIFIED BY THE USER.    :
                        ;                                                                             :
                        ; ENTRY REQUIREMENTS:                                                         :
                        ;                                                                             :
                        ;       (ES:SI) POINTS TO A DESCRIPTOR TABLE (GDT) BUILT BEFORE INTERRUPTING  :
                        ;       TO THIS FUNCTION.  THESE DESCRIPTORS ARE USED BY THIS FUNCTION TO     :
                        ;       INITIALIZE THE IDTR, THE GDTR AND THE STACK SEGMENT SELECTOR.  THE    :
                        ;       DATA SEGMENT (DS) SELECTOR AND THE EXTRA SEGMENT (ES) SELECTOR WILL   :
                        ;       BE INITIALIZE TO DESCRIPTORS BUILT BY THE ROUTINE USING THIS FUNCTION.:
                        ;       BH - OFFSET INTO THE INTERRUPT DESCRIPTOR TABLE STATING WHERE THE      :
                        ;            FIRST EIGHT HARDWARE INTERRUPTS WILL BEGIN. ( INTERRUPT LEVEL 1 ):
                        ;       BL - OFFSET INTO THE INTERRUPT DESCRIPTOR TABLE STATING WHERE THE      :
                        ;            SECOND EIGHT HARDWARE INTERRUPTS BEGIN. ( INTERRUPT LEVEL 2 )    :
                        ;                                                                             :
                        ; THE DESCRIPTORS ARE DEFINED AS FOLLOWS:                                     :
                        ;                                                                             :
                        ;       1.  THE FIRST DESCRIPTOR IS THE REQUIRED DUMMY.                       :
                        ;           (USER INITIALIZED TO 0)                                           :
                        ;       2.  THE SECOND DESCRIPTOR POINTS TO THE GDT TABLE AS                  :
                        ;           A DATA SEGMENT.                                                   :
                        ;           (USER INITIALIZED)                                                :
                        ;       3.  THE THIRD DESCRIPTOR POINTS TO THE USER DEFINED                   :
                        ;           INTERRUPT DESCRIPTOR TABLE (IDT).                                 :
                        ;           (USER INITIALIZED)                                                :
                        ;       4.  THE FORTH DESCRIPTOR POINTS TO THE USER'S DATA                    :
                        ;           SEGMENT (DS).                                                     :
                        ;           (USER INITIALIZED)                                                :
                        ;       5.  THE FIFTH DESCRIPTOR POINTS TO THE USER'S EXTRA                   :
                        ;           SEGMENT (ES).                                                     :
                        ;           (USER INITIALIZED)                                                :
                        ;       6.  THE SIXTH DESCRIPTOR POINTS TO THE USER'S STACK                   :
                        ;           SEGMENT (SS).                                                     :
                        ;           (USER INITIALIZED)                                                :
                        ;       7.  THE SEVENTH DESCRIPTOR POINTS TO THE CODE SEGMENT                 :
                        ;           THAT THIS FUNCTION WILL RETURN TO.                                :
                        ;           (USER INITIALIZED TO THE USER'S CODE SEGMENT.)                    :
                        ;       8.  THE EIGHTH DESCRIPTOR IS USED BY THIS FUNCTION TO                 :
                        ;           ESTABLISH A CODE SEGMENT FOR ITSELF. THIS IS                      :
                        ;           NEEDED SO THAT THIS FUNCTION CAN COMPLETE IT'S                    :
                        ;           EXECUTION WHILE IN PROTECTED MODE.  WHEN CONTROL                  :
                        ;           GETS PASSED TO THE USER'S CODE THIS DESCRIPTOR CAN                 :
                        ;           BE USED BY HIM IN ANY WAY HE CHOOSES.                             :
                        ;                                                                             :
                        ;   NOTE -  EACH DESCRIPTOR MUST CONTAIN ALL THE NECESSARY DATA               :
                        ;           I.E. THE LIMIT,  BASE ADDRESS AND THE ACCESS RIGHTS BYTE.         :
                        ;                                                                             :
                        ;       AH= 89H  (FUNCTION CALL)                                              :
                        ;       ES:SI = LOCATION OF THE GDT TABLE BUILD BY ROUTINE                    :
                        ;       USING THIS FUNCTION.                                                  :
                        ;                                                                             :
                        ; EXIT PARAMETERS:                                                            :
                        ;                                                                             :
                        ;       AH = 0  IF SUCCESSFUL                                                 :
                        ;       ALL SEGMENT REGISTERS ARE CHANGED, (AX) AND (BP) DESTROYED            :
                        ;                                                                             :
                        ; CONSIDERATIONS:                                                             :
                        ;                                                                             :
                        ;       1.  NO BIOS AVAILABLE TO USER.  USER MUST HANDLE ALL                  :
                        ;           I/O COMMANDS.                                                     :
                        ;       2.  INTERRUPTS - INTERRUPT VECTOR LOCATIONS MUST BE                   :
                        ;           MOVED,  DUE TO THE 286 RESERVED AREAS.  THE                       :
                        ;           HARDWARE INTERRUPT CONTROLLERS MUST BE REINITIALIZED               :
                        ;           TO DEFINE LOCATIONS THAT DO NOT RESIDE IN THE 286                 :
                        ;           RESERVED AREAS.                                                   :
                        ;       3.  EXCEPTION INTERRUPT TABLE AND HANDLER MUST BE                     :
                        ;           INITIALIZED BY THE USER.                                          :
                        ;       4.  THE INTERRUPT DESCRIPTOR TABLE MUST NOT OVERLAP                   :
                        ;           THE REAL MODE BIOS INTERRUPT DESCRIPTOR TABLE.                    :
                        ;       5.  THE FOLLOWING GIVES AN IDEA OF WHAT THE USER CODE                 :
                        ;           SHOULD LOOK LIKE WHEN INVOKING THIS FUNCTION.                     :
                        ;                                                                             :
                        ;       REAL MODE --->    "USER CODE"                                         :
                        ;                 *       MOV   AX,GDT SEGMENT                                :
                        ;                 *       MOV   ES,AX                                         :
                        ;                 *       MOV   SI,GDT OFFSET                                 :
                        ;                 *       MOV   BH,HARDWARE INT LEVEL 1 OFFSET                :
                        ;                 *       MOV   BL,HARDWARE INT LEVEL 2 OFFSET                :
                        ;                 *       MOV   AH,89H                                        :
                        ;                 *       INT   15H                                           :
                        ;   VIRTUAL MODE --->     "USER CODE"                                         :
                        ;                                                                             :
                        ; DESCRIPTION:                                                                :
                        ;                                                                             :
                        ;       1.  CLI (NO INTERRUPTS ALLOWED) WHILE THIS FUNCTION IS EXECUTING.     :
                        ;       2.  ADDRESS LINE 20 IS GATED ACTIVE.                                  :
                        ;       3.  THE CURRENT USER STACK SEGMENT DESCRIPTOR IS INITIALIZED.          :
                        ;       4.  THE GDTR IS LOADED WITH THE GDT BASE ADDRESS.                     :
                        ;       5.  THE IDTR IS LOADED WITH THE IDT BASE ADDRESS.                     :
                        ;       6.  THE 8259 IS REINITIALIZED WITH THE NEW INTERRUPT OFFSETS.         :
                        ;       7.  THE PROCESSOR IS PUT IN VIRTUAL MODE WITH THE CODE                :
                        ;           SEGMENT DESIGNATED FOR THIS FUNCTION.                             :
                        ;       8.  DATA SEGMENT IS LOADED WITH THE USER DEFINED                      :
                        ;           SELECTOR FOR THE DS REGISTER.                                     :
                        ;       9.  EXTRA SEGMENT IS LOADED WITH THE USER DEFINED                     :
                        ;           SELECTOR FOR THE ES REGISTER.                                     :
                        ;      10.  STACK SEGMENT IS LOADED WITH THE USER DEFINED                     :
                        ;           SELECTOR FOR THE SS REGISTER.                                     :
                        ;      11.  CODE SEGMENT DESCRIPTOR SELECTOR VALUE IS                         :
                        ;           SUBSTITUTED ON THE STACK FOR RETURN TO USER.                      :
                        ;      12.  WE TRANSFER CONTROL TO THE USER WITH INTERRUPTS DISABLED.         :
                        ;-------------------------------------------------------------------------------
```

```
                        PAGE
                        ;
                        ;       THE FOLLOWING DIAGRAM DEPICTS THE ORGANIZATION
                        ;       OF GDT.
                        ;-----------------------------------------------------------------------
                        ;                               G D T                                  :
                        ;                                .-------------.                       :
                        ;                                v             |                       :
                        ;     (ES:SI)-->>  +00 .---------------.      |                       :
                        ;                    |     DUMMY       |      |                       :
                        ;                +08 |---------------  |      |                       :
                        ;                    |     GDT         |  ----'                       :
                        ;                +10 |---------------  |                              :
                        ;                    |     IDT         |                              :
                        ;                +18 |---------------  |                              :
                        ;                    |     DS          |                              :
                        ;                +20 |---------------  |                              :
                        ;                    |     ES          |                              :
                        ;                +28 |---------------  |                              :
                        ;                    |     SS          |                              :
                        ;                +30 |---------------  |                              :
                        ;                    |     CS          |                              :
                        ;                +38 |---------------  |                              :
                        ;                    |   TEMP BIOS     |                              :
                        ;                    |     CS          |                              :
                        ;                    '---------------'                              :
                        ;-----------------------------------------------------------------------

                        ;-----------------------------------------------------------------------
                        ;       THE GLOBAL DESCRIPTOR TABLE (ACTUAL LOCATION POINTED TO BY ES:SI)       :
                        ;-----------------------------------------------------------------------

                        VIRTUAL_ENABLE_GDT_DEF  STRUC
0000 ????????????????   GDTPTR          DQ      ?       ; FIRST DESCRIPTOR NOT ACCESSIBLE
0008 ????????????????   IDTPTR          DQ      ?       ; GDT DESCRIPTOR
0010 ????????????????   USER_DS         DQ      ?       ; IDT DESCRIPTOR
0018 ????????????????   USER_ES         DQ      ?       ; USER DATA SEGMENT DESCRIPTOR
0020 ????????????????   USER_SS         DQ      ?       ; USER EXTRA SEGMENT DESCRIPTOR
0028 ????????????????   USER_CS         DQ      ?       ; USER STACK SEGMENT DESCRIPTOR
0030 ????????????????   BIO_CS          DQ      ?       ; USER CODE SEGMENT DESCRIPTOR
0038 ????????????????   VIRTUAL_ENABLE_GDT_DEF  ENDS    ; TEMPORARY BIOS DESCRIPTOR
0040

                                ASSUME  DS:DATA

0408                    X_VIRTUAL       PROC    FAR
0408                    SET_VMODE:

                        ;-----  ENABLE ADDRESS LATCH BIT 20

0408 FA                         CLI                     ; NO INTERRUPTS ALLOWED
0409 B4 DF                      MOV     AH,ENABLE_BIT20 ; ENABLE BIT 20 FOR ADDRESS GATE
040B E8 03DA R                  CALL    GATE_A20
040E 3C 00                      CMP     AL,0            ; WAS THE COMMAND ACCEPTED?
0410 74 04                      JZ      BIT20_ON        ; GO IF YES
0412 B4 FF                      MOV     AH,0FFH         ; SET THE ERROR FLAG
0414 F9                         STC                     ; SET CARRY
0415 CF                         IRET                    ; EARLY EXIT

0416                    BIT20_ON:
0416 06                         PUSH    ES              ; MOVE SEGMENT POINTER
0417 1F                         POP     DS              ; TO THE DATA SEGMENT

                        ;-----------------------------------------------------------------------
                        ; REINITIALIZE THE 8259 INTERRUPT CONTROLLER #1 TO THE USER SPECIFIED OFFSET   :
                        ;-----------------------------------------------------------------------

0418 B0 11                      MOV     AL,11H          ; START INITIALIZATION SEQUENCE-ICW1
041A E6 20                      OUT     INTA00,AL       ; EDGE,INTERVAL-8,MASTER,ICW4 NEEDED
041C EB 00                      JMP     $+2
041E 8A C7                      MOV     AL,BH           ; HARDWARE INT'S START AT INT # (BH)
0420 E6 21                      OUT     INTA01,AL       ; SEND ICW2
0422 EB 00                      JMP     $+2
0424 B0 04                      MOV     AL,04H          ; SEND ICW3 - MASTER LEVEL 2
0426 E6 21                      OUT     INTA01,AL
0428 EB 00                      JMP     $+2
042A B0 01                      MOV     AL,01H          ; SEND ICW4 - MASTER,8086 MODE
042C E6 21                      OUT     INTA01,AL
042E EB 00                      JMP     $+2
0430 B0 FF                      MOV     AL,0FFH         ; MASK OFF ALL INTERRUPTS
0432 E6 21                      OUT     INTA01,AL

                        ;-----------------------------------------------------------------------
                        ; REINITIALIZE THE 8259 INTERRUPT CONTROLLER #2 TO THE USER SPECIFIED OFFSET   :
                        ;-----------------------------------------------------------------------

0434 B0 11                      MOV     AL,11H          ; INITIALIZE SEQUENCE-ICW1 FOR SLAVE
0436 E6 A0                      OUT     INTB00,AL       ; EDGE,INTERVAL-8,MASTER,ICW4 NEEDED
0438 EB 00                      JMP     $+2
043A 8A C3                      MOV     AL,BL           ; HARDWARE INT'S START AT INT # (BL)
043C E6 A1                      OUT     INTB01,AL       ; SEND ICW2
043E B0 02                      MOV     AL,02H
0440 EB 00                      JMP     $+2
0442 E6 A1                      OUT     INTB01,AL       ; SEND ICW3 - SLAVE LEVEL 2
0444 EB 00                      JMP     $+2
0446 B0 01                      MOV     AL,01H
0448 E6 A1                      OUT     INTB01,AL       ; SEND ICW4 - SLAVE,8086 MODE
044A EB 00                      JMP     $+2
044C B0 FF                      MOV     AL,0FFH
044E E6 A1                      OUT     INTB01,AL       ; MASK OFF ALL INTERRUPTS
```

SECTION 5

```
                        PAGE
                        ;--------------------------------------
                        ; SETUP BIOS CODE SEGMENT DESCRIPTOR   :
                        ;--------------------------------------

0450 C7 44 38 FFFF              MOV     [SI].BIO_CS.SEG_LIMIT,MAX_SEG_LEN       ; SET LENGTH
0456 C6 44 3C 0F                MOV     [SI].BIO_CS.BASE_HI_BYTE,CSEG@_HI       ; SET HIGH BYTE OF CS=0F
0459 C7 44 3A 0000              MOV     [SI].BIO_CS.BASE_LO_WORD,CSEG@_LO       ; SET LOW WORD OF CS=0
045E C6 44 3D 9B                MOV     [SI].BIO_CS.DATA_ACC_RIGHTS,CPL0_CODE_ACCESS
0462 C7 44 3E 0000              MOV     [SI].BIO_CS.DATA_RESERVED,0             ; ZERO RESERVED AREA

                        ;--------------------------------------
                        ;       ENABLE PROTECTED MODE           :
                        ;--------------------------------------
                                LGDT    [SI].GDTPTR             ; LOAD GLOBAL DESCRIPTOR TABLE REGISTER
0467 0F                 +       DB      00FH
0468                    + ??0005  LABEL   BYTE
0468 8B 54 08           +       MOV     DX,WORD PTR [SI].GDTPTR
046B                    + ??0006  LABEL   BYTE
0468                    +       ORG     OFFSET CS:??0005
0468 01                 +       DB      001H
046B                    +       ORG     OFFSET CS:??0006
                                LIDT    [SI].IDTPTR             ; INTERRUPT DESCRIPTOR TABLE REGISTER
046B 0F                 +       DB      00FH
046C                    + ??0007  LABEL   BYTE
046C 8B 5C 10           +       MOV     BX,WORD PTR [SI].IDTPTR
046F                    + ??0008  LABEL   BYTE
046C                    +       ORG     OFFSET CS:??0007
046C 01                 +       DB      001H
046F                    +       ORG     OFFSET CS:??0008

046F B8 0001                    MOV     AX,VIRTUAL_ENABLE       ; MACHINE STATUS WORD NEEDED TO
                                LMSW    AX                      ;  SWITCH TO VIRTUAL MODE
0472 0F 01 F0           +       DB      00FH,001H,0F0H
0475 EA                          DB      0EAH                   ; PURGE PRE-FETCH QUEUE WITH FAR JUMP
0476 047A R                      DW      OFFSET VMODE           ;  - TO OFFSET
0478 0038                        DW      BIO_CS                 ;  - IN SEGMENT -PROTECTED MODE SELECTOR

047A                    VMODE:
                        ;--------------------------------------
                        ;       SETUP USER SEGMENT REGISTERS    :
                        ;--------------------------------------
047A B8 0018                    MOV     AX,USER_DS              ; SETUP USER'S DATA SEGMENT
047D 8E D8                      MOV     DS,AX                   ; TO PROTECTED MODE SELECTORS
047F B8 0020                    MOV     AX,USER_ES              ; SETUP USER'S EXTRA SEGMENT
0482 8E C0                      MOV     ES,AX
0484 B8 0028                    MOV     AX,USER_SS              ; SETUP USER'S STACK SEGMENT
0487 8E D0                      MOV     SS,AX
                        ;--------------------------------------
                        ;       PUT TRANSFER ADDRESS ON STACK   :
                        ;       AND RETURN TO THE USER          :
                        ;--------------------------------------
0489 5B                         POP     BX                      ; GET RETURN IP FROM THE STACK
048A 83 C4 04                   ADD     SP,4                    ; NORMALIZE STACK POINTER
048D 6A 30                      PUSH    USER_CS                 ; SET STACK FOR A RETURN FAR
048F 53                         PUSH    BX
0490 CB                         RET                             ; RETURN TO USER IN VIRTUAL MODE

0491                    X_VIRTUAL       ENDP

                        ;--- DEVICE BUSY AND INTERRUPT COMPLETE ------------------------
                        ;                                                               :
                        ;       THIS ROUTINE IS A TEMPORARY HANDLER FOR DEVICE BUSY     :
                        ;       AND INTERRUPT COMPLETE                                  :
                        ;                                                               :
                        ;       INPUT   - SEE PROLOGUE                                  :
                        ;---------------------------------------------------------------

0491                    DEVICE_BUSY     PROC    NEAR
0491 F8                         CLC                             ; TURN CARRY OFF
0492 E9 0055 R                  JMP     CI_F                    ; RETURN WITH CARRY FLAG
0495                    DEVICE_BUSY     ENDP

0495                    INT_COMPLETE    PROC    NEAR
0495 CF                         IRET                            ; RETURN
0496                    INT_COMPLETE    ENDP

0496                    CODE    ENDS
                                END
```

```
                        PAGE 118,121
                        TITLE BIOS2 ---- 06/10/85  BIOS INTERRUPT ROUTINES
                        .286C
                        .LIST
0000                    CODE    SEGMENT BYTE PUBLIC

                                PUBLIC  PRINT_SCREEN_1
                                PUBLIC  RTC_INT
                                PUBLIC  TIME_OF_DAY_1
                                PUBLIC  TIMER_INT_1

                                EXTRN   CMOS_READ:NEAR          ; READ CMOS LOCATION ROUTINE
                                EXTRN   CMOS_WRITE:NEAR         ; WRITE CMOS LOCATION ROUTINE
                                EXTRN   DDS:NEAR                ; LOAD (DS) WITH DATA SEGMENT SELECTOR

                        ;--- INT  1A H -- (TIME_OF_DAY) ----------------------------------------------------
                        ;       THIS BIOS ROUTINE ALLOWS THE CLOCKS TO BE SET OR READ                      :
                        ;                                                                                  :
                        ; PARAMETERS:                                                                      :
                        ;     (AH) = 00H  READ THE CURRENT CLOCK SETTING AND RETURN WITH,                  :
                        ;                       (CX) = HIGH PORTION OF COUNT                               :
                        ;                       (DX) = LOW PORTION OF COUNT                                :
                        ;                       (AL) = 0 TIMER HAS NOT PASSED 24 HOURS SINCE LAST READ     :
                        ;                              1 IF ON ANOTHER DAY. (RESET TO ZERO AFTER READ)     :
                        ;                                                                                  :
                        ;     (AH) = 01H  SET THE CURRENT CLOCK USING,                                     :
                        ;                       (CX) = HIGH PORTION OF COUNT                               :
                        ;                       (DX) = LOW PORTION OF COUNT.                               :
                        ;                                                                                  :
                        ;                 NOTE: COUNTS OCCUR AT THE RATE OF 1193180/65536 COUNTS/SECOND    :
                        ;                             (OR ABOUT 18.2 PER SECOND -- SEE EQUATES)            :
                        ;                                                                                  :
                        ;     (AH) = 02H  READ THE REAL TIME CLOCK AND RETURN WITH,                        :
                        ;                       (CH) = HOURS IN BCD (00-23)                                :
                        ;                       (CL) = MINUTES IN BCD (00-59)                              :
                        ;                       (DH) = SECONDS IN BCD (00-59)                              :
                        ;                       (DL) = DAYLIGHT SAVINGS ENABLE (00-01).                    :
                        ;                                                                                  :
                        ;     (AH) = 03H  SET THE REAL TIME CLOCK USING,                                   :
                        ;                       (CH) = HOURS IN BCD (00-23)                                :
                        ;                       (CL) = MINUTES IN BCD (00-59)                              :
                        ;                       (DH) = SECONDS IN BCD (00-59)                              :
                        ;                       (DL) = 01 IF DAYLIGHT SAVINGS ENABLE OPTION, ELSE 00.      :
                        ;                                                                                  :
                        ;                 NOTE: (DL)= 00 IF DAYLIGHT SAVINGS TIME ENABLE IS NOT ENABLED.   :
                        ;                       (DL)= 01 ENABLES TWO SPECIAL UPDATES THE LAST SUNDAY IN     :
                        ;                       APRIL   (1:59:59 --> 3:00:00 AM) AND THE LAST SUNDAY IN    :
                        ;                       OCTOBER (1:59:59 --> 1:00:00 AM) THE FIRST TIME.           :
                        ;                                                                                  :
                        ;     (AH) = 04H  READ THE DATE FROM THE REAL TIME CLOCK AND RETURN WITH,           :
                        ;                       (CH) = CENTURY IN BCD (19 OR 20)                           :
                        ;                       (CL) = YEAR IN BCD (00-99)                                 :
                        ;                       (DH) = MONTH IN BCD (01-12)                                :
                        ;                       (DL) = DAY IN BCD (01-31).                                 :
                        ;                                                                                  :
                        ;     (AH) = 05H  SET THE DATE INTO THE REAL TIME CLOCK USING,                     :
                        ;                       (CH) = CENTURY IN BCD (19 OR 20)                           :
                        ;                       (CL) = YEAR IN BCD (00 - 99)                               :
                        ;                       (DH) = MONTH IN BCD (01 - 12)                              :
                        ;                       (DL) = DAY IN BCD (01-31).                                 :
                        ;                                                                                  :
                        ;     (AH) = 06H  SET THE ALARM TO INTERRUPT AT SPECIFIED TIME,                    :
                        ;                       (CH) = HOURS IN BCD (00-23 (OR FFH))                       :
                        ;                       (CL) = MINUTES IN BCD (00-59 (OR FFH))                     :
                        ;                       (DH) = SECONDS IN BCD (00-59 (OR FFH)).                    :
                        ;                                                                                  :
                        ;     (AH) = 07H  RESET THE ALARM INTERRUPT FUNCTION.                              :
                        ;                                                                                  :
                        ; NOTES: FOR ALL RETURNS CY= 0 FOR SUCCESSFUL OPERATION.                           :
                        ;        FOR (AH)= 2, 4, 6 - CARRY FLAG SET IF REAL TIME CLOCK NOT OPERATING.      :
                        ;        FOR (AH)= 6 - CARRY FLAG SET IF ALARM ALREADY ENABLED.                    :
                        ;        FOR THE ALARM FUNCTION (AH = 6) THE USER MUST SUPPLY A ROUTINE AND        :
                        ;          INTERCEPT THE CORRECT ADDRESS IN THE VECTOR TABLE FOR INTERRUPT 4AH.     :
                        ;          USE 0FFH FOR ANY "DO NOT CARE" POSITION FOR INTERVAL INTERRUPTS.        :
                        ;        INTERRUPTS ARE DISABLED DURING DATA MODIFICATION.                         :
                        ;        AH & AL ARE RETURNED MODIFIED AND NOT DEFINED EXCEPT WHERE INDICATED.     :
                        ;------------------------------------------------------------------------------------
                                ASSUME  CS:CODE,DS:DATA

0000                    TIME_OF_DAY_1   PROC    FAR
0000 FB                         STI                             ; INTERRUPTS BACK ON
0001 80 FC 08                   CMP     AH,(RTC_TBE-RTC_TB)/2   ; CHECK IF COMMAND IN VALID RANGE (0-7)
0004 F5                         CMC                             ; COMPLEMENT CARRY FOR ERROR EXIT
0005 72 17                      JC      TIME_9                  ; EXIT WITH CARRY = 1 IF NOT VALID

0007 1E                         PUSH    DS                      ; SAVE USERS (DS) SEGMENT
0008 E8 0000 E                  CALL    DDS                     ; GET DATA SEGMENT SELECTOR
000B 56                         PUSH    SI                      ; SAVE WORK REGISTER
000C C1 E8 08                   SHR     AX,8                    ; CONVERT FUNCTION TO BYTE OFFSET
000F 03 C0                      ADD     AX,AX                   ; CONVERT FUNCTION TO WORD OFFSET (CY=0)
0011 8B F0                      MOV     SI,AX                   ; PLACE INTO ADDRESSING REGISTER
0013 FA                         CLI                             ; NO INTERRUPTS DURING TIME FUNCTIONS
0014 2E: FF 94 0021 R           CALL    CS:[SI]+OFFSET RTC_TB   ; VECTOR TO FUNCTION REQUESTED WITH CY=0
                                                                ;  RETURN WITH CARRY FLAG SET FOR RESULT
0019 FB                         STI                             ; INTERRUPTS BACK ON
001A B4 00                      MOV     AH,0                    ; CLEAR (AH) TO ZERO
001C 5E                         POP     SI                      ; RECOVER USERS REGISTER
001D 1F                         POP     DS                      ; RECOVER USERS SEGMENT SELECTOR
001E                    TIME_9:                                 ; RETURN WITH CY= 0 IF NO ERROR
001E CA 0002                    RET     2


                                                                ;       ROUTINE VECTOR TABLE (AH)=
0021 0031 R             RTC_TB  DW      RTC_00                  ; 0 = READ CURRENT CLOCK COUNT
0023 0042 R                     DW      RTC_10                  ; 1 = SET CLOCK COUNT
0025 0050 R                     DW      RTC_20                  ; 2 = READ THE REAL TIME CLOCK TIME
0027 0075 R                     DW      RTC_30                  ; 3 = SET REAL TIME CLOCK TIME
0029 00A8 R                     DW      RTC_40                  ; 4 = READ THE REAL TIME CLOCK DATE
002B 00CB R                     DW      RTC_50                  ; 5 = SET REAL TIME CLOCK DATE
002D 0104 R                     DW      RTC_60                  ; 6 = SET THE REAL TIME CLOCK ALARM
002F 0145 R                     DW      RTC_70                  ; 7 = RESET ALARM
= 0031                  RTC_TBE EQU     $

0031                    TIME_OF_DAY_1   ENDP
```

SECTION 5

```
                        PAGE
0031                    RTC_00  PROC    NEAR                    ;       READ TIME COUNT
0031 A0 0070 R                  MOV     AL,@TIMER_OFL           ; GET THE OVERFLOW FLAG
0034 C6 06 0070 R 00            MOV     @TIMER_OFL,0            ; AND THEN RESET THE OVERFLOW FLAG
0039 8B 0E 006E R               MOV     CX,@TIMER_HIGH          ; GET COUNT OF TIME HIGH WORD
003D 8B 16 006C R               MOV     DX,@TIMER_LOW           ; GET COUNT OF TIME LOW WORD
0041 C3                         RET                             ; RETURN WITH NO CARRY
                
0042                    RTC_10:                                 ;       SET TIME COUNT
0042 89 16 006C R               MOV     @TIMER_LOW,DX           ; SET TIME COUNT LOW WORD
0046 89 0E 006E R               MOV     @TIMER_HIGH,CX          ; SET THE TIME COUNT HIGH WORD
004A C6 06 0070 R 00            MOV     @TIMER_OFL,0            ; RESET OVERFLOW FLAG
004F C3                         RET                             ; RETURN WITH NO CARRY
                
0050                    RTC_20:                                ;       GET RTC TIME
0050 E8 016B R                  CALL    UPD_IPR                 ; CHECK FOR UPDATE IN PROCESS
0053 72 1F                      JC      RTC_29                  ; EXIT IF ERROR (CY= 1)
                
0055 B0 00                      MOV     AL,CMOS_SECONDS         ; SET ADDRESS OF SECONDS
0057 E8 0000 E                  CALL    CMOS_READ               ; GET SECONDS
005A 8A F0                      MOV     DH,AL                   ; SAVE
005C B0 0B                      MOV     AL,CMOS_REG_B           ; ADDRESS ALARM REGISTER
005E E8 0000 E                  CALL    CMOS_READ               ; READ CURRENT VALUE OF DSE BIT
0061 24 01                      AND     AL,00000001B            ; MASK FOR VALID DSE BIT
0063 8A D0                      MOV     DL,AL                   ; SET (DL) TO ZERO FOR NO DSE BIT
0065 B0 02                      MOV     AL,CMOS_MINUTES         ; SET ADDRESS OF MINUTES
0067 E8 0000 E                  CALL    CMOS_READ               ; GET MINUTES
006A 8A C8                      MOV     CL,AL                   ; SAVE
006C B0 04                      MOV     AL,CMOS_HOURS           ; SET ADDRESS OF HOURS
006E E8 0000 E                  CALL    CMOS_READ               ; GET HOURS
0071 8A E8                      MOV     CH,AL                   ; SAVE
0073 F8                         CLC                             ; SET CY= 0
0074                    RTC_29: 
0074 C3                         RET                             ; RETURN WITH RESULT IN CARRY FLAG
                
0075                    RTC_30:                                ;       SET RTC TIME
0075 E8 016B R                  CALL    UPD_IPR                 ; CHECK FOR UPDATE IN PROCESS
0078 73 03                      JNC     RTC_35                  ; GO AROUND IF CLOCK OPERATING
007A E8 0154 R                  CALL    RTC_STA                 ; ELSE TRY INITIALIZING CLOCK
007D                    RTC_35: 
007D 8A E6                      MOV     AH,DH                   ; GET TIME BYTE - SECONDS
007F B0 00                      MOV     AL,CMOS_SECONDS         ; ADDRESS SECONDS
0081 E8 0000 E                  CALL    CMOS_WRITE              ; UPDATE SECONDS
0084 8A E1                      MOV     AH,CL                   ; GET TIME BYTE - MINUTES
0086 B0 02                      MOV     AL,CMOS_MINUTES         ; ADDRESS MINUTES
0088 E8 0000 E                  CALL    CMOS_WRITE              ; UPDATE MINUTES
008B 8A E5                      MOV     AH,CH                   ; GET TIME BYTE - HOURS
008D B0 04                      MOV     AL,CMOS_HOURS           ; ADDRESS HOURS
008F E8 0000 E                  CALL    CMOS_WRITE              ; UPDATE ADDRESS
0092 B8 0B0B                    MOV     AX,X*CMOS_REG_B         ; ADDRESS ALARM REGISTER
0095 E8 0000 E                  CALL    CMOS_READ               ; READ CURRENT VALUE
0098 24 62                      AND     AL,01100010B            ; MASK FOR VALID BIT POSITIONS
009A 0C 02                      OR      AL,00000010B            ; TURN ON 24 HOUR MODE
009C 80 E2 01                   AND     DL,00000001B            ; USE ONLY THE DSE BIT
009F 0A C2                      OR      AL,DL                   ; GET DAY LIGHT SAVINGS TIME BIT (DSE)
00A1 86 E0                      XCHG    AH,AL                   ; PLACE IN WORK REGISTER AND GET ADDRESS
00A3 E8 0000 E                  CALL    CMOS_WRITE              ; SET NEW ALARM BITS
00A6 F8                         CLC                             ; SET CY= 0
00A7 C3                         RET                             ; RETURN WITH CY= 0
                
00A8                    RTC_40:                                ;       GET RTC DATE
00A8 E8 016B R                  CALL    UPD_IPR                 ; CHECK FOR UPDATE IN PROCESS
00AB 72 1D                      JC      RTC_49                  ; EXIT IF ERROR (CY= 1)
                
00AD B0 07                      MOV     AL,CMOS_DAY_MONTH       ; ADDRESS DAY OF MONTH
00AF E8 0000 E                  CALL    CMOS_READ               ; READ DAY OF MONTH
00B2 8A D0                      MOV     DL,AL                   ; SAVE
00B4 B0 08                      MOV     AL,CMOS_MONTH           ; ADDRESS MONTH
00B6 E8 0000 E                  CALL    CMOS_READ               ; READ MONTH
00B9 8A F0                      MOV     DH,AL                   ; SAVE
00BB B0 09                      MOV     AL,CMOS_YEAR            ; ADDRESS YEAR
00BD E8 0000 E                  CALL    CMOS_READ               ; READ YEAR
00C0 8A C8                      MOV     CL,AL                   ; SAVE
00C2 B0 32                      MOV     AL,CMOS_CENTURY         ; ADDRESS CENTURY LOCATION
00C4 E8 0000 E                  CALL    CMOS_READ               ; GET CENTURY BYTE
00C7 8A E8                      MOV     CH,AL                   ; SAVE
00C9 F8                         CLC                             ; SET CY=0
00CA                    RTC_49: 
00CA C3                         RET                             ; RETURN WITH RESULTS IN CARRY FLAG
                
00CB                    RTC_50:                                ;       SET RTC DATE
00CB E8 016B R                  CALL    UPD_IPR                 ; CHECK FOR UPDATE IN PROCESS
00CE 73 03                      JNC     RTC_55                  ; GO AROUND IF NO ERROR
00D0 E8 0154 R                  CALL    RTC_STA                 ; ELSE INITIALIZE CLOCK
00D3                    RTC_55: 
00D3 B8 0006                    MOV     AX,CMOS_DAY_WEEK        ; ADDRESS OF DAY OF WEEK BYTE
00D6 E8 0000 E                  CALL    CMOS_WRITE              ; LOAD ZEROS TO DAY OF WEEK
00D9 8A E2                      MOV     AH,DL                   ; GET DAY OF MONTH BYTE
00DB B0 07                      MOV     AL,CMOS_DAY_MONTH       ; ADDRESS DAY OF MONTH BYTE
00DD E8 0000 E                  CALL    CMOS_WRITE              ; WRITE OF DAY OF MONTH REGISTER
00E0 8A E6                      MOV     AH,DH                   ; GET MONTH
00E2 B0 08                      MOV     AL,CMOS_MONTH           ; ADDRESS MONTH BYTE
00E4 E8 0000 E                  CALL    CMOS_WRITE              ; WRITE MONTH REGISTER
00E7 8A E1                      MOV     AH,CL                   ; GET YEAR BYTE
00E9 B0 09                      MOV     AL,CMOS_YEAR            ; ADDRESS YEAR REGISTER
00EB E8 0000 E                  CALL    CMOS_WRITE              ; WRITE YEAR REGISTER
00EE 8A E5                      MOV     AH,CH                   ; GET CENTURY BYTE
00F0 B0 32                      MOV     AL,CMOS_CENTURY         ; ADDRESS CENTURY BYTE
00F2 E8 0000 E                  CALL    CMOS_WRITE              ; WRITE CENTURY LOCATION
00F5 B8 0B0B                    MOV     AX,X*CMOS_REG_B         ; ADDRESS ALARM REGISTER
00F8 E8 0000 E                  CALL    CMOS_READ               ; READ CURRENT SETTINGS
00FB 24 7F                      AND     AL,07FH                 ; CLEAR 'SET BIT'
00FD 86 E0                      XCHG    AH,AL                   ; MOVE TO WORK REGISTER
00FF E8 0000 E                  CALL    CMOS_WRITE              ;  AND START CLOCK UPDATING
0102 F8                         CLC                             ; SET CY= 0
0103 C3                         RET                             ; RETURN CY=0
```

```
                        PAGE
0104                    RTC_60:                         ;       SET RTC ALARM
0104 B0 0B                      MOV     AL,CMOS_REG_B   ; ADDRESS ALARM
0106 E8 0000 E                  CALL    CMOS_READ       ; READ ALARM REGISTER
0109 A8 20                      TEST    AL,20H          ; CHECK FOR ALARM ALREADY ENABLED
010B F9                         STC                     ; SET CARRY IN CASE OF ERROR
010C 75 33                      JNZ     RTC_69          ; ERROR EXIT IF ALARM SET
                        
010E E8 016B R                  CALL    UPD_IPR         ; CHECK FOR UPDATE IN PROCESS
0111 73 03                      JNC     RTC_65          ; SKIP INITIALIZATION IF NO ERROR
0113 E8 0154 R                  CALL    RTC_STA         ; ELSE INITIALIZE CLOCK
0116                    RTC_65:
0116 8A E6                      MOV     AH,DH           ; GET SECONDS BYTE
0118 B0 01                      MOV     AL,CMOS_SEC_ALARM       ; ADDRESS THE SECONDS ALARM REGISTER
011A E8 0000 E                  CALL    CMOS_WRITE      ; INSERT SECONDS
011D 8A E1                      MOV     AH,CL           ; GET MINUTES PARAMETER
011F B0 03                      MOV     AL,CMOS_MIN_ALARM       ; ADDRESS MINUTES ALARM REGISTER
0121 E8 0000 E                  CALL    CMOS_WRITE      ; INSERT MINUTES
0124 8A E5                      MOV     AH,CH           ; GET HOURS PARAMETER
0126 B0 05                      MOV     AL,CMOS_HR_ALARM        ; ADDRESS HOUR ALARM REGISTER
0128 E8 0000 E                  CALL    CMOS_WRITE      ; INSERT HOURS
012B E4 A1                      IN      AL,INTB01       ; READ SECOND INTERRUPT MASK REGISTER
012D 24 FE                      AND     AL,0FEH         ; ENABLE ALARM TIMER BIT (CY= 0)
012F E6 A1                      OUT     INTB01,AL       ; WRITE UPDATED MASK
0131 B8 0B0B                    MOV     AX,X*CMOS_REG_B ; ADDRESS ALARM REGISTER
0134 E8 0000 E                  CALL    CMOS_READ       ; READ CURRENT ALARM REGISTER
0137 24 7F                      AND     AL,07FH         ; ENSURE SET BIT TURNED OFF
0139 0C 20                      OR      AL,20H          ; TURN ON ALARM ENABLE
013B 86 E0                      XCHG    AH,AL           ; MOVE MASK TO OUTPUT REGISTER
013D E8 0000 E                  CALL    CMOS_WRITE      ; WRITE NEW ALARM MASK
0140 F8                         CLC                     ; SET CY= 0
0141                    RTC_69:
0141 B8 0000                    MOV     AX,0            ; CLEAR AX REGISTER
0144 C3                         RET                     ; RETURN WITH RESULTS IN CARRY FLAG
                        
0145                    RTC_70:                         ;       RESET ALARM
0145 B8 0B0B                    MOV     AX,X*CMOS_REG_B ; ADDRESS ALARM REGISTER (TO BOTH AH,AL)
0148 E8 0000 E                  CALL    CMOS_READ       ; READ ALARM REGISTER
014B 24 57                      AND     AL,57H          ; TURN OFF ALARM ENABLE
014D 86 E0                      XCHG    AH,AL           ; SAVE DATA AND RECOVER ADDRESS
014F E8 0000 E                  CALL    CMOS_WRITE      ; RESTORE NEW VALUE
0152 F8                         CLC                     ; SET CY= 0
0153 C3                         RET                     ; RETURN WITH NO CARRY
                        
0154                    RTC_00  ENDP
                        
0154                    RTC_STA PROC    NEAR            ;       INITIALIZE REAL TIME CLOCK
0154 B8 260A                    MOV     AX,26H*H+CMOS_REG_A     ; ADDRESS REGISTER A AND LOAD DATA MASK
0157 E8 0000 E                  CALL    CMOS_WRITE      ; INITIALIZE STATUS REGISTER A
015A B8 820B                    MOV     AX,82H*H+CMOS_REG_B     ; SET "SET BIT" FOR CLOCK INITIALIZATION
015D E8 0000 E                  CALL    CMOS_WRITE      ; AND 24 HOUR MODE TO REGISTER B
0160 B0 0C                      MOV     AL,CMOS_REG_C   ; ADDRESS REGISTER C
0162 E8 0000 E                  CALL    CMOS_READ       ; READ REGISTER C TO INITIALIZE
0165 B0 0D                      MOV     AL,CMOS_REG_D   ; ADDRESS REGISTER D
0167 E8 0000 E                  CALL    CMOS_READ       ; READ REGISTER D TO INITIALIZE
016A C3                         RET
                        
016B                    RTC_STA ENDP
                        
016B                    UPD_IPR PROC    NEAR            ;       WAIT TILL UPDATE NOT IN PROGRESS
016B 51                         PUSH    CX              ; SAVE CALLERS REGISTER
016C B9 0320                    MOV     CX,800          ; SET TIMEOUT LOOP COUNT
016F                    UPD_10:
016F B0 0A                      MOV     AL,CMOS_REG_A   ; ADDRESS STATUS REGISTER A
0171 FA                         CLI                     ; NO TIMER INTERRUPTS DURING UPDATES
0172 E8 0000 E                  CALL    CMOS_READ       ; READ UPDATE IN PROCESS FLAG
0175 A8 80                      TEST    AL,80H          ; IF UIP BIT IS ON ( CANNOT READ TIME )
0177 74 06                      JZ      UPD_90          ; EXIT WITH CY= 0 IF CAN READ CLOCK NOW
0179 FB                         STI                     ; ALLOW INTERRUPTS WHILE WAITING
017A E2 F3                      LOOP    UPD_10          ; LOOP TILL READY OR TIMEOUT
017C 33 C0                      XOR     AX,AX           ; CLEAR RESULTS IF ERROR
017E F9                         STC                     ; SET CARRY FOR ERROR
017F                    UPD_90:
017F 59                         POP     CX              ; RESTORE CALLERS REGISTER
0180 FA                         CLI                     ; INTERRUPTS OFF DURING SET
0181 C3                         RET                     ; RETURN WITH CY FLAG SET
                        
0182                    UPD_IPR ENDP
```

```
                        PAGE
                        ;--- HARDWARE INT  70 H -- ( IRQ LEVEL  8 ) ------------------------------------
                        ; ALARM INTERRUPT HANDLER (RTC)                                               :
                        ;       THIS ROUTINE HANDLES THE PERIODIC AND ALARM INTERRUPTS FROM THE CMOS   :
                        ;       TIMER.  INPUT FREQUENCY IS 1.024 KHZ OR APPROXIMATELY 1024 INTERRUPTS :
                        ;       EVERY SECOND FOR THE PERIODIC INTERRUPT.  FOR THE ALARM FUNCTION,      :
                        ;       THE INTERRUPT WILL OCCUR AT THE DESIGNATED TIME.                       :
                        ;                                                                              :
                        ;       INTERRUPTS ARE ENABLED WHEN THE EVENT OR ALARM FUNCTION IS ACTIVATED.  :
                        ;       FOR THE EVENT INTERRUPT, THE HANDLER WILL DECREMENT THE WAIT COUNTER    :
                        ;       AND WHEN IT EXPIRES WILL SET THE DESIGNATED LOCATION TO 80H.  FOR       :
                        ;       THE ALARM INTERRUPT, THE USER MUST PROVIDE A ROUTINE TO INTERCEPT       :
                        ;       THE CORRECT ADDRESS FROM THE VECTOR TABLE INVOKED BY INTERRUPT 4AH      :
                        ;       PRIOR TO SETTING THE REAL TIME CLOCK ALARM (INT 1AH, AH= 06H).         :
                        ;------------------------------------------------------------------------------

0182                    RTC_INT PROC    FAR                     ;       ALARM INTERRUPT
0182 1E                         PUSH    DS                      ; LEAVE INTERRUPTS DISABLED
0183 50                         PUSH    AX                      ; SAVE REGISTERS
0184 57                         PUSH    DI

0185                    RTC_I_1:                                ;       CHECK FOR SECOND INTERRUPT
0185 B8 8B8C                    MOV     AX,(CMOS_REG_B+NMI)*H+CMOS_REG_C+NMI ; ALARM AND STATUS
0188 E6 70                      OUT     CMOS_PORT,AL            ; WRITE ALARM FLAG MASK ADDRESS
018A 90                         NOP                             ; I/O DELAY
018B E4 71                      IN      AL,CMOS_DATA            ; READ AND RESET INTERRUPT REQUEST FLAGS
018D A8 60                      TEST    AL,01100000B            ; CHECK FOR EITHER INTERRUPT PENDING
018F 74 5C                      JZ      RTC_I_9                 ; EXIT IF NOT A VALID RTC INTERRUPT

0191 86 E0                      XCHG    AH,AL                   ; SAVE FLAGS AND GET ENABLE ADDRESS
0193 E6 70                      OUT     CMOS_PORT,AL            ; WRITE ALARM ENABLE MASK ADDRESS
0195 90                         NOP                             ; I/O DELAY
0196 E4 71                      IN      AL,CMOS_DATA            ; READ CURRENT ALARM ENABLE MASK
0198 22 C4                      AND     AL,AH                   ; ALLOW ONLY SOURCES THAT ARE ENABLED
019A A8 40                      TEST    AL,01000000B            ; CHECK FOR PERIODIC INTERRUPT
019C 74 3F                      JZ      RTC_I_5                 ; SKIP IF NOT A PERIODIC INTERRUPT

                        ;-----  DECREMENT WAIT COUNT BY INTERRUPT INTERVAL

019E E8 0000 E                  CALL    DDS                     ; ESTABLISH DATA SEGMENT ADDRESSABILITY
01A1 81 2E 009C R 03D0          SUB     @RTC_LOW,0976           ; DECREMENT COUNT LOW BY 1/1024
01A7 83 1E 009E R 00            SBB     @RTC_HIGH,0             ; ADJUST HIGH WORD FOR LOW WORD BORROW
01AC 73 2F                      JNC     RTC_T_5                 ; SKIP TILL 32 BIT WORD LESS THAN ZERO

                        ;-----  TURN OFF PERIODIC INTERRUPT ENABLE

01AE 50                         PUSH    AX                      ; SAVE INTERRUPT FLAG MASK
01AF B8 8B8B                    MOV     AX,X*(CMOS_REG_B+NMI)   ; INTERRUPT ENABLE REGISTER
01B2 E6 70                      OUT     CMOS_PORT,AL            ; WRITE ADDRESS TO CMOS CLOCK
01B4 90                         NOP                             ; I/O DELAY
01B5 E4 71                      IN      AL,CMOS_DATA            ; READ CURRENT ENABLES
01B7 24 BF                      AND     AL,0BFH                 ; TURN OFF PIE
01B9 86 C4                      XCHG    AL,AH                   ; GET CMOS ADDRESS AND SAVE VALUE
01BB E6 70                      OUT     CMOS_PORT,AL            ; ADDRESS REGISTER B
01BD 86 C4                      XCHG    AL,AH                   ; GET NEW INTERRUPT ENABLE MASK
01BF E6 71                      OUT     CMOS_DATA,AL            ; SET MASK IN INTERRUPT ENABLE REGISTER
01C1 58                         POP     AX                      ; GET INTERRUPT SOURCE BACK
01C2 F6 06 00A0 R 02            TEST    @RTC_WAIT_FLAG,02H      ; CHECK FOR "WAIT" FUNCTION ACTIVE
01C7 C6 06 00A0 R 00            MOV     @RTC_WAIT_FLAG,0        ;  SET FUNCTION ACTIVE FLAGS OFF
01CC C5 3E 0098 R               LDS     DI,DWORD PTR @USER_FLAG ;  SET UP (DS:DI) TO POINT TO USER FLAG
01D0 C6 05 80                   MOV     BYTE PTR [DI],80H       ;  TURN ON USERS POSTED FLAG
01D3 74 08                      JZ      RTC_I_5                 ;  SKIP IF "EVENT_WAIT" FUNCTION

01D5 E8 0000 E                  CALL    DDS                     ; ESTABLISH DATA SEGMENT ADDRESSABILITY
01D8 C6 06 00A0 R 83            MOV     @RTC_WAIT_FLAG,083H     ; AND SET "WAIT" BACK TO BUSY & POSTED
01DD                    RTC_I_5:
01DD A8 20                      TEST    AL,00100000B            ; TEST FOR ALARM INTERRUPT
01DF 74 0A                      JZ      RTC_I_7                 ; SKIP USER INTERRUPT CALL IF NOT ALARM

01E1 B0 0F                      MOV     AL,CMOS_SHUT_DOWN       ; POINT TO DEFAULT READ ONLY REGISTER
01E3 E6 70                      OUT     CMOS_PORT,AL            ; ENABLE NMI AND CMOS ADDRESS TO DEFAULT
01E5 FB                         STI                             ; INTERRUPTS BACK ON NOW
01E6 52                         PUSH    DX
01E7 CD 4A                      INT     4AH                     ; TRANSFER TO USER ROUTINE
01E9 5A                         POP     DX
01EA FA                         CLI                             ; BLOCK INTERRUPT FOR RETRY
01EB                    RTC_I_7:                                ; RESTART ROUTINE TO HANDLE DELAYED
01EB EB 98                      JMP     RTC_I_1                 ;  ENTRY AND SECOND EVENT BEFORE DONE

01ED                    RTC_I_9:                                ;       EXIT - NO PENDING INTERRUPTS
01ED B0 0F                      MOV     AL,CMOS_SHUT_DOWN       ; POINT TO DEFAULT READ ONLY REGISTER
01EF E6 70                      OUT     CMOS_PORT,AL            ; ENABLE NMI AND CMOS ADDRESS TO DEFAULT
01F1 90                         NOP                             ; I/O DELAY
01F2 E4 71                      IN      AL,CMOS_DATA            ; OPEN STANDBY LATCH
01F4 B0 20                      MOV     AL,EOI                  ; END OF INTERRUPT MASK TO 8259 - 2
01F6 E6 A0                      OUT     INTB00,AL               ;  TO 8259 - 2
01F8 E6 20                      OUT     INTA00,AL               ;  TO 8259 - 1
01FA 5F                         POP     DI                      ; RESTORE REGISTERS
01FB 58                         POP     AX
01FC 1F                         POP     DS
01FD CF                         IRET                            ; END OF INTERRUPT

01FE                    RTC_INT ENDP
```

```
                        PAGE
                        ;--- INT  05 H ----------------------------------------------------------------
                        ; PRINT_SCREEN                                                               :
                        ;       THIS LOGIC WILL BE INVOKED BY INTERRUPT 05H TO PRINT THE SCREEN.     :
                        ;       THE CURSOR POSITION AT THE TIME THIS ROUTINE IS INVOKED WILL BE      :
                        ;       SAVED AND RESTORED UPON COMPLETION.  THE ROUTINE IS INTENDED TO      :
                        ;       RUN WITH INTERRUPTS ENABLED.   IF A SUBSEQUENT PRINT SCREEN KEY      :
                        ;       IS DEPRESSED WHILE THIS ROUTINE IS PRINTING IT WILL BE IGNORED.      :
                        ;       THE BASE PRINTERS STATUS IS CHECKED FOR NOT BUSY AND NOT OUT OF      :
                        ;       PAPER.  AN INITIAL STATUS ERROR WILL ABEND THE PRINT REQUEST.        :
                        ;       ADDRESS  0050:0000  CONTAINS THE STATUS OF THE PRINT SCREEN:         :
                        ;                                                                            :
                        ;       50:0    = 0     PRINT SCREEN HAS NOT BEEN CALLED OR UPON RETURN      :
                        ;                        FROM A CALL THIS INDICATES A SUCCESSFUL OPERATION.  :
                        ;               = 1     PRINT SCREEN IS IN PROGRESS - IGNORE THIS REQUEST.   :
                        ;               = 255   ERROR ENCOUNTERED DURING PRINTING.                   :
                        ;--------------------------------------------------------------------------
                        
 01FE                   PRINT_SCREEN_1  PROC   FAR
                                                        ; DELAY INTERRUPT ENABLE TILL FLAG SET
 01FE 1E                        PUSH    DS
 01FF 50                        PUSH    AX              ; SAVE WORK REGISTERS
 0200 53                        PUSH    BX
 0201 51                        PUSH    CX
 0202 52                        PUSH    DX              ; USE 0040:0100 FOR STATUS AREA STORAGE
 0203 E8 0000 E                 CALL    DDS             ; GET STATUS_BYTE DATA SEGMENT
 0206 80 3E 0100 R 01           CMP     @STATUS_BYTE,1  ; SEE IF PRINT ALREADY IN PROGRESS
 020B 74 74                     JE      PRI90           ; EXIT IF PRINT ALREADY IN PROGRESS
 020D C6 06 0100 R 01           MOV     @STATUS_BYTE,1  ; INDICATE PRINT NOW IN PROGRESS
 0212 FB                        STI                     ; MUST RUN WITH INTERRUPTS ENABLED
 0213 B4 0F                     MOV     AH,0FH          ; WILL REQUEST THE CURRENT SCREEN MODE
 0215 CD 10                     INT     10H             ;       (AL)= MODE
                                                        ;       (AH)= NUMBER COLUMNS/LINE
                                                        ;       (BH)= VISUAL PAGE
 0217 8A CC                     MOV     CL,AH           ; WILL MAKE USE OF (CX) REGISTER TO
 0219 8A 2E 0084 R              MOV     CH,@ROWS        ;  CONTROL ROWS ON SCREEN & COLUMNS
 021D FE C5                     INC     CH              ; ADJUST ROWS ON DISPLAY COUNT
                                                        ;       (CL)= NUMBER COLUMNS/LINE
                                                        ;       (CH)= NUMBER OF ROWS ON DISPLAY
                                ;---------------------------------------------------------
                                ;       AT THIS POINT WE KNOW THE COLUMNS/LINE COUNT IS IN (CL) :
                                ;       AND THE NUMBER OF ROWS ON THE DISPLAY IS IN (CH).       :
                                ;       THE PAGE IF APPLICABLE IS IN (BH).  THE STACK HAS       :
                                ;       (DS),(AX),(BX),(CX),(DX)  PUSHED.                       :
                                ;---------------------------------------------------------
 021F 33 D2                     XOR     DX,DX           ; FIRST PRINTER
 0221 B4 02                     MOV     AH,02H          ; SET PRINTER STATUS REQUEST COMMAND
 0223 CD 17                     INT     17H             ; REQUEST CURRENT PRINTER STATUS
 0225 80 F4 80                  XOR     AH,080H         ; CHECK FOR PRINTER BUSY (NOT CONNECTED)
 0228 F6 C4 A0                  TEST    AH,0A0H         ;  OR  OUT OF PAPER
 022B 75 4E                     JNZ     PRI80           ; ERROR EXIT IF PRINTER STATUS ERROR

 022D E8 0287 R                 CALL    CRLF            ; CARRIAGE RETURN LINE FEED TO PRINTER

 0230 51                        PUSH    CX              ; SAVE SCREEN BOUNDS
 0231 B4 03                     MOV     AH,03H          ; NOW READ THE CURRENT CURSOR POSITION
 0233 CD 10                     INT     10H             ; AND RESTORE AT END OF ROUTINE
 0235 59                        POP     CX              ; RECALL SCREEN BOUNDS
 0236 52                        PUSH    DX              ; PRESERVE THE ORIGINAL POSITION
 0237 33 D2                     XOR     DX,DX           ; INITIAL CURSOR (0,0) AND FIRST PRINTER
                                ;---------------------------------------------------------
                                ;       THIS LOOP IS TO READ EACH CURSOR POSITION FROM THE      :
                                ;       SCREEN AND PRINT IT.  (BH)= VISUAL PAGE  (CH)= ROWS     :
                                ;---------------------------------------------------------
 0239                   PRI10:
 0239 B4 02                     MOV     AH,02H          ; INDICATE CURSOR SET REQUEST
 023B CD 10                     INT     10H             ; NEW CURSOR POSITION ESTABLISHED
 023D B4 08                     MOV     AH,08H          ; INDICATE READ CHARACTER FROM DISPLAY
 023F CD 10                     INT     10H             ; CHARACTER NOW IN (AL)
 0241 0A C0                     OR      AL,AL           ; SEE IF VALID CHAR
 0243 75 02                     JNZ     PRI20           ; JUMP IF VALID CHAR
 0245 B0 20                     MOV     AL,' '          ; ELSE MAKE IT A BLANK
 0247                   PRI20:
 0247 52                        PUSH    DX              ; SAVE CURSOR POSITION
 0248 33 D2                     XOR     DX,DX           ; INDICATE FIRST PRINTER (DX= 0)
 024A 32 E4                     XOR     AH,AH           ; INDICATE PRINT CHARACTER IN (AL)
 024C CD 17                     INT     17H             ; PRINT THE CHARACTER
 024E 5A                        POP     DX              ; RECALL CURSOR POSITION
 024F F6 C4 29                  TEST    AH,29H          ; TEST FOR PRINTER ERROR
 0252 75 22                     JNZ     PRI70           ; EXIT IF ERROR DETECTED
 0254 FE C2                     INC     DL              ; ADVANCE TO NEXT COLUMN
 0256 3A CA                     CMP     CL,DL           ; SEE IF AT END OF LINE
 0258 75 DF                     JNZ     PRI10           ; IF NOT LOOP FOR NEXT COLUMN
 025A 32 D2                     XOR     DL,DL           ; BACK TO COLUMN 0
 025C 8A E2                     MOV     AH,DL           ; (AH)=0
 025E 52                        PUSH    DX              ; SAVE NEW CURSOR POSITION
 025F E8 0287 R                 CALL    CRLF            ; LINE FEED CARRIAGE RETURN
 0262 5A                        POP     DX              ; RECALL CURSOR POSITION
 0263 FE C6                     INC     DH              ; ADVANCE TO NEXT LINE
 0265 3A EE                     CMP     CH,DH           ; FINISHED?
 0267 75 D0                     JNZ     PRI10           ; IF NOT LOOP FOR NEXT LINE

 0269 5A                        POP     DX              ; GET CURSOR POSITION
 026A B4 02                     MOV     AH,02H          ; INDICATE REQUEST CURSOR SET
 026C CD 10                     INT     10H             ; CURSOR POSITION RESTORED
 026E FA                        CLI                     ; BLOCK INTERRUPTS TILL STACK CLEARED
 026F C6 06 0100 R 00           MOV     @STATUS_BYTE,0  ; MOVE OK RESULTS FLAG TO STATUS_BYTE
 0274 EB 0B                     JMP     SHORT PRI90     ; EXIT PRINTER ROUTINE

 0276                   PRI70:                          ;       ERROR EXIT
 0276 5A                        POP     DX              ; GET CURSOR POSITION
 0277 B4 02                     MOV     AH,02H          ; INDICATE REQUEST CURSOR SET
 0279 CD 10                     INT     10H             ; CURSOR POSITION RESTORED
 027B                   PRI80:
 027B FA                        CLI                     ; BLOCK INTERRUPTS TILL STACK CLEARED
 027C C6 06 0100 R FF           MOV     @STATUS_BYTE,0FFH ; SET ERROR FLAG
 0281                   PRI90:
 0281 5A                        POP     DX              ;       EXIT ROUTINE
 0282 59                        POP     CX              ; RESTORE ALL THE REGISTERS USED
 0283 5B                        POP     BX
 0284 58                        POP     AX
 0285 1F                        POP     DS
 0286 CF                        IRET                    ; RETURN WITH INITIAL INTERRUPT MASK
 0287                   PRINT_SCREEN_1  ENDP
```

SECTION 5

```
                        ;-----  CARRIAGE RETURN, LINE FEED SUBROUTINE

0287                    CRLF    PROC    NEAR
                                                        ;       SEND CR,LF TO FIRST PRINTER
0287 33 D2                      XOR     DX,DX           ; ASSUME FIRST PRINTER (DX= 0)
0289 B8 000D                    MOV     AX,CR           ; GET THE PRINT CHARACTER COMMAND AND
028C CD 17                      INT     17H             ;   THE CARRIAGE RETURN CHARACTER
028E B8 000A                    MOV     AX,LF           ; NOW GET THE LINE FEED AND
0291 CD 17                      INT     17H             ;   SEND IT TO THE BIOS PRINTER ROUTINE
0293 C3                         RET
0294                    CRLF    ENDP


                        ;-- HARDWARE INT  08 H -- ( IRQ LEVEL 0 ) ------------------------------------
                        ;                                                                              :
                        ;       THIS ROUTINE HANDLES THE TIMER INTERRUPT FROM FROM CHANNEL 0 OF         :
                        ;       THE 8254 TIMER.  INPUT FREQUENCY IS 1.19318 MHZ AND THE DIVISOR        :
                        ;       IS 65536, RESULTING IN APPROXIMATELY 18.2 INTERRUPTS EVERY SECOND.     :
                        ;                                                                              :
                        ;       THE INTERRUPT HANDLER MAINTAINS A COUNT (40:6C) OF INTERRUPTS SINCE     :
                        ;       POWER ON TIME, WHICH MAY BE USED TO ESTABLISH TIME OF DAY.             :
                        ;       THE INTERRUPT HANDLER ALSO DECREMENTS THE MOTOR CONTROL COUNT (40:40)   :
                        ;       OF THE DISKETTE, AND WHEN IT EXPIRES, WILL TURN OFF THE                :
                        ;       DISKETTE MOTOR(s), AND RESET THE MOTOR RUNNING FLAGS.                  :
                        ;       THE INTERRUPT HANDLER WILL ALSO INVOKE A USER ROUTINE THROUGH           :
                        ;       INTERRUPT 1CH AT EVERY TIME TICK.  THE USER MUST CODE A                :
                        ;       ROUTINE AND PLACE THE CORRECT ADDRESS IN THE VECTOR TABLE.              :
                        ;------------------------------------------------------------------------------

0294                    TIMER_INT_1     PROC    FAR
0294 FB                         STI                     ; INTERRUPTS BACK ON
0295 1E                         PUSH    DS
0296 50                         PUSH    AX
0297 52                         PUSH    DX              ; SAVE MACHINE STATE
0298 E8 0000 E                  CALL    DDS             ; ESTABLISH ADDRESSABILITY
029B FF 06 006C R               INC     @TIMER_LOW      ; INCREMENT TIME
029F 75 04                      JNZ     T4              ; GO TO TEST_DAY
02A1 FF 06 006E R               INC     @TIMER_HIGH     ; INCREMENT HIGH WORD OF TIME
02A5                    T4:                             ;       TEST_DAY
02A5 83 3E 006E R 18            CMP     @TIMER_HIGH,018H ; TEST FOR COUNT EQUALING 24 HOURS
02AA 75 15                      JNZ     T5              ; GO TO DISKETTE_CTL
02AC 81 3E 006C R 00B0          CMP     @TIMER_LOW,0B0H
02B2 75 0D                      JNZ     T5              ; GO TO DISKETTE_CTL

                        ;-----  TIMER HAS GONE 24 HOURS

02B4 2B C0                      SUB     AX,AX
02B6 A3 006E R                  MOV     @TIMER_HIGH,AX
02B9 A3 006C R                  MOV     @TIMER_LOW,AX
02BC C6 06 0070 R 01            MOV     @TIMER_OFL,1

                        ;-----  TEST FOR DISKETTE TIME OUT

02C1                    T5:
02C1 FE 0E 0040 R               DEC     @MOTOR_COUNT    ; DECREMENT DISKETTE MOTOR CONTROL
02C5 75 0B                      JNZ     T6              ; RETURN IF COUNT NOT OUT
02C7 80 26 003F R F0            AND     @MOTOR_STATUS,0F0H ; TURN OFF MOTOR RUNNING BITS
02CC B0 0C                      MOV     AL,0CH
02CE BA 03F2                    MOV     DX,03F2H        ; FDC CTL PORT
02D1 EE                         OUT     DX,AL           ; TURN OFF THE MOTOR

02D2                    T6:                             ;       TIMER TICK INTERRUPT
02D2 CD 1C                      INT     1CH             ; TRANSFER CONTROL TO A USER ROUTINE

02D4 5A                         POP     DX              ; RESTORE (DX)
02D5 B0 20                      MOV     AL,EOI          ; GET END OF INTERRUPT MASK
02D7 FA                         CLI                     ; DISABLE INTERRUPTS TILL STACK CLEARED
02D8 E6 20                      OUT     INTA00,AL       ; END OF INTERRUPT TO 8259 - 1
02DA 58                         POP     AX
02DB 1F                         POP     DS              ; RESET MACHINE STATE
02DC CF                         IRET                    ; RETURN FROM INTERRUPT

02DD                    TIMER_INT_1     ENDP

02DD                    CODE    ENDS
                                END
```

```
                        PAGE 118,121
                        TITLE ORGS ----- 04/21/86  COMPATIBILITY MODULE
                        .LIST
0000                    CODE    SEGMENT BYTE PUBLIC

                                PUBLIC  A1
                                PUBLIC  CONF_TBL
                                PUBLIC  CRT_CHAR_GEN
                                PUBLIC  D1
                                PUBLIC  D2
                                PUBLIC  D2A
                                PUBLIC  DISK_BASE
                                PUBLIC  DUMMY_RETURN
                                PUBLIC  E101
                                PUBLIC  E102
                                PUBLIC  E103
                                PUBLIC  E104
                                PUBLIC  E105
                                PUBLIC  E106
                                PUBLIC  E107
                                PUBLIC  E108
                                PUBLIC  E109
                                PUBLIC  E161
                                PUBLIC  E162
                                PUBLIC  E163
                                PUBLIC  E164
                                PUBLIC  E201
                                PUBLIC  E202
                                PUBLIC  E203
                                PUBLIC  E301
                                PUBLIC  E302
                                PUBLIC  E303
                                PUBLIC  E304
                                PUBLIC  E401
                                PUBLIC  E501
                                PUBLIC  E601
                                PUBLIC  E602
                                PUBLIC  F1780
                                PUBLIC  F1781
                                PUBLIC  F1782
                                PUBLIC  F1790
                                PUBLIC  F1791
                                PUBLIC  F3A
                                PUBLIC  F3D
                                PUBLIC  F3D1
                                PUBLIC  FD_TBL
                                PUBLIC  FLOPPY
                                PUBLIC  HRD
                                PUBLIC  K6
                                PUBLIC  K6L
                                PUBLIC  K7
                                PUBLIC  K8
                                PUBLIC  K10
                                PUBLIC  K11
                                PUBLIC  K12
                                PUBLIC  K14
                                PUBLIC  K15
                                PUBLIC  M4
                                PUBLIC  M5
                                PUBLIC  M6
                                PUBLIC  M7
                                PUBLIC  NMI_INT
                                PUBLIC  PRINT_SCREEN
                                PUBLIC  P_O_R
                                PUBLIC  SEEKS_1
                                PUBLIC  SLAVE_VECTOR_TABLE
                                PUBLIC  TUTOR
                                PUBLIC  VECTOR_TABLE
                                PUBLIC  VIDEO_PARMS

                                EXTRN   BOOT_STRAP_1:NEAR
                                EXTRN   CASSETTE_IO_1:NEAR
                                EXTRN   D11:NEAR
                                EXTRN   DISK_INT_1:NEAR
                                EXTRN   DISK_SETUP:NEAR
                                EXTRN   DISKETTE_IO_1:NEAR
                                EXTRN   DSKETTE_SETUP:NEAR
                                EXTRN   EQUIPMENT_1:NEAR
                                EXTRN   INT_287:NEAR
                                EXTRN   K16:NEAR
                                EXTRN   KEYBOARD_IO_1:NEAR
                                EXTRN   KB_INT_1:NEAR
                                EXTRN   MEMORY_SIZE_DET_1:NEAR
                                EXTRN   NMI_INT_1:NEAR
                                EXTRN   PRINT_SCREEN_1:NEAR
                                EXTRN   PRINTER_IO_1:NEAR
                                EXTRN   RE_DIRECT:NEAR
                                EXTRN   RS232_IO_1:NEAR
                                EXTRN   RTC_INT:NEAR
                                EXTRN   SEEK:NEAR
                                EXTRN   START_1:NEAR
                                EXTRN   TIME_OF_DAY_1:NEAR
                                EXTRN   TIMER_INT_1:NEAR
                                EXTRN   VIDEO_IO_1:NEAR

                                ASSUME  CS:CODE,DS:DATA

                        ;-------------------------------------------------------------------------------
                        ;                                                                              :
                        ;  THIS MODULE HAS BEEN ADDED TO FACILITATE THE EXPANSION OF THIS PROGRAM.     :
                        ;  IT ALLOWS FOR THE FIXED ORG STATEMENT ENTRY POINTS THAT HAVE TO REMAIN      :
                        ;  AT THE SAME ADDRESSES.   THE USE OF ENTRY POINTS AND TABLES WITHIN THIS     :
                        ;  MODULE SHOULD BE AVOIDED AND ARE INCLUDED ONLY TO SUPPORT EXISTING CODE     :
                        ;  THAT VIOLATE THE STRUCTURE AND DESIGN OF BIOS.   ALL BIOS ACCESS SHOULD     :
                        ;  USE THE DOCUMENTED INTERRUPT VECTOR INTERFACE FOR COMPATIBILITY.            :
                        ;                                                                              :
                        ;-------------------------------------------------------------------------------
```

```
                        PAGE
                        ;--------------------------------
                        ;       COPYRIGHT NOTICE        :
                        ;--------------------------------
                        ;;-     ORG     0E000H
0000                            ORG     00000H

0000 37 38 58 37 34 36          DB      '78X7462 COPR. IBM 1981, 1986    '
     32 20 43 4F 50 52
     2E 20 49 42 4D 20
     31 39 38 31 2C 20
     31 39 38 36 20 20
     20 20


                        ;---------------------------------------
                        ;       PARITY ERROR MESSAGES           :
                        ;---------------------------------------

0020 50 41 52 49 54 59  D1      DB      'PARITY CHECK 1',CR,LF  ; PLANAR BOARD PARITY CHECK LATCH SET
     20 43 48 45 43 4B
     20 31 0D 0A
0030 50 41 52 49 54 59  D2      DB      'PARITY CHECK 2',CR,LF  ; I/O CHANNEL CHECK LATCH SET
     20 43 48 45 43 4B
     20 32 0D 0A
0040 3F 3F 3F 3F 3F 0D  D2A     DB      '?????',CR,LF
     0A
= 0047                  IP      =       $
                        ;;-     ORG     0E05BH
005B                            ORG     0005BH
005B                    RESET:                          ;       RESET START
005B E9 0000 E                  JMP     START_1         ; VECTOR ON TO THE MOVED POST CODE


                        ;--------------------------------
                        ;       POST ERROR MESSAGES     :
                        ;--------------------------------

005E 20 31 30 31 2D 53  E101    DB      ' 101-System Board Error',CR,LF ; INTERRUPT FAILURE
     79 73 74 65 6D 20
     42 6F 61 72 64 20
     45 72 72 6F 72 0D
     0A
0077 20 31 30 32 2D 53  E102    DB      ' 102-System Board Error',CR,LF ; TIMER FAILURE
     79 73 74 65 6D 20
     42 6F 61 72 64 20
     45 72 72 6F 72 0D
     0A
0090 20 31 30 33 2D 53  E103    DB      ' 103-System Board Error',CR,LF ; TIMER INTERRUPT FAILURE
     79 73 74 65 6D 20
     42 6F 61 72 64 20
     45 72 72 6F 72 0D
     0A
00A9 20 31 30 34 2D 53  E104    DB      ' 104-System Board Error',CR,LF ; PROTECTED MODE FAILURE
     79 73 74 65 6D 20
     42 6F 61 72 64 20
     45 72 72 6F 72 0D
     0A
00C2 20 31 30 35 2D 53  E105    DB      ' 105-System Board Error',CR,LF ; LAST 8042 COMMAND NOT ACCEPTED
     79 73 74 65 6D 20
     42 6F 61 72 64 20
     45 72 72 6F 72 0D
     0A
00DB 20 31 30 36 2D 53  E106    DB      ' 106-System Board Error',CR,LF ; CONVERTING LOGIC TEST
     79 73 74 65 6D 20
     42 6F 61 72 64 20
     45 72 72 6F 72 0D
     0A
00F4 20 31 30 37 2D 53  E107    DB      ' 107-System Board Error',CR,LF ; HOT NMI TEST
     79 73 74 65 6D 20
     42 6F 61 72 64 20
     45 72 72 6F 72 0D
     0A
010D 20 31 30 38 2D 53  E108    DB      ' 108-System Board Error',CR,LF ; TIMER BUS TEST
     79 73 74 65 6D 20
     42 6F 61 72 64 20
     45 72 72 6F 72 0D
     0A
0126 20 31 30 39 2D 53  E109    DB      ' 109-System Board Error',CR,LF ; LOW MEG CHIP SELECT TEST
     79 73 74 65 6D 20
     42 6F 61 72 64 20
     45 72 72 6F 72 0D
     0A
013F 20 31 36 31 2D 53  E161    DB      ' 161-System Options Not Set-(Run SETUP)',CR,LF ; DEAD BATTERY
     79 73 74 65 6D 20
     4F 70 74 69 6F 6E
     73 20 4E 6F 74 20
     53 65 74 2D 28 52
     75 6E 20 53 45 54
     55 50 29 0D 0A
0168 20 31 36 32 2D 53  E162    DB      ' 162-System Options Not Set-(Run SETUP)',CR,LF ;CHECKSUM/CONFIG
     79 73 74 65 6D 20
     4F 70 74 69 6F 6E
     73 20 4E 6F 74 20
     53 65 74 2D 28 52
     75 6E 20 53 45 54
     55 50 29 0D 0A
0191 20 31 36 33 2D 54  E163    DB      ' 163-Time & Date Not Set-(Run SETUP)',CR,LF ;CLOCK NOT UPDATING
     69 6D 65 20 26 20
     44 61 74 65 20 4E
     6F 74 20 53 65 74
     2D 28 52 75 6E 20
     53 45 54 55 50 29
     0D 0A
01B7 20 31 36 34 2D 4D  E164    DB      ' 164-Memory Size Error-(Run SETUP)',CR,LF ; CMOS DOES NOT MATCH
     65 6D 6F 72 79 20
     53 69 7A 65 20 45
     72 72 6F 72 2D 28
     52 75 6E 20 53 45
     54 55 50 29 0D 0A
01DB 20 32 30 31 2D 4D  E201    DB      ' 201-Memory Error',CR,LF
     65 6D 6F 72 79 20
     45 72 72 6F 72 0D
     0A
01EE 20 32 30 32 2D 4D  E202    DB      ' 202-Memory Address Error',CR,LF               ; LINE ERROR 00->15
     65 6D 6F 72 79 20
```

```
                41 64 64 72 65 73
                73 20 45 72 72 6F
                72 0D 0A
0209 20 32 30 33 2D 4D  E203    DB      ' 203-Memory Address Error',CR,LF           ; LINE ERROR 16->23
     65 6D 6F 72 79 20
                41 64 64 72 65 73
                73 20 45 72 72 6F
                72 0D 0A
0224 20 33 30 31 2D 4B  E301    DB      ' 301-Keyboard Error',CR,LF                 ; KEYBOARD ERROR
     65 79 62 6F 61 72
                64 20 45 72 72 6F
                72 0D 0A
0239 20 33 30 32 2D 53  E302    DB      ' 302-System Unit Keylock is Locked',CR,LF  ; KEYBOARD LOCK ON
     79 73 74 65 6D 20
                55 6E 69 74 20 4B
                65 79 6C 6F 63 6B
                20 69 73 20 4C 6F
                63 6B 65 64 0D 0A
025D 20 28 52 45 53 55  F3D     DB      ' (RESUME = "F1" KEY)',CR,LF
     4D 45 20 3D 20 22
                46 31 22 20 4B 45
                59 29 0D 0A

                        ;-----  NMI ENTRY
= 02T3                  IP      =       $
                        ;;-     ORG     0E2C3H
02C3                            ORG     002C3H
= 02C3                  NMI_INT EQU     $
02C3 E9 0000 E                  JMP     NMI_INT_1               ; VECTOR ON TO MOVED NMI CODE

02C6 20 33 30 33 2D 4B  E303    DB      ' 303-Keyboard Or System Unit Error',CR,LF
     65 79 62 6F 61 72
                64 20 4F 72 20 53
                79 73 74 65 6D 20
                55 6E 69 74 20 45
                72 72 6F 72 0D 0A
                        ;-----  KEYBOARD/SYSTEM ERROR
02EA 20 33 30 34 2D 4B  E304    DB      ' 304-Keyboard Or System Unit Error',CR,LF ; KEYBOARD CLOCK HIGH
     65 79 62 6F 61 72
                64 20 4F 72 20 53
                79 73 74 65 6D 20
                55 6E 69 74 20 45
                72 72 6F 72 0D 0A
030E 20 34 30 31 2D 43  E401    DB      ' 401-CRT Error',CR,LF                      ; MONOCHROME
     52 54 20 45 72 72
                6F 72 0D 0A
031E 20 35 30 31 2D 43  E501    DB      ' 501-CRT Error',CR,LF                      ; COLOR
     52 54 20 45 72 72
                6F 72 0D 0A
032E 20 36 30 31 2D 44  E601    DB      ' 601-Diskette Error',CR,LF                 ; DISKETTE ERROR
     69 73 6B 65 74 74
                65 20 45 72 72 6F
                72 0D 0A
                        ;-----  DISKETTE BOOT RECORD IS NOT VALID
0343 20 36 30 32 2D 44  E602    DB      ' 602-Diskette Boot Record Error',CR,LF
     69 73 6B 65 74 74
                65 20 42 6F 6F 74
                20 52 65 63 6F 72
                64 20 45 72 72 6F
                72 0D 0A
                        ;-----  HARD FILE ERROR MESSAGE
0364 31 37 38 30 2D 44  F1780   DB      '1780-Disk 0 Failure',CR,LF
     69 73 6B 20 30 20
                46 61 69 6C 75 72
                65 0D 0A
0379 31 37 38 31 2D 44  F1781   DB      '1781-Disk 1 Failure',CR,LF
     69 73 6B 20 31 20
                46 61 69 6C 75 72
                65 0D 0A
038E 31 37 38 32 2D 44  F1782   DB      '1782-Disk Controller Failure',CR,LF
     69 73 6B 20 43 6F
                6E 74 72 6F 6C 6C
                65 72 20 46 61 69
                6C 75 72 65 0D 0A
03AC 31 37 39 30 2D 44  F1790   DB      '1790-Disk 0 Error',CR,LF
     69 73 6B 20 30 20
                45 72 72 6F 72 0D
                0A
03BF 31 37 39 31 2D 44  F1791   DB      '1791-Disk 1 Error',CR,LF
     69 73 6B 20 31 20
                45 72 72 6F 72 0D
                0A

03D2 52 4F 4D 20 20 45  F3A     DB      'ROM  Error ',CR,LF                         ; ROM CHECKSUM
     72 72 6F 72 20 0D
                0A
03DF 20 20 20 20 2D 55  F3D1    DB      '    -Unlock System Unit Keylock ',CR,LF
     6E 6C 6F 63 6B 20
                53 79 73 74 65 6D
                20 55 6E 69 74 20
                4B 65 79 6C 6F 63
                6B 20 0D 0A
```

```
                        PAGE
                        ;--------------------------------------------------------
                        ; INITIALIZE DRIVE CHARACTERISTICS                      :
                        ;                                                      :
                        ; FIXED DISK PARAMETER TABLE                           :
                        ;                                                      :
                        ;  -  THE TABLE IS COMPOSED OF A BLOCK DEFINED AS:    :
                        ;                                                      :
                        ;  +0    (1 WORD) - MAXIMUM NUMBER OF CYLINDERS        :
                        ;  +2    (1 BYTE) - MAXIMUM NUMBER OF HEADS            :
                        ;  +3    (1 WORD) - NOT USED/SEE PC-XT                 :
                        ;  +5    (1 WORD) - STARTING WRITE PRECOMPENSATION CYL  :
                        ;  +7    (1 BYTE) - NOT USED/SEE PC-XT                 :
                        ;  +8    (1 BYTE) - CONTROL BYTE                       :
                        ;                    BIT    7 DISABLE RETRIES -OR-    :
                        ;                    BIT    6 DISABLE RETRIES         :
                        ;                    BIT    3 MORE THAN 8 HEADS       :
                        ;  +9    (3 BYTES)- NOT USED/SEE PC-XT                 :
                        ; +12    (1 WORD) - LANDING ZONE                       :
                        ; +14    (1 BYTE) - NUMBER OF SECTORS/TRACK            :
                        ; +15    (1 BYTE) - RESERVED FOR FUTURE USE            :
                        ;                                                      :
                        ;         - TO DYNAMICALLY DEFINE A SET OF PARAMETERS  :
                        ;           BUILD A TABLE FOR UP TO 15 TYPES AND PLACE :
                        ;           THE CORRESPONDING VECTOR INTO INTERRUPT 41  :
                        ;           FOR DRIVE 0 AND INTERRUPT 46 FOR DRIVE 1.  :
                        ;                                                      :
                        ;--------------------------------------------------------

0401                    FD_TBL:

                        ;-----  DRIVE TYPE 01

0401 0132                       DW      0306D                   ; CYLINDERS
0403 04                         DB      04D                     ; HEADS
0404 0000                       DW      0
0406 0080                       DW      0128D                   ; WRITE PRE-COMPENSATION CYLINDER
0408 00                         DB      0
0409 00                         DB      0                       ; CONTROL BYTE
040A 00 00 00                   DB      0,0,0
040D 0131                       DW      0305D                   ; LANDING ZONE
040F 11                         DB      17D                     ; SECTORS/TRACK
0410 00                         DB      0

                        ;-----  DRIVE TYPE 02

0411 0267                       DW      0615D                   ; CYLINDERS
0413 04                         DB      04D                     ; HEADS
0414 0000                       DW      0
0416 012C                       DW      0300D                   ; WRITE PRE-COMPENSATION CYLINDER
0418 00                         DB      0
0419 00                         DB      0                       ; CONTROL BYTE
041A 00 00 00                   DB      0,0,0
041D 0267                       DW      0615D                   ; LANDING ZONE
041F 11                         DB      17D                     ; SECTORS/TRACK
0420 00                         DB      0

                        ;-----  DRIVE TYPE 03

0421 0267                       DW      0615D                   ; CYLINDERS
0423 06                         DB      06D                     ; HEADS
0424 0000                       DW      0
0426 012C                       DW      0300D                   ; WRITE PRE-COMPENSATION CYLINDER
0428 00                         DB      0
0429 00                         DB      0                       ; CONTROL BYTE
042A 00 00 00                   DB      0,0,0
042D 0267                       DW      0615D                   ; LANDING ZONE
042F 11                         DB      17D                     ; SECTORS/TRACK
0430 00                         DB      0

                        ;-----  DRIVE TYPE 04

0431 03AC                       DW      0940D                   ; CYLINDERS
0433 08                         DB      08D                     ; HEADS
0434 0000                       DW      0
0436 0200                       DW      0512D                   ; WRITE PRE-COMPENSATION CYLINDER
0438 00                         DB      0
0439 00                         DB      0                       ; CONTROL BYTE
043A 00 00 00                   DB      0,0,0
043D 03AC                       DW      0940D                   ; LANDING ZONE
043F 11                         DB      17D                     ; SECTORS/TRACK
0440 00                         DB      0

                        ;-----  DRIVE TYPE 05

0441 03AC                       DW      0940D                   ; CYLINDERS
0443 06                         DB      06D                     ; HEADS
0444 0000                       DW      0
0446 0200                       DW      0512D                   ; WRITE PRE-COMPENSATION CYLINDER
0448 00                         DB      0
0449 00                         DB      0                       ; CONTROL BYTE
044A 00 00 00                   DB      0,0,0
044D 03AC                       DW      0940D                   ; LANDING ZONE
044F 11                         DB      17D                     ; SECTORS/TRACK
0450 00                         DB      0

                        ;-----  DRIVE TYPE 06

0451 0267                       DW      0615D                   ; CYLINDERS
0453 04                         DB      04D                     ; HEADS
0454 0000                       DW      0
0456 FFFF                       DW      0FFFFH                  ; NO WRITE PRE-COMPENSATION
0458 00                         DB      0
0459 00                         DB      0                       ; CONTROL BYTE
045A 00 00 00                   DB      0,0,0
045D 0267                       DW      0615D                   ; LANDING ZONE
045F 11                         DB      17D                     ; SECTORS/TRACK
0460 00                         DB      0
```

```
                        PAGE
                        ;-----  DRIVE TYPE 07

0461 01CE                       DW      0462D           ; CYLINDERS
0463 08                         DB      08D             ; HEADS
0464 0000                       DW      0
0466 0100                       DW      0256D           ; WRITE PRE-COMPENSATION CYLINDER
0468 00                         DB      0
0469 00                         DB      0               ; CONTROL BYTE
046A 00 00 00                   DB      0,0,0
046D 01FF                       DW      0511D           ; LANDING ZONE
046F 11                         DB      17D             ; SECTORS/TRACK
0470 00                         DB      0

                        ;-----  DRIVE TYPE 08

0471 02DD                       DW      0733D           ; CYLINDERS
0473 05                         DB      05D             ; HEADS
0474 0000                       DW      0
0476 FFFF                       DW      0FFFFH          ; NO WRITE PRE-COMPENSATION
0478 00                         DB      0
0479 00                         DB      0               ; CONTROL BYTE
047A 00 00 00                   DB      0,0,0
047D 02DD                       DW      0733D           ; LANDING ZONE
047F 11                         DB      17D             ; SECTORS/TRACK
0480 00                         DB      0

                        ;-----  DRIVE TYPE 09

0481 0384                       DW      0900D           ; CYLINDERS
0483 0F                         DB      15D             ; HEADS
0484 0000                       DW      0
0486 FFFF                       DW      0FFFFH          ; NO WRITE PRE-COMPENSATION
0488 00                         DB      0
0489 08                         DB      008H            ; CONTROL BYTE
048A 00 00 00                   DB      0,0,0
048D 0385                       DW      0901D           ; LANDING ZONE
048F 11                         DB      17D             ; SECTORS/TRACK
0490 00                         DB      0

                        ;-----  DRIVE TYPE 10

0491 0334                       DW      0820D           ; CYLINDERS
0493 03                         DB      03D             ; HEADS
0494 0000                       DW      0
0496 FFFF                       DW      0FFFFH          ; NO WRITE PRE-COMPENSATION
0498 00                         DB      0
0499 00                         DB      0               ; CONTROL BYTE
049A 00 00 00                   DB      0,0,0
049D 0334                       DW      0820D           ; LANDING ZONE
049F 11                         DB      17D             ; SECTORS/TRACK
04A0 00                         DB      0

                        ;-----  DRIVE TYPE 11

04A1 0357                       DW      0855D           ; CYLINDERS
04A3 05                         DB      05D             ; HEADS
04A4 0000                       DW      0
04A6 FFFF                       DW      0FFFFH          ; NO WRITE PRE-COMPENSATION
04A8 00                         DB      0
04A9 00                         DB      0               ; CONTROL BYTE
04AA 00 00 00                   DB      0,0,0
04AD 0357                       DW      0855D           ; LANDING ZONE
04AF 11                         DB      17D             ; SECTORS/TRACK
04B0 00                         DB      0

                        ;-----  DRIVE TYPE 12

04B1 0357                       DW      0855D           ; CYLINDERS
04B3 07                         DB      07D             ; HEADS
04B4 0000                       DW      0
04B6 FFFF                       DW      0FFFFH          ; NO WRITE PRE-COMPENSATION
04B8 00                         DB      0
04B9 00                         DB      0               ; CONTROL BYTE
04BA 00 00 00                   DB      0,0,0
04BD 0357                       DW      0855D           ; LANDING ZONE
04BF 11                         DB      17D             ; SECTORS/TRACK
04C0 00                         DB      0

                        ;-----  DRIVE TYPE 13

04C1 0132                       DW      0306D           ; CYLINDERS
04C3 08                         DB      08D             ; HEADS
04C4 0000                       DW      0
04C6 0080                       DW      0128D           ; WRITE PRE-COMPENSATION CYLINDER
04C8 00                         DB      0
04C9 08                         DB      0               ; CONTROL BYTE
04CA 00 00 00                   DB      0,0,0
04CD 013F                       DW      0319D           ; LANDING ZONE
04CF 11                         DB      17D             ; SECTORS/TRACK
04D0 00                         DB      0

                        ;-----  DRIVE TYPE 14

04D1 02DD                       DW      0733D           ; CYLINDERS
04D3 07                         DB      07D             ; HEADS
04D4 0000                       DW      0
04D6 FFFF                       DW      0FFFFH          ; NO WRITE PRE-COMPENSATION
04D8 00                         DB      0
04D9 00                         DB      0               ; CONTROL BYTE
04DA 00 00 00                   DB      0,0,0
04DD 02DD                       DW      0733D           ; LANDING ZONE
04DF 11                         DB      17D             ; SECTORS/TRACK
04E0 00                         DB      0
```

```
                        PAGE
                        ;-----  DRIVE TYPE 15    RESERVED      ****  DO NOT USE****
04E1 0000                       DW      0000D                   ; CYLINDERS
04E3 00                         DB      00D                     ; HEADS
04E4 0000                       DW      0
04E6 0000                       DW      0000D                   ; WRITE PRE-COMPENSATION CYLINDER
04E8 00                         DB      0
04E9 00                         DB      0                       ; CONTROL BYTE
04EA 00 00 00                   DB      0,0,0
04ED 0000                       DW      0000D                   ; LANDING ZONE
04EF 00                         DB      00D                     ; SECTORS/TRACK
04F0 00                         DB      0

                        ;-----  DRIVE TYPE 16

04F1 0264                       DW      0612D                   ; CYLINDERS
04F3 04                         DB      04D                     ; HEADS
04F4 0000                       DW      0
04F6 0000                       DW      0000D                   ; WRITE PRE-COMPENSATION ALL CYLINDER
04F8 00                         DB      0
04F9 00                         DB      0                       ; CONTROL BYTE
04FA 00 00 00                   DB      0,0,0
04FD 0297                       DW      0663D                   ; LANDING ZONE
04FF 11                         DB      17D                     ; SECTORS/TRACK
0500 00                         DB      0

                        ;-----  DRIVE TYPE 17

0501 03D1                       DW      0977D                   ; CYLINDERS
0503 05                         DB      05D                     ; HEADS
0504 0000                       DW      0
0506 012C                       DW      0300D                   ; WRITE PRE-COMPENSATION CYL
0508 00                         DB      0
0509 00                         DB      0                       ; CONTROL BYTE
050A 00 00 00                   DB      0,0,0
050D 03D1                       DW      0977D                   ; LANDING ZONE
050F 11                         DB      17D                     ; SECTORS/TRACK
0510 00                         DB      0

                        ;-----  DRIVE TYPE 18

0511 03D1                       DW      0977D                   ; CYLINDERS
0513 07                         DB      07D                     ; HEADS
0514 0000                       DW      0
0516 FFFF                       DW      0FFFFH                  ; NO WRITE PRE-COMPENSATION
0518 00                         DB      0
0519 00                         DB      0                       ; CONTROL BYTE
051A 00 00 00                   DB      0,0,0
051D 03D1                       DW      0977D                   ; LANDING ZONE
051F 11                         DB      17D                     ; SECTORS/TRACK
0520 00                         DB      0

                        ;-----  DRIVE TYPE 19

0521 0400                       DW      1024D                   ; CYLINDERS
0523 07                         DB      07D                     ; HEADS
0524 0000                       DW      0
0526 0200                       DW      0512D                   ; WRITE PRE-COMPENSATION CYLINDER
0528 00                         DB      0
0529 00                         DB      0                       ; CONTROL BYTE
052A 00 00 00                   DB      0,0,0
052D 03FF                       DW      1023D                   ; LANDING ZONE
052F 11                         DB      17D                     ; SECTORS/TRACK
0530 00                         DB      0

                        ;-----  DRIVE TYPE 20

0531 02DD                       DW      0733D                   ; CYLINDERS
0533 05                         DB      05D                     ; HEADS
0534 0000                       DW      0
0536 012C                       DW      0300D                   ; WRITE PRE-COMPENSATION CYL
0538 00                         DB      0
0539 00                         DB      0                       ; CONTROL BYTE
053A 00 00 00                   DB      0,0,0
053D 02DC                       DW      0732D                   ; LANDING ZONE
053F 11                         DB      17D                     ; SECTORS/TRACK
0540 00                         DB      0

                        ;-----  DRIVE TYPE 21

0541 02DD                       DW      0733D                   ; CYLINDERS
0543 07                         DB      07D                     ; HEADS
0544 0000                       DW      0
0546 012C                       DW      0300D                   ; WRITE PRE-COMPENSATION CYL
0548 00                         DB      0
0549 00                         DB      0                       ; CONTROL BYTE
054A 00 00 00                   DB      0,0,0
054D 02DC                       DW      0732D                   ; LANDING ZONE
054F 11                         DB      17D                     ; SECTORS/TRACK
0550 00                         DB      0

                        ;-----  DRIVE TYPE 22

0551 02DD                       DW      0733D                   ; CYLINDERS
0553 05                         DB      05D                     ; HEADS
0554 0000                       DW      0
0556 012C                       DW      0300D                   ; WRITE PRE-COMPENSATION CYL
0558 00                         DB      0
0559 00                         DB      0                       ; CONTROL BYTE
055A 00 00 00                   DB      0,0,0
055D 02DD                       DW      0733D                   ; LANDING ZONE
055F 11                         DB      17D                     ; SECTORS/TRACK
0560 00                         DB      0
```

```
                        PAGE
                        ;-----  DRIVE TYPE 23

0561 0132                       DW      0306D                   ; CYLINDERS
0563 04                         DB      04D                     ; HEADS
0564 0000                       DW      0
0566 0000                       DW      0000D                   ; WRITE PRE-COMPENSATION ALL CYL
0568 00                         DB      0
0569 00                         DB      0                       ; CONTROL BYTE
056A 00 00 00                   DB      0,0,0
056D 0150                       DW      0336D                   ; LANDING ZONE
056F 11                         DB      17D                     ; SECTORS/TRACK
0570 00                         DB      0

                        ;-----  DRIVE TYPE 24

0571 0264                       DW      0612D                   ; CYLINDERS
0573 04                         DB      04D                     ; HEADS
0574 0000                       DW      0
0576 0131                       DW      0305D                   ; WRITE PRE-COMPENSATION CYL
0578 00                         DB      0
0579 00                         DB      0                       ; CONTROL BYTE
057A 00 00 00                   DB      0,0,0
057D 0297                       DW      0663D                   ; LANDING ZONE
057F 11                         DB      17D                     ; SECTORS/TRACK
0580 00                         DB      0

                        ;-----  DRIVE TYPE 25   *** RESERVED***

0581 0000                       DW      0000D                   ; CYLINDERS
0583 00                         DB      00D                     ; HEADS
0584 0000                       DW      0
0586 0000                       DW      0000D                   ; WRITE PRE-COMPENSATION CYL
0588 00                         DB      0
0589 00                         DB      0                       ; CONTROL BYTE
058A 00 00 00                   DB      0,0,0
058D 0000                       DW      0000D                   ; LANDING ZONE
058F 00                         DB      00D                     ; SECTORS/TRACK
0590 00                         DB      0

                        ;-----  DRIVE TYPE 26   *** RESERVED***

0591 0000                       DW      0000D                   ; CYLINDERS
0593 00                         DB      00D                     ; HEADS
0594 0000                       DW      0
0596 0000                       DW      0000D                   ; WRITE PRE-COMPENSATION CYL
0598 00                         DB      0
0599 00                         DB      0                       ; CONTROL BYTE
059A 00 00 00                   DB      0,0,0
059D 0000                       DW      0000D                   ; LANDING ZONE
059F 00                         DB      00D                     ; SECTORS/TRACK
05A0 00                         DB      0

                        ;-----  DRIVE TYPE 27   *** RESERVED***

05A1 0000                       DW      0000D                   ; CYLINDERS
05A3 00                         DB      00D                     ; HEADS
05A4 0000                       DW      0
05A6 0000                       DW      0000D                   ; WRITE PRE-COMPENSATION CYL
05A8 00                         DB      0
05A9 00                         DB      0                       ; CONTROL BYTE
05AA 00 00 00                   DB      0,0,0
05AD 0000                       DW      0000D                   ; LANDING ZONE
05AF 00                         DB      00D                     ; SECTORS/TRACK
05B0 00                         DB      0

                        ;-----  DRIVE TYPE 28   *** RESERVED***

05B1 0000                       DW      0000D                   ; CYLINDERS
05B3 00                         DB      00D                     ; HEADS
05B4 0000                       DW      0
05B6 0000                       DW      0000D                   ; WRITE PRE-COMPENSATION CYL
05B8 00                         DB      0
05B9 00                         DB      0                       ; CONTROL BYTE
05BA 00 00 00                   DB      0,0,0
05BD 0000                       DW      0000D                   ; LANDING ZONE
05BF 00                         DB      00D                     ; SECTORS/TRACK
05C0 00                         DB      0

                        ;-----  DRIVE TYPE 29   *** RESERVED***

05C1 0000                       DW      0000D                   ; CYLINDERS
05C3 00                         DB      00D                     ; HEADS
05C4 0000                       DW      0
05C6 0000                       DW      0000D                   ; WRITE PRE-COMPENSATION CYL
05C8 00                         DB      0
05C9 00                         DB      0                       ; CONTROL BYTE
05CA 00 00 00                   DB      0,0,0
05CD 0000                       DW      0000D                   ; LANDING ZONE
05CF 00                         DB      00D                     ; SECTORS/TRACK
05D0 00                         DB      0

                        ;-----  DRIVE TYPE 30   *** RESERVED***

05D1 0000                       DW      0000D                   ; CYLINDERS
05D3 00                         DB      00D                     ; HEADS
05D4 0000                       DW      0
05D6 0000                       DW      0000D                   ; WRITE PRE-COMPENSATION CYL
05D8 00                         DB      0
05D9 00                         DB      0                       ; CONTROL BYTE
05DA 00 00 00                   DB      0,0,0
05DD 0000                       DW      0000D                   ; LANDING ZONE
05DF 00                         DB      00D                     ; SECTORS/TRACK
05E0 00                         DB      0
```

```
                        PAGE
                        ;-----  DRIVE TYPE 31    *** RESERVED***

05E1 0000                       DW      0000D                   ; CYLINDERS
05E3 00                         DB      00D                     ; HEADS
05E4 0000                       DW      0
05E6 0000                       DW      0000D                   ; WRITE PRE-COMPENSATION CYL
05E8 00                         DB      0
05E9 00                         DB      0                       ; CONTROL BYTE
05EA 00 00 00                   DB      0,0,0
05ED 0000                       DW      0000D                   ; LANDING ZONE
05EF 00                         DB      00D                     ; SECTORS/TRACK
05F0 00                         DB      0

                        ;-----  DRIVE TYPE 32    *** RESERVED***

05F1 0000                       DW      0000D                   ; CYLINDERS
05F3 00                         DB      00D                     ; HEADS
05F4 0000                       DW      0
05F6 0000                       DW      0000D                   ; WRITE PRE-COMPENSATION CYL
05F8 00                         DB      0
05F9 00                         DB      0                       ; CONTROL BYTE
05FA 00 00 00                   DB      0,0,0
05FD 0000                       DW      0000D                   ; LANDING ZONE
05FF 00                         DB      00D                     ; SECTORS/TRACK
0600 00                         DB      0

                        ;-----  DRIVE TYPE 33    *** RESERVED***

0601 0000                       DW      0000D                   ; CYLINDERS
0603 00                         DB      00D                     ; HEADS
0604 0000                       DW      0
0606 0000                       DW      0000D                   ; WRITE PRE-COMPENSATION CYL
0608 00                         DB      0
0609 00                         DB      0                       ; CONTROL BYTE
060A 00 00 00                   DB      0,0,0
060D 0000                       DW      0000D                   ; LANDING ZONE
060F 00                         DB      00D                     ; SECTORS/TRACK
0610 00                         DB      0

                        ;-----  DRIVE TYPE 34    *** RESERVED***

0611 0000                       DW      0000D                   ; CYLINDERS
0613 00                         DB      00D                     ; HEADS
0614 0000                       DW      0
0616 0000                       DW      0000D                   ; WRITE PRE-COMPENSATION CYL
0618 00                         DB      0
0619 00                         DB      0                       ; CONTROL BYTE
061A 00 00 00                   DB      0,0,0
061D 0000                       DW      0000D                   ; LANDING ZONE
061F 00                         DB      00D                     ; SECTORS/TRACK
0620 00                         DB      0

                        ;-----  DRIVE TYPE 35    *** RESERVED***

0621 0000                       DW      0000D                   ; CYLINDERS
0623 00                         DB      00D                     ; HEADS
0624 0000                       DW      0
0626 0000                       DW      0000D                   ; WRITE PRE-COMPENSATION CYL
0628 00                         DB      0
0629 00                         DB      0                       ; CONTROL BYTE
062A 00 00 00                   DB      0,0,0
062D 0000                       DW      0000D                   ; LANDING ZONE
062F 00                         DB      00D                     ; SECTORS/TRACK
0630 00                         DB      0

                        ;-----  DRIVE TYPE 36    *** RESERVED***

0631 0000                       DW      0000D                   ; CYLINDERS
0633 00                         DB      00D                     ; HEADS
0634 0000                       DW      0
0636 0000                       DW      0000D                   ; WRITE PRE-COMPENSATION CYL
0638 00                         DB      0
0639 00                         DB      0                       ; CONTROL BYTE
063A 00 00 00                   DB      0,0,0
063D 0000                       DW      0000D                   ; LANDING ZONE
063F 00                         DB      00D                     ; SECTORS/TRACK
0640 00                         DB      0

                        ;-----  DRIVE TYPE 37    *** RESERVED***

0641 0000                       DW      0000D                   ; CYLINDERS
0643 00                         DB      00D                     ; HEADS
0644 0000                       DW      0
0646 0000                       DW      0000D                   ; WRITE PRE-COMPENSATION CYL
0648 00                         DB      0
0649 00                         DB      0                       ; CONTROL BYTE
064A 00 00 00                   DB      0,0,0
064D 0000                       DW      0000D                   ; LANDING ZONE
064F 00                         DB      00D                     ; SECTORS/TRACK
0650 00                         DB      0

                        ;-----  DRIVE TYPE 38    *** RESERVED***

0651 0000                       DW      0000D                   ; CYLINDERS
0653 00                         DB      00D                     ; HEADS
0654 0000                       DW      0
0656 0000                       DW      0000D                   ; WRITE PRE-COMPENSATION CYL
0658 00                         DB      0
0659 00                         DB      0                       ; CONTROL BYTE
065A 00 00 00                   DB      0,0,0
065D 0000                       DW      0000D                   ; LANDING ZONE
065F 00                         DB      00D                     ; SECTORS/TRACK
0660 00                         DB      0
```

```
                        PAGE
                        ;-----  DRIVE TYPE 39    *** RESERVED***

0661 0000                       DW      0000D                   ; CYLINDERS
0663 00                         DB      00D                     ; HEADS
0664 0000                       DW      0
0666 0000                       DW      0000D                   ; WRITE PRE-COMPENSATION CYL
0668 00                         DB      0
0669 00                         DB      0                       ; CONTROL BYTE
066A 00 00 00                   DB      0,0,0
066D 0000                       DW      0000D                   ; LANDING ZONE
066F 00                         DB      00D                     ; SECTORS/TRACK
0670 00                         DB      0

                        ;-----  DRIVE TYPE 40    *** RESERVED***

0671 0000                       DW      0000D                   ; CYLINDERS
0673 00                         DB      00D                     ; HEADS
0674 0000                       DW      0
0676 0000                       DW      0000D                   ; WRITE PRE-COMPENSATION CYL
0678 00                         DB      0
0679 00                         DB      0                       ; CONTROL BYTE
067A 00 00 00                   DB      0,0,0
067D 0000                       DW      0000D                   ; LANDING ZONE
067F 00                         DB      00D                     ; SECTORS/TRACK
0680 00                         DB      0

                        ;-----  DRIVE TYPE 41    *** RESERVED***

0681 0000                       DW      0000D                   ; CYLINDERS
0683 00                         DB      00D                     ; HEADS
0684 0000                       DW      0
0686 0000                       DW      0000D                   ; WRITE PRE-COMPENSATION CYL
0688 00                         DB      0
0689 00                         DB      0                       ; CONTROL BYTE
068A 00 00 00                   DB      0,0,0
068D 0000                       DW      0000D                   ; LANDING ZONE
068F 00                         DB      00D                     ; SECTORS/TRACK
0690 00                         DB      0

                        ;-----  DRIVE TYPE 42    *** RESERVED***

0691 0000                       DW      0000D                   ; CYLINDERS
0693 00                         DB      00D                     ; HEADS
0694 0000                       DW      0
0696 0000                       DW      0000D                   ; WRITE PRE-COMPENSATION CYL
0698 00                         DB      0
0699 00                         DB      0                       ; CONTROL BYTE
069A 00 00 00                   DB      0,0,0
069D 0000                       DW      0000D                   ; LANDING ZONE
069F 00                         DB      00D                     ; SECTORS/TRACK
06A0 00                         DB      0

                        ;-----  DRIVE TYPE 43    *** RESERVED***

06A1 0000                       DW      0000D                   ; CYLINDERS
06A3 00                         DB      00D                     ; HEADS
06A4 0000                       DW      0
06A6 0000                       DW      0000D                   ; WRITE PRE-COMPENSATION CYL
06A8 00                         DB      0
06A9 00                         DB      0                       ; CONTROL BYTE
06AA 00 00 00                   DB      0,0,0
06AD 0000                       DW      0000D                   ; LANDING ZONE
06AF 00                         DB      00D                     ; SECTORS/TRACK
06B0 00                         DB      0

                        ;-----  DRIVE TYPE 44    *** RESERVED***

06B1 0000                       DW      0000D                   ; CYLINDERS
06B3 00                         DB      00D                     ; HEADS
06B4 0000                       DW      0
06B6 0000                       DW      0000D                   ; WRITE PRE-COMPENSATION CYL
06B8 00                         DB      0
06B9 00                         DB      0                       ; CONTROL BYTE
06BA 00 00 00                   DB      0,0,0
06BD 0000                       DW      0000D                   ; LANDING ZONE
06BF 00                         DB      00D                     ; SECTORS/TRACK
06C0 00                         DB      0

                        ;-----  DRIVE TYPE 45    *** RESERVED***

06C1 0000                       DW      0000D                   ; CYLINDERS
06C3 00                         DB      00D                     ; HEADS
06C4 0000                       DW      0
06C6 0000                       DW      0000D                   ; WRITE PRE-COMPENSATION CYL
06C8 00                         DB      0
06C9 00                         DB      0                       ; CONTROL BYTE
06CA 00 00 00                   DB      0,0,0
06CD 0000                       DW      0000D                   ; LANDING ZONE
06CF 00                         DB      00D                     ; SECTORS/TRACK
06D0 00                         DB      0

                        ;-----  DRIVE TYPE 46    *** RESERVED***

06D1 0000                       DW      0000D                   ; CYLINDERS
06D3 00                         DB      00D                     ; HEADS
06D4 0000                       DW      0
06D6 0000                       DW      0000D                   ; WRITE PRE-COMPENSATION CYL
06D8 00                         DB      0
06D9 00                         DB      0                       ; CONTROL BYTE
06DA 00 00 00                   DB      0,0,0
06DD 0000                       DW      0000D                   ; LANDING ZONE
06DF 00                         DB      00D                     ; SECTORS/TRACK
06E0 00                         DB      0
```

```
                        PAGE
                        ;-----  DRIVE TYPE 47    *** RESERVED***

                                DW      0000D                   ; CYLINDERS
                                DB      00D                     ; HEADS
                                DW      0
                                DW      0000D                   ; WRITE PRE-COMPENSATION CYL
                                DB      0
                                DB      0                       ; CONTROL BYTE
                                DB      0,0,0
                                DW      0000D                   ; LANDING ZONE
                                DB      00D                     ; SECTORS/TRACK
                                DB      0


                        ;-----  BOOT LOADER INTERRUPT

                        IP      =       $
                        ;;-     ORG     0E4F2H
                                ORG     004F2H
                        BOOT_STRAP      EQU     $
                                JMP     BOOT_STRAP_1            ; VECTOR ON TO MOVED BOOT CODE

                                                                ;       USE INT 15 H  AH= 0C0H
                        CONF_TBL;                               ; CONFIGURATION TABLE FOR THIS SYSTEM
                                DW      CONF_E-CONF_TBL-2       ; LENGTH OF FOLLOWING TABLE
                                DB      MODEL_BYTE              ; SYSTEM MODEL BYTE
                                DB      SUB_MODEL_BYTE          ; SYSTEM SUB MODEL TYPE BYTE
                                DB      BIOS_LEVEL              ; BIOS REVISION LEVEL
                                DB      01110000B               ; 10000000 = DMA CHANNEL 3 USE BY BIOS
                                                                ; 01000000 = CASCADED INTERRUPT LEVEL 2
                                                                ; 00100000 = REAL TIME CLOCK AVAILABLE
                                                                ; 00010000 = KEYBOARD SCAN CODE HOOK 1AH
                                DB      0                       ; RESERVED
                                DB      0                       ; RESERVED
                                DB      0                       ; RESERVED
                                DB      0                       ; RESERVED
                        CONF_E  EQU     $                       ; RESERVED FOR EXPANSION

                        ;-----  BAUD RATE INITIALIZATION TABLE

                        IP      =       $
                        ;;-     ORG     0E729H
                                ORG     00729H
                        A1      DW      1047                    ; 110 BAUD      ; TABLE OF VALUES
                                DW      768                     ; 150           ; FOR INITIALIZATION
                                DW      384                     ; 300
                                DW      192                     ; 600
                                DW      96                      ; 1200
                                DW      48                      ; 2400
                                DW      24                      ; 4800
                                DW      12                      ; 9600

                        ;-----  RS232

                        ;;-     ORG     0E739H
                                ORG     00739H
                        RS232_IO        EQU     $
                                JMP     RS232_IO_1              ; VECTOR ON TO MOVED RS232 CODE

                        ;-----  KEYBOARD

                        ;;-     ORG     0E82EH
                                ORG     0082EH
                        KEYBOARD_IO     EQU     $
                                JMP     KEYBOARD_IO_1           ; VECTOR ON TO MOVED KEYBOARD CODE

                        ;--------------------------------------------------------------------------
                        ;       KEY IDENTIFICATION SCAN TABLES                                     -
                        ;--------------------------------------------------------------------------
                        ;;-     ORG     0E87EH
                                ORG     0087EH
                        ;-------------- TABLE OF SHIFT KEYS AND MASK VALUES ----------------
                        ;------ KEY_TABLE
                        K6      LABEL   BYTE
                                DB      INS_KEY                 ; INSERT KEY
                                DB      CAPS_KEY,NUM_KEY,SCROLL_KEY,ALT_KEY,CTL_KEY
                                DB      LEFT_KEY,RIGHT_KEY
                        K6L     EQU     $-K6

                        ;------ MASK_TABLE
                        K7      LABEL   BYTE
                                DB      INS_SHIFT               ; INSERT MODE SHIFT
                                DB      CAPS_SHIFT,NUM_SHIFT,SCROLL_SHIFT,ALT_SHIFT,CTL_SHIFT
                                DB      LEFT_SHIFT,RIGHT_SHIFT

                        ;----------------------- TABLES FOR CTRL CASE ------------------------

                        K8      LABEL   BYTE                    ;-------- CHARACTERS ---------
                                DB      27,-1,00,-1,-1,-1       ; Esc, 1, 2, 3, 4, 5
                                DB      30,-1,-1,-1,-1,31       ; 6, 7, 8, 9, 0, -
                                DB      -1,127,148,17,23,5      ; =, Bksp, Tab, Q, W, E
                                DB      18,20,25,21,09,15       ; R, T, Y, U, I, O
                                DB      16,27,29,10,-1,01       ; P, [,], Enter, Ctrl, A
                                DB      19,04,06,07,08,10       ; S, D, F, G, H, J
                                DB      11,12,-1,-1,-1,-1       ; K, L, ;, ', `, LShift
                                DB      28,26,24,03,22,02       ; Bkslash, Z, X, C, V, B
                                DB      14,13,-1,-1,-1,-1       ; N, M, ,, ., /, RShift
                                DB      150,-1,' ',-1           ; *, Alt, Space, CL
                                                                ;-------- FUNCTIONS ---------
                                DB      94,95,96,97,98,99       ; F1 - F6
                                DB      100,101,102,103,-1,-1   ; F7 - F10, NL, SL
                                DB      119,141,132,142,115,143 ; Home, Up, PgUp, -, Left, Pad5
                                DB      116,144,117,145,118,146 ; Right, +, End, Down, PgDn, Ins
                                DB      147,-1,-1,-1,137,138    ; Del, SysReq, Undef, WT, F11, F12
```

```
                                PAGE
                                ;----------------------  TABLES FOR LOWER CASE  -----------------------

1048                            K10     LABEL   BYTE
1049                                    DB      27,'!2345'
1050                                    DB      '67890-='
1051 08E6                               DB      '=',08,09,'qwe'
1052 08E6 1B 31 32 33 34 35             DB      'rtyuio'
1053 08EC 36 37 38 39 30 2D             DB      'p[]',0DH,-1,'a'        ; LETTERS, Return, Ctrl
1054 08F2 3D 08 09 71 77 65             DB      'sdfghj'
1055 08F8 72 74 79 75 69 6F             DB      'kl;',27H,-1            ; LETTERS, L Shift
1056 08FE 70 5B 5D 0D FF 61             DB      92,'zxcvb'
1057 0904 73 64 66 67 68 6A             DB      'nm,./'
1058 090A 6B 6C 3B 27 60 FF             DB      -1,'*',-1,' ',-1        ; R Shift,*, Alt, Space, CL
1059 0910 5C 7A 78 63 76 62
1060 0916 6E 6D 2C 2E 2F
1061 091B FF 2A FF 20 FF
1062
1063                            ;------ LC TABLE SCAN
1064 0920 3B 3C 3D 3E 3F                DB      59,60,61,62,63          ; BASE STATE OF F1 - F10
1065 0925 40 41 42 43 44                DB      64,65,66,67,68
1066 092A FF FF                         DB      -1,-1                   ; NL, SL
1067
1068                            ;------ KEYPAD TABLE
1069 092C                       K15     LABEL   BYTE
1070 092C 47 48 49 FF 4B FF             DB      71,72,73,-1,75,-1       ; BASE STATE OF KEYPAD KEYS
1071 0932 4D FF 4F 50 51 52             DB      77,-1,79,80,81,82
1072 0938 53                            DB      83
1073 0939 FF FF 5C 85 86                DB      -1,-1,92,133,134        ; SysRq, Undef, WT, F11, F12
1074
1075                            ;-----  KEYBOARD INTERRUPT
1076
1077                            !!-     ORG     0E987H
1078 0987                               ORG     0098TH
1079 = 0987                     KB_INT  EQU     $
1080 0987 E9 0000 E                     JMP     KB_INT_1                ; VECTOR ON TO MOVED KEYBOARD HANDLER
1081
1082                            ;----------------------  TABLES FOR UPPER CASE  -----------------------
1083
1084 098A                       K11     LABEL   BYTE
1085 098A 1B 21 40 23 24 25             DB      27,'!@#$%'
1086 0990 5E 26 2A 28 29 5F             DB      94,'&*()_'
1087 0996 2B 08 00 51 57 45             DB      '+',08,00,'QWE'
1088 099C 52 54 59 55 49 4F             DB      'RTYUIO'
1089 09A2 50 7B 7D 0D FF 41             DB      'P{}',0DH,-1,'A'        ; LETTERS, Return, Ctrl
1090 09A8 53 44 46 47 48 4A             DB      'SDFGHJ'
1091 09AE 4B 4C 3A 22 7E FF             DB      'KL:"',126,-1           ; LETTERS, L Shift
1092 09B4 7C 5A 58 43 56 42             DB      124,'ZXCVB'
1093 09BA 4E 4D 3C 3E 3F                DB      'NM<>?'
1094 09BF FF 2A FF 20 FF                DB      -1,'*',-1,' ',-1        ; R Shift,*, Alt, Space, CL
1095
1096                            ;------ UC TABLE SCAN
1097 09C4                       K12     LABEL   BYTE
1098 09C4 54 55 56 57 58                DB      84,85,86,87,88          ; SHIFTED STATE OF F1 - F10
1099 09C9 59 5A 5B 5C 5D                DB      89,90,91,92,93
1100 09CE FF FF                         DB      -1,-1                   ; NL, SL
1101
1102                            ;------ NUM STATE TABLE
1103 09D0                       K14     LABEL   BYTE
1104 09D0 37 38 39 2D 34 35             DB      '789-456+1230.'         ; NUMLOCK STATE OF KEYPAD KEYS
1105          36 2B 31 32 33 30
1106          2E
1107 09DD FF FF 7C 87 88                DB      -1,-1,124,135,136       ; SysRq, Undef, WT, F11, F12
```

```
                        PAGE
                        ;-----  DISKETTE I/O

                        ;;-     ORG     0EC59H
                                ORG     00C59H
0C59                    DISKETTE_IO     EQU     $
0C59 E9 0000 E                  JMP     DISKETTE_IO_1           ; VECTOR ON TO MOVED DISKETTE CODE

                        ;-----  DISKETTE INTERRUPT

                        ;;-     ORG     0EF57H
                                ORG     00F57H
0F57                    DISK_INT        EQU     $
0F57 E9 0000 E                  JMP     DISK_INT_1              ; VECTOR ON TO MOVED DISKETTE HANDLER

                        ;-----  DISKETTE PARAMETERS

                        ;;-     ORG     0EFC7H
                                ORG     00FC7H

                        ;--------------------------------------------------------------
                        ; DISK_BASE                                                   :
                        ;       THIS IS THE SET OF PARAMETERS REQUIRED FOR           :
                        ;       DISKETTE OPERATION.  THEY ARE POINTED AT BY THE      :
                        ;       DATA VARIABLE @DISK_POINTER.  TO MODIFY THE PARAMETERS, :
                        ;       BUILD ANOTHER PARAMETER BLOCK AND POINT AT IT        :
                        ;--------------------------------------------------------------

0FC7                    DISK_BASE       LABEL   BYTE

0FC7 DF                         DB      11011111B       ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
0FC8 02                         DB      2               ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
0FC9 25                         DB      MOTOR_WAIT      ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
0FCA 02                         DB      2               ; 512 BYTES/SECTOR
0FCB 0F                         DB      15              ; EOT ( LAST SECTOR ON TRACK)
0FCC 1B                         DB      01BH            ; GAP LENGTH
0FCD FF                         DB      0FFH            ; DTL
0FCE 54                         DB      054H            ; GAP LENGTH FOR FORMAT
0FCF F6                         DB      0F6H            ; FILL BYTE FOR FORMAT
0FD0 0F                         DB      15              ; HEAD SETTLE TIME (MILLISECONDS)
0FD1 08                         DB      8               ; MOTOR START TIME (1/8 SECONDS)

                        ;-----  PRINTER I/O

                        ;;-     ORG     0EFD2H
                                ORG     00FD2H
0FD2                    PRINTER_IO      EQU     $
0FD2 E9 0000 E                  JMP     PRINTER_IO_1            ; VECTOR ON TO MOVED PRINTER CODE

                        ;-----  FOR POSSIBLE COMPATIBILITY ENTRY POINTS

                        ;;-     ORG     0F045H
1045                            ORG     01045H
                                ASSUME  CS:CODE,DS:DATA

                                EXTRN   SET_MODE:NEAR
                                EXTRN   SET_CTYPE:NEAR
                                EXTRN   SET_CPOS:NEAR
                                EXTRN   READ_CURSOR:NEAR
                                EXTRN   READ_LPEN:NEAR
                                EXTRN   ACT_DISP_PAGE:NEAR
                                EXTRN   SCROLL_UP:NEAR
                                EXTRN   SCROLL_DOWN:NEAR
                                EXTRN   READ_AC_CURRENT:NEAR
                                EXTRN   WRITE_AC_CURRENT:NEAR
                                EXTRN   WRITE_C_CURRENT:NEAR
                                EXTRN   SET_COLOR:NEAR
                                EXTRN   WRITE_DOT:NEAR
                                EXTRN   READ_DOT:NEAR
                                EXTRN   WRITE_TTY:NEAR
                                EXTRN   VIDEO_STATE:NEAR

1045 0000 E             M1      DW      OFFSET  SET_MODE        ;  TABLE OF ROUTINES WITHIN VIDEO I/O
1047 0000 E                     DW      OFFSET  SET_CTYPE       ;       EXIT STACK VALUES MAY BE
1049 0000 E                     DW      OFFSET  SET_CPOS        ;       DIFFERENT DEPENDING ON THE
104B 0000 E                     DW      OFFSET  READ_CURSOR     ;       SYSTEM AND MODEL
104D 0000 E                     DW      OFFSET  READ_LPEN
104F 0000 E                     DW      OFFSET  ACT_DISP_PAGE
1051 0000 E                     DW      OFFSET  SCROLL_UP
1053 0000 E                     DW      OFFSET  SCROLL_DOWN
1055 0000 E                     DW      OFFSET  READ_AC_CURRENT
1057 0000 E                     DW      OFFSET  WRITE_AC_CURRENT
1059 0000 E                     DW      OFFSET  WRITE_C_CURRENT
105B 0000 E                     DW      OFFSET  SET_COLOR
105D 0000 E                     DW      OFFSET  WRITE_DOT
105F 0000 E                     DW      OFFSET  READ_DOT
1061 0000 E                     DW      OFFSET  WRITE_TTY
1063 0000 E                     DW      OFFSET  VIDEO_STATE
0020                    M1L     EQU     $-M1

                        ;;-     ORG     0F065H
1065                            ORG     01065H
1065                    VIDEO_IO        EQU     $
1065 E9 0000 E                  JMP     VIDEO_IO_1              ; VECTOR ON TO MOVED VIDEO CODE

                        ;-----  VIDEO PARAMETERS --- INIT_TABLE

                        ;;-     ORG     0F0A4H
10A4                            ORG     010A4H

10A4                    VIDEO_PARMS     LABEL   BYTE
10A4 38 28 2D 0A 1F 06          DB      38H,28H,2DH,0AH,1FH,6,19H       ; SET UP FOR 40X25
     19
10AB 1C 02 07 06 07             DB      1CH,2,7,6,7
10B0 00 00 00 00                DB      0,0,0,0
0010                    M4      EQU     $-VIDEO_PARMS

10B4 71 50 5A 0A 1F 06          DB      71H,50H,5AH,0AH,1FH,6,19H       ; SET UP FOR 80X25
     19
10BB 1C 02 07 06 07             DB      1CH,2,7,6,7
10C0 00 00 00 00                DB      0,0,0,0

10C4 38 28 2D 0A 7F 06          DB      38H,28H,2DH,0AH,7FH,6,64H       ; SET UP FOR GRAPHICS
     64
```

```
10CB 70 02 01 06 07                DB      70H,2,1,6,7
10D0 00 00 00 00 00                DB      0,0,0,0,0

10D4 61 50 52 0F 19 06             DB      61H,50H,52H,0FH,19H,6,19H       ; SET UP FOR 80X25 B&W CARD
     19
10DB 19 02 0D 0B 0C                DB      19H,2,0DH,0BH,0CH
10E0 00 00 00 00                   DB      0,0,0,0
                                                                ; TABLE OF REGEN LENGTHS
10E4 0800                  M5      DW      2048                 ; 40X25
10E6 1000                          DW      4096                 ; 80X25
10E8 4000                          DW      16384                ; GRAPHICS
10EA 4000                          DW      16384

                           ;-----  COLUMNS

10EC 28 28 50 50 28 28     M6      DB      40,40,80,80,40,40,80,80
     50 50

                           ;-----  C_REG_TAB

10F4 2C 28 2D 29 2A 2E     M7      DB      2CH,28H,2DH,29H,2AH,2EH,1EH,29H ; TABLE OF MODE SETS
     1E 29

                           ;-----  MEMORY SIZE

                           ;-      ORG     0F841H
                                   ORG     01841H
1841                       MEMORY_SIZE_DET EQU  $
1841 E9 0000 E                     JMP     MEMORY_SIZE_DET_1       ; VECTOR ON TO MOVED BIOS CODE

                           ;-----  EQUIPMENT DETERMINE

                           ;-      ORG     0F84DH
                                   ORG     0184DH
184D                       EQUIPMENT       EQU     $
184D E9 0000 E                     JMP     EQUIPMENT_1             ; VECTOR ON TO MOVED BIOS CODE

                           ;-----  CASSETTE (NO BIOS SUPPORT)

                           ;-      ORG     0F859H
                                   ORG     01859H
1859                       CASSETTE_IO     EQU     $
1859 E9 0000 E                     JMP     CASSETTE_IO_1           ; VECTOR ON TO MOVED BIOS CODE

                           ;--------------------------------------------------------
                           ; CHARACTER GENERATOR GRAPHICS FOR 320X200 AND 640X200 GRAPHICS      :
                           ;--------------------------------------------------------
                           ;-      ORG     0FA6EH
                                   ORG     01A6EH
1A6E                       CRT_CHAR_GEN    LABEL   BYTE
1A6E 00 00 00 00 00 00             DB      000H,000H,000H,000H,000H,000H,000H,000H ; D_00   BLANK
     00 00
1A76 7E 81 A5 81 BD 99             DB      07EH,081H,0A5H,081H,0BDH,099H,081H,07EH ; D_01   SMILING FACE
     81 7E
1A7E 7E FF DB FF C3 E7             DB      07EH,0FFH,0DBH,0FFH,0C3H,0E7H,0FFH,07EH ; D_02   SMILING FACE N
     FF 7E
1A86 6C FE FE FE 7C 38             DB      06CH,0FEH,0FEH,0FEH,07CH,038H,010H,000H ; D_03   HEART
     10 00
1A8E 10 38 7C FE 7C 38             DB      010H,038H,07CH,0FEH,07CH,038H,010H,000H ; D_04   DIAMOND
     10 00
1A96 38 7C 38 FE FE 7C             DB      038H,07CH,038H,0FEH,0FEH,07CH,038H,07CH ; D_05   CLUB
     38 7C
1A9E 10 10 38 7C FE 7C             DB      010H,010H,038H,07CH,0FEH,07CH,038H,07CH ; D_06   SPADE
     38 7C
1AA6 00 00 18 3C 3C 18             DB      000H,000H,018H,03CH,03CH,018H,000H,000H ; D_07   BULLET
     00 00
1AAE FF FF E7 C3 C3 E7             DB      0FFH,0FFH,0E7H,0C3H,0C3H,0E7H,0FFH,0FFH ; D_08   BULLET NEG
     FF FF
1AB6 00 3C 66 42 42 66             DB      000H,03CH,066H,042H,042H,066H,03CH,000H ; D_09   CIRCLE
     3C 00
1ABE FF C3 99 BD BD 99             DB      0FFH,0C3H,099H,0BDH,0BDH,099H,0C3H,0FFH ; D_0A   CIRCLE NEG
     C3 FF
1AC6 0F 07 0F 7D CC CC             DB      00FH,007H,00FH,07DH,0CCH,0CCH,0CCH,078H ; D_0B   MALE
     CC 78
1ACE 3C 66 66 66 3C 18             DB      03CH,066H,066H,066H,03CH,018H,07EH,018H ; D_0C   FEMALE
     7E 18
1AD6 3F 33 3F 30 30 70             DB      03FH,033H,03FH,030H,030H,070H,0F0H,0E0H ; D_0D   EIGHTH NOTE
     F0 E0
1ADE 7F 63 7F 63 63 67             DB      07FH,063H,07FH,063H,063H,067H,0E6H,0C0H ; D_0E   TWO 1/16 NOTE
     E6 C0
1AE6 99 5A 3C E7 E7 3C             DB      099H,05AH,03CH,0E7H,0E7H,03CH,05AH,099H ; D_0F   SUN
     5A 99

1AEE 80 E0 F8 FE F8 E0             DB      080H,0E0H,0F8H,0FEH,0F8H,0E0H,080H,000H ; D_10   R ARROWHEAD
     80 00
1AF6 02 0E 3E FE 3E 0E             DB      002H,00EH,03EH,0FEH,03EH,00EH,002H,000H ; D_11   L ARROWHEAD
     02 00
1AFE 18 3C 7E 18 18 7E             DB      018H,03CH,07EH,018H,018H,07EH,03CH,018H ; D_12   ARROW 2 VERT
     3C 18
1B06 66 66 66 66 66 00             DB      066H,066H,066H,066H,066H,000H,066H,000H ; D_13   2 EXCLAMATIONS
     66 00
1B0E 7F DB DB 7B 1B 1B             DB      07FH,0DBH,0DBH,07BH,01BH,01BH,01BH,000H ; D_14   PARAGRAPH
     1B 00
1B16 3E 63 38 6C 6C 38             DB      03EH,063H,038H,06CH,06CH,038H,0CCH,078H ; D_15   SECTION
     CC 78
1B1E 00 00 00 00 7E 7E             DB      000H,000H,000H,000H,07EH,07EH,07EH,000H ; D_16   RECTANGLE
     7E 00
1B26 18 3C 7E 18 7E 3C             DB      018H,03CH,07EH,018H,07EH,03CH,018H,0FFH ; D_17   ARROW 2 VRT UP
     18 FF
1B2E 18 3C 7E 18 18 18             DB      018H,03CH,07EH,018H,018H,018H,018H,000H ; D_18   ARROW VRT UP
     18 00
1B36 18 18 18 18 7E 3C             DB      018H,018H,018H,018H,07EH,03CH,018H,000H ; D_19   ARROW VRT DOWN
     18 00
1B3E 00 18 0C FE 0C 18             DB      000H,018H,00CH,0FEH,00CH,018H,000H,000H ; D_1A   ARROW RIGHT
     00 00
1B46 00 30 60 FE 60 30             DB      000H,030H,060H,0FEH,060H,030H,000H,000H ; D_1B   ARROW LEFT
     00 00
1B4E 00 00 C0 C0 C0 FE             DB      000H,000H,0C0H,0C0H,0C0H,0FEH,000H,000H ; D_1C   NOT INVERTED
     00 00
1B56 00 24 66 FF 66 24             DB      000H,024H,066H,0FFH,066H,024H,000H,000H ; D_1D   ARROW 2 HORZ
     00 00
1B5E 00 18 3C 7E FF FF             DB      000H,018H,03CH,07EH,0FFH,0FFH,000H,000H ; D_1E   ARROWHEAD UP
     00 00
1B66 00 FF FF 7E 3C 18             DB      000H,0FFH,0FFH,07EH,03CH,018H,000H,000H ; D_1F   ARROWHEAD DOWN
     00 00
```

```
1336 1B6E 00 00 00 00 00 00          DB      000H,000H,000H,000H,000H,000H,000H,000H ; D_20     SPACE
1337      00 00
1338 1B76 30 78 78 30 30 00          DB      030H,078H,078H,030H,030H,000H,030H,000H ; D_21 !   EXCLAMATION
1339      30 00
1340 1B7E 6C 6C 6C 00 00 00          DB      06CH,06CH,06CH,000H,000H,000H,000H,000H ; D_22 "   QUOTATION
1341      00 00
1342 1B86 6C 6C FE 6C FE 6C          DB      06CH,06CH,0FEH,06CH,0FEH,06CH,06CH,000H ; D_23 #   LB.
1343      6C 00
1344 1B8E 30 7C C0 78 0C F8          DB      030H,07CH,0C0H,078H,00CH,0F8H,030H,000H ; D_24 $   DOLLAR SIGN
1345      30 00
1346 1B96 00 C6 CC 18 30 66          DB      000H,0C6H,0CCH,018H,030H,066H,0C6H,000H ; D_25 %   PERCENT
1347      C6 00
1348 1B9E 38 6C 38 76 DC CC          DB      038H,06CH,038H,076H,0DCH,0CCH,076H,000H ; D_26 &   AMPERSAND
1349      76 00
1350 1BA6 60 60 C0 00 00 00          DB      060H,060H,0C0H,000H,000H,000H,000H,000H ; D_27 '   APOSTROPHE
1351      00 00
1352 1BAE 18 30 60 60 60 30          DB      018H,030H,060H,060H,060H,030H,018H,000H ; D_28 (   L. PARENTHESIS
1353      18 00
1354 1BB6 60 30 18 18 18 30          DB      060H,030H,018H,018H,018H,030H,060H,000H ; D_29 )   R. PARENTHESIS
1355      60 00
1356 1BBE 00 66 3C FF 3C 66          DB      000H,066H,03CH,0FFH,03CH,066H,000H,000H ; D_2A *   ASTERISK
1357      00 00
1358 1BC6 00 30 30 FC 30 30          DB      000H,030H,030H,0FCH,030H,030H,000H,000H ; D_2B +   PLUS
1359      00 00
1360 1BCE 00 00 00 00 00 30          DB      000H,000H,000H,000H,000H,030H,030H,060H ; D_2C ,   COMMA
1361      30 60
1362 1BD6 00 00 00 FC 00 00          DB      000H,000H,000H,0FCH,000H,000H,000H,000H ; D_2D -   DASH
1363      00 00
1364 1BDE 00 00 00 00 00 30          DB      000H,000H,000H,000H,000H,030H,030H,000H ; D_2E .   PERIOD
1365      30 00
1366 1BE6 06 0C 18 30 60 C0          DB      006H,00CH,018H,030H,060H,0C0H,080H,000H ; D_2F /   SLASH
1367      80 00
1368
1369 1BEE 7C C6 CE DE F6 E6          DB      07CH,0C6H,0CEH,0DEH,0F6H,0E6H,07CH,000H ; D_30 0
1370      7C 00
1371 1BF6 30 70 30 30 30 30          DB      030H,070H,030H,030H,030H,030H,0FCH,000H ; D_31 1
1372      FC 00
1373 1BFE 78 CC 0C 38 60 CC          DB      078H,0CCH,00CH,038H,060H,0CCH,0FCH,000H ; D_32 2
1374      FC 00
1375 1C06 78 CC 0C 38 0C CC          DB      078H,0CCH,00CH,038H,00CH,0CCH,078H,000H ; D_33 3
1376      78 00
1377 1C0E 1C 3C 6C CC FE 0C          DB      01CH,03CH,06CH,0CCH,0FEH,00CH,01EH,000H ; D_34 4
1378      1E 00
1379 1C16 FC C0 F8 0C 0C CC          DB      0FCH,0C0H,0F8H,00CH,00CH,0CCH,078H,000H ; D_35 5
1380      78 00
1381 1C1E 38 60 C0 F8 CC CC          DB      038H,060H,0C0H,0F8H,0CCH,0CCH,078H,000H ; D_36 6
1382      78 00
1383 1C26 FC CC 0C 18 30 30          DB      0FCH,0CCH,00CH,018H,030H,030H,030H,000H ; D_37 7
1384      30 00
1385 1C2E 78 CC CC 78 CC CC          DB      078H,0CCH,0CCH,078H,0CCH,0CCH,078H,000H ; D_38 8
1386      78 00
1387 1C36 78 CC CC 7C 0C 18          DB      078H,0CCH,0CCH,07CH,00CH,018H,070H,000H ; D_39 9
1388      70 00
1389 1C3E 00 30 30 00 00 30          DB      000H,030H,030H,000H,000H,030H,030H,000H ; D_3A :   COLON
1390      30 00
1391 1C46 00 30 30 00 00 30          DB      000H,030H,030H,000H,000H,030H,030H,060H ; D_3B ;   SEMICOLON
1392      30 60
1393 1C4E 18 30 60 C0 60 30          DB      018H,030H,060H,0C0H,060H,030H,018H,000H ; D_3C <   LESS THAN
1394      18 00
1395 1C56 00 00 FC 00 00 FC          DB      000H,000H,0FCH,000H,000H,0FCH,000H,000H ; D_3D =   EQUAL
1396      00 00
1397 1C5E 60 30 18 0C 18 30          DB      060H,030H,018H,00CH,018H,030H,060H,000H ; D_3E >   GREATER THAN
1398      60 00
1399 1C66 78 CC 0C 18 30 00          DB      078H,0CCH,00CH,018H,030H,000H,030H,000H ; D_3F ?   QUESTION MARK
1400      30 00
1401
1402 1C6E 7C C6 DE DE DE C0          DB      07CH,0C6H,0DEH,0DEH,0DEH,0C0H,078H,000H ; D_40 @   AT
1403      78 00
1404 1C76 30 78 CC CC FC CC          DB      030H,078H,0CCH,0CCH,0FCH,0CCH,0CCH,000H ; D_41 A
1405      CC 00
1406 1C7E FC 66 66 7C 66 66          DB      0FCH,066H,066H,07CH,066H,066H,0FCH,000H ; D_42 B
1407      FC 00
1408 1C86 3C 66 C0 C0 C0 66          DB      03CH,066H,0C0H,0C0H,0C0H,066H,03CH,000H ; D_43 C
1409      3C 00
1410 1C8E F8 6C 66 66 66 6C          DB      0F8H,06CH,066H,066H,066H,06CH,0F8H,000H ; D_44 D
1411      F8 00
1412 1C96 FE 62 68 78 68 62          DB      0FEH,062H,068H,078H,068H,062H,0FEH,000H ; D_45 E
1413      FE 00
1414 1C9E FE 62 68 78 68 60          DB      0FEH,062H,068H,078H,068H,060H,0F0H,000H ; D_46 F
1415      F0 00
1416 1CA6 3C 66 C0 C0 CE 66          DB      03CH,066H,0C0H,0C0H,0CEH,066H,03EH,000H ; D_47 G
1417      3E 00
1418 1CAE CC CC CC FC CC CC          DB      0CCH,0CCH,0CCH,0FCH,0CCH,0CCH,0CCH,000H ; D_48 H
1419      CC 00
1420 1CB6 78 30 30 30 30 30          DB      078H,030H,030H,030H,030H,030H,078H,000H ; D_49 I
1421      78 00
1422 1CBE 1E 0C 0C 0C CC CC          DB      01EH,00CH,00CH,00CH,0CCH,0CCH,078H,000H ; D_4A J
1423      78 00
1424 1CC6 E6 66 6C 78 6C 66          DB      0E6H,066H,06CH,078H,06CH,066H,0E6H,000H ; D_4B K
1425      E6 00
1426 1CCE F0 60 60 60 62 66          DB      0F0H,060H,060H,060H,062H,066H,0FEH,000H ; D_4C L
1427      FE 00
1428 1CD6 C6 EE FE FE D6 C6          DB      0C6H,0EEH,0FEH,0FEH,0D6H,0C6H,0C6H,000H ; D_4D M
1429      C6 00
1430 1CDE C6 E6 F6 DE CE C6          DB      0C6H,0E6H,0F6H,0DEH,0CEH,0C6H,0C6H,000H ; D_4E N
1431      C6 00
1432 1CE6 38 6C C6 C6 C6 6C          DB      038H,06CH,0C6H,0C6H,0C6H,06CH,038H,000H ; D_4F O
1433      38 00
1434
1435 1CEE FC 66 66 7C 60 60          DB      0FCH,066H,066H,07CH,060H,060H,0F0H,000H ; D_50 P
1436      F0 00
1437 1CF6 78 CC CC CC DC 78          DB      078H,0CCH,0CCH,0CCH,0DCH,078H,01CH,000H ; D_51 Q
1438      1C 00
1439 1CFE FC 66 66 7C 6C 66          DB      0FCH,066H,066H,07CH,06CH,066H,0E6H,000H ; D_52 R
1440      E6 00
1441 1D06 78 CC E0 70 1C CC          DB      078H,0CCH,0E0H,070H,01CH,0CCH,078H,000H ; D_53 S
1442      78 00
1443 1D0E FC B4 30 30 30 30          DB      0FCH,0B4H,030H,030H,030H,030H,078H,000H ; D_54 T
1444      78 00
1445 1D16 CC CC CC CC CC CC          DB      0CCH,0CCH,0CCH,0CCH,0CCH,0CCH,0FCH,000H ; D_55 U
1446      FC 00
1447 1D1E CC CC CC CC CC 78          DB      0CCH,0CCH,0CCH,0CCH,0CCH,078H,030H,000H ; D_56 V
1448      30 00
1449 1D26 C6 C6 C6 D6 FE EE          DB      0C6H,0C6H,0C6H,0D6H,0FEH,0EEH,0C6H,000H ; D_57 W
```

```
                C6 00
1D2E C6 C6 6C 38 38 6C          DB      0C6H,0C6H,06CH,038H,038H,06CH,0C6H,000H ; D_58 X
                C6 00
1D36 CC CC CC 78 30 30          DB      0CCH,0CCH,0CCH,078H,030H,030H,078H,000H ; D_59 Y
                78 00
1D3E FE C6 8C 18 32 66          DB      0FEH,0C6H,08CH,018H,032H,066H,0FEH,000H ; D_5A Z
                FE 00
1D46 78 60 60 60 60 60          DB      078H,060H,060H,060H,060H,060H,078H,000H ; D_5B [  LEFT BRACKET
                78 00
1D4E C0 60 30 18 0C 06          DB      0C0H,060H,030H,018H,00CH,006H,002H,000H ; D_5C \  BACKSLASH
                02 00
1D56 78 18 18 18 18 18          DB      078H,018H,018H,018H,018H,018H,078H,000H ; D_5D ]  RIGHT BRACKET
                78 00
1D5E 10 38 6C C6 00 00          DB      010H,038H,06CH,0C6H,000H,000H,000H,000H ; D_5E ^  CIRCUMFLEX
                00 00
1D66 00 00 00 00 00 00          DB      000H,000H,000H,000H,000H,000H,000H,0FFH ; D_5F _  UNDERSCORE
                00 FF

1D6E 30 30 18 00 00 00          DB      030H,030H,018H,000H,000H,000H,000H,000H ; D_60 `  APOSTROPHE REV
                00 00
1D76 00 00 78 0C 7C CC          DB      000H,000H,078H,00CH,07CH,0CCH,076H,000H ; D_61 a
                76 00
1D7E E0 60 60 7C 66 66          DB      0E0H,060H,060H,07CH,066H,066H,0DCH,000H ; D_62 b
                DC 00
1D86 00 00 78 CC C0 CC          DB      000H,000H,078H,0CCH,0C0H,0CCH,078H,000H ; D_63 c
                78 00
1D8E 1C 0C 0C 7C CC CC          DB      01CH,00CH,00CH,07CH,0CCH,0CCH,076H,000H ; D_64 d
                76 00
1D96 00 00 78 CC FC C0          DB      000H,000H,078H,0CCH,0FCH,0C0H,078H,000H ; D_65 e
                78 00
1D9E 38 6C 60 F0 60 60          DB      038H,06CH,060H,0F0H,060H,060H,0F0H,000H ; D_66 f
                F0 00
1DA6 00 00 76 CC CC 7C          DB      000H,000H,076H,0CCH,0CCH,07CH,00CH,0F8H ; D_67 g
                0C F8
1DAE E0 60 6C 76 66 66          DB      0E0H,060H,06CH,076H,066H,066H,0E6H,000H ; D_68 h
                E6 00
1DB6 30 00 70 30 30 30          DB      030H,000H,070H,030H,030H,030H,078H,000H ; D_69 i
                78 00
1DBE 0C 00 0C 0C 0C CC          DB      00CH,000H,00CH,00CH,00CH,0CCH,0CCH,078H ; D_6A j
                CC 78
1DC6 E0 60 66 6C 78 6C          DB      0E0H,060H,066H,06CH,078H,06CH,0E6H,000H ; D_6B k
                E6 00
1DCE 70 30 30 30 30 30          DB      070H,030H,030H,030H,030H,030H,078H,000H ; D_6C l
                78 00
1DD6 00 00 CC FE FE D6          DB      000H,000H,0CCH,0FEH,0FEH,0D6H,0C6H,000H ; D_6D m
                C6 00
1DDE 00 00 F8 CC CC CC          DB      000H,000H,0F8H,0CCH,0CCH,0CCH,0CCH,000H ; D_6E n
                CC 00
1DE6 00 00 78 CC CC CC          DB      000H,000H,078H,0CCH,0CCH,0CCH,078H,000H ; D_6F o
                78 00

1DEE 00 00 DC 66 66 7C          DB      000H,000H,0DCH,066H,066H,07CH,060H,0F0H ; D_70 p
                60 F0
1DF6 00 00 76 CC CC 7C          DB      000H,000H,076H,0CCH,0CCH,07CH,00CH,01EH ; D_71 q
                0C 1E
1DFE 00 00 DC 76 66 60          DB      000H,000H,0DCH,076H,066H,060H,0F0H,000H ; D_72 r
                F0 00
1E06 00 00 7C C0 78 0C          DB      000H,000H,07CH,0C0H,078H,00CH,0F8H,000H ; D_73 s
                F8 00
1E0E 10 30 7C 30 30 34          DB      010H,030H,07CH,030H,030H,034H,018H,000H ; D_74 t
                18 00
1E16 00 00 CC CC CC CC          DB      000H,000H,0CCH,0CCH,0CCH,0CCH,076H,000H ; D_75 u
                76 00
1E1E 00 00 CC CC CC 78          DB      000H,000H,0CCH,0CCH,0CCH,078H,030H,000H ; D_76 v
                30 00
1E26 00 00 C6 D6 FE FE          DB      000H,000H,0C6H,0D6H,0FEH,0FEH,06CH,000H ; D_77 w
                6C 00
1E2E 00 00 C6 6C 38 6C          DB      000H,000H,0C6H,06CH,038H,06CH,0C6H,000H ; D_78 x
                C6 00
1E36 00 00 CC CC CC 7C          DB      000H,000H,0CCH,0CCH,0CCH,07CH,00CH,0F8H ; D_79 y
                0C F8
1E3E 00 00 FC 98 30 64          DB      000H,000H,0FCH,098H,030H,064H,0FCH,000H ; D_7A z
                FC 00
1E46 1C 30 30 E0 30 30          DB      01CH,030H,030H,0E0H,030H,030H,01CH,000H ; D_7B {  LEFT BRACE
                1C 00
1E4E 18 18 18 00 18 18          DB      018H,018H,018H,000H,018H,018H,018H,000H ; D_7C |  BROKEN STROKE
                18 00
1E56 E0 30 30 1C 30 30          DB      0E0H,030H,030H,01CH,030H,030H,0E0H,000H ; D_7D }  RIGHT BRACE
                E0 00
1E5E 76 DC 00 00 00 00          DB      076H,0DCH,000H,000H,000H,000H,000H,000H ; D_7E ~  TILDE
                00 00
1E66 00 10 38 6C C6 C6          DB      000H,010H,038H,06CH,0C6H,0C6H,0FEH,000H ; D_7F    DELTA
                FE 00


                        ;-----  TIME OF DAY

                        ;;-     ORG     0FE6EH
1E6E                            ORG     01E6EH
= 1E6E                  TIME_OF_DAY     EQU     $
1E6E E9 0000 E                  JMP     TIME_OF_DAY_I           ; VECTOR ON TO MOVED BIOS CODE

                        ;-----  TIMER INTERRUPT

                        ;;-     ORG     0FEA5H
1EA5                            ORG     01EA5H
= 1EA5                  TIMER_INT       EQU     $
1EA5 E9 0000 E                  JMP     TIMER_INT_I             ; VECTOR ON TO MOVED BIOS CODE
```

SECTION 5

```
                        PAGE
                        ;-----  VECTOR TABLE
                        ;;-     ORG     0FEF3H
                        ;       ORG     01EF3H                  ;       AT LOCATION 0FEF3H
1EF3                    VECTOR_TABLE    LABEL   WORD            ; VECTOR TABLE VALUES FOR POST TESTS
1EF3 1EA6 R                     DW      OFFSET TIMER_INT        ; INT 08H - HARDWARE TIMER 0    IRQ  0
1EF5 0987 R                     DW      OFFSET KB_INT           ; INT 09H - KEYBOARD            IRQ  1
1EF7 0000 E                     DW      OFFSET  DTI             ; INT 0AH - SLAVE INTERRUPT INPUT
1EF9 0000 E                     DW      OFFSET  DII             ; INT 0BH -                     IRQ  3
1EFB 0000 E                     DW      OFFSET  DII             ; INT 0CH -                     IRQ  4
1EFD 0000 E                     DW      OFFSET  DII             ; INT 0DH -                     IRQ  5
1EFF 0F57 R                     DW      OFFSET DISK_INT         ; INT 0EH - DISKETTE            IRQ  6
1F01 0000 E                     DW      OFFSET  DII             ; INT 0FH -                     IRQ  7

                        ;-----  SOFTWARE INTERRUPTS  ( BIOS CALLS AND POINTERS )

1F03 1065 R                     DW      OFFSET VIDEO_IO         ; INT 10H -- VIDEO DISPLAY
1F05 184D R                     DW      OFFSET EQUIPMENT        ; INT 11H -- GET EQUIPMENT FLAG WORD
1F07 1841 R                     DW      OFFSET MEMORY_SIZE_DET  ; INT 12H -- GET REAL MODE MEMORY SIZE
1F09 0C59 R                     DW      OFFSET DISKETTE_IO      ; INT 13H -- DISKETTE
1F0B 0739 R                     DW      OFFSET RS232_IO         ; INT 14H -- COMMUNICATION ADAPTER
1F0D 1859 R                     DW      OFFSET CASSETTE_IO      ; INT 15H -- EXPANDED BIOS FUNCTION CALL
1F0F 082E R                     DW      OFFSET KEYBOARD_IO      ; INT 16H -- KEYBOARD INPUT
1F11 0FD2 R                     DW      OFFSET PRINTER_TO       ; INT 17H -- PRINTER OUTPUT
1F13 0000                       DW      00000H                  ; INT 18H -- 0F600H INSERTED FOR BASIC
1F15 06F2 R                     DW      OFFSET BOOT_STRAP       ; INT 19H -- BOOT FROM SYSTEM MEDIA
1F17 1E6E R                     DW      OFFSET TIME_OF_DAY      ; INT 1AH -- TIME OF DAY
1F19 1F53 R                     DW      OFFSET DUMMY_RETURN     ; INT 1BH -- KEYBOARD BREAK ADDRESS
1F1B 1F53 R                     DW      OFFSET DUMMY_RETURN     ; INT 1CH -- TIMER BREAK ADDRESS
1F1D 10A4 R                     DW      OFFSET VIDEO_PARMS      ; INT 1DH -- VIDEO PARAMETERS
1F1F 0FC7 R                     DW      OFFSET DISK_BASE        ; INT 1EH -- DISKETTE PARAMETERS
1F21 0000                       DW      00000H                  ; INT 1FH -- POINTER TO VIDEO EXTENSION

1F23                    SLAVE_VECTOR_TABLE      LABEL WORD      ; ( INTERRUPT 70H THRU 7FH )

1F23 0000 E                     DW      OFFSET RTC_INT          ; INT 70H - REAL TIME CLOCK     IRQ  8
1F25 0000 E                     DW      OFFSET RE_DIRECT        ; INT 71H - REDIRECT TO INT 0AH IRQ  9
1F27 0000 E                     DW      OFFSET  DTI             ; INT 72H -                     IRQ 10
1F29 0000 E                     DW      OFFSET  DII             ; INT 73H -                     IRQ 11
1F2B 0000 E                     DW      OFFSET  DII             ; INT 74H -                     IRQ 12
1F2D 0000 E                     DW      OFFSET INT_287          ; INT 75H -  -MATH COPROCESSOR  IRQ 13
1F2F 0000 E                     DW      OFFSET  DIT             ; INT 76H -  -FIXED DISK        IRQ 14
1F31 0000 E                     DW      OFFSET  DII             ; INT 77H -                     IRQ 15

                        ;-----  DUMMY INTERRUPT HANDLER

                        ;;-     ORG     0FF53H
                        ;       ORG     01F53H

1F53                    DUMMY_RETURN    EQU     $               ; BIOS DUMMY (NULL) INTERRUPT RETURN

1F53 CF                         IRET

                        ;-----  PRINT SCREEN

                        ;;-     ORG     0FF54H
                        ;       ORG     01F54H
1F54                    PRINT_SCREEN    EQU     $
1F54 E9 0000 E                  JMP     PRINT_SCREEN_1          ; VECTOR ON TO MOVED BIOS CODE
                        .LIST                                   ; TUTOR
                        ;---------------------------------------
                        ;                                      :
                        ;       POWER ON RESET VECTOR          :
                        ;                                      :
                        ;---------------------------------------
                        ;;-     ORG     0FFF0H
                        ;       ORG     01FF0H

                        ;-----  POWER ON RESET

1FF0                    P_O_R   LABEL  FAR                      ; POWER ON RESTART EXECUTION LOCATION

1FF0 EA                         DB      0EAH                    ; HARD CODE FAR JUMP TO SET
1FF1 005B R                      DW      OFFSET  RESET          ;  OFFSET
1FF3 F000                        DW      0F000H                 ;   SEGMENT

1FF5 30 34 2F 32 31 2F          DB      '04/21/86'              ; RELEASE MARKER
     38 36

1FFE                            ORG     01FFEH
1FFE FC                         DB      MODEL_BYTE              ; THIS PC'S ID ( MODEL BYTE )

1FFF                    CODE    ENDS                            ; CHECKSUM AT LAST LOCATION
                        END
```

SECTION 6. INSTRUCTION SET

80286 Instruction Set

Data Transfer

MOV = move

Register to Register/Memory

1000100w	mod reg r/w

Register/Memory to Register

1000101w	mod reg r/w

Immediate to Register/Memory

1100011w	mod 000 r/w	data	data if w = 1

Immediate to Register

1011wreg	data	data if w = 1

Memory to Accumulator

1010000w	addr-low	addr-high

Accumulator to Memory

1010001w	addr-low	addr-high

Register/Memory to Segment Register

10001110	mod0reg r/w reg ≠ 01

Segment Register to Register/Memory

10001100	mod0reg r/w

PUSH = Push

Memory

11111111	mod110 r/w

Register

01010reg

Segment Register

000reg110

Immediate

011010s0	data	data if s = 0

PUSHA = Push All

01100000

POP = Pop

Memory

10001111	mod000 r/m

Register

01011reg

Segment Register

000reg111	reg ≠ 01

POPA = Pop All

01100001

XCHG = Exchange

Register/Memory with Register

1000011w	mod reg r/m

Register with Accumulator

10010reg

IN = Input From

Fixed Port

1110010w	port

Variable Port

1110110w

OUT = Output To

Fixed Port

1110011w	port

Variable Port

1110111w

XLAT = Translate Byte to AL

11010111

LEA = Load EA to Register

10001101	mod reg r/m

LDS = Load Pointer to DS

11000101	mod reg r/m mod ≠ 11

LES = Load Pointer to ES

11000100	mod reg r/m mod ≠ 11

LAHF = Load AH with Flags

10011111

SAHF = Store AH with Flags

10011110

PUSHF = Push Flags

10011100

POPF = Pop Flags

10011101

Arithmetic

ADD = Add

Register/Memory with Register to Either

0000000w	mod reg r/m

Immediate to Register Memory

100000sw	mod000 r/m	data	data if sw = 01

Immediate to Accumulator

0000010w	data	data if w = 1

ADC = Add with Carry

Register/Memory with Register to Either

000100dw	mod reg r/m

Immediate to Register/Memory

100000sw	mod000 r/m	data	data if sw = 01

Immediate to Accumulator

0001010w	data	data if w = 1

INC = Increment

Register/Memory

1111111w	mod000 r/m

Register

01000reg

SUB = Subtract

Register/Memory with Register to Either

001010dw	mod reg r/m

Immediate from Register/Memory

100000sw	mod101 r/m	data	data if sw = 01

Immediate from Accumulator

0010110w	data	data if w = 1

SBB = Subtract with Borrow

Register/Memory with Register to Either

000110dw	mod reg r/m

Immediate to Register/Memory

100000sw	mod011 r/m	data	data if sw = 01

Immediate to Accumulator

0001110w	data	data if w = 1

DEC = Decrement

Register/Memory

1111111w	mod001 r/m

Register

01001reg

CMP = Compare

Register/Memory with Register

0011101w	mod reg r/m

Register with Register/Memory

0011100w	mod reg r/m

Immediate with Register/Memory

100000sw	mod111 r/m	data	data if sw = 01

Immediate with Accumulator

0001110w	data	data if w = 1

NEG = Change Sign

1111011w	mod011 r/m

AAA = ASCII Adjust for Add

00110111

DAA = Decimal Adjust for Add

00100111

AAS = ASCII Adjust for Subtract

00111111

DAS = Decimal Adjust for Subtract

00110111

MUL = Multiply (Unsigned)

1111011w	mod100 r/m

IMUL = Integer Multiply (Signed)

1111011w	mod101 r/m

IIMUL = Integer Immediate Multiply (Signed)

011010s1	mod reg r/m	Data	Data if s = 0

DIV = Divide (Unsigned)

1111011w	mod110 r/m

IDIV = Integer Divide (Signed)

1111011w	mod111 r/m

AAM = ASCII Adjust for Multiply

11010100	00001010

AAD = ASCII Adjust for Divide

11010101	00001010

CBW = Convert Byte to Word

10011000

CWD = Convert Word to Double Word

10011001

Logic

Shift/Rotate Instructions

Register/Memory by 1

1101000w	mod TTT r/m

Register/Memory by CL

1101001w	mod TTT r/m

Register/Memory by Count

1100000w	mod TTT r/m	count

T T T	Instruction
000	ROL
001	ROR
010	RCL
011	RCR
100	SHL/SAL
101	SHR
111	SAR

AND = And

Register/Memory and Register to Either

001000dw	mod reg r/m

Immediate to Register/Memory

1000000w	mod000 r/m	data	data if w = 1

Immediate to Accumulator

0010010w	data	data if w = 1

TEST = AND Function to Flags; No Result

Register/Memory and Register

1000010w	mod reg r/m

Immediate Data and Register/Memory

1111011w	mod000 r/m	data	data if w = 1

Immediate to Accumulator

0000110w	data	data if w = 1

Or = Or

Register/Memory and Register to Either

000010dw	mod reg r/m

Immediate to Register/Memory

1000000w	mod001 r/m	data	data if w = 1

Immediate to Accumulator

0000110w	data	data if w = 1

XOR = Exclusive OR

Register/Memory and Register to Either

001100dw	mod reg r/m

Immediate to Register/Memory

1000000w	mod110 r/m	data	data if w = 1

Immediate to Accumulator

0010010w	data	data if w = 1

NOT = Invert Register/Memory

1111011w	mod010 r/m

String Manipulation

MOVS = Move Byte Word

1010010w

CMPS = Compare Byte Word

1010011w

SCAS = Scan Byte Word

1010111w

LODS = Load Byte Word to AL/AX

1010110w

STOS = Store Byte Word from AL/AX

1010101w

INS = Input Byte from DX Port

0110110w

OUTS = Output Byte Word to DX Port

0110111w

REP/REPNE, REPZ/REPNZ = Repeat String

Repeat Move String

11110011	1010010w

Repeat Compare String (z/Not z)

1111001z	1010011w

Repeat Scan String (z/Not z)

1111001z	1010111w

Repeat Load String

11110011	1010110w

Repeat Store String

11110011	1010101w

Repeat Input String

11110011	0110110w

Repeat Output String

11110011	1010011w

Control Transfer

CALL = Call

Direct Within Segment

11101000	disp-low	disp-high

Register/Memory Indirect Within Segment

11111111	mod010 r/m

Direct Intersegment

10011010	Segment Offset	Segment Selector

Indirect Intersegment

11111111	mod011 r/m (mod ≠ 11)

JMP = Unconditional Jump

Short/Long

11101011	disp-low

Direct within Segment

11101001	disp-low	disp-high

Register/Memory Indirect Within Segment

11111111	mod100 r/m

Direct Intersegment

11101010	Segment Offset	Segment Selector

Indirect Intersegment

11111111	mod101 r/m (mod ≠ 11)

RET = Return from Call

Within Segment

11000011

Within Segment Adding Immediate to SP

11000010	data-low	data-high

Intersegment

11001011

Intersegment Adding Immediate to SP

11001010	data-low	data-high

JE/JZ = Jump on Equal/Zero

01110100	disp

JL/JNGE = Jump on Less/Not Greater, or Equal

01111100	disp

JLE/JNG = Jump on Less, or Equal/Not Greater

01111110	disp

JB/JNAE = Jump on Below/Not Above, or Equal

01110010	disp

JBE/JNA = Jump on Below, or Equal/Not Above

01110110	disp

JP/JPE = Jump on Parity/Parity Even

01111010	disp

JO = Jump on Overflow

01110000	disp

JS = Jump on Sign

01111000	disp

JNE/JNZ = Jump on Not Equal/Not Zero

01110101	disp

JNL/JGE = Jump on Not Less/Greater, or Equal

01111101	disp

JNLE/JG = Jump on Not Less, or Equal/Greater

01111111	disp

JNB/JAE = Jump on Not Below/Above, or Equal

01110011	disp

JNBE/JA = Jump on Not Below, or Equal/Above

01110111	disp

JNP/JPO = Jump on Not Parity/Parity Odd

01111011	disp

JNO = Jump on Not Overflow

01110001	disp

JNS = Jump on Not Sign

01111011	disp

LOOP = Loop CX Times

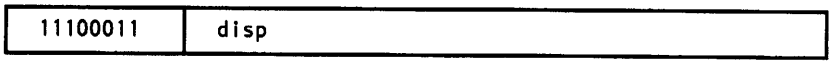
11100010	disp

LOOPZ/LOOPE = Loop while Zero/Equal

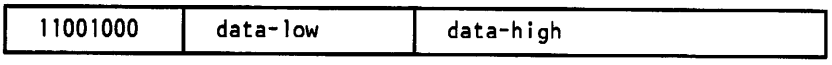
11100001	disp

LOOPNZ/LOOPNE = Loop while Not Zero/Not Equal

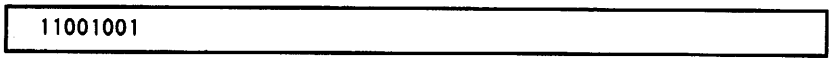
JCXZ = Jump on CX Zero



ENTER = Enter Procedure

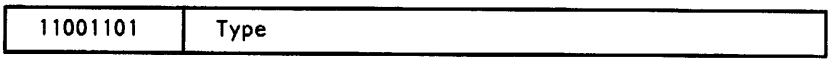


LEAVE = Leave Procedure

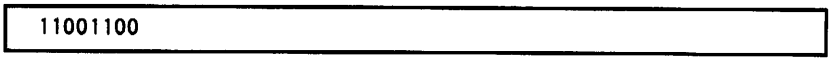


INT = Interrupt

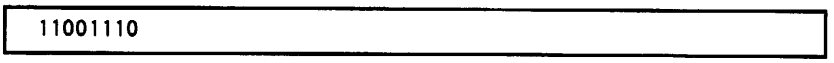
Type Specified



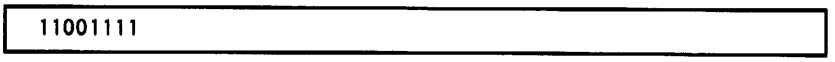
Type 3



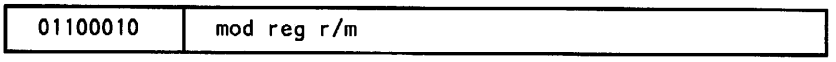
INTO = Interrupt on Overflow



IRET = Interrupt Return



BOUND = Detect Value Out of Range



Processor Control

CLC = Clear Carry

11111000

CMC = Complement Carry

11110101

STC = Set Carry

11111001

CLD = Clear Direction

11111100

STD = Set Direction

11111101

CLI Clear Interrupt

11111010

STI = Set Interrupt

11111011

HLT = Halt

11110100

WAIT = Wait

10011011

LOCK = Bus Lock Prefix

11110000

CTS = Clear Task Switched Flag

00001111	00000110

ESC = Processor Extension Escape

11011TTT	modLLL r/m

Protection Control

LGDT = Load Global Descriptor Table Register

00001111	00000001	mod010 r/m

SGDT = Store Global Descriptor Table Register

00001111	00000001	mod000 r/m

LIDT = Load Interrupt Descriptor Table Register

00001111	00000001	mod011 r/m

SIDT = Store Interrupt Descriptor Table Register

00001111	00000001	mod001 r/m

LLDT = Load Local Descriptor Table Register from Register/Memory

00001111	00000000	mod010 r/m

SLDT = Store Local Descriptor Table Register from Register/Memory

00001111	00000000	mod000 r/m

LTR = Load Task Register from Register/Memory

00001111	00000000	mod011 r/m

STR = Store Task Register to Register/Memory

00001111	00000000	mod001 r/m

LMSW = Load Machine Status Word from Register/Memory

00001111	00000001	mod110 r/m

SMSW = Store Machine Status Word

00001111	00000001	mod100 r/m

LAR = Load Access Rights from Register/Memory

00001111	00000010	mod reg r/m

LSL = Load Segment Limit from Register/Memory

00001111	00000011	mod reg r/m

ARPL = Adjust Requested Privilege Level from Register/Memory

	01100011	mod reg r/m

VERR = Verify Read Access; Register/Memory

00001111	00000000	mod100 r/m

VERR = Verify Write Access

00001111	00000000	mod101 r/m

The effective address (EA) of the memory operand is computed according to the mod and r/m fields:

If mod = 11, then r/m is treated as a reg field.
If mod = 00, then disp = 0, disp-low and disp-high are absent.
If mod = 01, then disp = disp-low sign-extended to 16 bits, disp-high is absent.
If mod = 10, then disp = disp-high:disp-low.

If r/m = 000, then EA = (BX) + (SI) + DISP
If r/m = 001, then EA = (BX) + (SI) + DISP
If r/m = 010, then EA = (BP) + (SI) + DISP
If r/m = 011, then EA = (BP) + (DI) + DISP
If r/m = 100, then EA = (SI) + DISP
If r/m = 101, then EA = (DI) + DISP
If r/m = 110, then EA = (BP) + DISP
If r/m = 111, then EA = (BX) + DISP

DISP follows the second byte of the instruction (before data if required).

Note: An exception to the above statements occurs when mod=00 and r/m=110, in which case EA = disp-high; disp-low.

Segment Override Prefix

The 2-bit and 3-bit reg fields are defined as follows:

2-Bit reg Field

reg	Segment Register	reg	Segment Register
00	ES	10	SS
01	CS	11	DS

3-Bit reg Field

16-bit (w = 1)	8-bit (w = 0)
000 AX	000 AL
001 CX	001 CL
010 DX	010 DL
011 BX	011 BL
100 SP	100 AH
101 BP	101 CH
110 SI	110 DH
111 DI	111 BH

The physical addresses of all operands addressed by the BP register are computed using the SS segment register. The physical addresses of the destination operands of the string primitive operations (those addressed by the DI register) are computed using the ES segment, which may not be overridden.

80287 Coprocessor Instruction Set

The following is an instruction set summary for the 80287 coprocessor. In the following, the bit pattern for escape is 11011.

Data Transfer

FLD = Load

Integer/Real Memory to ST(0)

escape MF 1	mod 000 r/m

Long Integer Memory to ST(0)

escape 111	mod 101 r/m

Temporary Real Memory to ST(0)

escape 011	mod 101 r/m

BCD Memory to ST(0)

escape 111	mod 100 r/m

ST(i) to ST(0)

escape 001	11000ST(i)

FST = Store

ST(0) to Integer/Real Memory

escape MF 1	mod 010 r/m

ST(0) to ST(i)

escape 101	11010 ST(i)

FSTP = Store and Pop

ST(0) to Integer/Real Memory

escape MF 1	mod 011 r/m

ST(0) to Long Integer Memory

escape 111	mod 111 r/m

ST(0) to Temporary Real Memory

escape 011	mod 111 r/m

ST(0) to BCD Memory

escape 111	mod 110 r/m

ST(0) to ST(i)

escape 101	11011 ST(i)

FXCH = Exchange ST(i) and ST(0)

escape 001	11001 ST(i)

Comparison

FCOM = Compare

Integer/Real Memory to ST(0)

escape MF 0	mod 010 r/m

ST(i) to ST(0)

escape 000	11010 ST(i)

FCOMP = Compare and Pop

Integer/Real Memory to ST(0)

escape MF 0	mod 011 r/m

ST(i) to ST(0)

escape 000	11010 ST(i)

FCOMPP = Compare ST(i) to ST(0) and Pop Twice

escape 110	11011001

FTST = Test ST(0)

escape 001	11100100

FXAM = Examine ST(0)

escape 001	11100101

Constants

FLDZ = Load + 0.0 into ST(0)

escape 000	11101110

FLD1 = Load + 1.0 into ST(0)

escape 001	11101000

FLDP1 = Load π into ST(0)

escape 001	11101011

FLDL2T = Load $\log_2 10$ into ST(0)

escape 001	11101001

FLDLG2 = Load $\log_{10} 2$ into ST(0)

escape 001	11101100

FLDLN2 = Load $\log_e 2$ into ST(0)

escape 001	11101101

Arithmetic

FADD = Addition

Integer/Real Memory with ST(0)

escape MF 0	mod 000 r/m

ST(i) and ST(0)

escape dP0	11000 ST(i)

FSUB = Subtraction

Integer/Real Memory with ST(0)

escape MF 0	mod 10R r/m

ST(i) and ST(0)

escape dP0	1110R r/m

FMUL = Multiplication

Integer/Real Memory with ST(0)

escape MF 0	mod 001 r/m

ST(i) and ST(0)

escape dP0	11001 r/m

FDIV = Division

Integer/Real Memory with ST(0)

escape MF 0	mod 11R r/m

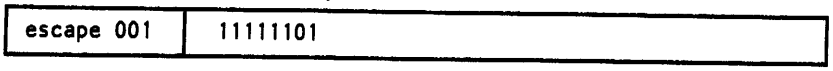
ST(i) and ST(0)

escape dP0	1111R r/m

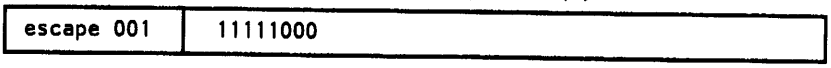
FSQRT = Square Root of ST(0)

escape 001	11111010

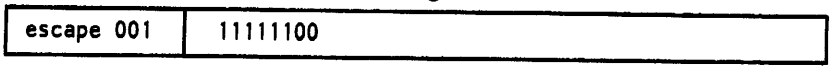
FSCALE = Scale ST(0) by ST(1)



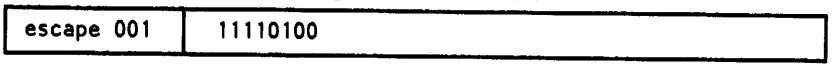
FPREM = Partial Remainder of ST(0) + ST(1)



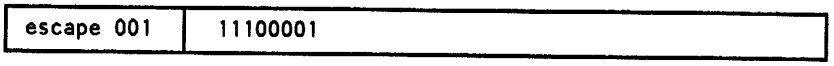
FRNDINT = Round ST(0) to Integer



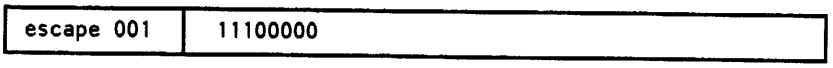
FXTRACT = Extract Components of ST(0)



FABS = Absolute Value of ST(0)

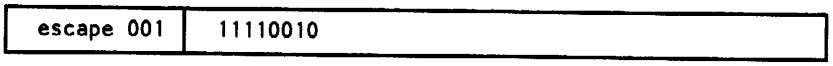


FCHS = Change Sign of ST(0)



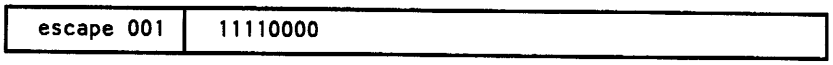
Transcendental

FPTAN = Partial Tangent of ST(0)

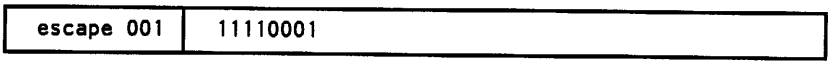


FPATAN = Partial Arctangent of ST(0) ÷ ST(1)

F2XM1 = $2^{ST(0)}$ -1



FYL2X = ST(1) x Log_2 [ST(0)]



FYL2XP1 = ST(1) x Log_2 [ST(0) + 1]

escape 001	11111001

FINIT = Initialize NPX

escape 011	11100011

FSETPM = Enter Protected Mode

escape 011	11100100

FSTSWAX = Store Control Word

escape 111	11100000

FLDCW = Load Control Word

escape 001	mod 101 r/m

FSTCW = Store Control Word

escape 001	mod 111 r/m

FSTSW = Store Status Word

escape 101	mod 101 r/m

FCLEX = Clear Exceptions

escape 011	11100010

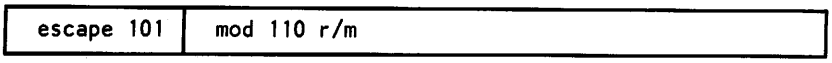
FSTENV = Store Environment

escape 001	mod 110 r/m

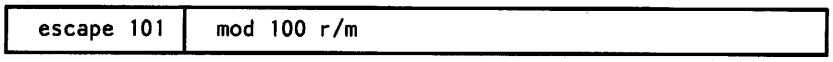
FLDENV = Load Environment

escape 001	mod 100 r/m

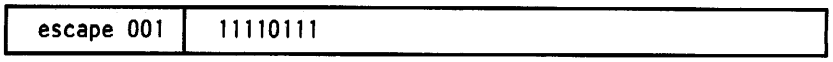
FSAVE = Save State



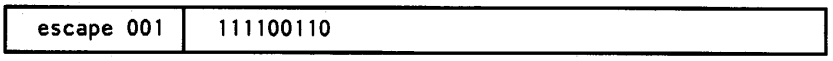
FRSTOR = Restore State



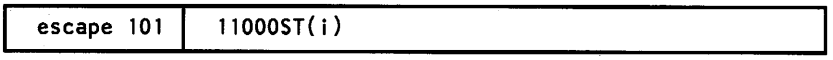
FINCSTP = Increment Stack Pointer



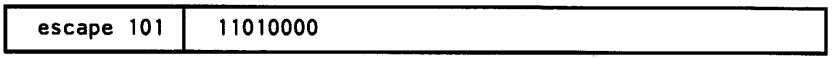
FDECSTP = Decrement Stack Pointer



FFREE = Free ST(i)



FNOP = No Operation



MF is assigned as follows:

MF	Memory Format
00	32-bit Real
01	32-bit Integer
10	64-bit Real
11	16-bit Integer

The other abbreviations are as follows:

Term	Definition	Bit = 0	Bit ≠ 0
ST	Stack top	Stack top	(i)= ith register from the top
d	Destination	Dest. is ST(0)	Dest. is ST(i)
P	Pop	No pop	Pop
R	Reverse*	Dest. (op) source	Source (op) dest.
* When d=1, reverse the sense of R.			

Notes:

SECTION 7. CHARACTERS, KEYSTROKES, AND COLORS

Character Codes

Value		As Characters			As Text Attributes		
					Color/Graphics Monitor Adapter		IBM Monochrome Display Adapter
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground	
00	0	Blank (Null)	Ctrl 2		Black	Black	Non-Display
01	1	☺	Ctrl A		Black	Blue	Underline
02	2	☻	Ctrl B		Black	Green	Normal
03	3	♥	Ctrl C		Black	Cyan	Normal
04	4	♦	Ctrl D		Black	Red	Normal
05	5	♣	Ctrl E		Black	Magenta	Normal
06	6	♠	Ctrl F		Black	Brown	Normal
07	7	•	Ctrl G		Black	Light Grey	Normal
08	8	◘	Ctrl H, Backspace, Shift Backspace		Black	Dark Grey	Non-Display
09	9	○	Ctrl I		Black	Light Blue	High Intensity Underline
0A	10	◙	Ctrl J, Ctrl ↵		Black	Light Green	High Intensity
0B	11	♂	Ctrl K		Black	Light Cyan	High Intensity
0C	12	♀	Ctrl L		Black	Light Red	High Intensity
0D	13	♪	Ctrl M, ↵, Shift ↵		Black	Light Magenta	High Intensity
0E	14	♫	Ctrl N		Black	Yellow	High Intensity
0F	15	☼	Ctrl O		Black	White	High Intensity
10	16	►	Ctrl P		Blue	Black	Normal
11	17	◄	Ctrl Q		Blue	Blue	Underline
12	18	↕	Ctrl R		Blue	Green	Normal
13	19	‼	Ctrl S		Blue	Cyan	Normal
14	20	¶	Ctrl T		Blue	Red	Normal
15	21	§	Ctrl U		Blue	Magenta	Normal
16	22	▬	Ctrl V		Blue	Brown	Normal
17	23	↨	Ctrl W		Blue	Light Grey	Normal

Value		As Characters			As Text Attributes		
					Color/Graphics Monitor Adapter		IBM Monochrome Display Adapter
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground	
18	24	↑	Ctrl X		Blue	Dark Grey	High Intensity
19	25	↓	Ctrl Y		Blue	Light Blue	High Intensity Underline
1A	26	→	Ctrl Z		Blue	Light Green	High Intensity
1B	27	←	Ctrl [, Esc, Shift Esc, Crtl Esc		Blue	Light Cyan	High Intensity
1C	28	∟	Ctrl \		Blue	Light Red	High Intensity
1D	29	↔	Ctrl]		Blue	Light Magenta	High Intensity
1E	30	▲	Ctrl 6		Blue	Yellow	High Intensity
1F	31	▼	Ctrl —		Blue	White	High Intensity
20	32	Blank Space	Space Bar, Shift, Space, Ctrl Space, Alt Space		Green	Black	Normal
21	33	!	!	Shift	Green	Blue	Underline
22	34	"	"	Shift	Green	Green	Normal
23	35	#	#	Shift	Green	Cyan	Normal
24	36	$	$	Shift	Green	Red	Normal
25	37	%	%	Shift	Green	Magenta	Normal
26	38	&	&	Shift	Green	Brown	Normal
27	39	'	'		Green	Light Grey	Normal
28	40	(	(	Shift	Green	Dark Grey	High Intensity
29	41	)	)	Shift	Green	Light Blue	High Intensity Underline
2A	42	*	*	Note 1	Green	Light Green	High Intensity
2B	43	+	+	Shift	Green	Light Cyan	High Intensity
2C	44	,	,		Green	Light Red	High Intensity
2D	45	-	-		Green	Light Magenta	High Intensity
2E	46	.	.	Note 2	Green	Yellow	High Intensity

Value		As Characters			As Text Attributes		
					Color/Graphics Monitor Adapter		IBM Monochrome Display Adapter
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground	
2F	47	/	/		Green	White	High Intensity
30	48	0	0	Note 3	Cyan	Black	Normal
31	49	1	1	Note 3	Cyan	Blue	Underline
32	50	2	2	Note 3	Cyan	Green	Normal
33	51	3	3	Note 3	Cyan	Cyan	Normal
34	52	4	4	Note 3	Cyan	Red	Normal
35	53	5	5	Note 3	Cyan	Magenta	Normal
36	54	6	6	Note 3	Cyan	Brown	Normal
37	55	7	7	Note 3	Cyan	Light Grey	Normal
38	56	8	8	Note 3	Cyan	Dark Grey	High Intensity
39	57	9	9	Note 3	Cyan	Light Blue	High Intensity Underline
3A	58	:	:	Shift	Cyan	Light Green	High Intensity
3B	59	;	;		Cyan	Light Cyan	High Intensity
3C	60	<	<	Shift	Cyan	Light Red	High Intensity
3D	61	=	=		Cyan	Light Magenta	High Intensity
3E	62	>	>	Shift	Cyan	Yellow	High Intensity
3F	63	?	?	Shift	Cyan	White	High Intensity
40	64	@	@	Shift	Red	Black	Normal
41	65	A	A	Note 4	Red	Blue	Underline
42	66	B	B	Note 4	Red	Green	Normal
43	67	C	C	Note 4	Red	Cyan	Normal
44	68	D	D	Note 4	Red	Red	Normal
45	69	E	E	Note 4	Red	Magenta	Normal
46	70	F	F	Note 4	Red	Brown	Normal
47	71	G	G	Note 4	Red	Light Grey	Normal
48	72	H	H	Note 4	Red	Dark Grey	High Intensity
49	73	I	I	Note 4	Red	Light Blue	High Intensity Underline
4A	74	J	J	Note 4	Red	Light Green	High Intensity

Value		As Characters			As Text Attributes		
					Color/Graphics Monitor Adapter		IBM Monochrome Display Adapter
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground	
4B	75	K	K	Note 4	Red	Light Cyan	High Intensity
4C	76	L	L	Note 4	Red	Light Red	High Intensity
4D	77	M	M	Note 4	Red	Light Magenta	High Intensity
4E	78	N	N	Note 4	Red	Yellow	High Intensity
4F	79	O	O	Note 4	Red	White	High Intensity
50	80	P	P	Note 4	Magenta	Black	Normal
51	81	Q	Q	Note 4	Magenta	Blue	Underline
52	82	R	R	Note 4	Magenta	Green	Normal
53	83	S	S	Note 4	Magenta	Cyan	Normal
54	84	T	T	Note 4	Magenta	Red	Normal
55	85	U	U	Note 4	Magenta	Magenta	Normal
56	86	V	V	Note 4	Magenta	Brown	Normal
57	87	W	W	Note 4	Magenta	Light Grey	Normal
58	88	X	X	Note 4	Magenta	Dark Grey	High Intensity
59	89	Y	Y	Note 4	Magenta	Light Blue	High Intensity Underline
5A	90	Z	Z	Note 4	Magenta	Light Green	High Intensity
5B	91	[	[		Magenta	Light Cyan	High Intensity
5C	92	\	\		Magenta	Light Red	High Intensity
5D	93	]	]		Magenta	Light Magenta	High Intensity
5E	94	^	^	Shift	Magenta	Yellow	High Intensity
5F	95	—	—	Shift	Magenta	White	High Intensity
60	96	`	`		Brown	Black	Normal
61	97	a	a	Note 5	Brown	Blue	Underline
62	98	b	b	Note 5	Brown	Green	Normal
63	99	c	c	Note 5	Brown	Cyan	Normal
64	100	d	d	Note 5	Brown	Red	Normal
65	101	e	e	Note 5	Brown	Magenta	Normal
66	102	f	f	Note 5	Brown	Brown	Normal

Value		As Characters			As Text Attributes		
					Color/Graphics Monitor Adapter		IBM Monochrome Display Adapter
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground	
67	103	g	g	Note 5	Brown	Light Grey	Normal
68	104	h	h	Note 5	Brown	Dark Grey	High Intensity
69	105	i	i	Note 5	Brown	Light Blue	High Intensity Underline
6A	106	j	j	Note 5	Brown	Light Green	High Intensity
6B	107	k	k	Note 5	Brown	Light Cyan	High Intensity
6C	108	l	l	Note 5	Brown	Light Red	High Intensity
6D	109	m	m	Note 5	Brown	Light Magenta	High Intensity
6E	110	n	n	Note 5	Brown	Yellow	High Intensity
6F	111	o	o	Note 5	Brown	White	High Intensity
70	112	p	p	Note 5	Light Grey	Black	Reverse Video
71	113	q	q	Note 5	Light Grey	Blue	Underline
72	114	r	r	Note 5	Light Grey	Green	Normal
73	115	s	s	Note 5	Light Grey	Cyan	Normal
74	116	t	t	Note 5	Light Grey	Red	Normal
75	117	u	u	Note 5	Light Grey	Magenta	Normal
76	118	v	v	Note 5	Light Grey	Brown	Normal
77	119	w	w	Note 5	Light Grey	Light Grey	Normal
78	120	x	x	Note 5	Light Grey	Dark Grey	Reverse Video
79	121	y	y	Note 5	Light Grey	Light Blue	High Intensity Underline
7A	122	z	z	Note 5	Light Grey	Light Green	High Intensity
7B	123	{	{	Shift	Light Grey	Light Cyan	High Intensity
7C	124	|	|	Shift	Light Grey	Light Red	High Intensity
7D	125	}	}	Shift	Light Grey	Light Magenta	High Intensity
7E	126	~	~	Shift	Light Grey	Yellow	High Intensity
7F	127	△	Ctrl ←		Light Grey	White	High Intensity

Value		As Characters			As Text Attributes		
					Color/Graphics Monitor Adapter		IBM Monochrome Display Adapter
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground	
* * * * 80 to FF Hex are Flashing in both Color & IBM Monochrome * * * *							
80	128	Ç	Alt 128	Note 6	Black	Black	Non-Display
81	129	ü	Alt 129	Note 6	Black	Blue	Underline
82	130	é	Alt 130	Note 6	Black	Green	Normal
83	131	â	Alt 131	Note 6	Black	Cyan	Normal
84	132	ä	Alt 132	Note 6	Black	Red	Normal
85	133	à	Alt 133	Note 6	Black	Magenta	Normal
86	134	å	Alt 134	Note 6	Black	Brown	Normal
87	135	ç	Alt 135	Note 6	Black	Light Grey	Normal
88	136	ê	Alt 136	Note 6	Black	Dark Grey	Non-Display
89	137	ë	Alt 137	Note 6	Black	Light Blue	High Intensity Underline
8A	138	è	Alt 138	Note 6	Black	Light Green	High Intensity
8B	139	ï	Alt 139	Note 6	Black	Light Cyan	High Intensity
8C	140	î	Alt 140	Note 6	Black	Light Red	High Intensity
8D	141	ì	Alt 141	Note 6	Black	Light Magenta	High Intensity
8E	142	Ä	Alt 142	Note 6	Black	Yellow	High Intensity
8F	143	Å	Alt 143	Note 6	Black	White	High Intensity
90	144	É	Alt 144	Note 6	Blue	Black	Normal
91	145	æ	Alt 145	Note 6	Blue	Blue	Underline
92	146	Æ	Alt 146	Note 6	Blue	Green	Normal
93	147	ô	Alt 147	Note 6	Blue	Cyan	Normal
94	148	ö	Alt 148	Note 6	Blue	Red	Normal
95	149	ò	Alt 149	Note 6	Blue	Magenta	Normal
96	150	û	Alt 150	Note 6	Blue	Brown	Normal
97	151	ù	Alt 151	Note 6	Blue	Light Grey	Normal
98	152	ÿ	Alt 152	Note 6	Blue	Dark Grey	High Intensity
99	153	Ö	Alt 153	Note 6	Blue	Light Blue	High Intensity Underline
9A	154	Ü	Alt 154	Note 6	Blue	Light Green	High Intensity

Value		As Characters			As Text Attributes		
					Color/Graphics Monitor Adapter		IBM Monochrome Display Adapter
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground	
9B	155	¢	Alt 155	Note 6	Blue	Light Cyan	High Intensity
9C	156	£	Alt 156	Note 6	Blue	Light Red	High Intensity
9D	157	¥	Alt 157	Note 6	Blue	Light Magenta	High Intensity
9E	158	Pt	Alt 158	Note 6	Blue	Yellow	High Intensity
9F	159	*f*	Alt 159	Note 6	Blue	White	High Intensity
A0	160	á	Alt 160	Note 6	Green	Black	Normal
A1	161	í	Alt 161	Note 6	Green	Blue	Underline
A2	162	ó	Alt 162	Note 6	Green	Green	Normal
A3	163	ú	Alt 163	Note 6	Green	Cyan	Normal
A4	164	ñ	Alt 164	Note 6	Green	Red	Normal
A5	165	Ñ	Alt 165	Note 6	Green	Magenta	Normal
A6	166	a	Alt 166	Note 6	Green	Brown	Normal
A7	167	o	Alt 167	Note 6	Green	Light Grey	Normal
A8	168	¿	Alt 168	Note 6	Green	Dark Grey	High Intensity
A9	169	⌐	Alt 169	Note 6	Green	Light Blue	High Intensity Underline
AA	170	¬	Alt 170	Note 6	Green	Light Green	High Intensity
AB	171	½	Alt 171	Note 6	Green	Light Cyan	High Intensity
AC	172	¼	Alt 172	Note 6	Green	Light Red	High Intensity
AD	173	¡	Alt 173	Note 6	Green	Light Magenta	High Intensity
AE	174	<<	Alt 174	Note 6	Green	Yellow	High Intensity
AF	175	>>	Alt 175	Note 6	Green	White	High Intensity
B0	176	⁙	Alt 176	Note 6	Cyan	Black	Normal
B1	177	▒	Alt 177	Note 6	Cyan	Blue	Underline
B2	178	▓	Alt 178	Note 6	Cyan	Green	Normal
B3	179	│	Alt 179	Note 6	Cyan	Cyan	Normal
B4	180	┤	Alt 180	Note 6	Cyan	Red	Normal
B5	181	╡	Alt 181	Note 6	Cyan	Magenta	Normal
B6	182	╢	Alt 182	Note 6	Cyan	Brown	Normal

Value		As Characters			As Text Attributes		
					Color/Graphics Monitor Adapter		IBM Monochrome Display Adapter
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground	
B7	183		Alt 183	Note 6	Cyan	Light Grey	Normal
B8	184		Alt 184	Note 6	Cyan	Dark Grey	High Intensity
B9	185		Alt 185	Note 6	Cyan	Light Blue	High Intensity Underline
BA	186		Alt 186	Note 6	Cyan	Light Green	High Intensity
BB	187		Alt 187	Note 6	Cyan	Light Cyan	High Intensity
BC	188		Alt 188	Note 6	Cyan	Light Red	High Intensity
BD	189		Alt 189	Note 6	Cyan	Light Magenta	High Intensity
BE	190		Alt 190	Note 6	Cyan	Yellow	High Intensity
BF	191		Alt 191	Note 6	Cyan	White	High Intensity
C0	192		Alt 192	Note 6	Red	Black	Normal
C1	193		Alt 193	Note 6	Red	Blue	Underline
C2	194		Alt 194	Note 6	Red	Green	Normal
C3	195		Alt 195	Note 6	Red	Cyan	Normal
C4	196		Alt 196	Note 6	Red	Red	Normal
C5	197		Alt 197	Note 6	Red	Magenta	Normal
C6	198		Alt 198	Note 6	Red	Brown	Normal
C7	199		Alt 199	Note 6	Red	Light Grey	Normal
C8	200		Alt 200	Note 6	Red	Dark Grey	High Intensity
C9	201		Alt 201	Note 6	Red	Light Blue	High Intensity Underline
CA	202		Alt 202	Note 6	Red	Light Green	High Intensity
CB	203		Alt 203	Note 6	Red	Light Cyan	High Intensity
CC	204		Alt 204	Note 6	Red	Light Red	High Intensity
CD	205		Alt 205	Note 6	Red	Light Magenta	High Intensity
CE	206		Alt 206	Note 6	Red	Yellow	High Intensity
CF	207		Alt 207	Note 6	Red	White	High Intensity
D0	208		Alt 208	Note 6	Magenta	Black	Normal

Value		As Characters			As Text Attributes		
					Color/Graphics Monitor Adapter		IBM Monochrome Display Adapter
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground	
D1	209		Alt 209	Note 6	Magenta	Blue	Underline
D2	210		Alt 210	Note 6	Magenta	Green	Normal
D3	211		Alt 211	Note 6	Magenta	Cyan	Normal
D4	212		Alt 212	Note 6	Magenta	Red	Normal
D5	213		Alt 213	Note 6	Magenta	Magenta	Normal
D6	214		Alt 214	Note 6	Magenta	Brown	Normal
D7	215		Alt 215	Note 6	Magenta	Light Grey	Normal
D8	216		Alt 216	Note 6	Magenta	Dark Grey	High Intensity
D9	217		Alt 217	Note 6	Magenta	Light Blue	High Intensity Underline
DA	218		Alt 218	Note 6	Magenta	Light Green	High Intensity
DB	219		Alt 219	Note 6	Magenta	Light Cyan	High Intensity
DC	220		Alt 220	Note 6	Magenta	Light Red	High Intensity
DD	221		Alt 221	Note 6	Magenta	Light Magenta	High Intensity
DE	222		Alt 222	Note 6	Magenta	Yellow	High Intensity
DF	223		Alt 223	Note 6	Magenta	White	High Intensity
E0	224	α	Alt 224	Note 6	Brown	Black	Normal
E1	225	β	Alt 225	Note 6	Brown	Blue	Underline
E2	226	Γ	Alt 226	Note 6	Brown	Green	Normal
E3	227	π	Alt 227	Note 6	Brown	Cyan	Normal
E4	228	Σ	Alt 228	Note 6	Brown	Red	Normal
E5	229	σ	Alt 229	Note 6	Brown	Magenta	Normal
E6	230	μ	Alt 230	Note 6	Brown	Brown	Normal
E7	231	τ	Alt 231	Note 6	Brown	Light Grey	Normal
E8	232	Φ	Alt 232	Note 6	Brown	Dark Grey	High Intensity
E9	233	θ	Alt 233	Note 6	Brown	Light Blue	High Intensity Underline
EA	234	Ω	Alt 234	Note 6	Brown	Light Green	High Intensity
EB	235	δ	Alt 235	Note 6	Brown	Light Cyan	High Intensity

Value		As Characters			As Text Attributes		
					Color/Graphics Monitor Adapter		IBM Monochrome Display Adapter
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground	
EC	236	∞	Alt 236	Note 6	Brown	Light Red	High Intensity
ED	237	ϕ	Alt 237	Note 6	Brown	Light Magenta	High Intensity
EE	238	ϵ	Alt 238	Note 6	Brown	Yellow	High Intensity
EF	239	∩	Alt 239	Note 6	Brown	White	High Intensity
F0	240	≡	Alt 240	Note 6	Light Grey	Black	Reverse Video
F1	241	±	Alt 241	Note 6	Light Grey	Blue	Underline
F2	242	≥	Alt 242	Note 6	Light Grey	Green	Normal
F3	243	≤	Alt 243	Note 6	Light Grey	Cyan	Normal
F4	244	⌠	Alt 244	Note 6	Light Grey	Red	Normal
F5	245	⌡	Alt 245	Note 6	Light Grey	Magenta	Normal
F6	246	÷	Alt 246	Note 6	Light Grey	Brown	Normal
F7	247	≈	Alt 247	Note 6	Light Grey	Light Grey	Normal
F8	248	°	Alt 248	Note 6	Light Grey	Dark Grey	Reverse Video
F9	249	∙	Alt 249	Note 6	Light Grey	Light Blue	High Intensity Underline
FA	250	·	Alt 250	Note 6	Light Grey	Light Green	High Intensity
FB	251	√	Alt 251	Note 6	Light Grey	Light Cyan	High Intensity
FC	252	^{n}	Alt 252	Note 6	Light Grey	Light Red	High Intensity
FD	253	^{2}	Alt 253	Note 6	Light Grey	Light Magenta	High Intensity
FE	254	■	Alt 254	Note 6	Light Grey	Yellow	High Intensity
FF	255	BLANK	Alt 255	Note 6	Light Grey	White	High Intensity

Notes

1. Asterisk (*) can be typed using two methods: press the (*) key or, in the shift mode, press the 8 key.

2. Period (.) can be typed using two methods: press the . key or, in the shift or Num Lock mode, press the Del key.

3. Numeric characters 0-9 can be typed using two methods: press the numeric keys on the top row of the keyboard or, in the shift or Num Lock mode, press the numeric keys in the keypad portion of the keyboard.

4. Uppercase alphabetic characters (A-Z) can be typed in two modes: the shift mode or the Caps Lock mode.

5. Lowercase alphabetic characters (a-z) can be typed in two modes: in the normal mode or in Caps Lock and shift mode combined.

6. The three digits after the Alt key must be typed from the numeric keypad. Character codes 1-255 may be entered in this fashion (with Caps Lock activated, character codes 97-122 will display uppercase).

Quick Reference

DECIMAL VALUE	➧	0	16	32	48	64	80	96	112
⬇	HEXA-DECIMAL VALUE	0	1	2	3	4	5	6	7
0	0	BLANK (NULL)	►	BLANK (SPACE)	0	@	P	‘	p
1	1	☺	◄	!	1	A	Q	a	q
2	2	☻	↕	"	2	B	R	b	r
3	3	♥	!!	#	3	C	S	c	s
4	4	♦	¶	$	4	D	T	d	t
5	5	♣	§	%	5	E	U	e	u
6	6	♠	▬	&	6	F	V	f	v
7	7	•	↨	’	7	G	W	g	w
8	8	◘	↑	(	8	H	X	h	x
9	9	○	↓	)	9	I	Y	i	y
10	A	◙	→	*	:	J	Z	j	z
11	B	♂	←	+	;	K	[	k	{
12	C	♀	∟	,	<	L	\	l	¦
13	D	♪	↔	—	=	M	]	m	}
14	E	♫	▲	.	>	N	^	n	~
15	F	☼	▼	/	?	O	_	o	△

DECIMAL VALUE	➡	128	144	160	176	192	208	224	240
⬇	HEXA-DECIMAL VALUE	8	9	A	B	C	D	E	F
0	0	Ç	É	á	▒	└	╨	∝	≡
1	1	ü	æ	í	▒	┴	╤	β	±
2	2	é	Æ	ó	▒	┬	╥	Γ	≥
3	3	â	ô	ú	│	├	╙	π	≤
4	4	ä	ö	ñ	┤	─	╘	Σ	⌠
5	5	à	ò	Ñ	╡	┼	╒	σ	⌡
6	6	å	û	ª	╢	╞	╓	μ	÷
7	7	ç	ù	º	╖	╟	╫	τ	≈
8	8	ê	ÿ	¿	╕	╚	╪	Φ	°
9	9	ë	Ö	⌐	╣	╔	┘	Θ	•
10	A	è	Ü	¬	║	╩	┌	Ω	•
11	B	ï	¢	½	╗	╦	■	δ	√
12	C	î	£	¼	╝	╠	▄	∞	n
13	D	ì	¥	¡	╜	═	▌	φ	2
14	E	Ä	₧	«	╛	╬	▐	∈	■
15	F	Å	ƒ	»	┐	╧	▀	∩	BLANK 'FF'

Notes:

SECTION 8. IBM PERSONAL COMPUTER COMPATIBILITY

This section describes the differences among the members of the IBM Personal Computer family. It also contains information necessary to design hardware and programs that will be compatible with all members of the IBM Personal Computer family.

Hardware Considerations

To design compatible hardware or programs, you must consider hardware differences among the IBM Personal Computers. The following are hardware features of the IBM Personal Computer XT Model 286 that are not supported by all of the IBM Personal Computer family.

System Board

The IBM Personal Computer XT Model 286 system board uses an Intel 80286-6 Microprocessor. This microprocessor uses the 80287 Math Coprocessor and is generally compatible with the Intel 8088 Microprocessor used in other IBM Personal Computers.

The following table identifies the microprocessor and describes the I/O channel used with each type of IBM Personal Computer.

System Name	System Unit Microprocessor	I/O Channel Description
Personal Computer	8088	5 62-Pin
PCjr	8088	Not Compatible
Personal Computer XT	8088	8 62-Pin
Portable Personal Computer	8088	8 62-Pin
Personal Computer XT Model 286	80286-6	3 62-pin 5 98-Pin (62 Pin + 36 Pin)
Personal Computer AT	80286(-6 or -8)	2 62-pin 6 98-Pin (62 Pin + 36 Pin)

System Hardware Identification Chart

The faster processing capability of the 80286, compared to the 8088, creates special programming considerations, which are discussed later in this section under "Application Guidelines."

Adapters designed to use all 98 pins on the 98-pin connectors are not compatible with all members of the IBM Personal Computer family. Some options not supported by the IBM Personal Computer XT Model 286 are:

Expansion Unit
AT 128KB Memory Expansion
AT 512KB Memory Expansion
AT 128/640KB Memory Expansion
AT Fixed Disk And Diskette Drive Adapter
256KB Memory Expansion
64/256KB Memory Expansion
64KB Memory Module Kit
Full-high diskette drives
AT 30MB Fixed Disk Drive
AT 20MB Fixed Disk Drive
10MB Fixed Disk Drive
AT Prototype Card
Diskette Drive Adapter
Fixed Disk Adapter
8087 Math Coprocessor
Professional Graphics Adapter and Display
Game Control Adapter
Color Printer
Other keyboards

On the I/O channel:

- The system clock signal should be used only for synchronization and not for applications requiring a fixed frequency.
- The 14.31818-MHz oscillator is not synchronous with the system clock.
- The ALE signal remains high during DMA cycles.
- Pin B04 supports IRQ 9.

Fixed Disk Drive

Reading from and writing to this drive is initiated in the same way as with other IBM Personal Computers; however, the Fixed Disk and Diskette Drive Adapter may be addressed from different BIOS locations.

Diskette Drive Compatibility

The following chart shows the read, write, and format capabilities for each of the diskette drives used by IBM Personal Computers.

Diskette Drive Name	160/180K Mode	320/360K Mode	1.2M Mode	720K Mode
5-1/4 In. Diskette Drive				
Type 1	R W F	---	---	---
Type 2	R W F	R W F	---	---
Type 3	R W F	R W F	---	---
Slimline Diskette Drive	R W F	R W F	---	---
Double Sided Diskette Drive	R W F	R W F	---	---
High Capacity Diskette Drive	R W*	R W*	R W F	---
3-1/2 In.- 720K Drive	---	---	---	R W F

R=Read W=Write F=Format W*= If a diskette is formatted in either 160/180K mode or 320/360K mode and written on by a High Capacity Drive, that diskette may be read by only a High Capacity Drive.

Diskette Drive Compatibility Chart

Note: Diskettes designed for the 1.2M mode may not be used in either a 160/180K or a 320/360K diskette drive.

Copy Protection

The following methods of copy protection may not work on systems using the High Capacity Diskette Drive:

- Bypassing BIOS
- Diskette drive controls
- Write current control

Bypassing BIOS

Copy protection that tries to bypass the following BIOS routines will not work on the High Capacity Diskette Drive:

Track Density: The High Capacity Diskette Drive records tracks at a density of 96 tracks per inch (TPI). This drive has to double-step in the 48 TPI mode, which is performed by BIOS.

Data Transfer Rate: BIOS selects the proper data transfer rate for the media being used.

Disk_Base: Copy protection, which creates its own disk_base will not work on the High Capacity Diskette Drive.

Diskette Drive Controls

Copy protection that uses the following will not work on the High Capacity Diskette Drive:

Rotational Speed: The time between two events on a diskette is controlled by the Fixed Disk and Diskette Drive Adapter.

Access Time: Diskette BIOS routines must set the track-to-track access time for the different types of media used on the IBM Personal Computer XT Model 286.

Head Geometry: See "Diskette Drive Compatibility" on page 8-5.

Diskette Change Signal: Copy protection may not be able to reset this signal.

Write Current Control

Copy protection that uses write current control will not work because the Fixed Disk and Diskette Drive Adapter selects the proper write current for the media being used.

Application Guidelines

The following information should be used to develop application programs for the IBM Personal Computer family.

High-Level Language Considerations

The IBM-supported languages of BASIC, FORTRAN, COBOL, Pascal, and APL are the best choices for writing compatible programs.

If a program uses specific features of the hardware, that program may not be compatible with all IBM Personal Computers. Specifically, the use of assembler language subroutines or hardware-specific commands (In, Out, Peek, Poke, ...) must follow the assembler language rules (see "Assembler Language Programming Considerations" on page 8-8).

Any program that requires precise timing information should obtain it through a DOS or language interface; for example, TIME$ in BASIC. If greater precision is required, the assembler techniques in "Assembler Language Programming Considerations" are available. The use of programming loops may prevent a program from being compatible with other IBM Personal Computers.

Assembler Language Programming Considerations

The following OP codes work differently on systems using the 80286 microprocessor than they do on systems using the 8088 microprocessor.

- If the system microprocessor executes a POPF instruction in either the real or the virtual address mode with CPL≤IOPL, then a pending maskable interrupt (the INTR pin active) may be improperly recognized after executing the POPF instruction even if maskable interrupts were disabled before the POPF instruction and the value popped had IF=0. If the interrupt is improperly recognized, the interrupt is still correctly executed. This errata has no effect when interrupts are enabled in either real or virtual address mode. This errata has no effect in the virtual address mode when CPL>IOPL.

  The POPF instruction may be simulated with the following code macro:

```
POPFF       Macro        ; use POPFF instead of POPF
                         ; simulate popping flags
                         ; using IRET
EB 01       JMP $+3      ; jump around IRET
CF          IRET         ; POP CS, IP, flags
0E          PUSH CS
E8 FB FF    CALL $-2     ; CALL within segment
                         ; program will continue here
```

- PUSH SP

  80286 microprocessor pushes the current stack pointer.

  8088 microprocessor pushes the new stack pointer.

- Single step interrupt (when TF=1) on the interrupt instruction (OP code hex CC,CD):

  80286 microprocessor does **not** interrupt on the INT instruction.

8088 microprocessor does interrupt on the INT instruction.

- The divide error exception (interrupt 0):

  80286 microprocessor pushes the CS:IP of the instruction, causing the exception.

  8088 microprocessor pushes the CS:IP **following** the instruction, causing the exception.

- Shift counts are masked to five bits. Shift counts greater than 31 are treated mod 32. For example, a shift count of 36, shifts the operand four places.

The following describes anomalies which may occur in systems which contain 80286 processors with 1983 and 1984 date codes (S40172, S54036, S40093, S54012).

In protected mode, the contents of the CX register may be unexpectedly altered under the following conditions:

Note: The value in parenthesis indicates the type of error code pushed onto the exception handler's stack.

Exception #NP() = Exception #11 = Not-present Fault
Exception #SS() = Exception #12 = Stack Fault
Exception #GP() = Exception #13 = General Protection Fault

- Exception #GP(0) from attempted access to data segment or extra segment when the corresponding segment register holds a null selector.

- Exception #GP(0) from attempted data read from code segment when code segment has the "execute only" attribute.

- Exception #GP(0) from attempted write to code segment (code segments are not writable in protected mode), or to data segment of extra segment if the data or extra segment has the read only attribute.

- Exception #GP(0) from attempted load of a selector referencing the local descriptor table into CS, DS, ES or SS, when the LDT is not present.

- Exception #GP(0) from attempted input or output instruction when CPL > IOPL.

- Exception #GP(selector) from attempted access to a descriptor is GDT, LDT, or IDT, beyond the defined limit of the descriptor table.

- Exception #GP(0) from attempted read or write (except for "PUSH" onto stack) beyond the defined limit of segment.

- Exception #SS(0) from attempted "PUSH" below the defined limit of the stack segment.

Restarting applications which generate the above exceptions may result in errors.

In the protected mode, when any of the null selector values (0000H, 0001H, 0002H, 0003H) are loaded into the DS or ES registers via a MOV or POP instruction or a task switch, the 80286 always loads the null selector 0000H into the corresponding register.

If a coprocessor (80287) operand is read from an "executable and readable" and conforming (ERC) code segment, and the coprocessor operand is sufficiently near the segment's limit that the second or subsequent byte lies outside the limit, no protection exception #9 will be generated.

The following correctly describes the operation of all 80286 parts:

- Instructions longer than 10 bytes (instructions using multiple redundant prefixes) generate exception #13 (General Purpose Exception) in both the real and protected modes.

- If the second operand of an ARPL instruction is a null selector, the instruction generates an exception #13.

Assembler language programs should perform all I/O operations through ROM BIOS or DOS function calls.

- Program interrupts are used for access to these functions. This practice removes the absolute addressing from the program. Only the interrupt number is required.

- The coprocessor detects six different exception conditions that can occur during instruction execution. If the appropriate exception mask within the coprocessor is not set, the coprocessor sets its error signal. This error signal generates a hardware interrupt (interrupt 13) and causes the 'busy' signal to the coprocessor to be held in the busy state. The 'busy' signal may be cleared by an 8-bit I/O Write command to address hex F0 with D0 through D7 equal to 0.

  The power-on-self-test code in the system ROM enables hardware IRQ 13 and sets up its vector to point to a routine in ROM. The ROM routine clears the 'busy' signal latch and then transfers control to the address pointed to by the NMI interrupt vector. This allows code written for any IBM Personal Computer to work on an IBM Personal Computer XT Model 286. The NMI interrupt handler should read the coprocessor's status to determine if the NMI was caused by the coprocessor. If the interrupt was not generated by the coprocessor, control should be passed to the original NMI interrupt handler.

- Back to back I/O commands to the same I/O ports will not permit enough recovery time for I/O chips. To ensure enough time, a JMP SHORT $+2 must be inserted between IN/OUT instructions to the same I/O chip.

  Note: MOV AL,AH type instruction does not allow enough recovery time. An example of the correct procedure follows:

```
OUT  IO_ADD,AL
JMP  SHORT $+2
MOV  AL,AH
OUT  IO_ADD,AL
```

- In systems using the 80286 microprocessor, IRQ 9 is redirected to INT hex 0A (hardware IRQ 2). This insures that hardware designed to use IRQ 2 will operate in the IBM Personal Computer XT Model 286.

- The system can mask hardware sensitivity. New devices can change the ROM BIOS to accept the same programming interface on the new device.

- In cases where BIOS provides parameter tables, such as for video or diskette, a program may substitute new parameter values by building a new copy of the table and changing the vector to point to that table. However, the program should copy the current table, using the current vector, and then modify those locations in the table that need to be changed. In this way, the program will not inadvertently change any values that should be left the same.

- Disk_Base consists of 11 parameters required for diskette operation. They are pointed at by the data variable, Disk_Pointer, at absolute address 0:78. It is strongly recommended that the values supplied in ROM be used. If it becomes necessary to modify any of the parameters, build another parameter block and modify the address in Disk_Pointer to point to the new block.

  The parameters were established to operate both the High Capacity Diskette Drive and the Double Sided Diskette Drive. Three of the parameters in this table are under control of BIOS in the following situations.

  The Gap Length Parameter is no longer retrieved from the parameter block.

  The gap length used during diskette read, write, and verify operations is derived from within diskette BIOS.

  The gap length for format operations is still obtained from the parameter block.

  Special considerations are required for formatting operations. See the prolog of Diskette BIOS for the required details. If a parameter block contains a head settle time parameter value of 0 milliseconds, and a write operation is being performed, at least 15 milliseconds of head settle time will be enforced for a High Capacity Diskette Drive and 20 milliseconds will be enforced for a Double Sided Diskette Drive. If a parameter block contains a motor start wait parameter of less than 1 second for a write or format operation of 625 milliseconds for a read or verify operation, Diskette BIOS will enforce those times listed above.

- The following procedure is used to determine the type of media inserted in the High Capacity Diskette Drive:

  1. Read Track 0, Head 0, Sector 1 to allow diskette BIOS to establish the media/drive combination. If this is successful, continue with the next step.

  2. Read Track 0, Sector 15. If an error occurs, a double sided diskette is in the drive.

     Note: Refer to the *DOS Technical Reference* manual for the File Allocation Table (FAT) parameters for single- and double-sided diskettes.
     If a successful read occurs, a high capacity diskette is in the drive.

  3. If Step 1 fails, issue the reset function (AH=0) to diskette BIOS and retry. If a successful read cannot be done, the media needs to be formatted or is defective.

ROM BIOS and DOS do not provide for all functions. The following are the allowable I/O operations with which IBM will maintain compatibility in future systems.

- Control of the sound, using port hex 61, and the sound channel of the timer/counter. A program can control timer/counter channels 0 and 2, ports hex 40, 42, and 43. A program must not change the value in port hex 41, because this port controls the dynamic-memory refresh. Channel 0 provides the time-of-day interrupt, and can also be used for timing short intervals. Channel 2 of the timer/counter is the output for the speaker and cassette ports. This channel may also be used for timing short intervals, although it cannot interrupt at the end of the period.

  Note: Programs should use the timer for delay on the paddle input rather than a program loop.

- Interrupt Mask Register (IMR), port hex 21, can be used to selectively mask and unmask the hardware features.

The following information pertains to absolute memory locations.

- Interrupt Vectors Segment (hex 0)--A program may change these to point at different processing routines. When an interrupt vector is modified, the original value should be retained. If the interrupt, either hardware or program, is not directed toward this device handler, the request should be passed to the next item in the list.

- Video Display Buffers (hex B0000 and B8000)-- For each mode of operation defined in the video display BIOS, the memory map will remain the same. For example, the bit map for the 320 x 200 medium-resolution graphics mode of the Color/Graphics Monitor adapter will be retained on any future adapter that supports that mode. If the bit map is modified, a different mode number will be used.

- ROM BIOS Data Area (hex 40:0)--Any variables in this area will retain their current definition, whenever it is reasonable to do so. IBM may use these data areas for other purposes when the variable no longer has meaning in the system. In general, ROM BIOS data variables should be read or modified through BIOS calls whenever possible, and not with direct access to the variable.

A program that requires timing information should use either the time-of-day clock or the timing channels of the timer/counter. The input frequency to the timer will be maintained at 1.19 MHz, providing a constant time reference. Program loops should be avoided.

Programs that use copy protection schemes should use the ROM BIOS diskette calls to read and verify the diskette and should not be timer dependent. Any method can be used to create the diskette, although manufacturing capability should be considered. The verifying program can look at the diskette controller's status bytes in the ROM BIOS data area for additional information about embedded errors. More information about copy protection may be found on page 8-5 under "Copy Protection".

Any DOS program must be relocatable and insensitive to the size of DOS or its own load addresses. A program's memory requirement should be identified and contiguous with the load module. A program should not assume that all of memory is available to it.

There are several 80286 instructions that, when executed, lock out external bus signals. DMA requests are not honored during the execution of these instructions. Consecutive instructions of this type prevent DMA activity from the start of the first instruction to the end of the last instruction. To allow for necessary DMA cycles, as required by the diskette controller in a multitasking system, multiple lock-out instructions must be separated by JMP SHORT $+2.

Multitasking Provisions

The IBM Personal Computer XT Model 286 BIOS contains a feature to assist multitasking implementation. "Hooks" are provided for a multitasking dispatcher. Whenever a busy (wait) loop occurs in the BIOS, a hook is provided for the program to break out of the loop. Also, whenever BIOS services an interrupt, a corresponding wait loop is exited, and another hook is provided. Thus a program may be written that employs the bulk of the device driver code. The following is valid only in the microprocessor's real address mode and must be taken by the code to allow this support.

The program is responsible for the serialization of access to the device driver. The BIOS code is not reentrant.

The program is responsible for matching corresponding wait and post calls.

Interfaces

There are four interfaces to be used by the multitasking dispatcher:

Startup

First, the startup code hooks interrupt hex 15. The dispatcher is responsible to check for function codes of AH= hex 90 or 91. The "Wait" and "Post" sections describe these codes. The dispatcher must pass all other functions to the previous user of interrupt hex 15. This can be done by a JMP or a CALL. If the function code is hex 90 or 91, the dispatcher should do the appropriate processing and return by the IRET instruction.

Serialization

It is up to the multitasking system to ensure that the device driver code is used serially. Multiple entries into the code can result in serious errors.

Wait (Busy)

Whenever the BIOS is about to enter a busy loop, it first issues an interrupt hex 15 with a function code of hex 90 in AH. This signals a wait condition. At this point, the dispatcher should save the task status and dispatch another task. This allows overlapped execution of tasks when the hardware is busy. The following is an outline of the code that has been added to the BIOS to perform this function.

```
MOV AX, 90XXH          ; wait code in AH and
                       ; type code in AL
INT 15H                ; issue call
JC  TIMEOUT            ; optional: for time-out or
                       ; if carry is set, time-out
                       ; occurred
NORMAL TIMEOUT LOGIC ; normal time-out
```

Post (Interrupt)

Whenever the BIOS has set an interrupt flag for a corresponding busy loop, an interrupt 15 occurs with a function code of hex 91 in AH. This signals a post condition. At this point, the dispatcher should set the task status to "ready to run" and return to the interrupt routine. The following is an outline of the code added to BIOS that performs this function.

```
MOV AX, 91XXH          ; post code AH and
                       ; type code AL
INT 15H                ; issue call
```


Classes

The following types of wait loops are supported:

- The class for hex 0 to 7F is serially reusable. This means that for the devices that use these codes, access to the BIOS must be restricted to only one task at a time.
- The class for hex 80 to BF is reentrant. There is no restriction on the number of tasks that may access the device.
- The class for hex C0 to FF is non-interrupt. There is no corresponding interrupt for the wait loop. Therefore, it is the responsibility of the dispatcher to determine what satisfies this condition to exit the loop.

Function Code Classes

Type Code (AL)	Description
00H->7FH	Serially reusable devices; operating system must serialize access
80H->0BFH	Reentrant devices; ES:BX is used to distinguish different calls (multiple I/O calls are allowed simultaneously)

0C0H->0FH	Wait only calls; there is no complementary POST for these waits--these are time-out only. Times are function-number dependent.

Function Code Assignments

The following are specific assignments for the IBM Personal Computer XT Model 286 BIOS. Times are approximate. They are grouped according to the classes described under "Function Code Classes".

Type Code (AL)	Time-out	Description
00H	yes (6 second)	fixed disk
01H	yes (2 second)	diskette
02H	no (2 second)	keyboard
0FDH	yes (1 second-write)	diskette motor start
--	(625 ms-read)	--
0FEH	yes (18 second)	printer

The asynchronous support has been omitted. The Serial/Parallel Adapter will generate interrupts, but BIOS does not support it in the interrupt mode. Therefore, the support should be included in the multitasking system code if that device is to be supported.

Time-Outs

To support time-outs properly, the multitasking dispatcher must be aware of time. If a device enters a busy loop, it generally should remain there for a specific amount of time before indicating an error. The dispatcher should return to the BIOS wait loop with the carry bit set if a time-out occurs.

Machine-Sensitive Code

Programs may select machine specific features, but they must test for specific machine type. Location of the specific machine identification codes can be found through interrupt 15 function code AH (See 'Configuration Parameters' in BIOS Listing). The code is two bytes. The first byte shows the machine type and the second byte shows the series type. They are as follows:

First Byte	Second Byte	Machine Identification
FF	00	IBM Personal Computer
FE	00	IBM Personal Computer XT
FE	00	IBM Portable Personal Computer
FD	00	IBM PCjr
FC	00	IBM Personal Computer AT
FC	02	IBM Personal Computer XT Model 286
FB	00	IBM Personal Computer XT with 256/640 system board

Machine Identification Code

IBM will define methods for uniquely determining the specific machine type or I/O feature for any new device.

Notes:

Glossary

This glossary includes terms and definitions from the *IBM Vocabulary for Data Processing, Telecommunications, and Office Systems,* GC20-1699.

μ. Prefix micro; 0.000 001.

μs. Microsecond; 0.000 001 second.

A. Ampere.

ac. Alternating current.

accumulator. A register in which the result of an operation is formed.

active high. Designates a signal that has to go high to produce an effect. Synonymous with positive true.

active low. Designates a signal that has to go low to produce an effect. Synonymous with negative true.

adapter. An auxiliary device or unit used to extend the operation of another system.

address bus. One or more conductors used to carry the binary-coded address from the processor throughout the rest of the system.

algorithm. A finite set of well-defined rules for the solution of a problem in a finite number of steps.

all points addressable (APA). A mode in which all points of a displayable image can be controlled by the user.

alphameric. Synonym for alphanumeric.

alphanumeric (A/N). Pertaining to a character set that contains letters, digits, and usually other characters, such as punctuation marks. Synonymous with alphameric.

alternating current (ac). A current that periodically reverses its direction of flow.

American National Standard Code for Information Interchange (ASCII). The standard code, using a coded character set consisting of 7-bit coded characters (8 bits including parity check), used for information exchange between data processing systems, data communication systems, and associated equipment. The ASCII set consists of control characters and graphic characters.

ampere (A). The basic unit of electric current.

A/N. Alphanumeric

analog. (1) Pertaining to data in the form of continuously variable physical quantities. (2) Contrast with digital.

AND. A logic operator having the property that if P is a statement, Q is a statement, R is a statement,..., then the AND of P, Q, R,...is true if all statements are true, false if any statement is false.

AND gate. A logic gate in which the output is 1 only if all inputs are 1.

AND operation. The boolean operation whose result has the boolean value 1, if and only if, each operand has the boolean value 1. Synonymous with conjunction.

APA. All points addressable.

ASCII. American National Standard Code for Information Interchange.

assemble. To translate a program expressed in an assembler language into a computer language.

assembler. A computer program used to assemble.

assembler language. A computer-oriented language whose instructions are usually in one-to-one correspondence with computer instructions.

asynchronous transmission. (1) Transmission in which the time of occurrence of the start of each character, or block of characters, is arbitrary; once started, the time of occurrence of each signal representing a bit within a character, or block, has the same relationship to significant instants of a fixed time frame. (2) Transmission in which each information character is individually transmitted (usually timed by the use of start elements and stop elements).

audio frequencies. Frequencies that can be heard by the human ear (approximately 15 hertz to 20,000 hertz).

auxiliary storage. (1) A storage device that is not main storage. (2) Data storage other than main storage; for example, storage on magnetic disk. (3) Contrast with main storage.

BASIC. Beginner's all-purpose symbolic instruction code.

basic input/output system (BIOS). The feature of the IBM Personal Computer that provides the level control of the major I/O devices, and relieves the programmer from concern about hardware device characteristics.

baud. (1) A unit of signaling speed equal to the number of discrete conditions or signal events per second. For example, one baud equals one bit per second in a train of binary signals, one-half dot cycle per second in Morse code, and one 3-bit value per second in a train of signals each of which can assume one of eight different states. (2) In asynchronous transmission, the unit of modulation rate corresponding to one unit of interval per second; that is, if the duration of the unit interval is 20 milliseconds, the modulation rate is 50 baud.

BCC. Block-check character.

beginner's all-purpose symbolic instruction code (BASIC). A programming language with a small repertoire of commands and a simple syntax, primarily designed for numeric applications.

binary. (1) Pertaining to a selection, choice, or condition that has two possible values or states. (2) Pertaining to a fixed radix numeration system having a radix of 2.

binary digit. (1) In binary notation, either of the characters 0 or 1. (2) Synonymous with bit.

binary notation. Any notation that uses two different characters, usually the binary digits 0 and 1.

binary synchronous communications (BSC). A uniform procedure, using a standardized set of control characters and control character sequences for synchronous transmission of binary–coded data between stations.

BIOS. Basic input/output system.

bit. Synonym for binary digit

bits per second (bps). A unit of measurement representing the number of discrete binary digits transmitted by a device in one second.

block. (1) A string of records, a string of words, or a character string formed for technical or logic reasons to be treated as an entity. (2) A set of things, such as words, characters, or digits, treated as a unit.

block-check character (BCC). In cyclic redundancy checking, a character that is transmitted by the sender after each message block and is compared with a block-check character computed by the receiver to determine if the transmission was successful.

boolean operation. (1) Any operation in which each of the operands and the result take one of two values. (2) An operation that follows the rules of boolean algebra.

bootstrap. A technique or device designed to bring itself into a desired state by means of its own action; for example, a machine routine whose first few instructions are sufficient to bring the rest of itself into the computer from an input device.

bps. Bits per second.

BSC. Binary synchronous communications.

buffer. (1) An area of storage that is temporarily reserved for use in performing an input/output operation, into which data is read or from which data is written. Synonymous with I/O area. (2) A portion of storage for temporarily holding input or output data.

bus. One or more conductors used for transmitting signals or power.

byte. (1) A sequence of eight adjacent binary digits that are operated upon as a unit. (2) A binary character operated upon as a unit. (3) The representation of a character.

C. Celsius.

capacitor. An electronic circuit component that stores an electric charge.

Cartesian coordinates. A system of coordinates for locating a point on a plane by its distance from each of two intersecting lines, or in space by its distance from each of three mutually perpendicular planes.

CAS. Column address strobe.

cathode ray tube (CRT). A vacuum tube in which a stream of electrons is projected onto a fluorescent screen producing a luminous spot. The location of the spot can be controlled.

cathode ray tube display (CRT display). (1) A CRT used for displaying data. For example, the electron beam can be controlled to form alphanumeric data by use of a dot matrix.
(2) Synonymous with monitor.

CCITT. International Telegraph and Telephone Consultative Committee.

Celsius (C). A temperature scale. Contrast with Fahrenheit (F).

central processing unit (CPU). Term for processing unit.

channel. A path along which signals can be sent; for example, data channel, output channel.

character generator. (1) In computer graphics, a functional unit that converts the coded representation of a graphic character into the shape of the character for display. (2) In word processing, the means within equipment for generating visual characters or symbols from coded data.

character set. (1) A finite set of different characters upon which agreement has been reached and that is considered complete for some purpose. (2) A set of unique representations called characters. (3) A defined collection of characters.

characters per second (cps). A standard unit of measurement for the speed at which a printer prints.

check key. A group of characters, derived from and appended to a data item, that can be used to detect errors in the data item during processing.

clipping. In computer graphics, removing parts of a display image that lie outside a window.

closed circuit. A continuous unbroken circuit; that is, one in which current can flow. Contrast with open circuit.

CMOS. Complementary metal oxide semiconductor.

code. (1) A set of unambiguous rules specifying the manner in which data may be represented in a discrete form. Synonymous with coding scheme. (2) A set of items, such as abbreviations, representing the members of another set. (3) To represent data or a computer program in a symbolic form that can be accepted by a data processor. (4) Loosely, one or more computer programs, or part of a computer program.

coding scheme. Synonym for code.

collector. An element in a transistor toward which current flows.

color cone. An arrangement of the visible colors on the surface of a double-ended cone where lightness varies along the axis of

the cone, and hue varies around the circumference. Lightness includes both the intensity and saturation of color.

column address strobe (CAS). A signal that latches the column addresses in a memory chip.

compile. (1) To translate a computer program expressed in a problem-oriented language into a computer-oriented language. (2) To prepare a machine-language program from a computer program written in another programming language by making use of the overall logic structure of the program, or generating more than one computer instruction for each symbolic statement, or both, as well as performing the function of an assembler.

complement. A number that can be derived from a specified number by subtracting it from a second specified number.

complementary metal oxide semiconductor (CMOS). A logic circuit family that uses very little power. It works with a wide range of power supply voltages.

computer. A functional unit that can perform substantial computation, including numerous arithmetic operations or logic operations, without human intervention during a run.

computer instruction code. A code used to represent the instructions in an instruction set. Synonymous with machine code.

computer program. A sequence of instructions suitable for processing by a computer.

computer word. A word stored in one computer location and capable of being treated as a unit.

configuration. (1) The arrangement of a computer system or network as defined by the nature, number, and the chief characteristics of its functional units. More specifically, the term configuration may refer to a hardware configuration or a software configuration. (2) The devices and programs that make up a system, subsystem, or network.

conjunction. Synonym for AND operation.

contiguous. Touching or joining at the edge or boundary; adjacent.

control character. A character whose occurrence in a particular context initiates, modifies, or stops a control operation.

control operation. An action that affects the recording, processing, transmission, or interpretation of data; for example, starting or stopping a process, carriage return, font change, rewind, and end of transmission.

control storage. A portion of storage that contains microcode.

coordinate space. In computer graphics, a system of Cartesian coordinates in which an object is defined.

cps. Characters per second.

CPU. Central processing unit.

CRC. Cyclic redundancy check.

CRT. Cathode ray tube.

CRT display. Cathode ray tube display.

CTS. Clear to send. Associated with modem control.

cursor. (1) In computer graphics, a movable marker that is used to indicate position on a display. (2) A displayed symbol that acts as a marker to help the user locate a point in text, in a system command, or in storage. (3) A movable spot of light on the screen of a display device, usually indicating where the next character is to be entered, replaced, or deleted.

cyclic redundancy check (CRC). (1) A redundancy check in which the check key is generated by a cyclic algorithm. (2) A system of error checking performed at both the sending and receiving station after a block-check character has been accumulated.

cylinder. (1) The set of all tracks with the same nominal distance from the axis about which the disk rotates. (2) The

tracks of a disk storage device that can be accessed without repositioning the access mechanism.

daisy-chained cable. A type of cable that has two or more connectors attached in series.

data. (1) A representation of facts, concepts, or instructions in a formalized manner suitable for communication, interpretation, or processing by human or automatic means. (2) Any representations, such as characters or analog quantities, to which meaning is, or might be assigned.

data base. A collection of data that can be immediately accessed and operated upon by a data processing system for a specific purpose.

data processing system. A system that performs input, processing, storage, output, and control functions to accomplish a sequence of operations on data.

data transmission. Synonym for transmission.

dB. Decibel.

dBa. Adjusted decibels.

dc. Direct current.

debounce. (1) An electronic means of overcoming the make/break bounce of switches to obtain one smooth change of signal level. (2) The elimination of undesired signal variations caused by mechanically generated signals from contacts.

decibel. (1) A unit that expresses the ratio of two power levels on a logarithmic scale. (2) A unit for measuring relative power.

decoupling capacitor. A capacitor that provides a low impedance path to ground to prevent common coupling between circuits.

Deutsche Industrie Norm (DIN). (1) German Industrial Norm. (2) The committee that sets German dimension standards.

digit. (1) A graphic character that represents an integer; for example, one of the characters 0 to 9. (2) A symbol that

represents one of the non-negative integers smaller than the radix. For example, in decimal notation, a digit is one of the characters 0 to 9.

digital. (1) Pertaining to data in the form of digits. (2) Contrast with analog.

DIN. Deutsche Industrie Norm.

DIN connector. One of the connectors specified by the DIN committee.

DIP. Dual in-line package.

DIP switch. One of a set of small switches mounted in a dual in-line package.

direct current (dc). A current that always flows in one direction.

direct memory access (DMA). A method of transferring data between main storage and I/O devices that does not require processor intervention.

disable. To stop the operation of a circuit or device.

disabled. Pertaining to a state of a processing unit that prevents the occurrence of certain types of interruptions. Synonymous with masked.

disk. Loosely, a magnetic disk.

diskette. A thin, flexible magnetic disk and a semirigid protective jacket, in which the disk is permanently enclosed. Synonymous with flexible disk.

diskette drive. A device for storing data on and retrieving data from a diskette.

display. (1) A visual presentation of data. (2) A device for visual presentation of information on any temporary character imaging device. (3) To present data visually. (4) See cathode ray tube display.

display attribute. In computer graphics, a particular property that is assigned to all or part of a display; for example, low intensity, green color, blinking status.

display element. In computer graphics, a basic graphic element that can be used to construct a display image; for example, a dot, a line segment, a character.

display group. In computer graphics, a collection of display elements that can be manipulated as a unit and that can be further combined to form larger groups.

display image. In computer graphics, a collection of display elements or display groups that are represented together at any one time in a display space.

display space. In computer graphics, that portion of a display surface available for a display image. The display space may be all or part of a display surface.

display surface. In computer graphics, that medium on which display images may appear; for example, the entire screen of a cathode ray tube.

DMA. Direct memory access.

dot matrix. (1) In computer graphics, a two-dimensional pattern of dots used for constructing a display image. This type of matrix can be used to represent characters by dots. (2) In word processing, a pattern of dots used to form characters. This term normally refers to a small section of a set of addressable points; for example, a representation of characters by dots.

dot printer. Synonym for matrix printer.

dot-matrix character generator. In computer graphics, a character generator that generates character images composed of dots.

drawing primitive. A group of commands that draw defined geometric shapes.

DSR. Data set ready. Associated with modem control.

DTR. In the IBM Personal Computer, data terminal ready. Associated with modem control.

dual in-line package (DIP). A widely used container for an integrated circuit. DIPs have pins in two parallel rows. The pins are spaced 1/10 inch apart. See also DIP switch.

duplex. (1) In data communication, pertaining to a simultaneous two-way independent transmission in both directions.
(2) Contrast with half-duplex.

duty cycle. In the operation of a device, the ratio of on time to idle time. Duty cycle is expressed as a decimal or percentage.

dynamic memory. RAM using transistors and capacitors as the memory elements. This memory requires a refresh (recharge) cycle every few milliseconds. Contrast with static memory.

EBCDIC. Extended binary-coded decimal interchange code.

ECC. Error checking and correction.

edge connector. A terminal block with a number of contacts attached to the edge of a printed-circuit board to facilitate plugging into a foundation circuit.

EIA. Electronic Industries Association.

electromagnet. Any device that exhibits magnetism only while an electric current flows through it.

enable. To initiate the operation of a circuit or device.

end of block (EOB). A code that marks the end of a block of data.

end of file (EOF). An internal label, immediately following the last record of a file, signaling the end of that file. It may include control totals for comparison with counts accumulated during processing.

end-of-text (ETX). A transmission control character used to terminate text.

end-of-transmission (EOT). A transmission control character used to indicate the conclusion of a transmission, which may have included one or more texts and any associated message headings.

end-of-transmission-block (ETB). A transmission control character used to indicate the end of a transmission block of data when data is divided into such blocks for transmission purposes.

EOB. End of block.

EOF. End of file.

EOT. End-of-transmission.

EPROM. Erasable programmable read-only memory.

erasable programmable read-only memory (EPROM). A PROM in which the user can erase old information and enter new information.

error checking and correction (ECC). The detection and correction of all single-bit errors, plus the detection of double-bit and some multiple-bit errors.

ESC. The escape character.

escape character (ESC). A code extension character used, in some cases, with one or more succeeding characters to indicate by some convention or agreement that the coded representations following the character or the group of characters are to be interpreted according to a different code or according to a different coded character set.

ETB. End-of-transmission-block.

ETX. End-of-text.

extended binary-coded decimal interchange code (EBCDIC). A set of 256 characters, each represented by eight bits.

F. Fahrenheit.

Fahrenheit (F). A temperature scale. Contrast with Celsius (C).

falling edge. Synonym for negative-going edge.

FCC. Federal Communications Commission.

fetch. To locate and load a quantity of data from storage.

FF. The form feed character.

field. (1) In a record, a specified area used for a particular category of data. (2) In a data base, the smallest unit of data that can be referred to.

field-programmable logic sequencer (FPLS). An integrated circuit containing a programmable, read-only memory that responds to external inputs and feedback of its own outputs.

FIFO (first-in-first out). A queuing technique in which the next item to be retrieved is the item that has been in the queue for the longest time.

fixed disk drive. In the IBM Personal Computer, a unit consisting of nonremovable magnetic disks, and a device for storing data on and retrieving data from the disks.

flag. (1) Any of various types of indicators used for identification. (2) A character that signals the occurrence of some condition, such as the end of a word. (3) Deprecated term for mark.

flexible disk. Synonym for diskette.

flip-flop. A circuit or device containing active elements, capable of assuming either one of two stable states at a given time.

font. A family or assortment of characters of a given size and style; for example, 10 point Press Roman medium.

foreground. (1) In multiprogramming, the environment in which high-priority programs are executed. (2) On a color display screen, the characters as opposed to the background.

form feed. (1) Paper movement used to bring an assigned part of a form to the printing position. (2) In word processing, a

function that advances the typing position to the same character position on a predetermined line of the next form or page.

form feed character. A control character that causes the print or display position to move to the next predetermined first line on the next form, the next page, or the equivalent.

format. The arrangement or layout of data on a data medium.

FPLS. Field-programmable logic sequencer.

frame. (1) In SDLC, the vehicle for every command, every response, and all information that is transmitted using SDLC procedures. Each frame begins and ends with a flag. (2) In data transmission, the sequence of contiguous bits bracketed by and including beginning and ending flag sequences.

g. Gram.

G. (1) Prefix giga; 1,000,000,000. (2) When referring to computer storage capacity, 1,073,741,824. (1,073,741,824 = 2 to the 30th power.)

gate. (1) A combinational logic circuit having one output channel and one or more input channels, such that the output channel state is completely determined by the input channel states. (2) A signal that enables the passage of other signals through a circuit.

Gb. 1,073,741,824 bytes.

general-purpose register. A register, usually explicitly addressable within a set of registers, that can be used for different purposes; for example, as an accumulator, as an index register, or as a special handler of data.

giga (G). Prefix 1,000,000,000.

gram (g). A unit of weight (equivalent to 0.035 ounces).

graphic. A symbol produced by a process such as handwriting, drawing, or printing.

graphic character. A character, other than a control character, that is normally represented by a graphic.

half-duplex. (1) In data communication, pertaining to an alternate, one way at a time, independent transmission. (2) Contrast with duplex.

hardware. (1) Physical equipment used in data processing, as opposed to programs, procedures, rules, and associated documentation. (2) Contrast with software.

head. A device that reads, writes, or erases data on a storage medium; for example, a small electromagnet used to read, write, or erase data on a magnetic disk.

hertz (Hz). A unit of frequency equal to one cycle per second.

hex. Common abbreviation for hexadecimal. Also, hexadecimal can be noted as X' '.

hexadecimal. (1) Pertaining to a selection, choice, or condition that has 16 possible different values or states. These values or states are usually symbolized by the ten digits 0 through 9 and the six letters A through F. (2) Pertaining to a fixed radix numeration system having a radix of 16.

high impedance state. A state in which the output of a device is effectively isolated from the circuit.

highlighting. In computer graphics, emphasizing a given display group by changing its attributes relative to other display groups in the same display field.

high-order position. The leftmost position in a string of characters. See also most-significant digit.

hither plane. In computer graphics, a plane that is perpendicular to the line joining the viewing reference point and the view point and that lies between these two points. Any part of an object between the hither plane and the view point is not seen. See also yon plane.

housekeeping. Operations or routines that do not contribute directly to the solution of the problem but do contribute directly to the operation of the computer.

Hz. Hertz

image. A fully processed unit of operational data that is ready to be transmitted to a remote unit; when loaded into control storage in the remote unit, the image determines the operations of the unit.

immediate instruction. An instruction that contains within itself an operand for the operation specified, rather than an address of the operand.

index register. A register whose contents may be used to modify an operand address during the execution of computer instructions.

indicator. (1) A device that may be set into a prescribed state, usually according to the result of a previous process or on the occurrence of a specified condition in the equipment, and that usually gives a visual or other indication of the existence of the prescribed state, and that may in some cases be used to determine the selection among alternative processes; for example, an overflow indicator. (2) An item of data that may be interrogated to determine whether a particular condition has been satisfied in the execution of a computer program; for example, a switch indicator, an overflow indicator.

inhibited. (1) Pertaining to a state of a processing unit in which certain types of interruptions are not allowed to occur. (2) Pertaining to the state in which a transmission control unit or an audio response unit cannot accept incoming calls on a line.

initialize. To set counters, switches, addresses, or contents of storage to 0 or other starting values at the beginning of, or at prescribed points in, the operation of a computer routine.

input/output (I/O). (1) Pertaining to a device or to a channel that may be involved in an input process, and, at a different time, in an output process. In the English language, "input/output" may be used in place of such terms as "input/output data," "input/output signal," and "input/output terminals," when such usage is clear in a given context. (2) Pertaining to a device

whose parts can be performing an input process and an output process at the same time. (3) Pertaining to either input or output, or both.

instruction. In a programming language, a meaningful expression that specifies one operation and identifies its operands, if any.

instruction set. The set of instructions of a computer, of a programming language, or of the programming languages in a programming system.

intensity. In computer graphics, the amount of light emitted at a display point

interface. A device that alters or converts actual electrical signals between distinct devices, programs, or systems.

interleave. To arrange parts of one sequence of things or events so that they alternate with parts of one or more other sequences of the same nature and so that each sequence retains its identity.

interrupt. (1) A suspension of a process, such as the execution of a computer program, caused by an event external to that process, and performed in such a way that the process can be resumed. (2) In a data transmission, to take an action at a receiving station that causes the transmitting station to terminate a transmission. (3) Synonymous with interruption.

I/O. Input/output.

I/O area. Synonym for buffer.

irrecoverable error. An error that makes recovery impossible without the use of recovery techniques external to the computer program or run.

joystick. In computer graphics, a lever that can pivot in all directions and that is used as a locator device.

k. Prefix kilo; 1000.

K. When referring to storage capacity, 1024. (1024 = 2 to the 10th power.)

KB. 1024 bytes.

key lock. A device that deactivates the keyboard and locks the cover on for security.

kg. Kilogram; 1000 grams.

kHz. Kilohertz; 1000 hertz.

kilo (k). Prefix 1000

kilogram (kg). 1000 grams.

kilohertz (kHz). 1000 hertz

latch. (1) A simple logic-circuit storage element. (2) A feedback loop in sequential digital circuits used to maintain a state.

least-significant digit. The rightmost digit. See also low-order position.

LED. Light-emitting diode.

light-emitting diode (LED). A semiconductor device that gives off visible or infrared light when activated.

load. In programming, to enter data into storage or working registers.

look-up table (LUT). (1) A technique for mapping one set of values into a larger set of values. (2) In computer graphics, a table that assigns a color value (red, green, blue intensities) to a color index.

low power Schottky TTL. A version (LS series) of TTL giving a good compromise between low power and high speed. See also transistor-transistor logic and Schottky TTL.

low-order position. The rightmost position in a string of characters. See also least-significant digit.

luminance. The luminous intensity per unit projected area of a given surface viewed from a given direction.

LUT. Look-up table.

m. (1) Prefix milli; 0.001. (2) Meter.

M. (1) Prefix mega; 1,000,000. (2) When referring to computer storage capacity, 1,048,576. (1,048,576 = 2 to the 20th power.)

mA. Milliampere; 0.001 ampere.

machine code. The machine language used for entering text and program instructions onto the recording medium or into storage and which is subsequently used for processing and printout.

machine language. (1) A language that is used directly by a machine. (2) Deprecated term for computer instruction code.

magnetic disk. (1) A flat circular plate with a magnetizable surface layer on which data can be stored by magnetic recording. (2) See also diskette.

main storage. (1) Program-addressable storage from which instructions and other data can be loaded directly into registers for subsequent execution or processing. (2) Contrast with auxiliary storage.

mark. A symbol or symbols that indicate the beginning or the end of a field, of a word, of an item of data, or of a set of data such as a file, a record, or a block.

mask. (1) A pattern of characters that is used to control the retention or elimination of portions of another pattern of characters. (2) To use a pattern of characters to control the retention or elimination of portions of another pattern of characters.

masked. Synonym for disabled.

matrix. (1) A rectangular array of elements, arranged in rows and columns, that may be manipulated according to the rules of matrix algebra. (2) In computers, a logic network in the form of an array of input leads and output leads with logic elements connected at some of their intersections.

matrix printer. A printer in which each character is represented by a pattern of dots; for example, a stylus printer, a wire printer. Synonymous with dot printer.

MB. 1,048,576 bytes.

mega (M). Prefix 1,000,000.

megahertz (MHz). 1,000,000 hertz.

memory. Term for main storage.

meter (m). A unit of length (equivalent to 39.37 inches).

MFM. Modified frequency modulation.

MHz. Megahertz; 1,000,000 hertz.

micro (μ). Prefix 0.000,001.

microcode. (1) One or more microinstructions. (2) A code, representing the instructions of an instruction set, implemented in a part of storage that is not program-addressable.

microinstruction. (1) An instruction of microcode. (2) A basic or elementary machine instruction.

microprocessor. An integrated circuit that accepts coded instructions for execution; the instructions may be entered, integrated, or stored internally.

microsecond (μs). 0.000,001 second.

milli (m). Prefix 0.001.

milliampere (mA). 0.001 ampere.

millisecond (ms). 0.001 second.

mnemonic. A symbol chosen to assist the human memory; for example, an abbreviation such as "mpy" for "multiply."

mode. (1) A method of operation; for example, the binary mode, the interpretive mode, the alphanumeric mode. (2) The most frequent value in the statistical sense.

modeling transformation. Operations on the coordinates of an object (usually matrix multiplications) that cause the object to be rotated about any axis, translated (moved without rotating), and/or scaled (changed in size along any or all dimensions). See also viewing transformation.

modem (modulator-demodulator). A device that converts serial (bit by bit) digital signals from a business machine (or data communication equipment) to analog signals that are suitable for transmission in a telephone network. The inverse function is also performed by the modem on reception of analog signals.

modified frequency modulation (MFM). The process of varying the amplitude and frequency of the 'write' signal. MFM pertains to the number of bytes of storage that can be stored on the recording media. The number of bytes is twice the number contained in the same unit area of recording media at single density.

modulation. The process by which some characteristic of one wave (usually high frequency) is varied in accordance with another wave or signal (usually low frequency). This technique is used in modems to make business-machine signals compatible with communication facilities.

modulation rate. The reciprocal of the measure of the shortest nominal time interval between successive significant instants of the modulated signal. If this measure is expressed in seconds, the modulation rate is expressed in baud.

module. (1) A program unit that is discrete and identifiable with respect to compiling, combining with other units, and loading. (2) A packaged functional hardware unit designed for use with other components.

modulo check. A calculation performed on values entered into a system. This calculation is designed to detect errors.

modulo-N check. A check in which an operand is divided by a number N (the modulus) to generate a remainder (check digit)

that is retained with the operand. For example, in a modulo-7 check, the remainder will be 0, 1, 2, 3, 4, 5, or 6. The operand is later checked by again dividing it by the modulus; if the remainder is not equal to the check digit, an error is indicated.

modulus. In a modulo-N check, the number by which the operand is divided.

monitor. Synonym for cathode ray tube display (CRT display).

most-significant digit. The leftmost (non-zero) digit. See also high-order position.

ms. Millisecond; 0.001 second.

multiplexer. A device capable of interleaving the events of two or more activities, or capable of distributing the events of an interleaved sequence to the respective activities.

multiprogramming. (1) Pertaining to the concurrent execution of two or more computer programs by a computer. (2) A mode of operation that provides for the interleaved execution of two or more computer programs by a single processor.

n. Prefix nano; 0.000,000,001.

NAND. A logic operator having the property that if P is a statement, Q is a statement, R is a statement,..., then the NAND of P, Q ,R,... is true if at least one statement is false, false if all statements are true.

NAND gate. A gate in which the output is 0 only if all inputs are 1.

nano (n). Prefix 0.000,000,001.

nanosecond (ns). 0.000,000,001 second.

negative true. Synonym for active low.

negative-going edge. The edge of a pulse or signal changing in a negative direction. Synonymous with falling edge.

non-return-to-zero change-on-ones recording (NRZI). A transmission encoding method in which the data terminal equipment changes the signal to the opposite state to send a binary 1 and leaves it in the same state to send a binary 0.

non-return-to-zero (inverted) recording (NRZI). Deprecated term for non-return-to-zero change-on-ones recording.

NOR. A logic operator having the property that if P is a statement, Q is a statement, R is a statement,..., then the NOR of P, Q, R,... is true if all statements are false, false if at least one statement is true.

NOR gate. A gate in which the output is 0 only if at least one input is 1.

NOT. A logical operator having the property that if P is a statement, then the NOT of P is true if P is false, false if P is true.

NRZI. Non-return-to-zero change-on-ones recording.

ns. Nanosecond; 0.000,000,001 second.

NUL. The null character.

null character (NUL). A control character that is used to accomplish media-fill or time-fill, and that may be inserted into or removed from, a sequence of characters without affecting the meaning of the sequence; however, the control of the equipment or the format may be affected by this character.

odd-even check. Synonym for parity check.

offline. Pertaining to the operation of a functional unit without the continual control of a computer.

one-shot. A circuit that delivers one output pulse of desired duration for each input (trigger) pulse.

open circuit. (1) A discontinuous circuit; that is, one that is broken at one or more points and, consequently, cannot conduct current. Contrast with closed circuit. (2) Pertaining to a no-load condition; for example, the open-circuit voltage of a power supply.

open collector. A switching transistor without an internal connection between its collector and the voltage supply. A connection from the collector to the voltage supply is made through an external (pull-up) resistor.

operand. (1) An entity to which an operation is applied. (2) That which is operated upon. An operand is usually identified by an address part of an instruction.

operating system. Software that controls the execution of programs; an operating system may provide services such as resource allocation, scheduling, input/output control, and data management.

OR. A logic operator having the property that if P is a statement, Q is a statement, R is a statement,..., then the OR of P, Q, R,...is true if at least one statement is true, false if all statements are false.

OR gate. A gate in which the output is 1 only if at least one input is 1.

output. Pertaining to a device, process, or channel involved in an output process, or to the data or states involved in an output process.

output process. (1) The process that consists of the delivery of data from a data processing system, or from any part of it. (2) The return of information from a data processing system to an end user, including the translation of data from a machine language to a language that the end user can understand.

overcurrent. A current of higher than specified strength.

overflow indicator. (1) An indicator that signifies when the last line on a page has been printed or passed. (2) An indicator that is set on if the result of an arithmetic operation exceeds the capacity of the accumulator.

overrun. Loss of data because a receiving device is unable to accept data at the rate it is transmitted.

overvoltage. A voltage of higher than specified value.

parallel. (1) Pertaining to the concurrent or simultaneous operation of two or more devices, or to the concurrent performance of two or more activities. (2) Pertaining to the concurrent or simultaneous occurrence of two or more related activities in multiple devices or channels. (3) Pertaining to the simultaneity of two or more processes. (4) Pertaining to the simultaneous processing of the individual parts of a whole, such as the bits of a character and the characters of a word, using separate facilities for the various parts. (5) Contrast with serial.

parameter. (1) A variable that is given a constant value for a specified application and that may denote the application. (2) A name in a procedure that is used to refer to an argument passed to that procedure.

parity bit. A binary digit appended to a group of binary digits to make the sum of all the digits either always odd (odd parity) or always even (even parity).

parity check. (1) A redundancy check that uses a parity bit. (2) Synonymous with odd-even check.

PEL. Picture element.

personal computer. A small home or business computer that has a processor and keyboard and that can be connected to a television or some other monitor. An optional printer is usually available.

phototransistor. A transistor whose switching action is controlled by light shining on it.

picture element (PEL). The smallest displayable unit on a display.

polling. (1) Interrogation of devices for purposes such as to avoid contention, to determine operational status, or to determine readiness to send or receive data. (2) The process whereby stations are invited, one at a time, to transmit.

port. An access point for data entry or exit.

positive true. Synonym for active high.

positive-going edge. The edge of a pulse or signal changing in a positive direction. Synonymous with rising edge.

potentiometer. A variable resistor with three terminals, one at each end and one on a slider (wiper).

power supply. A device that produces the power needed to operate electronic equipment.

printed circuit. A pattern of conductors (corresponding to the wiring of an electronic circuit) formed on a board of insulating material.

printed-circuit board. A usually copper-clad plastic board used to make a printed circuit.

priority. A rank assigned to a task that determines its precedence in receiving system resources.

processing program. A program that performs such functions as compiling, assembling, or translating for a particular programming language.

processing unit. A functional unit that consists of one or more processors and all or part of internal storage.

processor. (1) In a computer, a functional unit that interprets and executes instructions. (2) A functional unit, a part of another unit such as a terminal or a processing unit, that interprets and executes instructions. (3) Deprecated term for processing program. (4) See microprocessor.

program. (1) A series of actions designed to achieve a certain result. (2) A series of instructions telling the computer how to handle a problem or task. (3) To design, write, and test computer programs.

programmable read-only memory (PROM). A read-only memory that can be programmed by the user.

programming language. (1) An artificial language established for expressing computer programs. (2) A set of characters and rules with meanings assigned prior to their use, for writing computer programs.

programming system. One or more programming languages and the necessary software for using these languages with particular automatic data-processing equipment.

PROM. Programmable read-only memory.

propagation delay. (1) The time necessary for a signal to travel from one point on a circuit to another. (2) The time delay between a signal change at an input and the corresponding change at an output.

protocol. (1) A specification for the format and relative timing of information exchanged between communicating parties. (2) The set of rules governing the operation of functional units of a communication system that must be followed if communication is to be achieved.

pulse. A variation in the value of a quantity, short in relation to the time schedule of interest, the final value being the same as the initial value.

radio frequency (RF). An ac frequency that is higher than the highest audio frequency. So called because of the application to radio communication.

radix. (1) In a radix numeration system, the positive integer by which the weight of the digit place is multiplied to obtain the weight of the digit place with the next higher weight; for example, in the decimal numeration system the radix of each digit place is 10. (2) Another term for base.

radix numeration system. A positional representation system in which the ratio of the weight of any one digit place to the weight of the digit place with the next lower weight is a positive integer (the radix). The permissible values of the character in any digit place range from 0 to one less than the radix.

RAM. Random access memory. Read/write memory.

random access memory (RAM). Read/write memory.

RAS. In the IBM Personal Computer, row address strobe.

raster. In computer graphics, a predetermined pattern of lines that provides uniform coverage of a display space.

read. To acquire or interpret data from a storage device, from a data medium, or from another source.

read-only memory (ROM). A storage device whose contents cannot be modified. The memory is retained when power is removed.

read/write memory. A storage device whose contents can be modified. Also called RAM.

recoverable error. An error condition that allows continued execution of a program.

red-green-blue-intensity (RGBI). The description of a direct-drive color monitor that accepts input signals of red, green, blue, and intensity.

redundancy check. A check that depends on extra characters attached to data for the detection of errors. See cyclic redundancy check.

register. (1) A storage device, having a specified storage capacity such as a bit, a byte, or a computer word, and usually intended for a special purpose. (2) A storage device in which specific data is stored.

retry. To resend the current block of data (from the last EOB or ETB) a prescribed number of times, or until it is entered correctly or accepted.

reverse video. A form of highlighting a character, field, or cursor by reversing the color of the character, field, or cursor with its background; for example, changing a red character on a black background to a black character on a red background.

RF. Radio frequency.

RF modulator. The device used to convert the composite video signal to the antenna level input of a home TV.

RGBI. Red-green-blue-intensity.

rising edge. Synonym for positive-going edge.

ROM. Read-only memory.

ROM/BIOS. The ROM resident basic input/output system, which provides the level control of the major I/O devices in the computer system.

row address strobe (RAS). A signal that latches the row address in a memory chip.

RS-232C. A standard by the EIA for communication between computers and external equipment.

RTS. Request to send. Associated with modem control.

run. A single continuous performance of a computer program or routine.

saturation. In computer graphics, the purity of a particular hue. A color is said to be saturated when at least one primary color (red, blue, or green) is completely absent.

scaling. In computer graphics, enlarging or reducing all or part of a display image by multiplying the coordinates of the image by a constant value.

schematic. The representation, usually in a drawing or diagram form, of a logical or physical structure.

Schottky TTL. A version (S series) of TTL with faster switching speed, but requiring more power. See also transistor-transistor logic and low power Schottky TTL.

SDL. Shielded Data Link

SDLC. Synchronous Data Link Control.

sector. That part of a track or band on a magnetic drum, a magnetic disk, or a disk pack that can be accessed by the magnetic heads in the course of a predetermined rotational displacement of the particular device.

SERDES. Serializer/deserializer.

serial. (1) Pertaining to the sequential performance of two or more activities in a single device. In English, the modifiers serial and parallel usually refer to devices, as opposed to sequential and consecutive, which refer to processes. (2) Pertaining to the sequential or consecutive occurrence of two or more related activities in a single device or channel. (3) Pertaining to the sequential processing of the individual parts of a whole, such as the bits of a character or the characters of a word, using the same facilities for successive parts. (4) Contrast with parallel.

serializer/deserializer (SERDES). A device that serializes output from, and deserializes input to, a business machine.

setup. (1) In a computer that consists of an assembly of individual computing units, the arrangement of interconnections between the units, and the adjustments needed for the computer to operate. (2) The preparation of a computing system to perform a job or job step. Setup is usually performed by an operator and often involves performing routine functions, such as mounting tape reels. (3) The preparation of the system for normal operation.

short circuit. A low-resistance path through which current flows, rather than through a component or circuit.

signal. A variation of a physical quantity, used to convey data.

sink. A device or circuit into which current drains.

software. (1) Computer programs, procedures, and rules concerned with the operation of a data processing system. (2) Contrast with hardware.

source. The origin of a signal or electrical energy.

square wave. An alternating or pulsating current or voltage whose waveshape is square.

square wave generator. A signal generator delivering an output signal having a square waveform.

SS. Start-stop.

start bit. (1) A signal to a receiving mechanism to get ready to receive data or perform a function. (2) In a start-stop system, a signal preceding a character or block that prepares the receiving device for the reception of the code elements.

start-of-text (STX). A transmission control character that precedes a text and may be used to terminate the message heading.

start-stop system. A data transmission system in which each character is preceded by a start bit and is followed by a stop bit.

start-stop (SS) transmission. (1) Asynchronous transmission such that a group of signals representing a character is preceded by a start bit and followed by a stop bit. (2) Asynchronous transmission in which a group of bits is preceded by a start bit that prepares the receiving mechanism for the reception and registration of a character and is followed by at least one stop bit that enables the receiving mechanism to come to an idle condition pending the reception of the next character.

static memory. RAM using flip-flops as the memory elements. Data is retained as long as power is applied to the flip-flops. Contrast with dynamic memory.

stop bit. (1) A signal to a receiving mechanism to wait for the next signal. (2) In a start-stop system, a signal following a character or block that prepares the receiving device for the reception of a subsequent character or block.

storage. (1) A storage device. (2) A device, or part of a device, that can retain data. (3) The retention of data in a storage device. (4) The placement of data into a storage device.

strobe. An instrument that emits adjustable-rate flashes of light. Used to measure the speed of rotating or vibrating objects.

STX. Start-of-text.

symbol. (1) A conventional representation of a concept. (2) A representation of something by reason of relationship, association, or convention.

synchronization. The process of adjusting the corresponding significant instants of two signals to obtain the desired phase relationship between these instants.

Synchronous Data Link Control (SDLC). A protocol for management of data transfer over a data link.

synchronous transmission. (1) Data transmission in which the time of occurrence of each signal representing a bit is related to a fixed time frame. (2) Data transmission in which the sending and receiving devices are operating continuously at substantially the same frequency and are maintained, by means of correction, in a desired phase relationship.

syntax. (1) The relationship among characters or groups of characters, independent of their meanings or the manner of their interpretation and use. (2) The structure of expressions in a language. (3) The rules governing the structure of a language. (4) The relationships among symbols.

text. In ASCII and data communication, a sequence of characters treated as an entity if preceded and terminated by one STX and one ETX transmission control character, respectively.

time-out. (1) A parameter related to an enforced event designed to occur at the conclusion of a predetermined elapsed time. A time-out condition can be cancelled by the receipt of an appropriate time-out cancellation signal. (2) A time interval allotted for certain operations to occur; for example, response to polling or addressing before system operation is interrupted and must be restarted.

track. (1) The path or one of the set of paths, parallel to the reference edge on a data medium, associated with a single reading or writing component as the data medium moves past the component. (2) The portion of a moving data medium such as a drum, or disk, that is accessible to a given reading head position.

transistor-transistor logic (TTL). A popular logic circuit family that uses multiple-emitter transistors.

translate. To transform data from one language to another.

transmission. (1) The sending of data from one place for reception elsewhere. (2) In ASCII and data communication, a series of characters including headings and text. (3) The dispatching of a signal, message, or other form of intelligence by wire, radio, telephone, or other means. (4) One or more blocks or messages. For BSC and start-stop devices, a transmission is terminated by an EOT character. (5) Synonymous with data transmission.

TTL. Transistor-transistor logic.

typematic key. A keyboard key that repeats its function when held pressed.

V. Volt.

vector. In computer graphics, a directed line segment.

video. Computer data or graphics displayed on a cathode ray tube, monitor, or display.

view point. In computer graphics, the origin from which angles and scales are used to map virtual space into display space.

viewing reference point. In computer graphics, a point in the modeling coordinate space that is a defined distance from the view point.

viewing transformation. Operations on the coordinates of an object (usually matrix multiplications) that cause the view of the object to be rotated about any axis, translated (moved without rotating), and/or scaled (changed in size along any or all dimensions). Viewing transformation differs from modeling transformation in that perspective is considered. See also modeling transformation.

viewplane. The visible plane of a CRT display screen that completely contains a defined window.

viewport. In computer graphics, a predefined part of the CRT display space.

volt. The basic practical unit of electric pressure. The potential that causes electrons to flow through a circuit.

W. Watt.

watt. The practical unit of electric power.

window. (1) A predefined part of the virtual space. (2) The visible area of a viewplane.

word. (1) A character string or a bit string considered as an entity. (2) See computer word.

write. To make a permanent or transient recording of data in a storage device or on a data medium.

write precompensation. The varying of the timing of the head current from the outer tracks to the inner tracks of the diskette to keep a constant 'write' signal.

yon plane. In computer graphics, a plane that is perpendicular to the line joining the viewing reference point and the view point, and that lies beyond the viewing reference point. Any part of an object beyond the yon plane is not seen. See also hither plane.

Bibliography

- Microprocessor and Peripheral Handbook
  - INTEL Corporation.*210844.001*
- Introduction to the iAPX 286
  - INTEL Corporation.*210308.001*
- iAPX 286 Operating Systems Writer's Guide
  - INTEL Corporation.*121960.001*
- iAPX 286 Programmer's Reference Manual
  - INTEL Corporation.*210498.001*
- iAPX 286 Hardware Reference Manual
  - INTEL Corporation.*210760.001*
- Numeric Processor Extension Data Sheet
  - INTEL Corporation.*210920*
- 80287 Support Library Reference Manual
  - INTEL Corporation.*122129*
- National Semiconductor Corporation. *NS16450*
- Motorola Microprocessor's Data Manual
  - Motorola Inc. *Series B*

Notes:

Index

C

D

E

F

G

H

I

L

M

Q

R

S

T

U

V

W

X

Y

Z

Numerics

Notes:

Notes: