IBM

Personal Computer Hardware Reference Library

Technical Reference

6183355

Personal Computer
Hardware Reference
Library

Technical Reference

Revised Edition (March, 1986)

Federal Communications Commission Radio Frequency Interference Statement

Warning: The equipment described herein has been certified to comply with the limits for a Class B computing device, pursuant to Subpart J of Part 15 of the FCC rules. Only peripherals (computer input/output devices, terminals, printers, etc.) certified to comply with the Class B limits may be attached to the computer. Operation with non-certified peripherals is likely to result in interference to radio and TV reception. If peripherals not offered by IBM are used with the equipment, it is suggested to use shielded grounded cables with in-line filters if necessary.

CAUTION
This product described herein is equipped with a grounded plug for the user's safety. It is to be used in conjunction with a properly grounded receptacle to avoid electrical shock.

Notes:

Preface

This manual describes the various units of the IBM PERSONAL COMPUTER AT® and how they interact. It also has information about the basic input/output system (BIOS) and about programming support. Where timing considerations between 6- and 8-MHz are different, the 8-MHz time is shown in parentheses.

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 AT.

This manual consists of nine sections:

- The first three sections describe the IBM PERSONAL COMPUTER AT 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 has general communications information
- Section 9 contains information about the compatibility of the IBM PERSONAL COMPUTER AT and the rest of the IBM Personal Computer family.

A glossary and a bibliography are included.

Prerequisite Publications

Guide to Operations for the IBM PERSONAL COMPUTER AT

Suggested Reading

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

Contents

Notes:

INDEX TAB LISTING

SECTION 1

SECTION 2

SECTION 3

SECTION 4

SECTION 5

SECTION 6

Notes:

SECTION 7

SECTION 8

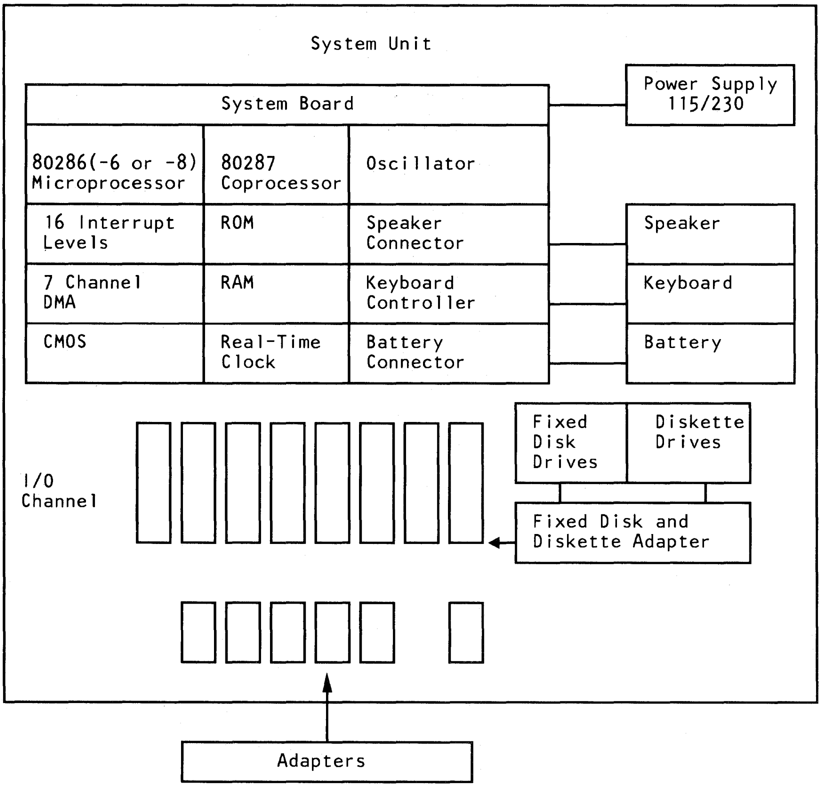
SECTION 9

GLOSSARY

BIBLIOGRAPHY

INDEX

System Block Diagram



SECTION 1. SYSTEM BOARD

The type 1 system board is approximately 30.5 by 35 centimeters (12 by 13.8 inches). The type 2 system board is approximately 23.8 by 35 centimeters (9.3 by 13.8 inches). Both types of system boards use very large scale integration (VLSI) technology and have 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 512K random-access memory (RAM) Subsystem
- Eight input/output (I/O) slots:
  - Six with a 36-pin and a 62-pin card-edge socket
  - Two with only the 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 type 1 system board has four banks of memory sockets, each supporting 9 128K-by-1-bit modules for a total memory size of 512K, with parity checking.

The type 2 system board has two banks of memory sockets, each supporting 9 256K-by-1-bit modules for a total memory size of 512K, 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 or 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.

---

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

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. For example, a word with its low-order byte at offset FFFF and its high-order byte at 0000. If, in the real 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

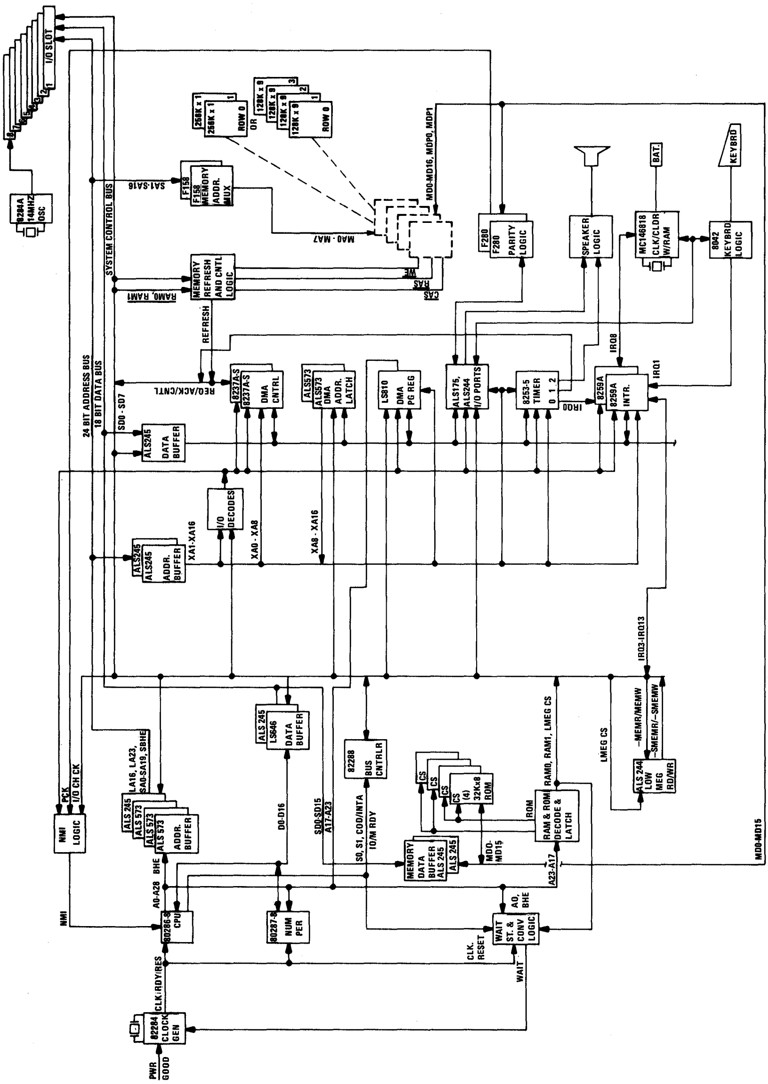
The protected mode offers extended physical and virtual memory address space, memory protection mechanisms, and new operations to support operating systems and virtual memory.

> **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*.

Following is a block diagram of the system board.



System Performance

Note: Where timing considerations between 6- and 8-MHz are different, the 8-MHz time is shown in parentheses.

The 80286 microprocessor operates at 6 MHz (8 MHz), resulting in a clock cycle time of 167 nanoseconds (125 nanoseconds).

A bus cycle requires 3 clock cycles (which includes 1 wait state) so that a 500-nanosecond (375-nanosecond), 16-bit, microprocessor cycle time is achieved. Eight-bit bus operations to 8-bit devices take 6 clock cycles (which include 4 wait states), resulting in a 1000-nanosecond (750-nanosecond) microprocessor cycle. Sixteen-bit bus operations to 8-bit devices take 12 clock cycles (which include 10 wait states) resulting in a 2-microsecond (1.5-microsecond) microprocessor cycle.

The refresh controller steps one refresh address every 15 microseconds. Each refresh cycle requires 8 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 3.89ms. The following formula determines the percentage of bandwidth used for refresh for the 6 MHz clock.

```
% Bandwidth used     8 cycles X 256    2048
   for Refresh    = ------------- = ----- = 8.7%
                     3.89ms/167ns      23293
```

The following formula determines the percentage of bandwidth used for refresh for the 8 MHz clock.

```
% Bandwidth used     8 cycles X 256    2048
   for Refresh    = ------------- = ----- = 6.5%
                     3.89ms/125ns     31120
```

The DMA controller operates at 3 MHz (4 MHz), which results in a clock cycle time of 333 nanoseconds (250 nanoseconds). All DMA data-transfer bus cycles are 5 clock cycles or 1.66 microseconds (1.25 microseconds). Cycles spent in the transfer of bus control are not included.

DMA channels 0, 1, 2, and 3 are used for 8-bit data transfers, and channels 5, 6, and 7 process 16-bit transfers. Channel 4 is used to cascade channels 0 through 3 to the microprocessor.

The following figure is a system memory map.

Address	Name	Function
000000 to 07FFFF	512K system board	System board memory
080000 to 09FFFF	128K	I/O channel memory - IBM Personal Computer AT 128K Memory Expansion Option or 128/640K Memory Card
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 - 512K 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 Map

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 (IBM Personal Computer) Ch 3 - Reserved	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 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.

The DMA controller command code addresses follow.

Hex Address	Register Function
0C0	CH0 base and current address
0C2	CH0 base and current word count
0C4	CH1 base and current address
0C6	CH1 base and current word count
0C8	CH2 base and current address
0CA	CH2 base and current word count
0CC	CH3 base and current address
0CE	CH3 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

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.) Cluster (Primary) 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 Aquisition and Control *** GPIB ** Cluster (Secondary)

* The PC Network is jumper selectable.
** The GPIB Adapter can be set to interrupts 2 through 7.
*** The Data Acquisition Adapter can be set to interrupts 3 through 7. The default interrupt is 7.

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

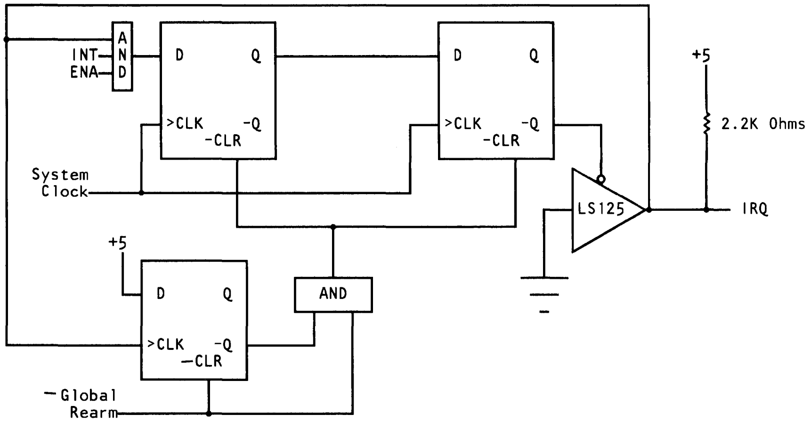
Most interrupt-supporting adapters hold the 'interrupt request' line (IRQ) at a low level and then drive the line high to cause an interrupt. In contrast, the shared-interrupt hardware design allows IRQ to float high through pull-up resistors on each adapter. Each adapter on the line may cause an interrupt by pulsing the line to a low level. The leading edge of the pulse arms the 8259A Interrupt Controller; the trailing edge signals the interrupt controller to cause 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 hardware 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:

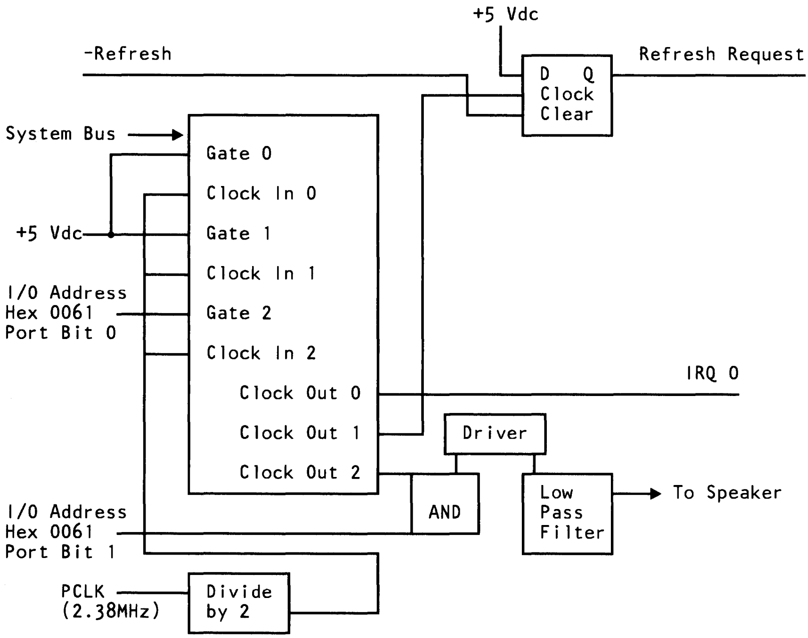
Channel 0	**System Timer**
GATE 0	Tied on
CLK IN 0	1.193182 MHz OSC
CLK OUT 0	8259A IRQ 0

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 either a 12-MHz or 16-MHz crystal. Its output 'clock' signal (CLK) is the input to the system microprocessor, the coprocessor, and 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 260 nanoseconds (190ns) and its maximum cycle time is 480ns (360ns).

RAM Subsystem

The system board's RAM subsystem starts at address 000000 of the 16M address space. It is 256K or 512K of 128K-by-1-bit RAM modules (type 1 system board) or 512K of 256K-by-1-bit RAM modules (type 2 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:

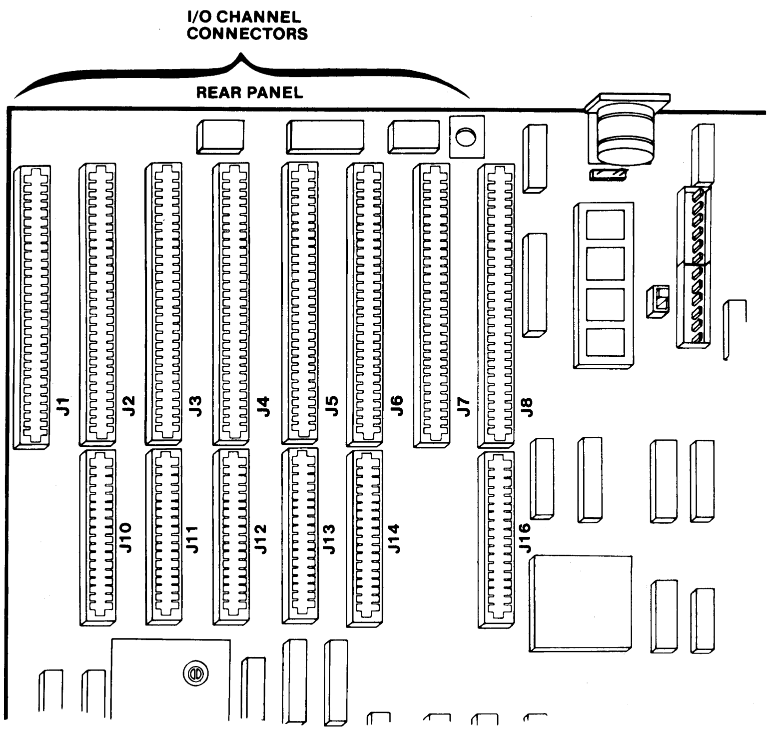
- I/O address space hex 100 to hex 3FF
- 24-bit memory addresses (16M)
- Selection of data accesses (either 8- or 16-bit)
- Interrupts
- DMA channels
- I/O wait-state generation

- Open-bus structure (allowing multiple microprocessors to share the system's resources, including memory)
- Refresh of system memory from channel microprocessors.

Connectors

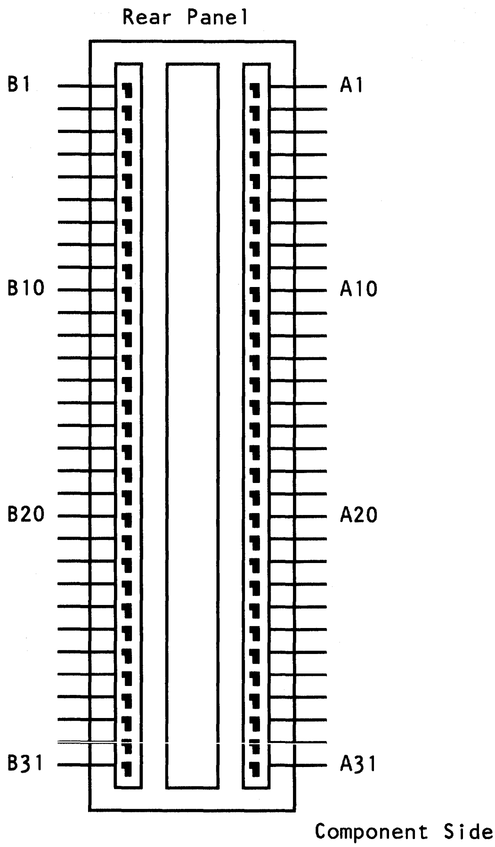
The following figure shows the location and the numbering of the I/O channel connectors. These connectors consist of six 36-pin and eight 62-pin edge connector sockets.

> **Note:** The 36-pin connector is not present in two positions on the I/O channel. These positions can support only 62-pin I/O bus adapters.



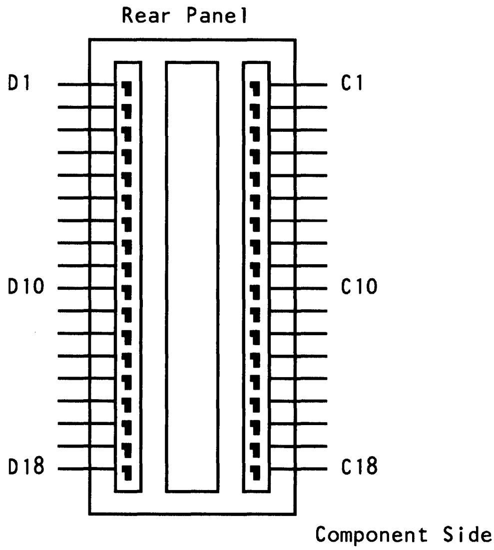
I/O Channel Connector Location

The following figure shows the pin numbering for I/O channel connectors J1 through J8.



I/O Channel Pin Numbering (J1-J8)

The following figure shows the pin numbering for I/O channel connectors J10 through J14 and J16.



I/O Channel Pin Numbering (J10-J14 and J16)

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, J1 through J8)

SECTION 1

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	OSC	O
B31	GND	Ground

I/O Channel (B-Side, J1 through J8)

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, J10 through J14 and 16)

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, J10 through J14 and 16)

SECTION 1

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.

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 when BALE is high. LA17 through LA23 are not latched during microprocessor cycles and therefore do not stay valid for the whole cycle. Their purpose is to generate memory decodes for 16-bit, 1 wait-state, memory cycles. These decodes should be latched by I/O adapters on the falling edge of 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 6- or 8-MHz system 'clock' signal. It is a synchronous microprocessor cycle clock with a cycle time of 167 nanoseconds (125 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 bus 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 provided by the 82288 Bus Controller and is used on the system board to latch valid addresses and memory decodes from the microprocessor. It 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.

-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).

> **Note:** Interrupt 13 is used on the system board and is not available on the I/O channel. IRQ 8 is used for the real-time clock.

-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 high.

-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.

OSC (O)

The '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. In order to run a memory cycle to an 8-bit device with a minimum of two wait states, 0WS should

SECTION 1

be driven active one clock cycle after the Read or Write command is active, and gated with the address decode for the device. Memory Read and Write commands to an 8-bit device are active on the falling edge of CLK. 0WS is active low and should be driven with an open collector or tri-state driver capable of sinking 20 mA.

The following figure is an I/O address map.

Hex Range	Device
000-01F	DMA controller 1, 8237A-5
020-03F	Interrupt controller 1, 8259A, Master
040-05F	Timer, 8254-2
060-06F	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
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.	

I/O Address Map (Part 1 of 2)

Hex Range	Device
1F0-1F8	Fixed Disk
200-207	Game I/O
20C-20D	Reserved
21F	Reserved
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 card
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
390-393	Cluster
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)
790-793	Cluster (Adapter 1)
AE2 & AE3	Data Acquisition (Adapter 2)
B90-B93	Cluster (Adapter 2)
EE2 & EE3	Data Acquisition (Adapter 3)
1390-1393	Cluster (Adapter 3)
22E1	GPIB (Adapter 1)
2390-2393	Cluster (Adapter 4)
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.	

I/O Address Map (Part 2 of 2)

NMI and Coprocessor Controls

At power-on time, 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 Write to I/O address hex 070, with data bit 7 equal to a logic 0.

Mask Off Write to I/O address hex 070, with data bit 7 equal to a logic 1.

Note: At the end of POST, the system sets the NMI mask on (NMI enabled).

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.

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).

The '-I/O channel check' signal (-I/O CH CK) is used to report non-correctable errors on RAM adapters on the I/O channel. This check will create an NMI if the NMI is enabled. At power-on time, 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 will establish 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) will result. Read the status bits to determine the source of the NMI (see the figure, "I/O Address Map", on page 1-37). 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 will be reset to inactive.

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

RAM Jumpers

The system board has a 3-pin, Berg-strip connector (J18). Starting at the front of the system, the pins are numbered 1 through 3. Jumper placement across these pins determines how much system board RAM is enabled. Pin assignments follow.

Pin	Assignments
1	No Connection
2	- RAM SEL
3	Ground

RAM Jumper Connector (J18)

The following shows how the jumpers affect RAM.

Jumper Positions	Function
1 and 2	Enable 2nd 256K of system board RAM
2 and 3	Disable 2nd 256K of system board RAM

RAM Jumper

> **Note:** The normal mode is the enable mode. The other mode permits the additional RAM to reside on adapters plugged into the I/O bus.

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 or Professional Graphics Controller.

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.

Variable Capacitor

The system board has a variable capacitor. Its purpose is to adjust the 14.31818 MHz oscillator signal (OSC), used to obtain the color-burst signal required for color televisions.

Keyboard Controller

The keyboard controller is a single-chip microcomputer (Intel 8042) 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 either 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 both keyboards are 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 101/102-Key Keyboard to establish Mode 2 protocol (see "Data Stream" on page 4-61).

Receiving Data from the Keyboard

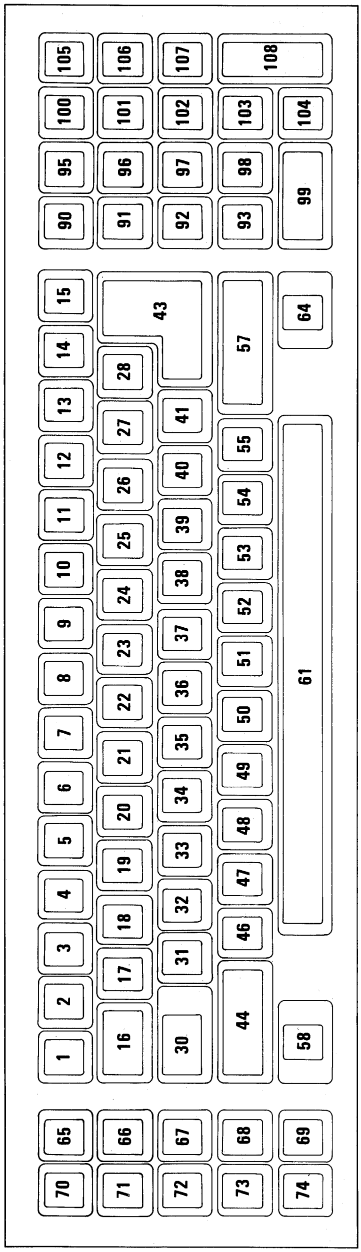
The keyboard sends data in a serial format using an 11-bit frame. The first bit is a start bit, and is followed by eight data bits, an odd parity bit, and a stop bit. Data sent is synchronized by a clock supplied by the keyboard. At the end of a transmission, the keyboard controller disables the interface 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 1-51 for a detailed description of the command byte.

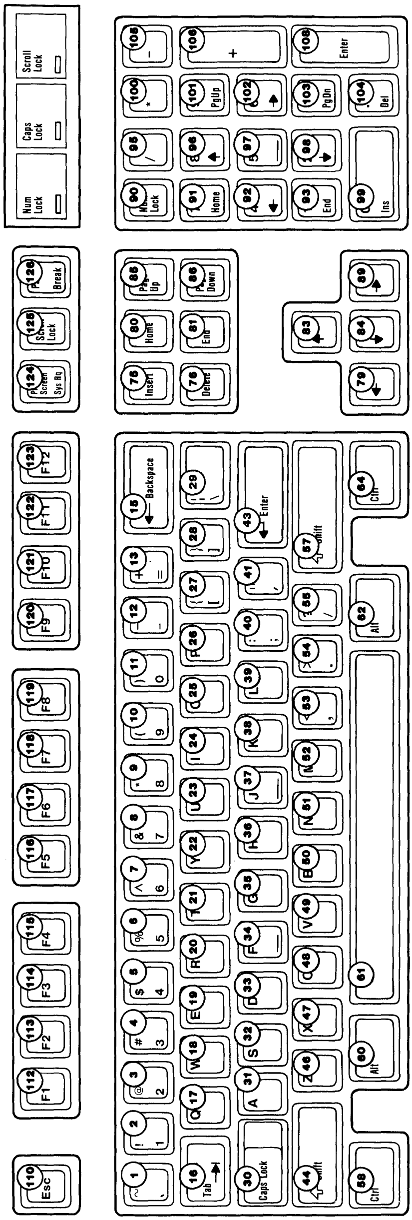
Scan Code Translation

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 84-key and the 101/102-key keyboard layouts. Each key position is numbered for reference.

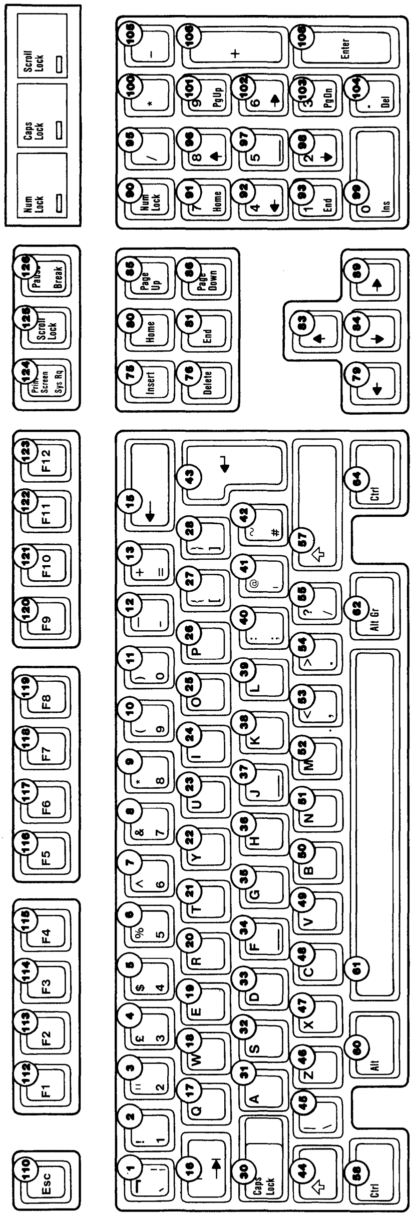
84-Key Keyboard



101-Key Keyboard



102-Key Keyboard



SECTION 1

The following figure is the scan-code translation table.

System Scan Code	Keyboard Scan Code	Key (101/102-key)	Key (84-key)
01	76	110	90
02	16	2	2
03	1E	3	3
04	26	4	4
05	25	5	5
06	2E	6	6
07	36	7	7
08	3D	8	8
09	3E	9	9
0A	46	10	10
0B	45	11	11
0C	4E	12	12
0D	55	13	13
0E	66	15	15
0F	0D	16	16
10	15	17	17
11	1D	18	18
12	24	19	19
13	2D	20	20
14	2C	21	21
15	35	22	22
16	3C	23	23
17	43	24	24
18	44	25	25
19	4D	26	26
1A	54	27	27
1B	5B	28	28
1C	5A	43	43
1D	14	58	30
1E	1C	31	31
1F	1B	32	32
20	23	33	33
21	2B	34	34
22	34	35	35
23	33	36	36
24	3B	37	37
25	42	38	38
26	4B	39	39
27	4C	40	40
28	52	41	41
29	0E	1	1
2A	12	44	44
2B	5D	29 (U.S. only) 42 (except U.S.)	14
2C	1A	46	46
2D	22	47	47
2E	21	48	48
2F	2A	49	49

Scan-Code Translation Table (Part 1 of 3)

System Scan Code	Keyboard Scan Code	Key (101/102-key)	Key (84-key)
30	32	50	50
31	31	51	51
32	3A	52	52
33	41	53	53
34	49	54	54
35	4A	55	55
36	59	57	57
38	11	60	58
39	29	61	61
3A	58	30	64
3B	05	112	70
3C	06	113	65
3D	04	114	71
3E	0C	115	66
3F	03	116	72
40	0B	117	67
41	02 or 83	118	73
42	0A	119	68
43	01	120	74
44	09	121	69
45	77	-	95
46	7E	125	100
47	6C	91	91
48	75	96	96
49	7D	101	101
4A	7B	105	107
4B	6B	92	92
4C	73	97	97
4D	74	102	102
4E	79	106	108
4F	69	93	93
50	72	98	98
51	7A	103	103
52	70	99	99
53	71	104	104
54	7F or 84	-	105
D5	F0 60	45 (except U.S.)	-
D9	F0 0F	122	-
DA	F0 17	123	-
FF	00	-	-
2A 37	12 7C	124	-
45 C5	77 F0 77	90	-
E0 1C	F0 47 5A	108	-
E0 1D	F0 47 14	64	-
E0 35	F0 47 4A	95	-
E0 37	F0 47 7C	100	-
E0 38	F0 47 11	62	-
E0 47	F0 47 6C	80	-

Scan-Code Translation Table (Part 2 of 3)

System Scan Code	Keyboard Scan Code	Key (101/102-key)	Key (84-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	-
1D E0 45 E0 C5 9D	14 F0 47 77 F0 47 F0 77 F0 14	126	-

Scan-Code Translation Table (Part 3 of 3)

Notes:

SECTION 1

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.

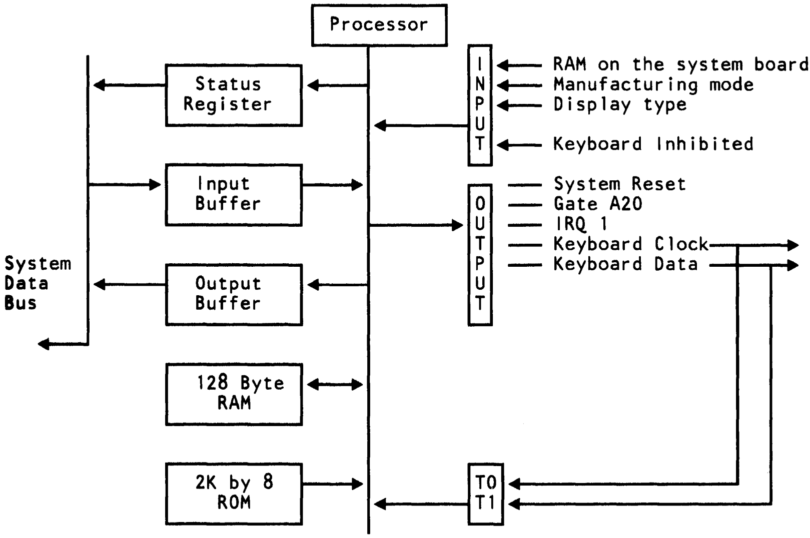
The keyboard is required to respond to all transmissions. The keyboard responds to any valid command and parameter, other than Echo and Resend, with an Acknowledge (ACK) response, hex FA. If the response contains a parity error, the keyboard controller places a hex FE in its output buffer, and the transmit 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.

Inhibit

The keyboard interface may be inhibited by setting input port bit 7 (keyboard inhibit switch) to 0. All transmissions to the keyboard will be allowed regardless of the state of this bit. The keyboard controller tests data received from the keyboard to determine if the byte received is a command response or a scan code. If the byte is a command response, it is placed in the keyboard controller's output buffer. If the byte is a scan code, it is ignored.

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.

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 Inhibit Switch—This bit is updated whenever data is placed in the keyboard controller's output buffer. It reflects the state of the keyboard-inhibit switch. A 0 indicates the keyboard is inhibited.

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 Inhibit Override—Writing a 1 to this bit disables the keyboard inhibit function.

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.

AC Diagnostic Dump—Sends 16 bytes of the controller's RAM, the current state of the input port, the current state of the output port, and the controller's program status word to the system. All items are sent in scan-code format.

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	Keyboard inhibit switch 0 = Keyboard inhibited 1 = Keyboard not inhibited
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	RAM on the system board 0 = Enable 512K of system board RAM 1 = Enable 256K of system board RAM
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

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

Test-Input Bit Definitions

Real-Time Clock/CMOS RAM Information

The RT/CMOS RAM chip (Motorola MC146818) 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 type byte - types 1-14
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 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 Information (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, and a 1 indicates that the chip lost power.

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

SECTION 1

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).

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).

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 first 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 1-65.
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 1-65.

The following table shows the BIOS fixed disk parameters.

Type	Cylinders	Heads	Write Pre-Comp	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 one of the following:

- the Enhanced Graphics Adapter
- the Professional Graphics Controller.

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:

0100H 256K–system board RAM
0200H 512K–system board RAM
0280H 640K–512K system board RAM, the IBM Personal Computer AT 128KB Memory Expansion Option, or the 128/640KB Memory Expansion Option

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).

SECTION 1

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 23.

Note: Types 24 through 255 are reserved.

Type	Cylinders	Heads	Write Pre-Comp	Landing Zone
16	612	4	All Cyl	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	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 CMOS RAM involves two steps:

1. OUT to port hex 70 with the CMOS address that will be written to.

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 CMOS address that is to be read from.

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

Specifications

System Unit

Size

- Length: 538 millimeters (21.2 inches)
- Depth: 429 millimeters (16.9 inches)
- Height: 142 millimeters (5.6 inches)

Weight

- 19.5 kilograms (43 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: 2133.6 meters (7000 feet)

Heat Output

- 1229 British Thermal Units (BTU) per hour

Noise Level

- 42 decibels average-noise rating (without printer)

Electrical

- Power: 450 VA
- Range 1
  - Nominal: 115 Vac
  - Minimum: 100 Vac
  - Maximum: 125 Vac
- Range 2
  - Nominal: 230 Vac
  - Minimum: 200 Vac
  - Maximum: 240 Vac

Connectors

The system board has the following additional connectors:

- Two power-supply connectors (PS8 and PS9)
- Speaker connector (J19)
- Power LED and key lock connector (J20)
- Battery connector (J21)
- Keyboard connector (J22)

The pin assignments for the power-supply connectors, PS8 and PS9, are as follows. The pins are numbered 1 through 6 from the rear of the system.

Connector	Pin	Assignments
PS8	1	Power Good
	2	+5 Vdc
	3	+12 Vdc
	4	-12 Vdc
	5	Ground
	6	Ground
PS9	1	Ground
	2	Ground
	3	-5 Vdc
	4	+5 Vdc
	5	+5 Vdc
	6	+5 Vdc

Power Supply Connectors (PS8, PS9)

The speaker connector, J19, is a 4-pin, keyed, Berg strip. The pins are numbered 1 through 4 from the front of the system. The pin assignments are as follows:

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

Speaker Connector (J19)

The power LED and key lock connector, J20, is a 5-pin Berg strip. The pins are numbered 1 through 5 from the front of the system. The pin assignments are as follows:

Pin	Assignments
1	LED Power
2	Key
3	Ground
4	Keyboard Inhibit
5	Ground

Power LED and Key Lock Connector (J20)

The battery connector, J21, is a 4-pin, keyed, Berg strip. The pins are numbered 1 through 4 from the right of the system. The pin assignments are as follows:

Pin	Assignments
1	Ground
2	Not Used
3	Key
4	6 Vdc

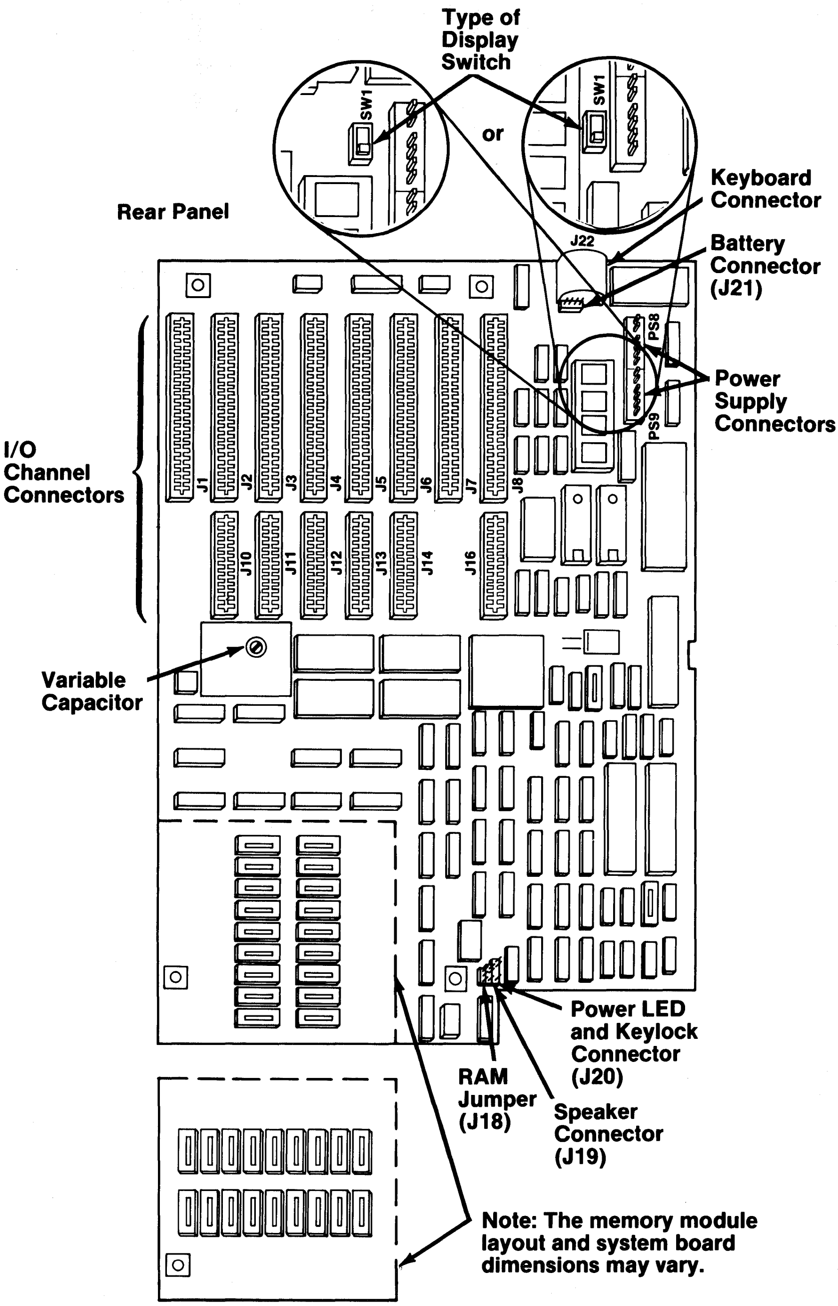
Battery Connector (J21)

The keyboard connector, J22, is a 5-pin, 90-degree Printed Circuit Board (PCB) mounting, DIN connector. For pin numbering, see the "Keyboard" Section. The pin assignments are as follows:

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

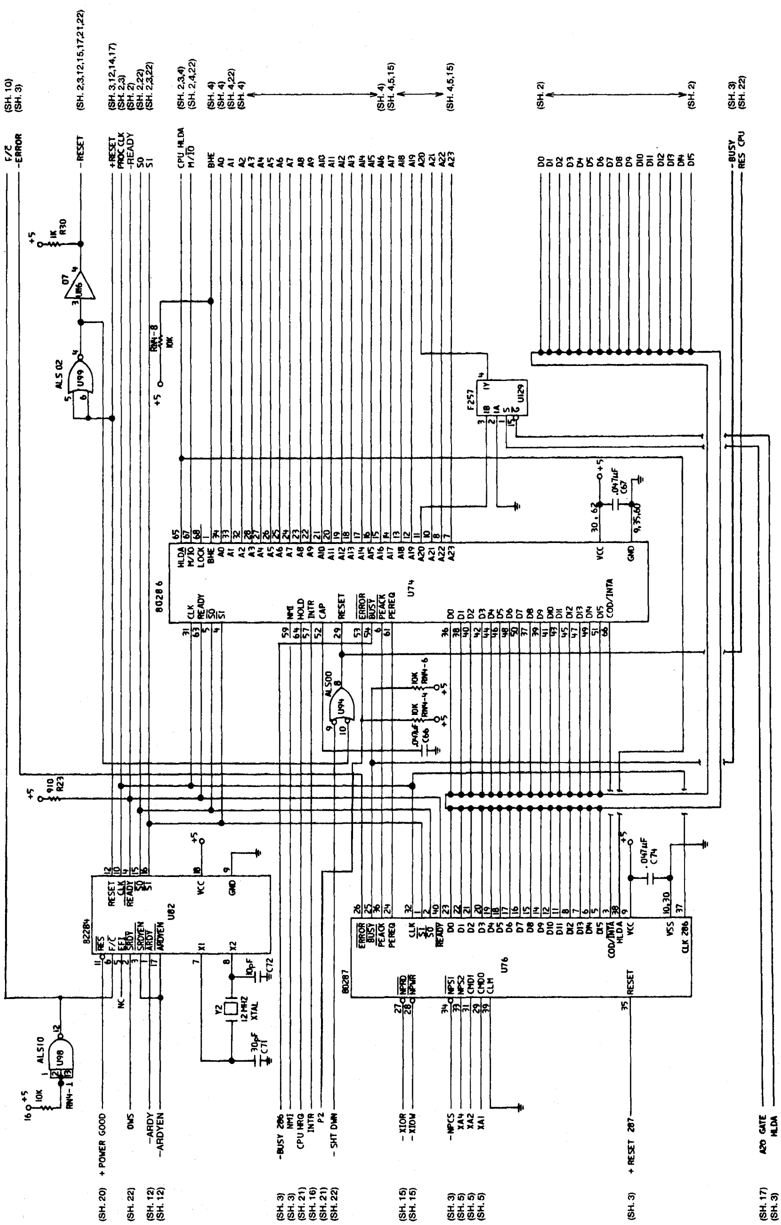
Keyboard Connector (J22)

The following figure shows the layout of the system board.

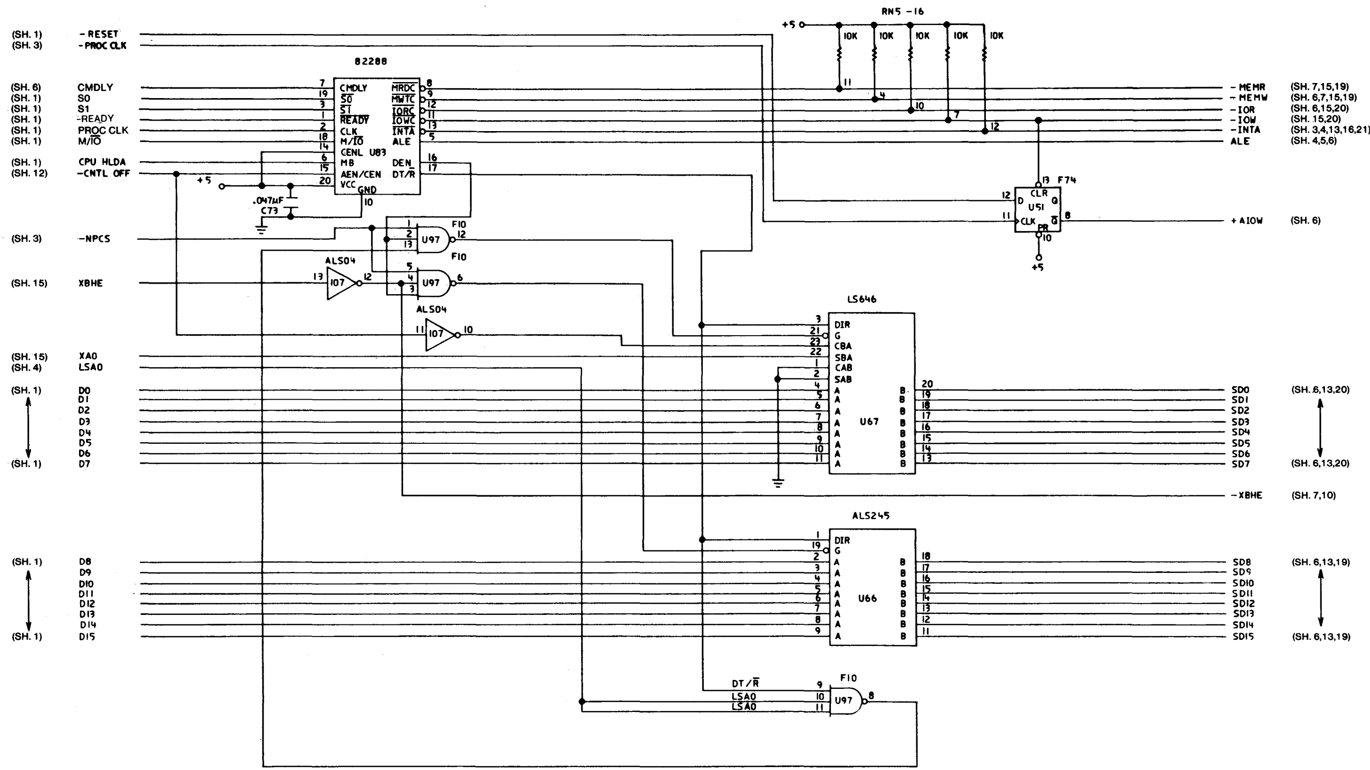


Notes:

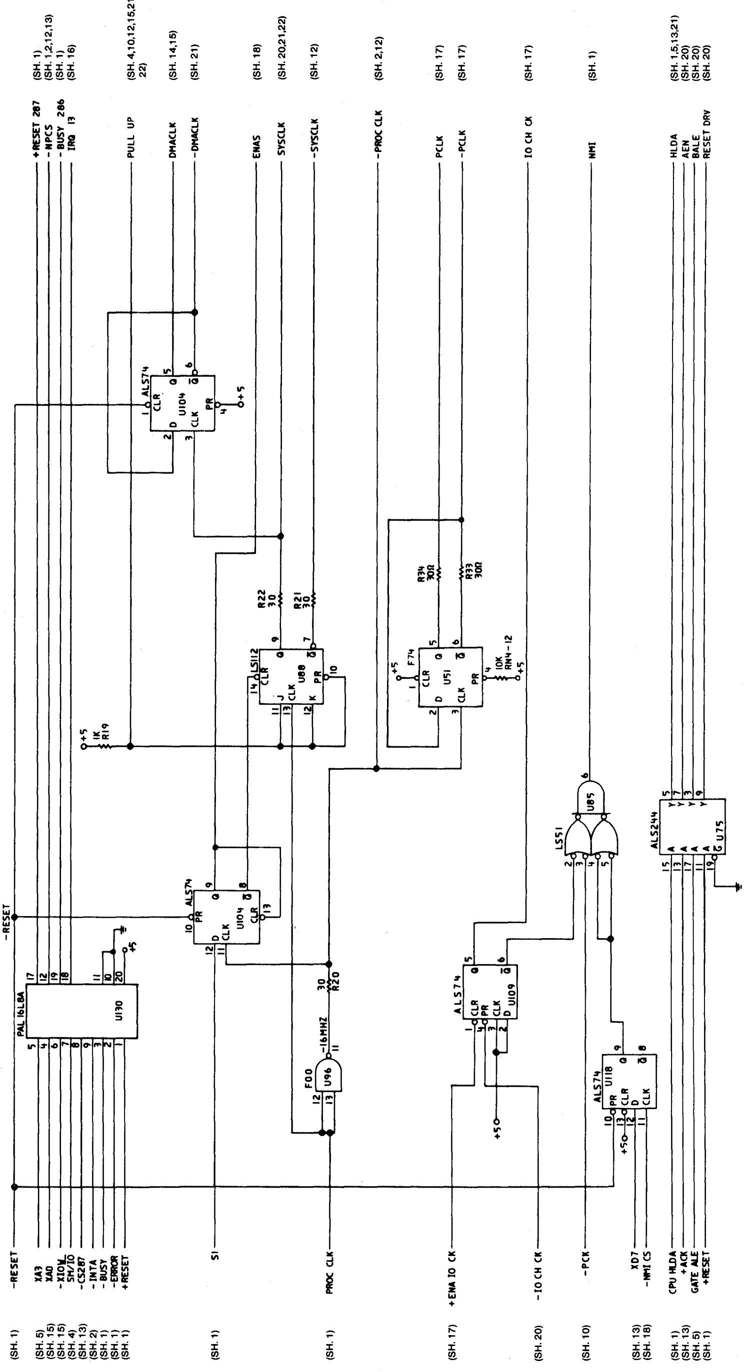
Logic Diagrams - Type 1



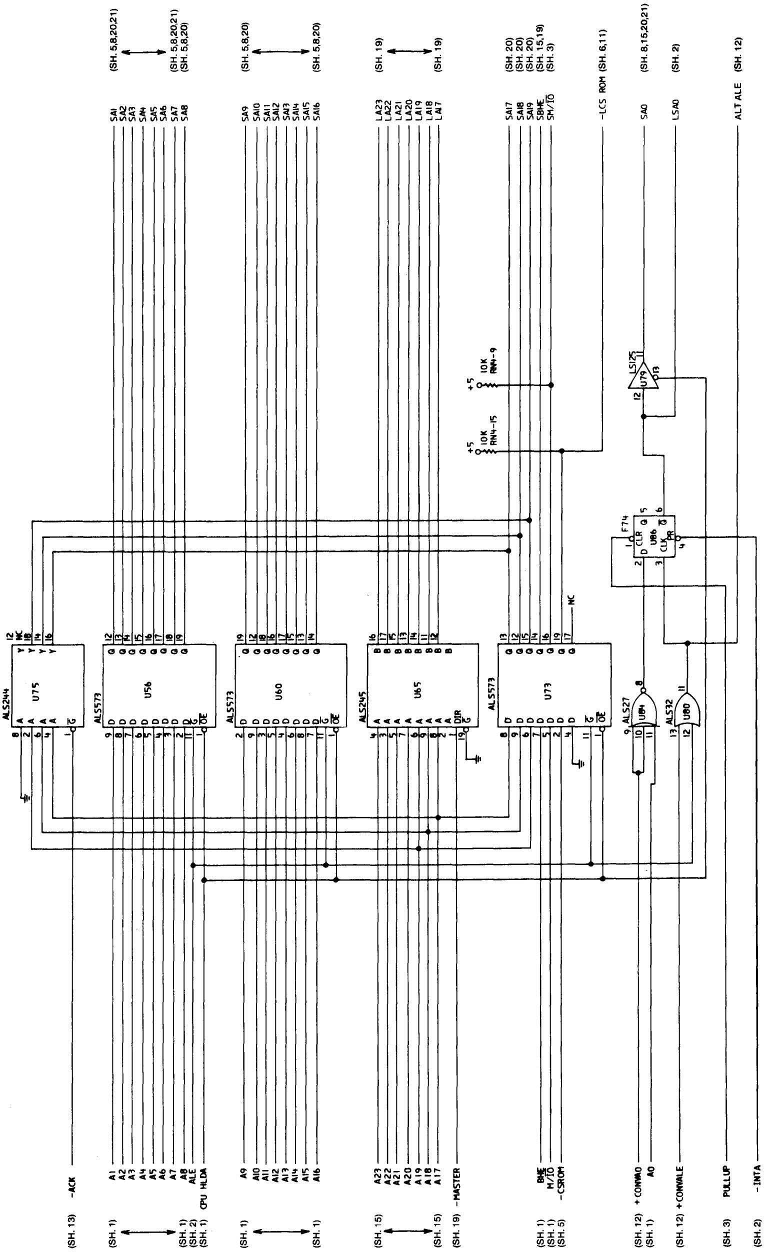
Type 1 512KB Planar (Sheet 1 of 22)



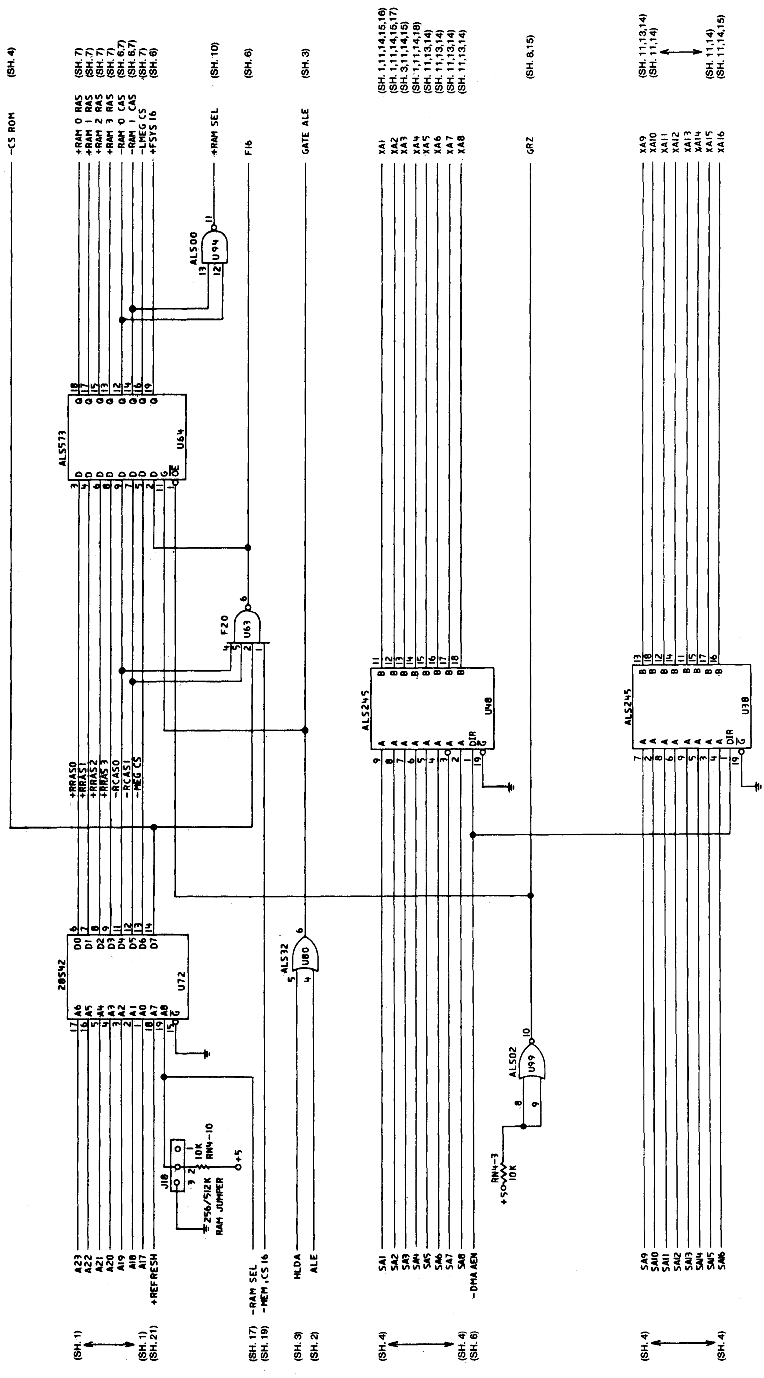
Type 1 512KB Planar (Sheet 2 of 22)



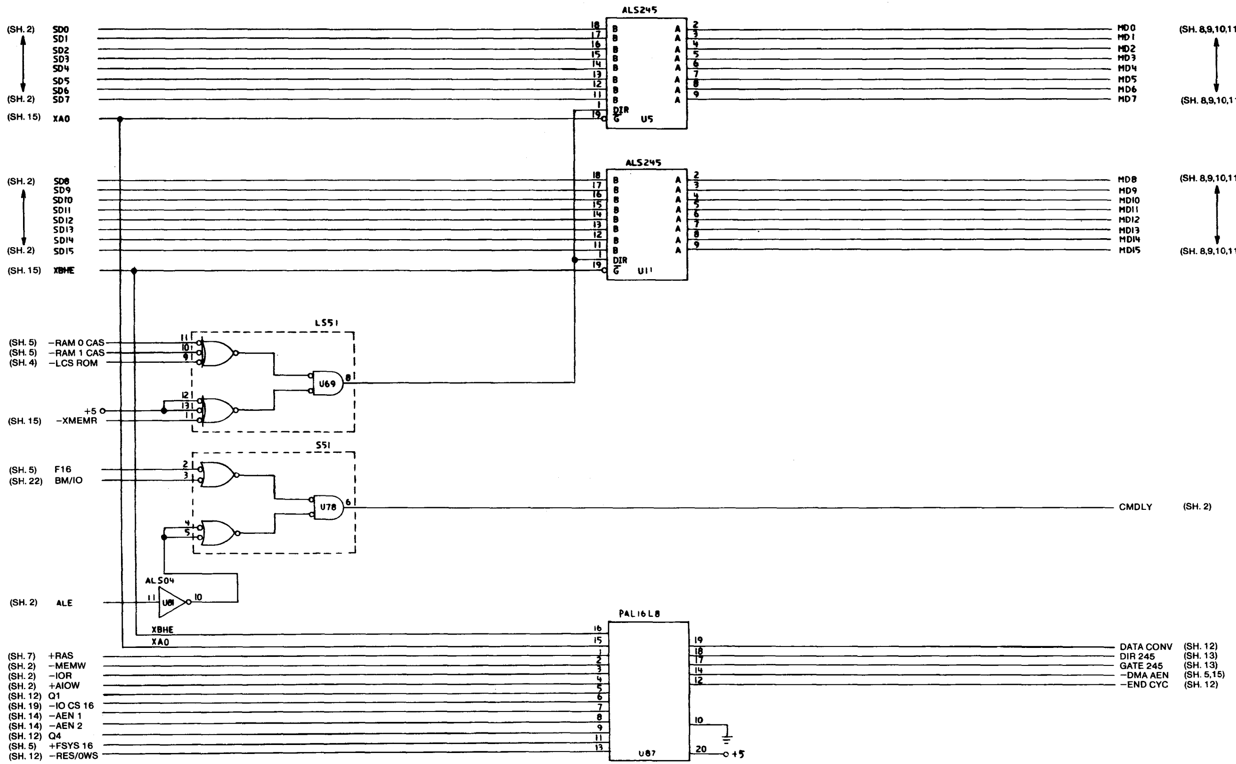
Type 1 512KB Planar (Sheet 3 of 22)



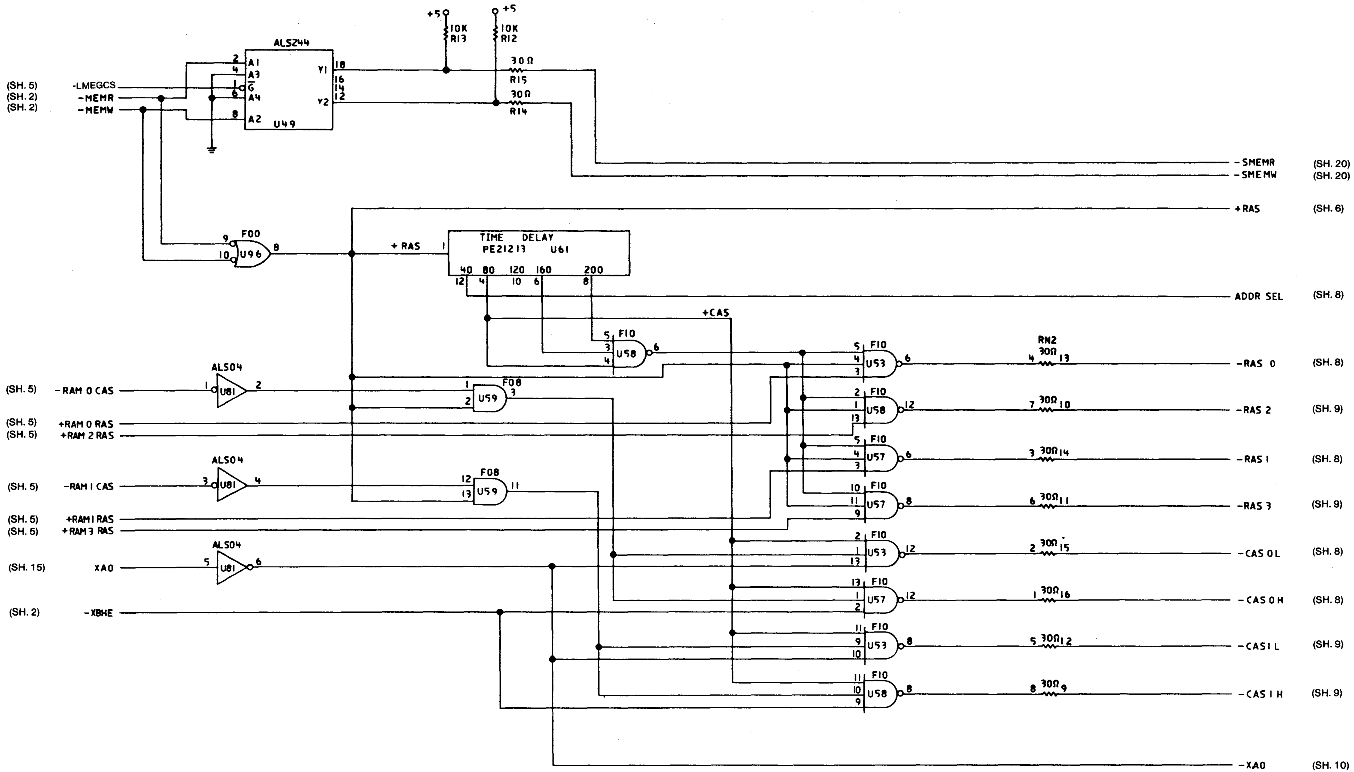
Type 1 512KB Planar (Sheet 4 of 22)



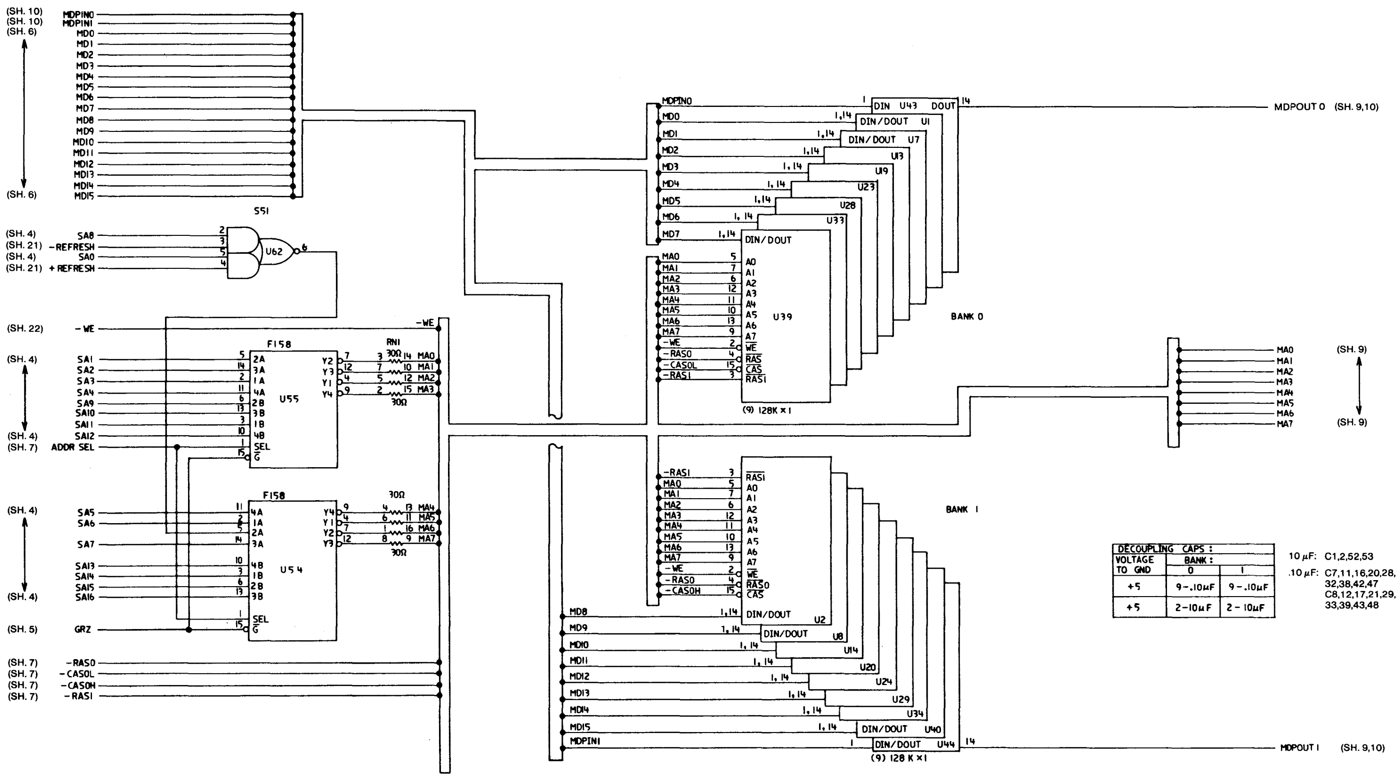
Type 1 512KB Planar (Sheet 5 of 22)



Type 1 512KB Planar (Sheet 6 of 22)



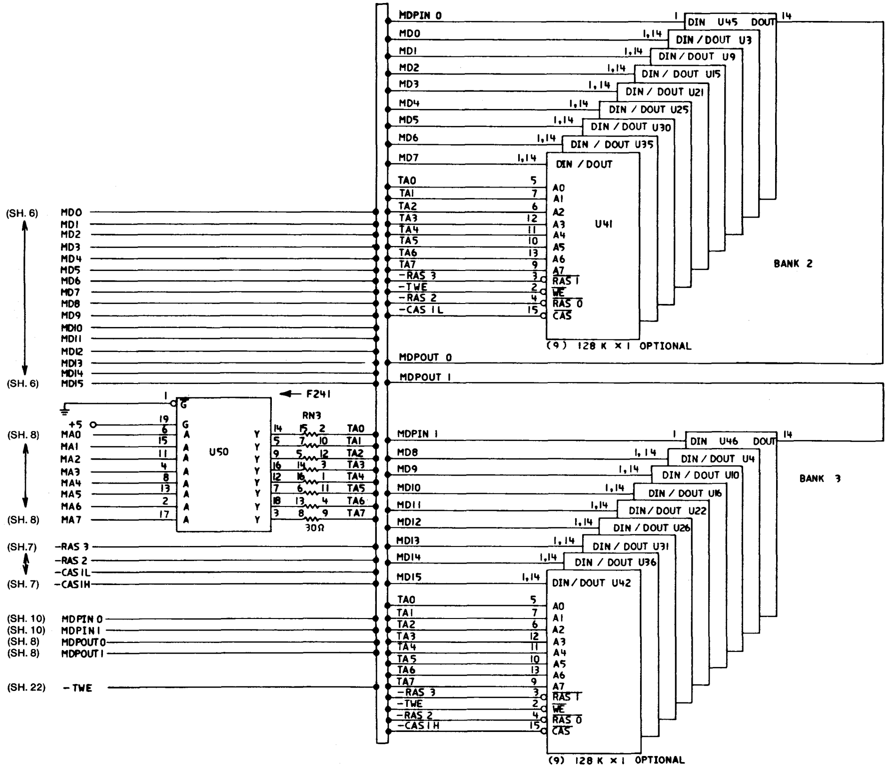
Type 1 512KB Planar (Sheet 7 of 22)



DECOUPLING CAPS :		
VOLTAGE TO GND	BANK : 0	1
+5	9-.10µF	9-.10µF
+5	2-10µF	2-10µF

10 µF: C1,2,52,53
.10 µF: C7,11,16,20,28, 32,38,42,47 C8,12,17,21,29, 33,39,43,48

Type 1 512KB Planar (Sheet 8 of 22)

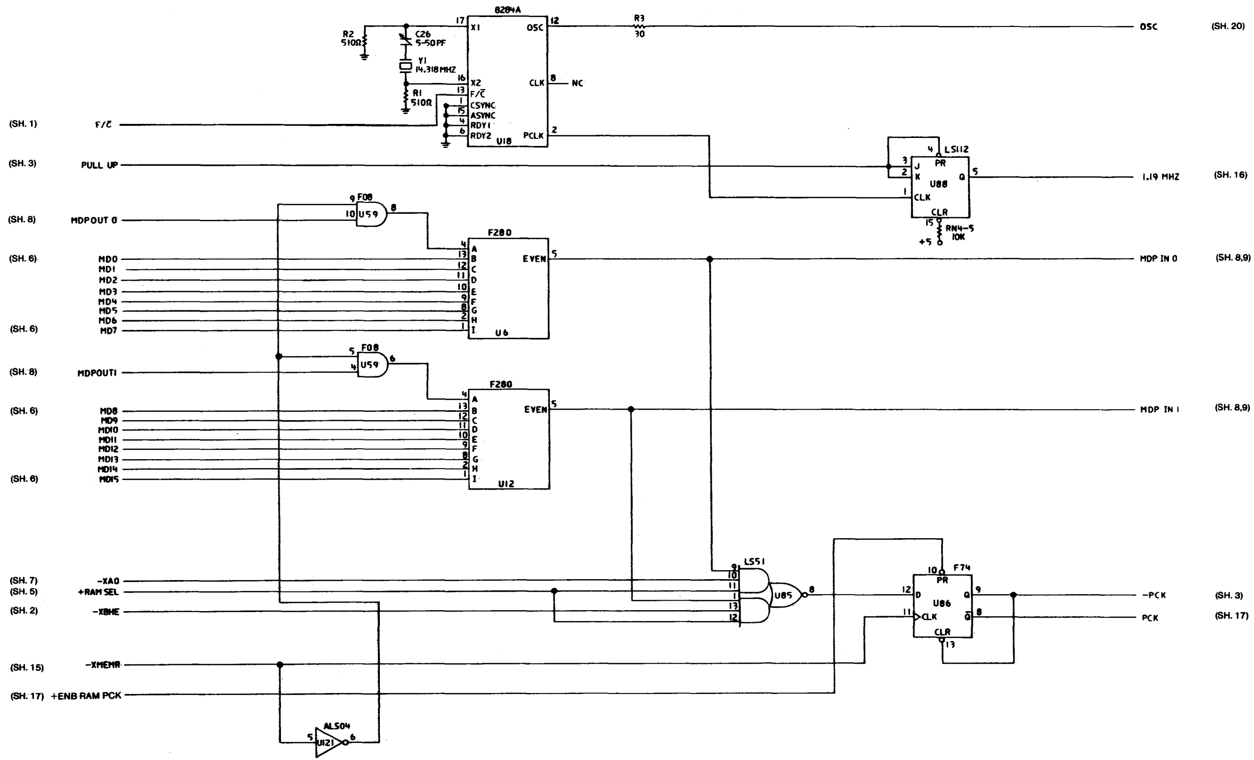


DECOUPLING CAP:		
VOLTAGE TO GND	BANK 2	BANK 3
+ 5	9 − .10µF	9 − .10µF
+ 5	2 − 10µF	2 − 10µF

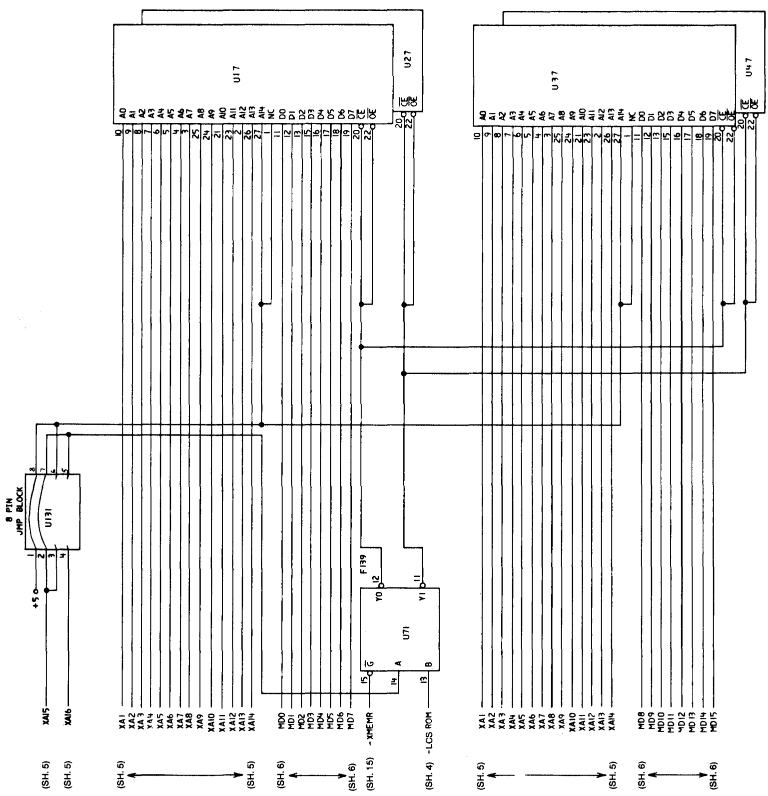
10µF: C3,4,54,55

.10µF: C9,13,18,22,30, 34,40,44,49 C10,14,19,23,31, 35,41,45,50

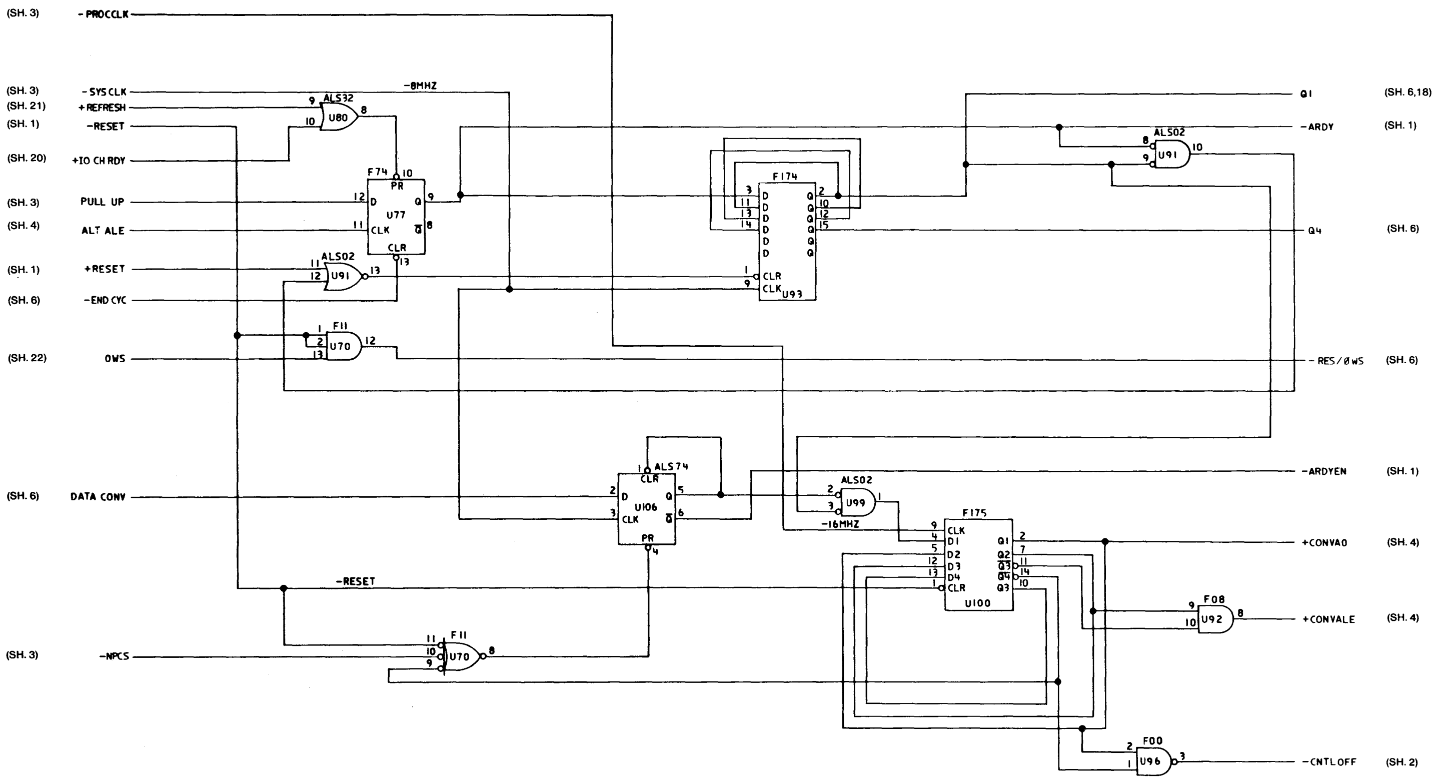
Type 1 512KB Planar (Sheet 9 of 22)



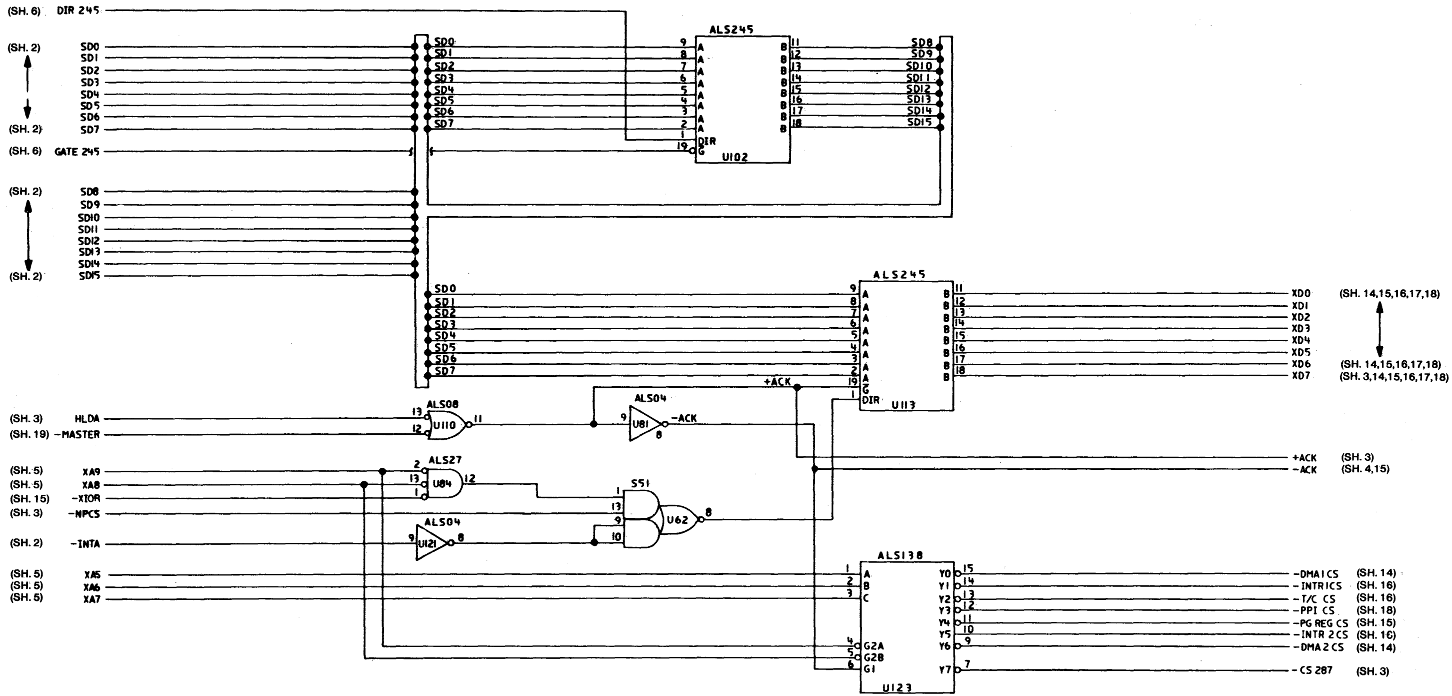
Type 1 512KB Planar (Sheet 10 of 22)



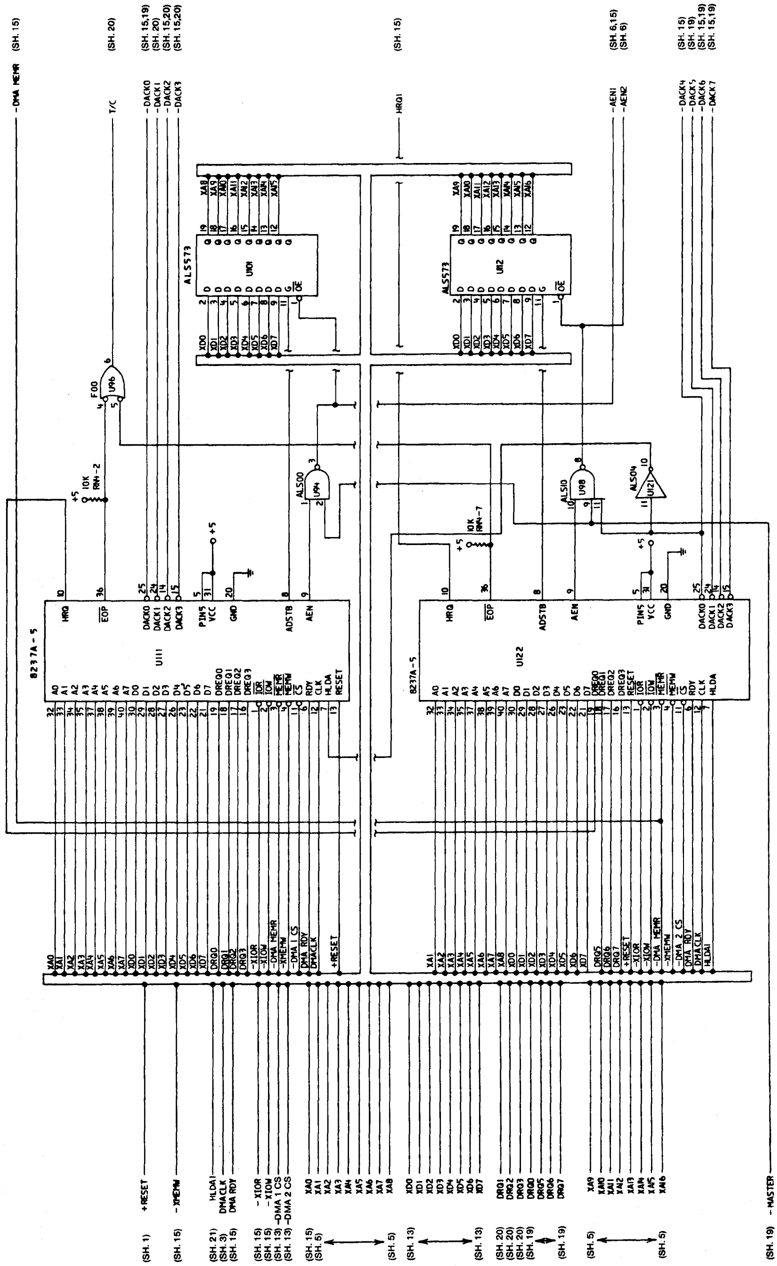
Type 1 512KB Planar (Sheet 11 of 22)



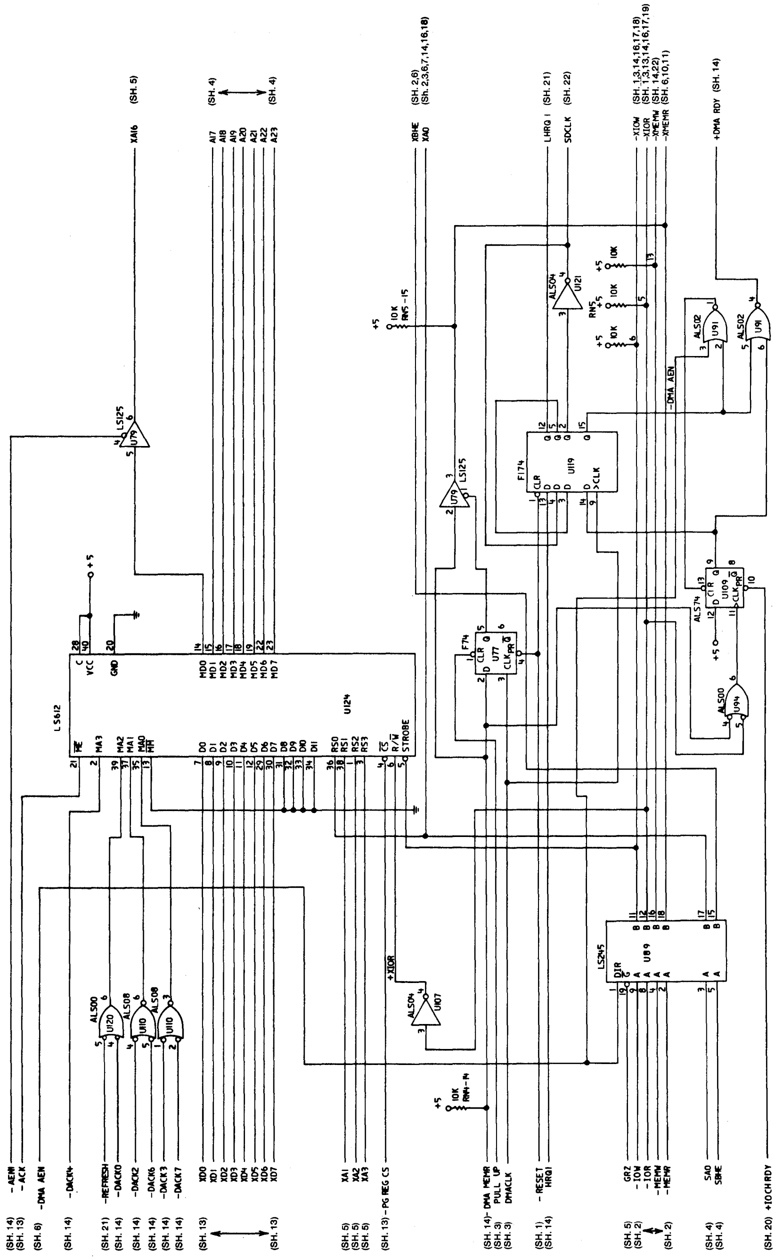
Type 1 512KB Planar (Sheet 12 of 22)



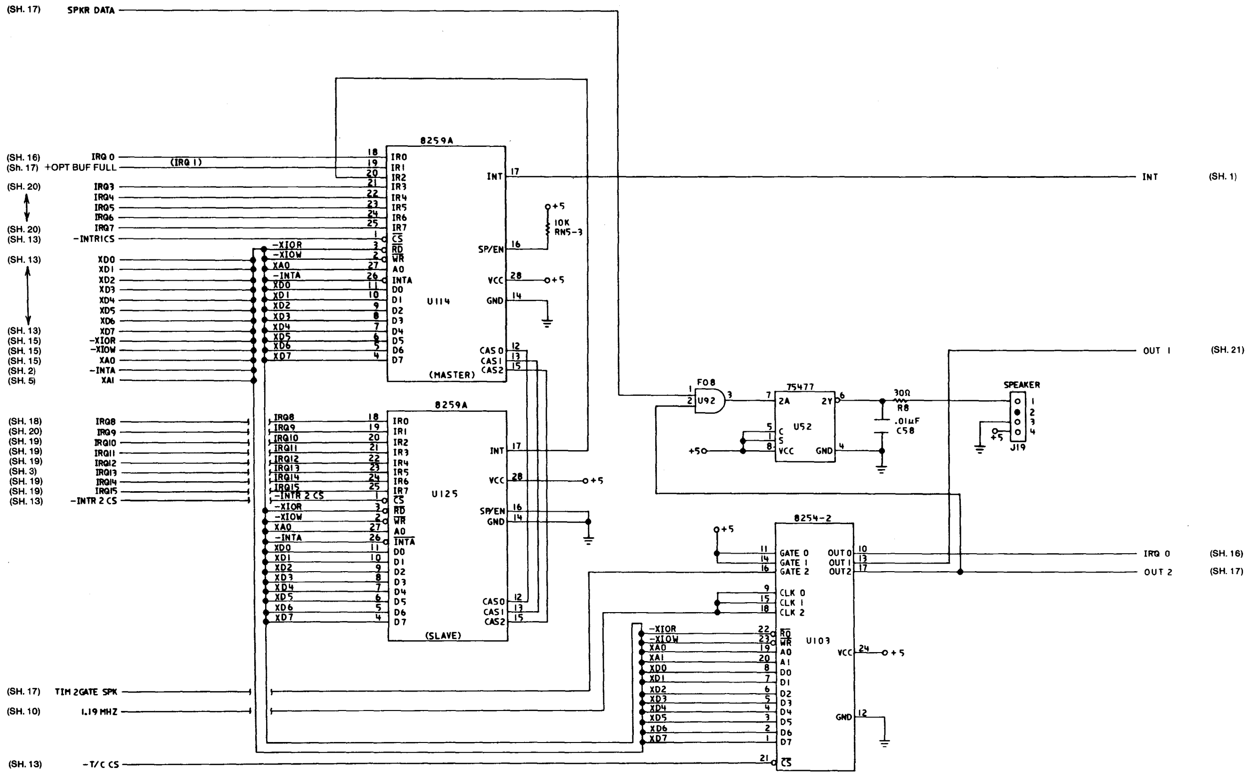
Type 1 512KB Planar (Sheet 13 of 22)



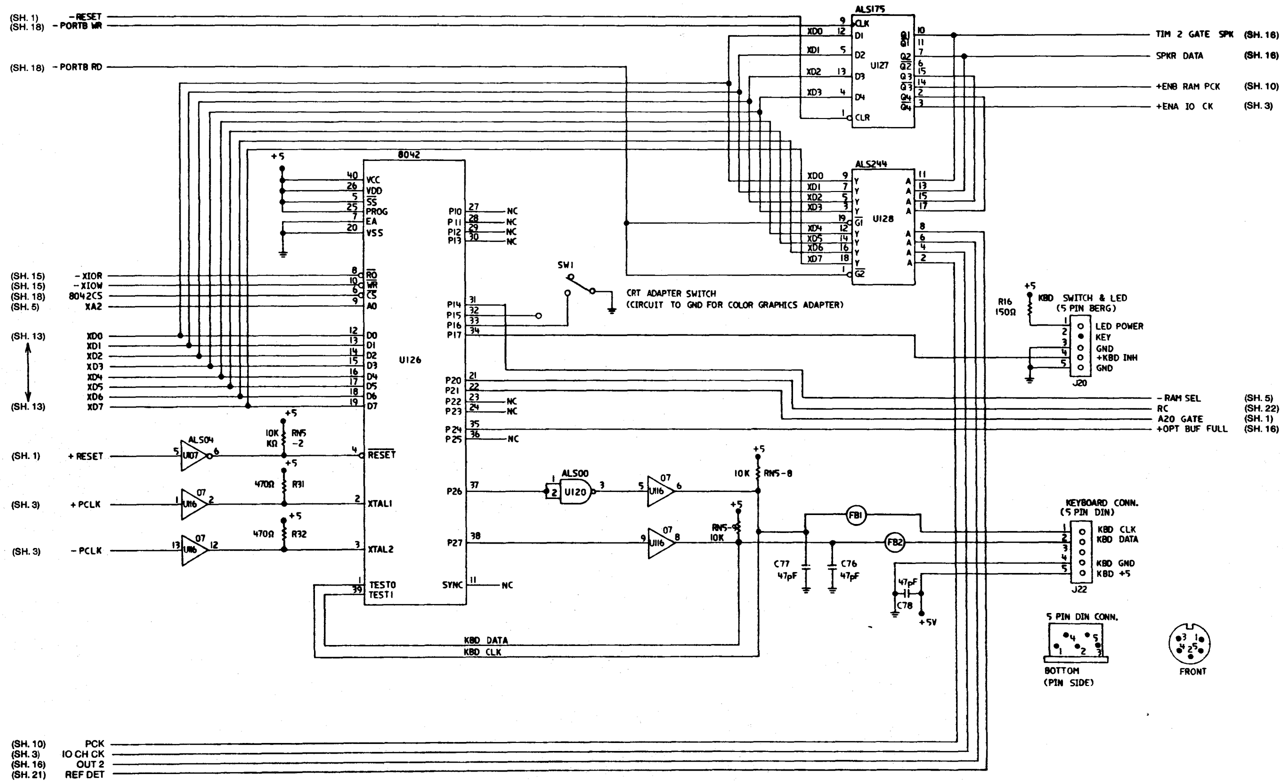
Type 1 512KB Planar (Sheet 14 of 22)



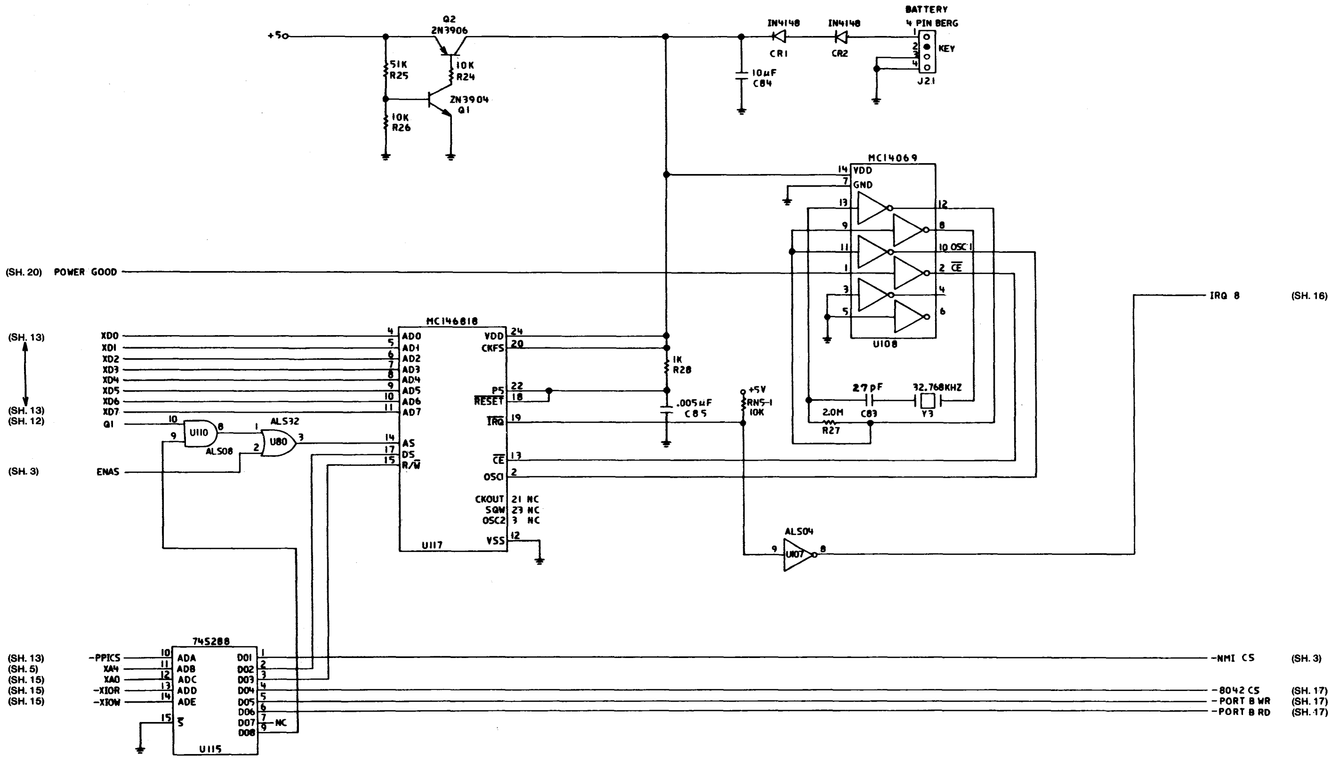
Type 1 512KB Planar (Sheet 15 of 22)



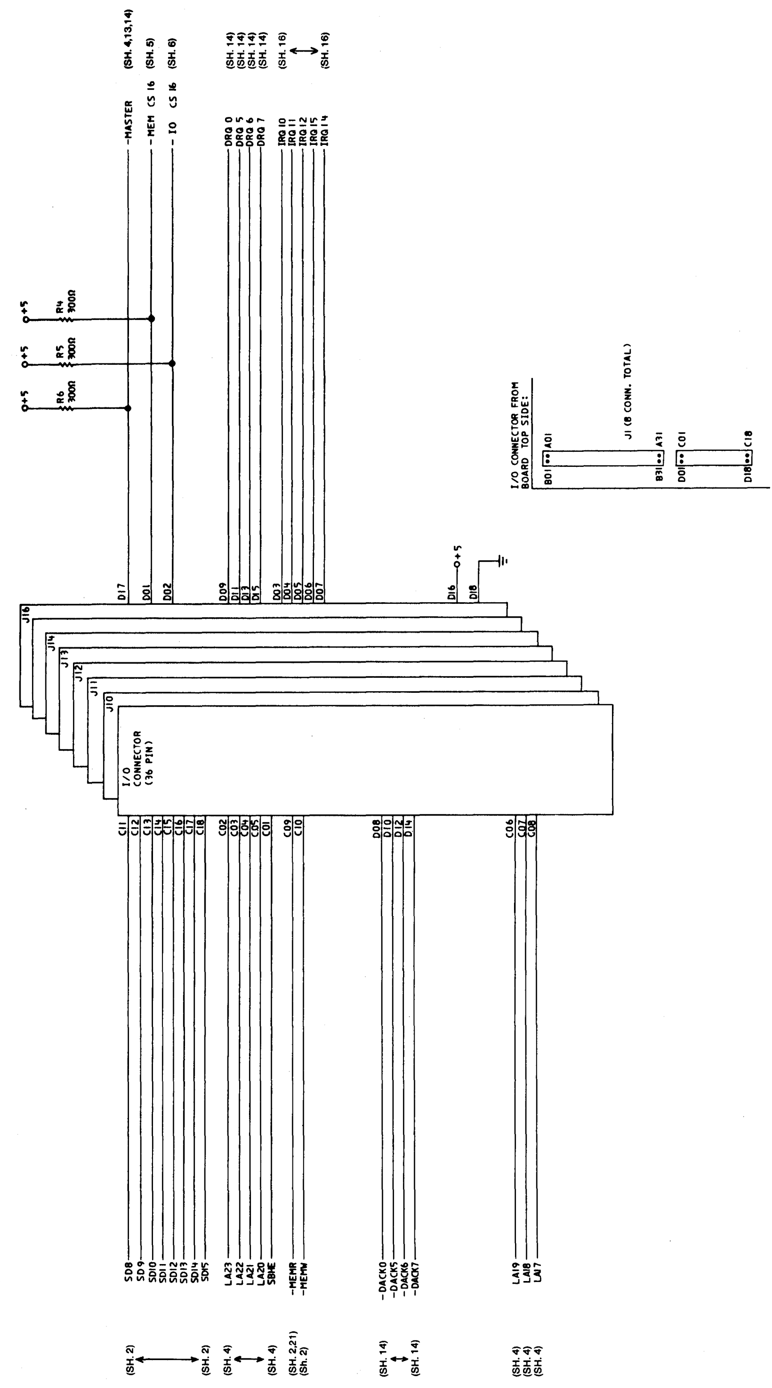
Type 1 512KB Planar (Sheet 16 of 22)



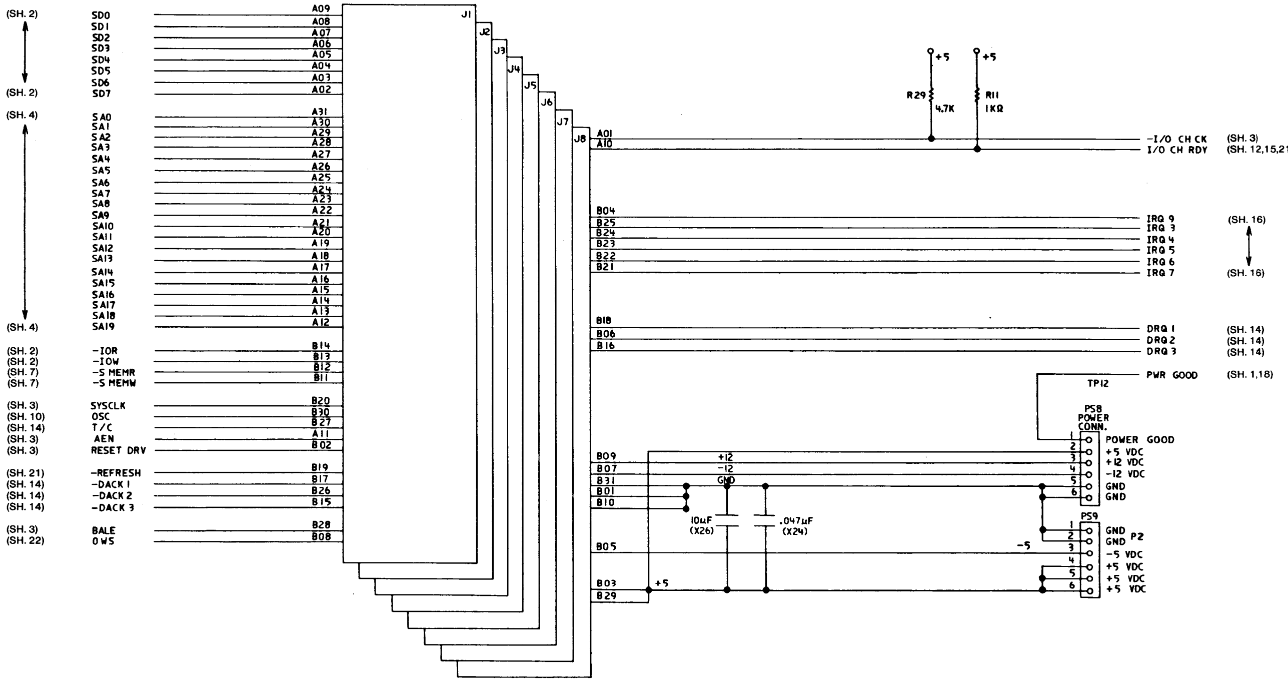
Type 1 512KB Planar (Sheet 17 of 22)



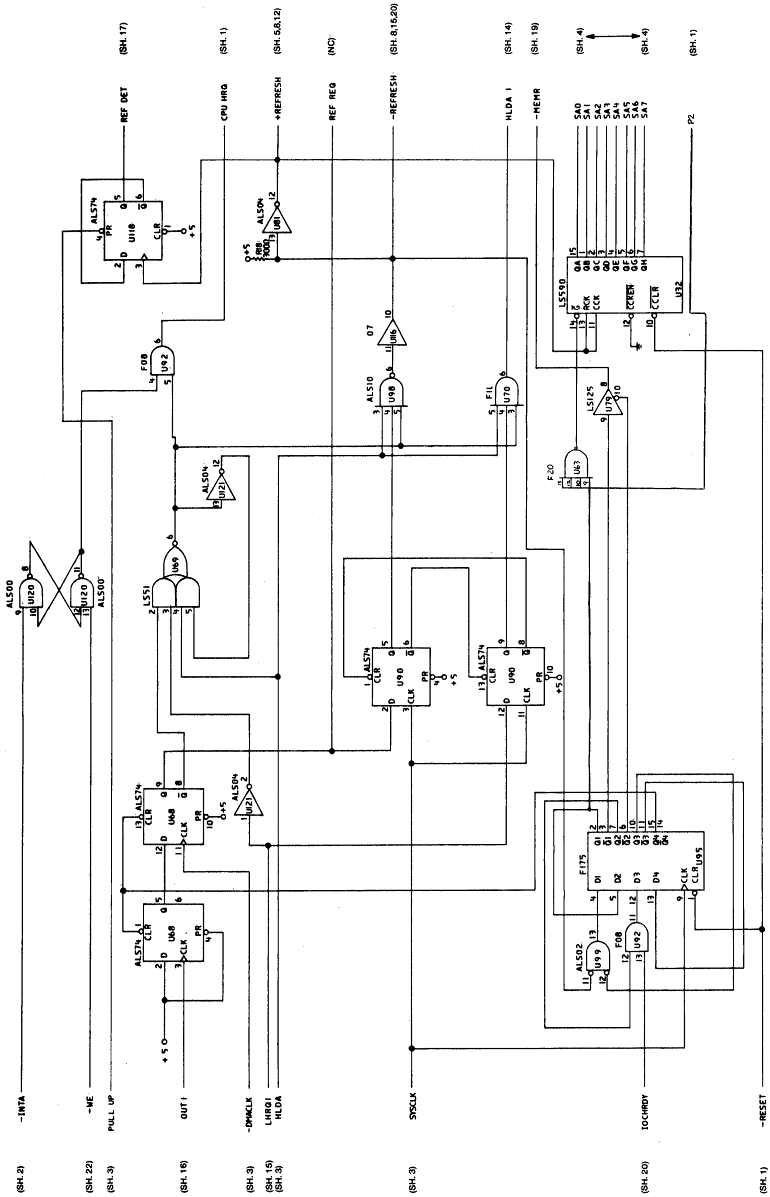
Type 1 512KB Planar (Sheet 18 of 22)



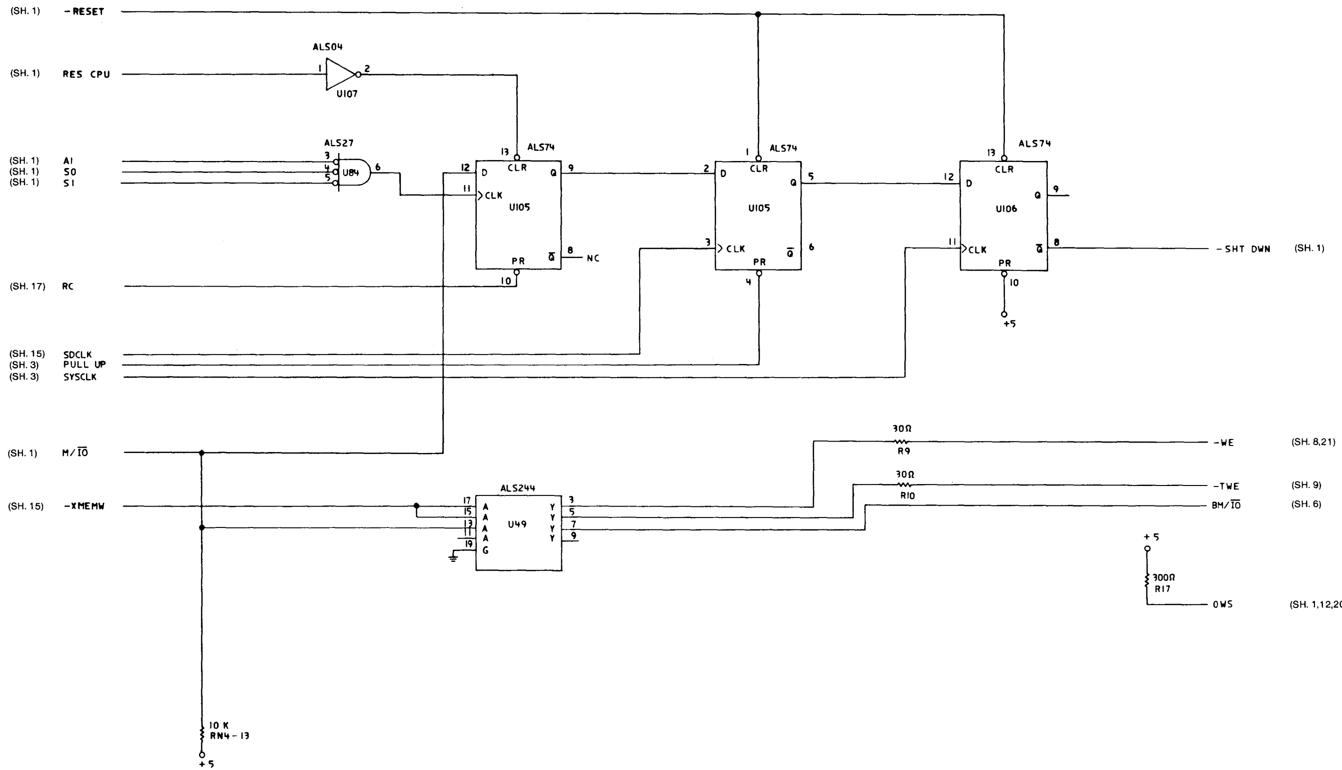
Type 1 512KB Planar (Sheet 19 of 22)



Type 1 512KB Planar (Sheet 20 of 22)

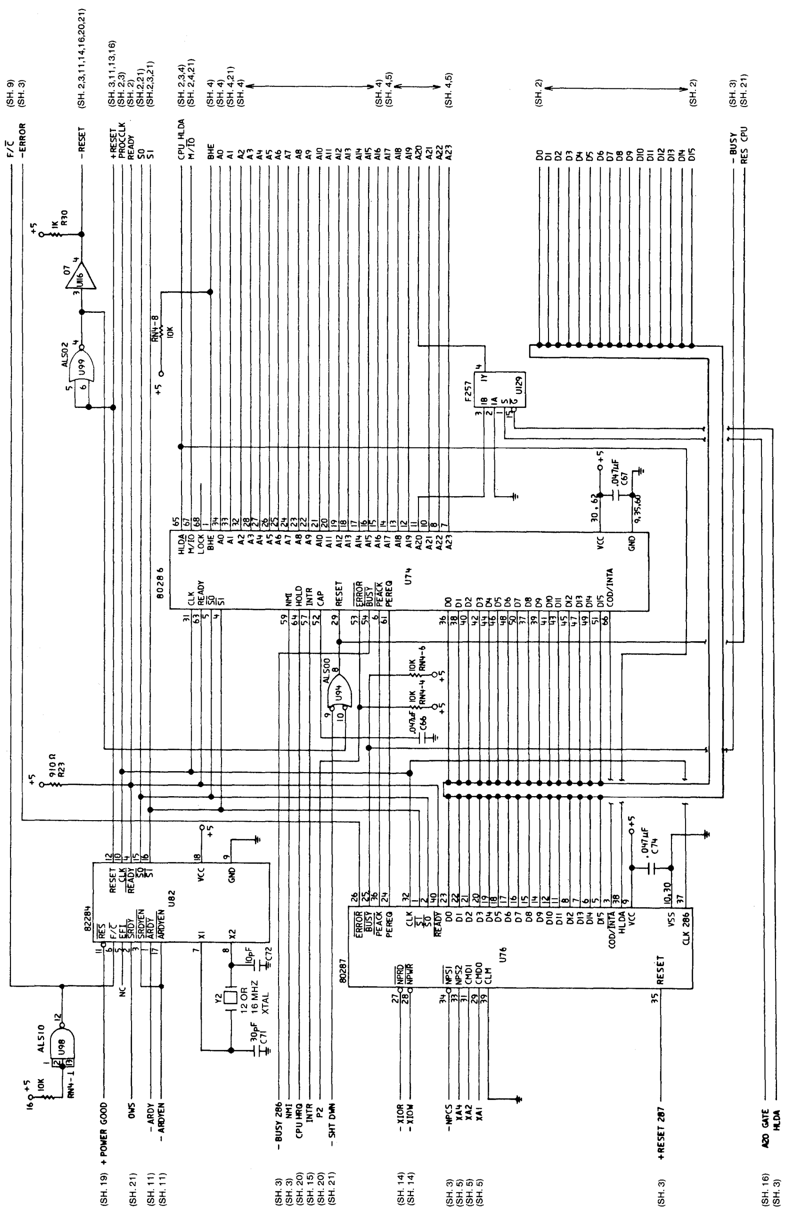


Type 1 512KB Planar (Sheet 21 of 22)

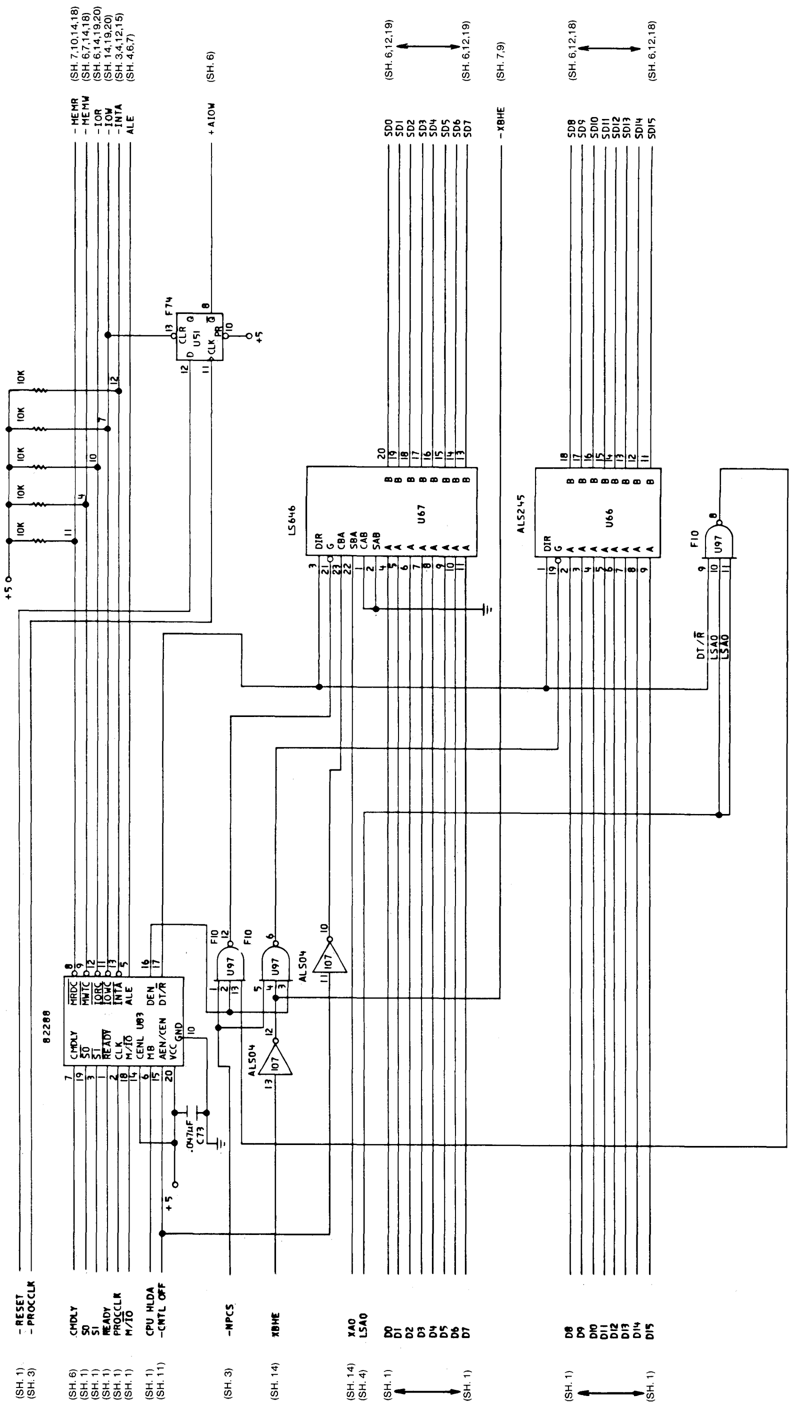


Type 1 512KB Planar (Sheet 22 of 22)

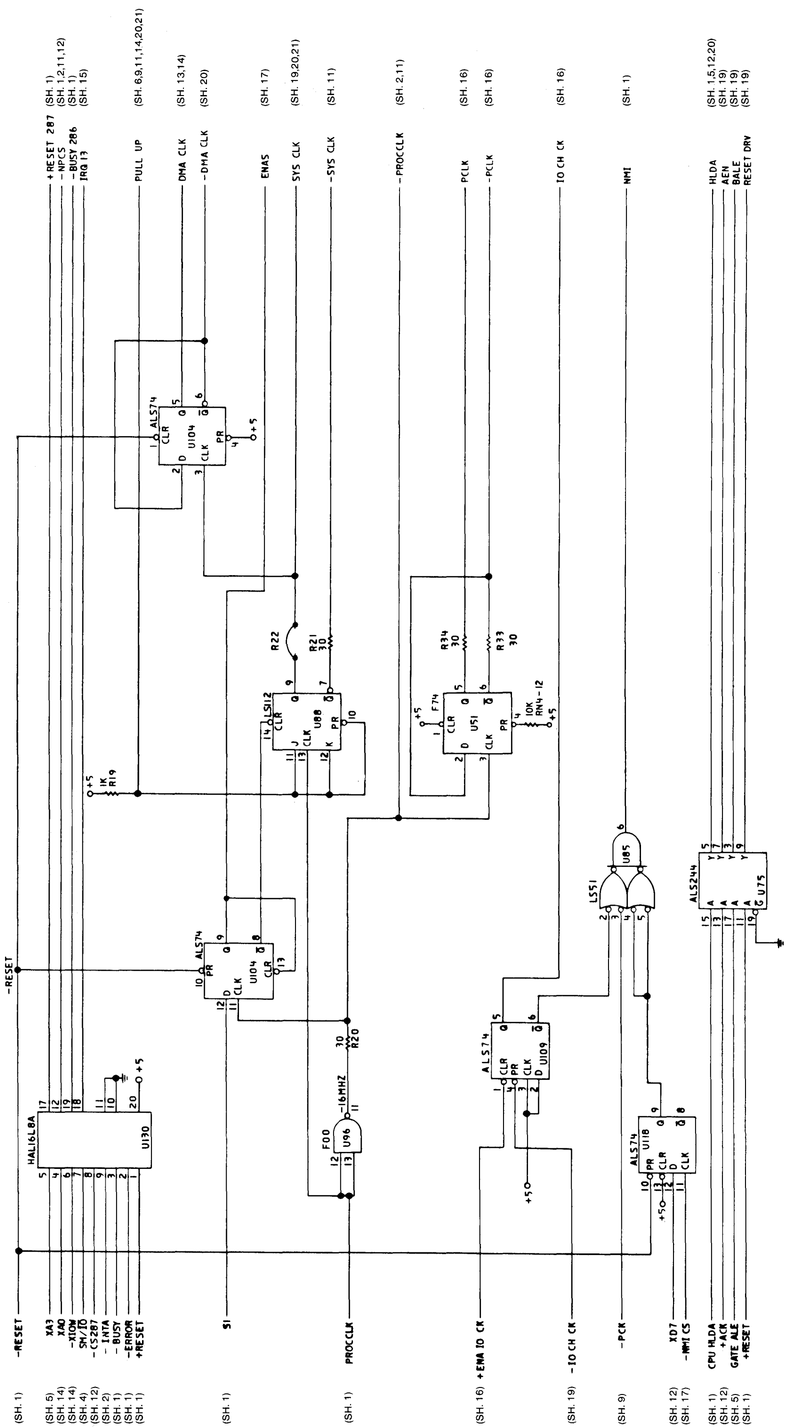
Logic Diagrams - Type 2



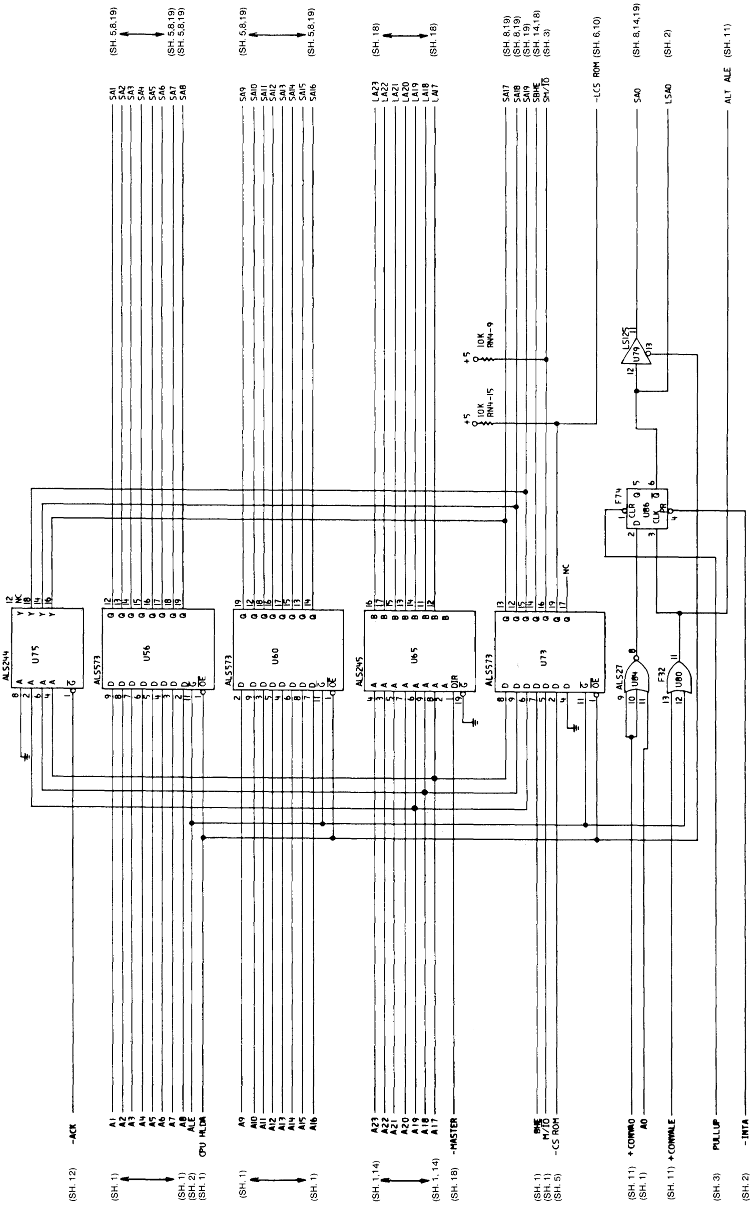
Type 2 512KB System Board (Sheet 1 of 21)



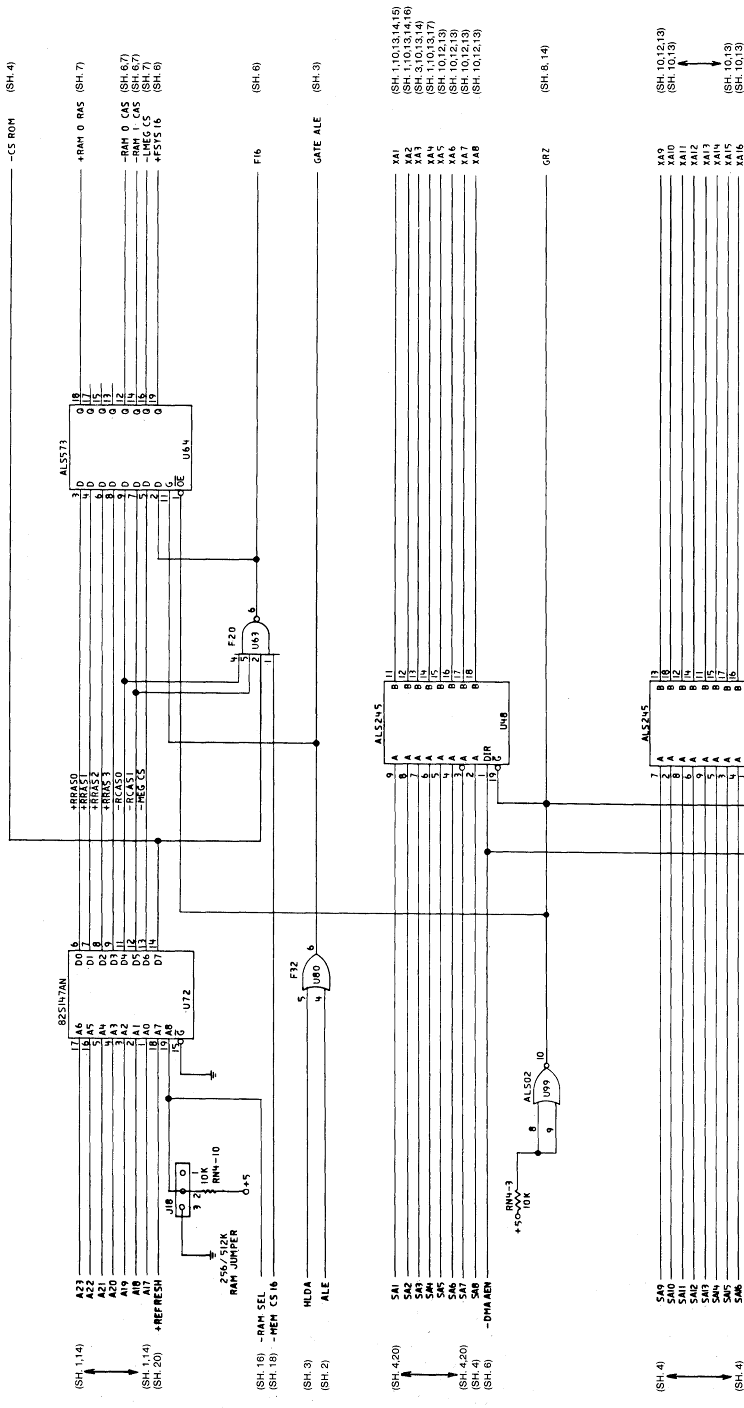
Type 2 512KB System Board (Sheet 2 of 21)



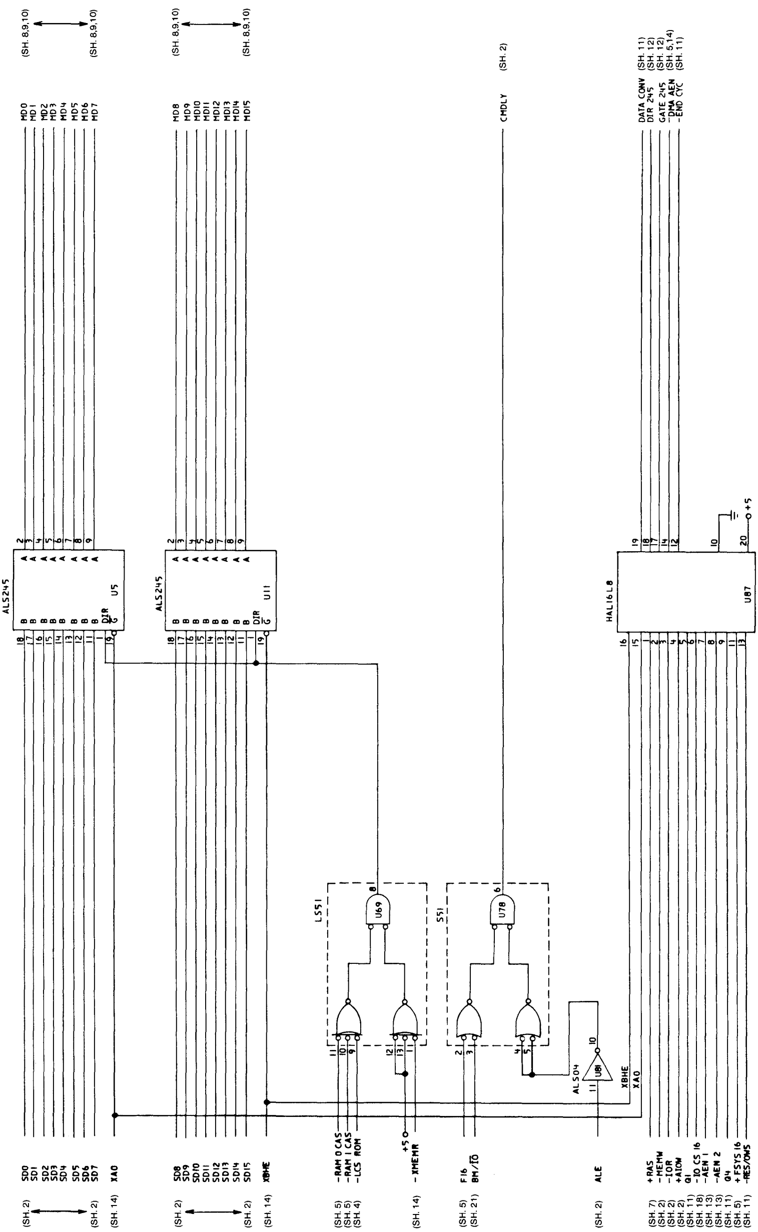
Type 2 512KB System Board (Sheet 3 of 21)



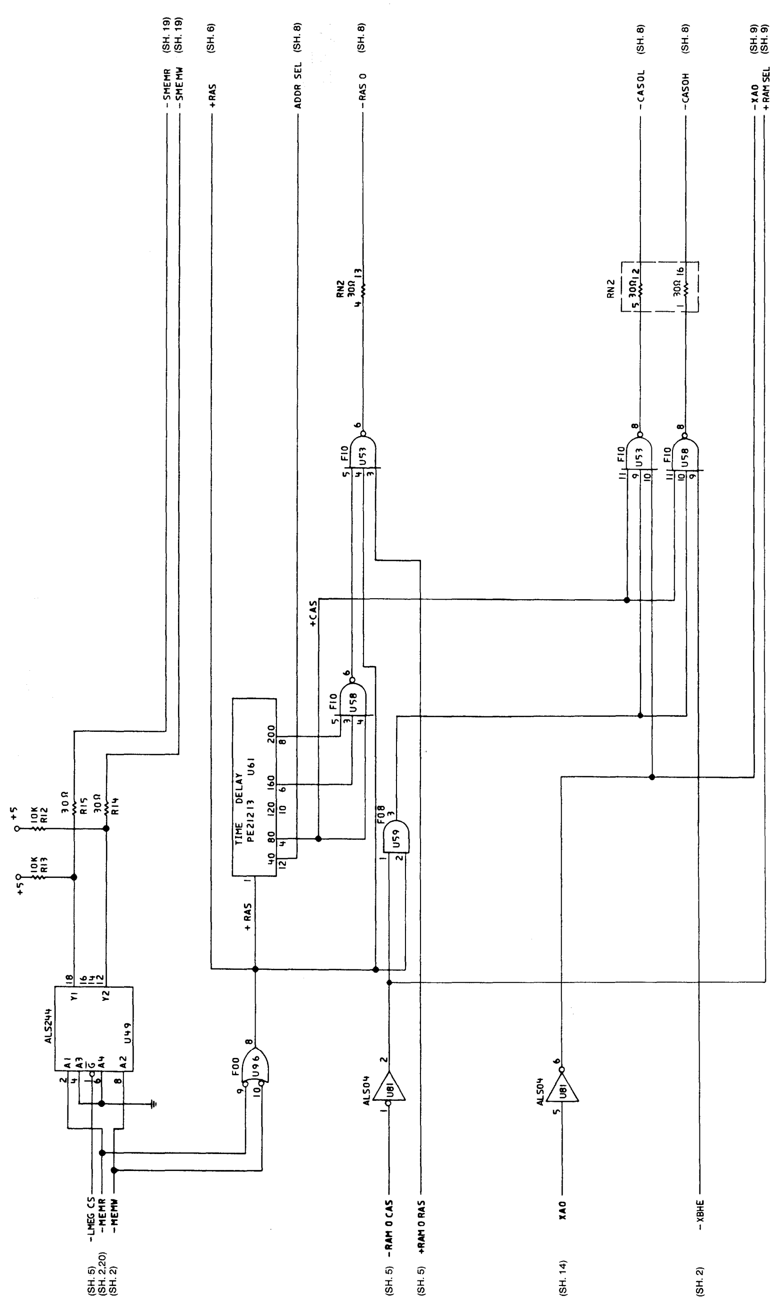
Type 2 512KB System Board (Sheet 4 of 21)



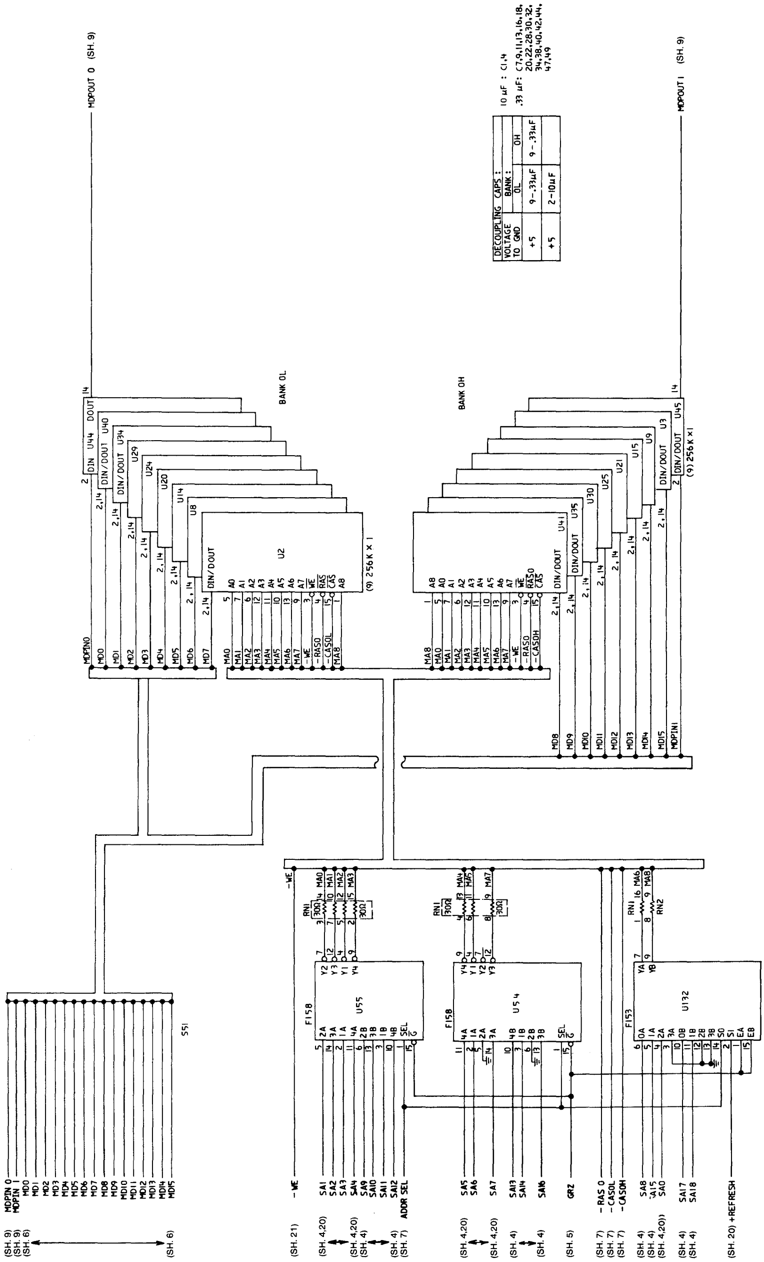
Type 2 512KB System Board (Sheet 5 of 21)



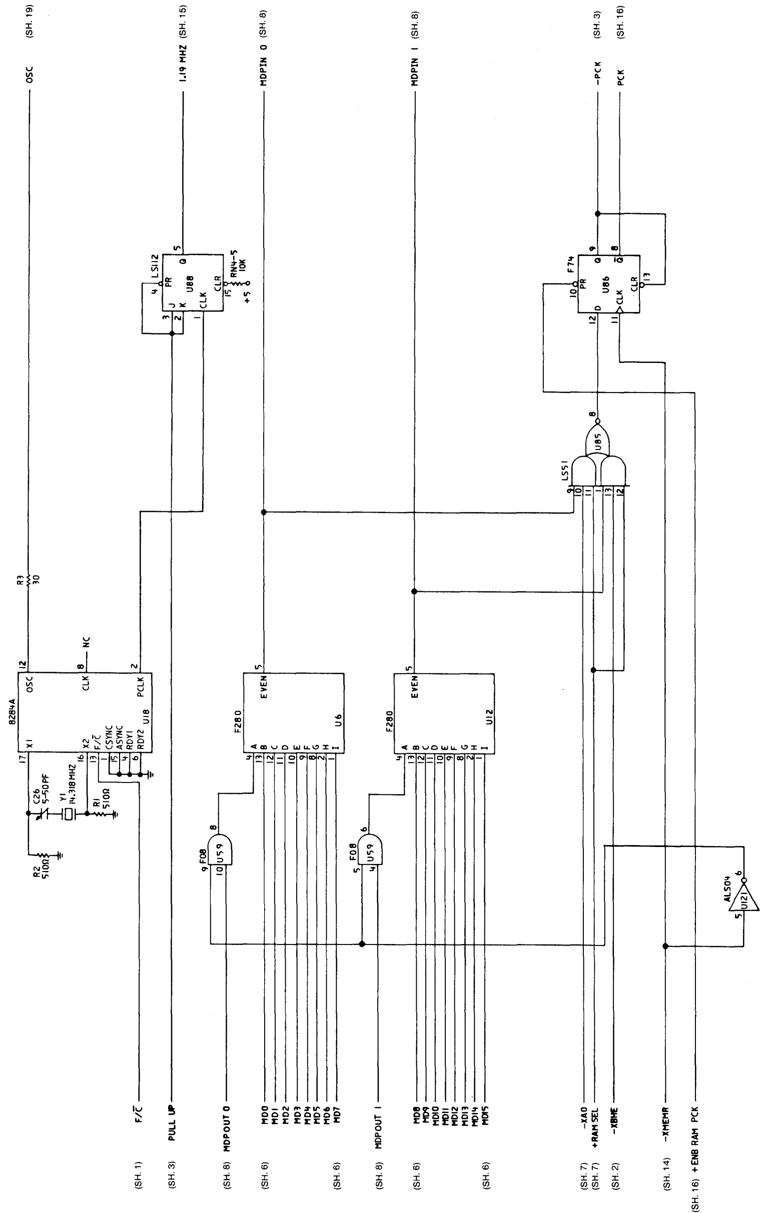
Type 2 512KB System Board (Sheet 6 of 21)



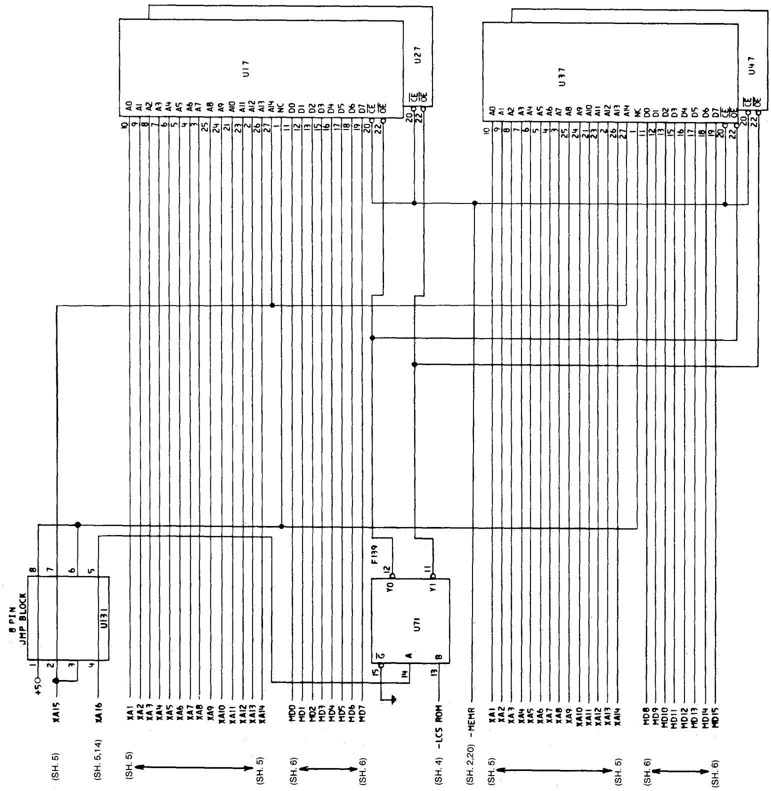
Type 2 512KB System Board (Sheet 7 of 21)



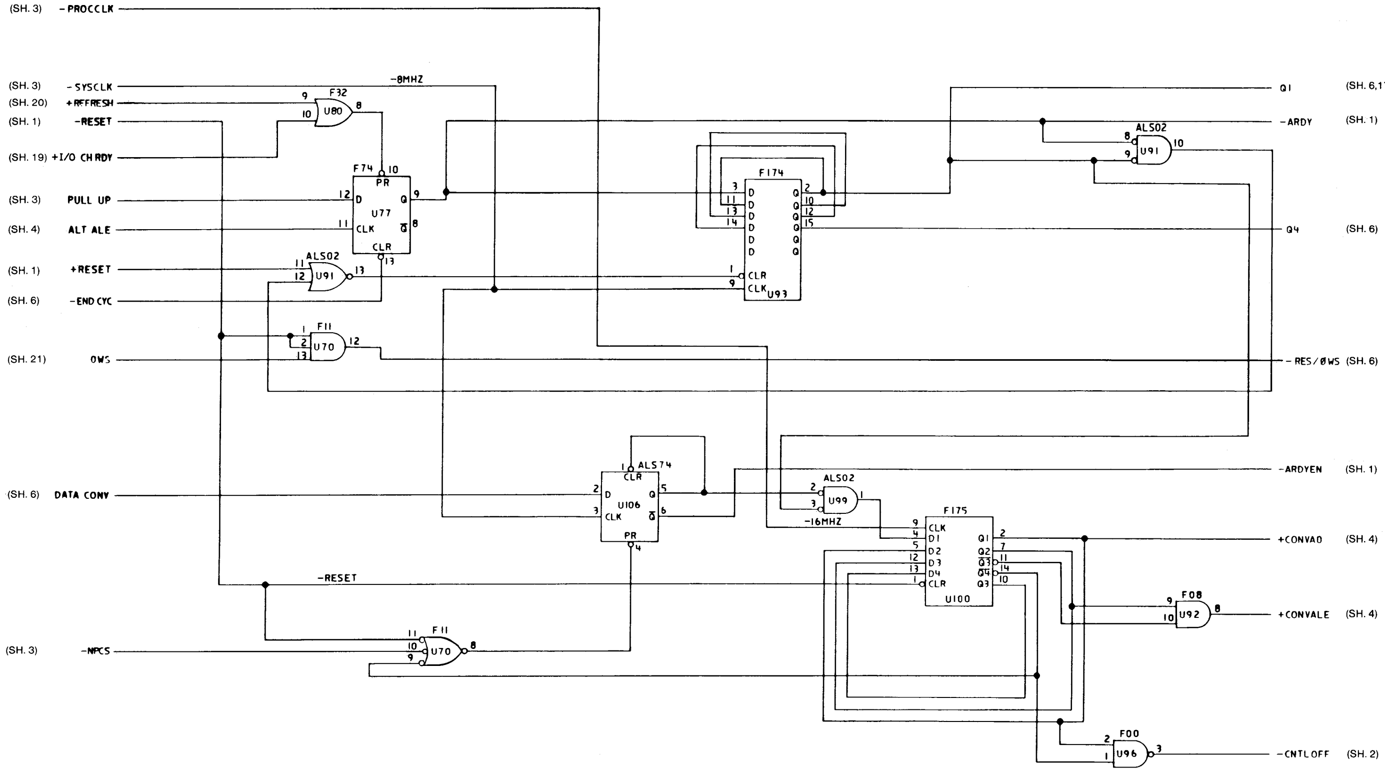
Type 2 512KB System Board (Sheet 8 of 21)



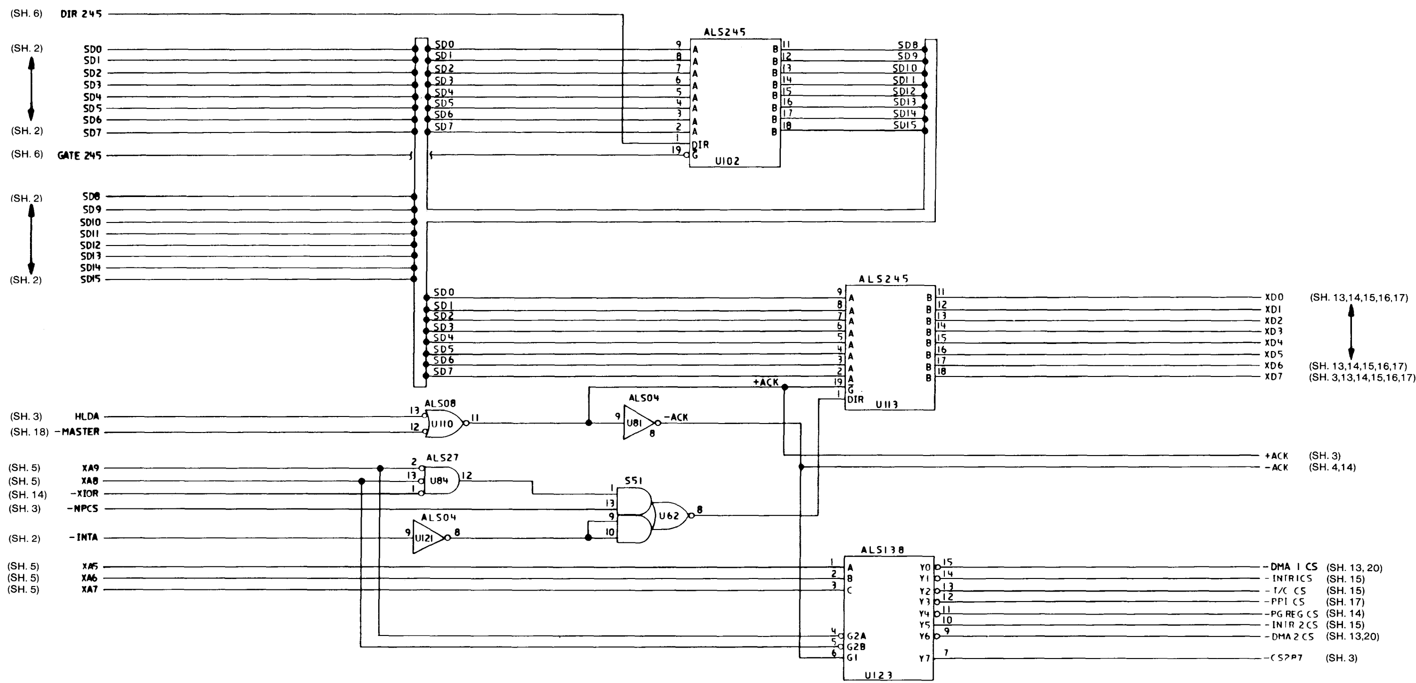
Type 2 512KB System Board (Sheet 9 of 21)



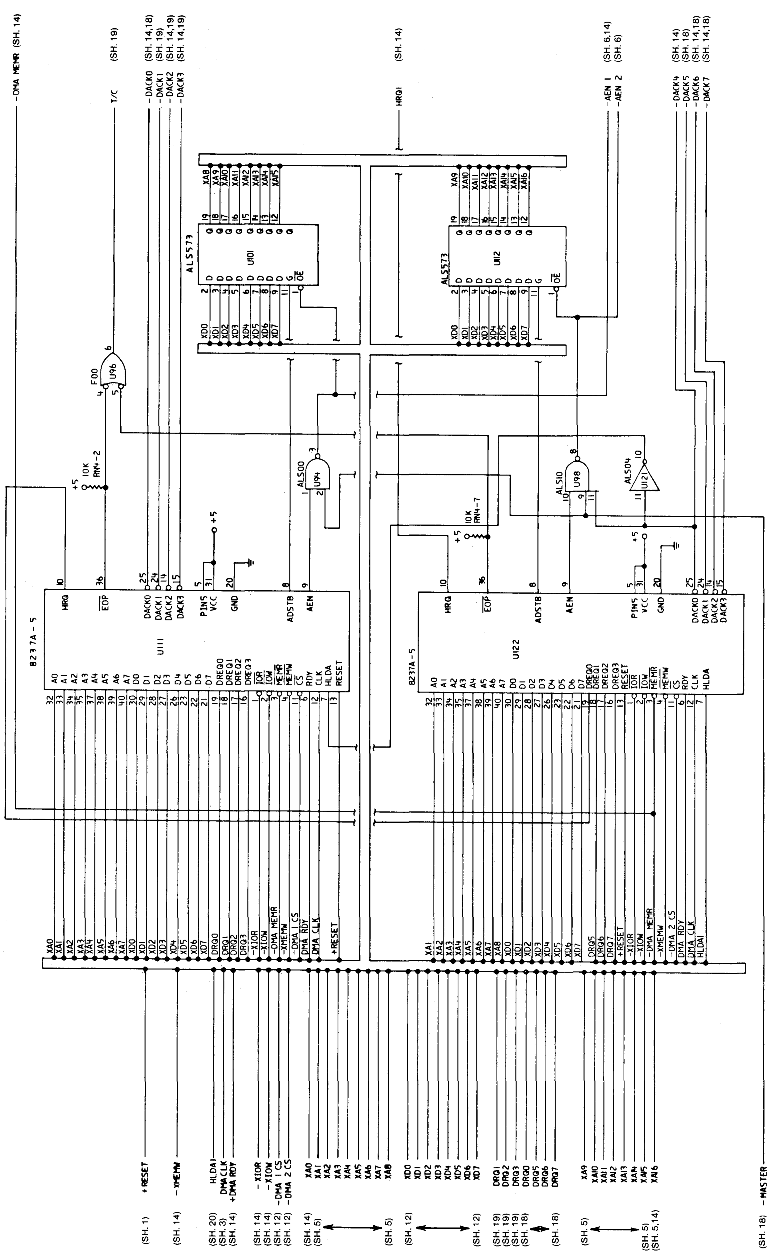
Type 2 512KB System Board (Sheet 10 of 21)



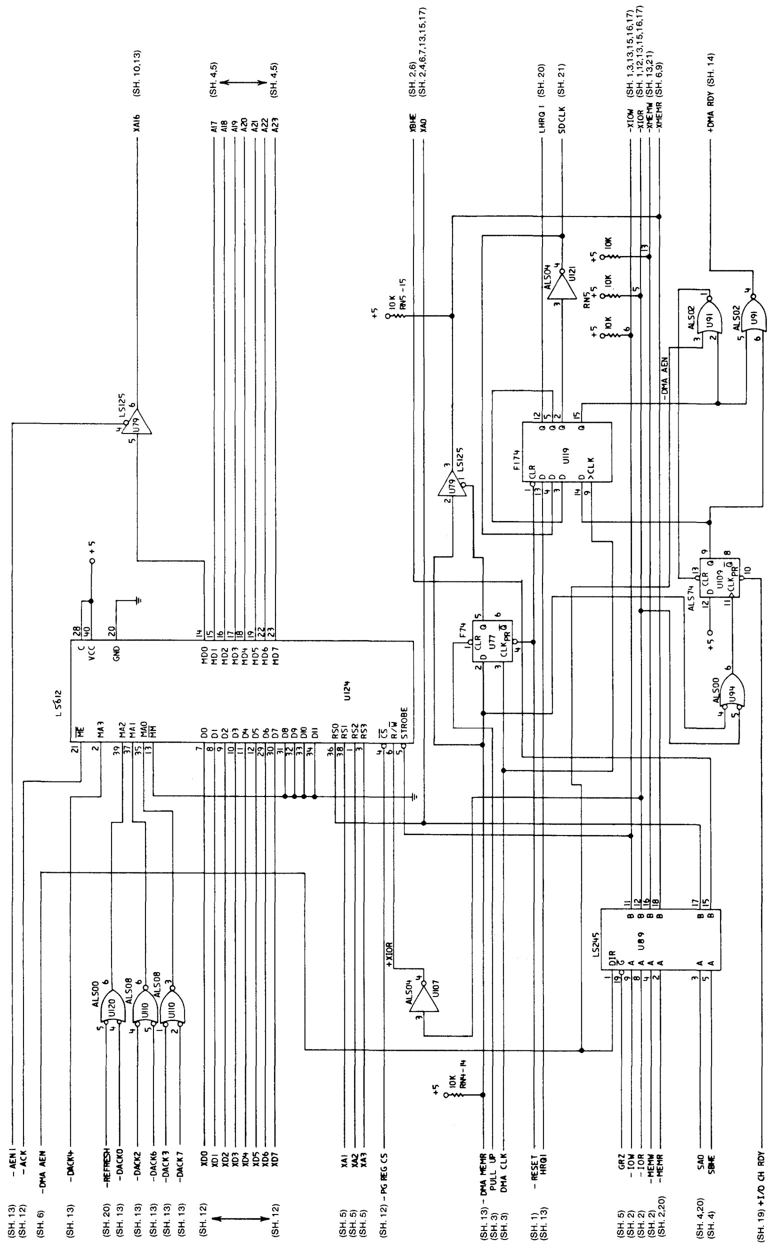
Type 2 512KB System Board (Sheet 11 of 21)



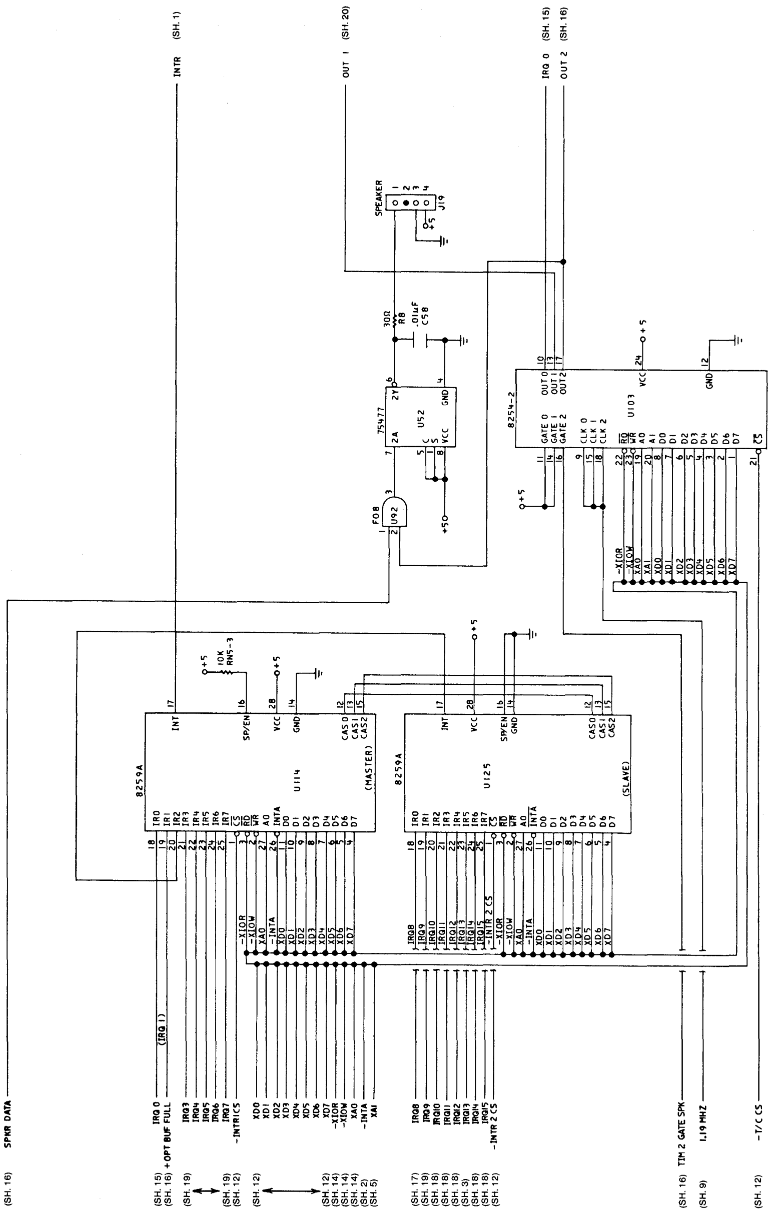
Type 2 512KB System Board (Sheet 12 of 21)



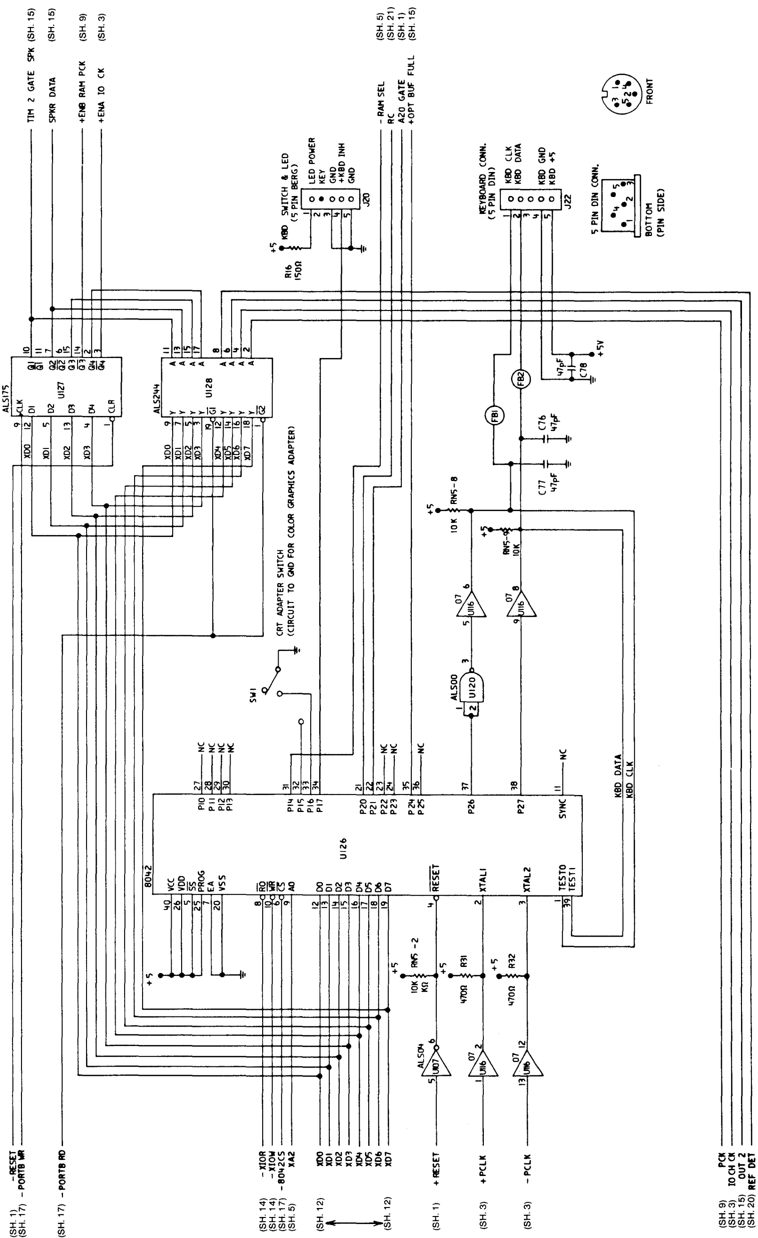
Type 2 512KB System Board (Sheet 13 of 21)



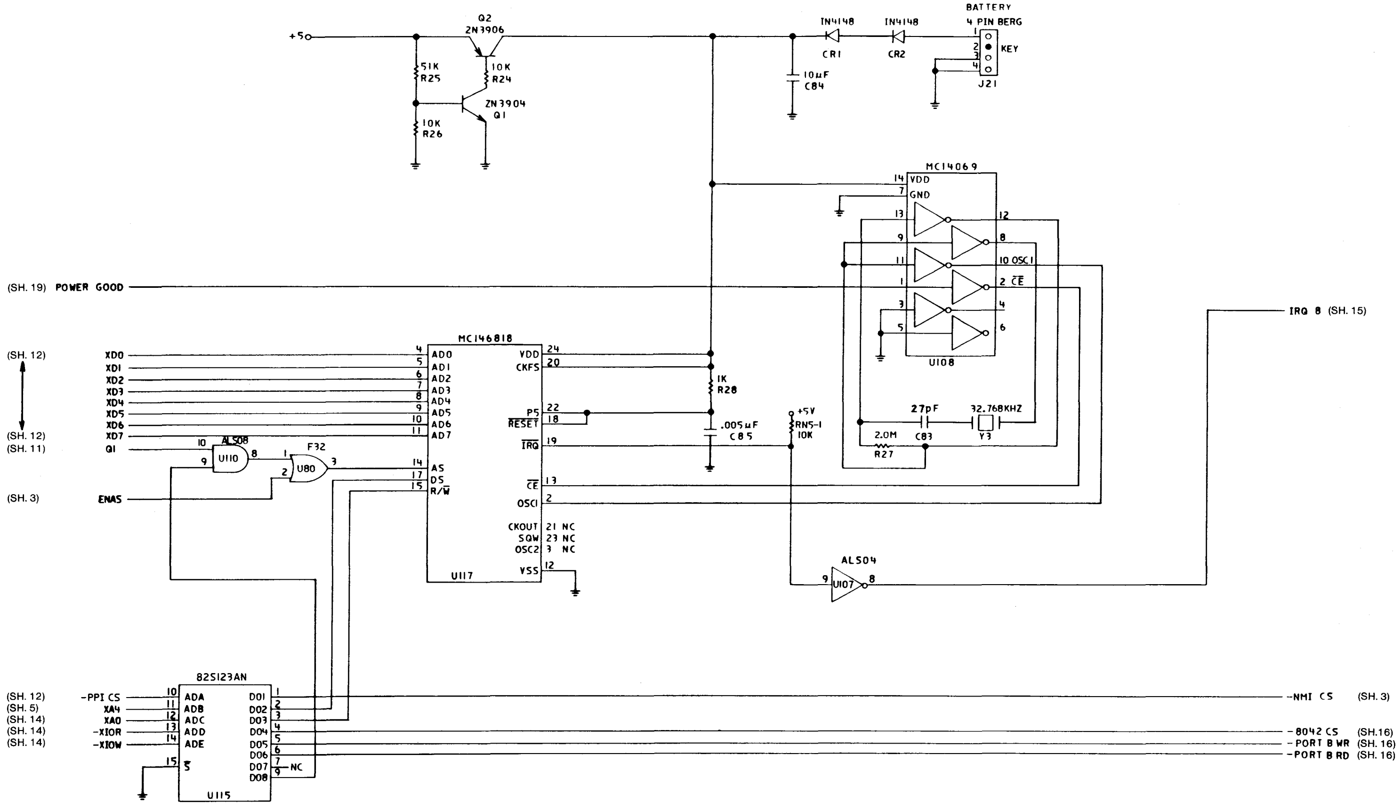
Type 2 512KB System Board (Sheet 14 of 21)



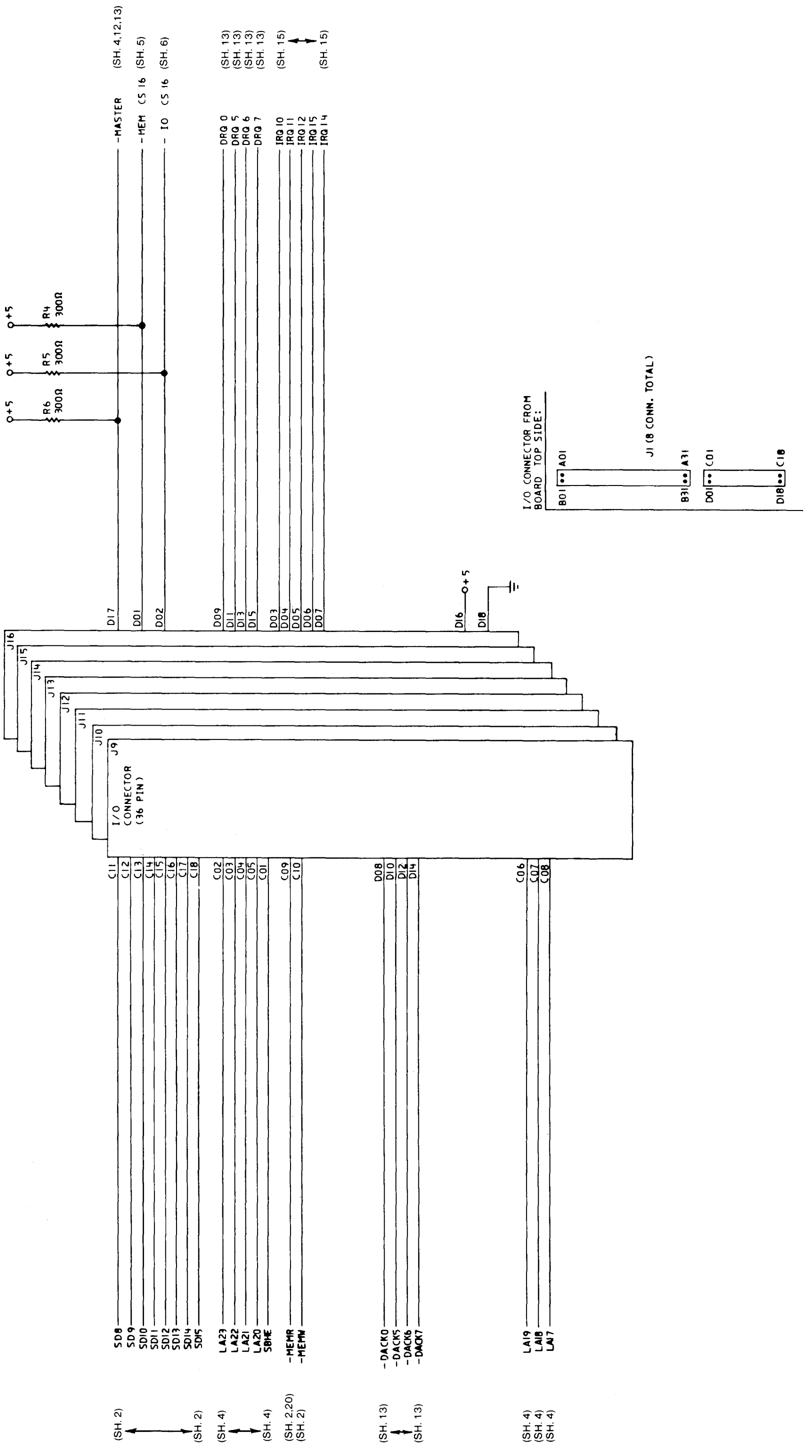
Type 2 512KB System Board (Sheet 15 of 21)



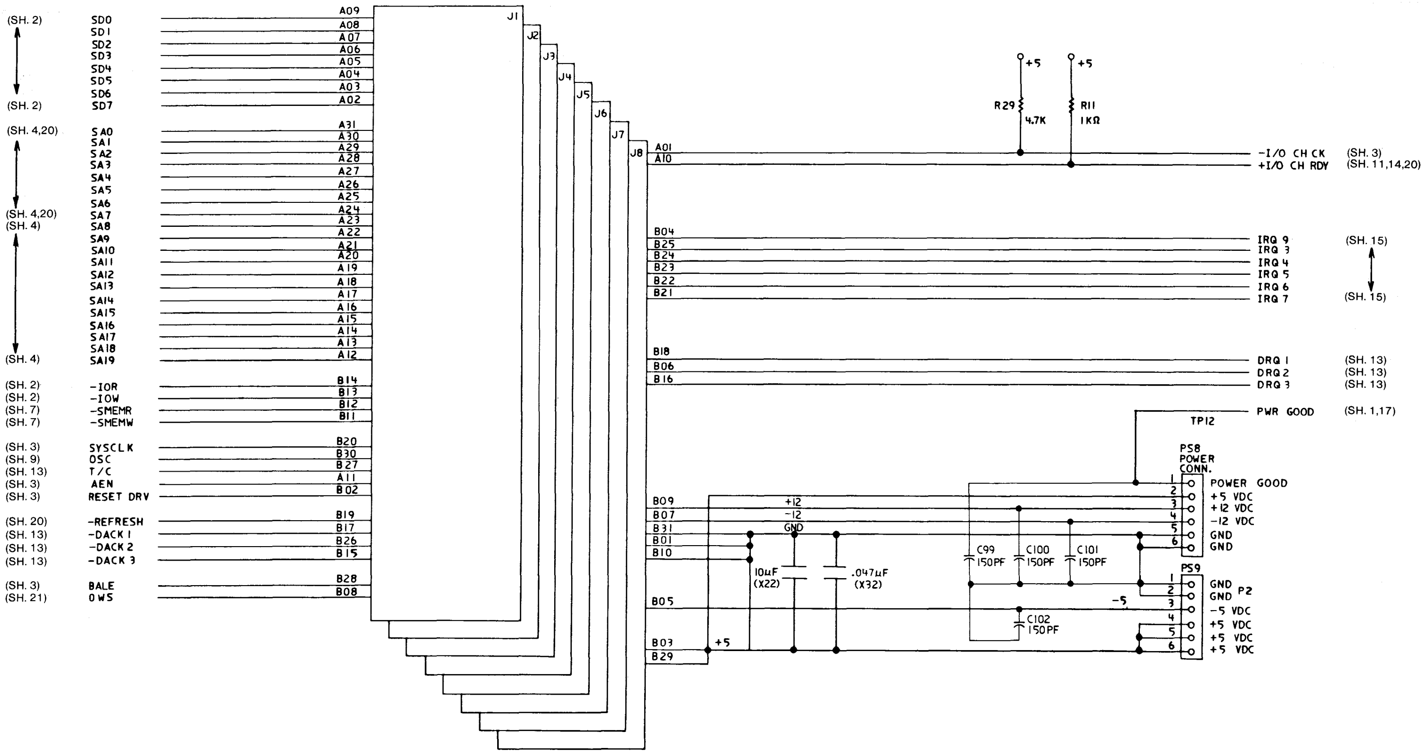
Type 2 512KB System Board (Sheet 16 of 21)



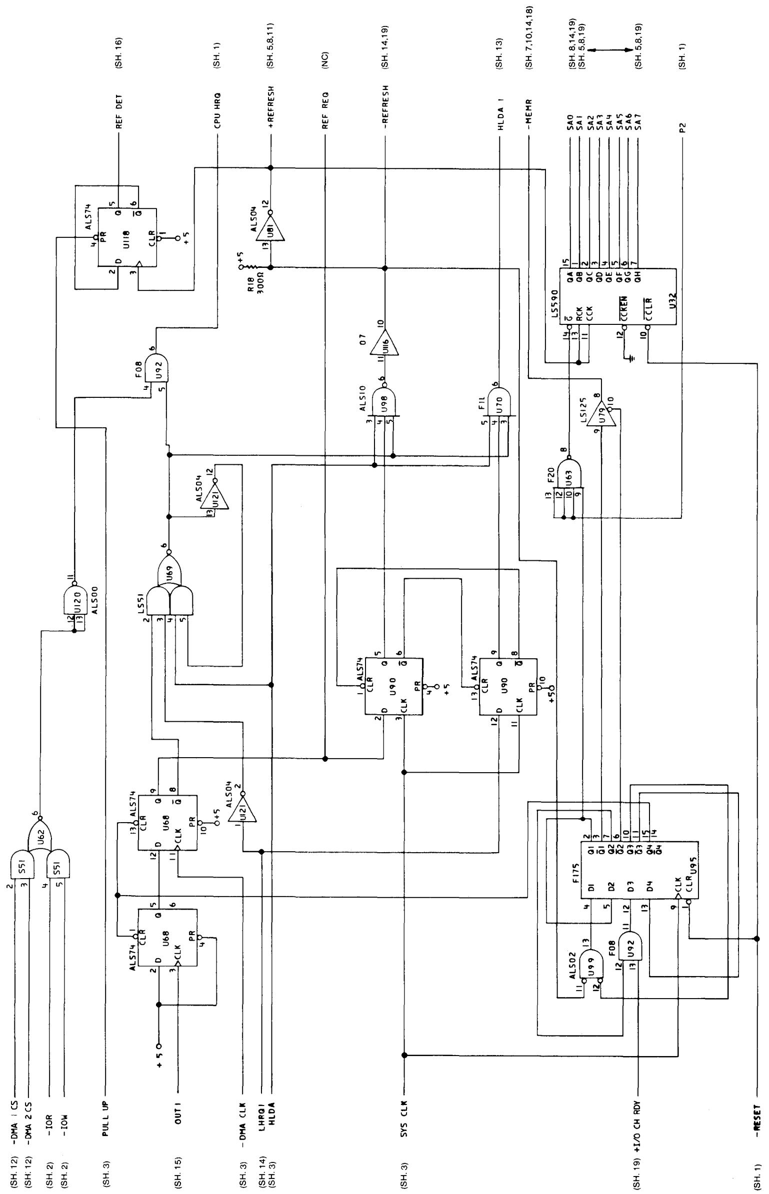
Type 2 512KB System Board (Sheet 17 of 21)



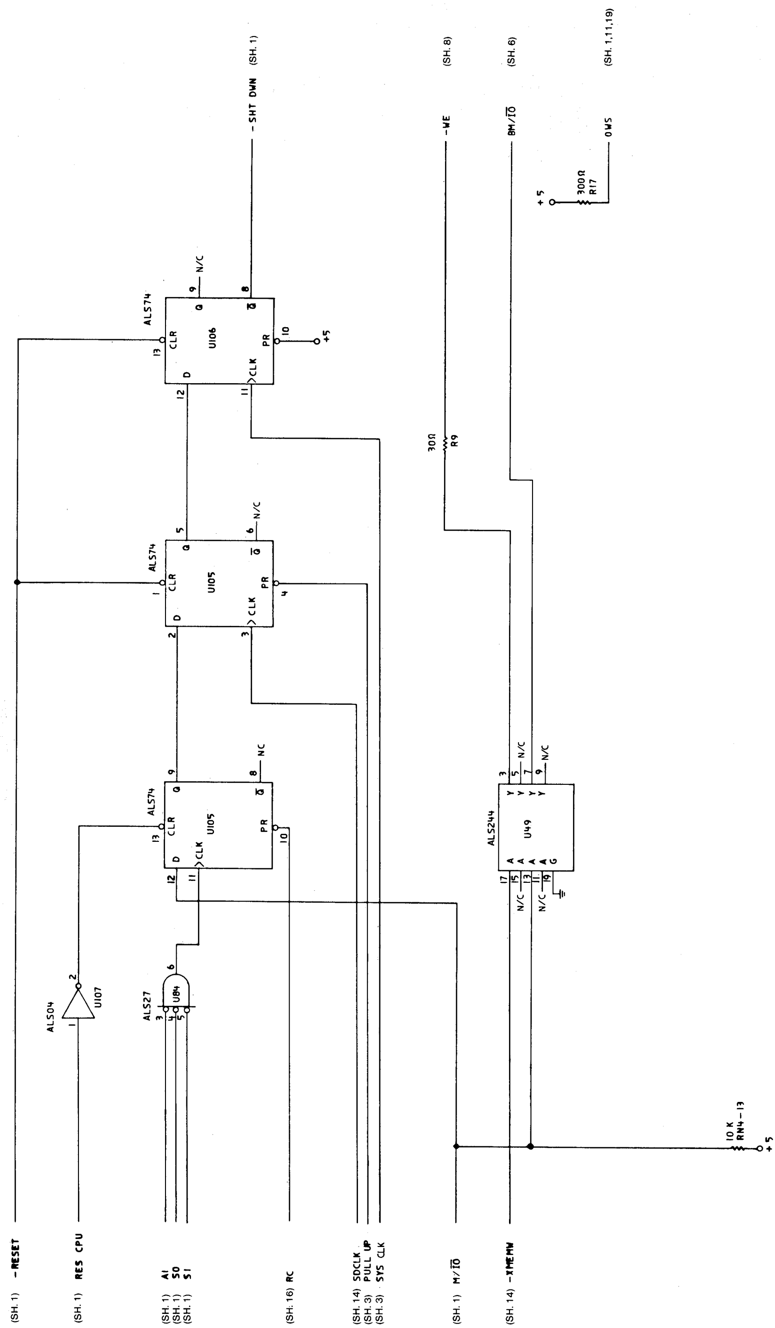
Type 2 512KB System Board (Sheet 18 of 21)



Type 2 512KB System Board (Sheet 19 of 21)



Type 2 512KB System Board (Sheet 20 of 21)



Type 2 512KB System Board (Sheet 21 of 21)

SECTION 2. COPROCESSOR

Contents

SECTION 2

Notes:

Description

The IBM Personal Computer AT Math Coprocessor enables the IBM PERSONAL COMPUTER AT to perform high-speed arithmetic, logarithmic functions, and trigonometric operations.

SECTION 2

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 provides 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	$-32{,}768 \le X \le +32{,}767$
Short Integer	32	9	$-2\times10^{9} \le X \le +2\times10^{9}$
Long Integer	64	18	$-9\times10^{18} \le X \le +9\times10^{18}$
Packed Decimal	80	18	$-9..99 \le X \le +9..99$ (18 digits)
Short Real *	32	6-7	$8.43\times10^{-37} \le \|X\| \le 3.37\times10^{38}$
Long Real *	64	15-16	$4.19\times10^{-307} \le \|X\| \le 1.67\times10^{308}$
Temporary Real	80	19	$3.4\times10^{-4932} \le \|X\| \le 1.2\times10^{4932}$

Data Types

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

Hardware Interface

The coprocessor uses the same clock generator as the microprocessor. It works at one-third the frequency of the system microprocessor. 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 AT. 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

Contents

Notes:

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

Inputs

The power supply can operate at a frequency of either 60 ±3 Hz or 50 ±3 Hz and it can operate at 110 Vac, 5 A or 220/240 Vac, 2.5 A. The voltage is selected with the switch above the power-cord plug at the rear of the power supply. The following figure shows the input requirements.

Range	Voltage (Vac)	Current (Amperes)
115 Vac	Minimum 100 Maximum 125	Maximum 5
230 Vac	Minimum 200 Maximum 240	Maximum 3.0

Input Requirements

Note: The maximum in-rush current is 100 A.

SECTION 3

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 supply also supplies either 115 Vac or 230 Vac for the IBM Monochrome Display.

Nominal Output	Load Current (A) Min	Max	Regulation Tolerance
+5 Vdc	7.0	19.8	+5% to -4%
-5 Vdc	0.0	0.3	+10% to -8%
+12 Vdc	2.5	7.3	+5% to -4%
-12 Vdc	0.0	0.3	+10% to -9%

DC Load Requirements

DC Output Protection

If any output becomes overloaded, the power supply will switch off within 20 milliseconds. An overcurrent condition will not damage the power supply.

Output Voltage Sequencing

Under normal conditions, the output voltage levels track within 300 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, or 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.

Load Resistor

If no fixed disk drive is connected to the power supply, the load resistor must be connected to P10. The load resistor is a 5 ohm, 50 watt resistor.

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 can drive six standard TTL loads.

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

Level (Vdc)	Minimum (Vdc)
+5	+4.5
-5	-3.75
+12	+10.8
-12	-10.4

Sense Level

Connectors

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

Load Point	Voltage (Vdc)	Max. Current (A)
PS8-1	Power Good	See Note
PS8-2	+5	3.8
PS8-3	+12	0.7
PS8-4	-12	0.3
PS8-5	Ground	0.0
PS8-6	Ground	0.0
PS9-1	Ground	0.0
PS9-2	Ground	0.0
PS9-3	-5	0.3
PS9-4	+5	3.8
PS9-5	+5	3.8
PS9-6	+5	3.8
P10-1	+12	2.8
P10-2	Ground	0.0
P10-3	Ground	0.0
P10-4	+5	1.8
P11-1	+12	2.8
P11-2	Ground	0.0
P11-3	Ground	0.0
P11-4	+5	1.8
P12-1	+12	1.0
P12-2	Ground	0.0
P12-3	Ground	0.0
P12-4	+5	0.6

DC Load Distribution

Note: For more details, see "Power-Good Signal".

Notes:

SECTION 4. KEYBOARD

Notes:

Introduction

The 84-Key Keyboard information starts below. Information about the Enhanced Personal Computer Keyboard, hereafter referred to as the 101/102-Key Keyboard, begins on page 4-36.

84-Key Keyboard Description

The keyboard is a low-profile, 84-key, detachable unit. A bidirectional serial interface in the keyboard is used to carry signals between the keyboard and system unit.

Cabling

The keyboard cable connects to the system board through a 5-pin DIN connector. The following table lists the connector pins and their signals.

DIN Connector Pins	Signal Name
1	+KBD CLK
2	+KBD DATA
3	Reserved
4	Ground
5	+5.0 Vdc

SECTION 4

Sequencing Key Code Scanning

The keyboard is able to detect all keys that are pressed, and their scan codes will be sent to the interface in correct sequence, regardless of the number of keys held down. Keystrokes entered while the interface is inhibited (when the key lock is on) will be lost. Keystrokes are stored only when the keyboard is not serviced by the system.

Keyboard Buffer

The keyboard has a 16-character first-in-first-out (FIFO) buffer where data is stored until the interface is ready to receive it.

A buffer-overrun condition will occur if more than 16 codes are placed in the buffer before the first keyed data is sent. The 17th code will be replaced with the overrun code, hex 00. (The 17th position is reserved for overrun codes). If more keys are pressed before the system allows a keyboard output, the data will be lost. When the keyboard is allowed to send data, the characters in the buffer will be sent as in normal operation, and new data entered will be detected and sent.

Keys

All keys are classified as *make/break*, which means when a key is pressed, the keyboard sends a make code for that key to the keyboard controller. When the key is released, its break code is sent (the break code for a key is its make code preceded by hex F0).

All keys are *typematic*. When a key is pressed and held down, the keyboard continues to send the make code for that key until the key is released. The rate at which the make code is sent is known as the *typematic rate* (The typematic rate is described under "Set Typematic Rate/Delay"). When 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. When a key is pressed and held down while the interface is inhibited, only the first make code is stored in the buffer. This prevents buffer overflow as a result of typematic action.

Power-On Routine

Power-On Reset

The keyboard logic generates a POR when power is applied to the keyboard. The POR lasts a minimum of 300 milliseconds and a maximum of 9 seconds.

Note: The keyboard may issue a false return during the first 200 milliseconds after the +5 Vdc is established at the 90% level. Therefore, the keyboard interface is disabled for this period.

Basic Assurance Test

Immediately following the POR, the keyboard executes a basic assurance test (BAT). This test consists of a checksum of all read-only memory (ROM), and a stuck-bit and addressing test of all random-access memory (RAM) in the keyboard's microprocessor. The mode indicators—three light emitting diodes (LEDs) on the upper right-hand corner of the keyboard—are turned on then off, and must be observed to ensure they are operational.

Execution of the BAT will take from 600 to 900 milliseconds. (This is in addition to the time required for the POR.)

The BAT can also be started by a Reset command.

After the BAT, and when the interface is enabled ('clock' and 'data' lines are set high), the keyboard sends a completion code to the interface—either hex AA for satisfactory completion or hex FC (or any other code) for a failure. If the system issues a Resend command, the keyboard sends the BAT completion code again. Otherwise, the keyboard sets the keys to typematic and make/break.

Commands from the System

The commands described below may be sent to the keyboard at any time. The keyboard will respond within 20 milliseconds.

Note: The following commands are those sent by the system. They have a different meaning when issued by the keyboard.

Default Disable (Hex F5)

This command is similar to Set Default, except 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 continues scanning if the keyboard was previously enabled.

Enable (Hex F4)

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

No-Operation (NOP) (Hex FD through F7)

These commands are reserved and are effectively no-operation or NOP. The system does not use these codes. If sent, the keyboard will acknowledge the command and continue in its prior scanning state. No other operation will occur.

No-Operation (NOP) (Hex F2 through EF)

These commands are reserved and are effectively no-operation (NOP). The system does not use these codes. If sent, the keyboard acknowledges the command and continues in its prior scanning state. No other operation will occur.

Resend (Hex FE)

The system can send this command when it detects an error in any transmission from the keyboard. It can be sent only after a keyboard transmission and before the system enables the interface to allow the next keyboard output. Upon receipt of Resend, the keyboard sends the previous output again unless the previous output was Resend. In this case, the keyboard will resend 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 'acknowledge' signal (ACK) and ensures the system accepts the ACK before executing the command. The system signals acceptance of the ACK by raising the clock and data for a minimum of 500 microseconds. The keyboard is disabled from the time it receives the Reset command until the ACK is accepted or until another command overrides the previous one. Following acceptance of the ACK, the keyboard begins the reset operation, which is similar to a power-on reset. The keyboard clears the output buffer and sets up default values for typematic and delay rates.

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 default conditions, and continues scanning (only if the keyboard was previously enabled).

Set Typematic Rate/Delay (Hex F3)

The system issues this command, followed by a parameter, to change the typematic rate and delay. The typematic rate and delay parameters are determined by the value of the byte following the command. Bits 6 and 5 serve as the delay parameter and bits 4, 3, 2, 1, and 0 (the least-significant bit) are the rate parameter. 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 = $(8 + A) \times (2^B) \times 0.00417$ seconds.
where:
A = binary value of bits 2, 1, and 0.
B = binary value of bits 4 and 3.

The typematic rate (make code per second) is 1 per period. The period is determined by the first equation above. The following table results.

Bit 4 - 0	Typematic Rate ± 20%	Bit 4 - 0	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 keyboard responds to the Set Typematic Rate/Delay command with an ACK, stops scanning, and waits for the rate parameter. The keyboard responds to the rate parameter with another ACK, sets the rate and delay, and continues scanning (if the keyboard was previously enabled). If a command is received instead of the rate parameter, the set-typematic-rate function ends with no change to the existing rate, and the new command is processed. However, the keyboard will not resume scanning unless instructed to do so by an Enable command.

The default rate for the system keyboard is as follows:

Typematic rate = 10 characters per second ±20%
Delay = 500 ms ±20%.

Set/Reset Mode Indicators (Hex ED)

Three mode indicators on the keyboard are accessible to the system. The keyboard activates or deactivates these indicators when it receives a valid command from the system. They can be activated or deactivated in any combination.

The system remembers the previous state of an indicator so that its setting does not change when a command sequence is issued to change the state of another indicator.

A Set/Reset Mode Indicators command consists of 2 bytes. The first is the command byte and has the following bit setup:

11101101 – hex ED

The second byte is an option byte. It has a list of the indicators to be acted upon. 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 0's)

Note: Bit 7 is the most-significant bit; bit 0 is the least-significant.

The keyboard will respond to the Set/Reset Mode Indicators command with an ACK, discontinue scanning, and wait for the option byte. The keyboard will respond to the option byte with an ACK, set the indicators, and continue scanning if the keyboard was previously enabled. If another command is received in place of the option byte, execution of the function of the Set/Reset Mode Indicators command is stopped with no change to the indicator states, and the new command is processed. Then scanning is resumed.

Commands to the System

The commands described here are those sent by the keyboard. They have a different meaning when issued by the system.

ACK (Hex FA)

The keyboard issues an ACK response to any valid input other than an Echo or Resend command. If the keyboard is interrupted while sending ACK, it will discard ACK and accept and respond to the new command.

BAT Completion Code (Hex AA)

Following satisfactory completion of the BAT, the keyboard sends hex AA. Hex FC (or any other code) means the keyboard microprocessor check failed.

Break Code Prefix (Hex F0)

This code is sent as the first byte of a 2-byte sequence to indicate the release of a key.

Diagnostic Failure (Hex FD)

The keyboard periodically tests the sense amplifier and sends a diagnostic failure code if it detects any problems. If a failure occurs during BAT, the keyboard stops scanning and waits for a system command or power-down to restart. If a failure is reported after scanning is enabled, scanning continues.

ECHO Response (Hex EE)

This is sent in response to an Echo command from the system.

Overrun (Hex 00)

An overrun character is placed in position 17 of the keyboard buffer, overlaying the last code if the buffer becomes full. The code is sent to the system as an overrun when it reaches the top of the buffer.

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-Code Outputs

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. Refer to "Keyboard Layouts" beginning on page 4-27 to determine the character associated with each key number.

The following table lists the positions of the keys and their make scan codes.

Key Number	Make Code	Key Number	Make Code	Key Number	Make Code
1	0E	31	1C	67	0B
2	16	32	1B	68	0A
3	1E	33	23	69	09
4	26	34	2B	70	05
5	25	35	34	71	04
6	2E	36	33	72	03
7	36	37	3B	73	83
8	3D	38	42	74	01
9	3E	39	4B	90	76
10	46	40	4C	91	6C
11	45	41	52	92	6B
12	4E	43	5A	93	69
13	55	44	12	95	77
14	5D	46	1A	96	75
15	66	47	22	97	73
16	0D	48	21	98	72
17	15	49	2A	99	70
18	1D	50	32	100	7E
19	24	51	31	101	7D
20	2D	52	3A	102	74
21	2C	53	3C	103	7A
22	35	54	49	104	71
23	3C	55	4A	105	84
24	43	57	59	106	7C
25	44	58	11	107	7B
26	4D	61	29	108	79
27	54	64	58		
28	5B	65	06		
30	14	66	0C		

Note: Break code consists of 2 bytes; the first is hex F0, the second is the make scan code for that key.

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 a negative level. When no communication is occurring, both the 'clock' and 'data' lines are at a positive level.

Data transmissions to and from the keyboard consist of 11-bit data streams that are sent serially over the 'data' line. The following table shows the structure of the data stream.

Bit	Function
1	Start bit (always 1)
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 eight data bits plus the parity bit always equals an odd number.

When the system sends data to the keyboard, it forces the 'data' line to a negative level and allows the 'clock' line to go to a positive level.

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 a negative level; the 'data' line may go high or low during this time.

During the BAT, the keyboard allows the 'clock' and 'data' lines to go to a positive level.

SECTION 4

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 low (inhibit status), data is stored in the keyboard buffer. If the 'clock' line is high and 'data' is low (request-to-send), data is stored in the keyboard buffer, and the keyboard receives system data.

If 'clock' and 'data' are both high, the keyboard sends the 0 start bit, 8 data bits, the parity bit and the stop bit. Data will be valid after the rising edge and before the falling edge of the 'clock' line. During transmission, the keyboard checks the 'clock' line for a positive level at least every 60 milliseconds. If the system lowers the 'clock' line from a positive 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 rising edge of the 10th clock (parity bit), the keyboard buffer returns the 'data' and 'clock' lines to a positive level. If contention does not occur by the tenth clock, the keyboard completes the transmission.

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 if the keyboard is sending data. If the keyboard is sending but has not reached the tenth clock, the system can override the keyboard output by forcing the 'clock' line to a negative level. If the keyboard transmission is beyond the tenth clock, 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 'clock' line to a negative level for more than 60 microseconds while preparing to send. When the system is ready to send the start bit ('data' line will be low), it allows the 'clock' line to go to a positive level.

The keyboard checks the state of the 'clock' line at intervals of no less than 60 milliseconds. If a request-to-send is detected, the keyboard counts 11 bits. After the tenth bit, the keyboard forces the 'data' line low and counts one more (the stop 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.

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. A Resend command should not be sent in this case.

Keyboard Encoding and Usage

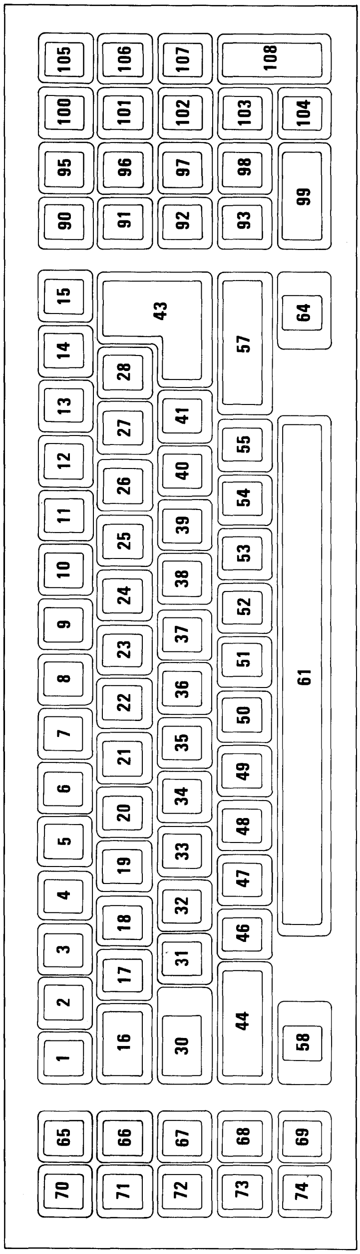
The keyboard routine, provided by IBM in the ROM BIOS, is responsible for converting the keyboard scan codes into what will be termed *Extended ASCII*. The extended ASCII codes returned by the ROM routine are mapped to the U.S. English keyboard layout. Some operating systems may make provisions for alternate 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 table shows the keyboard layout and key positions.



Key	Base Case	Uppercase	Ctrl	Alt
1	`	~	-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	(*)
14	\	|	FS(028)	-1
15	Backspace (008)	Backspace (008)	Del(127)	-1
16	→| (009)	|← (*)	-1	-1
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)	-1
30 Ctrl	-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	-1
41	'	"	-1	-1
43	CR	CR	LF(010)	-1
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	-1
54	.	>	-1	-1
55	/	?	-1	-1
57 Shift (Right)	-1	-1	-1	-1
58 Alt	-1	-1	-1	-1
61	Space	Space	Space	Space
64 Caps Lock	-1	-1	-1	-1
90	Esc	Esc	Esc	-1
95 Num Lock	-1	-1 (*)	Pause (**)	-1
100 Scroll Lock	-1	-1	Break (**)	-1
107	-	-	(*)	(*)
108	Enter	Enter	LF(010)	-1
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(*)

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	-1
97	5	-1	-1	-1
98	2	↓ (*)	-1	-1
99	0	Ins	-1	-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 (*,**)	(**)	(**)
105	-	Sys Request	-1	-1
106	+	+ (*)	-1	-1

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.

SECTION 4

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

Second Code	Function
3	Nul Character
15	|← (Back-tab)
16-25	Alt Q, W, E, R, T, Y, U, I, O, P
30-38	Alt A, S, D, F, G, H, J, K, L
44-50	Alt Z, X, C, V, B, N, M
59-68	F1 to F10 Function Keys (Base Case)
71	Home
72	↑ (Cursor Up)
73	Page Up and Home Cursor
75	← (Cursor Left)
77	→ (Cursor Right)
79	End
80	↓ (Cursor Down)
81	Page Down and Home Cursor
82	Ins (Insert)
83	Del (Delete)
84-93	F11 to F20 (Shift-F1 through Shift-F10)
94-103	F21 to F30 (Ctrl-F1 through Ctrl-F10)
104-113	F31 to F40 (Alt-F1 through Alt-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)

Keyboard Extended Functions

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 14, 16 through 28, 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, 17 through 28, 31 through 39, 43, 46 through 52, 91 through 93, and 101 through 103 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 13, 17 through 26, 31 through 39, and 46 through 52 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.

Note: Character codes 97-122 will display uppercase with Caps Lock activated. 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.

If the keyboard Num Lock Mode indicator and the system get out of synchronization, pressing the key combination of Shift and Num Lock will synchronize them. This key combination changes the Num Lock bit in the keyboard memory, but sends only the scan code for the Shift key to the system.

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 the 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 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 (Ctrl and Num Lock) causes the keyboard interrupt routine to loop, waiting for any key except Num Lock 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 PrtSc 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 (SysReq) key is pressed, a hex 8500 is placed in AX, and an interrupt hex 15 is executed. When the SysReq 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 09H from the keyboard, an interrupt 15H, function (AH)=4FH 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 09H 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 15H, the scan code will be ignored by the interrupt handler.

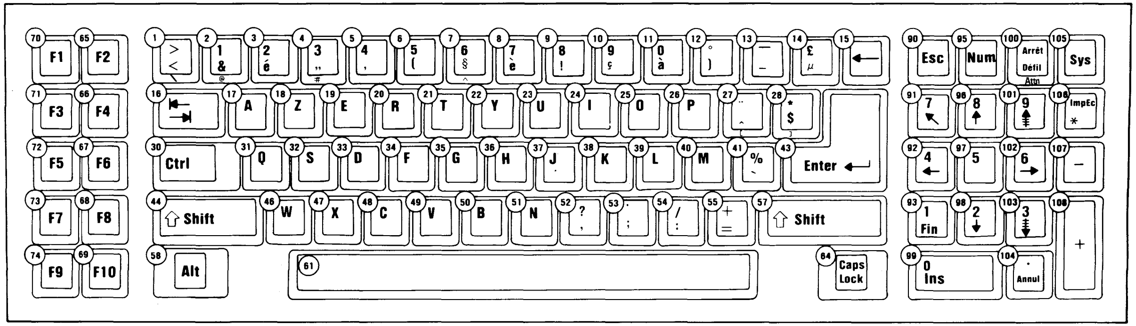
Keyboard Layouts

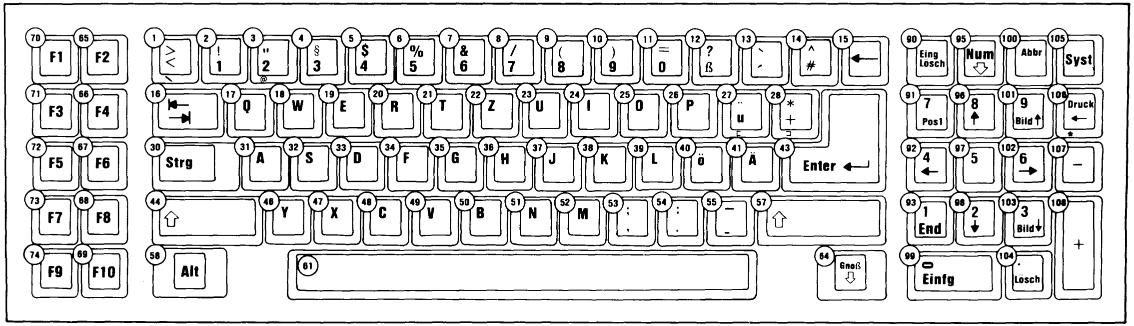
The keyboard has six different layouts:

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

The following pages show the six keyboard layouts.

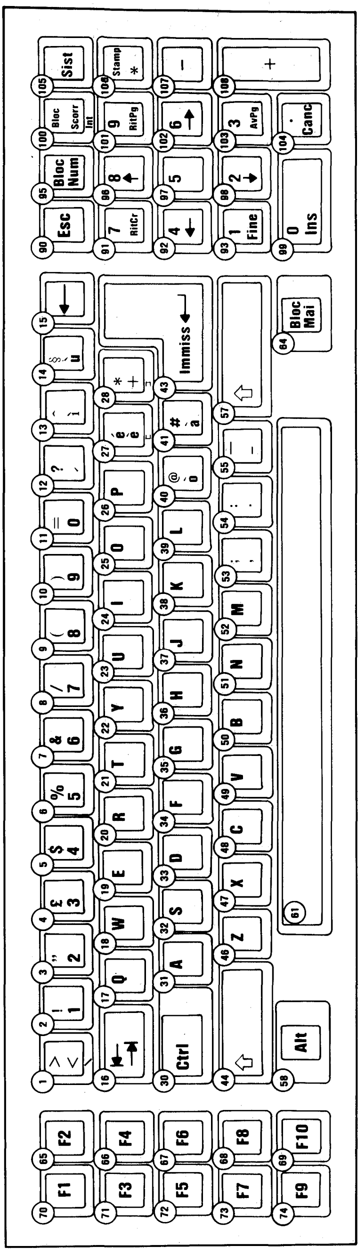
French Keyboard



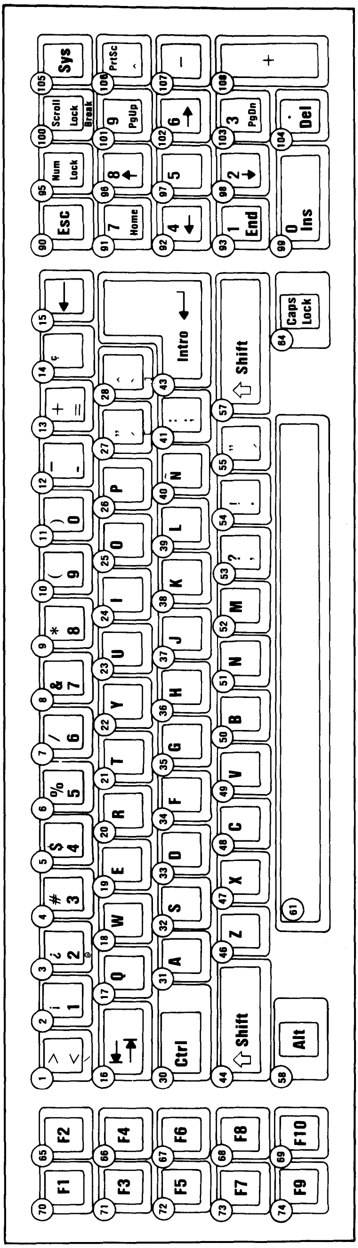


F1
F2
F3
F4
F5
F6
F7
F8
F9
F10
1 ! 2 " @ 3 § 4 $ 5 % 6 & 7 / 8 (9) 0 = ? ß # ^
Q W E R T Z U I O P ü * +
Strg A S D F G H J K L ö Ä Enter
Y X C V B N M ; , : . _ -
Alt
Gnoß
Eing Lösch
Num
Abbr
Syst
7 Pos1
8
9 Bild
Druck
4
5
6
-
1 End
2
3 Bild
+
Einfg
Lösch

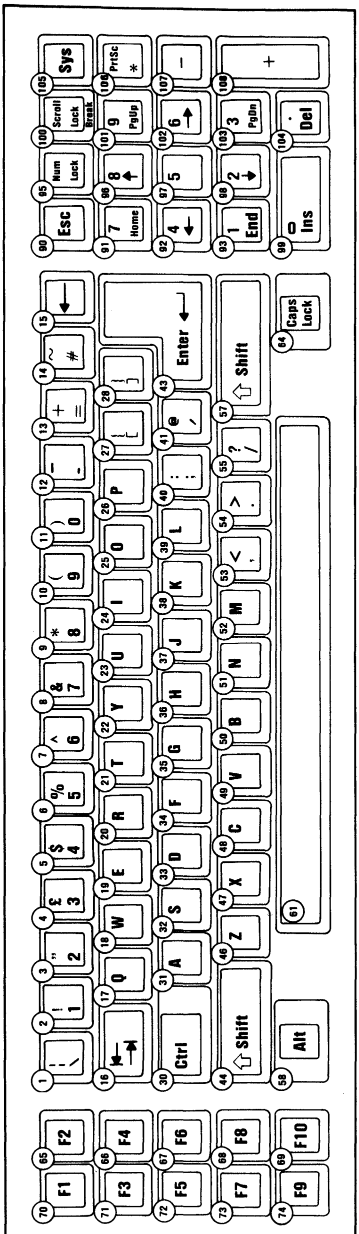
Italian Keyboard



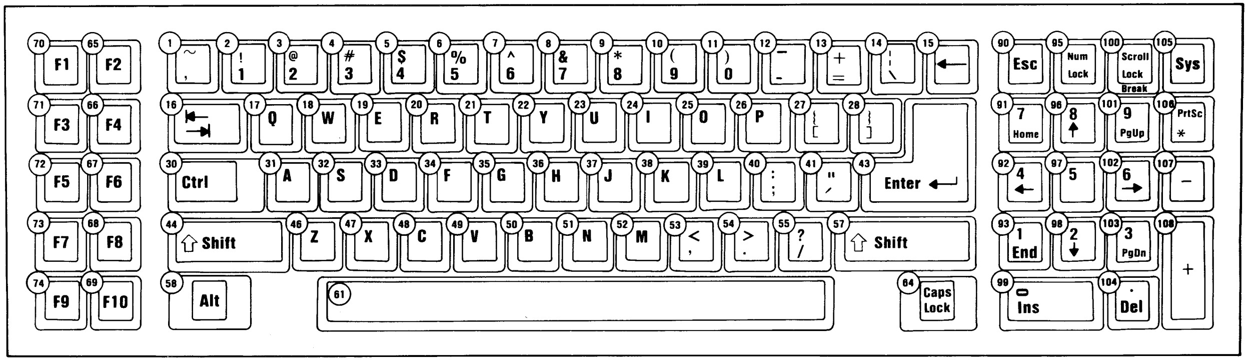
Spanish Keyboard



U.K. English Keyboard



U.S. English Keyboard



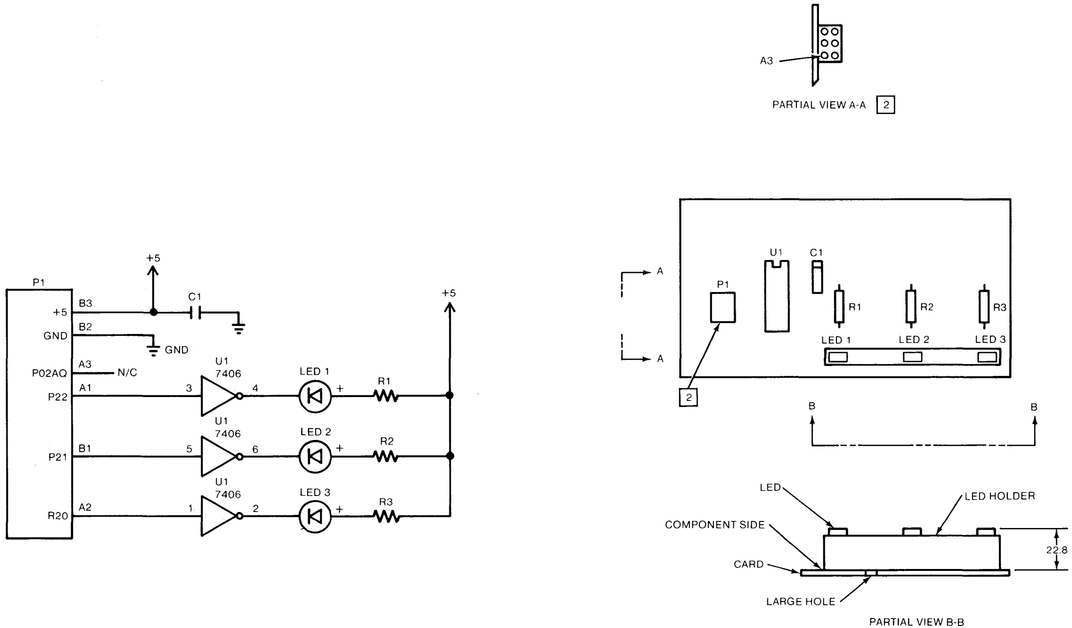
Specifications

Size

- Length: 503 millimeters (19.8 inches)
- Depth: 213 millimeters (8.4 inches)
- Height: 58 millimeters (2.3 inches)

Weight

- 2.8 kilograms (6.2 pounds)



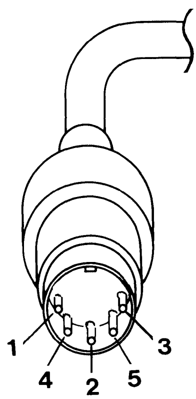
Enhancement Logic Card Assembly

101/102-Key Keyboard Description

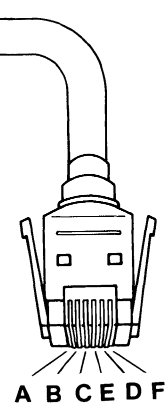
The keyboard has 101 keys (102 in countries outside the U. S.). At system power-on, the keyboard monitors the signals on the 'clock' and 'data' lines and establishes its line protocol. A bidirectional serial interface in the keyboard converts the 'clock' and 'data' signals and sends this information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

Cabling

The keyboard cable connects to the system with a 5-pin DIN connector, and to the keyboard with a 6-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	Power
5	E	+5.0 Vdc	Power
	A	Not used	
Shield	Shield	Frame Ground	

SECTION 4

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 4-45).

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 4-58 for the default settings. Commands to change the default settings are listed in "Commands from the System" on page 4-40.

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 4-47).

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.

SECTION 4

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

SECTION 4

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 4-40).

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 4-42).

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.

SECTION 4

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 4-74 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.

SECTION 4

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.		

SECTION 4

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 4-74 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.

SECTION 4

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.		

SECTION 4

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 4-74 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

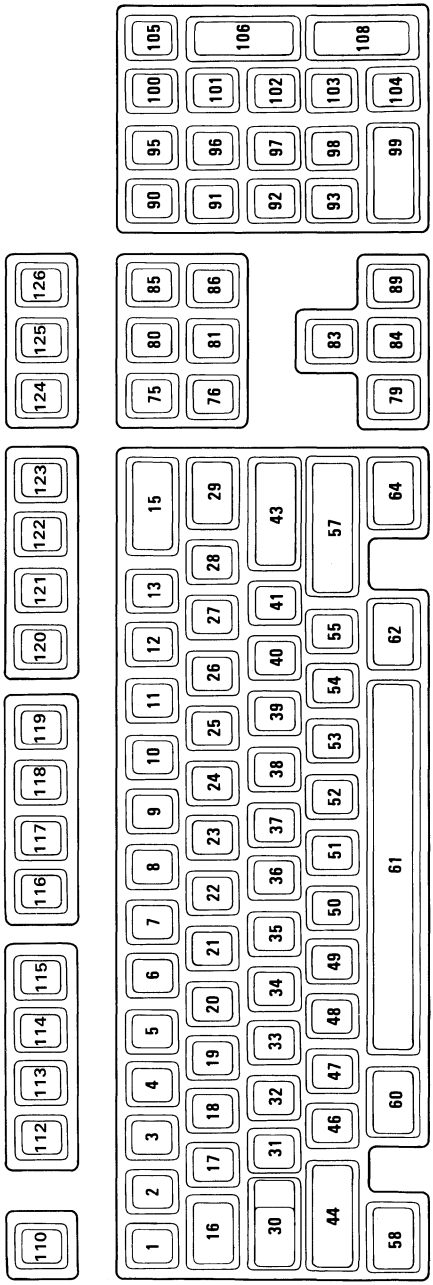
The keyboard routine, provided by IBM in the ROM BIOS, is responsible for converting the keyboard scan codes into what will be termed *Extended ASCII.* The extended ASCII codes returned by the ROM routine are mapped to the U.S. English keyboard layout. Some operating systems may make provisions for alternate 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)

SECTION 4

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.

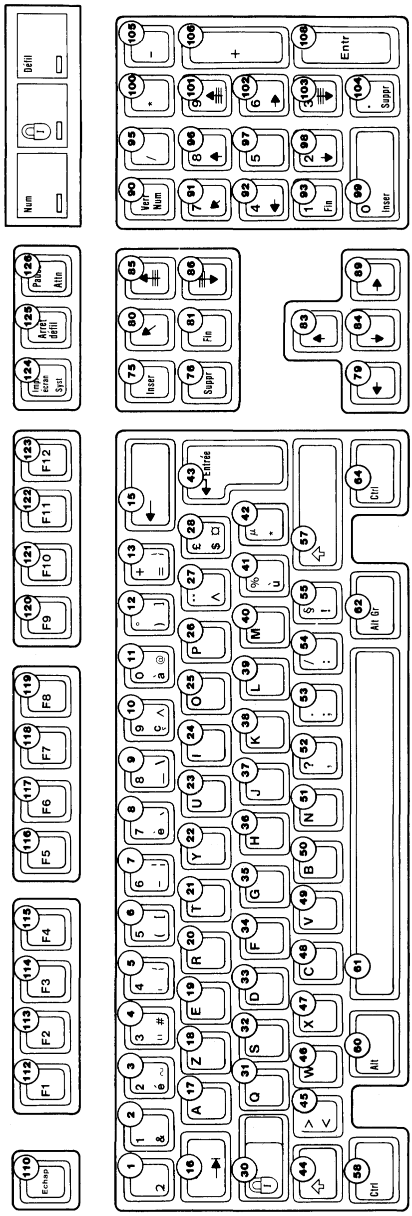
Keyboard Layouts

The keyboard is available in six layouts:

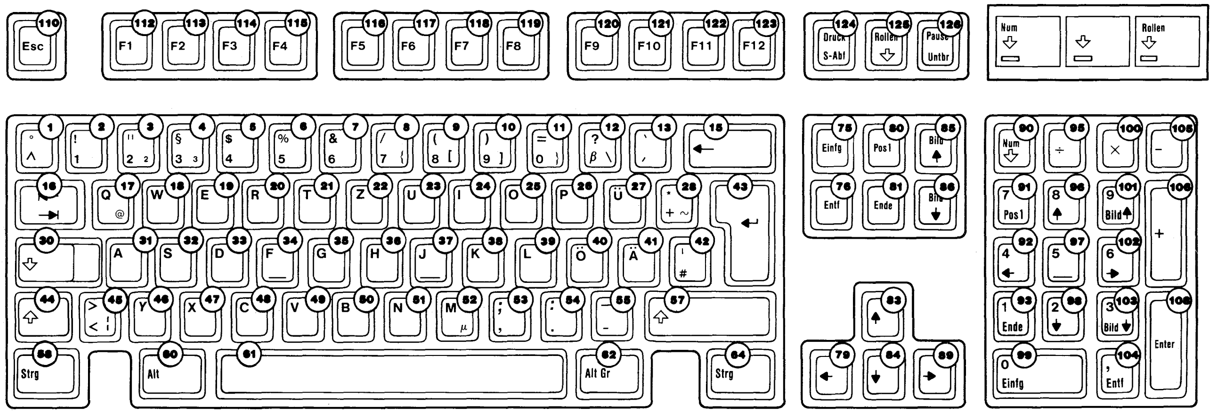
- French
- German
- Italian
- Spanish
- U.K. English
- U.S. English

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.

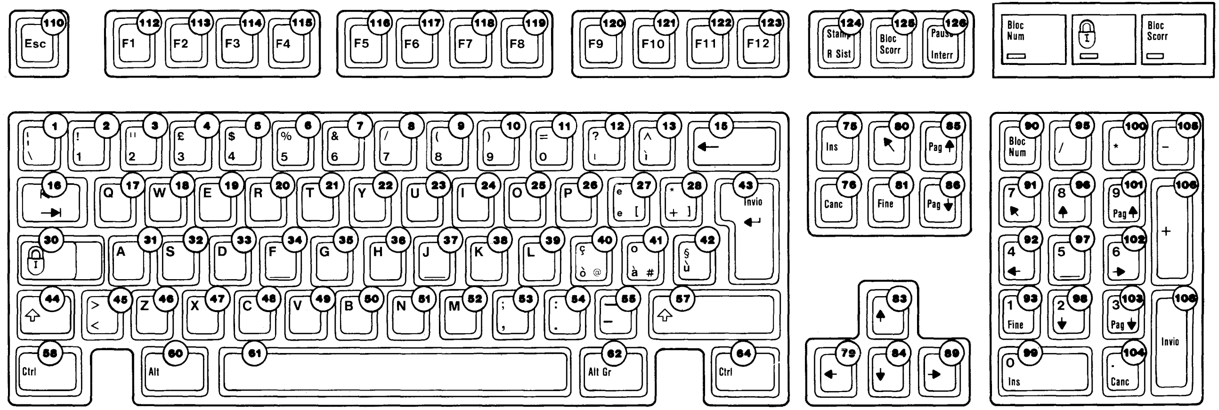
French Keyboard



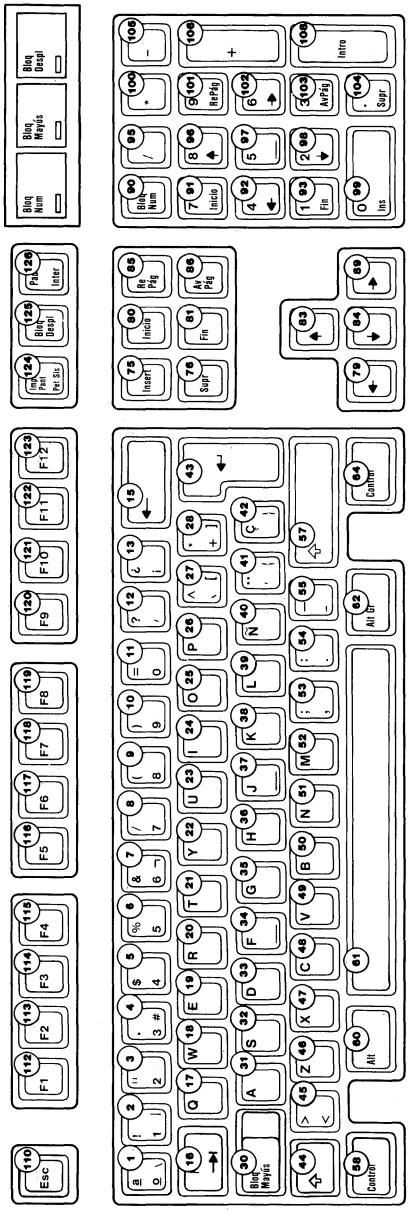
German Keyboard



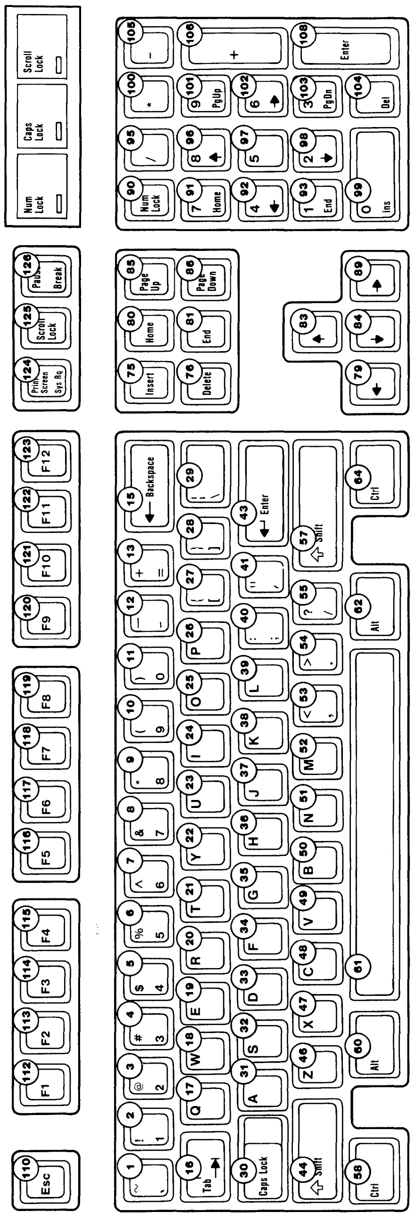
Italian Keyboard



Spanish Keyboard

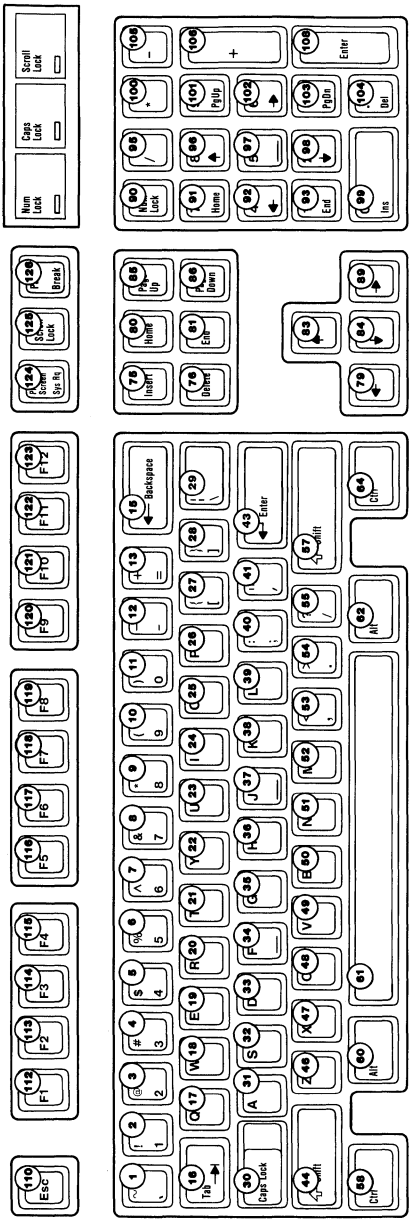


U.K. English Keyboard



SECTION 4

U.S. English Keyboard



Specifications

The specifications for the keyboard follow.

Power Requirements

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

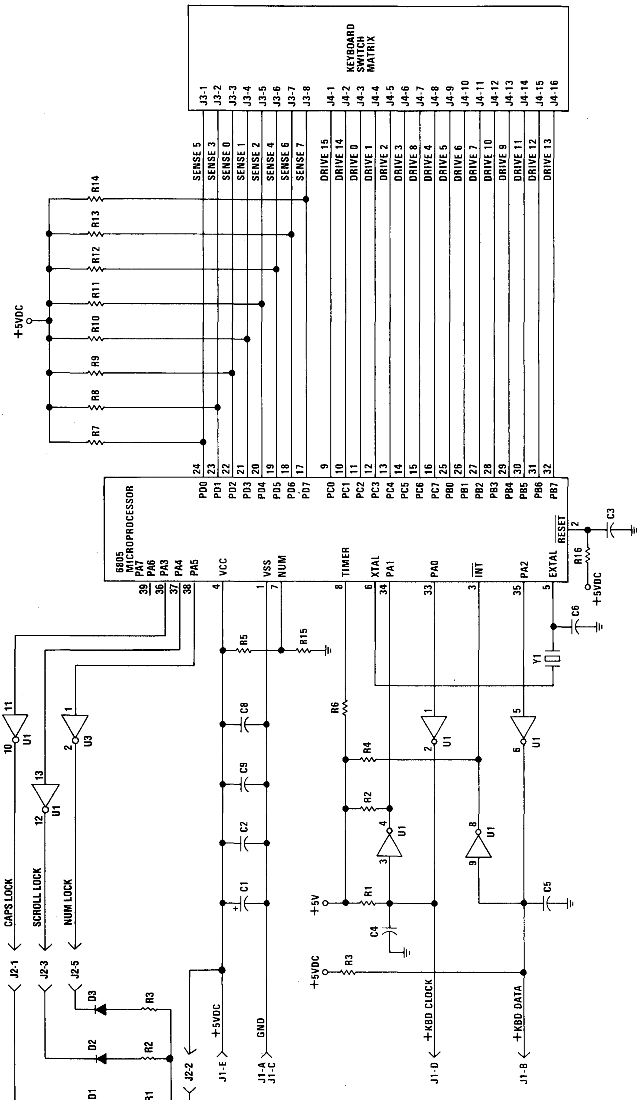
Size

- 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)

Logic Diagram



101/102-KEY KEYBOARD

SECTION 5. SYSTEM BIOS

SECTION 5

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.

If the sockets labeled U17 and U37 on the system board are empty, additional ROM modules may be installed in these sockets. During POST, a test is made for valid code at this location, starting at address hex E0000 and ending at hex EFFFF. More information about these sockets may be found under "Additional System Board ROM Modules" on page 5-13.

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	Nonmaskable	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.

SECTION 5

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 D11)
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 9 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

SECTION 5

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

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

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.

Additional System Board ROM Modules

The POST provides a way to integrate the code for additional ROM modules into the system. These modules are placed in the sockets marked U17 and U37. A test for additional ROM modules on the system board occurs. At this point, the additional ROM, if valid, will gain control.

The absolute addresses, E0000 through EFFFF, are scanned in 64K blocks for a valid checksum. Valid ROM is defined as follows:

Byte 0 Hex 55

Byte 1 Hex AA

Byte 2 Not used

Byte 3 Entry by a CALL FAR

A checksum is done to test the integrity of the ROM modules. Each byte in the ROM modules is summed modulo hex 100. This sum must be 0 for the modules to be valid. This checksum is located at address EFFFF.

When the POST identifies a valid ROM at this segment, it does a CALL FAR to byte 3 of the ROM, which should be executable code.

Quick Reference

```
Warning: No STACK segment

 Start  Stop   Length  Name                  Class
 00000H 0FFFEH FFFFH   CODE

 Origin   Group

  Address            Publics by Name          Address               Publics by Value

 F000:E729          A1                        F000:0000            POST1
 F000:3BEA          ACT_DISP_PAGE             F000:0008  Abs       K6L
 F000:6000          BASIC                     F000:0010  Abs       M4
 F000:19F0          BEEP                      F000:0050            START_1
 F000:1B1A          BLINK_INT                 F000:0396            C8042
 F000:2022          BOOT_STRAP_1              F000:03A2            OBF_42
 F000:0C96          C21                       F000:0C96            POST2
 F000:0396          C8042                     F000:0C96            C21
 F000:42FC          CASSETTE_IO_1             F000:1052            SHUT3
 F000:1941          CMOS_READ                 F000:10B6            SHUT2
 F000:195B          CMOS_WRITE                F000:10B9            SHUT7
 F000:1A45          CONFIG_BAD                F000:10DA            SHUT6
 F000:E6F5          CONF_TBL                  F000:1613            SHUT4
 F000:FA6E          CRT_CHAR_GEN              F000:1671            POST3
 F000:E020          D1                        F000:1941            CMOS_READ
 F000:1BCA          D11                       F000:1941            POST4
 F000:E030          D2                        F000:195B            CMOS_WRITE
 F000:E040          D2A                       F000:1975            DDS
 F000:1975          DDS                       F000:197D            E_MSG
 F000:2143          DISKETTE_IO_1             F000:19A4            P_MSG
 F000:EFC7          DISK_BASE                 F000:19B2            ERR_BEEP
 F000:2BDE          DISK_INT_1                F000:19F0            BEEP
 F000:2DF2          DISK_IO                   F000:1A36            WAITF
 F000:2C49          DISK_SETUP                F000:1A45            CONFIG_BAD
 F000:2BF5          DSKETTE_SETUP             F000:1A59            XPC_BYTE
 F000:FF53          DUMMY_RETURN              F000:1A69            PRT_HEX
 F000:1C18          DUMMY_RETURN_1            F000:1A70            PRT_SEG
 F000:E05E          E101                      F000:1A85            PROT_PRT_HEX
 F000:E077          E102                      F000:1AB1            ROM_CHECKSUM
 F000:E090          E103                      F000:1ABD            ROM_CHECK
 F000:E0A9          E104                      F000:1AEF            KBD_RESET
 F000:E0C2          E105                      F000:1B1A            BLINK_INT
 F000:E0DB          E106                      F000:1B28            SET_TOD
 F000:E0F4          E107                      F000:1BCA            D11
 F000:E10D          E108                      F000:1C18            DUMMY_RETURN_1
 F000:E126          E109                      F000:1C19            RE_DIRECT
 F000:E13F          E161                      F000:1C22            INT_287
 F000:E168          E162                      F000:1C31            PROC_SHUTDOWN
 F000:E191          E163                      F000:1C38            POST5
 F000:E1B7          E164                      F000:1D2A            SYSINIT1
 F000:E1DB          E201                      F000:1EB5            POST6
 F000:E1EE          E202                      F000:1EB5            STGTST_CNT
 F000:E209          E203                      F000:1FB5            ROM_ERR
 F000:E224          E301                      F000:1FE1            XMIT_8042
 F000:E239          E302                      F000:2022            BOOT_STRAP_1
 F000:E2C6          E303                      F000:2143            DISKETTE_IO_1
 F000:E2EA          E304                      F000:2A88            SEEK
 F000:E30E          E401                      F000:2BDE            DISK_INT_1
 F000:E31E          E501                      F000:2BF5            DSKETTE_SETUP
 F000:E32E          E601                      F000:2C49            DISK_SETUP
 F000:E343          E602                      F000:2DF2            DISK_IO
 F000:426F          EQUIPMENT_1               F000:3316            HD_INT
 F000:19B2          ERR_BEEP                  F000:3339            KEYBOARD_IO_1
 F000:197D          E_MSG                     F000:33C5            KB_INT_1
 F000:E364          F1780                     F000:342E            K16
 F000:E379          F1781                     F000:3833            SND_DATA
 F000:E38E          F1782                     F000:38DD            PRINTER_IO_1
 F000:E3AC          F1790                     F000:3967            RS232_IO_1
 F000:E3BF          F1791                     F000:3A77            VIDEO_IO_1
 F000:E3D2          F3A                       F000:3AB6            SET_MODE
 F000:E25D          F3D                       F000:3B86            SET_CTYPE
 F000:E3DF          F3D1                      F000:3BAB            SET_CPOS
 F000:E401          FD_TBL                    F000:3BD3            READ_CURSOR
 F000:4A4F          FILL                      F000:3BEA            ACT_DISP_PAGE
 F000:FF5E          FLOPPY                    F000:3C0E            SET_COLOR
 F000:46C8          GATE_A20                  F000:3C34            VIDEO_STATE
 F000:3316          HD_INT                    F000:3C57            SCROLL_UP
 F000:FF5A          HRD                       F000:3CF6            SCROLL_DOWN
 F000:1C22          INT_287                   F000:3D48            READ_AC_CURRENT
 F000:E8E1          K10                       F000:3DA2            WRITE_AC_CURRENT
 F000:E91B          K11                       F000:3DD4            WRITE_C_CURRENT
 F000:E955          K12                       F000:3E84            READ_DOT
 F000:E95F          K13                       F000:3E95            WRITE_DOT
 F000:E969          K14                       F000:4139            WRITE_TTY
 F000:E976          K15                       F000:41C0            READ_LPEN
 F000:342E          K16                       F000:4265            MEMORY_SIZE_DET_1
 F000:E87E          K6                        F000:426F            EQUIPMENT_1
 F000:0008  Abs     K6L                       F000:4279            NMI_INT_1
 F000:E886          K7                        F000:42FC            CASSETTE_IO_1
 F000:E88E          K8                        F000:4586            SHUT9
 F000:E8C8          K9                        F000:46C8            GATE_A20
 F000:1AEF          KBD_RESET                 F000:4784            TIME_OF_DAY_1
 F000:33C5          KB_INT_1                  F000:4906            RTC_INT
 F000:3339          KEYBOARD_IO_1             F000:4970            PRINT_SCREEN_1
 F000:0010  Abs     M4                        F000:4A06            TIMER_INT_1
 F000:F0E4          M5                        F000:4A4F            FILL
 F000:F0EC          M6                        F000:6000            BASIC
 F000:F0F4          M7                        F000:E020            D1
 F000:4265          MEMORY_SIZE_DET_1         F000:E030            D2
 F000:E2C3          NMI_INT                   F000:E040            D2A
 F000:4279          NMI_INT_1                 F000:E05E            E101
 F000:03A2          OBF_42                    F000:E077            E102
 F000:0000          POST1                     F000:E090            E103
 F000:0C96          POST2                     F000:E0A9            E104
 F000:1671          POST3                     F000:E0C2            E105
 F000:1941          POST4                     F000:E0DB            E106
 F000:1C38          POST5                     F000:E0F4            E107
 F000:1EB5          POST6                     F000:E10D            E108
 F000:38DD          PRINTER_IO_1              F000:E126            E109
 F000:FF54          PRINT_SCREEN              F000:E13F            E161
 F000:4970          PRINT_SCREEN_1            F000:E168            E162
 F000:1C31          PROC_SHUTDOWN             F000:E191            E163
 F000:1A85          PROT_PRT_HEX              F000:E1B7            E164
 F000:1A69          PRT_HEX                   F000:E1DB            E201
 F000:1A70          PRT_SEG                   F000:E1EE            E202
 F000:19A4          P_MSG                     F000:E209            E203
 F000:FFF0          P_O_R                     F000:E224            E301
 F000:3D48          READ_AC_CURRENT           F000:E239            E302
 F000:3BD3          READ_CURSOR               F000:E25D            F3D
```

F000:3E84	READ_DOT	F000:E2C3	NMI_INT
F000:41C0	READ_LPEN	F000:E2C6	E303
F000:1C19	RE_DIRECT	F000:E2EA	E304
F000:1ABD	ROM_CHECK	F000:E30E	E401
F000:1AB1	ROM_CHECKSUM	F000:E31E	E501
F000:1FB5	ROM_ERR	F000:E32E	E601
F000:3967	RS232_IO_1	F000:E343	E602
F000:4906	RTC_INT	F000:E364	F1780
F000:3CF6	SCROLL_DOWN	F000:E379	F1781
F000:3C57	SCROLL_UP	F000:E38E	F1782
F000:2A88	SEEK	F000:E3AC	F1790
F000:FF62	SEEKS_1	F000:E3BF	F1791
F000:3C0E	SET_COLOR	F000:E3D2	F3A
F000:3BAB	SET_CPOS	F000:E3DF	F3D1
F000:3B86	SET_CTYPE	F000:E401	FD_TBL
F000:3AB6	SET_MODE	F000:E6F5	CONF_TBL
F000:1B28	SET_TOD	F000:E729	A1
F000:10B6	SHUT2	F000:E87E	K6
F000:1052	SHUT3	F000:E886	K7
F000:1613	SHUT4	F000:E88E	K8
F000:10DA	SHUT6	F000:E8C8	K9
F000:10B9	SHUT7	F000:E8E1	K10
F000:4586	SHUT9	F000:E91B	K11
F000:FF23	SLAVE_VECTOR_TABLE	F000:E955	K12
F000:3833	SND_DATA	F000:E95F	K13
F000:0050	START_1	F000:E969	K14
F000:1EB5	STGTST_CNT	F000:E976	K15
F000:1D2A	SYSINIT1	F000:EFC7	DISK_BASE
F000:4A06	TIMER_INT_1	F000:F0A4	VIDEO_PARMS
F000:4784	TIME_OF_DAY_1	F000:F0E4	M5
F000:FF66	TUTOR	F000:F0EC	M6
F000:FEF3	VECTOR_TABLE	F000:F0F4	M7
F000:3A77	VIDEO_IO_1	F000:FA6E	CRT_CHAR_GEN
F000:F0A4	VIDEO_PARMS	F000:FEF3	VECTOR_TABLE
F000:3C34	VIDEO_STATE	F000:FF23	SLAVE_VECTOR_TABLE
F000:1A36	WAITF	F000:FF53	DUMMY_RETURN
F000:3DA2	WRITE_AC_CURRENT	F000:FF54	PRINT_SCREEN
F000:3DD4	WRITE_C_CURRENT	F000:FF5A	HRD
F000:3E95	WRITE_DOT	F000:FF5E	FLOPPY
F000:4139	WRITE_TTY	F000:FF62	SEEKS_1
F000:1FE1	XMIT_8042	F000:FF66	TUTOR
F000:1A59	XPC_BYTE	F000:FFF0	P_O_R

SECTION 5

```
        PAGE 118,121
        TITLE TEST1 ---- 11/15/85  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 ??H :
;                          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  ;----------------------------------------
                          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
                          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
                          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
```

```
                          PAGE
                       C  INCLUDE POSTEQU.INC
                       C  ;----------------------------------------
                       C  ;       EQUATES USED BY POST AND BIOS   :
                       C  ;----------------------------------------
                       C
 = 00FC                C  MODEL_BYTE       EQU    0FCH          ; SYSTEM MODEL BYTE
 = 0001                C  SUB_MODEL_BYTE   EQU    001H          ; SYSTEM SUB-MODEL TYPE
 = 0000                C  BIOS_LEVEL       EQU    000H          ; BIOS REVISION LEVEL
 = F600                C  RATE_UPPER       EQU    0F600H        ; UPPER LIMIT + 5%
 = F9FD                C  RATE_LOWER       EQU    0F9FDH        ; LOWER LIMIT - 10%
                       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  ;---------- 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  ;                       00000001B     ; SCROLL LOCK INDICATOR
                       C  ;                       00000010B     ; NUM LOCK INDICATOR
                       C  ;                       00000100B     ; 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
          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
= 0085    C  ID_2A           EQU     085H               ; 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  ;----------------------------------------
                     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_DISK_1       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  ;---------- 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  ;---------- 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  ;---------- 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
                            C     INCLUDE SYSDATA.INC
                            C     ;--------------------------------------------------------------------------------
                            C     ;       PROTECTED MODE EQUATES FOR POST TESTS AND BIOS ROUTINES              :
                            C     ;--------------------------------------------------------------------------------
                            C
                            C     ;-----          LENGTH EQUATES FOR PROTECTED MODE TESTS
                            C
 = 0300                     C     SDA_LEN         EQU     00300H          ; SYSTEM DATA AREA LENGTH
 = 0800                     C     SYS_IDT_LEN     EQU     256*8           ; 256 SYSTEM IDT ENTRIES, 8 BYTES EACH
 = 0088                     C     GDT_LEN         EQU     TYPE GDT_DEF    ; GDT STRUCTURE LENGTH
 = 0008                     C     DESC_LEN        EQU     TYPE DATA_DESC  ; LENGTH OF A DESCRIPTOR
 = 1000                     C     MCRT_SIZE       EQU     4*1024          ; MONOCHROME CRT SIZE
 = 4000                     C     CCRT_SIZE       EQU     16*1024         ; COMPATIBLE COLOR CRT SIZE
 = FFFF                     C     ECCRT_SIZE      EQU     0FFFFH          ; SIZE OF EACH PORTION OF THE ENHANCED
 = FFFF                     C     MAX_SEG_LEN     EQU     0FFFFH          ; MAXIMUM SEGMENT LENGTH = 64K
 = 0000                     C     NULL_SEG_LEN    EQU     00000H          ; NULL SEGMENT LENGTH = 0
                            C
                            C     ;-----          LOCATION EQUATES FOR PROTECTED MODE TESTS
                            C
 = D0A0                     C     SYS_IDT_LOC     EQU     0D0A0H          ; THE SYSTEM IDT IS AT THE BOTTOM
 = 0400                     C     SDA_LOC         EQU     00400H          ; SAME AS REAL
 = D8A0                     C     GDT_LOC         EQU     (SYS_IDT_LOC + SYS_IDT_LEN)
 = 0000                     C     MCRT@_LO        EQU     0000H           ; MONOCHROME CRT ADDRESS
 = 000B                     C     MCRT@_HI        EQU     0BH             ; (0B0000H)
 = 8000                     C     CCRT@_LO        EQU     8000H           ; COMPATIBLE COLOR CRT ADDRESS
 = 000B                     C     CCRT@_HI        EQU     0BH             ; (0B8000H)
 = 0000                     C     ECCRT@_LO_LO    EQU     0000H
 = 000A                     C     ECCRT@_LO_HI    EQU     0AH             ; (0A0000H)
 = 0000                     C     ECCRT@_HI_LO    EQU     0000H
 = 000B                     C     ECCRT@_HI_HI    EQU     0BH             ; (0B0000H)
 = 0000                     C     CSEG@_LO        EQU     0000H           ; CODE SEGMENT POST/BIOS
 = 000F                     C     CSEG@_HI        EQU     0FH             ; (0F0000H) FOR TESTS
 = 0000                     C     NSEG@_LO        EQU     0000H           ; ABS0
 = 0000                     C     NSEG@_HI        EQU     00H
                            C
                            C     ;-----  DEFINITIONS FOR ACCESS RIGHTS BYTES
                            C
 = 00F3                     C     CPL3_DATA_ACCESS        EQU     11110011B       ; PRESENT
                            C                                                     ; DPL = 3
                            C                                                     ; CODE/DATA SEGMENT
                            C                                                     ; NOT EXECUTABLE
                            C                                                     ; GROW-UP (OFFSET <= LIMIT)
                            C                                                     ; WRITABLE
                            C                                                     ; ACCESSED
 = 0093                     C     CPL0_DATA_ACCESS        EQU     10010011B       ; DPL = 0
 = 009B                     C     CPL0_CODE_ACCESS        EQU     10011011B       ; CPL 0 - NON-CONFORMING
 = 00E2                     C     LDT_DESC                EQU     11100010B
 = 0081                     C     FREE_TSS                EQU     10000001B
 = 0086                     C     INT_GATE                EQU     10000110B
 = 0087                     C     TRAP_GATE               EQU     10000111B
                            C
 = 0001                     C     VIRTUAL_ENABLE          EQU     0000000000000001B      ; PROTECTED MODE ENABLE
                            C
                            C
                            C     ;-----  THE GLOBAL DESCRIPTOR TABLE DEFINITION FOR POWER ON SELF TESTS
                            C
                            C     GDT_DEF         STRUC
 0000 ????????????????C                           DQ      ?               ; UNUSED ENTRY
 0008 ????????????????C           GDT_PTR         DQ      ?               ; THIS ENTRY POINTS TO THIS TABLE
 0010 ????????????????C           SYS_IDT_PTR     DQ      ?               ; POST INTERRUPT DESCRIPTOR TABLE
 0018 ????????????????C           RSDA_PTR        DQ      ?               ; THE REAL SYSTEM DATA AREA FOR POST
 0020 ????????????????C           C_BWCRT_PTR     DQ      ?               ; COMPATIBLE BW CRT FOR POST
 0028 ????????????????C           C_CCRT_PTR      DQ      ?               ; COMPATIBLE COLOR CRT FOR POST
 0030 ????????????????C           E_CCRT_PTR      DQ      ?               ; ENHANCED COLOR GRAPHICS CRT (16 BYTES)
 0038 ????????????????C           E_CCRT_PTR2     DQ      ?
 0040 ????????????????C           SYS_ROM_CS      DQ      ?               ; CS - POST IDT, ROM RESIDENT
 0048 ????????????????C           ES_TEMP         DQ      ?               ; DYNAMIC POINTER FOR ES
 0050 ????????????????C           CS_TEMP         DQ      ?               ; DYNAMIC POINTER FOR CS
 0058 ????????????????C           SS_TEMP         DQ      ?               ; DYNAMIC POINTER FOR SS
 0060 ????????????????C           DS_TEMP         DQ      ?               ; DYNAMIC POINTER FOR DS
 0068 ????????????????C           POST_TR         DQ      ?               ; TR VALUE FOR THIS MACHINE'S TSS
 0070 ????????????????C           POST_TSS_PTR    DQ      ?
 0078 ????????????????C           POST_LDTR       DQ      ?               ; LDTR VALUE FOR THIS MACHINE'S LDT
 0080 ????????????????C           POST_LDT_PTR    DQ      ?
 0088                       C     GDT_DEF         ENDS
                            C
                            C     ;-----  SEGMENT DESCRIPTOR TABLE ENTRY STRUCTURE
                            C
                            C     DATA_DESC       STRUC
 0000 ????                  C     SEG_LIMIT       DW      ?               ; SEGMENT LIMIT (1 - 65535 BYTES)
 0002 ????                  C     BASE_LO_WORD    DW      ?               ; 24 BIT SEGMENT PHYSICAL
 0004 ??                    C     BASE_HI_BYTE    DB      ?               ;       ADDRESS (0 - (16M-1))
 0005 ??                    C     DATA_ACC_RIGHTS DB      ?               ; ACCESS RIGHTS BYTE
 0006 ????                  C     DATA_RESERVED   DW      ?               ; RESERVED - MUST BE 0000 FOR THE 80286
 0008                       C     DATA_DESC       ENDS
                            C
                            C     ;-----  GATE DESCRIPTOR TABLE ENTRY STRUCTURE
                            C
                            C     GATE_DESC       STRUC
 0000 ????                  C     ENTRY_POINT     DW      ?               ; DESTINATION ROUTINE ENTRY POINT
 0002 ????                  C     CS_SELECTOR     DW      ?               ; SELECTOR FOR DESTINATION SEGMENT
 0004 ??                    C     WORD_COUNT      DB      ?               ; NUMBER OF WORDS TO COPY FROM STACK
 0005 ??                    C     GATE_ACC_RIGHTS DB      ?               ; ACCESS RIGHTS BYTE
 0006 ????                  C     GATE_RESERVED   DW      ?               ; RESERVED - MUST BE 0000 FOR THE 80286
 0008                       C     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 36 31 58 39 32 36                  DB      '62X0820COPR. IBM CORP. 1981,1985 '          ;COPYRIGHT NOTICE
     36 43 4F 50 52 2E
     20 49 42 4D 20 43
     4F 52 50 2E 20 31
     39 38 31 2C 31 39
     38 35 20 20
                                        EVEN                                                  ;EVEN BOUNDARY
                                        ;        6 2 X 0 8 2 0   C O P R.   I B M   1 9 8 5   ;EVEN MODULE
                                        ;         6 2 X 0 2 8 1   C O P R..  I B M   1 9 8 5  ;ODD  MODULE
0022 36 36 31 31 58 58                  DB      '6622XX00882201  CCOOPPRR..  IIBBMM  11998855' ;COPYRIGHT NOTICE
     39 39 32 32 36 36
     36 35 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
     35 35
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
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      C7A                     ; CONTINUE CONFIDENCE TESTS IF "ZF" SET
0080                            ERR02:
0080 F4                                 HLT                             ;       ERROR HALT
0081 EB FD                              JMP     ERR02                   ; ERROR LOOP TRAP
```

SECTION 5

```
0083                    C7A:
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     C7B                     ; GO IF YES
008E E9 0123 R                  JMP     SHUT0                   ; ELSE CONTINUE NORMAL POWER ON CODE
```

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

                          ;----- 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 098E 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 0791 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)
```

SECTION 5

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

 0119                           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

 0123                           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 EC                            MOV     BP,SP
 013D  8B F5                            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:                                     ; TST1A
 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  B8 8B8B                          MOV     AX,X*(CMOS_REG_B+NMI)   ; ADDRESS TO BOTH (AH) AND (AL)
 0152  E6 70                            OUT     CMOS_PORT,AL            ; ADDRESS CMOS ALARM BYTE WITH NMI=OFF
 0154  90                               NOP                             ; I/O DELAY
 0155  E4 71                            IN      AL,CMOS_DATA            ; GET THE CURRENT CONTROL REGISTER
 0157  24 07                            AND     AL,00000111B            ; CLEAR SET,PIE,AIE, AND SQWE BITS
 0159  86 C4                            XCHG    AL,AH                   ; SAVE IT
 015B  E6 70                            OUT     CMOS_PORT,AL
 015D  86 C4                            XCHG    AL,AH
 015F  E6 71                            OUT     CMOS_DATA,AL

 0161  B0 8C                            MOV     AL,CMOS_REG_C+NMI       ; ADDRESS CMOS FLAGS BYTE WITH NMI=OFF
 0163  90                               NOP                             ; I/O DELAY
 0164  E6 70                            OUT     CMOS_PORT,AL
 0166  90                               NOP                             ; I/O DELAY
 0167  E4 71                            IN      AL,CMOS_DATA            ; READ STATUS TO CLEAR PENDING INTERRUPT

                                ;-----  RESET VIDEO

 0169  B0 00                            MOV     AL,0                    ; CLEAR DATA BYTE TO DISABLE VIDEO
 016B  BA 03D8                          MOV     DX,03D8H                ; GET COLOR MODE CONTROL PORT ADDRESS
 016E  EE                               OUT     DX,AL                   ; DISABLE COLOR VIDEO
 016F  FE C0                            INC     AL                      ; MONOCHROME MODE RESET MASK
 0171  B2 B8                            MOV     DL,0B8H                 ; GET ADDRESS OF MONOCHROME MODE CONTROL
 0173  EE                               OUT     DX,AL                   ; DISABLE B/W VIDEO, ENABLE HIGH RES
 0174  B2 BA                            MOV     DL,0BAH                 ; ADDRESS OF MONOCHROME STATUS REGISTER
 0176  EC                               IN      AL,DX                   ; READ STATUS TO DISABLE EGA VIDEO
 0177  B2 DA                            MOV     DL,0DAH                 ; ADDRESS OF COLOR MODE STATUS REGISTER
 0179  EC                               IN      AL,DX                   ; READ STATUS TO DISABLE EGA VIDEO
 017A  B0 00                            MOV     AL,0                    ; SELECT ATTRIBUTE PALETTE REGISTER 0
 017C  B2 C0                            MOV     DL,0C0H                 ; WRITE 0 TO ATTRIBUTE ADDRESS REGISTER
 017E  EE                               OUT     DX,AL                   ; TO DISABLE EGA VIDEO
 017F  B0 FC                            MOV     AL,11111100B            ; DISABLE PARITY CHECKERS
 0181  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

 0183  B0 02                            MOV     AL,02H                  ;       <><><><><><><><><><><><><>
 0185  E6 80                            OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  02 <><>

                                        ASSUME  SS:CODE
 0187  8C C8                            MOV     AX,CS                   ; SETUP SS SEGMENT REGISTER
 0189  8E D0                            MOV     SS,AX
 018B  8E D8                            MOV     DS,AX                   ; SET UP DATA SEGMENT TO POINT TO
 018D  33 F6                            XOR     SI,SI                   ; ROM ADDRESS START
 018F  33 DB                            XOR     BX,BX                   ; CLEAR CHECK REGISTER
 0191  B5 80                            MOV     CH,080H                 ; COUNT FOR 32K WORDS
 0193                           C11:
 0193  AD                               LODSW                           ; MOVE TWO BYTES INTO AX -- SI=SI+2
 0194  02 DC                            ADD     BL,AH                   ; ADD ODD BYTE AT DS:SI+1 TO CHECKSUM
 0196  02 D8                            ADD     BL,AL                   ; ADD EVEN BYTE AT DS:SI TO CHECKSUM
 0198  E2 F9                            LOOP    C11                     ; LOOP COUNT FOR 65K BYTES (32K WORDS)
 019A  73 02                            JNC     C11E                    ;  EXIT IF "LOOP" RESET THE CARRY FLAG
                                                                        ;  (NOTE: MODEL BYTE MUST NOT = ZERO)
 019C  74 01                            JZ      C11A                    ; CONTINUE IF CHECKSUM VALID (ZERO)
 019E                           C11E:
 019E  F4                               HLT                             ; ELSE HALT IF CHECKSUM PROBLEM
                                ;---------------------------------------
                                ; TEST.03                              :
                                ;       VERIFY CMOS SHUTDOWN BYTE      :
```

```
                           ; DESCRIPTION                            :
                           ;       ROLLING BIT WRITTEN AND          :
                           ;       VERIFIED AT SHUTDOWN ADDRESS.    :
                           ;----------------------------------------

                           ;-----  VERIFY AND CLEAR SHUTDOWN FLAG
019F                       C11A:
019F B0 03                         MOV     AL,03H                  ;       <><><><><><><><><><><><><>
01A1 E6 80                         OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  03 <><>

01A3 B9 0009                       MOV     CX,09H                  ; LOOP COUNT
01A6 B4 01                         MOV     AH,1                    ; START WITH BIT 0
01A8                       C11B:
01A8 B0 8F                         MOV     AL,CMOS_SHUT_DOWN+NMI
01AA E6 70                         OUT     CMOS_PORT,AL
01AC 8A C4                         MOV     AL,AH                   ; OUTPUT ROLLING BIT
01AE E6 71                         OUT     CMOS_DATA,AL
01B0 B0 8F                         MOV     AL,CMOS_SHUT_DOWN+NMI   ; READ CMOS
01B2 90                            NOP                             ; I/O DELAY
01B3 E6 70                         OUT     CMOS_PORT,AL
01B5 90                            NOP                             ; I/O DELAY
01B6 E4 71                         IN      AL,CMOS_DATA
01B8 3A C4                         CMP     AL,AH                   ; MUST BE THE SAME
01BA 75 92                         JNZ     ERR01                   ; ERROR IF NOT
01BC D0 D4                         RCL     AH,1                    ; ROLL A BIT THROUGH SHUTDOWN BYTE
01BE E2 E8                         LOOP    C11B                    ; LOOP TILL DONE


                           ;----------------------------------------
                           ; TEST.04                                :
                           ;       8254 CHECK TIMER 1 ALL BITS ON   :
                           ; DESCRIPTION                            :
                           ;       SET TIMER COUNT                  :
                           ;       CHECK THAT TIMER 1 ALL BITS ON   :
                           ;----------------------------------------
                                   ASSUME  DS:DATA
01C0 B8 ---- R                     MOV     AX,DATA                 ; SET DATA SEGMENT
01C3 8E D8                         MOV     DS,AX
01C5 B0 04                         MOV     AL,04H                  ;       <><><><><><><><><><><><><>
01C7 E6 80                         OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  04 <><>

                           ;-----  DISABLE DMA CONTROLLER
                                                                   ; (AL) ALREADY = 04H
01C9 E6 08                         OUT     DMA08,AL                ; DISABLE DMA CONTROLLER 1
01CB E6 D0                         OUT     DMA18,AL                ; DISABLE DMA CONTROLLER 2

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

01CD 8B 16 0072 R                  MOV     DX,@RESET_FLAG          ; SAVE RESET FLAG WHILE REFRESH IS OFF
01D1 B0 54                         MOV     AL,54H                  ; SELECT TIMER 1,LSB,MODE 2
01D3 E6 43                         OUT     TIMER+3,AL
01D5 EB 00                         JMP     $+2                     ; I/O DELAY
01D7 8A C1                         MOV     AL,CL                   ; SET INITIAL TIMER COUNT TO 0
01D9 E6 41                         OUT     TIMER+1,AL
01DB B7 05                         MOV     BH,05H                  ; LOOP COUNT
01DD                       C12:                                    ; TIMER1_BITS_ON
01DD B0 40                         MOV     AL,40H                  ; LATCH TIMER 1 COUNT
01DF EB 00                         JMP     $+2                     ; I/O DELAY
01E1 E6 43                         OUT     TIMER+3,AL
01E3 80 FB FF                      CMP     BL,0FFH                 ; YES - SEE IF ALL BITS GO OFF
01E6 74 0B                         JE      C13                     ; TIMER1_BITS_OFF
01E8 E4 41                         IN      AL,TIMER+1              ; READ TIMER 1 COUNT
01EA 0A D8                         OR      BL,AL                   ; ALL BITS ON IN TIMER
01EC E2 EF                         LOOP    C12                     ; TIMER1_BITS_ON
01EE FE CF                         DEC     BH
01F0 75 EB                         JNZ     C12                     ; TRY AGAIN
01F2 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

01F3 B0 05                 C13:    MOV     AL,05H                  ;       <><><><><><><><><><><><><>
01F5 E6 80                         OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  05 <><>

01F7 8A C3                         MOV     AL,BL                   ; SET TIMER 1 COUNT
01F9 2B C9                         SUB     CX,CX
01FB E6 41                         OUT     TIMER+1,AL
01FD B7 05                         MOV     BH,05H                  ; SET TRY AGAIN COUNT
01FF                       C14:                                    ; TIMER_LOOP
01FF B0 40                         MOV     AL,40H                  ; LATCH TIMER 1 COUNT
0201 E6 43                         OUT     TIMER+3,AL
0203 EB 00                         JMP     $+2                     ; DELAY FOR TIMER
0205 EB 00                         JMP     $+2                     ; ADDED DELAY FOR TIMER
0207 E4 41                         IN      AL,TIMER+1              ; READ TIMER 1 COUNT
0209 22 D8                         AND     BL,AL
020B 74 07                         JZ      C15                     ; GO TO WRAP DMA REGISTER TESTS
020D E2 F0                         LOOP    C14                     ; TIMER_LOOP
020F FE CF                         DEC     BH
0211 75 EC                         JNZ     C14
0213 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

0214                       C15:
0214 B8 ---- R                     MOV     AX,DATA                 ; SET DATA SEGMENT
0217 8E D8                         MOV     DS,AX
0219 B0 06                         MOV     AL,06H                  ;       <><><><><><><><><><><><><>
```

SECTION 5

```
021B E6 80                      OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  06 <><>
021D 89 16 0072 R               MOV     @RESET_FLAG,DX          ; RESTORE SOFT RESET FLAG
0221 E6 0D                      OUT     DMA+0DH,AL              ; SEND MASTER CLEAR TO DMA

                        ;-----  WRAP DMA 0 CHANNEL ADDRESS AND COUNT REGISTERS

0223 B0 FF                      MOV     AL,0FFH                 ; WRITE PATTERN "FF" TO ALL REGISTERS
0225 8A D8              C16:    MOV     BL,AL                   ; SAVE PATTERN FOR COMPARE
0227 8A F8                      MOV     BH,AL
0229 B9 0008                    MOV     CX,8                    ; SETUP LOOP COUNT
022C BA 0000                    MOV     DX,DMA                  ; SETUP I/O PORT ADDRESS OF REGISTER
022F EE                 C17:    OUT     DX,AL                   ; WRITE PATTERN TO REGISTER, LSB
0230 EB 00                      JMP     $+2                     ; I/O DELAY
0232 EE                         OUT     DX,AL                   ; MSB OF 16 BIT REGISTER
0233 B0 01                      MOV     AL,01H                  ; AL TO ANOTHER PATTERN BEFORE READ
0235 EB 00                      JMP     $+2                     ; I/O DELAY
0237 EC                         IN      AL,DX                   ; READ 16-BIT DMA CH REG, LSB  2ST DMA
0238 EB 00                      JMP     $+2                     ; I/O DELAY
023A 8A E0                      MOV     AH,AL                   ; SAVE LSB OF 16-BIT REGISTER
023C EC                         IN      AL,DX                   ; READ MSB OF DMA CHANNEL REGISTER
023D 3B D8                      CMP     BX,AX                   ; PATTERN READ AS WRITTEN?
023F 74 01                      JE      C18                     ; YES - CHECK NEXT REGISTER
0241 F4                         HLT                             ; NO - HALT THE SYSTEM
0242                    C18:                                    ; NXT_DMA_CH
0242 42                         INC     DX                      ; SET I/O PORT TO NEXT CHANNEL REGISTER
0243 E2 EA                      LOOP    C17                     ; WRITE PATTERN TO NEXT REGISTER
0245 FE C0                      INC     AL                      ; SET PATTERN TO 0
0247 74 DC                      JZ      C16                     ; YES CONTINUE

                        ;-----  WRITE DMA WITH 55 PATTERN

0249 80 FB 55                   CMP     BL,055H                 ; CHECK IF "55" PATTERN DONE
024C 74 09                      JZ      C19                     ; GO IF YES
024E 80 FB AA                   CMP     BL,0AAH                 ; CHECK IF "AA" PATTERN DONE
0251 74 08                      JZ      C20                     ; GO IF YES
0253 B0 55                      MOV     AL,055H
0255 EB CE                      JMP     C16

                        ;-----  WRITE DMA WITH AA PATTERN

0257 B0 AA              C19:    MOV     AL,0AAH
0259 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

025B B0 07              C20:    MOV     AL,07H                  ;       <><><><><><><><><><><><>
025D E6 80                      OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  07 <><>
025F E6 DA                      OUT     DMA1+0DH*2,AL           ; SEND MASTER CLEAR TO 2ND DMA

                        ;-----  WRAP DMA 1 CHANNEL ADDRESS AND COUNT REGISTERS

0261 B0 FF                      MOV     AL,0FFH                 ; WRITE PATTERN FF TO ALL REGISTERS
0263 8A D8              C16A:   MOV     BL,AL                   ; SAVE PATTERN FOR COMPARE
0265 8A F8                      MOV     BH,AL
0267 B9 0008                    MOV     CX,8                    ; SETUP LOOP COUNT
026A BA 00C0                    MOV     DX,DMA1                 ; SETUP I/O PORT ADDRESS OF REGISTER
026D EE                 C17A:   OUT     DX,AL                   ; WRITE PATTERN TO REGISTER, LSB
026E EB 00                      JMP     $+2                     ; I/O DELAY
0270 EE                         OUT     DX,AL                   ; MSB OF 16 BIT REGISTER
0271 B0 01                      MOV     AL,01H                  ; AL TO ANOTHER PAT BEFORE RD
0273 EB 00                      JMP     $+2                     ; I/O DELAY
0275 EC                         IN      AL,DX                   ; READ 16-BIT DMA CH REG, LSB  2ST DMA
0276 EB 00                      JMP     $+2                     ; I/O DELAY
0278 8A E0                      MOV     AH,AL                   ; SAVE LSB OF 16-BIT REGISTER
027A EC                         IN      AL,DX                   ; READ MSB OF DMA CH REGISTER
027B 3B D8                      CMP     BX,AX                   ; PATTERN READ AS WRITTEN?
027D 74 01                      JE      C18A                    ; YES - CHECK NEXT REGISTER
027F F4                         HLT                             ; NO - HALT THE SYSTEM
0280                    C18A:                                   ; NXT_DMA_CH
0280 83 C2 02                   ADD     DX,2                    ; SET I/O PORT TO NEXT CHANNEL REGISTER
0283 E2 E8                      LOOP    C17A                    ; WRITE PATTERN TO NEXT REGISTER
0285 FE C0                      INC     AL                      ; SET PATTERN TO 0
0287 74 DA                      JZ      C16A                    ; YES CONTINUE

                        ;-----  WRITE DMA WITH 55 PATTERN

0289 80 FB 55                   CMP     BL,55H                  ; CHECK IF 55 PATTERN DONE
028C 74 09                      JZ      C20A                    ; GO IF YES
028E 80 FB AA                   CMP     BL,0AAH                 ; CHECK IF AA PATTERN DONE
0291 74 08                      JZ      C21                     ; GO IF YES
0293 B0 55                      MOV     AL,55H
0295 EB CC                      JMP     C16A

                        ;-----  WRITE DMA WITH AA PATTERN

0297 B0 AA              C20A:   MOV     AL,0AAH
0299 EB C8                      JMP     C16A

                        ;-----  INITIALIZE AND START MEMORY REFRESH

029B                    C21:
029B 8B 1E 0072 R               MOV     BX,@RESET_FLAG          ; GET THE RESET FLAG
029F A3 0010 R                  MOV     @EQUIP_FLAG,AX          ; DO A DUMMY MEMORY WRITE BEFORE REFRESH
02A2 B0 12                      MOV     AL,18                   ; START REFRESH TIMER
02A4 E6 41                      OUT     TIMER+1,AL

                        ;-----  SET DMA COMMAND

02A6 2A C0                      SUB     AL,AL                   ; DACK SENSE LOW,DREQ SENSE HIGH
02A8 E6 08                      OUT     DMA+8,AL                ; LATE WRITE, FIXED PRIORITY, NORMAL
                                                                ; TIMING, CONTROLLER ENABLE, CH0 ADDRESS
                                                                ; HOLD DISABLE, MEMORY TO MEMORY DISABLE
02AA E6 D0                      OUT     DMA18,AL                ; SAME TO SECOND CONTROLLER
```

```

                            ;----- MODE SET ALL DMA CHANNELS

02AC B0 40                          MOV     AL,40H                  ; SET MODE FOR CHANNEL 0
02AE E6 0B                          OUT     DMA+0BH,AL
02B0 B0 C0                          MOV     AL,0C0H                 ; SET CASCADE MODE ON CHANNEL 4
02B2 E6 D6                          OUT     DMA18+06H,AL
02B4 EB 00                          JMP     $+2                     ; I/O DELAY
02B6 B0 41                          MOV     AL,41H                  ; SET MODE FOR CHANNEL 1
02B8 E6 0B                          OUT     DMA+0BH,AL
02BA E6 D6                          OUT     DMA18+06H,AL            ; SET MODE FOR CHANNEL 5
02BC EB 00                          JMP     $+2                     ; I/O DELAY
02BE B0 42                          MOV     AL,42H                  ; SET MODE FOR CHANNEL 2
02C0 E6 0B                          OUT     DMA+0BH,AL
02C2 E6 D6                          OUT     DMA18+06H,AL            ; SET MODE FOR CHANNEL 6
02C4 EB 00                          JMP     $+2                     ; I/O DELAY
02C6 B0 43                          MOV     AL,43H                  ; SET MODE FOR CHANNEL 3
02C8 E6 0B                          OUT     DMA+0BH,AL
02CA E6 D6                          OUT     DMA18+06H,AL            ; SET MODE FOR CHANNEL 7

                            ;----- RESTORE RESET FLAG

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

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

                            ;----- CHECKPOINT 08

02D0 B0 08                          MOV     AL,08H                  ;       <><><><><><><><><><><><><>
02D2 E6 80                          OUT     MFG_PORT,AL             ;       <><> CHECKPOINT 08 <><>
02D4 2A C0                          SUB     AL,AL
02D6 BA 0081                        MOV     DX,DMA_PAGE
02D9 B9 00FF                        MOV     CX,0FFH                 ; DO ALL DATA PATTERNS
02DC EE                     C22A:   OUT     DX,AL
02DD 42                             INC     DX
02DE FE C0                          INC     AL
02E0 81 FA 008F                     CMP     DX,8FH                  ; TEST DMA PAGES 81 THROUGH 8EH
02E4 75 F6                          JNZ     C22A
02E6 86 E0                          XCHG    AH,AL                   ; SAVE CURRENT DATA PATTERN
02E8 FE CC                          DEC     AH                      ; CHECK LAST WRITTEN
02EA 4A                             DEC     DX                      ;
02EB 2A C0                  C22B:   SUB     AL,AL                   ; CHANGE DATA BEFORE READ
02ED EC                             IN      AL,DX
02EE 3A C4                          CMP     AL,AH                   ; DATA AS WRITTEN?
02F0 75 30                          JNZ     C26                     ; GO ERROR HALT IF NOT
02F2 FE CC                          DEC     AH
02F4 4A                             DEC     DX
02F5 81 FA 0080                     CMP     DX,MFG_PORT             ; CONTINUE TILL PORT 80
02F9 75 F0                          JNZ     C22B
02FB FE C4                          INC     AH                      ; NEXT PATTERN TO RIPPLE
02FD 8A C4                          MOV     AL,AH
02FF E2 DB                          LOOP    C22A

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

0301 B0 CC                          MOV     AL,0CCH                 ; WRITE AN CC TO PAGE REGISTERS
0303 BA 008F                C22:    MOV     DX,LAST_DMA_PAGE
0306 8A E0                          MOV     AH,AL                   ; SAVE THE DATA PATTERN
0308 EE                             OUT     DX,AL                   ; OUTPUT PAGE REGISTER

                            ;----- VERIFY PAGE REGISTER 8F

0309 2A C0                          SUB     AL,AL                   ; CHANGE DATA PATTERN BEFORE READ
030B EC                             IN      AL,DX                   ; GET THE DATA FROM PAGE REGISTER
030C 3A C4                          CMP     AL,AH
030E 75 12                          JNZ     C26                     ; GO IF ERROR
0310 80 FC CC                       CMP     AH,0CCH
0313 75 04                          JNZ     C25                     ; GO IF ERROR
0315 B0 33                          MOV     AL,033H                 ; SET UP DATA PATTERN OF 33
0317 EB EA                          JMP     C22                     ; DO DATA 33
0319                        C25:
0319 80 FC 00                       CMP     AH,0                    ; CHECK DONE
031C 74 05                          JZ      C27                     ; GO IF YES
031E 2A C0                          SUB     AL,AL                   ; SET UP FOR DATA PATTERN 00
0320 EB E1                          JMP     C22                     ; DO DATA 0

                            ;----- ERROR HALT
0322                        C26:
0322 F4                             HLT                             ; HALT SYSTEM

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

                            ;----- CHECKPOINT 09 - TEST MEMORY REFRESH
0323                        C27:
0323 B0 09                          MOV     AL,09H                  ;       <><><><><><><><><><><><><>
0325 E6 80                          OUT     MFG_PORT,AL             ;       <><> CHECKPOINT 09 <><>
0327 2B C9                          SUB     CX,CX                   ;
0329                        C28:
0329 E4 61                          IN      AL,PORT_B               ; INSURE REFRESH BIT IS TOGGLING
032B A8 10                          TEST    AL,REFRESH_BIT
032D E1 FA                          LOOPZ   C28                     ; INSURE REFRESH IS OFF
032F E3 F1                          JCXZ    C26                     ; ERROR HALT IF TIMEOUT
0331                        C29:
0331 E4 61                          IN      AL,PORT_B
0333 A8 10                          TEST    AL,REFRESH_BIT          ; INSURE REFRESH IS ON
0335 E0 FA                          LOOPNZ  C29
0337 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.   :
```

SECTION 5

```
                                ;       INSURE A 55H IS RECEIVED.             :
                                ;       READ MANUFACTURING AND DISPLAY        :
                                ;       JUMPERS AND SAVE IN MFG_TEST.         :
                                ;---------------------------------------------

                                ;-----  CHECKPOINT 0A

 0339  B0 0A                            MOV     AL,0AH                  ;       <><><><><><><><><><><><><>
 033B  E6 80                            OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  0A <><>

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

 033D  2B C9                            SUB     CX,CX                   ; 100 MILLISECONDS FOR THIS LOOP
 033F  E4 64                    TST1:   IN      AL,STATUS_PORT          ; CHECK FOR INPUT BUFFER FULL
 0341  8A E0                            MOV     AH,AL
 0343  F6 C4 01                         TEST    AH,OUT_BUF_FULL
 0346  74 02                            JZ      TST2                    ; GO IF NOT
 0348  E4 60                            IN      AL,PORT_A               ; FLUSH
 034A  F6 C4 02                 TST2:   TEST    AH,INPT_BUF_FULL        ; IS THE OUTPUT BUFFER ALSO FULL?
 034D  E0 F0                            LOOPNZ  TST1                    ; TRY AGAIN
 034F  74 01                            JZ      TST4                    ; CONTINUE IF OK

 0351  F4                       ERR0:   HLT                             ; HALT SYSTEM IF BUFFER FULL

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

 0352  B0 0B                    TST4:   MOV     AL,0BH                  ;       <><><><><><><><><><><><><>
 0354  E6 80                            OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  0B <><>

 0356  B0 AA                            MOV     AL,SELF_TEST            ; SELF TEST COMMAND
 0358  BC 03EE R                        MOV     SP,OFFSET C8042A        ; SET RETURN ADDRESS
 035B  EB 39                            JMP     SHORT C8042
 035D  A8 01                    TST4_B: TEST    AL,OUT_BUF_FULL         ; IS THE OUTPUT BUFFER FULL?
 035F  74 02                            JZ      TST4_A                  ; GO IF NOT
 0361  E4 60                            IN      AL,PORT_A               ; FLUSH
 0363  BC 03F0 R                TST4_A: MOV     SP,OFFSET OBF_42A       ; SET RETURN ADDRESS
 0366  EB 3A                            JMP     SHORT OBF_42            ; GO WAIT FOR BUFFER
 0368  E4 60                    TST4_C: IN      AL,PORT_A               ; GET THE ENDING RESPONSE
 036A  3C 55                            CMP     AL,55H

 036C  B0 0C                            MOV     AL,0CH                  ;       <><><><><><><><><><><><><>
 036E  E6 80                            OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  0C <><>

 0370  75 DF                            JNZ     ERR0                    ; GO IF NOT OK

                                ;-----  GET THE SWITCH SETTINGS

 0372  B0 C0                            MOV     AL,READ_8042_INPUT      ; READ INPUT COMMAND
 0374  BC 03F4 R                        MOV     SP,OFFSET C8042C        ; SET RETURN ADDRESS
 0377  EB 1D                            JMP     SHORT C8042             ; ISSUE COMMAND
 0379  BC 03F6 R                E30B:   MOV     SP,OFFSET OBF_42B       ; SET RETURN ADDRESS
 037C  EB 24                            JMP     SHORT OBF_42            ; GO WAIT FOR RESPONSE
 037E  E4 60                    E30C:   IN      AL,PORT_A               ; GET THE SWITCH
 0380  E6 82                            OUT     DMA_PAGE+1,AL           ; SAVE TEMPORARY

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

 0382  B0 60                            MOV     AL,WRITE_8042_LOC       ; WRITE BYTE COMMAND
 0384  BC 03F2 R                        MOV     SP,OFFSET C8042B        ; SET RETURN ADDRESS
 0387  EB 0D                            JMP     SHORT C8042             ; ISSUE THE COMMAND
 0389  74 05                    TST4_D: JZ      TST4_D1                 ; CONTINUE IF COMMAND ACCEPTED

 038B  B0 0D                            MOV     AL,0DH                  ;       <><><><><><><><><><><><><>
 038D  E6 80                            OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  0D <><>
 038F  F4                               HLT
 0390                           TST4_D1:
 0390  B0 5D                            MOV     AL,5DH                  ; ENABLE OUTPUT BUFFER FULL INTERRUPT,
 0392  E6 60                            OUT     PORT_A,AL               ; DISABLE KEYBOARD, SET SYSTEM FLAG,
 0394  EB 1D                            JMP     SHORT E30A              ; PC 1 COMPATIBILITY, INHIBIT OVERRIDE

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

 0396  FA                       C8042:  CLI                             ; NO INTERRUPTS ALLOWED
 0397  E6 64                            OUT     STATUS_PORT,AL          ; SEND COMMAND IN AL REGISTER

 0399  2B C9                            SUB     CX,CX                   ; LOOP COUNT
 039B  E4 64                    C42_1:  IN      AL,STATUS_PORT          ; WAIT FOR THE COMMAND ACCEPTED
 039D  A8 02                            TEST    AL,INPT_BUF_FULL
 039F  E0 FA                            LOOPNZ  C42_1
 03A1  C3                               RET

                                ;-----  WAIT FOR 8042 RESPONSE

 03A2  2B C9                    OBF_42: SUB     CX,CX
 03A4  B3 06                            MOV     BL,6                    ; 200MS/PER LOOP * 6  =1200 MS +
 03A6  E4 64                    C42_2:  IN      AL,STATUS_PORT          ; CHECK FOR RESPONSE
 03A8  A8 01                            TEST    AL,OUT_BUF_FULL
 03AA  75 06                            JNZ     C42_3                   ; GO IF RESPONSE
 03AC  E2 F8                            LOOP    C42_2                   ; TRY AGAIN
 03AE  FE CB                            DEC     BL                      ; DECREMENT LOOP COUNT
 03B0  75 F4                            JNZ     C42_2
 03B2  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

 03B3  B0 0E                    E30A:   MOV     AL,0EH                  ;       <><><><><><><><><><><><><>
 03B5  E6 80                            OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  0E <><>

 03B7  B8 ---- R                        MOV     AX,DATA                 ; GET THE SYSTEM SEGMENT
 03BA  8E D8                            MOV     DS,AX                   ;  OF DATA
 03BC  8B 1E 0072 R                     MOV     BX,@RESET_FLAG          ; SAVE @RESET_FLAG IN BX
 03C0  FC                               CLD                             ; SET DIRECTION FLAG TO INCREMENT
 03C1  B9 8000                          MOV     CX,2000H*4              ; SET FOR 32K WORDS
 03C4  2B FF                            SUB     DI,DI                   ; FIRST 16K
```

```
03C6 2B F6                      SUB     SI,SI
03C8 2B C0                      SUB     AX,AX
03CA 8E D8                      MOV     DS,AX
03CC 8E C0                      MOV     ES,AX
03CE 81 FB 1234                 CMP     BX,1234H                ; WARM START?
03D2 75 03                      JNZ     E30A_0                  ; GO IF NOT
03D4 E9 0582 R                  JMP     CLR_STG

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

03D7 B0 0F              E30A_0: MOV     AL,0FH                  ;       <><><><><><><><><><><><><>
03D9 E6 80                      OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  0F <><>

03DB B0 80                      MOV     AL,PARITY_CHECK         ; SET BASE MEMORY PARITY
03DD E6 87                      OUT     DMA_PAGE+6,AL           ; USE AS TEMPORARY SAVE
03DF BC 03EC R                  MOV     SP,OFFSET C2            ; SET RETURN ADDRESS
03E2 E9 0000 E                  JMP     STGTST_CNT
03E5 8B D8              C30:    MOV     BX,AX                   ; SAVE FAILING BIT PATTERN
03E7 75 0F                      JNZ     C31
03E9 E9 058D R                  JMP     C33                     ; STORAGE OK, CONTINUE

                        ;-----  TEMPORARY STACK FOR POST ROUTINES

03EC 03E5 R             C2      DW      C30
03EE 035D R             C8042A  DW      TST4_B
03F0 0368 R             OBF_42A DW      TST4_C
03F2 0389 R             C8042B  DW      TST4_D
03F4 0379 R             C8042C  DW      E30B
03F6 037E R             OBF_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
                        ;----------------------------------------

03F8                    C31:
03F8 8A C7                      MOV     AL,BH                   ; SAVE HIGH BYTE
03FA E6 81                      OUT     MFG_PORT+1,AL
03FC 8A C3                      MOV     AL,BL                   ; SAVE LOW BYTE
03FE E6 82                      OUT     MFG_PORT+2,AL

                        ;-----  CHECK FOR VIDEO ROM

0400 B9 C000                    MOV     CX,0C000H               ; START OF I/O ROM
0403 8E D9              M1:     MOV     DS,CX                   ; POINT TO SEGMENT
0405 2B DB                      SUB     BX,BX                   ; GET THE FIRST 2 LOCATIONS
0407 8B 07                      MOV     AX,[BX]
0409 EB 00                      JMP     $+2                     ; BUS SETTLE
040B 3D AA55                    CMP     AX,0AA55H               ; IS THE VIDEO ROM PRESENT?
040E 6A 00                      POP
0410 74 0C                      JZ      Z5                      ; GO IF YES
0412 81 C1 0080                 ADD     CX,080H                 ; POINT TO NEXT 2K BLOCK
0416 81 F9 C800                 CMP     CX,0C800H               ; TOP OF VIDEO ROM AREA YET?
041A 7C E7                      JL      M1                      ; TRY AGAIN
041C 23 C9                      AND     CX,CX                   ; SET NON ZERO FLAG
041E                    Z5:
041E 75 03                      JNZ     C32                     ; GO IF NOT
0420 E9 050F R                  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

0423 BA 03D8            C32:    MOV     DX,3D8H                 ; CONTROL REGISTER ADDRESS OF COLOR CARD
0426 2A C0                      SUB     AL,AL                   ; MODE SET
0428 EE                         OUT     DX,AL

0429 BA 03B8                    MOV     DX,03B8H                ; CONTROL REGISTER ADDRESS OF B/W CARD
042C B0 01                      MOV     AL,1                    ; MODE SET FOR CARD
042E EE                         OUT     DX,AL                   ; RESET VIDEO
042F 83 EA 04                   SUB     DX,4                    ; BACK TO BASE REGISTER

= 0010                  M4      EQU     10H

0432 BB 0030 E                  MOV     BX,OFFSET VIDEO_PARMS+M4*3  ; POINT TO VIDEO PARAMETERS
                                ASSUME  DS:CODE
0435 B9 0010            Z_2:    MOV     CX,M4                   ; COUNT OF MONOCHROME VIDEO PARAMETERS

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

0438 32 E4                      XOR     AH,AH                   ; AH IS REGISTER NUMBER DURING LOOP

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

043A 8A C4              M10:    MOV     AL,AH                   ; GET 6845 REGISTER NUMBER
043C EE                         OUT     DX,AL
043D 42                         INC     DX                      ; POINT TO DATA PORT
043E FE C4                      INC     AH                      ; NEXT REGISTER VALUE
0440 2E: 8A 07                  MOV     AL,CS:[BX]              ; GET TABLE VALUE
0443 EE                         OUT     DX,AL                   ; OUT TO CHIP
0444 43                         INC     BX                      ; NEXT IN TABLE
0445 4A                         DEC     DX                      ; BACK TO POINTER REGISTER
0446 E2 F2                      LOOP    M10                     ; DO THE WHOLE TABLE
0448 8A E2                      MOV     AH,DL                   ; CHECK IF COLOR CARD DONE
044A 80 E4 F0                   AND     AH,0F0H                 ; STRIP UNWANTED BITS
044D 80 FC D0                   CMP     AH,0D0H                 ; IS IT THE COLOR CARD?
0450 74 08                      JZ      Z_3                     ; CONTINUE IF COLOR
0452 BB 0000 E                  MOV     BX,OFFSET VIDEO_PARMS   ; POINT TO VIDEO PARAMETERS
0455 BA 03D4                    MOV     DX,3D4H                 ; COLOR BASE
0458 EB DB                      JMP     Z_2                     ; CONTINUE

                        ;-----  FILL REGEN AREA WITH BLANK
```

```
0045A 33 FF            Z_3:    XOR     DI,DI                   ; SET UP POINTER FOR REGEN
045C B8 B000                   MOV     AX,0B000H               ; SET UP ES TO VIDEO REGEN
045F 8E C0                     MOV     ES,AX

0461 B9 0800                   MOV     CX,2048                 ; NUMBER OF WORDS IN MONOCHROME CARD
0464 B8 0720                   MOV     AX,' '+7*H              ; FILL CHARACTER FOR ALPHA + ATTRIBUTE
0467 F3/ AB                    REP     STOSW                   ; FILL THE REGEN BUFFER WITH BLANKS

0469 33 FF                     XOR     DI,DI                   ; CLEAR COLOR VIDEO BUFFER MEMORY
046B BB B800                   MOV     BX,0B800H               ; SET UP ES TO COLOR VIDEO MEMORY
046E 8E C3                     MOV     ES,BX
0470 B9 2000                   MOV     CX,8192
0473 F3/ AB                    REP     STOSW                   ; FILL WITH BLANKS

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

0475 BA 03B8                   MOV     DX,3B8H
0478 B0 29                     MOV     AL,29H
047A EE                        OUT     DX,AL                   ; SET VIDEO ENABLE PORT

                       ;-----  SET UP OVERSCAN REGISTER

047B 42                        INC     DX                      ; SET OVERSCAN PORT TO A DEFAULT
047C B0 30                     MOV     AL,30H                  ; VALUE 30H FOR ALL MODES EXCEPT 640X200
047E EE                        OUT     DX,AL                   ; OUTPUT THE CORRECT VALUE TO 3D9 PORT

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

047F BA 03D8                   MOV     DX,3D8H
0482 B0 28                     MOV     AL,28H
0484 EE                        OUT     DX,AL                   ; SET VIDEO ENABLE PORT

                       ;-----  SET UP OVERSCAN REGISTER

0485 42                        INC     DX                      ; SET OVERSCAN PORT TO A DEFAULT
0486 B0 30                     MOV     AL,30H                  ; VALUE 30H FOR ALL MODES EXCEPT 640X200
0488 EE                        OUT     DX,AL                   ; OUTPUT THE CORRECT VALUE TO 3D9 PORT

                       ;-----  DISPLAY FAILING CHECKPOINT AND

0489 8C C8                     MOV     AX,CS                   ; SET STACK SEGMENT TO CODE SEGMENT
048B 8E D0                     MOV     SS,AX

048D BB B000                   MOV     BX,0B000H
0490 8E DB                     MOV     DS,BX                   ; SET DS TO B/W DISPLAY BUFFER

0492 B0 30             Z_0:    MOV     AL,'0'                  ; DISPLAY BANK 000000
0494 B9 0006                   MOV     CX,6
0497 2B FF                     SUB     DI,DI                   ; START AT 0
0499 88 05             Z:      MOV     [DI],AL                 ; WRITE TO DISPLAY REGEN BUFFER
049B 47                        INC     DI                      ; POINT TO NEXT POSITION
049C 47                        INC     DI
049D E2 FA                     LOOP    Z

049F 80 FF B8                  CMP     BH,0B8H                 ; CHECK THAT COLOR BUFFER WRITTEN
04A2 74 0C                     JZ      Z_1
04A4 2B FF                     SUB     DI,DI                   ; POINT TO START OF BUFFER

04A6 B7 B0                     MOV     BH,0B0H
04A8 8E C3                     MOV     ES,BX                   ; ES = MONOCHROME
04AA B7 B8                     MOV     BH,0B8H                 ; SET SEGMENT TO COLOR
04AC 8E DB                     MOV     DS,BX                   ; DS = COLOR
04AE EB E2                     JMP     Z_0

                       ;-----  PRINT FAILING BIT PATTERN

04B0 B0 20             Z_1:    MOV     AL,' '                  ; DISPLAY A BLANK
04B2 88 05                     MOV     [DI],AL                 ; WRITE TO COLOR BUFFER
04B4 26: 88 05                 MOV     ES:[DI],AL              ; WRITE TO MONOCHROME REGEN BUFFER
04B7 47                        INC     DI                      ; POINT TO NEXT POSITION
04B8 47                        INC     DI
04B9 E4 81                     IN      AL,MFG_PORT+1           ; GET THE HIGH BYTE OF FAILING PATTERN
04BB B1 04                     MOV     CL,4                    ; SHIFT COUNT
04BD D2 E8                     SHR     AL,CL                   ; NIBBLE SWAP
04BF BC 057A R                 MOV     SP,OFFSET Z1_0
04C2 EB 1B                     JMP     SHORT PR

04C4 E4 81             Z1:     IN      AL,MFG_PORT+1
04C6 24 0F                     AND     AL,0FH                  ; ISOLATE TO LOW NIBBLE
04C8 BC 057C R                 MOV     SP,OFFSET Z2_0
04CB EB 12                     JMP     SHORT PR
04CD E4 82             Z2:     IN      AL,MFG_PORT+2           ; GET THE HIGH BYTE OF FAILING PATTERN
04CF B1 04                     MOV     CL,4                    ; SHIFT COUNT
04D1 D2 E8                     SHR     AL,CL                   ; NIBBLE SWAP
04D3 BC 057E R                 MOV     SP,OFFSET Z3_0
04D6 EB 07                     JMP     SHORT PR
04D8 E4 82             Z3:     IN      AL,MFG_PORT+2
04DA 24 0F                     AND     AL,0FH                  ; ISOLATE TO LOW NIBBLE
04DC BC 0580 R                 MOV     SP,OFFSET Z4_0          ; RETURN TO Z4:

                       ;-----  CONVERT AND PRINT
                                                               ; CONVERT 00-0F TO ASCII CHARACTER
04DF 04 90             PR:     ADD     AL,090H                 ; ADD FIRST CONVERSION FACTOR
04E1 27                        DAA                             ; ADJUST FOR NUMERIC AND ALPHA RANGE
04E2 14 40                     ADC     AL,040H                 ; ADD CONVERSION AND ADJUST LOW NIBBLE
04E4 27                        DAA                             ; ADJUST HIGH NIBBLE TO ASCII RANGE

04E5 88 05                     MOV     [DI],AL                 ; WRITE TO COLOR BUFFER
04E7 26: 88 05                 MOV     ES:[DI],AL              ; WRITE TO MONOCHROME BUFFER
04EA 47                        INC     DI                      ; POINT TO NEXT POSITION
04EB 47                        INC     DI
04EC C3                        RET

                       ;-----  DISPLAY 201 ERROR

04ED B0 20             Z4:     MOV     AL,' '                  ; DISPLAY A BLANK
04EF 88 05                     MOV     [DI],AL                 ; WRITE TO DISPLAY REGEN BUFFER
04F1 26: 88 05                 MOV     ES:[DI],AL              ; WRITE TO MONOCHROME BUFFER
04F4 47                        INC     DI                      ; POINT TO NEXT POSITION
04F5 47                        INC     DI
04F6 B0 32                     MOV     AL,'2'                  ; DISPLAY 201 ERROR
04F8 88 05                     MOV     [DI],AL                 ; WRITE TO DISPLAY REGEN BUFFER
04FA 26: 88 05                 MOV     ES:[DI],AL              ; WRITE TO MONOCHROME BUFFER
04FD 47                        INC     DI                      ; POINT TO NEXT POSITION
```

```
04FE 47                                 INC     DI
04FF B0 30                              MOV     AL,'0'
0501 88 05                              MOV     [DI],AL                 ; WRITE TO DISPLAY REGEN BUFFER
0503 26: 88 05                          MOV     ES:[DI],AL              ; WRITE TO MONOCHROME BUFFER
0506 47                                 INC     DI                      ; POINT TO NEXT POSITION
0507 47                                 INC     DI
0508 B0 31                              MOV     AL,'1'
050A 88 05                              MOV     [DI],AL                 ; WRITE TO DISPLAY REGEN BUFFER
050C 26: 88 05                          MOV     ES:[DI],AL              ; WRITE TO MONOCHROME BUFFER

                                ;-----  ROLL ERROR CODE IN MFG_PORT --> FIRST THE CHECKPOINT

050F B0 DD                      C31_0:  MOV     AL,0DDH                 ;       <><><><><><><><><><><><><>
0511 E6 80                              OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  DD <><>
0513 E6 83                              OUT     MFG_PORT+3,AL           ; ALSO DISPLAY CHECK POINT IN PORT 83
0515 2B C9                              SUB     CX,CX
0517                            C31_A:

0517 2B C0                              SUB     AX,AX                   ; SETUP SEGMENT
0519 8E D8                              MOV     DS,AX
051B B8 AA55                            MOV     AX,0AA55H               ; WRITE AN AA55
051E 2B FF                              SUB     DI,DI
0520 89 05                              MOV     [DI],AX
0522 8B 05                              MOV     AX,[DI]                 ; READ THE FIRST WORD
0524 E2 F1                              LOOP    C31_A                   ; DISPLAY CHECKPOINT LONGER
0526                            C31_B:
0526 89 05                              MOV     [DI],AX
0528 8B 05                              MOV     AX,[DI]
052A E2 FA                              LOOP    C31_B
052C                            C31_C:
052C 89 05                              MOV     [DI],AX
052E 8B 05                              MOV     AX,[DI]
0530 E2 FA                              LOOP    C31_C
0532                            C31_D:
0532 89 05                              MOV     [DI],AX
0534 8B 05                              MOV     AX,[DI]
0536 E2 FA                              LOOP    C31_D
0538                            C31_E:
0538 89 05                              MOV     [DI],AX
053A 8B 05                              MOV     AX,[DI]
053C E2 FA                              LOOP    C31_E

                                ;-----  ROLL ERROR CODE IN MFG_PORT --> NEXT THE HIGH BYTE

053E E4 81                              IN      AL,MFG_PORT+1           ; XOR OF FAILING BIT PATTERN
0540 E6 80                              OUT     MFG_PORT,AL             ; HIGH BYTE
0542                            C31_G:
0542 B8 AA55                            MOV     AX,0AA55H               ; WRITE AN AA55
0545 89 05                              MOV     [DI],AX
0547 8B 05                              MOV     AX,[DI]                 ; READ THE FIRST WORD
0549 E2 F7                              LOOP    C31_G
054B                            C31_H:
054B 89 05                              MOV     [DI],AX
054D 8B 05                              MOV     AX,[DI]
054F E2 FA                              LOOP    C31_H
0551                            C31_I:
0551 89 05                              MOV     [DI],AX
0553 8B 05                              MOV     AX,[DI]
0555 E2 FA                              LOOP    C31_I

                                ;-----  ROLL ERROR CODE IN MFG_PORT --> THEN THE LOW BYTE

0557 E4 82                              IN      AL,MFG_PORT+2           ; LOW BYTE
0559 E6 80                              OUT     MFG_PORT,AL
055B B8 AA55                            MOV     AX,0AA55H               ; WRITE AN AA55
055E 2B FF                      C31_K:  SUB     DI,DI
0560 89 05                              MOV     [DI],AX
0562 8B 05                              MOV     AX,[DI]                 ; READ THE FIRST WORD
0564 E2 F8                              LOOP    C31_K
0566                            C31_L:
0566 89 05                              MOV     [DI],AX
0568 8B 05                              MOV     AX,[DI]
056A E2 FA                              LOOP    C31_L
056C                            C31_M:
056C 89 05                              MOV     [DI],AX
056E 8B 05                              MOV     AX,[DI]
0570 E2 FA                              LOOP    C31_M
0572                            C31_N:
0572 89 05                              MOV     [DI],AX
0574 8B 05                              MOV     AX,[DI]
0576 E2 FA                              LOOP    C31_N
0578 EB 95                              JMP     C31_0                   ; DO AGAIN

057A 04C4 R                     Z1_0    DW      Z1                      ; TEMPORARY STACK
057C 04CD R                     Z2_0    DW      Z2                      ; TEMPORARY STACK
057E 04D8 R                     Z3_0    DW      Z3                      ; TEMPORARY STACK
0580 04ED R                     Z4_0    DW      Z4                      ; TEMPORARY STACK


                                ;-----  CLEAR STORAGE ENTRY


0582                            CLR_STG:
                                        ASSUME  DS:DATA
0582 F3/ AB                             REP     STOSW                   ; STORE 32K WORDS OF 0000
0584 B8 ---- R                          MOV     AX,DATA                 ; RESTORE DATA SEGMENT
0587 8E D8                              MOV     DS,AX
0589 89 1E 0072 R                       MOV     @RESET_FLAG,BX          ; RESTORE RESET FLAG

                                ;-----  SETUP STACK SEGMENT AND SP

058D                            C33:
058D B8 ---- R                          MOV     AX,DATA                 ; SET DATA SEGMENT
0590 8E D8                              MOV     DS,AX
0592 BC 0000                            MOV     SP,POST_SS              ; GET STACK VALUE
0595 8E D4                              MOV     SS,SP                   ; SET THE STACK UP
0597 BC 8000                            MOV     SP,POST_SP              ; STACK IS READY TO GO

                                ;-----  INITIALIZE DISPLAY ROW COUNT

059A C6 06 0084 R 18                    MOV     @ROWS,25-1              ; SET ROWS FOR PRINT SCREEN DEFAULT

059F B0 11                              MOV     AL,11H                  ;       <><><><><><><><><><><><><>
05A1 E6 80                              OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  11 <><>
```

SECTION 5

```
                                 ;-----  VERIFY SPEED/REFRESH CLOCK RATES  ( ERROR = 1 LONG AND 1 SHORT BEEP )

05A3 32 DB                               XOR     BL,BL                   ; CLEAR REFRESH CYCLE REPEAT COUNT
05A5 33 C9                               XOR     CX,CX                   ; INITIALIZE SPEED RATE REGISTER
05A7 90                                  EVEN                            ; PLACE ON EVEN WORD BOUNDARY
05A8                             C34:
05A8 E4 61                               IN      AL,PORT_B               ; READ REFRESH BIT REGISTER
05AA A8 10                               TEST    AL,REFRESH_BIT          ; MASK FOR BIT
05AC E1 FA                               LOOPZ   C34                     ; DECREMENT LOOP COUNTER TILL ON
05AE                             C35:
05AE E4 61                               IN      AL,PORT_B               ; READ REFRESH BIT REGISTER
05B0 A8 10                               TEST    AL,REFRESH_BIT          ; MASK FOR BIT
05B2 E0 FA                               LOOPNZ  C35                     ; DECREMENT LOOP COUNTER TILL OFF

05B4 FE CB                               DEC     BL                      ; DECREMENT REFRESH CYCLE REPEAT COUNT

05B6 75 F0                               JNZ     C34                     ;  REPEAT TILL CYCLE COUNT DONE    ;
05B8 81 F9 F600                          CMP     CX,RATE_UPPER           ;  CHECK FOR RATE BELOW UPPER LIMIT;
05BC 73 07                               JAE     C36                     ;  SKIP ERROR BEEP IF BELOW MAXIMUM;

05BE                             C36E:
05BE BA 0101                             MOV     DX,0101H                ; GET BEEP COUNTS FOR REFRESH ERROR
05C1 E8 0000 E                           CALL    ERR_BEEP                ; CALL TO POST ERROR BEEP ROUTINES
05C4 F4                                  HLT                             ; HALT SYSTEM - BAD REFRESH RATE
05C5                             C36:
05C5 81 F9 F9FD                          CMP     CX,RATE_LOWER           ; CHECK FOR RATE ABOVE LOWER LIMIT
05C9 77 F3                               JA      C36E                    ; GO TO ERROR BEEP IF BELOW MINIMUM

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

05CB E4 82                               IN      AL,DMA_PAGE+1           ; GET THE SWITCH SETTINGS
05CD 24 F8                               AND     AL,KEY_BD_INHIB+DSP_JMP+MFG_LOOP+BASE_MEM+BASE_MEM8 ; STRIP BITS
05CF A2 0012 R                           MOV     @MFG_TST,AL             ; SAVE SETTINGS
05D2 2A C0                               SUB     AL,AL                   ; RESET DMA_PAGE
05D4 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
05D6 0F 01 E0               +            DB      00FH,001H,0E0H
05D9 A9 000F                             TEST    AX,0FH                  ; PE/MP/EM/TS BITS SHOULD BE ZERO
05DC 75 34                               JNZ     ERR_PROT                ; GO IF STATUS NOT REAL MODE

                                 ;-----  TEST PROTECTED MODE REGISTERS

05DE B0 12                               MOV     AL,12H                  ;       <><><><><><><><><><><><>
05E0 E6 80                               OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  12 <><>

05E2 1E                                  PUSH    DS                      ; SET ES TO SAME SEGMENT AS DS
05E3 07                                  POP     ES
05E4 BF D0A0                             MOV     DI,SYS_IDT_LOC          ; USE THIS AREA TO BUILD TEST PATTERN
05E7 B9 0003                             MOV     CX,3
05EA B8 AAAA                             MOV     AX,0AAAAH               ; FIRST PATTERN
05ED E8 0615 R                           CALL    WRT_PAT
05F0 B8 5555                             MOV     AX,05555H
05F3 E8 0615 R                           CALL    WRT_PAT                 ; WRITE NEXT PATTERN
05F6 2B C0                               SUB     AX,AX                   ; WRITE 0
05F8 E8 0615 R                           CALL    WRT_PAT

                                 ;-----  TEST 286 CONTROL FLAGS

05FB FD                                  STD                             ; SET DIRECTION FLAG FOR DECREMENT
05FC 9C                                  PUSHF                           ; GET THE FLAGS
05FD 58                                  POP     AX
05FE A9 0200                             TEST    AX,0200H                ; INTERRUPT FLAG SHOULD BE OFF
0601 75 0F                               JNZ     ERR_PROT                ; GO IF NOT
0603 A9 0400                             TEST    AX,0400H                ; CHECK DIRECTION FLAG
0606 74 0A                               JZ      ERR_PROT                ; GO IF NOT SET
0608 FC                                  CLD                             ; CLEAR DIRECTION FLAG
0609 9C                                  PUSHF                           ; INSURE DIRECTION FLAG IS RESET
060A 58                                  POP     AX
060B A9 0400                             TEST    AX,0400H
060E 75 02                               JNZ     ERR_PROT                ; GO IF NOT

0610 EB 3D                               JMP     SHORT C37A              ; TEST OK CONTINUE
0612                             ERR_PROT:
0612 F4                                  HLT                             ; PROTECTED MODE REGISTER FAILURE
0613 EB FD                               JMP     SHORT ERR_PROT          ; INSURE NO BREAKOUT OF HALT

                                 ;-----  WRITE TO 286 REGISTERS

0615 B9 0003                     WRT_PAT:MOV     CX,3
0618 F3/ AB                              REP     STOSW                   ; STORE 6 BYTES OF PATTERN
061A BD D0A0                             MOV     BP,SYS_IDT_LOC
                                         SEGOV   ES                      ; LOAD THE IDT
061D 26                     +            DB      026H
                                         LIDT    [BP]                    ; REGISTER FROM THIS AREA
061E 0F                     +            DB      00FH
061F                        + ??0001     LABEL   BYTE
061F 8B 5E 00               +            MOV     BX,WORD PTR [BP]
0622                        + ??0002     LABEL   BYTE
```

```
061F                 +           ORG     OFFSET CS:??0001
061F 01              +           DB      001H
0622                 +           ORG     OFFSET CS:??0002
0622 BD D0A0                     MOV     BP,SYS_IDT_LOC
                                 SEGOV   ES                      ; LOAD THE GDT
0625 26              +           DB      026H
                                 LGDT    [BP]                    ; FROM THE SAME AREA
0626 0F              +           DB      00FH
0627                 + ??0004    LABEL   BYTE
0627 8B 56 00        +           MOV     DX,WORD PTR [BP]
062A                 + ??0005    LABEL   BYTE
0627                 +           ORG     OFFSET CS:??0004
0627 01              +           DB      001H
062A                 +           ORG     OFFSET CS:??0005

                       ;-----    READ AND VERIFY 286 REGISTERS

062A BD D8A0                     MOV     BP,GDT_LOC              ; STORE THE REGISTERS HERE
                                 SEGOV   ES
062D 26              +           DB      026H
                                 SIDT    [BP]                    ; GET THE IDT REGISTERS
062E 0F              +           DB      00FH
062F                 + ??0007    LABEL   BYTE
062F 8B 4E 00        +           MOV     CX,[BP]
0632                 + ??0008    LABEL   BYTE
062F                 +           ORG     OFFSET CS:??0007
062F 01              +           DB      001H
0632                 +           ORG     OFFSET CS:??0008
0632 BD D8A5                     MOV     BP,GDT_LOC+5
                                 SEGOV   ES
0635 26              +           DB      026H
                                 SGDT    [BP]                    ; GET THE GDT REGISTERS
0636 0F              +           DB      00FH
0637                 + ??000A    LABEL   BYTE
0637 03 46 00        +           ADD     AX,[BP]
063A                 + ??000B    LABEL   BYTE
0637                 +           ORG     OFFSET CS:??000A
0637 01              +           DB      001H
063A                 +           ORG     OFFSET CS:??000B
063A BF D0A0                     MOV     DI,SYS_IDT_LOC
063D 8B 05                       MOV     AX,[DI]                 ; GET THE PATTERN WRITTEN
063F B9 0005                     MOV     CX,5                    ; CHECK ALL REGISTERS
0642 BE D8A0                     MOV     SI,GDT_LOC              ; POINT TO THE BEGINNING
0645 26: 3B 04         C37B:     CMP     AX,ES:[SI]
0648 75 C8                       JNZ     ERR_PROT                ; HALT IF ERROR
064A 46                          INC     SI                      ; POINT TO NEXT WORD
064B 46                          INC     SI
064C E2 F7                       LOOP    C37B                    ; CONTINUE TILL DONE
064E C3                          RET


                       ;--------------------------------------------------------
                       ;         INITIALIZE THE 8259 INTERRUPT #1 CONTROLLER CHIP :
                       ;--------------------------------------------------------
064F                   C37A:
064F 2A C0                       SUB     AL,AL                   ; RESET MATH PROCESSOR
0651 E6 F1                       OUT     X287+1,AL
0653 B0 11                       MOV     AL,11H                  ; ICW1 - EDGE, MASTER, ICW4
0655 E6 20                       OUT     INTA00,AL
0657 EB 00                       JMP     $+2                     ; I/O DELAY
0659 B0 08                       MOV     AL,8                    ; SETUP ICW2 - INTERRUPT TYPE 8 (8-F)
065B E6 21                       OUT     INTA01,AL
065D EB 00                       JMP     $+2                     ; I/O DELAY

065F B0 04                       MOV     AL,04H                  ; SETUP ICW3 - MASTER LEVEL 2
0661 E6 21                       OUT     INTA01,AL
0663 EB 00                       JMP     $+2                     ; I/O DELAY
0665 B0 01                       MOV     AL,01H                  ; SETUP ICW4 - MASTER,8086 MODE
0667 E6 21                       OUT     INTA01,AL
0669 EB 00                       JMP     $+2                     ; I/O DELAY
066B B0 FF                       MOV     AL,0FFH                 ; MASK ALL INTERRUPTS OFF
066D E6 21                       OUT     INTA01,AL               ; (VIDEO ROUTINE ENABLES INTERRUPTS)

                       ;--------------------------------------------------------
                       ;         INITIALIZE THE 8259 INTERRUPT #2 CONTROLLER CHIP :
                       ;--------------------------------------------------------

066F B0 13                       MOV     AL,13H                  ;       <><><><><><><><><><><><><>
0671 E6 80                       OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  13 <><>

0673 B0 11                       MOV     AL,11H                  ; ICW1 - EDGE, SLAVE ICW4
0675 E6 A0                       OUT     INTB00,AL
0677 EB 00                       JMP     $+2                     ; I/O DELAY
0679 B0 70                       MOV     AL,INT_TYPE             ; SETUP ICW2 - INTERRUPT TYPE 70 (70-7F)
067B E6 A1                       OUT     INTB01,AL
067D B0 02                       MOV     AL,02H                  ; SETUP ICW3 - SLAVE LEVEL 2
067F EB 00                       JMP     $+2
0681 E6 A1                       OUT     INTB01,AL
0683 EB 00                       JMP     $+2                     ; I/O DELAY
0685 B0 01                       MOV     AL,01H                  ; SETUP ICW4 - 8086 MODE, SLAVE
0687 E6 A1                       OUT     INTB01,AL
0689 EB 00                       JMP     $+2                     ; I/O DELAY
068B B0 FF                       MOV     AL,0FFH                 ; MASK ALL INTERRUPTS OFF
068D E6 A1                       OUT     INTB01,AL

                       ;-----    SET UP THE INTERRUPT VECTORS TO TEMPORARY INTERRUPT

068F B0 14                       MOV     AL,14H                  ;       <><><><><><><><><><><><><>
0691 E6 80                       OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  14 <><>

0693 B9 0078                     MOV     CX,78H                  ; FILL ALL INTERRUPT LOCATIONS
0696 2B FF                       SUB     DI,DI                   ; FIRST INTERRUPT LOCATION
0698 8E C7                       MOV     ES,DI                   ; SET (ES) ALSO
069A B8 0000 E         D3:       MOV     AX,OFFSET D11           ; GET ADDRESS OF INTERRUPT OFFSET
069D AB                          STOSW                           ; PLACE IN INTERRUPT VECTOR LOCATION
069E 8C C8                       MOV     AX,CS                   ; GET THE CURRENT CODE SEGMENT
06A0 AB                          STOSW                           ; PLACE CODE SEGMENT IN VECTOR LOCATION
06A1 E2 F7                       LOOP    D3

                       ;-----    ESTABLISH BIOS SUBROUTINE CALL INTERRUPT VECTORS

06A3 B0 15                       MOV     AL,15H                  ;       <><><><><><><><><><><><><>
06A5 E6 80                       OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  15 <><>
```

```
06A7 BF 0040 R                          MOV     DI,OFFSET @VIDEO_INT    ; SET VIDEO INTERRUPT AREA
06AA 0E                                 PUSH    CS
06AB 1F                                 POP     DS                      ; SET UP ADDRESS OF VECTOR TABLE
                                ;       MOV     AX,DS                   ; SET AX=SEGMENT
06AC BE 0010 E                          MOV     SI,OFFSET VECTOR_TABLE+16 ;START WITH VIDEO ENTRY
06AF B9 0010                            MOV     CX,16

06B2 A5                         D3A:    MOVSW                           ; MOVE VECTOR TABLE TO LOW MEMORY
06B3 47                                 INC     DI
06B4 47                                 INC     DI                      ; SKIP SEGMENT POINTER
06B5 E2 FB                              LOOP    D3A


                                ;---------------------------------------
                                ; TEST.12                              :
                                ;       VERIFY CMOS CHECKSUM/BATTERY OK :
                                ; DESCRIPTION                          :
                                ;       DETERMINE IF CONFIG RECORD     :
                                ;       CAN BE USED FOR INITIALIZATION. :
                                ;---------------------------------------
                                        ASSUME  DS:DATA
06B7 E8 0000 E                          CALL    DDS                     ; SET THE DATA SEGMENT

06BA B0 16                              MOV     AL,16H                  ;       <><><><><><><><><><><><><>
06BC E6 80                              OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  16 <><>

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

06BE B0 8D                              MOV     AL,CMOS_REG_D+NMI       ; CHECK BATTERY CONDITION
06C0 E8 0000 E                          CALL    CMOS_READ               ; READ THE BATTERY STATUS
06C3 A8 80                              TEST    AL,10000000B            ; IS THE BATTERY LOW?
06C5 74 0B                              JZ      CMOS1A                  ; ERROR IF YES

06C7 B0 8E                              MOV     AL,CMOS_DIAG+NMI        ; GET THE OLD STATUS
06C9 E8 0000 E                          CALL    CMOS_READ               ; FROM DIAGNOSTIC STATUS BYTE
06CC A8 80                              TEST    AL,BAD_BAT              ; HAS CUSTOMER SETUP BEEN EXECUTED?
06CE 74 15                              JZ      CMOS1                   ; GO CHECK CHECKSUM IF YES

06D0 EB 64                              JMP     SHORT CMOS4             ; CONTINUE WITHOUT CONFIGURATION

                                ;-----  SET DEFECTIVE BATTERY FLAG

06D2 B0 17                      CMOS1A: MOV     AL,17H                  ;       <><><><><><><><><><><><><>
06D4 E6 80                              OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  17 <><>

06D6 B8 8E8E                            MOV     AX,X*(CMOS_DIAG+NMI)    ; CMOS DIAGNOSTIC STATUS BYTE
06D9 E8 0000 E                          CALL    CMOS_READ               ; GET THE CURRENT STATUS
06DC 0C 80                              OR      AL,BAD_BAT              ; SET THE DEAD BATTERY FLAG
06DE 86 C4                              XCHG    AL,AH                   ; SAVE
06E0 E8 0000 E                          CALL    CMOS_WRITE              ; OUTPUT THE STATUS
06E3 EB 51                              JMP     SHORT  CMOS4            ; GO TO MINIMUM CONFIGURATION

                                ;-----  VERIFY CHECKSUM

06E5 B8 8E8E                    CMOS1:  MOV     AX,X*(CMOS_DIAG+NMI)    ; CLEAR OLD STATUS
06E8 E8 0000 E                          CALL    CMOS_READ               ; GET THE CURRENT STATUS
06EB 81 3E 0072 R 1234                  CMP     @RESET_FLAG,1234H       ; IS THIS A SOFT RESET
06F1 75 04                              JNZ     CMOS1_A                 ; GO IF NOT

06F3 24 10                              AND     AL,W_MEM_SIZE           ; CLEAR ALL BUT THE CMOS/POR MEMORY SIZE
06F5 EB 02                              JMP     SHORT CMOS1_B
06F7                            CMOS1_A:
06F7 2A C0                              SUB     AL,AL                   ; CLEAR STATUS IF POWER ON RESET
06F9                            CMOS1_B:
06F9 86 C4                              XCHG    AL,AH                   ; SAVE THE CURRENT STATUS
06FB E8 0000 E                          CALL    CMOS_WRITE

06FE 2B DB                              SUB     BX,BX
0700 2B C9                              SUB     CX,CX
0702 B1 90                              MOV     CL,CMOS_DISKETTE+NMI    ; SET START OF CMOS CHECKSUMED AREA
0704 B5 AE                              MOV     CH,CMOS_CKSUM_HI+NMI    ; SET END OF CMOS CHECKSUMED AREA +1
                                                                        ;  (FIRST BYTE OF CHECKSUM)
0706 8A C1                      CMOS2:  MOV     AL,CL
0708 E8 0000 E                          CALL    CMOS_READ               ; ADDRESS THE BEGINNING
070B 2A E4                              SUB     AH,AH                   ; INSURE AH=0
070D 03 D8                              ADD     BX,AX                   ; ADD TO CURRENT VALUE
070F FE C1                              INC     CL                      ; POINT TO NEXT BYTE ADDRESS IN CMOS
0711 3A E9                              CMP     CH,CL                   ; FINISHED?  (AT CHECKSUM BYTE HIGH)
0713 75 F1                              JNZ     CMOS2                   ; GO IF NOT
0715 0B DB                              OR      BX,BX                   ; BX MUST NOT BE 0
0717 74 10                              JZ      CMOS3                   ; CMOS BAD IF CHECKSUM=0
0719 B0 AE                              MOV     AL,CMOS_CKSUM_HI+NMI    ; GET THE CHECK SUM HIGH BYTE
071B E8 0000 E                          CALL    CMOS_READ               ; FIRST BYTE OF CHECKSUM
071E 8A E0                              MOV     AH,AL                   ; SAVE IT
0720 B0 AF                              MOV     AL,CMOS_CKSUM_LO+NMI    ; SECOND BYTE OF CHECKSUM
0722 E8 0000 E                          CALL    CMOS_READ
0725 3B C3                              CMP     AX,BX                   ; IS THE CHECKSUM OK
0727 74 0D                              JZ      CMOS4                   ; GO IF YES

                                ;-----  SET CMOS CHECKSUM ERROR

0729 B8 8E8E                    CMOS3:  MOV     AX,X*(CMOS_DIAG+NMI)    ; ADDRESS DIAGNOSTIC STATUS
072C E8 0000 E                          CALL    CMOS_READ               ; GET THE CURRENT STATUS
072F 0C 40                              OR      AL,BAD_CKSUM            ; SET BAD CHECKSUM FLAG
0731 86 C4                              XCHG    AL,AH                   ; SAVE IT
0733 E8 0000 E                          CALL    CMOS_WRITE              ; SET FLAG

                                ;-----  INSURE CMOS DIVIDERS SET

0736                            CMOS4:
0736 B8 8A8A                            MOV     AX,X*(CMOS_REG_A+NMI)   ; ADDRESS CMOS REGISTER A
0739 E8 0000 E                          CALL    CMOS_READ               ; GET CURRENT DIVISORS
073C 24 0F                              AND     AL,00FH                 ; LOOK AT PERIODIC RATE BITS
073E 75 07                              JNZ     CMOS9                   ; EXIT IF SET TO SOMETHING USEFUL

0740 B0 26                              MOV     AL,26H                  ; ELSE SET THE STANDARD DEFAULT USED BY
0742 86 C4                              XCHG    AL,AH                   ;  BIOS FOR THE 976.56 US RATE
0744 E8 0000 E                          CALL    CMOS_WRITE              ;  FOR THE PERIODIC CLOCK
0747                            CMOS9:
0747 B0 18                              MOV     AL,18H                  ;       <><><><><><><><><><><><><>
0749 E6 80                              OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  18 <><>

                                ;-----  ENABLE PROTECTED MODE

```

```
0748 E4 61                    IN      AL,PORT_B               ; DISABLE MEMORY AND I/O PARITY CHECKS
074D 0C 0C                    OR      AL,RAM_PAR_OFF
074F E6 61                    OUT     PORT_B,AL

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

0751 B0 19                    MOV     AL,19H                  ;       <><><><><><><><><><><><><>
0753 E6 80                    OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  19 <><>

0755 B8 018F                  MOV     AX,1*H+(CMOS_SHUT_DOWN+NMI)    ; SET THE RETURN ADDRESS FOR
0758 E8 0000 E                CALL    CMOS_WRITE              ; THE FIRST SHUTDOWN RETURN ADDRESS

075B BC 0000                  MOV     SP,POST_SS              ; SET STACK FOR SYSINIT1
075E 8E D4                    MOV     SS,SP
0760 BC 8000                  MOV     SP,POST_SP
0763 E8 0000 E                CALL    SYSINIT1                ; CALL THE DESCRIPTOR TABLE BUILDER
                                                              ; AND REAL-TO-PROTECTED MODE SWITCHER

0766 B0 1A                    MOV     AL,1AH                  ;       <><><><><><><><><><><><><>
0768 E6 80                    OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  1A <><>

                      ;-----  SET TEMPORARY STACK

076A 6A 08                    PUSH    BYTE PTR GDT_PTR        ; SET (DS:) SELECTOR TO GDT SEGMENT
076C 1F                       POP     DS
076D C7 06 005A 0000          MOV     DS:SS_TEMP.BASE_LO_WORD,0
0773 C6 06 005C 00            MOV     BYTE PTR DS:(SS_TEMP.BASE_HI_BYTE),0
0778 BE 0058                  MOV     SI,SS_TEMP
077B 8E D6                    MOV     SS,SI
077D BC FFFD                  MOV     SP,MAX_SEG_LEN-2

                      ;--------------------------------------------------------------------------
                      ; 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

                              SMSW    AX                      ; GET THE MACHINE STATUS WORD
0780 0F 01 E0         +       DB      00FH,001H,0E0H
0783 A9 0001                  TEST    AX,VIRTUAL_ENABLE       ; ARE WE IN PROTECTED MODE
0786 75 0C                    JNZ     VIR_OK

0788 B8 088F          SHUT_8: MOV     AX,8*H+(CMOS_SHUT_DOWN+NMI)    ; SET THE RETURN ADDRESS
078B E8 0000 E                CALL    CMOS_WRITE              ; AND SET SHUTDOWN 8
078E E9 0000 E                JMP     PROC_SHUTDOWN           ; CAUSE A SHUTDOWN

                      ;-----  VIRTUAL MODE ERROR HALT

0791 F4               SHUT8:  HLT
0792 EB FD                    JMP     SHUT8                   ; ERROR HALT

                      ;-----  64K SEGMENT LIMIT

0794 C7 06 0048 FFFF  VIR_OK: MOV     DS:ES_TEMP.SEG_LIMIT,MAX_SEG_LEN

                      ;-----  CPL0, DATA ACCESS RIGHTS

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

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

079F C6 06 004C 01            MOV     BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE),01H
07A4 C7 06 004A 0000          MOV     DS:ES_TEMP.BASE_LO_WORD,0H

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

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

                      ;-----  START STORAGE SIZE/CLEAR

07B1                  NOT_DONE:
07B1 6A 48                    PUSH    BYTE PTR ES_TEMP        ; POINT ES TO DATA
07B3 07                       POP     ES                      ; POINT TO SEGMENT TO TEST
07B4 E8 07D0 R                CALL    HOW_BIG                 ; DO THE FIRST 64K
07B7 74 03                    JZ      NOT_FIN                 ; CHECK IF TOP OF MEMORY
07B9 E9 086E R                JMP     DONE

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

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

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

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

07C3 80 3E 004C 0A            CMP     BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE),0AH
07C8 75 E7                    JNZ     NOT_DONE                ; GO IF NOT
07CA E8 084B R                CALL    HOW_BIG_END             ; GO SET MEMORY SIZE
07CD E9 086E R                JMP     DONE

                      ;-----  FILL/CHECK LOOP

07D0                  HOW_BIG:
07D0 2B FF                    SUB     DI,DI
07D2 B8 AA55                  MOV     AX,0AA55H               ; TEST PATTERN
07D5 8B C8                    MOV     CX,AX                   ; SAVE PATTERN
07D7 26: 89 05                MOV     ES:[DI],AX              ; WRITE PATTERN TO MEMORY
07DA B0 0F                    MOV     AL,0FH                  ; PUT SOMETHING IN AL
07DC 26: 8B 05                MOV     AX,ES:[DI]              ; GET PATTERN
07DF 26: 89 05                MOV     ES:[DI],AX              ; INSURE NO PARITY I/O CHECK
07E2 33 C1                    XOR     AX,CX                   ; COMPARE PATTERNS
07E4 75 65                    JNZ     HOW_BIG_END             ; GO END IF NO COMPARE
```

SECTION 5

```

07E6 1E                          PUSH    DS
07E7 6A 18                       PUSH    BYTE PTR RSDA_PTR       ; POINT TO SYSTEM DATA AREA
07E9 1F                          POP     DS                      ; GET (DS:)

                         ;-----  IS THIS A SOFT RESET

07EA 81 3E 0072 R 1234           CMP     @RESET_FLAG,1234H       ; SOFT RESET
07F0 1F                          POP     DS                      ; RESTORE DS
07F1 75 36                       JNZ     HOW_BIG_2               ; GO IF NOT SOFT RESET

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

07F3 26: C7 05 0101              MOV     WORD PTR ES:[DI],0101H  ; TURN OFF BOTH PARITY BITS

07F8 E4 61                       IN      AL,PORT_B
07FA 0C 0C                       OR      AL,RAM_PAR_OFF          ; TOGGLE PARITY CHECK ENABLES
07FC E6 61                       OUT     PORT_B,AL
07FE 24 F3                       AND     AL,RAM_PAR_ON
0800 E6 61                       OUT     PORT_B,AL
0802 6A FF                       PUSH    BYTE PTR 0FFH           ; PLACE 0FFFFH IN STACK (BUS BITS ON)
0804 58                          POP     AX                      ; DELAY - CAUSING BUS BITS ON
0805 26: 8B 05                   MOV     AX,ES:[DI]              ; CHECK PARITY

0808 E4 61                       IN      AL,PORT_B               ; CHECK FOR PLANAR OR I/O PARITY CHECK
080A 24 C0                       AND     AL,PARITY_ERR
080C 26: 89 05                   MOV     ES:[DI],AX              ; CLEAR POSSIBLE PARITY ERROR
080F 75 3A                       JNZ     HOW_BIG_END             ; GO IF PLANAR OR I/O PARITY CHECK

                         ;-----  CHECK ALL BITS WRITE OK

0811 26: C7 05 FFFF              MOV     WORD PTR ES:[DI],0FFFFH ; TURN ON ALL BITS
0816 26: 8B 05                   MOV     AX,ES:[DI]              ; CHECK FOR FFFFH
0819 50                          PUSH    AX                      ; SAVE RESULTS
081A E4 61                       IN      AL,PORT_B               ; CHECK FOR PLANAR OR I/O PARITY CHECK
081C 24 C0                       AND     AL,PARITY_ERR
081E 26: 89 05                   MOV     ES:[DI],AX              ; CLEAR POSSIBLE PARITY ERROR
0821 58                          POP     AX                      ; GET RESULTS
0822 75 27                       JNZ     HOW_BIG_END             ; GO IF PARITY CHECK
0824 3D FFFF                     CMP     AX,0FFFFH
0827 75 22                       JNZ     HOW_BIG_END

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

0829                     HOW_BIG_2:
0829 2B C0                       SUB     AX,AX                   ; WRITE ZEROS
082B B9 8000                     MOV     CX,2000H*4              ; SET COUNT FOR 32K WORDS
082E F3/ AB                      REP     STOSW                   ; FILL 32K WORDS

0830 1E                          PUSH    DS
0831 06                          PUSH    ES
0832 06                          PUSH    ES                      ; GET ES TO DS
0833 1F                          POP     DS
0834 B9 8000                     MOV     CX,2000H*4              ; SET COUNT FOR 32K WORDS
0837 2B F6                       SUB     SI,SI
0839 F3/ AD                      REP     LODSW
083B 2B FF                       SUB     DI,DI                   ; SET TO BEGINNING OF BLOCK
083D E4 61                       IN      AL,PORT_B               ; CHECK FOR PLANAR OR I/O PARITY CHECK
083F 24 C0                       AND     AL,PARITY_ERR
0841 26: C7 05 0000              MOV     WORD PTR ES:[DI],0      ; CLEAR POSSIBLE PARITY ERROR
0846 07                          POP     ES                      ; RESTORE SEGMENTS
0847 1F                          POP     DS
0848 75 01                       JNZ     HOW_BIG_END             ; GO IF PLANAR OR I/O PARITY CHECK

084A C3                          RET

084B                     HOW_BIG_END:
084B 9C                          PUSHF                           ; SAVE THE CURRENT FLAGS
084C B0 1C                       MOV     AL,1CH                  ;       <><><><><><><><><><><><>
084E E6 80                       OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  1C <><>

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

0850 B8 B3B3                     MOV     AX,X*(CMOS_INFO128+NMI) ; SET/RESET 640K STATUS FLAG
0853 E8 0000 E                   CALL    CMOS_READ               ; GET THE DIAGNOSTIC STATUS
0856 0C 80                       OR      AL,M640K
0858 81 FB 0200                  CMP     BX,512                  ; CHECK MEMORY SIZE
085C 77 02                       JA      K640                    ; SET FLAG FOR 512 -> 640 INSTALLED
085E 24 7F                       AND     AL,NOT M640K
0860                     K640:
0860 86 C4                       XCHG    AL,AH                   ; SAVE THE STATUS
0862 E8 0000 E                   CALL    CMOS_WRITE              ; RESTORE THE STATUS

0865 6A 18                       PUSH    BYTE PTR RSDA_PTR       ; RESTORE THE DATA SEGMENT
0867 1F                          POP     DS
0868 89 1E 0013 R                MOV     @MEMORY_SIZE,BX         ; SAVE MEMORY SIZE
086C 9D                          POPF                            ; RESTORE THE FLAG REGISTER
086D C3                          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.                        :
                         ;--------------------------------------------------------------------------

086E                     DONE:
086E 6A 08                       PUSH    BYTE PTR GDT_PTR        ; POINT DS TO THE DESCRIPTOR TABLE
0870 1F                          POP     DS

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

0871 C6 06 004C 10               MOV     BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE),10H
0876 C7 06 004A 0000             MOV     DS:ES_TEMP.BASE_LO_WORD,0H

087C B0 1D                       MOV     AL,1DH                  ;       <><><><><><><><><><><><>
087E E6 80                       OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  1D <><>

0880 2B DB                       SUB     BX,BX                   ; START WITH COUNT 0
```

```

                                ;-----  START STORAGE SIZE/CLEAR

0882                            NOT_DONE1:
0882 6A 48                              PUSH    BYTE PTR ES_TEMP        ; POINT ES TO DATA
0884 07                                 POP     ES                      ; POINT TO SEGMENT TO TEST
0885 E8 08A1 R                          CALL    HOW_BIG1                ; DO THE FIRST 64K
0888 74 03                              JZ      DONEA                   ; CHECK IF TOP

088A E9 0928 R                          JMP     DONE1                   ; GO IF TOP

088D 83 C3 40                   DONEA:  ADD     BX,16*4                 ; BUMP MEMORY COUNT BY 64K

                                ;-----  DO NEXT 64K (XX0000) BLOCK

0890 FE 06 004C                         INC     BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE)

                                ;-----  CHECK FOR TOP OF MEMORY (FE0000)

0894 80 3E 004C FE                      CMP     BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE),0FEH ; LAST OF MEMORY?
0899 75 E7                              JNZ     NOT_DONE1                               ; GO IF NOT
089B E8 0915 R                          CALL    HOW_BIG_END1                            ; GO SET MEMORY SIZE
089E E9 0928 R                          JMP     DONE1

                                ;-----  FILL/CHECK LOOP

08A1                            HOW_BIG1:
08A1 2B FF                              SUB     DI,DI
08A3 B8 AA55                            MOV     AX,0AA55H               ; TEST PATTERN
08A6 8B C8                              MOV     CX,AX                   ; SAVE PATTERN
08A8 26: 89 05                          MOV     ES:[DI],AX              ; SEND PATTERN TO MEMORY
08AB B0 0F                              MOV     AL,0FH                  ; PUT SOMETHING IN AL
08AD 26: 8B 05                          MOV     AX,ES:[DI]              ; GET PATTERN
08B0 26: 89 05                          MOV     ES:[DI],AX              ; INSURE NO PARITY I/O CHECK
08B3 33 C1                              XOR     AX,CX                   ; COMPARE PATTERNS
08B5 75 5E                              JNZ     HOW_BIG_END1            ; GO END IF NO COMPARE

                                ;-----  IS THIS A SOFT RESET

08B7 1E                                 PUSH    DS
08B8 6A 18                              PUSH    BYTE PTR RSDA_PTR       ; POINT TO SYSTEM DATA AREA
08BA 1F                                 POP     DS
08BB 81 3E 0072 R 1234                  CMP     @RESET_FLAG,1234H       ; SOFT RESET
08C1 1F                                 POP     DS                      ; RESTORE DS
08C2 75 2F                              JNZ     HOW_BIG_2A              ; GO IF NOT SOFT RESET

                                ;-----  CHECK PARITY WITH PARITY BITS OFF

08C4 26: C7 05 0101                     MOV     WORD PTR ES:[DI],0101H  ; TURN OFF BOTH PARITY BITS
08C9 6A FF                              PUSH    BYTE PTR 0FFH           ; PLACE 0FFFFH IN STACK (BUS BITS ON)
08CB 58                                 POP     AX                      ; DELAY - CAUSING BUS BITS ON
08CC 26: 8B 05                          MOV     AX,ES:[DI]              ; CHECK PARITY

08CF E4 61                              IN      AL,PORT_B               ; CHECK FOR PLANAR OR I/O PARITY CHECK
08D1 24 C0                              AND     AL,PARITY_ERR
08D3 26: 89 05                          MOV     ES:[DI],AX              ; CLEAR POSSIBLE PARITY ERROR
08D6 75 3D                              JNZ     HOW_BIG_END1            ; GO IF PLANAR OR I/O PARITY CHECK

                                ;-----  CHECK ALL BITS

08D8 26: C7 05 FFFF                     MOV     WORD PTR ES:[DI],0FFFFH ; TURN ON ALL BITS
08DD 6A 00                              PUSH    BYTE PTR 0              ; PLACE 00000H IN STACK (BUS BITS OFF)
08DF 58                                 POP     AX                      ; DELAY - CAUSING BUS BITS OFF
08E0 26: 8B 05                          MOV     AX,ES:[DI]              ; CHECK FOR FFFFH
08E3 50                                 PUSH    AX                      ; SAVE RESULTS
08E4 E4 61                              IN      AL,PORT_B               ; CHECK FOR PLANAR OR I/O PARITY CHECK
08E6 24 C0                              AND     AL,PARITY_ERR
08E8 26: 89 05                          MOV     ES:[DI],AX              ; CLEAR POSSIBLE PARITY ERROR
08EB 58                                 POP     AX                      ; GET RESULTS
08EC 75 27                              JNZ     HOW_BIG_END1            ; GO IF PLANAR OR I/O PARITY CHECK
08EE 3D FFFF                            CMP     AX,0FFFFH
08F1 75 22                              JNZ     HOW_BIG_END1

                                ;-----  CLEAR 64K BLOCK OF MEMORY

08F3                            HOW_BIG_2A:
08F3 2B C0                              SUB     AX,AX                   ; WRITE ZEROS
08F5 B9 8000                            MOV     CX,2000H*4              ; SET COUNT FOR 32K WORDS
08F8 F3/ AB                             REP     STOSW                   ; FILL 32K WORDS

                                ;-----  CHECK 64K BLOCK FOR PARITY CHECK (VALID TEST DURING SOFT RESET ONLY)

08FA 1E                                 PUSH    DS
08FB 06                                 PUSH    ES
08FC 06                                 PUSH    ES                      ; GET ES TO DS
08FD 1F                                 POP     DS
08FE B9 8000                            MOV     CX,2000H*4              ; SET COUNT FOR 32K WORDS
0901 2B F6                              SUB     SI,SI
0903 F3/ AD                             REP     LODSW
0905 2B FF                              SUB     DI,DI                   ; SET TO BEGINNING OF BLOCK
0907 E4 61                              IN      AL,PORT_B               ; CHECK FOR PLANAR OR I/O PARITY CHECK
0909 24 C0                              AND     AL,PARITY_ERR
090B 26: C7 05 0000                     MOV     WORD PTR ES:[DI],0      ; CLEAR POSSIBLE PARITY ERROR
0910 07                                 POP     ES                      ; RESTORE SEGMENT
0911 1F                                 POP     DS
0912 75 01                              JNZ     HOW_BIG_END1            ; GO IF PLANAR OR I/O PARITY CHECK

0914 C3                                 RET

0915                            HOW_BIG_END1:
0915 B0 1E                              MOV     AL,1EH                  ;       <><><><><><><><><><><><><>
0917 E6 80                              OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  1E <><>

                                ;-----  SET EXPANSION MEMORY SIZE DETERMINED IN CMOS

0919 B0 B0                              MOV     AL,CMOS_U_M_S_LO+NMI    ; ADDRESS LOW BYTE
091B 8A E3                              MOV     AH,BL                   ; GET LOW MEMORY SIZE
091D E8 0000 E                          CALL    CMOS_WRITE              ; SET LOW BYTE
0920 B0 B1                              MOV     AL,CMOS_U_M_S_HI+NMI    ; ADDRESS HI BYTE
0922 8A E7                              MOV     AH,BH                   ; GET THE HIGH MEMORY SIZE
0924 E8 0000 E                          CALL    CMOS_WRITE              ; PLACE IN CMOS
0927 C3                                 RET

                                ;-----  TEST ADDRESS LINES 19 - 23
```

```
0928 B0 1F            DONE1:  MOV     AL,1FH                  ;       <><><><><><><><><><><><><>
092A E6 80                    OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  1F <><>
092C C6 06 004C 00            MOV     BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE),00H
0931 2B FF                    SUB     DI,DI                   ; SET LOCATION POINTER TO ZERO
0933 BA FFFF                  MOV     DX,0FFFFH               ; WRITE FFFF AT ADDRESS 0
0936 E8 0965 R                CALL    SD0
0939 2B D2                    SUB     DX,DX                   ; WRITE 0
093B C6 06 004C 08            MOV     BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE),08H
0940 E8 0965 R                CALL    SD0
0943 C6 06 004C 10            MOV     BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE),10H
0948 E8 0965 R                CALL    SD0
094B C6 06 004C 20            MOV     BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE),20H
0950 E8 0965 R                CALL    SD0
0953 C6 06 004C 40            MOV     BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE),40H
0958 E8 0965 R                CALL    SD0
095B C6 06 004C 80            MOV     BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE),80H
0960 E8 0965 R                CALL    SD0
0963 EB 18                    JMP     SHORT SD2               ; TEST PASSED CONTINUE
0965                  SD0:
0965 6A 48                    PUSH    BYTE PTR ES_TEMP        ; POINT ES TO DATA
0967 07                       POP     ES                      ; POINT TO SEGMENT TO TEST
0968 26: 89 15                MOV     ES:[DI],DX              ; WRITE THE PATTERN
096B C6 06 004C 00            MOV     BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE),00H
0970 6A 48                    PUSH    BYTE PTR ES_TEMP        ; POINT ES TO DATA
0972 07                       POP     ES                      ; POINT TO SEGMENT TO TEST
0973 26: 83 3D FF             CMP     WORD PTR ES:[DI],0FFFFH ; DID LOCATION 0 CHANGE?
0977 74 03                    JZ      SD1                     ; CONTINUE IF NOT
0979 E9 0788 R                JMP     SHUT_8                  ; GO HALT IF YES
097C                  SD1:
097C C3                       RET
                      ;-----  CAUSE A SHUTDOWN
097D B0 20            SD2:    MOV     AL,20H                  ;       <><><><><><><><><><><><><>
097F E6 80                    OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  20 <><>
0981 E4 61                    IN      AL,PORT_B
0983 0C 0C                    OR      AL,RAM_PAR_OFF          ; TOGGLE PARITY CHECK ENABLES
0985 E6 61                    OUT     PORT_B,AL
0987 24 F3                    AND     AL,RAM_PAR_ON
0989 E6 61                    OUT     PORT_B,AL
098B E9 0000 E                JMP     PROC_SHUTDOWN           ; CAUSE A SHUTDOWN (RETURN VIA JUMP)
                      ;----------------------------------------
                      ; RETURN 1 FROM SHUTDOWN                 :
                      ;----------------------------------------
098E B0 21            SHUT1:  MOV     AL,21H                  ;       <><><><><><><><><><><><><>
0990 E6 80                    OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  21 <><>
0992 BC ---- R                MOV     SP,ABS0                 ; SET REAL MODE STACK
0995 8E D4                    MOV     SS,SP
0997 BC 0400 R                MOV     SP,OFFSET @TOS
                      ;-----  SET DIVIDE 0 VECTOR OFFSET
099A 2B FF                    SUB     DI,DI                   ; POINT TO FIRST INTERRUPT LOCATION
099C 8E C7                    MOV     ES,DI                   ; SET ES TO ABS0 SEGMENT
099E B8 0000 E                MOV     AX,OFFSET D11           ; GET ADDRESS OF INTERRUPT OFFSET
09A1 AB                       STOSW                           ; PLACE OFFSET IF NULL HANDLER IN VECTOR
09A2 E8 0000 E                CALL    DDS                     ; SET UP THE REAL DATA AREA
                      ;-----  GET THE CONFIGURATION FROM CMOS
09A5 B8 8E8E                  MOV     AX,X*(CMOS_DIAG+NMI)    ; CHECK CMOS GOOD
09A8 E8 0000 E                CALL    CMOS_READ               ; GET THE STATUS
09AB A8 C0                    TEST    AL,BAD_BAT+BAD_CKSUM    ; VALID CMOS ?
09AD 74 03                    JZ      M_OK                    ; GO IF YES
09AF E9 0A38 R                JMP     BAD_MOS                 ; GO IF NOT
09B2                  M_OK:
09B2 24 DF                    AND     AL,0DFH                 ; CLEAR THE MINIMUM CONFIG BIT
09B4 86 C4                    XCHG    AL,AH                   ; SAVE THE STATUS BYTE
09B6 E8 0000 E                CALL    CMOS_WRITE              ; BACK INTO CMOS
                      ;-----  CHECK FOR CMOS RUN IN MODE
09B9 81 3E 0072 R 1234        CMP     @RESET_FLAG,1234H       ; CHECK FOR SOFT RESET
09BF 74 10                    JE      M_OK_64                 ; BYPASS IF SOFT RESET
09C1 B0 96                    MOV     AL,CMOS_B_M_S_HI+NMI    ; GET THE BASE MEMORY SIZE HIGH BYTE
09C3 E8 0000 E                CALL    CMOS_READ
09C6 24 C0                    AND     AL,0C0H                 ; MASK FOR MANUFACTURING TEST BITS
09C8 3C C0                    CMP     AL,0C0H                 ; CHECK FOR MANUFACTURING TEST MODE SET
09CA 75 05                    JNE     M_OK_64                 ; SKIP IF NOT MANUFACTURING LINE TEST
09CC C6 06 0072 R 64          MOV     BYTE PTR @RESET_FLAG,64H; ELSE SET THE MFG TEST FLAG
                      ;-----  INSURE CONFIGURATION HAS CORRECT VIDEO TYPE
09D1                  M_OK_64:
09D1 B0 94                    MOV     AL,CMOS_EQUIP+NMI       ; GET THE EQUIPMENT BYTE
09D3 E8 0000 E                CALL    CMOS_READ
09D6 8A E0                    MOV     AH,AL                   ; SAVE VIDEO TYPE
09D8 A8 30                    TEST    AL,030H                 ; ANY VIDEO?
09DA 75 31                    JNZ     MOS_OK_1                ; CONTINUE
09DC E8 09EA R                CALL    CHK_VIDEO               ; INSURE VIDEO ROM PRESENT
09DF 74 4C                    JZ      MOS_OK                  ; CONTINUE
09E1 F6 06 0012 R 20          TEST    @MFG_TST,MFG_LOOP       ; EXCEPT IF MFG JUMPER IS INSTALLED
09E6 74 6F                    JZ      NORMAL_CONFIG           ; GO IF INSTALLED
09E8 EB 4E                    JMP     SHORT BAD_MOS           ; GO DEFAULT
                      ;-----  ROUTINE CHECK FOR VIDEO FEATURE ROM PRESENT
09EA                  CHK_VIDEO:
09EA B9 C000                  MOV     CX,0C000H               ; START OF FEATURE I/O ROM
09ED                  CHK_VIDEO1:
09ED 50                       PUSH    AX                      ; SAVE THE CONFIGURATION
```

```
09EE 1E                         PUSH    DS                      ; SAVE THE DATA SEGMENT
09EF 57                         PUSH    DI                      ; SAVE COMPARE REGISTER
09F0 8E D9                      MOV     DS,CX                   ; GET ROM SEGMENT
09F2 BF AA55                    MOV     DI,0AA55H               ; GET THE PRESENCE SIGNATURE
09F5 2B DB                      SUB     BX,BX                   ; CLEAR INDEX POINTER
09F7 8B 07                      MOV     AX,[BX]                 ; GET THE FIRST 2 LOCATIONS
09F9 3B C7                      CMP     AX,DI                   ; IS THE VIDEO FEATURE ROM PRESENT?
09FB 5F                         POP     DI                      ; RESTORE WORK REGISTER
09FC 1F                         POP     DS                      ; RESTORE DATA SEGMENT
09FD 58                         POP     AX                      ; GET THE CONFIGURATION
09FE 74 0C                      JZ      CHK_VIDEO2              ; GO IF VIDEO ROM INSTALLED

0A00 81 C1 0080                 ADD     CX,080H                 ; POINT TO NEXT 2K BLOCK
0A04 81 F9 C800                 CMP     CX,0C800H               ; TOP OF VIDEO ROM AREA YET?
0A08 7C E3                      JL      CHK_VIDEO1              ; TRY AGAIN
0A0A 23 C9                      AND     CX,CX                   ; SET NON ZERO FLAG
0A0C                    CHK_VIDEO2:
0A0C C3                         RET                             ; RETURN TO CALLER

                        ;-----  CMOS VIDEO BITS NON ZERO (CHECK FOR PRIMARY DISPLAY AND NO VIDEO ROM)

0A0D                    MOS_OK_1:
0A0D E8 09EA R                  CALL    CHK_VIDEO               ; IS THE VIDEO ROM INSTALLED?
0A10 74 26                      JZ      BAD_MOS                 ; WRONG CONFIGURATION IN CONFIG BYTE

0A12 8A C4                      MOV     AL,AH                   ; RESTORE CONFIGURATION
0A14 F6 06 0012 R 40            TEST    @MFG_TST,DSP_JMP        ; CHECK FOR DISPLAY JUMPER
0A19 74 0A                      JZ      MOS_OK_2                ; GO IF COLOR CARD IS PRIMARY DISPLAY

                        ;-----  MONOCHROME CARD IS PRIMARY DISPLAY     (NO JUMPER INSTALLED)

0A1B 24 30                      AND     AL,30H                  ; INSURE MONOCHROME IS PRIMARY
0A1D 3C 30                      CMP     AL,30H                  ; CONFIGURATION OK?
0A1F 75 17                      JNZ     BAD_MOS                 ; GO IF NOT
0A21 8A C4                      MOV     AL,AH                   ; RESTORE CONFIGURATION
0A23 EB 08                      JMP     SHORT MOS_OK            ; USE THE CONFIGURATION BYTE FOR DISPLAY

                        ;-----  COLOR CARD

0A25                    MOS_OK_2:
0A25 24 30                      AND     AL,30H                  ; STRIP UNWANTED BITS
0A27 3C 30                      CMP     AL,30H                  ; MUST NOT BE MONO WITH JUMPER INSTALLED
0A29 8A C4                      MOV     AL,AH                   ; RESTORE CONFIGURATION
0A2B 74 0B                      JZ      BAD_MOS                 ; GO IF YES

                        ;-----  CONFIGURATION MUST HAVE AT LEAST ONE DISKETTE

0A2D A8 01              MOS_OK: TEST    AL,01H                  ; MUST HAVE AT LEAST ONE DISKETTE
0A2F 75 26                      JNZ     NORMAL_CONFIG           ; GO SET CONFIGURATION IF OK
0A31 F6 06 0012 R 20            TEST    @MFG_TST,MFG_LOOP       ; EXCEPT IF MFG JUMPER IS INSTALLED
0A36 74 1F                      JZ      NORMAL_CONFIG           ; GO IF INSTALLED

                        ;-----  MINIMUM CONFIGURATION WITH BAD CMOS OR NON VALID VIDEO

0A38                    BAD_MOS:
0A38 B8 008E                    MOV     AX,CMOS_DIAG+NMI        ; GET THE DIAGNOSTIC STATUS
0A3B E8 0000 E                  CALL    CMOS_READ
0A3E A8 C0                      TEST    AL,BAD_BAT+BAD_CKSUM    ; WAS BATTERY DEFECTIVE OR BAD CHECKSUM
0A40 75 03                      JNZ     BAD_MOS1                ; GO IF YES

0A42 E8 0000 E                  CALL    CONFIG_BAD              ; SET THE MINIMUM CONFIGURATION FLAG
0A45                    BAD_MOS1:
0A45 E8 09EA R                  CALL    CHK_VIDEO               ; CHECK FOR VIDEO ROM
0A48 B0 01                      MOV     AL,01H                  ; DISKETTE ONLY
0A4A 74 0B                      JZ      NORMAL_CONFIG           ; GO IF VIDEO ROM PRESENT

0A4C F6 06 0012 R 40            TEST    @MFG_TST,DSP_JMP        ; CHECK FOR DISPLAY JUMPER
0A51 B0 11                      MOV     AL,11H                  ; DEFAULT TO 40X25 COLOR
0A53 74 02                      JZ      NORMAL_CONFIG           ; GO IF JUMPER IS INSTALLED

0A55 B0 31                      MOV     AL,31H                  ; DISKETTE / B/W DISPLAY 80X25

                        ;---------------------------------------
                        ;       CONFIGURATION AND MFG MODE     :
                        ;---------------------------------------

0A57                    NORMAL_CONFIG:
0A57 F6 06 0012 R 20            TEST    @MFG_TST,MFG_LOOP       ; IS THE MANUFACTURING JUMPER INSTALLED
0A5C 75 02                      JNZ     NORM1                   ; GO IF NOT
0A5E 24 3E                      AND     AL,03EH                 ; STRIP DISKETTE FOR MFG TEST

0A60 2A E4              NORM1:  SUB     AH,AH
0A62 A3 0010 R                  MOV     @EQUIP_FLAG,AX          ; SAVE SWITCH INFORMATION
0A65 81 3E 0072 R 1234          CMP     @RESET_FLAG,1234H       ; BYPASS IF SOFT RESET
0A6B 74 2C                      JZ      E6

                        ;-----  GET THE FIRST SELF TEST RESULTS FROM KEYBOARD

0A6D B0 60                      MOV     AL,WRITE_8042_LOC       ; ENABLE KEYBOARD
0A6F E8 0396 R                  CALL    C8042                   ; ISSUE WRITE BYTE COMMAND
0A72 B0 4D                      MOV     AL,4DH                  ; ENABLE OUTPUT BUFFER FULL INTERRUPT,
                                                                ; SET SYSTEM FLAG, PC 1 COMPATIBILITY,
0A74 E6 60                      OUT     PORT_A,AL               ; INHIBIT OVERRIDE, ENABLE KEYBOARD

0A76 2B C9                      SUB     CX,CX                   ; WAIT FOR COMMAND ACCEPTED
0A78 E8 039B R                  CALL    C42_1

0A7B B9 7FFF                    MOV     CX,07FFFH               ; SET LOOP COUNT FOR APPROXIMATELY 100MS
                                                                ; TO RESPOND
0A7E E4 64              TST6:   IN      AL,STATUS_PORT          ; WAIT FOR OUTPUT BUFFER FULL
0A80 A8 01                      TEST    AL,OUT_BUF_FULL
0A82 E1 FA                      LOOPZ   TST6                    ; TRY AGAIN IF NOT

0A84 9C                         PUSHF                           ; SAVE FLAGS
0A85 B0 AD                      MOV     AL,DIS_KBD              ; DISABLE KEYBOARD
0A87 E8 0396 R                  CALL    C8042                   ; ISSUE THE COMMAND
0A8A 9D                         POPF                            ; RESTORE FLAGS
0A8B 74 0C                      JZ      E6                      ; CONTINUE WITHOUT RESULTS

0A8D E4 60                      IN      AL,PORT_A               ; GET INPUT FROM KEYBOARD
0A8F A2 0072 R                  MOV     BYTE PTR @RESET_FLAG,AL ; TEMPORARY SAVE FOR AA RECEIVED

                        ;-----  CHECK FOR MFG REQUEST

```

```
0A92 3C 65                     CMP     AL,065H                 ; LOAD MANUFACTURING TEST REQUEST?
0A94 75 03                     JNE     E6                      ; CONTINUE IF NOT
0A96 E9 0C25 R                 JMP     MFG_BOOT                ; ELSE GO TO MANUFACTURING BOOTSTRAP

                          ;------------------------------------------------------
                          ; 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                    :
                          ;------------------------------------------------------

0A99                      E6:
0A99 A1 0010 R                 MOV     AX,@EQUIP_FLAG          ; GET SENSE INFORMATION
0A9C 50                        PUSH    AX                      ; SAVE IT
0A9D B0 30                     MOV     AL,30H                  ; FORCE MONOCHROME TYPE
0A9F A3 0010 R                 MOV     @EQUIP_FLAG,AX          ; INTO EQUIPMENT FLAG
0AA2 2B C0                     SUB     AX,AX                   ; MODE SET COMMAND FOR DEFAULT MODE
0AA4 CD 10                     INT     INT_VIDEO               ; SEND INITIALIZATION TO B/W CARD
0AA6 B0 20                     MOV     AL,20H                  ; FORCE COLOR AT 80 BY 25
0AA8 A3 0010 R                 MOV     @EQUIP_FLAG,AX          ; INTO EQUIPMENT FLAG TO CLEAR BUFFERS
0AAB B8 0003                   MOV     AX,0003H                ; AND INITIALIZATION COLOR CARD 80X25
0AAE CD 10                     INT     INT_VIDEO               ; MODE SET 80 X 25
0AB0 B8 0001                   MOV     AX,0001H                ; SET COLOR 40 X 25 MODE
0AB3 CD 10                     INT     INT_VIDEO               ; SET DEFAULT COLOR MODE
0AB5 58                        POP     AX                      ; RECOVER REAL SWITCH INFORMATION
0AB6 A3 0010 R                 MOV     @EQUIP_FLAG,AX          ; RESTORE IT
0AB9 24 30                     AND     AL,30H                  ; ISOLATE VIDEO SWITCHES
0ABB 75 11                     JNZ     E7                      ; VIDEO SWITCHES SET TO 0?
0ABD 1E                        PUSH    DS                      ; SAVE THE DATA SEGMENT
0ABE 50                        PUSH    AX
0ABF 2B C0                     SUB     AX,AX                   ; SET DATA SEGMENT TO 0
0AC1 8E D8                     MOV     DS,AX
0AC3 BF 0040 R                 MOV     DI,OFFSET @VIDEO_INT    ; SET INTERRUPT 10H TO DUMMY
0AC6 C7 05 0000 E              MOV     WORD PTR [DI],OFFSET DUMMY_RETURN  ; RETURN IF NO VIDEO CARD
0ACA 58                        POP     AX                      ; RESTORE REGISTERS
0ACB 1F                        POP     DS
0ACC EB 7F                     JMP     SHORT E18_1             ; BYPASS VIDEO TEST
0ACE                      E7:
0ACE 3C 30                     CMP     AL,30H                  ; B/W CARD ATTACHED?
0AD0 74 08                     JE      E8                      ; YES - SET MODE FOR B/W CARD
0AD2 FE C4                     INC     AH                      ; SET COLOR MODE FOR COLOR CARD
0AD4 3C 20                     CMP     AL,20H                  ; 80X25 MODE SELECTED?
0AD6 75 02                     JNE     E8                      ; NO - SET MODE FOR 40X25
0AD8 B4 03                     MOV     AH,3                    ; SET MODE FOR 80X25
0ADA                      E8:
0ADA 86 E0                     XCHG    AH,AL
0ADC 50                        PUSH    AX                      ; SAVE VIDEO MODE ON STACK
0ADD 2A E4                     SUB     AH,AH                   ; INITIALIZE TO ALPHANUMERIC MD
0ADF CD 10                     INT     INT_VIDEO               ; CALL VIDEO_IO
0AE1 58                        POP     AX                      ; RESTORE VIDEO SENSE SWITCHES IN AH
0AE2 50                        PUSH    AX                      ; SAVE VALUE
0AE3 BB B000                   MOV     BX,0B000H               ; STARTING VIDEO MEMORY ADDRESS B/W CARD
0AE6 BA 03B8                   MOV     DX,3B8H                 ; MODE REGISTER FOR B/W
0AE9 B9 0800                   MOV     CX,2048                 ; MEMORY WORD COUNT FOR B/W CARD
0AEC 80 FC 30                  CMP     AH,30H                  ; B/W VIDEO CARD ATTACHED?
0AEF 74 07                     JE      E9                      ; YES - GO TEST VIDEO STORAGE
0AF1 B7 B8                     MOV     BH,0B8H                 ; STARTING MEMORY ADDRESS FOR COLOR CARD
0AF3 BA 03D8                   MOV     DX,3D8H                 ; MODE REGISTER FOR COLOR CARD
0AF6 B5 20                     MOV     CH,20H                  ; MEMORY WORD COUNT FOR COLOR CARD
0AF8                      E9:
0AF8 A0 0065 R                 MOV     AL,@CRT_MODE_SET        ; GET CURRENT MODE SET VALUE
0AFB 24 37                     AND     AL,037H                 ; SET VIDEO BIT OFF
0AFD EE                        OUT     DX,AL                   ; DISABLE VIDEO FOR COLOR CARD
0AFE 8E C3                     MOV     ES,BX                   ; POINT ES TO VIDEO MEMORY
0B00 8E DB                     MOV     DS,BX                   ; POINT DS TO VIDEO MEMORY
0B02 D1 C9                     ROR     CX,1                    ; DIVIDE BY 2 FOR WORD COUNT
0B04 E8 0000 E                 CALL    STGTST_CNT              ; GO TEST VIDEO READ/WRITE STORAGE
0B07 75 70                     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. :
                          ;--------------------------------------------

0B09 B0 22                     MOV     AL,22H                  ;       <><><><><><><><><><><><>
0B0B E6 80                     OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  22 <><>

0B0D 58                        POP     AX                      ; GET VIDEO SENSE SWITCHES (AH)
0B0E 50                        PUSH    AX                      ; SAVE IT
0B0F B4 00                     MOV     AH,0                    ; ENABLE VIDEO AND SET MODE
0B11 CD 10                     INT     INT_VIDEO               ; VIDEO
0B13 B8 7020                   MOV     AX,7020H                ; WRITE BLANKS IN REVERSE VIDEO
0B16 2B FF                     SUB     DI,DI                   ; SETUP STARTING LOCATION
0B18 B9 0028                   MOV     CX,40                   ; NUMBER OF BLANKS TO DISPLAY
0B1B 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.                  :
                          ;-------------------------------------

0B1D 58                        POP     AX                      ; GET VIDEO SENSE SWITCH INFORMATION
0B1E 50                        PUSH    AX                      ; SAVE IT
0B1F 80 FC 30                  CMP     AH,30H                  ; B/W CARD ATTACHED?
0B22 BA 03BA                   MOV     DX,03BAH                ; SETUP ADDRESS OF B/W STATUS PORT
0B25 74 03                     JE      E11                     ; YES - GO TEST LINES
0B27 BA 03DA                   MOV     DX,03DAH                ; COLOR CARD IS ATTACHED
0B2A                      E11:
0B2A B4 08                     MOV     AH,8
0B2C                      E12:
0B2C 2B C9                     SUB     CX,CX
```

```
0B2E                    E13:
0B2E EC                         IN      AL,DX                   ; READ CRT STATUS PORT
0B2F 22 C4                      AND     AL,AH                   ; CHECK VIDEO/HORIZONTAL LINE
0B31 75 04                      JNZ     E14                     ; ITS ON - CHECK IF IT GOES OFF
0B33 E2 F9                      LOOP    E13                     ; LOOP UNTIL ON OR TIMEOUT
0B35 EB 42                      JMP     SHORT E17               ; GO PRINT ERROR MESSAGE
0B37                    E14:
0B37 2B C9                      SUB     CX,CX
0B39                    E15:
0B39 EC                         IN      AL,DX                   ; READ CRT STATUS PORT
0B3A 22 C4                      AND     AL,AH                   ; CHECK VIDEO/HORIZONTAL LINE
0B3C 74 04                      JZ      E16                     ; ITS ON - CHECK NEXT LINE
0B3E E2 F9                      LOOP    E15                     ; LOOP IF ON UNTIL IT GOES OFF
0B40 EB 37                      JMP     SHORT E17               ; GO ERROR BEEP

                        ;-----  CHECK HORIZONTAL LINE

0B42 B1 03              E16:    MOV     CL,3                    ; GET NEXT BIT TO CHECK
0B44 D2 EC                      SHR     AH,CL
0B46 75 E4                      JNZ     E12                     ; CONTINUE
0B48                    E18:
0B48 58                         POP     AX                      ; GET VIDEO SENSE SWITCHES (AH)
0B49 B4 00                      MOV     AH,0                    ; SET MODE AND DISPLAY CURSOR
0B4B CD 10                      INT     INT_VIDEO               ; CALL VIDEO I/O PROCEDURE

                        ;-----  CHECK FOR THE ADVANCED VIDEO CARD

0B4D BA C000            E18_1:  MOV     DX,0C000H               ; SET THE LOW SEGMENT VALUE
0B50                    E18A:
0B50 B0 23                      MOV     AL,23H                  ;       <><><><><><><><><><><><><>
0B52 E6 80                      OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  23 <><>
0B54 8E DA                      MOV     DS,DX
0B56 57                         PUSH    DI                      ; SAVE WORK REGISTER
0B57 BF AA55                    MOV     DI,0AA55H               ; PRESENCE SIGNATURE
0B5A 2B DB                      SUB     BX,BX                   ; CLEAR POINTER
0B5C 8B 07                      MOV     AX,[BX]                 ; GET FIRST 2 LOCATIONS
0B5E 3B C7                      CMP     AX,DI                   ; PRESENT?
0B60 5F                         POP     DI                      ; RECOVER REGISTER
0B61 75 05                      JNZ     E18B                    ; NO? GO LOOK FOR OTHER MODULES

0B63 E8 0000 E                  CALL    ROM_CHECK               ; GO SCAN MODULE
0B66 EB 04                      JMP     SHORT E18C
0B68                    E18B:
0B68 81 C2 0080                 ADD     DX,0080H                ; POINT TO NEXT 2K BLOCK
0B6C                    E18C:
0B6C 81 FA C800                 CMP     DX,0C800H               ; TOP OF VIDEO ROM AREA YET?
0B70 7C DE                      JL      E18A                    ; GO SCAN FOR ANOTHER MODULE

0B72 B0 24                      MOV     AL,24H                  ;       <><><><><><><><><><><><><>
0B74 E6 80                      OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  24 <><>

0B76 E9 0000 E                  JMP     POST2                   ; GO TO NEXT TEST


                        ;-----  CRT ERROR SET MFG CHECKPOINT AND ERROR BEEP

0B79 E8 0000 E          E17:    CALL    DDS                     ; POINT TO DATA

                        ;-----  CHECKPOINT 0C = MONOCHROME FAILED

0B7C C6 06 0015 R 0C            MOV     @MFG_ERR_FLAG,0CH       ; <><> CRT ERROR CHECKPOINT  0C <><>
0B81 80 3E 0072 R 64            CMP     BYTE PTR @RESET_FLAG,064H ; IS THIS A MFG REQUEST?
0B86 74 0D                      JZ      E19                     ; BY PASS ERROR BEEP IF YES
0B88 F6 06 0012 R 20            TEST    @MFG_TST,MFG_LOOP       ; IS THE MFG LOOP JUMPER INSTALLED?
0B8D 74 06                      JZ      E19                     ; BY PASS ERROR BEEP IF YES
0B8F BA 0102                    MOV     DX,102H
0B92 E8 0000 E                  CALL    ERR_BEEP                ; GO BEEP SPEAKER
0B95                    E19:
0B95 1E                         PUSH    DS
0B96 A1 0010 R                  MOV     AX,@EQUIP_FLAG          ; GET THE CURRENT VIDEO
0B99 24 30                      AND     AL,30H                  ; STRIP OTHER BITS
0B9B 3C 30                      CMP     AL,30H                  ; IS IT MONOCHROME ?
0B9D 74 30                      JZ      TRY_COLOR               ; GO IF YES

                        ;-----  COLOR FAILED TRY MONOCHROME - CHECKPOINT 0D = COLOR FAILED

0B9F C6 06 0015 R 0D            MOV     @MFG_ERR_FLAG,0DH       ; <><> CRT ERROR CHECKPOINT  0D <><>

0BA4 BA 03B8                    MOV     DX,3B8H                 ; DISABLE B/W
0BA7 B0 01                      MOV     AL,1
0BA9 EE                         OUT     DX,AL                   ; OUTPUT THE DISABLE
0BAA BB B000                    MOV     BX,0B000H               ; CHECK FOR MONOCHROME VIDEO MEMORY
0BAD 8E DB                      MOV     DS,BX
0BAF B8 AA55                    MOV     AX,0AA55H               ; WRITE AN AA55
0BB2 2B DB                      SUB     BX,BX                   ; TO THE FIRST LOCATION
0BB4 89 07                      MOV     [BX],AX
0BB6 EB 00                      JMP     $+2                     ; ALLOW BUS TO SETTLE
0BB8 8B 07                      MOV     AX,[BX]                 ; READ THE FIRST LOCATION
0BBA 3D AA55                    CMP     AX,0AA55H               ; IS THE MONOCHROME VIDEO CARD THERE?
0BBD 1F                         POP     DS                      ; RESTORE THE DATA SEGMENT
0BBE 75 55                      JNZ     E17_3                   ; GO IF NOT
0BC0 81 0E 0010 R 0030          OR      @EQUIP_FLAG,30H         ; TURN ON MONOCHROME BITS IN EQUIP FLAG
0BC6 A1 0010 R                  MOV     AX,@EQUIP_FLAG          ; ENABLE VIDEO
0BC9 2A E4                      SUB     AH,AH
0BCB CD 10                      INT     INT_VIDEO
0BCD EB 34                      JMP     SHORT E17_1             ; CONTINUE

                        ;-----  MONOCHROME FAILED TRY COLOR

0BCF                    TRY_COLOR:
0BCF B0 01                      MOV     AL,01H                  ; SET MODE COLOR 40X25
0BD1 2A E4                      SUB     AH,AH
0BD3 CD 10                      INT     INT_VIDEO
0BD5 BA 03D8                    MOV     DX,3D8H                 ; DISABLE COLOR
0BD8 B0 00                      MOV     AL,0
0BDA EE                         OUT     DX,AL                   ; OUTPUT THE DISABLE
0BDB BB B800                    MOV     BX,0B800H               ; CHECK FOR COLOR VIDEO MEMORY
0BDE 8E DB                      MOV     DS,BX
0BE0 B8 AA55                    MOV     AX,0AA55H               ; WRITE AN AA55
0BE3 2B DB                      SUB     BX,BX                   ; TO THE FIRST LOCATION
0BE5 89 07                      MOV     [BX],AX
0BE7 EB 00                      JMP     $+2                     ; ALLOW BUS TO SETTLE
0BE9 8B 07                      MOV     AX,[BX]                 ; READ THE FIRST LOCATION
0BEB 3D AA55                    CMP     AX,0AA55H               ; IS THE COLOR VIDEO CARD THERE?
```

SECTION 5

```
0BEE 1F                              POP     DS                      ; RESTORE THE DATA SEGMENT
0BEF 75 24                           JNZ     E17_3                   ; GO IF NOT
0BF1 81 26 0010 R FFCF               AND     @EQUIP_FLAG,0FFCFH      ; TURN OFF VIDEO BITS
0BF7 81 0E 0010 R 0010               OR      @EQUIP_FLAG,10H         ; SET COLOR 40X24
0BFD B0 01                           MOV     AL,01H
0BFF 2A E4                           SUB     AH,AH
0C01 CD 10                           INT     INT_VIDEO
0C03                         E17_1:
0C03 58                              POP     AX                      ; SET NEW VIDEO TYPE ON STACK
0C04 A1 0010 R                       MOV     AX,@EQUIP_FLAG
0C07 24 30                           AND     AL,30H
0C09 3C 30                           CMP     AL,30H                  ; IS IT THE B/W?
0C0B 2A C0                           SUB     AL,AL
0C0D 74 02                           JZ      E17_2                   ; GO IF YES
0C0F FE C0                           INC     AL                      ; INITIALIZE FOR 40X25
0C11                         E17_2:
0C11 50                              PUSH    AX
0C12                         E17_4:
0C12 E9 0B48 R                       JMP     E18

                             ;-----  BOTH VIDEO CARDS FAILED SET DUMMY RETURN IF RETRACE FAILURE

0C15                         E17_3:
0C15 1E                              PUSH    DS
0C16 2B C0                           SUB     AX,AX                   ; SET DS SEGMENT TO 0
0C18 8E D8                           MOV     DS,AX
0C1A BF 0040 R                       MOV     DI,OFFSET @VIDEO_INT    ; SET INTERRUPT 10H TO DUMMY
0C1D C7 05 0000 E                    MOV     WORD PTR [DI],OFFSET DUMMY_RETURN  ; RETURN IF NO VIDEO CARD
0C21 1F                              POP     DS
0C22 E9 0B4D 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

 0C25                              MFG_BOOT:
 0C25  B4 DD                               MOV     AH,DISABLE_BIT20        ; DEGATE COMMAND FOR ADDRESS LINE 20
 0C27  E8 0000 E                           CALL    GATE_A20                ; ISSUE TO KEYBOARD ADAPTER AND CLI

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

 0C2A  68 ---- R                           PUSH    ABS0                    ; SET ES SEGMENT REGISTER TO ABS0
 0C2D  07                                  POP     ES
 0C2E  B9 0018                             MOV     CX,24                   ; GET VECTOR COUNT
 0C31  8C C8                               MOV     AX,CS                   ; GET THE CURRENT CODE SEGMENT VALUE
 0C33  8E D8                               MOV     DS,AX                   ; SETUP DS SEGMENT REGISTER TO
 0C35  BE 0000 E                           MOV     SI,OFFSET VECTOR_TABLE  ; POINT TO THE ROUTINE ADDRESS TABLE
 0C38  BF 0020 R                           MOV     DI,OFFSET @INT_PTR      ; SET DESTINATION TO FIRST USED VECTOR
 0C3B                              MFG_B1:
 0C3B  A5                                  MOVSW                           ; MOVE ONE ROUTINE OFFSET ADDRESS
 0C3C  AB                                  STOSW                           ; INSERT CODE SEGMENT VALUE
 0C3D  E2 FC                               LOOP    MFG_B1                  ; MOVE THE NUMBER OF ENTRIES REQUIRED

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

 0C3F  B9 0008                             MOV     CX,08                   ; GET VECTOR COUNT
 0C42  BE 0000 E                           MOV     SI,OFFSET SLAVE_VECTOR_TABLE
 0C45  BF 01C0 R                           MOV     DI,OFFSET @SLAVE_INT_PTR
 0C48                              MFG_B2:
 0C48  A5                                  MOVSW                           ; MOVE ONE ROUTINE OFFSET ADDRESS
 0C49  AB                                  STOSW                           ; INSERT CODE SEGMENT VALUE
 0C4A  E2 FC                               LOOP    MFG_B2

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

                                           ASSUME  DS:ABS0,ES:ABS0
 0C4C  06                                  PUSH    ES                                      ; ES= ABS0
 0C4D  1F                                  POP     DS                                      ; SET DS TO ABS0
 0C4E  C7 06 0008 R 0000 E                 MOV     WORD PTR @NMI_PTR,OFFSET NMI_INT        ; NMI INTERRUPT
 0C54  C7 06 0014 R 0000 E                 MOV     WORD PTR @INT5_PTR,OFFSET PRINT_SCREEN  ; PRINT SCREEN
 0C5A  C7 06 0062 R F600                   MOV     WORD PTR @BASIC_PTR+2,0F600H            ; CASSETTE BASIC SEGMENT

                                   ;-----  ENABLE KEYBOARD PORT

 0C60  B0 60                               MOV     AL,60H                  ; WRITE 8042 MEMORY LOCATION 0
 0C62  E8 0396 R                           CALL    C8042                   ; ISSUE THE COMMAND
 0C65  B0 09                               MOV     AL,00001001B            ; SET INHIBIT OVERRIDE/ENABLE OBF
 0C67  E6 60                               OUT     PORT_A,AL               ;   INTERRUPT AND NOT PC COMPATIBLE

 0C69  E8 0C8B R                           CALL    MFG_B4                  ; GET COUNT LOW
 0C6C  8A F8                               MOV     BH,AL                   ; SAVE IT
 0C6E  E8 0C8B R                           CALL    MFG_B4                  ; GET COUNT HI
 0C71  8A E8                               MOV     CH,AL
 0C73  8A CF                               MOV     CL,BH                   ; CX NOW HAS COUNT
 0C75  FC                                  CLD                             ; SET DIRECTION FLAG TO INCREMENT
 0C76  BF 0500 R                           MOV     DI,OFFSET @MFG_TEST_RTN ; SET TARGET OFFSET (DS=0000)
 0C79                              MFG_B3:
 0C79  E4 64                               IN      AL,STATUS_PORT          ; GET 8042 STATUS PORT
 0C7B  A8 01                               TEST    AL,OUT_BUF_FULL         ; KEYBOARD REQUEST PENDING?
 0C7D  74 FA                               JZ      MFG_B3                  ; LOOP TILL DATA PRESENT
 0C7F  E4 60                               IN      AL,PORT_A               ; GET DATA
 0C81  AA                                  STOSB                           ; STORE IT
 0C82  E6 80                               OUT     MFG_PORT,AL             ; DISPLAY CHARACTER AT MFG PORT
 0C84  E2 F3                               LOOP    MFG_B3                  ; LOOP TILL ALL BYTES READ

 0C86  EA 0500 ---- R                      JMP     @MFG_TEST_RTN           ; FAR JUMP TO CODE THAT WAS JUST LOADED

 0C8B                              MFG_B4:
 0C8B  E4 64                               IN      AL,STATUS_PORT          ; CHECK FOR OUTPUT BUFFER FULL
 0C8D  A8 01                               TEST    AL,OUT_BUF_FULL         ;   HANG HERE IF NO DATA AVAILABLE
 0C8F  E1 FA                               LOOPZ   MFG_B4

 0C91  E4 60                               IN      AL,PORT_A               ; GET THE COUNT
 0C93  C3                                  RET

 0C94                              POST1   ENDP
 0C94                              CODE    ENDS
                                           END

2407tes free

WarnSevere
ErroErrors
0    0
```

SECTION 5

```
                                PAGE 118,121
                                TITLE TEST2 ---- 11/15/85  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   CONFIG_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 <><>
```

```

0021 B0 FF                        MOV   AL,0FFH                 ; DISABLE DEVICE INTERRUPTS
0023 E6 21                        OUT   INTA01,AL               ; WRITE TO INTERRUPT MASK REGISTER
0025 E6 A1                        OUT   INTB01,AL               ; WRITE TO 2ND INTERRUPT MASK REGISTER
0027 EB 00                        JMP   $+2                     ; I/O DELAY
0029 E4 21                        IN    AL,INTA01               ; READ INTERRUPT MASK REGISTER
002B 8A E0                        MOV   AH,AL                   ; SAVE RESULTS
002D E4 A1                        IN    AL,INTB01               ; READ 2ND INTERRUPT MASK REGISTER

002F 05 0001                      ADD   AX,1                    ; ALL IMR BITS ON?
0032 75 15                        JNZ   D6                      ; NO - GO TO ERR ROUTINE

                          ;-----  CHECK FOR HOT INTERRUPTS

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

0034 A2 006B R                    MOV   @INTR_FLAG,AL           ; CLEAR INTERRUPT FLAG

0037 B0 26                        MOV   AL,26H                  ;       <><><><><><><><><><><><><>
0039 E6 80                        OUT   MFG_PORT,AL             ;       <><> CHECKPOINT  26 <><>

003B FB                           STI                           ; ENABLE EXTERNAL INTERRUPTS
003C B9 19E4                      MOV   CX,6628                 ; WAIT 100 MILLISECONDS FOR ANY
003F E8 0000 E                    CALL  WAITF                   ;  INTERRUPTS THAT OCCUR
0042 80 3E 006B R 00              CMP   @INTR_FLAG,00H          ; DID ANY INTERRUPTS OCCUR?
0047 74 0D                        JZ    D7                      ; NO - GO TO NEXT TEST

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

0056 B0 27                D7:     MOV   AL,27H                  ;       <><><><><><><><><><><><><>
0058 E6 80                        OUT   MFG_PORT,AL             ;       <><> CHECKPOINT  27 <><>

005A B8 AA55                      MOV   AX,0AA55H
005D E7 82                        OUT   MFG_PORT+2,AX           ; WRITE A WORD
005F E4 82                        IN    AL,MFG_PORT+2           ;  GET THE FIRST BYTE
0061 86 C4                        XCHG  AL,AH                   ;  SAVE IT
0063 E4 83                        IN    AL,MFG_PORT+3           ;  GET THE SECOND BYTE
0065 3D 55AA                      CMP   AX,55AAH                ; IS IT OK?
0068 74 05                        JZ    D7_A                    ; GO IF YES

006A BE 0000 E                    MOV   SI,OFFSET E106          ; DISPLAY 106 ERROR
006D EB E2                        JMP   D6A

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

006F                      D7_A:
006F B0 0D                        MOV   AL,CMOS_REG_D           ; TURN ON NMI
0071 E6 70                        OUT   CMOS_PORT,AL            ; ADDRESS DEFAULT READ ONLY REGISTER
0073 B9 0007                      MOV   CX,7                    ; DELAY COUNT FOR 100 MICROSECONDS
0076 E8 0000 E                    CALL  WAITF                   ; WAIT FOR HOT NMI TO PROCESS
0079 B0 8D                        MOV   AL,CMOS_REG_D+NMI       ; TURN NMI ENABLE BACK OFF
007B E6 70                        OUT   CMOS_PORT,AL
007D 80 3E 006B R 00              CMP   @INTR_FLAG,00H          ; DID ANY INTERRUPTS OCCUR?
0082 74 09                        JZ    D7_C                    ; CONTINUE IF NOT

0084 B0 28                        MOV   AL,28H                  ;       <><><><><><><><><><><><><>
0086 E6 80                        OUT   MFG_PORT,AL             ;       <><> CHECKPOINT  28 <><>

0088 BE 0000 E                    MOV   SI,OFFSET E107          ; DISPLAY 107 ERROR
008B EB C4                        JMP   D6A

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

008D B0 29                D7_C:   MOV   AL,29H                  ;       <><><><><><><><><><><><><>
008F E6 80                        OUT   MFG_PORT,AL             ;       <><> CHECKPOINT  29 <><>
0091 E4 61                        IN    AL,PORT_B               ; GET CURRENT SETTING OF PORT
0093 8A E0                        MOV   AH,AL                   ; SAVE THAT SETTING
0095 24 FC                        AND   AL,0FCH                 ; INSURE SPEAKER OFF
0097 E6 61                        OUT   PORT_B,AL

0099 B0 B0                        MOV   AL,10110000B            ; SELECT TIM 2,LSB,MSB,BINARY,MODE 0
009B E6 43                        OUT   TIMER+3,AL              ; WRITE THE TIMER MODE REGISTER
009D EB 00                        JMP   $+2                     ; I/O DELAY
009F B8 AA55                      MOV   AX,0AA55H               ; WRITE AN AA55
00A2 E6 42                        OUT   TIMER+2,AL              ; WRITE TIMER 2 COUNT - LSB
00A4 EB 00                        JMP   $+2                     ; I/O DELAY
00A6 8A C4                        MOV   AL,AH
00A8 E6 42                        OUT   TIMER+2,AL              ; WRITE TIMER 2 COUNT - MSB
00AA EB 00                        JMP   $+2                     ; I/O DELAY
00AC E4 42                        IN    AL,TIMER+2              ; GET THE LSB
00AE 86 E0                        XCHG  AH,AL                   ; SAVE IT
00B0 EB 00                        JMP   $+2                     ; I/O DELAY
00B2 E4 42                        IN    AL,TIMER+2              ; GET THE MSB
00B4 3D 55AA                      CMP   AX,055AAH               ; BUS OK?
00B7 74 05                        JZ    D7_D                    ; GO IF OK

00B9 BE 0000 E                    MOV   SI,OFFSET E108          ; DISPLAY 108 ERROR
00BC EB 93                        JMP   D6A

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

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

00CE EB 00                        JMP   $+2                     ; I/O DELAY
```

SECTION 5

```
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 005A 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 BE 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 09A5 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 5B                      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
```

SECTION 5

```
                             ;-----  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 09A5 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 40BB                    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 09A5 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

                        ;-----  BACK TO REAL MODE - MEMORY TESTS DONE

0360                    KB_LOOP_3:
0360 B0 34                      MOV   AL,34H                ;       <><><><><><><><><><><><><>
0362 E6 80                      OUT   MFG_PORT,AL           ;       <><> CHECKPOINT  34 <><>
```

```
                                                                  ; BACK TO REAL MODE
0364 E9 0000 E                  JMP     PROC_SHUTDOWN             ; NEXT TEST VIA JUMP TABLE (SHUT2)


                        ;-----  PRINT FAILING ADDRESS AND XOR'ED PATTERN IF DATA COMPARE ERROR
                        ;-----  USE DMA PAGE REGISTERS AS TEMPORARY SAVE AREA FOR ERROR
                        ;       SET SHUTDOWN 3

0367 E6 82              E21A:   OUT     DMA_PAGE+1,AL             ; SAVE FAILING BIT PATTERN (LOW BYTE)
0369 8A C4                      MOV     AL,AH                     ; SAVE HIGH BYTE
036B E6 83                      OUT     DMA_PAGE+2,AL
036D 8B C6                      MOV     AX,SI                     ; GET THE FAILING OFFSET
036F E6 86                      OUT     DMA_PAGE+5,AL
0371 86 E0                      XCHG    AH,AL
0373 E6 85                      OUT     DMA_PAGE+4,AL

                        ;-----  CLEAR I/O CHANNEL CHECK OR R/W PARITY CHECK

0375 2B F6                      SUB     SI,SI                     ; WRITE TO FAILING BLOCK
0377 AB                         STOSW
0378 E4 61                      IN      AL,PORT_B                 ; GET PARITY CHECK LATCHES
037A E6 88                      OUT     DMA_PAGE+7,AL             ; SAVE FOR ERROR HANDLER
037C 0C 0C                      OR      AL,RAM_PAR_OFF            ; TOGGLE I/O-PARITY CHECK ENABLE
037E E6 61                      OUT     PORT_B,AL                 ;  TO RESET CHECKS
0380 24 F3                      AND     AL,RAM_PAR_ON
0382 E6 61                      OUT     PORT_B,AL

                        ;-----  GET THE LAST OF GOOD MEMORY

0384 58                         POP     AX                        ; CLEAR BLOCK COUNT
0385 58                         POP     AX                        ; GET THE LAST OF GOOD MEMORY
0386 5B                         POP     BX                        ; GET BASE MEMORY COUNTER
0387 C1 E3 06                   SHL     BX,6                      ; CONVERT TO MEMORY SIZE COUNTS
038A 2B C3                      SUB     AX,BX                     ; COMPARE LAST GOOD MEMORY WITH BASE
038C 73 17                      JAE     E211                      ; IF ABOVE OR EQUAL, USE REMAINDER IN
                                                                  ;  CMOS_U_M_S_(H/L)
                        ;-----  ELSE SET BASE MEMORY SIZE

038E 6A 18                      PUSH    BYTE PTR RSDA_PTR         ; SET THE DATA SEGMENT
0390 1F                         POP     DS                        ; IN PROTECTED MODE

0391 03 C3                      ADD     AX,BX                     ; CONVERT BACK TO LAST WORKING MEMORY
0393 A3 0013 R                  MOV     @MEMORY_SIZE,AX           ; TO INDICATE HOW MUCH MEMORY WORKING

                        ;-----  RESET 512K --> 640K OPTION IF SET

0396 B8 B3B3                    MOV     AX,X*(CMOS_INFO128+NMI)   ; ADDRESS OPTIONS INFORMATION BYTE
0399 E8 0000 E                  CALL    CMOS_READ                 ; READ THE MEMORY INFORMATION FLAG
039C 24 7F                      AND     AL,NOT M640K              ; SET 640K OPTION OFF
039E 86 C4                      XCHG    AL,AH                     ; MOVE TO WORK REGISTER
03A0 E8 0000 E                  CALL    CMOS_WRITE                ; UPDATE STATUS IF IT WAS ON
03A3 33 C0                      XOR     AX,AX                     ; CLEAR VALUE FOR EXTENSION MEMORY
03A5                    E211:
03A5 8B C8                      MOV     CX,AX                     ; SAVE ADJUSTED MEMORY SIZE
03A7 B0 B1                      MOV     AL,CMOS_U_M_S_HI+NMI
03A9 E8 0000 E                  CALL    CMOS_WRITE                ; SAVE THE HIGH BYTE MEMORY SIZE
03AC 8A E1                      MOV     AH,CL                     ; GET THE LOW BYTE
03AE B0 B0                      MOV     AL,CMOS_U_M_S_LO+NMI      ; DO THE LOW BYTE
03B0 E8 0000 E                  CALL    CMOS_WRITE                ; WRITE IT

                        ;-----  SET SHUTDOWN 3

03B3 B8 038F                    MOV     AX,3*H+CMOS_SHUT_DOWN+NMI         ; ADDRESS FOR SHUTDOWN RETURN
03B6 E8 0000 E                  CALL    CMOS_WRITE                        ; SET RETURN 3

                        ;-----  SHUTDOWN

03B9 E9 0000 E                  JMP     PROC_SHUTDOWN
```

```
                                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 0995 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 0995 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 ....
```

SECTION 5

```
                                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
```

```
048B E8 0000 E           LOOP1:  CALL    OBF_42                  ; CHECK FOR OUTPUT BUFFER FULL
048E 75 04                       JNZ     G10                     ; GO IF BUFFER FULL
0490 FE CF                       DEC     BH
0492 75 F7                       JNZ     LOOP1
0494 B0 AD               G10:    MOV     AL,DIS_KBD              ; DISABLE KEYBOARD
0496 E8 0000 E                   CALL    C8042
0499 E4 60                       IN      AL,PORT_A               ; FLUSH
049B B0 E0                       MOV     AL,KYBD_CLK_DATA        ; GET THE CLOCK AND DATA LINES
049D E8 0000 E                   CALL    C8042
04A0 E8 0000 E                   CALL    OBF_42                  ; WAIT FOR OUTPUT BUFFER FULL
04A3 E4 60                       IN      AL,PORT_A               ; GET THE RESULTS
04A5 A8 01                       TEST    AL,KYBD_CLK             ; KEYBOARD CLOCK MUST BE LOW
04A7 74 0A                       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           G11:    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 1 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
```

SECTION 5

```
054A BE 0000 E                          MOV     SI,OFFSET SLAVE_VECTOR_TABLE
054D BF 01C0 R                          MOV     DI,OFFSET @SLAVE_INT_PTR
0550 A5                       F7A1:     MOVSW
0551 47                                 INC     DI                      ; SKIP OVER SEGMENT
0552 47                                 INC     DI
0553 E2 FB                              LOOP    F7A1

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

                                        ASSUME  DS:ABS0
0555 2B C0                              SUB     AX,AX                   ; DS=0
0557 8E D8                              MOV     DS,AX
0559 C7 06 0008 R 0000 E                MOV     WORD PTR @NMI_PTR,OFFSET NMI_INT ; NMI INTERRUPT
055F C7 06 0014 R 0000 E                MOV     WORD PTR @INT5_PTR,OFFSET PRINT_SCREEN   ; PRINT SCREEN
0565 C7 06 0062 R F600                  MOV     WORD PTR @BASIC_PTR+2,0F600H     ; SEGMENT FOR CASSETTE BASIC
056B C7 06 007E R 0000                  MOV     WORD PTR @EXT_PTR+2,0   ; SEGMENT FOR GRAPHIC CHARS 128 - 255

                              ;-----    ZERO RESERVED VECTORS

0571 BF 0180                            MOV     DI,60H*4                ; FILL INTERRUPT 60 THRU 67 WITH ZERO
0574 B9 0010                            MOV     CX,16                   ; CLEAR 16 WORDS
0577 C7 05 0000               F7A2:     MOV     WORD PTR DS:[DI],0
057B 83 C7 02                           ADD     DI,2                    ; POINT TO NEXT LOCATION
057E E2 F7                              LOOP    F7A2

                              ;-----    SETUP TIMER 0 TO BLINK LED IF MANUFACTURING TEST MODE

                                        ASSUME  DS:DATA
0580 E8 0000 E                          CALL    DDS                     ; ESTABLISH DATA SEGMENT

0583 F6 06 0012 R 20                    TEST    @MFG_TST,MFG_LOOP       ; MFG. TEST MODE?
0588 75 0B                              JNZ     F9
058A 26: C7 06 0020 R 0000 E            MOV     WORD PTR ES:@INT_PTR,OFFSET BLINK_INT ; SETUP TIMER TO BLINK LED
0591 B0 FE                              MOV     AL,0FEH                 ; ENABLE TIMER INTERRUPT
0593 E6 21                              OUT     INTA01,AL
0595 FB                       F9:       STI                             ; ALLOW INTERRUPTS

                              ;-----    ISSUE A RESET TO THE HARD FILE IF SOFT RESET

0596 81 3E 0072 R 1234                  CMP     @RESET_FLAG,1234H       ; SOFT RESET?
059C 75 0E                              JNZ     F9A                     ; CONTINUE IF NOT
059E B9 00FF                            MOV     CX,0FFH
05A1 BA 03F6                            MOV     DX,03F6H
05A4 B0 04                              MOV     AL,04H                  ; RESET
05A6 EE                                 OUT     DX,AL
05A7 E2 FE                    F9_A:     LOOP    F9_A                    ; HOLD RESET
05A9 2A C0                              SUB     AL,AL
05AB EE                                 OUT     DX,AL                   ; REMOVE RESET

                              ;----------------------------------------------------------------
                              ; TEST.23                                                         :
                              ;         DISKETTE ATTACHMENT TEST                                :
                              ; DESCRIPTION                                                     :
                              ;         CHECK IF IPL DISKETTE DRIVE IS ATTACHED TO SYSTEM.  IF  :
                              ;         ATTACHED, VERIFY STATUS OF NEC FDC AFTER A RESET. ISSUE :
                              ;         A RECALIBRATE AND SEEK COMMAND TO FDC AND CHECK STATUS. :
                              ;         COMPLETE SYSTEM INITIALIZATION THEN PASS CONTROL TO THE :
                              ;         BOOT LOADER PROGRAM.                                    :
                              ;----------------------------------------------------------------

05AC B0 3C                    F9A:      MOV     AL,3CH                  ;       <><><><><><><><><><><><>
05AE E6 80                              OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  3C <><>

05B0 B0 02                              MOV     AL,02H                  ; SET DATA RATE TO 250 K BITS PER SECOND
05B2 BA 03F7                            MOV     DX,3F7H
05B5 EE                                 OUT     DX,AL
05B6 F6 06 0010 R 01                    TEST    BYTE PTR @EQUIP_FLAG,1H ; DISKETTE PRESENT?
05BB 74 55                              JZ      F15
05BD F6 06 0012 R 20                    TEST    @MFG_TST,MFG_LOOP       ; MFG JUMPER INSTALLED?
05C2 74 4E                              JZ      F15                     ; GO IF YES
05C4                          F10:                                      ; DISK_TEST:
05C4 E4 21                              IN      AL,INTA01
05C6 EB 00                              JMP     $+2                     ; I/O DELAY
05C8 24 BF                              AND     AL,0BFH                 ; ENABLE DISKETTE INTERRUPTS
05CA E6 21                              OUT     INTA01,AL
05CC B4 00                              MOV     AH,0                    ; RESET NEC FDC
05CE 8A D4                              MOV     DL,AH                   ; SET FOR DRIVE 0
05D0 CD 13                              INT     13H                     ; VERIFY STATUS AFTER RESET
05D2 F6 C4 FF                           TEST    AH,0FFH                 ; STATUS OK?
05D5 75 25                              JNZ     F13                     ; NO - FDC FAILED

                              ;-----    TURN DRIVE 0 MOTOR ON

05D7 BA 03F2                            MOV     DX,03F2H                ; GET ADDRESS OF FDC CARD
05DA B0 1C                              MOV     AL,1CH                  ; TURN MOTOR ON, ENABLE DMA, INTERRUPTS
05DC EE                                 OUT     DX,AL                   ; WRITE FDC CONTROL REGISTER
05DD 2B C9                              SUB     CX,CX                   ; WAITF COUNT FOR 0.988 SECONDS
05DF E8 0000 E                          CALL    WAITF                   ; WAIT 1 SECOND FOR MOTOR

05E2 33 FF                              XOR     DI,DI                   ; SELECT DRIVE 0
05E4 B5 01                              MOV     CH,1                    ; SELECT TRACK 1
05E6 C6 06 003E R 00                    MOV     @SEEK_STATUS,0          ; INSURE RECALIBRATE
05EB 80 0E 00A0 R 01                    OR      @RTC_WAIT_FLAG,01       ; NO REAL TIME CLOCK, USE WAIT LOOP
05F0 E8 0000 E                          CALL    SEEK                    ; RECALIBRATE DISKETTE
05F3 72 07                              JC      F13                     ; GO TO ERR SUBROUTINE IF ERR
05F5 B5 22                              MOV     CH,34                   ; SELECT TRACK 34
05F7 E8 0000 E                          CALL    SEEK                    ; SEEK TO TRACK 34
05FA 73 0B                              JNC     F14                     ; OK, TURN MOTOR OFF
05FC                          F13:                                      ; DSK_ERR:
05FC 80 0E 0016 R 40                    OR      @MFG_ERR_FLAG+1,DSK_FAIL;       <><><><><><><><><><><><><>
                                                                        ;       <><> DISKETTE FAILED  <><>
0601 BE 0000 E                          MOV     SI,OFFSET E601          ; GET ADDRESS OF MESSAGE
0604 E8 0000 E                          CALL    E_MSG                   ; GO PRINT ERROR MESSAGE

                              ;-----    TURN DRIVE 0 MOTOR OFF

0607                          F14:                                      ; DR0_OFF:
0607 80 26 00A0 R FE                    AND     @RTC_WAIT_FLAG,0FEH     ; ALLOW FOR RTC WAIT
060C B0 0C                              MOV     AL,0CH                  ; TURN DRIVE 0 MOTOR OFF
060E BA 03F2                            MOV     DX,03F2H                ; FDC CONTROLLER ADDRESS
0611 EE                                 OUT     DX,AL

                              ;-----    SETUP KEYBOARD PARAMETERS

```

```
0612 C6 06 006B R 00    F15:    MOV     @INTR_FLAG,00H          ; SET STRAY INTERRUPT FLAG = 00
0617 BE 001E R                  MOV     SI,OFFSET @KB_BUFFER    ; SETUP KEYBOARD PARAMETERS
061A 89 36 001A R               MOV     @BUFFER_HEAD,SI
061E 89 36 001C R               MOV     @BUFFER_TAIL,SI
0622 89 36 0080 R               MOV     @BUFFER_START,SI
0626 83 C6 20                   ADD     SI,32                   ; DEFAULT BUFFER OF 32 BYTES
0629 89 36 0082 R               MOV     @BUFFER_END,SI

                        ;-----  SET PRINTER TIMEOUT DEFAULT

062D BF 0078 R                  MOV     DI,OFFSET @PRINT_TIM_OUT; SET DEFAULT PRINTER TIMEOUT
0630 1E                         PUSH    DS
0631 07                         POP     ES
0632 B8 1414                    MOV     AX,1414H                ; DEFAULT=20
0635 AB                         STOSW
0636 AB                         STOSW

                        ;-----  SET RS232 DEFAULT

0637 B8 0101                    MOV     AX,0101H                ; RS232 DEFAULT=01
063A AB                         STOSW
063B AB                         STOSW

                        ;-----  ENABLE TIMER INTERRUPTS

063C E4 21                      IN      AL,INTA01
063E 24 FE                      AND     AL,0FEH                 ; ENABLE TIMER INTERRUPTS
0640 EB 00                      JMP     $+2                     ; I/O DELAY
0642 E6 21                      OUT     INTA01,AL

                        ;-----  CHECK CMOS BATTERY AND CHECKSUM

0644 F6 06 0012 R 20            TEST    @MFG_TST,MFG_LOOP       ; MFG JUMPER?
0649 75 03                      JNZ     B1_OK                   ; GO IF NOT
064B E9 0734 R                  JMP     F15C                    ; BYPASS IF YES
064E                    B1_OK:
064E B0 8E                      MOV     AL,CMOS_DIAG+NMI        ; ADDRESS DIAGNOSTIC STATUS BYTE
0650 E8 0000 E                  CALL    CMOS_READ               ; READ IT FROM CMOS

0653 BE 0000 E                  MOV     SI,OFFSET E161          ; LOAD BAD BATTERY MESSAGE 161
0656 A8 80                      TEST    AL,BAD_BAT              ; BATTERY BAD?
0658 75 07                      JNZ     B1_ER                   ; DISPLAY ERROR IF BAD

065A BE 0000 E                  MOV     SI,OFFSET E162          ; LOAD CHECKSUM BAD MESSAGE 162
065D A8 60                      TEST    AL,BAD_CKSUM+BAD_CONFIG ; CHECK FOR CHECKSUM OR NO DISKETTE
065F 74 09                      JZ      C_OK                    ; SKIP AND CONTINUE TESTING CMOS CLOCK
0661                    B1_ER:
0661 E8 0000 E                  CALL    E_MSG                   ; ELSE DISPLAY ERROR MESSAGE
0664 81 CD 8000                 OR      BP,08000H               ; FLAG "SET SYSTEM OPTIONS" DISPLAYED
0668 EB 45                      JMP     SHORT H_OK1A            ; SKIP CLOCK TESTING IF ERROR

                        ;-----  TEST CLOCK UPDATING

066A B3 04              C_OK:   MOV     BL,04H                  ; OUTER LOOP COUNT
066C 2B C9              D_OK:   SUB     CX,CX                   ; INNER LOOP COUNT
066E B0 8A              E_OK:   MOV     AL,CMOS_REG_A+NMI       ; GET THE CLOCK UPDATE BYTE
0670 E8 0000 E                  CALL    CMOS_READ
0673 A8 80                      TEST    AL,80H                  ; CHECK FOR UPDATE IN PROGRESS
0675 75 1B                      JNZ     G_OK                    ; GO IF YES
0677 E2 F5                      LOOP    E_OK                    ; TRY AGAIN
0679 FE CB                      DEC     BL                      ; DEC OUTER LOOP
067B 75 EF                      JNZ     D_OK                    ; TRY AGAIN
067D BE 0000 E          F_OK:   MOV     SI,OFFSET E163          ; PRINT MESSAGE
0680 E8 0000 E                  CALL    E_MSG

                        ;-----  SET CMOS DIAGNOSTIC STATUS TO 04 (CLOCK ERROR)

0683 B8 0E8E                    MOV     AX,X*CMOS_DIAG+NMI      ; SET CLOCK ERROR
0686 E8 0000 E                  CALL    CMOS_READ               ; GET THE CURRENT STATUS
0689 0C 04                      OR      AL,CMOS_CLK_FAIL        ; SET NEW STATUS
068B 86 C4                      XCHG    AL,AH                   ; GET STATUS ADDRESS AND SAVE NEW STATUS
068D E8 0000 E                  CALL    CMOS_WRITE              ; MOVE NEW DIAGNOSTIC STATUS TO CMOS
0690 EB 0E                      JMP     SHORT H_OK              ; CONTINUE

                        ;-----  CHECK CLOCK UPDATE

0692 B9 0320            G_OK:   MOV     CX,800                  ; LOOP COUNT
0695 B0 8A              I_OK:   MOV     AL,CMOS_REG_A+NMI       ; CHECK FOR OPPOSITE STATE
0697 E8 0000 E                  CALL    CMOS_READ
069A A8 80                      TEST    AL,80H
069C E0 F7                      LOOPNZ  I_OK                    ; TRY AGAIN
069E E3 DD                      JCXZ    F_OK                    ; PRINT ERROR IF TIMEOUT

                        ;-----  CHECK MEMORY SIZE DETERMINED = CONFIGURATION

06A0                    H_OK:
06A0 B0 8E                      MOV     AL,CMOS_DIAG+NMI        ; GET THE STATUS BYTE
06A2 E8 0000 E                  CALL    CMOS_READ
06A5 A8 10                      TEST    AL,W_MEM_SIZE           ; WAS THE CONFIG= MEM_SIZE_DETERMINED?
06A7 74 06                      JZ      H_OK1A                  ; GO IF YES

                        ;-----  MEMORY SIZE ERROR

06A9 BE 0000 E                  MOV     SI,OFFSET E164          ; PRINT SIZE ERROR
06AC E8 0000 E                  CALL    E_MSG                   ; DISPLAY ERROR

                        ;-----  CHECK FOR CRT ADAPTER ERROR

06AF 80 3E 0015 R 0C    H_OK1A: CMP     @MFG_ERR_FLAG,0CH       ; CHECK FOR MONOCHROME CRT ERROR
06B4 BE 0000 E                  MOV     SI,OFFSET E401          ; LOAD MONOCHROME CRT ERROR
06B7 74 0A                      JZ      H_OK1B                  ; GO IF YES

06B9 80 3E 0015 R 0D            CMP     @MFG_ERR_FLAG,0DH       ; CHECK FOR COLOR CRT ADAPTER ERROR
06BE 75 06                      JNZ     J_OK                    ; CONTINUE IF NOT
06C0 BE 0000 E                  MOV     SI,OFFSET E501          ; CRT ADAPTER ERROR MESSAGE
06C3                    H_OK1B:
06C3 E8 0000 E                  CALL    E_MSG

                        ;-----  CHECK FOR MULTIPLE DATA RATE CAPABILITY

06C6                    J_OK:
06C6 BA 03F1                    MOV     DX,03F1H                ; D/S/P DIAGNOSTIC REGISTER
06C9 EC                         IN      AL,DX                   ; READ D/S/P TYPE CODE
06CA 24 F8                      AND     AL,11111000B            ; KEEP ONLY UNIQUE CODE FOR D/S/P
```

SECTION 5

```
06CC 3C 50                          CMP     AL,01010000B            ; D/S/P CARD - MULTIPLE DATA RATE ?
06CE 74 46                          JZ      J_OK3                   ; IF SO JUMP

06D0 BA 05F7                        MOV     DX,05F7H                ; FIXED DISK DIAGNOSTIC REGISTER
06D3 EC                             IN      AL,DX                   ; READ FIXED DISK TYPE CODE
06D4 24 F0                          AND     AL,11110000B            ; KEEP ONLY UNIQUE CODE FOR F/D
06D6 3C A0                          CMP     AL,10100000B            ; FIXED DISK ADAPTER ?
06D8 74 2F                          JZ      J_FAIL                  ; MUST BE COMBO ELSE ERROR

06DA B3 0F                          MOV     BL,0FH                  ; OUTER LOOP COUNT WAIT FOR BUSY OFF
06DC 2B C9                          SUB     CX,CX
06DE BA 01F7                        MOV     DX,01F7H                ; HARD FILE STATUS PORT
06E1                        J_OK1:
06E1 EC                             IN      AL,DX                   ; GET THE STATUS
06E2 A8 80                          TEST    AL,080H                 ; IS THE CONTROLLER BUSY?
06E4 74 0C                          JZ      J_OK2                   ; CONTINUE IF NOT
06E6 E2 F9                          LOOP    J_OK1                   ; TRY AGAIN
06E8 FE CB                          DEC     BL                      ; DECREMENT OUTER LOOP
06EA 75 F5                          JNZ     J_OK1                   ; TRY AGAIN IF NOT ZERO
06EC 24 0C                          AND     AL,0CH                  ; BITS 2 & 3 = 0 IF MULTI DATA CAPABLE
06EE 74 26                          JZ      J_OK3                   ; GO IF YES
06F0 EB 17                          JMP     SHORT J_FAIL            ; NO MULTIPLE DATA RATE CAPABILITY
06F2                        J_OK2:
06F2 BA 01F4                        MOV     DX,1F4H                 ; VERIFY MULTIPLE DATA RATE CAPABLE
06F5 B0 55                          MOV     AL,055H                 ;  WRITE TO THE CYLINDER BYTE
06F7 EE                             OUT     DX,AL
06F8 EB 00                          JMP     $+2                     ; I/O DELAY
06FA EC                             IN      AL,DX                   ; CHECK DATA WRITTEN = DATA READ
06FB 3C 55                          CMP     AL,055H
06FD 75 0A                          JNZ     J_FAIL                  ; GO IF NOT
06FF B0 AA                          MOV     AL,0AAH                 ; WRITE ANOTHER PATTERN
0701 EE                             OUT     DX,AL
0702 EB 00                          JMP     $+2                     ; I/O DELAY
0704 EC                             IN      AL,DX
0705 3C AA                          CMP     AL,0AAH                 ; IS DATA PATTERN THE SAME?
0707 74 0D                          JZ      J_OK3                   ; GO IF SO

0709                        J_FAIL:
0709 80 0E 0016 R 40                OR      @MFG_ERR_FLAG+1,DSK_FAIL;        <><><><><><><><><><><><><><>
                                                                    ;        <><> DISKETTE FAILED  <><>
070E BE 0000 E                      MOV     SI,OFFSET E601          ; GET ADDRESS OF MESSAGE
0711 E8 0000 E                      CALL    E_MSG                   ; GO PRINT ERROR MESSAGE
0714 EB 1E                          JMP     SHORT F15C              ; SKIP SETUP IF ERROR

0716                        J_OK3:
0716 80 0E 008B R 01                OR      @LASTRATE,DUAL          ; TURN ON DSP/COMBO FLAG

                            ;-----  INITIALIZE FLOPPY FOR DRIVE TYPE

071B B0 3D                          MOV     AL,3DH                  ;        <><><><><><><><><><><><><><>
071D E6 80                          OUT     MFG_PORT,AL             ;        <><> CHECKPOINT  3D <><>
071F E8 0000 E                      CALL    DSKETTE_SETUP           ; INITIALIZE FLOPPY

                            ;-----  CHECK FOR 2ND DISKETTE DRIVE

0722 E8 0000 E                      CALL    DDS                     ; INSURE DATA SEGMENT
0725 8A 26 0091 R                   MOV     AH,@DSK_STATE+1         ; GET STATE OF SECOND DRIVE
0729 0A E4                          OR      AH,AH                   ; IS THERE A DRIVE 2 ATTACHED?
072B 74 07                          JZ      F15C                    ; SKIP IF NOT
072D 80 0E 0010 R 40                OR      BYTE PTR @EQUIP_FLAG,40H; ELSE SET SECOND DRIVE INSTALLED
0732 B4 FF                          MOV     AH,0FFH                 ; SET TEST MASK FOR DRIVE PRESENT
0734                        F15C:
0734 B0 8E                          MOV     AL,CMOS_DIAG+NMI        ; GET THE CMOS DIAGNOSTIC STATUS
0736 E8 0000 E                      CALL    CMOS_READ
0739 A8 C0                          TEST    AL,BAD_BAT+BAD_CKSUM    ; BATTERY/CHECKSUM OK
073B 75 22                          JNZ     ROM_SCAN1               ; BYPASS DISK SETUP IF NOT

073D B0 90                          MOV     AL,CMOS_DISKETTE+NMI    ; ADDRESS DISKETTE TYPE BYTE
073F E8 0000 E                      CALL    CMOS_READ               ; GET DISKETTE TYPES
0742 24 0F                          AND     AL,00FH                 ; LOOK AT SECOND DRIVE TYPE DEFINED
0744 3A C4                          CMP     AL,AH                   ; ARE BOTH INDICATORS ZERO
0746 74 07                          JE      F15D                    ; SKIP IF NO SECOND DRIVE

0748 22 C4                          AND     AL,AH                   ; ARE BOTH INDICATORS NON-ZERO
074A 75 03                          JNZ     F15D                    ; SKIP IF DRIVE FOUND AND DEFINED

074C E8 0000 E                      CALL    CONFIG_BAD              ; SET BAD_CONFIG IN CMOS_DIAG


                            ;-----  INITIALIZE HARD FILE
074F                        F15D:
074F B0 3E                          MOV     AL,3EH                  ;        <><><><><><><><><><><><><><>
0751 E6 80                          OUT     MFG_PORT,AL             ;        <><> CHECKPOINT  3E <><>

0753 B0 92                          MOV     AL,CMOS_DISK+NMI        ; INSURE CMOS DEFINES TYPE OF FIXED DISK
0755 E8 0000 E                      CALL    CMOS_READ
0758 3C 00                          CMP     AL,0H                   ; INSURE TYPE IS DEFINED
075A 74 03                          JZ      ROM_SCAN1               ; BYPASS DISK SETUP IF NOT

075C E8 0000 E                      CALL    DISK_SETUP              ; INITIALIZE HARD FILE(S)


                            ;---------------------------------------------------------
                            ; TEST.22                                                 :
                            ; CHECK FOR OPTIONAL ROM FROM C800->E000 IN 2K BLOCKS     :
                            ;         (A VALID MODULE HAS '55AA' IN THE FIRST 2 LOCATIONS:
                            ;         LENGTH INDICATOR (LENGTH/512) IN THE 3RD LOCATION  :
                            ;         AND TEST/INIT. CODE STARTING IN THE 4TH LOCATION)  :
                            ;---------------------------------------------------------
075F                        ROM_SCAN1:
075F FB                             STI                             ; ALLOW INTERRUPTS
0760 B0 3B                          MOV     AL,3BH                  ;        <><><><><><><><><><><><><><>
0762 E6 80                          OUT     MFG_PORT,AL             ;        <><> CHECKPOINT  3B <><>
0764 E8 0000 E                      CALL    DDS                     ; SET REAL MODE DATA SEGMENT
0767 B0 0A                          MOV     AL,10                   ; LINE FEED ON DISPLAY
0769 E8 0000 E                      CALL    PRT_HEX
076C                        ROM_SCAN:

                            ;-----  SET DMA MASK AND REQUEST REGISTERS

076C 2A C0                          SUB     AL,AL
076E E6 D2                          OUT     DMA18+2,AL              ; SEND ZERO TO MASK REGISTER
0770 EB 00                          JMP     $+2
0772 E6 D4                          OUT     DMA18+4,AL              ; SEND ZERO TO REQUEST REGISTER
0774 BA C800                        MOV     DX,0C800H               ; SET BEGINNING ADDRESS
0777                        ROM_SCAN2:
```

```
0777 8E DA                      MOV     DS,DX
0779 57                         PUSH    DI                      ; SAVE WORK REGISTER
077A BF AA55                    MOV     DI,0AA55H               ; GET TEST PATTERN
077D 2B DB                      SUB     BX,BX                   ; SET BX=0000
077F 8B 07                      MOV     AX,[BX]                 ; GET 1ST WORD FROM MODULE
0781 3B C7                      CMP     AX,DI                   ; = TO ID WORD?
0783 5F                         POP     DI                      ; RECOVER WORK REGISTER
0784 75 05                      JNZ     NEXT_ROM                ; PROCEED TO NEXT ROM IF NOT
0786 E8 0000 E                  CALL    ROM_CHECK               ; GO CHECK OUT MODULE
0789 EB 04                      JMP     SHORT ARE_WE_DONE       ; CHECK FOR END OF ROM SPACE
078B                    NEXT_ROM:
078B 81 C2 0080                 ADD     DX,0080H                ; POINT TO NEXT 2K ADDRESS
078F                    ARE_WE_DONE:
078F 81 FA E000                 CMP     DX,0E000H               ; AT E0000 YET?
0793 7C E2                      JL      ROM_SCAN2               ; GO CHECK ANOTHER ADD. IF NOT

                        ;-----  TEST FOR KEYBOARD LOCKED

0795 E8 0000 E                  CALL    DDS                     ; SET DATA SEGMENT
0798 E4 64                      IN      AL,STATUS_PORT          ; IS KEYBOARD UNLOCKED?
079A 24 10                      AND     AL,KYBD_INH
079C 74 02                      JZ      KEY1                    ; NO - SET ERROR FLAGS AND PRINT MESSAGE
079E EB 0B                      JMP     SHORT KEY10             ; GO IF OFF
07A0                    KEY1:
07A0 80 0E 0016 R 80            OR      @MFG_ERR_FLAG+1,KEY_FAIL;       <><><><><><><><><><><><><><>
                                                                ;       <><> KEYBOARD IS LOCKED <><>
                                ASSUME  DS:DATA
07A5 BE 0000 E                  MOV     SI,OFFSET E302          ; PRINT LOCKED MESSAGE  (302)
07A8 E8 0000 E                  CALL    E_MSG
07AB                    KEY10:
                                ;===================
                        ;-----   SETUP @PRINTER_BASE
                                ;===================

07AB BF 09DB R                  MOV     DI,OFFSET F4            ; OFFSET OF PRINTER ADDRESS TABLE
07AE BE 0000                    MOV     SI,0
07B1                    F16:
07B1 2E: 8B 15                  MOV     DX,CS:[DI]              ; GET PRINTER BASE ADDRESS
07B4 B0 AA                      MOV     AL,0AAH                 ; WRITE DATA TO PORT A
07B6 EE                         OUT     DX,AL
07B7 EB 00                      JMP     $+2                     ; I/O DELAY
07B9 1E                         PUSH    DS                      ; BUS SETTLING
07BA EC                         IN      AL,DX                   ; READ PORT A
07BB 1F                         POP     DS
07BC 3C AA                      CMP     AL,0AAH                 ; DATA PATTERN SAME
07BE 75 06                      JNE     F17                     ; NO - CHECK NEXT PRINTER CARD
07C0 89 94 0008 R               MOV     @PRINTER_BASE[SI],DX    ; YES - STORE PRINTER BASE ADDRESS
07C4 46                         INC     SI                      ; INCREMENT TO NEXT WORD
07C5 46                         INC     SI
07C6                    F17:
07C6 47                         INC     DI                      ; POINT TO NEXT BASE ADDRESS
07C7 47                         INC     DI
07C8 81 FF 09E1 R               CMP     DI,OFFSET F4E           ; ALL POSSIBLE ADDRESSES CHECKED?
07CC 75 E3                      JNE     F16                     ; PRT_BASE
                                ;============
                        ;-----  SETUP RS232
                                ;============
07CE BB 0000                    MOV     BX,0                    ; POINTER TO RS232 TABLE
07D1 BA 03FA                    MOV     DX,3FAH                 ; CHECK IF RS232 CARD 1 ATTACHED ?
07D4 EC                         IN      AL,DX                   ; READ INTERRUPT ID REGISTER
07D5 A8 F8                      TEST    AL,0F8H
07D7 75 08                      JNZ     F18
07D9 C7 87 0000 R 03F8          MOV     @RS232_BASE[BX],3F8H    ; SETUP RS232 CARD #1 ADDRESS
07DF 43                         INC     BX
07E0 43                         INC     BX
07E1 BA 02FA            F18:    MOV     DX,2FAH                 ; CHECK IF RS232 CARD 2 ATTACHED
07E4 EC                         IN      AL,DX                   ; READ INTERRUPT ID REGISTER
07E5 A8 F8                      TEST    AL,0F8H
07E7 75 08                      JNZ     F19                     ; BASE_END
07E9 C7 87 0000 R 02F8          MOV     @RS232_BASE[BX],2F8H    ; SETUP RS232 CARD #2
07EF 43                         INC     BX
07F0 43                         INC     BX
                                ;===========================================================
                        ;-----  SET UP @EQUIP_FLAG TO INDICATE NUMBER OF PRINTERS AND RS232 CARDS
                                ;===========================================================
07F1                    F19:                                    ; BASE_END:
07F1 8B C6                      MOV     AX,SI                   ; SI HAS 2* NUMBER OF RS232
07F3 B1 03                      MOV     CL,3                    ; SHIFT COUNT
07F5 D2 C8                      ROR     AL,CL                   ; ROTATE RIGHT 3 POSITIONS
07F7 0A C3                      OR      AL,BL                   ; OR IN THE PRINTER COUNT
07F9 A2 0011 R                  MOV     BYTE PTR @EQUIP_FLAG+1,AL       ; STORE AS SECOND BYTE

                        ;-----  INSURE CMOS CLOCK HAS VALID HOURS.MINUTES.SECONDS

07FC E8 0000 E                  CALL    SET_TOD                 ; INSURE CMOS CLOCK IS VALID

                        ;-----  ENABLE HARDWARE INTERRUPT IF MATH PROCESSOR (80287)

07FF B0 40                      MOV     AL,40H                  ;       <><><><><><><><><><><><>
0801 E6 80                      OUT     MFG_PORT,AL             ;       <><> CHECKPOINT 40 <><>

0803 BF 0067 R                  MOV     DI,OFFSET @IO_ROM_INIT  ; ADDRESS WORK STORAGE LOCATION
0806 33 C0                      XOR     AX,AX                   ; CLEAR WORK REGISTER  (AH)= 0 (NO 287)
0808 89 05                      MOV     WORD PTR [DI],AX        ; CLEAR THE WORK LOCATION
080A DB E3                      FNINIT                          ; INITIALIZE THE 80287 WITH NO WAIT
080C EB 00                      JMP     $+2                     ; DELAY
080E D9 3D                      FNSTCW  WORD PTR [DI]           ; WRITE THE CURRENT 80287 CONTROL WORD
0810 60                         PUSHA                           ; TIME FOR 80287 TO RESPOND
0811 61                         POPA
0812 81 25 1F3F                 AND     WORD PTR [DI],01F3FH    ; CLEAR UNUSED 80287 BITS
0816 81 3D 033F                 CMP     WORD PTR [DI],0033FH    ; IS THE 80287 INSTALLED?
081A 75 13                      JNE     NO_287                  ; GO IF MATH PROCESSOR IS NOT INSTALLED

081C 9B DD 3D                   FSTSW   WORD PTR [DI]           ; STORE THE STATUS WORD (WITH WAIT)
081F 60                         PUSHA                           ; TIME FOR 80287 TO RESPOND
0820 61                         POPA
0821 F7 05 B8BF                 TEST    WORD PTR [DI],0B8BFH    ; ALL BITS SHOULD BE OFF (OR ERROR)
0825 75 08                      JNZ     NO_287                  ; GO IF NOT INSTALLED

0827 E4 A1                      IN      AL,INTB01               ; GET THE SLAVE INTERRUPT MASK
0829 24 DF                      AND     AL,0DFH                 ; ENABLE 80287 INTERRUPTS
082B B4 02                      MOV     AH,002H                 ; SET WORK REGISTER FOR 80287 FOUND
082D E6 A1                      OUT     INTB01,AL
082F                    NO_287:
```

SECTION 5

```
082F A0 0010 R                  MOV     AL,BYTE PTR @EQUIP_FLAG ; GET LOW EQUIPMENT FLAG
0832 24 02                      AND     AL,002H                 ; STRIP OFF OTHER BITS
0834 3A C4                      CMP     AL,AH                   ; DOES CMOS MATCH HARDWARE ?
0836 74 08                      JE      OK_287                  ; SKIP IF EQUIPMENT FLAG CORRECT

0838 80 36 0010 R 02            XOR     BYTE PTR @EQUIP_FLAG,2H ; ELSE SET 80287 BIT TO CORRECT VALUE
083D E8 0000 E                  CALL    CONFIG_BAD              ; AND SET THE CONFIGURATION ERROR FLAG
0840                    OK_287:
                        ;-----  SET KEYBOARD STATE FLAGS

0840 C7 06 0017 R 0000          MOV     WORD PTR @KB_FLAG,0     ; RESET ALL KEYBOARD STATUS FLAGS

                        ;-----  ENABLE KEYBOARD/TIMER INTERRUPTS

0846 E4 21                      IN      AL,INTA01
0848 24 FC                      AND     AL,0FCH                 ; ENABLE TIMER AND KEYBOARD INTERRUPTS
084A EB 00                      JMP     $+2                     ; I/O DELAY
084C E6 21                      OUT     INTA01,AL
084E 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

0853 C6 06 0096 R A0            MOV     @KB_FLAG_3,RD_ID+SET_NUM_LK     ; SET READ ID COMMAND FOR KBX
0858 B0 F2                      MOV     AL,KB_READ_ID           ; GET THIS SYSTEMS KEYBOARD ID REQUEST
085A E8 0000 E                  CALL    SND_DATA                ; USE KEYBOARD TRANSMISSION ROUTINE
085D B9 067A                    MOV     CX,1658                 ; SET DELAY COUNT TO 25 MILLISECONDS
0860 E8 0000 E                  CALL    WAITF                   ; WAIT FOR READ ID RESPONSE (20 MS)
0863 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

0868 80 3E 0075 R 02            CMP     @HF_NUM,2               ; CHECK FOR TWO DRIVES DEFINED BY CMOS
086D 74 13                      JE      F15G                    ; SKIP TEST IF TWO DRIVES DEFINED

086F B4 10                      MOV     AH,010H                 ; GET TEST DRIVE READY COMMAND
0871 B2 81                      MOV     DL,081H                 ; POINT TO SECOND FIXED DISK
0873 FE 06 0075 R               INC     @HF_NUM                 ; TELL BIOS IT HAS TWO DRIVES
0877 CD 13                      INT     13H                     ; CHECK READY THROUGH BIOS
0879 FE 0E 0075 R               DEC     @HF_NUM                 ; RESTORE CORRECT COUNT (RETAIN CY)
087D 72 03                      JC      F15G                    ; SKIP IF SECOND DRIVE NOT READY
                                                                ;       SECOND DRIVE NOT DEFINED
087F E8 0000 E                  CALL    CONFIG_BAD              ; SET CONFIGURATION BAD
0882                    F15G:
                        ;----------------------------------------
                        ;       TEST FOR ANY ERRORS (BP NOT ZERO)    :
                        ;----------------------------------------

0882 0B ED                      OR      BP,BP                   ; CHECK (BP)= NON-ZERO  (ERROR HAPPENED)
0884 74 55                      JE      F15A_0                  ; SKIP PAUSE IF NO ERROR

0886 80 3E 0072 R 64            CMP     BYTE PTR @RESET_FLAG,64H; MFG RUN IN MODE?
088B BA 0002                    MOV     DX,2                    ; 2 SHORT BEEP COUNT FOR ERROR(S)
088E 75 0E                      JNZ     ERR_WAIT                ; GO IF NOT

                        ;-----  MFG RUN IN MODE -> SET ERROR FLAG

0890 C6 06 0015 R AA            MOV     @MFG_ERR_FLAG,0AAH      ; INDICATE ERROR
0895 E4 64                      IN      AL,STATUS_PORT          ; CHECK KEY LOCK STATUS
0897 24 10                      AND     AL,KYBD_INH             ; IS THE KEYBOARD LOCKED
0899 75 40                      JNZ     F15A_0                  ; CONTINUE MFG MODE IF NOT LOCKED
                                                                ;  ELSE
089B BA 0005                    MOV     DX,5                    ; 5 SHORT BEEPS FOR MFG SETUP ERROR
089E                    ERR_WAIT:
089E E8 0000 E                  CALL    ERR_BEEP                ; BEEPS FOR ERROR(S)
08A1 B0 0E                      MOV     AL,CMOS_DIAG            ; ADDRESS CMOS
08A3 E8 0000 E                  CALL    CMOS_READ               ; GET THE DIAGNOSTIC STATUS BYTE
08A6 A8 20                      TEST    AL,BAD_CONFIG           ; CHECK FOR BAD HARDWARE CONFIGURATION
08A8 74 0C                      JZ      ERR_WKEY                ; SKIP IF NOT SET

08AA F7 C5 8000                 TEST    BP,08000H               ; ELSE CHECK FOR E161/E162 POSTED
08AE 75 06                      JNZ     ERR_WKEY                ; SKIP IF DISPLAYED BEFORE NOW

08B0 BE 0000 E                  MOV     SI,OFFSET E162          ; ELSE DISPLAY "OPTIONS NOT SET"
08B3 E8 0000 E                  CALL    P_MSG                   ; WITH NON HALTING ROUTINE

                        ;-----  CHECK FOR "UNLOCK SYSTEM UNIT KEYLOCK" MESSAGE REQUIRED

08B6                    ERR_WKEY:
08B6 E4 64                      IN      AL,STATUS_PORT          ; CHECK IF RESUME MESSAGE NEEDED
08B8 24 10                      AND     AL,KYBD_INH             ; IS THE KEYBOARD LOCKED
08BA 75 06                      JNZ     ERR_WAIT2               ; SKIP LOCK MESSAGE IF NOT

08BC BE 0000 E                  MOV     SI,OFFSET F3D1          ; ERROR MESSAGE FOR KEYBOARD LOCKED
08BF E8 0000 E                  CALL    P_MSG

                        ;-----  DISPLAY '(RESUME = "F1" KEY)' FOR ERRORS

08C2                    ERR_WAIT2:
08C2 BE 0000 E                  MOV     SI,OFFSET F3D           ; RESUME ERROR MESSAGE
08C5 E8 0000 E                  CALL    P_MSG

                        ;-----  INITIALIZE PRINTER  (ALTERNATE DISPLAY DEVICE)

08C8 B4 01                      MOV     AH,1                    ;
08CA 2B D2                      SUB     DX,DX                   ; FIRST PRINTER
08CC CD 17                      INT     17H                     ;
08CE                    ERR_WAIT1:
08CE B0 3F                      MOV     AL,3FH                  ;       <><><><><><><><><><><><>
08D0 E6 80                      OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  3F <><>
08D2 B4 00                      MOV     AH,00
08D4 CD 16                      INT     16H                     ; WAIT FOR 'F1' KEY
08D6 80 FC 3B                   CMP     AH,3BH
08D9 75 F3                      JNE     ERR_WAIT1
08DB                    F15A_0:
08DB F6 06 0012 R 20            TEST    @MFG_TST,MFG_LOOP       ; MFG BURN IN MODE
08E0 75 03                      JNZ     F15A                    ; GO IF NOT
08E2 E9 0000 E                  JMP     START_1                 ; GO LOOP POST
08E5 80 3E 0072 R 64    F15A:   CMP     BYTE PTR @RESET_FLAG,64H; MFG RUN IN?
08EA 74 06                      JZ      F15B                    ; BYPASS BEEP IF YES

08EC BA 0001                    MOV     DX,1                    ; 1 SHORT BEEP (NO ERRORS)
08EF E8 0000 E                  CALL    ERR_BEEP
                                ;===============
                        ;-----  SET TIME OF DAY
```

```
                                          ;===============

     08F2 E8 0000 E            F15B:     CALL    SET_TOD

                               ;-----    CLEAR DISPLAY SCREEN

     08F5 2A E4                          SUB     AH,AH                    ; CLEAR FLAGS
     08F7 A0 0049 R                      MOV     AL,@CRT_MODE
     08FA CD 10                          INT     10H                      ; CLEAR SCREEN

                               ;-----    CLEAR DESCRIPTOR TABLES

     08FC B9 01F4              F20:      MOV     CX,0500                  ; CLEAR 1K
     08FF BF D0A0                        MOV     DI,SYS_IDT_LOC           ; POINT ES TO START OF DESCRIPTORS
     0902 2B C0                          SUB     AX,AX
     0904 8E C0                          MOV     ES,AX
     0906 26: 89 05            F20_A:    MOV     ES:[DI],AX               ; CLEAR
     0909 83 C7 02                       ADD     DI,2                     ; POINT TO NEXT LOCATION
     090C E2 F8                          LOOP    F20_A                    ; CONTINUE TILL DONE

                               ;-----    SET POST SYSTEM STACK

     090E B8 ---- R                      MOV     AX,ABS0                  ; GET THE POST STACK SEGMENT
     0911 8E D0                          MOV     SS,AX
     0913 BC 0400 R                      MOV     SP,OFFSET @TOS

                               ;-----    ENSURE THAT MASTER LEVEL 2 ENABLED

     0916 E4 21                          IN      AL,INTA01                ; GET THE CURRENT MASK
     0918 24 FB                          AND     AL,0FBH
     091A EB 00                          JMP     $+2                      ; I/O DELAY
     091C E6 21                          OUT     INTA01,AL

                               ;-----    TEST FOR MFG RUN-IN TEST

     091E 80 3E 0072 R 64                CMP     BYTE PTR @RESET_FLAG,64H; IS THE THE MFG RUN-IN TEST?
     0923 75 02                          JNZ     END_287                  ; GO IF NOT
     0925 EB 5C                          JMP     SHORT SHUT4              ; BOOT LOAD IF YES

                               ;-----    UNMASK SLAVE HARDWARE INTERRUPT 9 (LEVEL 71)
     0927                      END_287:
     0927 E4 A1                          IN      AL,INTB01                ; GET THE CURRENT MASK
     0929 24 FD                          AND     AL,0FDH
     092B EB 00                          JMP     $+2                      ; I/O DELAY
     092D 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
                               ;----------------------------------------------------------------
     092F B0 41                          MOV     AL,41H                   ;       <><><><><><><><><><><><><>
     0931 E6 80                          OUT     MFG_PORT,AL              ;       <><> CHECKPOINT  41 <><>

     0933 B0 8D                          MOV     AL,CMOS_REG_D+NMI        ; INSURE NMI OFF AND CMOS AT DEFAULT
     0935 E6 70                          OUT     CMOS_PORT,AL

                               ENDIF

     0937 C6 06 0072 R 00                MOV     BYTE PTR @RESET_FLAG,0   ; CLEAR FLAG
     093C B8 E000                        MOV     AX,0E000H                ; SEGMENT OF SYSTEM CODE
     093F 8E C0                          MOV     ES,AX
     0941 2B FF                          SUB     DI,DI
     0943 26: 8B 05                      MOV     AX,ES:[DI]               ; CHECK FOR AA55
     0946 53                             PUSH    BX                       ; BUS SETTLE
     0947 5B                             POP     BX
     0948 3D AA55                        CMP     AX,0AA55H
     094B 9C                             PUSHF                            ; SAVE FLAGS
     094C 26: 89 05                      MOV     ES:[DI],AX               ; CLEAR POSSIBLE PARITY CHECK
     094F E4 61                          IN      AL,PORT_B
     0951 0C 0C                          OR      AL,RAM_PAR_OFF           ; TOGGLE I/O-PARITY CHECK ENABLES
     0953 E6 61                          OUT     PORT_B,AL
     0955 24 F3                          AND     AL,RAM_PAR_ON
     0957 E6 61                          OUT     PORT_B,AL
     0959 9D                             POPF                             ; RESTORE FLAGS
     095A 75 27                          JNZ     SHUT4                    ; CONTINUE

                               ;-----    CHECKSUM SYSTEM CODE

     095C 1E                             PUSH    DS
     095D 06                             PUSH    ES                       ; SET SEGMENT TO TEST
     095E 1F                             POP     DS
     095F 2B DB                          SUB     BX,BX                    ; STARTING OFFSET
     0961 E8 0000 E                      CALL    ROM_CHECKSUM
     0964 1F                             POP     DS                       ; RESTORE DATA SEGMENT
     0965 75 1C                          JNZ     SHUT4                    ; GO IF CHECKSUM NOT OK

                               ;-----    ENABLE NMI AND I/O-MEMORY PARITY CHECKS

     0967 B0 0D                          MOV     AL,CMOS_REG_D            ; ENABLE NMI AND SET DEFAULT ADDRESS
     0969 E6 70                          OUT     CMOS_PORT,AL

     096B E4 61                          IN      AL,PORT_B                ; ENABLE PARITY
     096D 24 F3                          AND     AL,RAM_PAR_ON            ; ENABLE MEMORY PARITY CHECK / I/O CHECK
     096F E6 61                          OUT     PORT_B,AL

     0971 C7 06 0067 R 0003              MOV     @IO_ROM_INIT,0003H       ; SET THE OFFSET
     0977 8C 06 0069 R                   MOV     @IO_ROM_SEG,ES           ; SET THE SEGMENT

     097B B0 42                          MOV     AL,42H                   ;       <><><><><><><><><><><><><>
     097D E6 80                          OUT     MFG_PORT,AL              ;       <><> CHECKPOINT  42 <><>

                               ;-----    EXIT TO SYSTEM CODE

     097F FF 1E 0067 R                   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

     0983 B0 0D                SHUT4:    MOV     AL,CMOS_REG_D            ; ENABLE NMI AND SET DEFAULT ADDRESS
     0985 E6 70                          OUT     CMOS_PORT,AL
     0987 E4 61                          IN      AL,PORT_B                ; ENABLE PARITY
```

SECTION 5

```
0989 24 F3                     AND     AL,RAM_PAR_ON           ; ENABLE MEMORY PARITY CHECK / I/O CHECK
098B E6 61                     OUT     PORT_B,AL
098D B0 43                     MOV     AL,43H                  ;       <><><><><><><><><><><><><>
098F E6 80                     OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  43 <><>
0991 FB                        STI                             ; ENABLE INTERRUPTS IF DISABLED
0992 CD 19                     INT     19H                     ; GO TO BOOT LOADER
0994 F4                        HLT
0995                   PADING  PROC    NEAR                    ;       INSERT PADDING
0995 B9 000F                   MOV     CX,15                   ; GET BLANK CHARACTER COUNT
0998                   PAD1:
0998 B0 20                     MOV     AL,' '                  ; GET FILL SPACE
099A E8 0000 E                 CALL    PRT_HEX                 ; WRITE A SPACE
099D E2 F9                     LOOP    PAD1                    ; LOOP TILL INSERT DONE
099F B0 2D                     MOV     AL,'-'                  ; GET DASH CHARACTER
09A1 E8 0000 E                 CALL    PRT_HEX                 ; WRITE TO DISPLAY
09A4 C3                        RET
09A5                   PADING  ENDP
09A5                   PRT_OK  PROC    NEAR                    ;       PRINT "00000 KB OK"
09A5 50                        PUSH    AX                      ; SAVE WORK REGISTER
09A6 BB 000A                   MOV     BX,10                   ; SET DECIMAL CONVERT
                       ;-----  CONVERT AND SAVE
09A9 B9 0005                   MOV     CX,5                    ; OF 5 NIBBLES XX,XXX KB
09AC 2B FF                     SUB     DI,DI                   ; DISPLAY REGEN BUFFER POSITION
09AE                   PRT_DIV:
09AE 33 D2                     XOR     DX,DX
09B0 F7 F3                     DIV     BX                      ; DIVIDE BY 10
09B2 80 CA 30                  OR      DL,30H                  ; MAKE INTO ASCII
09B5 52                        PUSH    DX                      ; SAVE
09B6 E2 F6                     LOOP    PRT_DIV
                       ;-----  DISPLAY LAST OK MEMORY
09B8 B9 0005                   MOV     CX,5
09BB                   PRT_DEC:
09BB 58                        POP     AX                      ; RECOVER A NUMBER
09BC E8 0000 E                 CALL    PROT_PRT_HEX
09BF 47                        INC     DI                      ; POINT TO DISPLAY REGEN BUFFER
09C0 E2 F9                     LOOP    PRT_DEC
09C2 B9 0007                   MOV     CX,OFFSET F3B_PAD-OFFSET F3B    ; LOAD MESSAGE LENGTH
09C5 BE 09D4 R                 MOV     SI,OFFSET F3B           ; POINT TO PRINT ' KB OK',' ' MESSAGE
09C8                   PRT_LOOP:
09C8 2E: 8A 04                 MOV     AL,CS:[SI]
09CB 46                        INC     SI
09CC E8 0000 E                 CALL    PROT_PRT_HEX
09CF 47                        INC     DI                      ; INCREMENT BUFF PTR
09D0 E2 F6                     LOOP    PRT_LOOP
09D2 58                        POP     AX                      ; RECOVER WORK REGISTERS
09D3 C3                        RET
09D4 20 4B 42 20 4F 4B F3B     DB      ' KB OK'                ; OK MESSAGE
09DA 20                F3B_OK  DB      ' '                     ; PAD A SPACE
= 09DB                 F3B_PAD EQU     $
                       .LIST
09DB                   PRT_OK  ENDP
                       ;------------------------
                       ;       PRINTER TABLE   :
                       ;------------------------
09DB 03BC              F4      DW      03BCH                   ; ADDRESS OF MONOCHROME PARALLEL ADAPTER
09DD 0378                      DW      0378H                   ; BASE ADDRESS STANDARD PARALLEL ADAPTER
09DF 0278                      DW      0278H                   ; ADDRESS OF ALTERNATE PARALLEL ADAPTER
09E1                   F4E     LABEL   WORD
09E1                   POST2   ENDP
09E1                   CODE    ENDS
                               END
```

```
                                   PAGE 118,121
                                   TITLE TEST3 ---- 11/15/85  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

                                           SMSW    AX                      ; GET THE MACHINE STATUS WORD
 0034  0F 01 E0                +           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

                              ;-----  SET ES REGISTER

0074 6A 48                          PUSH    BYTE PTR ES_TEMP           ; LOAD ES
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

                        ;---------------------------------------
                        ;       VERIFY 286 BOUND INSTRUCTION    :
                        ; DESCRIPTION                           :
                        ;       CREATE A SIGNED ARRAY INDEX     :
                        ;       WITHIN AND OUTSIDE THE LIMITS   :
                        ;       (EXPECT INT 5)                  :
                        ;---------------------------------------

00D1                    T7_6:
00D1 B0 F4                      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
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                    T7_9:
013A B0 F5                      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
```

SECTION 5

```
                                 ;-----  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 0FFFFH
 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
01FF                   + ??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 LOAD SEGMENT LIMIT (LSL)
                         ;-----   AND LOAD ACCESS RIGHTS (LAR) INSTRUCTION

                         ;-----    CHECK THE LAR 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 BB 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

                         ;-----   CHECK THE LSL (LOAD SEGMENT LIMITS)

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 B8 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
0236                   +          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 B8 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
```

SECTION 5

```
                                ;-----  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
028A C6 06 004C 1A                      MOV     BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE),1AH
028F C7 06 004A 0000                    MOV     DS:ES_TEMP.BASE_LO_WORD,0
0295 6A 48                              PUSH    BYTE PTR ES_TEMP         ; LOAD ES REGISTER
0297 07                                 POP     ES
0298 26: C7 05 AA55                     MOV     WORD PTR ES:[DI],0AA55H ; WRITE A TEST PATTERN
                                ;-----  B/W VIDEO CARD
029D 6A 20                              PUSH    BYTE PTR  C_BWCRT_PTR
029F 1F                                 POP     DS                      ; SET DS TO B/W DISPLAY REGEN BUFFER
02A0 8B 05                              MOV     AX,DS:[DI]              ; GET THE WORD FROM B/W VIDEO
                                ;-----  COMPATIBLE COLOR
02A2 6A 28                              PUSH    BYTE PTR C_CCRT_PTR     ; SET DS TO COMPATIBLE COLOR MEMORY
02A4 1F                                 POP     DS
02A5 8B 1D                              MOV     BX,DS:[DI]              ; GET THE WORD FROM COLOR MEMORY
                                ;-----  EGA COLOR
02A7 6A 30                              PUSH    BYTE PTR E_CCRT_PTR             ; EGA COLOR CRT POINTER LOW 64K
02A9 1F                                 POP     DS
02AA 8B 0D                              MOV     CX,DS:[DI]
                                ;-----  TEST FOR ERROR
02AC 50                                 PUSH    AX                      ; SAVE RESULTS
02AD B0 35                              MOV     AL,35H                  ;       <><><><><><><><><><><><>
02AF E6 80                              OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  35 <><>
02B1 58                                 POP     AX
02B2 3D AA55                            CMP     AX,0AA55H
02B5 74 16                              JZ      ERROR_EXIT
02B7 81 FB AA55                         CMP     BX,0AA55H
02BB 74 10                              JZ      ERROR_EXIT
02BD 81 F9 AA55                         CMP     CX,0AA55H
02C1 74 0A                              JZ      ERROR_EXIT
02C3 B0 34                              MOV     AL,34H                  ; RESTORE CHECKPOINT
02C5 E6 80                              OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  34 <><>
                                ;-----  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 ---- 11/15/85  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  DDS
                                         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 1A                              MOV     AL,CMOS_REG_D*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 58                                 POP     AX                      ; RESTORE (AH) AND (AL)= CMOS BYTE
0014 0E                                 PUSH    CS                      ; *PLACE CODE SEGMENT IN STACK AND
0015 E8 0019 R                          CALL    CMOS_POPF               ; *HANDLE POPF FOR B- LEVEL 80286
0018 C3                                 RET                             ; RETURN WITH FLAGS RESTORED

0019                            CMOS_READ       ENDP

0019                            CMOS_POPF       PROC    NEAR            ;       POPF FOR LEVEL B- PARTS
0019 CF                                 IRET                            ; RETURN FAR AND RESTORE FLAGS

001A                            CMOS_POPF       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.         :
                                ;-------------------------------------------------------------------------

001A                            CMOS_WRITE      PROC    NEAR            ;       WRITE (AH) TO LOCATION (AL)
001A 9C                                 PUSHF                           ; SAVE INTERRUPT ENABLE STATUS AND FLAGS
001B 50                                 PUSH    AX                      ; SAVE WORK REGISTER VALUES
001C D0 C0                              ROL     AL,1                    ; MOVE NMI BIT TO LOW POSITION
001E F9                                 STC                             ; FORCE NMI BIT ON IN CARRY FLAG
001F D0 D8                              RCR     AL,1                    ; HIGH BIT ON TO DISABLE NMI - OLD IN CY
0021 FA                                 CLI                             ; DISABLE INTERRUPTS
0022 E6 70                              OUT     CMOS_PORT,AL            ; ADDRESS LOCATION AND DISABLE NMI
0024 8A C4                              MOV     AL,AH                   ; GET THE DATA BYTE TO WRITE
0026 E6 71                              OUT     CMOS_DATA,AL            ; PLACE IN REQUESTED CMOS LOCATION
0028 B0 1A                              MOV     AL,CMOS_REG_D*2         ; GET ADDRESS OF DEFAULT LOCATION
002A D0 D8                              RCR     AL,1                    ; PUT ORIGINAL NMI MASK BIT INTO ADDRESS
002C E6 70                              OUT     CMOS_PORT,AL            ; SET DEFAULT TO READ ONLY REGISTER
002E 58                                 POP     AX                      ; RESTORE WORK REGISTERS
002F 0E                                 PUSH    CS                      ; *PLACE CODE SEGMENT IN STACK AND
0030 E8 0019 R                          CALL    CMOS_POPF               ; *HANDLE POPF FOR B- LEVEL 80286
0033 C3                                 RET

0034                            CMOS_WRITE      ENDP
```

SECTION 5

```
                               PAGE
0034                           DDS      PROC     NEAR                      ;       LOAD (DS) TO DATA AREA
0034 2E: 8E 1E 003A R                   MOV      DS,CS:DDSDATA             ; PUT SEGMENT VALUE OF DATA AREA INTO DS
0039 C3                                 RET                                ; RETURN TO USER WITH (DS)= DATA

003A ---- R                    DDSDATA DW        DATA                      ; SEGMENT SELECTOR VALUE FOR DATA AREA

003C                           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 :
                               ;---------------------------------------------------------------------

003C                           E_MSG    PROC     NEAR
003C F7 C5 3FFF                         TEST     BP,03FFFH                 ; CHECK FOR NOT FIRST ERROR MESSAGE
0040 75 08                              JNZ      E_MSG1                    ; SKIP IF NOT FIRST ERROR MESSAGE

0042 56                                 PUSH     SI                        ; SAVE MESSAGE POINTER
0043 81 E6 3FFF                         AND      SI,03FFFH                 ; USE LOW 14 BITS OF MESSAGE OFFSET
0047 0B EE                              OR       BP,SI                     ;  AS FIRST ERROR MESSAGE FLAG
0049 5E                                 POP      SI                        ; (BIT 0 = E161/E162, BIT 1 = BAD_CONFIG
004A                           E_MSG1:
004A E8 0063 R                          CALL     P_MSG                     ; PRINT MESSAGE
004D 1E                                 PUSH     DS                        ; SAVE CALLERS (DS)
004E E8 0034 R                          CALL     DDS                       ; POINT TO POST/BIOS DATA SEGMENT
0051 F6 06 0010 R 01                    TEST     BYTE PTR @EQUIP_FLAG,01H; LOOP/HALT ON ERROR SWITCH ON ?
0056 74 02                              JZ       MFG_HALT                  ; YES - THEN GO TO MANUFACTURING HALT

0058 1F                                 POP      DS                        ; RESTORE CALLERS (DS)
0059 C3                                 RET

005A                           MFG_HALT:                                   ; MANUFACTURING LOOP MODE ERROR TRAP
005A FA                                 CLI                                ; DISABLE INTERRUPTS
005B A0 0015 R                          MOV      AL,@MFG_ERR_FLAG          ; RECOVER ERROR INDICATOR
005E E6 80                              OUT      MFG_PORT,AL               ; SET INTO MANUFACTURING PORT
0060 F4                                 HLT                                ; HALT SYSTEM
0061 EB F7                              JMP      MFG_HALT                  ; HOT NMI TRAP

0063                           E_MSG    ENDP


0063                           P_MSG    PROC     NEAR                      ;       DISPLAY STRING FROM (CS:)
0063 2E: 8A 04                          MOV      AL,CS:[SI]                ; PUT CHARACTER IN (AL)
0066 46                                 INC      SI                        ; POINT TO NEXT CHARACTER
0067 50                                 PUSH     AX                        ; SAVE PRINT CHARACTER
0068 E8 0128 R                          CALL     PRT_HEX                   ; CALL VIDEO_IO
006B 58                                 POP      AX                        ; RECOVER PRINT CHARACTER
006C 3C 0A                              CMP      AL,LF                     ; WAS IT LINE FEED?
006E 75 F3                              JNE      P_MSG                     ; NO, KEEP PRINTING STRING
0070 C3                                 RET

0071                           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.                          :
                               ;---------------------------------------------------------------------

0071                           ERR_BEEP          PROC     NEAR
0071 9C                                 PUSHF                              ; SAVE FLAGS
0072 FA                                 CLI                                ; DISABLE SYSTEM INTERRUPTS
0073 0A F6                              OR       DH,DH                     ; ANY LONG ONES TO BEEP
0075 74 1E                              JZ       G3                        ; NO, DO THE SHORT ONES
0077                           G1:                                         ;       LONG BEEPS
0077 B3 70                              MOV      BL,112                    ; COUNTER FOR LONG BEEPS (1-3/4 SECONDS)
0079 B9 0500                            MOV      CX,1280                   ; DIVISOR FOR 932 HZ
007C E8 00AF R                          CALL     BEEP                      ; DO THE BEEP
007F B9 C233                            MOV      CX,49715                  ; 2/3 SECOND DELAY AFTER LONG BEEP
0082 E8 00F5 R                          CALL     WAITF                     ; DELAY BETWEEN BEEPS
0085 FE CE                              DEC      DH                        ; ANY MORE LONG BEEPS TO DO
0087 75 EE                              JNZ      G1                        ; LOOP TILL DONE

0089 1E                                 PUSH     DS                        ; SAVE DS REGISTER CONTENTS
008A E8 0034 R                          CALL     DDS
008D 80 3E 0012 R 01                    CMP      @MFG_TST,01H              ; MANUFACTURING TEST MODE?
0092 1F                                 POP      DS                        ; RESTORE ORIGINAL CONTENTS OF (DS)
0093 74 C5                              JE       MFG_HALT                  ; YES - STOP BLINKING LED

0095                           G3:                                         ;       SHORT BEEPS
0095 B3 12                              MOV      BL,18                     ; COUNTER FOR A SHORT BEEP (9/32)
0097 B9 04B8                            MOV      CX,1208                   ; DIVISOR FOR 987 HZ
009A E8 00AF R                          CALL     BEEP                      ; DO THE SOUND
009D B9 8178                            MOV      CX,33144                  ; 1/2 SECOND DELAY AFTER SHORT BEEP
00A0 E8 00F5 R                          CALL     WAITF                     ; DELAY BETWEEN BEEPS
00A3 FE CA                              DEC      DL                        ; DONE WITH SHORT BEEPS COUNT
00A5 75 EE                              JNZ      G3                        ; LOOP TILL DONE

00A7 B9 8178                            MOV      CX,33144                  ; 1/2 SECOND DELAY AFTER LAST BEEP
00AA E8 00F5 R                          CALL     WAITF                     ; MAKE IT ONE SECOND DELAY BEFORE RETURN
00AD 9D                                 POPF                               ; RESTORE FLAGS TO ORIGINAL SETTINGS
00AE C3                                 RET                                ; RETURN TO CALLER

00AF                           ERR_BEEP          ENDP
```

```
                        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.                                       :
                        ;-----------------------------------------------------------------------

00AF                    BEEP    PROC    NEAR                    ;       SETUP TIMER 2
00AF 9C                         PUSHF                           ; SAVE INTERRUPT STATUS
00B0 FA                         CLI                             ; BLOCK INTERRUPTS DURING UPDATE
00B1 B0 B6                      MOV     AL,10110110B            ; SELECT TIMER 2,LSB,MSB,BINARY
00B3 E6 43                      OUT     TIMER+3,AL              ; WRITE THE TIMER MODE REGISTER
00B5 EB 00                      JMP     $+2                     ; I/O DELAY
00B7 8A C1                      MOV     AL,CL                   ; DIVISOR FOR HZ (LOW)
00B9 E6 42                      OUT     TIMER+2,AL              ; WRITE TIMER 2 COUNT - LSB
00BB EB 00                      JMP     $+2                     ; I/O DELAY
00BD 8A C5                      MOV     AL,CH                   ; DIVISOR FOR HZ (HIGH)
00BF E6 42                      OUT     TIMER+2,AL              ; WRITE TIMER 2 COUNT - MSB
00C1 E4 61                      IN      AL,PORT_B               ; GET CURRENT SETTING OF PORT
00C3 8A E0                      MOV     AH,AL                   ; SAVE THAT SETTING
00C5 0C 03                      OR      AL,GATE2+SPK2           ; GATE TIMER 2 AND TURN SPEAKER ON
00C7 E6 61                      OUT     PORT_B,AL               ; AND RESTORE INTERRUPT STATUS
00C9 9D                         POPF
00CA                    G7:                                     ;       1/64 SECOND PER COUNT (BL)
00CA B9 040B                    MOV     CX,1035                 ; DELAY COUNT FOR 1/64 OF A SECOND
00CD E8 00F5 R                  CALL    WAITF                   ; GO TO BEEP DELAY 1/64 COUNT
00D0 FE CB                      DEC     BL                      ; (BL) LENGTH COUNT EXPIRED?
00D2 75 F6                      JNZ     G7                      ; NO - CONTINUE BEEPING SPEAKER

00D4 9C                         PUSHF                           ; SAVE INTERRUPT STATUS
00D5 FA                         CLI                             ; BLOCK INTERRUPTS DURING UPDATE
00D6 E4 61                      IN      AL,PORT_B               ; GET CURRENT PORT VALUE
00D8 0C FC                      OR      AL,NOT (GATE2+SPK2)     ; ISOLATE CURRENT SPEAKER BITS IN CASE
00DA 22 E0                      AND     AH,AL                   ;  SOMEONE TURNED THEM OFF DURING BEEP
00DC 8A C4                      MOV     AL,AH                   ; RECOVER VALUE OF PORT
00DE 24 FC                      AND     AL,NOT (GATE2+SPK2)     ; FORCE SPEAKER DATA OFF
00E0 E6 61                      OUT     PORT_B,AL               ; AND STOP SPEAKER TIMER
00E2 9D                         POPF                            ; RESTORE INTERRUPT FLAG STATE
00E3 B9 040B                    MOV     CX,1035                 ; FORCE 1/64 SECOND DELAY (SHORT)
00E6 E8 00F5 R                  CALL    WAITF                   ; MINIMUM DELAY BETWEEN ALL BEEPS
00E9 9C                         PUSHF                           ; SAVE INTERRUPT STATUS
00EA FA                         CLI                             ; BLOCK INTERRUPTS DURING UPDATE
00EB E4 61                      IN      AL,PORT_B               ; GET CURRENT PORT VALUE IN CASE
00ED 24 03                      AND     AL,GATE2+SPK2           ;  SOMEONE TURNED THEM ON
00EF 0A C4                      OR      AL,AH                   ; RECOVER VALUE OF PORT_B
00F1 E6 61                      OUT     PORT_B,AL               ; RESTORE SPEAKER STATUS
00F3 9D                         POPF                            ; RESTORE INTERRUPT FLAG STATE
00F4 C3                         RET

00F5                    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                                                       :
                        ;-----------------------------------------------------------------------

00F5                    WAITF   PROC    NEAR                    ;       DELAY FOR  (CX)*15.085737 US
00F5 50                         PUSH    AX                      ; SAVE WORK REGISTER (AH)

00F6                    WAITF1:                                 ;       USE TIMER 1 OUTPUT BITS
00F6 E4 61                      IN      AL,PORT_B               ; READ CURRENT COUNTER OUTPUT STATUS
00F8 24 10                      AND     AL,REFRESH_BIT          ; MASK FOR REFRESH DETERMINE BIT
00FA 3A C4                      CMP     AL,AH                   ; DID IT JUST CHANGE
00FC 74 F8                      JE      WAITF1                  ; WAIT FOR A CHANGE IN OUTPUT LINE

00FE 8A E0                      MOV     AH,AL                   ; SAVE NEW FLAG STATE
0100 E2 F4                      LOOP    WAITF1                  ; DECREMENT HALF CYCLES TILL COUNT END

0102 58                         POP     AX                      ; RESTORE (AH)
0103 C3                         RET                             ; RETURN  (CX)= 0

0104                    WAITF   ENDP

                        ;--- CONFIG_BAD -------------------------------------------------------
                        ;       SET CMOS_DIAG WITH CONFIG ERROR BIT (WITH NMI DISABLED)        :
                        ;       (BP) BIT 14 SET ON TO INDICATE CONFIGURATION ERROR             :
                        ;-----------------------------------------------------------------------

0104                    CONFIG_BAD      PROC    NEAR
0104 50                                 PUSH    AX
0105 B8 8E8E                            MOV     AX,X*(CMOS_DIAG+NMI)    ; ADDRESS CMOS DIAGNOSTIC STATUS BYTE
0108 E8 0000 R                          CALL    CMOS_READ               ; GET CURRENT VALUE
010B 0C 20                              OR      AL,BAD_CONFIG           ; SET BAD CONFIGURATION BIT
010D 86 E0                              XCHG    AH,AL                   ; SETUP FOR WRITE
010F E8 001A R                          CALL    CMOS_WRITE              ; UPDATE CMOS WITH BAD CONFIGURATION
0112 58                                 POP     AX
0113 81 CD 4000                         OR      BP,04000H               ; SET CONFIGURATION BAD FLAG IN (BP)
0117 C3                                 RET

0118                    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.                                           :
                               ;-----------------------------------------------------------------------

 0118                          XPC_BYTE        PROC    NEAR            ;       DISPLAY TWO HEX DIGITS
 0118 50                               PUSH    AX                      ; SAVE FOR LOW NIBBLE DISPLAY
 0119 C0 E8 04                         SHR     AL,4                    ; NIBBLE SWAP
 011C E8 0122 R                        CALL    XLAT_PR                 ; DO THE HIGH NIBBLE DISPLAY
 011F 58                               POP     AX                      ; RECOVER THE NIBBLE
 0120 24 0F                            AND     AL,0FH                  ; ISOLATE TO LOW NIBBLE
                                                                       ; FALL INTO LOW NIBBLE CONVERSION

 0122                          XLAT_PR PROC    NEAR                    ;       CONVERT 00-0F TO ASCII CHARACTER
 0122 04 90                            ADD     AL,090H                 ; ADD FIRST CONVERSION FACTOR
 0124 27                               DAA                             ; ADJUST FOR NUMERIC AND ALPHA RANGE
 0125 14 40                            ADC     AL,040H                 ; ADD CONVERSION AND ADJUST LOW NIBBLE
 0127 27                               DAA                             ; ADJUST HIGH NIBBLE TO ASCII RANGE

 0128                          PRT_HEX PROC    NEAR
 0128 B4 0E                            MOV     AH,0EH                  ; DISPLAY CHARACTER IN (AL) COMMAND
 012A B7 00                            MOV     BH,0
 012C CD 10                            INT     10H                     ; CALL VIDEO_IO
 012E C3                               RET

 012F                          PRT_HEX ENDP
 012F                          XLAT_PR ENDP
 012F                          XPC_BYTE        ENDP

                               ;--- PRT_SEG -------------------------------------------------------
                               ;       PRINT A SEGMENT VALUE TO LOOK LIKE A 21 BIT ADDRESS       :
                               ;       DX MUST CONTAIN SEGMENT VALUE TO BE PRINTED               :
                               ;------------------------------------------------------------------

 012F                          PRT_SEG PROC    NEAR
 012F 8A C6                            MOV     AL,DH                   ; GET MSB
 0131 E8 0118 R                        CALL    XPC_BYTE                ; DISPLAY SEGMENT HIGH BYTE
 0134 8A C2                            MOV     AL,DL                   ; LSB
 0136 E8 0118 R                        CALL    XPC_BYTE                ; DISPLAY SEGMENT LOW BYTE
 0139 B0 30                            MOV     AL,'0'                  ; PRINT A '0 '
 013B E8 0128 R                        CALL    PRT_HEX                 ;  TO MAKE LOOK LIKE ADDRESS
 013E B0 20                            MOV     AL,' '                  ; ADD ENDING SPACE
 0140 E8 0128 R                        CALL    PRT_HEX
 0143 C3                               RET

 0144                          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                            :
                               ;-----------------------------------------------------------------------

 0144                          PROT_PRT_HEX    PROC    NEAR
 0144 06                               PUSH    ES                      ; SAVE CURRENT SEGMENT REGISTERS
 0145 57                               PUSH    DI
 0146 D1 E7                            SAL     DI,1                    ; MULTIPLY OFFSET BY TWO

                               ;-----  MONOCHROME VIDEO CARD

 0148 6A 20                            PUSH    BYTE PTR C_BWCRT_PTR    ; GET MONOCHROME BUFFER SEGMENT SELECTOR
 014A 07                               POP     ES                      ; SET (ES) TO B/W DISPLAY BUFFER
 014B AA                               STOSB                           ; PLACE CHARACTER IN BUFFER
 014C 4F                               DEC     DI                      ; ADJUST POINTER BACK

                               ;-----  ENHANCED GRAPHICS ADAPTER

 014D 6A 30                            PUSH    BYTE PTR E_CCRT_PTR     ; ENHANCED COLOR DISPLAY POINTER LOW 64K
 014F 07                               POP     ES                      ; LOAD SEGMENT SELECTOR
 0150 AA                               STOSB                           ; PLACE CHARACTER IN BUFFER
 0151 4F                               DEC     DI                      ; ADJUST POINTER BACK
 0152 6A 38                            PUSH    BYTE PTR E_CCRT_PTR2    ; ENHANCED COLOR DISPLAY POINTER HI 64K
 0154 07                               POP     ES                      ; LOAD SEGMENT SELECTOR
 0155 AA                               STOSB                           ; PLACE CHARACTER IN BUFFER
 0156 4F                               DEC     DI                      ; ADJUST POINTER BACK

                               ;-----  COMPATIBLE COLOR

 0157 6A 28                            PUSH    BYTE PTR C_CCRT_PTR     ; SET (DS) TO COMPATIBLE COLOR MEMORY
 0159 07                               POP     ES
 015A 53                               PUSH    BX                      ; SAVE WORK REGISTERS
 015B 52                               PUSH    DX
 015C 51                               PUSH    CX
 015D 33 C9                            XOR     CX,CX                   ; TIMEOUT LOOP FOR "BAD" HARDWARE
 015F BA 03DA                          MOV     DX,03DAH                ; STATUS ADDRESS OF COLOR CARD
 0162 93                               XCHG    AX,BX                   ; SAVE IN (BX) REGISTER
 0163                          PROT_S:
 0163 EC                               IN      AL,DX                   ; GET COLOR CARD STATUS
 0164 A8 09                            TEST    AL,RVRT+RHRZ            ; CHECK FOR VERTICAL RETRACE (OR HORZ)
 0166 E1 FB                            LOOPZ   PROT_S                  ; TIMEOUT LOOP TILL FOUND
 0168 93                               XCHG    AX,BX                   ; RECOVER CHARACTERS
 0169 AA                               STOSB                           ; PLACE CHARACTER IN BUFFER

 016A 59                               POP     CX                      ; RESTORE REGISTERS
 016B 5A                               POP     DX
 016C 5B                               POP     BX
 016D 5F                               POP     DI
 016E 07                               POP     ES
 016F C3                               RET

 0170                          PROT_PRT_HEX    ENDP
```

```
                                PAGE
                                ;----------------------------------------
                                ;       ROM CHECKSUM SUBROUTINE         :
                                ;----------------------------------------

 0170                           ROM_CHECKSUM    PROC    NEAR
 0170 2B C9                             SUB     CX,CX                   ; NUMBER OF BYTES TO ADD IS 64K

 0172                           ROM_CHECKSUM_CNT:                       ; ENTRY FOR OPTIONAL ROM TEST
 0172 32 C0                             XOR     AL,AL
 0174                           ROM_L:
 0174 02 07                             ADD     AL,[BX]                 ; GET (DS:BX)
 0176 43                                INC     BX                      ; POINT TO NEXT BYTE
 0177 E2 FB                             LOOP    ROM_L                   ; ADD ALL BYTES IN ROM MODULE

 0179 0A C0                             OR      AL,AL                   ; SUM = 0?
 017B C3                                RET

 017C                           ROM_CHECKSUM    ENDP

                                ;---------------------------------------------------------------------
                                ;       THIS ROUTINE CHECKSUMS OPTIONAL ROM MODULES AND             :
                                ;       IF CHECKSUM IS OK, CALLS INITIALIZATION/TEST CODE IN MODULE :
                                ;---------------------------------------------------------------------

 017C                           ROM_CHECK       PROC    NEAR
 017C B8 ---- R                         MOV     AX,DATA                 ; POINT ES TO DATA AREA
 017F 8E C0                             MOV     ES,AX
 0181 2A E4                             SUB     AH,AH                   ; ZERO OUT AH
 0183 8A 47 02                          MOV     AL,[BX+2]               ; GET LENGTH INDICATOR
 0186 C1 E0 09                          SHL     AX,9                    ; MULTIPLY BY 512
 0189 8B C8                             MOV     CX,AX                   ; SET COUNT
 018B C1 E8 04                          SHR     AX,4
 018E 03 D0                             ADD     DX,AX                   ; SET POINTER TO NEXT MODULE
 0190 E8 0172 R                         CALL    ROM_CHECKSUM_CNT        ; DO CHECKSUM
 0193 74 05                             JZ      ROM_CHECK_1

 0195 E8 0000 E                         CALL    ROM_ERR                 ; POST CHECKSUM ERROR
 0198 EB 13                             JMP     SHORT ROM_CHECK_END     ; AND EXIT

 019A                           ROM_CHECK_1:
 019A 52                                PUSH    DX                      ; SAVE POINTER
 019B 26: C7 06 0067 R 0003             MOV     ES:@IO_ROM_INIT,0003H   ; LOAD OFFSET
 01A2 26: 8C 1E 0069 R                  MOV     ES:@IO_ROM_SEG,DS       ; LOAD SEGMENT
 01A7 26: FF 1E 0067 R                  CALL    DWORD PTR ES:@IO_ROM_INIT; CALL INITIALIZE/TEST ROUTINE
 01AC 5A                                POP     DX

 01AD                           ROM_CHECK_END:
 01AD C3                                RET                             ; RETURN TO CALLER

 01AE                           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          :
                                ;---------------------------------------------------------------------

 01AE                           KBD_RESET       PROC    NEAR
 01AE B0 FF                             MOV     AL,0FFH                 ; SET KEYBOARD RESET COMMAND
 01B0 E8 0000 E                         CALL    XMIT_8042               ; GO ISSUE THE COMMAND
 01B3 E3 23                             JCXZ    G13                     ; EXIT IF ERROR

 01B5 3C FA                             CMP     AL,KB_ACK
 01B7 75 1F                             JNZ     G13

 01B9 B0 FD                             MOV     AL,0FDH                 ; ENABLE KEYBOARD INTERRUPTS
 01BB E6 21                             OUT     INTA01,AL               ; WRITE 8259 INTERRUPT MASK REGISTER
 01BD C6 06 006B R 00                   MOV     @INTR_FLAG,0            ; RESET INTERRUPT INDICATOR
 01C2 FB                                STI                             ; ENABLE INTERRUPTS
 01C3 B3 0A                             MOV     BL,10                   ; TRY FOR 400 MILLISECONDS
 01C5 2B C9                             SUB     CX,CX                   ; SETUP INTERRUPT TIMEOUT COUNT
 01C7                           G11:
 01C7 F6 06 006B R 02                   TEST    @INTR_FLAG,02H          ; DID A KEYBOARD INTERRUPT OCCUR ?
 01CC 75 06                             JNZ     G12                     ; YES - READ SCAN CODE RETURNED
 01CE E2 F7                             LOOP    G11                     ; NO - LOOP TILL TIMEOUT

 01D0 FE CB                             DEC     BL
 01D2 75 F3                             JNZ     G11                     ; TRY AGAIN
 01D4                           G12:
 01D4 E4 60                             IN      AL,PORT_A               ; READ KEYBOARD SCAN CODE
 01D6 8A D8                             MOV     BL,AL                   ; SAVE SCAN CODE JUST READ
 01D8                           G13:
 01D8 C3                                RET                             ; RETURN TO CALLER

 01D9                           KBD_RESET       ENDP


                                ;----------------------------------------------------
                                ;       BLINK LED PROCEDURE FOR MFG RUN-IN TESTS    :
                                ;        IF LED IS ON, TURN IT OFF. IF OFF, TURN ON. :
                                ;----------------------------------------------------

 01D9                           BLINK_INT       PROC    NEAR
 01D9 FB                                STI
 01DA 50                                PUSH    AX                      ; SAVE AX REGISTER CONTENTS
 01DB E4 80                             IN      AL,MFG_PORT             ; READ CURRENT VALUE OF MFG_PORT
 01DD 34 40                             XOR     AL,01000000B            ; FLIP CONTROL BIT
 01DF E6 80                             OUT     MFG_PORT,AL
 01E1 B0 20                             MOV     AL,EOI
 01E3 E6 20                             OUT     INTA00,AL
 01E5 58                                POP     AX                      ; RESTORE AX REGISTER
 01E6 CF                                IRET

 01E7                           BLINK_INT       ENDP
```

SECTION 5

```
                                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

 01E7                           SET_TOD PROC    NEAR
 01E7 60                                PUSHA
 01E8 1E                                PUSH    DS
 01E9 E8 0034 R                         CALL    DDS                     ; ESTABLISH SEGMENT
 01EC 2B C0                             SUB     AX,AX
 01EE A2 0070 R                         MOV     @TIMER_OFL,AL           ; RESET TIMER ROLL OVER INDICATOR
 01F1 A3 006C R                         MOV     @TIMER_LOW,AX           ; AND TIMER COUNT
 01F4 A3 006E R                         MOV     @TIMER_HIGH,AX
 01F7 B0 8E                             MOV     AL,CMOS_DIAG+NMI        ; CHECK CMOS VALIDITY
 01F9 E8 0000 R                         CALL    CMOS_READ               ; READ DIAGNOSTIC LOCATION IN CMOS
 01FC 24 C4                             AND     AL,BAD_BAT+BAD_CKSUM+CMOS_CLK_FAIL
                                                                        ; BAD BATTERY, CHKSUM ERROR, CLOCK ERROR
 01FE 75 68                             JNZ     POD_DONE                ; CMOS NOT VALID -- TIMER SET TO ZERO
 0200 2B C9                             SUB     CX,CX                   ;
 0202                           UIP:
 0202 B0 8A                             MOV     AL,CMOS_REG_A+NMI       ; ACCESS REGISTER A
 0204 E8 0000 R                         CALL    CMOS_READ               ; READ CMOS CLOCK REGISTER A
 0207 A8 80                             TEST    AL,UPDATE_TIMER
 0209 E1 F7                             LOOPZ   UIP                     ; WAIT TILL UPDATE BIT IS ON

 020B E3 5B                             JCXZ    POD_DONE                ; CMOS CLOCK STUCK IF TIMEOUT
 020D 2B C9                             SUB     CX,CX                   ;
 020F                           UIPOFF:
 020F B0 8A                             MOV     AL,CMOS_REG_A+NMI       ; ACCESS REGISTER A
 0211 E8 0000 R                         CALL    CMOS_READ               ; READ CMOS CLOCK REGISTER A
 0214 A8 80                             TEST    AL,UPDATE_TIMER
 0216 E0 F7                             LOOPNZ  UIPOFF                  ; NEXT WAIT TILL END OF UPDATE

 0218 E3 4E                             JCXZ    POD_DONE                ; CMOS CLOCK STUCK IF TIMEOUT

 021A B0 80                             MOV     AL,CMOS_SECONDS+NMI     ;       TIME JUST UPDATED
 021C E8 0000 R                         CALL    CMOS_READ               ; ACCESS SECONDS VALUE IN CMOS
 021F 3C 59                             CMP     AL,59H                  ; ARE THE SECONDS WITHIN LIMITS?
 0221 77 48                             JA      TOD_ERROR               ; GO IF NOT

 0223 E8 0281 R                         CALL    CVT_BINARY              ; CONVERT IT TO BINARY
 0226 8B C8                             MOV     CX,AX                   ; MOVE COUNT TO ACCUMULATION REGISTER
 0228 C1 E9 02                          SHR     CX,2                    ; ADJUST FOR SYSTEMATIC SECONDS ERROR
 022B B3 12                             MOV     BL,COUNTS_SEC
 022D F6 E3                             MUL     BL                      ; COUNT FOR SECONDS
 022F 03 C8                             ADD     CX,AX
 0231 B0 82                             MOV     AL,CMOS_MINUTES+NMI
 0233 E8 0000 R                         CALL    CMOS_READ               ; ACCESS MINUTES VALUE IN CMOS
 0236 3C 59                             CMP     AL,59H                  ; ARE THE MINUTES WITHIN LIMITS?
 0238 77 31                             JA      TOD_ERROR               ; GO IF NOT
 023A E8 0281 R                         CALL    CVT_BINARY              ; CONVERT IT TO BINARY
 023D 50                                PUSH    AX                      ; SAVE MINUTES COUNT
 023E D1 E8                             SHR     AX,1                    ; ADJUST FOR SYSTEMATIC MINUTES ERROR
 0240 03 C8                             ADD     CX,AX                   ; ADD ADJUSTMENT TO COUNT
 0242 58                                POP     AX                      ; RECOVER BCD MINUTES VALUE
 0243 BB 0444                           MOV     BX,COUNTS_MIN
 0246 F7 E3                             MUL     BX                      ; COUNT FOR MINUTES
 0248 03 C8                             ADD     CX,AX                   ; ADD TO ACCUMULATED VALUE
 024A B0 84                             MOV     AL,CMOS_HOURS+NMI
 024C E8 0000 R                         CALL    CMOS_READ               ; ACCESS HOURS VALUE IN CMOS
 024F 3C 23                             CMP     AL,23H                  ; ARE THE HOURS WITHIN LIMITS?
 0251 77 18                             JA      TOD_ERROR               ; GO IF NOT

 0253 E8 0281 R                         CALL    CVT_BINARY              ; CONVERT IT TO BINARY
 0256 8B D0                             MOV     DX,AX
 0258 B3 07                             MOV     BL,COUNTS_HOUR
 025A F6 E3                             MUL     BL                      ; COUNT FOR HOURS
 025C 03 C1                             ADD     AX,CX
 025E 83 D2 00                          ADC     DX,0000H
 0261 89 16 006E R                      MOV     @TIMER_HIGH,DX
 0265 A3 006C R                         MOV     @TIMER_LOW,AX
 0268                           POD_DONE:
 0268 1F                                POP     DS
 0269 61                                POPA
 026A C3                                RET

 026B                           TOD_ERROR:
 026B 1F                                POP     DS                      ; RESTORE SEGMENT
 026C 61                                POPA                            ; RESTORE REGISTERS
 026D BE 0000 E                         MOV     SI,OFFSET E163          ; DISPLAY CLOCK ERROR
 0270 E8 003C R                         CALL    E_MSG
 0273 B8 8E8E                           MOV     AX,X*(CMOS_DIAG+NMI)    ; SET CLOCK ERROR IN STATUS
 0276 E8 0000 R                         CALL    CMOS_READ               ; READ DIAGNOSTIC CMOS LOCATION
 0279 0C 04                             OR      AL,CMOS_CLK_FAIL        ; SET NEW STATUS WITH CMOS CLOCK ERROR
 027B 86 C4                             XCHG    AL,AH                   ; MOVE NEW STATUS TO WORK REGISTER
 027D E8 001A R                         CALL    CMOS_WRITE              ; UPDATE STATUS LOCATION
 0280 C3                                RET

 0281                           SET_TOD ENDP

 0281                           CVT_BINARY      PROC    NEAR
 0281 8A E0                             MOV     AH,AL                   ; UNPACK 2 BCD DIGITS IN AL
 0283 C0 EC 04                          SHR     AH,4
 0286 24 0F                             AND     AL,0FH                  ; RESULT IS IN AX
 0288 D5 0A                             AAD                             ; CONVERT UNPACKED BCD TO BINARY
 028A C3                                RET

 028B                           CVT_BINARY      ENDP
```

```
                                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. :
                                ;-----------------------------------------------------------------------------

 028B                           D11     PROC    NEAR
 028B 50                                PUSH    AX                      ; SAVE REGISTER AX CONTENTS
 028C 53                                PUSH    BX
 028D B0 0B                             MOV     AL,0BH                  ; READ IN-SERVICE REGISTER
 028F E6 20                             OUT     INTA00,AL               ; (FIND OUT WHAT LEVEL BEING
 0291 EB 00                             JMP     $+2                     ;  SERVICED)
 0293 E4 20                             IN      AL,INTA00               ; GET LEVEL
 0295 8A E0                             MOV     AH,AL                   ; SAVE IT
 0297 0A C4                             OR      AL,AH                   ; 00? (NO HARDWARE ISR ACTIVE)
 0299 75 04                             JNZ     HW_INT

 029B B4 FF                             MOV     AH,0FFH
 029D EB 2F                             JMP     SHORT SET_INTR_FLAG     ; SET FLAG TO "FF" IF NON-HARDWARE
 029F                           HW_INT:
 029F B0 0B                             MOV     AL,0BH                  ; READ IN-SERVICE REGISTER FROM
 02A1 E6 A0                             OUT     INTB00,AL               ; INTERRUPT CHIP #2
 02A3 EB 00                             JMP     $+2                     ; I/O DELAY
 02A5 E4 A0                             IN      AL,INTB00               ; CHECK THE SECOND INTERRUPT CHIP
 02A7 8A F8                             MOV     BH,AL                   ; SAVE IT
 02A9 0A FF                             OR      BH,BH
 02AB 74 10                             JZ      NOT_SEC                 ; CONTINUE IF NOT

 02AD E4 A1                             IN      AL,INTB01               ; GET SECOND INTERRUPT MASK
 02AF 0A C7                             OR      AL,BH                   ; MASK OFF LEVEL BEING SERVICED
 02B1 EB 00                             JMP     $+2                     ; I/O DELAY
 02B3 E6 A1                             OUT     INTB01,AL
 02B5 B0 20                             MOV     AL,EOI                  ; SEND EOI TO SECOND CHIP
 02B7 EB 00                             JMP     $+2                     ; I/O DELAY
 02B9 E6 A0                             OUT     INTB00,AL
 02BB EB 0D                             JMP     SHORT IS_SEC
 02BD                           NOT_SEC:
 02BD E4 21                             IN      AL,INTA01               ; GET CURRENT MASK VALUE
 02BF EB 00                             JMP     $+2                     ; I/O DELAY
 02C1 80 E4 FB                          AND     AH,0FBH                 ; DO NOT DISABLE SECOND CONTROLLER
 02C4 0A C4                             OR      AL,AH                   ; MASK OFF LEVEL BEING SERVICED
 02C6 E6 21                             OUT     INTA01,AL               ; SET NEW INTERRUPT MASK
 02C8 EB 00                             JMP     $+2                     ; I/O DELAY
 02CA                           IS_SEC:
 02CA B0 20                             MOV     AL,EOI
 02CC E6 20                             OUT     INTA00,AL
 02CE                           SET_INTR_FLAG:
 02CE 5B                                POP     BX                      ; RESTORE (BX) FROM STACK
 02CF 1E                                PUSH    DS                      ; SAVE ACTIVE (DS)
 02D0 E8 0034 R                         CALL    DDS                     ; SET DATA SEGMENT
 02D3 88 26 006B R                      MOV     @INTR_FLAG,AH           ; SET FLAG
 02D7 1F                                POP     DS
 02D8 58                                POP     AX                      ; RESTORE REGISTER AX CONTENTS
 02D9                           DUMMY_RETURN_1:                         ; NEED IRET FOR VECTOR TABLE
 02D9 CF                                IRET

 02DA                           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                        :
                                ;-----------------------------------------------------------------------------

 02DA                           RE_DIRECT PROC  NEAR
 02DA 50                                PUSH    AX                      ; SAVE (AX)
 02DB B0 20                             MOV     AL,EOI
 02DD E6 A0                             OUT     INTB00,AL               ; EOI TO SLAVE INTERRUPT CONTROLLER
 02DF 58                                POP     AX                      ; RESTORE (AX)
 02E0 CD 0A                             INT     0AH                     ; GIVE CONTROL TO HARDWARE LEVEL 2

 02E2 CF                                IRET                            ; RETURN

 02E3                           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.                                                       :
                                ;-----------------------------------------------------------------------------

 02E3                           INT_287 PROC    NEAR
 02E3 50                                PUSH    AX                      ; SAVE (AX)
 02E4 32 C0                             XOR     AL,AL
 02E6 E6 F0                             OUT     X287,AL                 ; REMOVE THE INTERRUPT REQUEST

 02E8 B0 20                             MOV     AL,EOI                  ; ENABLE THE INTERRUPT
 02EA E6 A0                             OUT     INTB00,AL               ;  THE SLAVE
 02EC E6 20                             OUT     INTA00,AL               ;  THE MASTER
 02EE 58                                POP     AX                      ; RESTORE (AX)
 02EF CD 02                             INT     02H                     ; GIVE CONTROL TO NMI

 02F1 CF                                IRET                            ; RETURN

 02F2                           INT_287 ENDP

 02F2                           PROC_SHUTDOWN   PROC                    ;       COMMON 80286 SHUTDOWN WAIT

 02F2 B0 FE                             MOV     AL,SHUT_CMD             ; SHUTDOWN COMMAND
 02F4 E6 64                             OUT     STATUS_PORT,AL          ; SEND TO KEYBOARD CONTROL PORT
 02F6                           PROC_S:
 02F6 F4                                HLT                             ; WAIT FOR 80286 RESET
 02F7 EB FD                             JMP     PROC_S                  ; INSURE HALT

 02F9                           PROC_SHUTDOWN   ENDP
 02F9                           CODE    ENDS
                                        END
```

SECTION 5

```
                            PAGE 118,121
                            TITLE TEST5 ---- 11/15/85  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                       IRET                            ; RETURN

                         ;--------------------------------------------------------------------------
                         ;            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,DI                    ; 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
011B AB                       STOSW                           ; PUT THAT IN THE LIMIT FIELD
```

SECTION 5

```
0 11C B8 D8A0                       MOV     AX,GDT_LOC              ; AX = LOW WORD OF GDT ADDRESS
011F AB                             STOSW                           ; PUT THAT IN BASE FIELD - LOW
0120 B8 0000                        MOV     AX,0                    ; AX = HIGH BYTE OF ADDRESS, AND
0123 AB                             STOSW                           ;   ACCESS RIGHTS BYTE IS UNDEFINED
                                    SEGOV   ES                      ; LOAD THE GDTR
0124 26                 +           DB      026H
                                    LGDT    [BP]                    ;   FROM THIS AREA
0125 0F                 +           DB      00FH
0126                    + ??0004    LABEL   BYTE
0126 8B 56 00           +           MOV     DX,WORD PTR [BP]
0129                    + ??0005    LABEL   BYTE
0126                    +           ORG     OFFSET CS:??0004
0126 01                 +           DB      001H
0129                    +           ORG     OFFSET CS:??0005
0129 8B FD                          MOV     DI,BP                   ; RESTORE THE ES:DI POINTER
012B AB                             STOSW
012C AB                             STOSW
012D 8B FD                          MOV     DI,BP

                        ;-----  SWITCH TO VIRTUAL MODE

012F 5D                             POP     BP                      ; RESTORE BP
0130 B8 0001                        MOV     AX,VIRTUAL_ENABLE       ; MACHINE STATUS WORD NEEDED TO
                                    LMSW    AX                      ;  SWITCH TO VIRTUAL MODE
0133 0F 01 F0           +           DB      00FH,001H,0F0H

0136 EA                             DB      0EAH                    ; JUMP FAR TO PURGE PRE-FETCH QUEUE
0137 013B R                         DW      OFFSET DONE             ;   TO OFFSET
0139 0040                           DW      SYS_ROM_CS              ;   IN SEGMENT
013B                    DONE:
013B B0 85                          MOV     AL,85H                  ;       <><><><><><><><><><><><>
013D E6 80                          OUT     MFG_PORT,AL             ;       <><> CHECKPOINT  82 <><>
013F C3                             RET                             ; SYSTEM INITIALIZATION

0140                    SYSINITI            ENDP


0140                    GDT_BLD PROC        NEAR
0140 BE 01AF R                      MOV     SI,OFFSET GDT_DATA_START           ; DS:SI --> GDT
0143 B9 0044                        MOV     CX,(OFFSET GDT_DATA_END-OFFSET GDT_DATA_START)/2  ; WORD COUNT
0146 F3/ A5                         REP     MOVSW                              ; COPY GDT INTO MEMORY
0148 C3                             RET
0149                    GDT_BLD ENDP


0149                    SIDT_BLD            PROC    NEAR

                        ;-----  BUILD THE IDT.  THE IDT WILL CONTAIN VECTORS FOR EXCEPTION HANDLERS

0149 BE 0237 R                      MOV     SI,OFFSET SYS_IDT_OFFSETS  ; MAKE DS:SI POINT TO
014C 8C C8                          MOV     AX,CS                   ;     INTERRUPT ENTRY POINTS
014E 8E D8                          MOV     DS,AX
0150 BF D0A0                        MOV     DI,SYS_IDT_LOC          ; POINT TO SYS_IDT_LOC
0153 2B C0                          SUB     AX,AX
0155 8E C0                          MOV     ES,AX                   ;     WHERE THE IDT WILL BE.
0157 BB 0040                        MOV     BX,SYS_ROM_CS           ; CS IS THE SAME FOR ALL INTERRUPTS
015A B6 87                          MOV     DH,TRAP_GATE            ; ACCESS RIGHTS BYTE FOR THE GATE
015C B2 00                          MOV     DL,0                    ; THE WORD COUNT FIELD IS UNUSED
015E B9 0020                        MOV     CX,32                   ; THERE ARE 32 RESERVED INTERRUPTS
0161                    LOW_IDT:                                    ; THIS LOOP BUILDS 32 DESCRIPTORS IN THE
                                                                    ;   IDT FOR THE RESERVED INTERRUPTS
0161 A5                             MOVSW                           ; GET A ROUTINE ENTRY POINT
                                                                    ;   AND PUT IT IN THE OFFSET FIELD
0162 8B C3                          MOV     AX,BX                   ; GET THE SYSTEM CODE SEGMENT SELECTOR
0164 AB                             STOSW                           ;   AND PUT IT IN THE SELECTOR FIELD
0165 8B C2                          MOV     AX,DX                   ; GET THE INTERRUPT GATE BYTE
0167 AB                             STOSW                           ;   AND PUT IN THE ACCESS RIGHTS FIELD
0168 B8 0000                        MOV     AX,0                    ; ZERO OUT
016B AB                             STOSW                           ;   THE RESERVED POSITIONS
016C E2 F3                          LOOP    LOW_IDT                 ;     AND REPEAT AS DIRECTED
016E B9 00E0                        MOV     CX,256-32               ; 256 TOTAL - 32 DONE = WHATEVER IS LEFT
0171 BD 0277 R                      MOV     BP,OFFSET FREE_INTS     ; THERE IS A COPY OF AN UN-INITIALIZED
                                                                    ;   INTERRUPT DESCRIPTOR AT FREE_INTS
0174                    HIGH_IDT:
0174 8B F5                          MOV     SI,BP                   ; DS:SI --> FREE DESCRIPTOR
                                                                    ; (ES:DI LEFT OFF AT INT 32)
0176 A5                             MOVSW                           ; MOVE OFFSET OF THE IRET INSTRUCTION
0177 A5                             MOVSW                           ; MOVE THE CS SELECTOR
0178 A5                             MOVSW                           ; MOVE THE ACCESS RIGHTS BYTE
0179 AB                             STOSW                           ; ZERO OUT THE RESERVED WORD
017A E2 F8                          LOOP    HIGH_IDT                ; FILL THE REMAINDER OF THE TABLE

                        ;-----  INITIALIZE THE ENTRY POINTS FOR POST TEST

017C 26: C7 06 D1A0 0098 R          MOV     ES:(SYS_IDT_LOC+(032*DESC_LEN).ENTRY_POINT),OFFSET SYS_32
0183 26: C7 06 D1A8 009C R          MOV     ES:(SYS_IDT_LOC+(033*DESC_LEN).ENTRY_POINT),OFFSET SYS_33
018A 26: C7 06 D1B0 00A0 R          MOV     ES:(SYS_IDT_LOC+(034*DESC_LEN).ENTRY_POINT),OFFSET SYS_34
0191 26: C7 06 D1B8 00A4 R          MOV     ES:(SYS_IDT_LOC+(035*DESC_LEN).ENTRY_POINT),OFFSET SYS_35
0198 26: C7 06 D1C0 00A8 R          MOV     ES:(SYS_IDT_LOC+(036*DESC_LEN).ENTRY_POINT),OFFSET SYS_36
019F 26: C7 06 D1C8 00AC R          MOV     ES:(SYS_IDT_LOC+(037*DESC_LEN).ENTRY_POINT),OFFSET SYS_37
01A6 26: C7 06 D1D0 00B0 R          MOV     ES:(SYS_IDT_LOC+(038*DESC_LEN).ENTRY_POINT),OFFSET SYS_38
01AD C3                             RET

01AE                    IRET_ADDR           LABEL   WORD            ; FOR UN-INITIALIZED INTERRUPTS
01AE CF                             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
```

```
                        PAGE 118,121
                        TITLE TEST6 ---- 11/15/85  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   E602: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
```

SECTION 5

```
                         ;-----  PARITY ERROR EXIT

0069 B8 0000                     MOV     AX,0                    ; RESTORE AX TO 0000
006C 75 1A                       JNZ     C13                     ; GO IF YES

                         ;-----  CHECK FOR END OF 64K BLOCK

006E 23 D2                       AND     DX,DX                   ; ENDING ZERO PATTERN WRITTEN TO MEMORY?
0070 74 16                       JZ      C13                     ; YES - RETURN TO CALLER WITH AL=0

                         ;-----  SETUP NEXT PATTERN

0072 81 FA 55AA                  CMP     DX,055AAH               ; CHECK IF LAST PATTERN =55AA
0076 74 0B                       JZ      C9                      ; GO IF NOT
0078 81 FA 0101                  CMP     DX,0101H                ; LAST PATTERN 0101?
007C 74 0B                       JZ      C10                     ; GO IF YES
007E BA 55AA                     MOV     DX,055AAH               ; WRITE 55AA TO STORAGE
0081 EB C9                       JMP     C3

                         ;-----  INSURE PARITY BITS ARE NOT STUCK ON

0083 BA 0101             C9:     MOV     DX,0101H                ; WRITE 0101 TO STORAGE
0086 EB C4                       JMP     C3

                         ;-----  EXIT STORAGE TEST
0088                     C13:
0088 C3                          RET                             ; ERROR IF ZF NOT SET

                         ;-----  CHECKER BOARD TEST

0089 2B FF               C10:    SUB     DI,DI                   ; POINT TO START OF BLOCK
008B 8B CB                       MOV     CX,BX                   ; GET THE BLOCK COUNT
008D D1 E9                       SHR     CX,1                    ; DIVIDE BY 2
008F B8 AAAA                     MOV     AX,1010101010101010B    ; SECOND CHECKER PATTERN
0092 BE 5555                     MOV     SI,0101010101010101B    ; FIRST CHECKER PATTERN
0095                     C11:
0095 96                          XCHG    AX,SI                   ; FIRST CHECKER PATTERN TO AX
0096 AB                          STOSW                           ; WRITE IT TO MEMORY
0097 96                          XCHG    AX,SI                   ; SECOND CHECKER PATTERN TO AX
0098 AB                          STOSW                           ; WRITE IT TO MEMORY
0099 E2 FA                       LOOP    C11                     ; DO IT FOR CX COUNT

009B 2B F6                       SUB     SI,SI                   ; POINT TO START OF BLOCK
009D 8B CB                       MOV     CX,BX                   ; GET THE BLOCK COUNT
009F D1 E9                       SHR     CX,1                    ; DIVIDE BY 2
00A1 BF 5555                     MOV     DI,0101010101010101B    ; CHECK CORRECT
00A4 BA AAAA                     MOV     DX,1010101010101010B
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 D5                       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 0015 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
```

SECTION 5

```
                                ;-----  CLEAR @BOOT_LOCN
017E 33 C0                              XOR     AX,AX
0180 B9 0100                            MOV     CX,256                  ; CLEAR 256 WORDS
0183 BF 7C00 R                          MOV     DI,OFFSET @BOOT_LOCN
0186 F3/ AB                             REP     STOSW
                                ;-----  LOAD SYSTEM FROM DISKETTE -- CX HAS RETRY COUNT
0188 FB                                 STI
0189 B9 0004                            MOV     CX,4                    ; SET RETRY COUNT
018C 51                         H1:     PUSH    CX                      ; IPL SYSTEM
018D B4 00                              MOV     AH,0                    ; RESET THE DISKETTE SYSTEM
018F CD 13                              INT     13H                     ; DISKETTE_IO
0191 72 0F                              JC      H2                      ; IF ERROR, TRY AGAIN
0193 B8 0201                            MOV     AX,201H                 ; READ IN THE SINGLE SECTOR
0196 2B D2                              SUB     DX,DX                   ; TO THE BOOT LOCATION
0198 8E C2                              MOV     ES,DX
019A BB 7C00 R                          MOV     BX,OFFSET @BOOT_LOCN    ; DRIVE 0, HEAD 0
019D B9 0001                            MOV     CX,1                    ; SECTOR 1, TRACK 0
01A0 CD 13                              INT     13H                     ; DISKETTE_IO
01A2 59                         H2:     POP     CX                      ; RECOVER RETRY COUNT
01A3 73 09                              JNC     H4                      ; CARRY FLAG SET BY UNSUCCESSFUL READ
01A5 80 FC 80                           CMP     AH,80H                  ; IF TIME OUT, NO RETRY
01A8 74 22                              JZ      H5                      ; TRY FIXED DISK
01AA E2 E0                              LOOP    H1                      ; DO IT FOR RETRY TIMES
01AC EB 1E                              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 71                              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
022E                            BOOT_STRAP_1    ENDP
022E                            POST6           ENDP
022E                            CODE            ENDS
                                                END
```

```
PAGE 118,123
TITLE DISKETTE ----- 11/15/85 DISKETTE BIOS
.286C
.LIST
SUBTTL (DSK1.ASM)
.LIST
;-- INT 13 -------------------------------------------------------------------
; 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)=0  RESET DISKETTE SYSTEM
;               HARD RESET TO NEC, PREPARE COMMAND, RECALIBRATE REQUIRED
;               ON ALL DRIVES
;------------------------------------------------------------------------------
;       (AH)=1  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)
;               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)=2  READ THE DESIRED SECTORS INTO MEMORY
;------------------------------------------------------------------------------
;       (AH)=3  WRITE THE DESIRED SECTORS FROM MEMORY
;------------------------------------------------------------------------------
;       (AH)=4  VERIFY THE DESIRED SECTORS
;------------------------------------------------------------------------------
;       (AH)=5  FORMAT THE DESIRED TRACK
;               (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)=8  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.
;           @DSKETTE_STATUS = 0 AND CY IS RESET.
;------------------------------------------------------------------------------
;       (AH)=15 READ DASD TYPE
;       OUTPUT REGISTERS
```

SECTION 5

```
;       (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)=16 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)=17 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)=18 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
;--------------------------------------------------------------------------------
;       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
;               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.
;--------------------------------------------------------------------------------
;;;;;;XLIST
; DATA TRANSFER RATE SAVE AREA - 40:8B
;   -----------------------------------------------------------------
;   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           RESERVED            I       RESERVED    I
;           I                               I                   I
;  LAST DATA TRANSFER RATE        DATA TRANSFER RATE THAT    ADAPTER IS
;    SENT TO CONTROLLER            OPERATION STARTED IN    DSP OR COMBO
;                                                               (DUAL)
;       00: 500 KBS                     00: 500 KBS
;       01: 300 KBS                     01: 300 KBS
;       10: 250 KBS                     10: 250 KBS
;       11: RESERVED                    11: RESERVED
;
;--------------------------------------------------------------------------------
; DRIVE INDICATORS - 40:8F, DRIVE A - LOW NIBBLE, DRIVE B - HIGH NIBBLE
;   - BEFORE RETURNING, DRIVE INFORMATION IS COPIED FROM STATE MACHINE
;   LOCATION TO DRIVE INDICATORS PRIOR TO TRANSLATING STATE MACHINE TO
;   COMPATIBILITY MODE.
;
;                -------------------------------
;                I       I       I       I      I
;  DRIVE B       I   7   I   6   I   5   I   4  I
;                I       I       I       I      I
;                -------------------------------
;                I       I       I       I      I
;  DRIVE A       I   3   I   2   I   1   I   0  I
;                I       I       I       I      I
;                -------------------------------
;                    I       I       I       I
;                    I       I       I  80 TRACK CAPABILITY
;                    I       I       I  (INDEPENDENT OF
;                    I       I       I  DRIVE DETERMINED;
;                    I       I       I  ALWAYS VALID)
;                    I       I       I
;                    I       I       -> MULTI DATA RATE FORMAT
;                    I       I       CAPABILITY (VALID WHEN DRIVE
;                    I       I       DETERMINED)
;                    I       I
;                    I       -> MULTI DATA RATE CAPABILITY
;                    I       DETERMINED (DRIVE DETERMINED)
;                    I
```

```
;                  -> RESERVED
;
;-------------------------------------------------------------------------------

.LIST
; DISKETTE STATE MACHINE - ABSOLUTE ADDRESS 40:90 (DRIVE A) & 91 (DRIVE B)
;;;;;;;;.XLIST
;-------------------------------------------------------------------------------
; DURING EXECUTION OF ANY DISKETTE BIOS FUNCTION THE STATE MACHINE WILL
; CONTAIN THE FOLLOWING INFORMATION:
;
;              MEDIA                             DRIVE
;              -----                             -----
;--------------------------------- ----------------------------------
;I       I       I       I       I I        I       I       I       I
;I   7   I   6   I   5   I   4   I 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 80 TRACK CAPABILITY
;     I            I        I         I        I       I (INDEPENDENT OF
;     I            I        I         I        I       I DRIVE DETERMINED;
;     I            I        I         I        I       I ALWAYS VALID)
;     I            I        I         I        I       I
;     I            I        I         I        I      -> MULTI DATA RATE FORMAT
;     I            I        I         I        I      CAPABILITY (VALID WHEN DRIVE
;     I            I        I         I        I      DETERMINED)
;     I            I        I         I        I
;     I            I        I         I       -> MULTI DATA RATE CAPABILITY
;     I            I        I         I       DETERMINED (DRIVE DETERMINED)
;     I            I        I         I
;     I            I        I         -> RESERVED
;     I            I        I
;     I            I        I
;     I            I       -> MEDIA DETERMINED/ESTABLISHED
;     I            I
;     I            -> MEDIA DOUBLE STEPPING REQUIRED (VALID WHEN MEDIA DET.)
;     I
;     -> DATA TRANSFER RATE (VALID WHEN MEDIA DETERMINED)
;
;         00: 500 KBS
;         01: 300 KBS
;         10: 250 KBS
;         11: RESERVED
;
; FOR THE SAKE OF COMPATIBILITY WITH PREVIOUS PC SYSTEMS, UPON RETURNING TO
; THE CALLER OF THE DISKETTE BIOS FUNCTION, THE STATE MACHINE WILL CONTAIN
; THE FOLLOWING INFORMATION:

.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.
                          ;
                          ;               DI     : DRIVE #
                          ;               SI-HI  : HEAD #
                          ;               SI-LOW : # OF SECTORS OR DASD TYPE FOR FORMAT
```

SECTION 5

```
                          ;               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.
                                                                ; (AH) = @DSKETTE_STATUS
00A1 2E: FF 17                    CALL    WORD PTR CS:[BX]      ; 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 = 00; RESET
00B7 013C R                       DW      DISK_STATUS           ; AH = 01; STATUS
00B9 0148 R                       DW      DISK_READ             ; AH = 02; READ
00BB 0154 R                       DW      DISK_WRITE            ; AH = 03; WRITE
00BD 0160 R                       DW      DISK_VERF             ; AH = 04; VERIFY
00BF 016C R                       DW      DISK_FORMAT           ; AH = 05; FORMAT
00C1 01C8 R                       DW      FNC_ERR               ; AH = 06; INVALID
00C3 01C8 R                       DW      FNC_ERR               ; AH = 07; INVALID
00C5 01D2 R                       DW      DISK_PARMS            ; AH = 08; READ DRIVE PARAMETERS
00C7 01C8 R                       DW      FNC_ERR               ; AH = 09; INVALID
00C9 01C8 R                       DW      FNC_ERR               ; AH = 0A; INVALID
00CB 01C8 R                       DW      FNC_ERR               ; AH = 0B; INVALID
00CD 01C8 R                       DW      FNC_ERR               ; AH = 0C; INVALID
00CF 01C8 R                       DW      FNC_ERR               ; AH = 0D; INVALID
00D1 01C8 R                       DW      FNC_ERR               ; AH = 0E; INVALID
00D3 01C8 R                       DW      FNC_ERR               ; AH = 0F; INVALID
00D5 01C8 R                       DW      FNC_ERR               ; AH = 10; INVALID
00D7 01C8 R                       DW      FNC_ERR               ; AH = 11; INVALID
00D9 01C8 R                       DW      FNC_ERR               ; AH = 12; INVALID
00DB 01C8 R                       DW      FNC_ERR               ; AH = 13; INVALID
00DD 01C8 R                       DW      FNC_ERR               ; AH = 14; INVALID
00DF 028B R                       DW      DISK_TYPE             ; AH = 15; READ DASD TYPE
00E1 02AD R                       DW      DISK_CHANGE           ; AH = 16; CHANGE STATUS
00E3 02D8 R                       DW      FORMAT_SET            ; AH = 17; SET DASD TYPE
00E5 0339 R                       DW      SET_MEDIA             ; AH = 18; SET MEDIA TYPE
= 00E7                    FNC_TAE EQU     $                     ; END
00E7                      DISKETTE_IO_1   ENDP
                          ;-------------------------------------------------------------
                          ; DISK_RESET                                                  :
                          ;       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 0A5D 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 0994 R                    CALL    NEC_OUTPUT
0115 58                           POP     AX                    ; THROW AWAY ERROR RETURN
0116 E8 0A85 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 03CC R                    CALL    SEND_SPEC             ; SEND SPECIFY COMMAND TO NEC

012C                      RESBAC:
012C E8 07F5 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                                                 :
```

```
                                ;       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 07F5 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                                                   :
                                ;       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 04A2 R                          CALL    RD_WR_VF                ; COMMON READ/WRITE/VERIFY
0153 C3                                 RET
0154                            DISK_READ       ENDP
                                ;-------------------------------------------------------------
                                ; DISK_WRITE                                                  :
                                ;       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 04A2 R                          CALL    RD_WR_VF                ; COMMON READ/WRITE/VERIFY
015F C3                                 RET
0160                            DISK_WRITE      ENDP
                                ;-------------------------------------------------------------
                                ; DISK_VERF                                                   :
                                ;       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 04A2 R                          CALL    RD_WR_VF                ; COMMON READ/WRITE/VERIFY
016B C3                                 RET
016C                            DISK_VERF       ENDP
                                ;-------------------------------------------------------------
                                ; DISK_FORMAT                                                 :
                                ;       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 0403 R                          CALL    XLAT_NEW                ; TRANSLATE STATE TO PRESENT ARCH.
016F E8 058F R                          CALL    FMT_INIT                ; ESTABLISH STATE IF UNESTABLISHED
0172 80 0E 003F R 80                    OR      @MOTOR_STATUS,10000000B ; INDICATE WRITE OPERATION
0177 E8 05DD R                          CALL    MED_CHANGE              ; CHECK MEDIA CHANGE AND RESET IF SO
017A 72 41                              JC      FM_DON                  ; MEDIA CHANGED, SKIP
017C E8 03CC R                          CALL    SEND_SPEC               ; SEND SPECIFY COMMAND TO NEC
017F E8 0631 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 0618 R                          CALL    SEND_RATE               ; SEND DATA RATE TO CONTROLLER
0187                            FM_WR:
0187 B0 4A                              MOV     AL,04AH                 ; WILL WRITE TO THE DISKETTE
0189 E8 0641 R                          CALL    DMA_SETUP               ; SET UP THE DMA
018C 72 2F                              JC      FM_DON                  ; RETURN WITH ERROR
018E B4 4D                              MOV     AH,04DH                 ; ESTABLISH THE FORMAT COMMAND
0190 E8 06A1 R                          CALL    NEC_INIT                ; INITIALIZE THE NEC
0193 72 28                              JC      FM_DON                  ; ERROR - EXIT
0195 B8 01BD R                          MOV     AX,OFFSET FM_DON        ; LOAD ERROR ADDRESS
0198 50                                 PUSH    AX                      ; PUSH NEC_OUT ERROR RETURN
0199 B2 03                              MOV     DL,3                    ; BYTES/SECTOR VALUE TO NEC
019B E8 08A1 R                          CALL    GET_PARM
019E E8 0994 R                          CALL    NEC_OUTPUT
01A1 B2 04                              MOV     DL,4                    ; SECTORS/TRACK VALUE TO NEC
01A3 E8 08A1 R                          CALL    GET_PARM
01A6 E8 0994 R                          CALL    NEC_OUTPUT
01A9 B2 07                              MOV     DL,7                    ; GAP LENGTH VALUE TO NEC
01AB E8 08A1 R                          CALL    GET_PARM
01AE E8 0994 R                          CALL    NEC_OUTPUT
01B1 B2 08                              MOV     DL,8                    ; FILLER BYTE TO NEC
01B3 E8 08A1 R                          CALL    GET_PARM
```

```
01B6 E8 0994 R                 CALL    NEC_OUTPUT
01B9 58                        POP     AX                      ; THROW AWAY ERROR
01BA E8 06FC R                 CALL    NEC_TERM                ; TERMINATE, RECEIVE STATUS, ETC.
01BD                   FM_DON:
01BD E8 0429 R                 CALL    XLAT_OLD                ; TRANSLATE STATE TO COMPATIBLE MODE
01C0 E8 07F5 R                 CALL    SETUP_END               ; VARIOUS CLEANUPS
01C3 8B DE                     MOV     BX,SI                   ; GET SAVED AL TO BL
01C5 8A C3                     MOV     AL,BL                   ; PUT BACK FOR RETURN
01C7 C3                        RET
01C8                   DISK_FORMAT     ENDP

                       ;-------------------------------------------------------------
                       ; FNC_ERR                                                     :
                       ;       INVALID FUNCTION REQUESTED OR INVALID DRIVE;          :
                       ;       SET BAD COMMAND IN STATUS.                            :
                       ;                                                             :
                       ; ON EXIT:      @DISKETTE_STATUS, CY REFLECT STATUS OF OPERATION :
                       ;-------------------------------------------------------------
01C8                   FNC_ERR PROC    NEAR                    ; INVALID FUNCTION REQUEST
01C8 8B C6                     MOV     AX,SI                   ; RESTORE AL
01CA B4 01                     MOV     AH,BAD_CMD              ; SET BAD COMMAND ERROR
01CC 88 26 0041 R              MOV     @DISKETTE_STATUS,AH     ; STORE IN DATA AREA
01D0 F9                        STC                             ; SET CARRY INDICATING ERROR
01D1 C3                        RET
01D2                   FNC_ERR ENDP
                       ;-------------------------------------------------------------
                       ; DISK_PARMS                                                  :
                       ;       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.                                              :
                       ;-------------------------------------------------------------
01D2                   DISK_PARMS      PROC    NEAR
01D2 E8 0403 R                 CALL    XLAT_NEW                ; TRANSLATE STATE TO PRESENT ARCH.
01D5 C7 46 02 0000             MOV     WORD PTR [BP+2],0       ; DRIVE TYPE = 0
01DA A1 0010 R                 MOV     AX,@EQUIP_FLAG          ; LOAD EQUIPMENT FLAG FOR # DISKETTES
01DD 24 C1                     AND     AL,11000001B            ; KEEP DISKETTE DRIVE BITS
01DF B2 02                     MOV     DL,2                    ; DISKETTE DRIVES = 2
01E1 3C 41                     CMP     AL,01000001B            ; 2 DRIVES INSTALLED ?
01E3 74 06                     JZ      STO_DL                  ; IF YES JUMP

01E5 FE CA                     DEC     DL                      ; DISKETTE DRIVES = 1
01E7 3C 01                     CMP     AL,00000001B            ; 1 DRIVE INSTALLED ?
01E9 75 6A                     JNZ     NON_DRV                 ; IF NO JUMP

01EB                   STO_DL:
01EB 88 56 04                  MOV     [BP+4],DL               ; STORE NUMBER OF DRIVES
01EE 83 FF 01                  CMP     DI,1                    ; CHECK FOR VALID DRIVE
01F1 77 66                     JA      NON_DRV1                ; DRIVE INVALID
01F3 C6 46 05 01               MOV     BYTE PTR[BP+5],1        ; MAXIMUM HEAD NUMBER = 1
01F7 E8 0888 R                 CALL    CMOS_TYPE               ; RETURN DRIVE TYPE IN AL
01FA 72 16                     JC      CHK_EST                 ; ON CMOS BAD CHECK ESTABLISHED
01FC 0A C0                     OR      AL,AL                   ; TEST FOR NO DRIVE TYPE
01FE 74 12                     JZ      CHK_EST                 ; JUMP IF SO
0200 E8 03AC R                 CALL    DR_TYPE_CHECK           ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
0203 72 0D                     JC      CHK_EST                 ; TYPE NOT IN TABLE (POSSIBLE BAD CMOS)
0205 88 46 02                  MOV     [BP+2],AL               ; STORE VALID CMOS DRIVE TYPE
0208 2E: 8A 4F 04              MOV     CL,CS:[BX].MD_SEC_TRK   ; GET SECTOR/TRACK
020C 2E: 8A 6F 0B              MOV     CH,CS:[BX].MD_MAX_TRK   ; GET MAX. TRACK NUMBER
0210 EB 32                     JMP     SHORT STO_CX            ; CMOS GOOD, USE CMOS

0212                   CHK_EST:
0212 8A A5 0090 R              MOV     AH,@DSK_STATE[DI]       ; LOAD STATE FOR THIS DRIVE
0216 F6 C4 10                  TEST    AH,MED_DET              ; CHECK FOR ESTABLISHED STATE
0219 74 3E                     JZ      NON_DRV1                ; CMOS BAD/INVALID AND UNESTABLISHED

021B                   USE_EST:
021B 80 E4 C0                  AND     AH,RATE_MSK             ; ISOLATE STATE
021E 80 FC 80                  CMP     AH,RATE_250             ; RATE 250 ?
0221 75 54                     JNE     USE_EST2                ; NO, GO CHECK OTHER RATE

                       ;--- DATA RATE IS 250 KBS, TRY 360 KB TABLE FIRST

0223 B0 01                     MOV     AL,01                   ; DRIVE TYPE 1 (360KB)
0225 E8 03AC R                 CALL    DR_TYPE_CHECK           ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
0228 2E: 8A 4F 04              MOV     CL,CS:[BX].MD_SEC_TRK   ; GET SECTOR/TRACK
022C 2E: 8A 6F 0B              MOV     CH,CS:[BX].MD_MAX_TRK   ; GET MAX. TRACK NUMBER
0230 F6 85 0090 R 01           TEST    @DSK_STATE[DI],TRK_CAPA ; 80 TRACK ?
0235 74 0D                     JZ      STO_CX                  ; MUST BE 360KB DRIVE

                       ;--- IT IS 1.44 MB DRIVE

0237                   PARM144:
0237 B0 04                     MOV     AL,04                   ; DRIVE TYPE 4 (1.44MB)
0239 E8 03AC R                 CALL    DR_TYPE_CHECK           ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
023C 2E: 8A 4F 04              MOV     CL,CS:[BX].MD_SEC_TRK   ; GET SECTOR/TRACK
0240 2E: 8A 6F 0B              MOV     CH,CS:[BX].MD_MAX_TRK   ; GET MAX. TRACK NUMBER

0244                   STO_CX:
0244 89 4E 00                  MOV     [BP],CX                 ; SAVE IN STACK FOR RETURN
0247                   ES_DI:
0247 89 5E 06                  MOV     [BP+6],BX               ; ADDRESS OF MEDIA/DRIVE PARM TABLE
024A 8C C8                     MOV     AX,CS                   ; SEGMENT MEDIA/DRIVE PARAMETER TABLE
024C 8E C0                     MOV     ES,AX                   ; ES IS SEGMENT OF TABLE

024E                   DP_OUT:
024E E8 0429 R                 CALL    XLAT_OLD                ; TRANSLATE STATE TO COMPATIBLE MODE
0251 33 C0                     XOR     AX,AX                   ; CLEAR
0253 F8                        CLC
```

```
0254 C3                         RET

                        ;-----  NO DRIVE PRESENT HANDLER

0255                    NON_DRV:
0255 C6 46 04 00                MOV     BYTE PTR [BP+4],0       ; CLEAR NUMBER OF DRIVES

0259                    NON_DRV1:
0259 81 FF 0080                 CMP     DI,80H                  ; CHECK FOR FIXED MEDIA TYPE REQUEST
025D 72 09                      JB      NON_DRV2                ; CONTINUE IF NOT REQUEST FALL THROUGH

                        ;-----  FIXED DISK REQUEST FALL THROUGH ERROR

025F E8 0429 R                  CALL    XLAT_OLD                ; ELSE TRANSLATE TO COMPATIBLE MODE
0262 8B C6                      MOV     AX,SI                   ; RESTORE AL
0264 B4 01                      MOV     AH,BAD_CMD              ; SET BAD COMMAND ERROR
0266 F9                         STC                             ; SET ERROR RETURN CODE
0267 C3                         RET

0268                    NON_DRV2:
0268 33 C0                      XOR     AX,AX                   ; CLEAR PARMS IF NO DRIVES OR CMOS BAD
026A 89 46 00                   MOV     [BP],AX                 ; TRACKS, SECTORS/TRACK = 0
026D 88 66 05                   MOV     [BP+5],AH               ; HEAD = 0
0270 89 46 06                   MOV     [BP+6],AX               ; OFFSET TO DISK_BASE = 0
0273 8E C0                      MOV     ES,AX                   ; ES IS SEGMENT OF TABLE
0275 EB D7                      JMP     SHORT DP_OUT

                        ;--- DATA RATE IS EITHER 300 KBS OR 500 KBS, TRY 1.2 MB TABLE FIRST

0277                    USE_EST2:
0277 B0 02                      MOV     AL,02                   ; DRIVE TYPE 2 (1.2MB)
0279 E8 03AC R                  CALL    DR_TYPE_CHECK           ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
027C 2E: 8A 4F 04               MOV     CL,CS:[BX].MD_SEC_TRK   ; GET SECTOR/TRACK
0280 2E: 8A 6F 0B               MOV     CH,CS:[BX].MD_MAX_TRK   ; GET MAX. TRACK NUMBER
0284 80 FC 40                   CMP     AH,RATE_300             ; RATE 300 ?
0287 74 BB                      JE      STO_CX                  ; MUST BE 1.2MB DRIVE
0289 EB AC                      JMP     SHORT PARM144           ; ELSE, IT IS 1.44MB DRIVE

028B                    DISK_PARMS      ENDP

                        ;----------------------------------------------------------------
                        ; DISK_TYPE                                                      :
                        ;       THIS ROUTINE RETURNS THE TYPE OF MEDIA INSTALLED.        :
                        ;  ON ENTRY:    DI = DRIVE #                                     :
                        ;                                                                :
                        ;  ON EXIT:     AH = DRIVE TYPE, CY=0                            :
                        ;----------------------------------------------------------------
028B                    DISK_TYPE       PROC    NEAR
028B E8 0403 R                  CALL    XLAT_NEW                ; TRANSLATE STATE TO PRESENT ARCH.
028E 8A 85 0090 R               MOV     AL,@DSK_STATE[DI]       ; GET PRESENT STATE INFORMATION
0292 0A C0                      OR      AL,AL                   ; CHECK FOR NO DRIVE
0294 74 13                      JZ      NO_DRV                  ;
0296 B4 01                      MOV     AH,NOCHGLN              ; NO CHANGE LINE FOR 40 TRACK DRIVE
0298 A8 01                      TEST    AL,TRK_CAPA             ; IS THIS DRIVE AN 80 TRACK DRIVE?
029A 74 02                      JZ      DT_BACK                 ; IF NO JUMP
029C B4 02                      MOV     AH,CHGLN                ; CHANGE LINE FOR 80 TRACK DRIVE

029E                    DT_BACK:
029E 50                         PUSH    AX                      ; SAVE RETURN VALUE
029F E8 0429 R                  CALL    XLAT_OLD                ; TRANSLATE STATE TO COMPATIBLE MODE
02A2 58                         POP     AX                      ; RESTORE RETURN VALUE
02A3 F8                         CLC                             ; NO ERROR
02A4 8B DE                      MOV     BX,SI                   ; GET SAVED AL TO BL
02A6 8A C3                      MOV     AL,BL                   ; PUT BACK FOR RETURN
02A8 C3                         RET
02A9                    NO_DRV:
02A9 32 E4                      XOR     AH,AH                   ; NO DRIVE PRESENT OR UNKNOWN
02AB EB F1                      JMP     SHORT DT_BACK
02AD                    DISK_TYPE ENDP
                        ;----------------------------------------------------------------
                        ; DISK_CHANGE                                                    :
                        ;       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        :
                        ;----------------------------------------------------------------
02AD                    DISK_CHANGE     PROC    NEAR
02AD E8 0403 R                  CALL    XLAT_NEW                ; TRANSLATE STATE TO PRESENT ARCH.
02B0 8A 85 0090 R               MOV     AL,@DSK_STATE[DI]       ; GET MEDIA STATE INFORMATION
02B4 0A C0                      OR      AL,AL                   ; DRIVE PRESENT ?
02B6 74 19                      JZ      DC_NON                  ; JUMP IF NO DRIVE
02B8 A8 01                      TEST    AL,TRK_CAPA             ; 80 TRACK DRIVE ?
02BA 74 05                      JZ      SETIT                   ; IF SO , CHECK CHANGE LINE

02BC E8 0AC4 R          DC0:    CALL    READ_DSKCHNG            ; GO CHECK STATE OF DISK CHANGE LINE
02BF 74 05                      JZ      FINIS                   ; CHANGE LINE NOT ACTIVE

02C1 C6 06 0041 R 06    SETIT:  MOV     @DSKETTE_STATUS,MEDIA_CHANGE    ; INDICATE MEDIA REMOVED

02C6 E8 0429 R          FINIS:  CALL    XLAT_OLD                ; TRANSLATE STATE TO COMPATIBLE MODE
02C9 E8 07F5 R                  CALL    SETUP_END               ; VARIOUS CLEANUPS
02CC 8B DE                      MOV     BX,SI                   ; GET SAVED AL TO BL
02CE 8A C3                      MOV     AL,BL                   ; PUT BACK FOR RETURN
02D0 C3                         RET

02D1                    DC_NON:
02D1 80 0E 0041 R 80            OR      @DSKETTE_STATUS,TIME_OUT        ; SET TIMEOUT, NO DRIVE
02D6 EB EE                      JMP     SHORT FINIS
02D8                    DISK_CHANGE     ENDP

                        ;----------------------------------------------------------------
                        ; FORMAT_SET                                                     :
                        ;       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                                  :
```

SECTION 5

```
                                ;--------------------------------------------------------------
02D8                            FORMAT_SET      PROC    NEAR
02D8  E8 0403 R                         CALL    XLAT_NEW                ; TRANSLATE STATE TO PRESENT ARCH.
02DB  56                                PUSH    SI                      ; SAVE DASD TYPE
02DC  8B C6                             MOV     AX,SI                   ; AH = ? , AL = DASD TYPE
02DE  32 E4                             XOR     AH,AH                   ; AH = 0 , AL = DASD TYPE
02E0  8B F0                             MOV     SI,AX                   ; SI = DASD TYPE
02E2  80 A5 0090 R 0F                   AND     @DSK_STATE[DI],NOT MED_DET+DBL_STEP+RATE_MSK    ; CLEAR STATE
02E7  4E                                DEC     SI                      ; CHECK FOR 320/360K MEDIA & DRIVE
02E8  75 07                             JNZ     NOT_320                 ; BYPASS IF NOT
02EA  80 8D 0090 R 90                   OR      @DSK_STATE[DI],MED_DET+RATE_250 ; SET TO 320/360
02EF  EB 37                             JMP     SHORT S0

02F1                            NOT_320:
02F1  E8 05DD R                         CALL    MED_CHANGE              ; CHECK FOR TIME_OUT
02F4  80 3E 0041 R 80                   CMP     @DSKETTE_STATUS,TIME_OUT
02F9  74 2D                             JZ      S0                      ; IF TIME OUT TELL CALLER

02FB  4E                        S3:     DEC     SI                      ; CHECK FOR 320/360K IN 1.2M DRIVE
02FC  75 07                             JNZ     NOT_320_12              ; BYPASS IF NOT
02FE  80 8D 0090 R 70                   OR      @DSK_STATE[DI],MED_DET+DBL_STEP+RATE_300 ; SET STATE
0303  EB 23                             JMP     SHORT S0

0305                            NOT_320_12:
0305  4E                                DEC     SI                      ; CHECK FOR 1.2M MEDIA IN 1.2M DRIVE
0306  75 07                             JNZ     NOT_12                  ; BYPASS IF NOT
0308  80 8D 0090 R 10                   OR      @DSK_STATE[DI],MED_DET+RATE_500  ; SET STATE VARIABLE
030D  EB 19                             JMP     SHORT S0                ; RETURN TO CALLER

030F                            NOT_12:
030F  4E                                DEC     SI                      ; CHECK FOR SET DASD TYPE 04
0310  75 20                             JNZ     FS_ERR                  ; BAD COMMAND EXIT IF NOT VALID TYPE

0312  F6 85 0090 R 04                   TEST    @DSK_STATE[DI],DRV_DET  ; DRIVE DETERMINED ?
0317  74 09                             JZ      ASSUME                  ; IF STILL NOT DETERMINED ASSUME
0319  B0 50                             MOV     AL,MED_DET+RATE_300
031B  F6 85 0090 R 02                   TEST    @DSK_STATE[DI],FMT_CAPA ; MULTIPLE FORMAT CAPABILITY ?
0320  75 02                             JNZ     OR_IT_IN                ; IF 1.2 M THEN DATA RATE 300

0322                            ASSUME:
0322  B0 90                             MOV     AL,MED_DET+RATE_250     ; SET UP

0324                            OR_IT_IN:
0324  08 85 0090 R                      OR      @DSK_STATE[DI],AL       ; OR IN THE CORRECT STATE

0328                            S0:
0328  E8 0429 R                         CALL    XLAT_OLD                ; TRANSLATE STATE TO COMPATIBLE MODE
032B  E8 07F5 R                         CALL    SETUP_END               ; VARIOUS CLEANUPS
032E  5B                                POP     BX                      ; GET SAVED AL TO BL
032F  8A C3                             MOV     AL,BL                   ; PUT BACK FOR RETURN
0331  C3                                RET

0332                            FS_ERR:
0332  C6 06 0041 R 01                   MOV     @DSKETTE_STATUS,BAD_CMD ; UNKNOWN STATE,BAD COMMAND
0337  EB EF                             JMP     SHORT S0

0339                            FORMAT_SET      ENDP

                                ;--------------------------------------------------------------
                                ; SET_MEDIA                                                    :
                                ;       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                                             :
                                ;--------------------------------------------------------------
0339                            SET_MEDIA       PROC    NEAR
0339  E8 0403 R                         CALL    XLAT_NEW                ; TRANSLATE STATE TO PRESENT ARCH.
033C  E8 0888 R                         CALL    CMOS_TYPE               ; RETURN DRIVE TYPE IN (AL)
033F  72 3F                             JC      SM_ASSUME               ; ERROR IN CMOS
0341  0A C0                             OR      AL,AL                   ; TEST FOR NO DRIVE
0343  74 60                             JZ      SM_RTN                  ; RETURN IF SO
0345  E8 03AC R                         CALL    DR_TYPE_CHECK           ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
0348  72 36                             JC      SM_ASSUME               ; TYPE NOT IN TABLE (BAD CMOS)
034A  57                                PUSH    DI                      ; SAVE REG.
034B  33 DB                             XOR     BX,BX                   ; BX = INDEX TO DR_TYPE TABLE
034D  B9 0006                           MOV     CX,DR_CNT               ; CX = LOOP COUNT
0350                            DR_SEARCH:
0350  2E: 8A A7 0000 R                  MOV     AH,CS:DR_TYPE[BX]       ; GET DRIVE TYPE
0355  80 E4 7F                          AND     AH,BIT7OFF              ; MASK OUT MSB
0358  3A C4                             CMP     AL,AH                   ; DRIVE TYPE MATCH ?
035A  75 17                             JNE     NXT_MD                  ; NO, CHECK NEXT DRIVE TYPE
035C                            DR_FND:
035C  2E: 8B BF 0001 R                  MOV     DI,CS:WORD PTR DR_TYPE[BX+1]    ; DI = MEDIA/DRIVE PARAMETER TABLE
0361                            MD_SEARCH:
0361  2E: 8A 65 04                      MOV     AH,CS:[DI].MD_SEC_TRK   ; GET SECTOR/TRACK
0365  38 66 00                          CMP     [BP],AH                 ; MATCH ?
0368  75 09                             JNE     NXT_MD                  ; NO, CHECK NEXT MEDIA
036A  2E: 8A 65 0B                      MOV     AH,CS:[DI].MD_MAX_TRK   ; GET MAX. TRACK #
036E  38 66 01                          CMP     [BP+1],AH               ; MATCH ?
0371  74 15                             JE      MD_FND                  ; YES, GO GET RATE
0373                            NXT_MD:
0373  83 C3 03                          ADD     BX,3                    ; CHECK NEXT DRIVE TYPE
0376  E2 D8                             LOOP    DR_SEARCH
0378  C6 06 0041 R 0C                   MOV     @DSKETTE_STATUS,MED_NOT_FND     ; ERROR, MEDIA TYPE NOT FOUND
037D  5F                                POP     DI                      ; RESTORE REG.
037E  EB 25                             JMP     SHORT SM_RTN            ; RETURN

0380                            SM_ASSUME:
0380  B9 0006                           MOV     CX,DR_CNT               ; RESET LOOP COUNT
0383  33 DB                             XOR     BX,BX                   ; START AT TOP OF TABLE
0385  57                                PUSH    DI                      ; SAVE REG.
0386  EB D4                             JMP     SHORT DR_FND
```

```

0388                     MD_FND:
0388 2E: 8A 45 0C                MOV     AL,CS:[DI].MD_RATE      ; GET RATE
038C 3C 40                       CMP     AL,RATE_300             ; DOUBLE STEP REQUIRED FOR RATE 300
038E 75 02                       JNE     MD_SET
0390 0C 20                       OR      AL,DBL_STEP
0392                     MD_SET:
0392 89 7E 06                    MOV     [BP+6],DI               ; SAVE TABLE POINTER IN STACK
0395 0C 10                       OR      AL,MED_DET              ; SET MESIA ESTABLISHED
0397 5F                          POP     DI                      ; RESTORE REG.
0398 80 A5 0090 R 0F             AND     @DSK_STATE[DI],NOT MED_DET+DBL_STEP+RATE_MSK     ; CLEAR STATE
039D 08 85 0090 R                OR      @DSK_STATE[DI],AL       ; SET STATE
03A1 8C C8                       MOV     AX,CS                   ; SEGMENT MEDIA/DRIVE PARAMETER TABLE
03A3 8E C0                       MOV     ES,AX                   ; ES IS SEGMENT OF TABLE
03A5                     SM_RTN:
03A5 E8 0429 R                   CALL    XLAT_OLD                ; TRANSLATE STATE TO COMPATIBLE MODE
03A8 E8 07F5 R                   CALL    SETUP_END               ; VARIOUS CLEANUPS
03AB C3                          RET
03AC                     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                                         :
                         ;---------------------------------------------------------------
03AC                     DR_TYPE_CHECK   PROC    NEAR
03AC 50                          PUSH    AX
03AD 51                          PUSH    CX
03AE 33 DB                       XOR     BX,BX                   ; BX = INDEX TO DR_TYPE TABLE
03B0 B9 0006                     MOV     CX,DR_CNT               ; CX = LOOP COUNT
03B3                     TYPE_CHK:
03B3 2E: 8A A7 0000 R            MOV     AH,CS:DR_TYPE[BX]       ; GET DRIVE TYPE
03B8 3A C4                       CMP     AL,AH                   ; DRIVE TYPE MATCH ?
03BA 74 08                       JE      DR_TYPE_VALID           ; YES, RETURN WITH CARRY RESET
03BC 83 C3 03                    ADD     BX,3                    ; CHECK NEXT DRIVE TYPE
03BF E2 F2                       LOOP    TYPE_CHK
03C1 F9                          STC                             ; DRIVE TYPE NOT FOUND IN TABLE
03C2 EB 05                       JMP     SHORT TYPE_RTN
03C4                     DR_TYPE_VALID:
03C4 2E: 8B 9F 0001 R            MOV     BX,CS:WORD PTR DR_TYPE[BX+1]     ; BX = MEDIA TABLE
03C9                     TYPE_RTN:
03C9 59                          POP     CX
03CA 58                          POP     AX
03CB C3                          RET
03CC                     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                                     :
                         ;---------------------------------------------------------------
03CC                     SEND_SPEC       PROC    NEAR
03CC 50                          PUSH    AX                      ; SAVE AX
03CD B8 03E7 R                   MOV     AX,OFFSET SPECBAC       ; LOAD ERROR ADDRESS
03D0 50                          PUSH    AX                      ; PUSH NEC_OUT ERROR RETURN
03D1 B4 03                       MOV     AH,03H                  ; SPECIFY COMMAND
03D3 E8 0994 R                   CALL    NEC_OUTPUT              ; OUTPUT THE COMMAND
03D6 2A D2                       SUB     DL,DL                   ; FIRST SPECIFY BYTE
03D8 E8 08A1 R                   CALL    GET_PARM                ; GET PARAMETER TO AH
03DB E8 0994 R                   CALL    NEC_OUTPUT              ; OUTPUT THE COMMAND
03DE B2 01                       MOV     DL,1                    ; SECOND SPECIFY BYTE
03E0 E8 08A1 R                   CALL    GET_PARM                ; GET PARAMETER TO AH
03E3 E8 0994 R                   CALL    NEC_OUTPUT              ; OUTPUT THE COMMAND
03E6 58                          POP     AX                      ; POP ERROR RETURN
03E7                     SPECBAC:
03E7 58                          POP     AX                      ; RESTORE ORIGINAL AX VALUE
03E8 C3                          RET
03E9                     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                                         :
                         ;---------------------------------------------------------------
03E9                     SEND_SPEC_MD    PROC    NEAR
03E9 50                          PUSH    AX                      ; SAVE RATE DATA
03EA B8 0401 R                   MOV     AX,OFFSET SPEC_ESBAC    ; LOAD ERROR ADDRESS
03ED 50                          PUSH    AX                      ; PUSH NEC_OUT ERROR RETURN
03EE B4 03                       MOV     AH,03H                  ; SPECIFY COMMAND
03F0 E8 0994 R                   CALL    NEC_OUTPUT              ; OUTPUT THE COMMAND
03F3 2E: 8A 27                   MOV     AH,CS:[BX].MD_SPEC1     ; GET 1ST SPECIFY BYTE
03F6 E8 0994 R                   CALL    NEC_OUTPUT              ; OUTPUT THE COMMAND
03F9 2E: 8A 67 01                MOV     AH,CS:[BX].MD_SPEC2     ; GET SECOND SPECIFY BYTE
03FD E8 0994 R                   CALL    NEC_OUTPUT              ; OUTPUT THE COMMAND
0400 58                          POP     AX                      ; POP ERROR RETURN
0401                     SPEC_ESBAC:
0401 58                          POP     AX                      ; RESTORE RATE
0402 C3                          RET
0403                     SEND_SPEC_MD    ENDP

                         ;---------------------------------------------------------------
                         ; XKAT_NEW                                                      :
                         ;         TRANSLATES DISKETTE STATE LOCATIONS FROM COMPATIBLE   :
                         ;         MODE TO NEW ARCHITECTURE.                             :
                         ;                                                               :
                         ; ON ENTRY:       DI : DRIVE                                    :
                         ;---------------------------------------------------------------
0403                     XLAT_NEW        PROC    NEAR            ;
0403 83 FF 01                    CMP     DI,1                    ; VALID DRIVE ?
0406 77 1C                       JA      XN_OUT                  ; IF INVALID BACK
0408 80 BD 0090 R 00             CMP     @DSK_STATE[DI],0        ; NO DRIVE ?
```

SECTION 5

```
040D 74 16                          JZ      DO_DET                  ; IF NO DRIVE ATTEMPT DETERMINE
040F 8B CF                          MOV     CX,DI                   ; CX = DRIVE NUMBER
0411 C0 E1 02                       SHL     CL,2                    ; CL = SHIFT COUNT, A=0, B=4
0414 A0 008F R                      MOV     AL,@HF_CNTRL            ; DRIVE INFORMATION
0417 D2 C8                          ROR     AL,CL                   ; TO LOW NIBBLE
0419 24 07                          AND     AL,DRV_DET+FMT_CAPA+TRK_CAPA    ; KEEP DRIVE BITS
041B 80 A5 0090 R F8                AND     @DSK_STATE[DI],NOT DRV_DET+FMT_CAPA+TRK_CAPA
0420 08 85 0090 R                   OR      @DSK_STATE[DI],AL       ; UPDATE DRIVE STATE
0424                        XN_OUT:
0424 C3                             RET                             ;

0425                        DO_DET:
0425 E8 0ACE R                      CALL    DRIVE_DET               ; TRY TO DETERMINE
0428 C3                             RET

0429                        XLAT_NEW        ENDP
                            ;----------------------------------------------------------------
                            ; XLAT_OLD                                                      :
                            ;       TRANSLATES DISKETTE STATE LOCATIONS FROM NEW            :
                            ;       ARCHITECTURE TO COMPATIBLE MODE.                        :
                            ;                                                               :
                            ; ON ENTRY:     DI : DRIVE                                      :
                            ;----------------------------------------------------------------
0429                        XLAT_OLD        PROC    NEAR
0429 83 FF 01                       CMP     DI,1                    ; VALID DRIVE ?
042C 77 73                          JA      XO_OUT                  ; IF INVALID BACK
042E 80 BD 0090 R 00                CMP     @DSK_STATE[DI],0        ; NO DRIVE ?
0433 74 6C                          JZ      XO_OUT                  ; IF NO DRIVE TRANSLATE DONE

                            ;-----  TEST FOR SAVED DRIVE INFORMATION ALREADY SET

0435 8B CF                          MOV     CX,DI                   ; CX = DRIVE NUMBER
0437 C0 E1 02                       SHL     CL,2                    ; CL = SHIFT COUNT, A=0, B=4
043A B4 02                          MOV     AH,FMT_CAPA             ; LOAD MULTI DATA RATE BIT MASK
043C D2 CC                          ROR     AH,CL                   ; ROTATE BY MASK
043E 84 26 008F R                   TEST    @HF_CNTRL,AH            ; MULTI-DATA RATE DETERMINED ?
0442 75 16                          JNZ     SAVE_SET                ; IF SO, NO NEED TO RE-SAVE

                            ;-----  ERASE DRIVE BITS IN @HF_CNTRL FOR THIS DRIVE

0444 B4 07                          MOV     AH,DRV_DET+FMT_CAPA+TRK_CAPA    ; MASK TO KEEP
0446 D2 CC                          ROR     AH,CL                   ; FIX MASK TO KEEP
0448 F6 D4                          NOT     AH                      ; TRANSLATE MASK
044A 20 26 008F R                   AND     @HF_CNTRL,AH            ; KEEP BITS FROM OTHER DRIVE INTACT

                            ;-----  ACCESS CURRENT DRIVE BITS AND STORE IN @HF_CNTRL

044E 8A 85 0090 R                   MOV     AL,@DSK_STATE[DI]       ; ACCESS STATE
0452 24 07                          AND     AL,DRV_DET+FMT_CAPA+TRK_CAPA    ; KEEP DRIVE BITS
0454 D2 C8                          ROR     AL,CL                   ; FIX FOR THIS DRIVE
0456 08 06 008F R                   OR      @HF_CNTRL,AL            ; UPDATE SAVED DRIVE STATE

                            ;-----  TRANSLATE TO COMPATIBILITY MODE

045A                        SAVE_SET:
045A 8A A5 0090 R                   MOV     AH,@DSK_STATE[DI]       ; ACCESS STATE
045E 8A FC                          MOV     BH,AH                   ; TO BH FOR LATER
0460 80 E4 C0                       AND     AH,RATE_MSK             ; KEEP ONLY RATE
0463 80 FC 00                       CMP     AH,RATE_500             ; RATE 500 ?
0466 74 10                          JZ      CHK_144                 ; YES, 1.2/1.2 OR 1.44/1.44
0468 B0 01                          MOV     AL,M3D1U                ; AL = 360 IN 1.2 UNESTABLISHED
046A 80 FC 40                       CMP     AH,RATE_300             ; RATE 300 ?
046D 75 16                          JNZ     CHK_250                 ; NO, 360/360 ,720/720 OR 720/1.44
046F F6 C7 20                       TEST    BH,DBL_STEP             ; YES, DOUBLE STEP ?
0472 75 1D                          JNZ     TST_DET                 ; YES, MUST BE 360 IN 1.2

0474                        UNKNO:
0474 B0 07                          MOV     AL,MED_UNK              ; 'NONE OF THE ABOVE'
0476 EB 20                          JMP     SHORT AL_SET            ; PROCESS COMPLETE

0478                        CHK_144:
0478 E8 0888 R                      CALL    CMOS_TYPE               ; RETURN DRIVE TYPE IN (AL)
047B 72 F7                          JC      UNKNO                   ; ERROR, SET 'NONE OF THE ABOVE'
047D 3C 02                          CMP     AL,02                   ; 1.2MB DRIVE ?
047F 75 F3                          JNE     UNKNO                   ; NO, GO SET 'NONE OF THE ABOVE'
0481 B0 02                          MOV     AL,M1D1U                ; AL = 1.2 IN 1.2 UNESTABLISHED
0483 EB 0C                          JMP     SHORT TST_DET

0485                        CHK_250:
0485 B0 00                          MOV     AL,M3D3U                ; AL = 360 IN 360 UNESTABLISHED
0487 80 FC 80                       CMP     AH,RATE_250             ; RATE 250 ?
048A 75 E8                          JNZ     UNKNO                   ; IF SO FALL THRU
048C F6 C7 01                       TEST    BH,TRK_CAPA             ; 80 TRACK CAPABILITY ?
048F 75 E3                          JNZ     UNKNO                   ; IF SO JUMP, FALL THRU TEST DET

0491                        TST_DET:
0491 F6 C7 10                       TEST    BH,MED_DET              ; DETERMINED ?
0494 74 02                          JZ      AL_SET                  ; IF NOT THEN SET
0496 04 03                          ADD     AL,3                    ; MAKE DETERMINED/ESTABLISHED

0498                        AL_SET:
0498 80 A5 0090 R F8                AND     @DSK_STATE[DI],NOT DRV_DET+FMT_CAPA+TRK_CAPA  ; CLEAR DRIVE
049D 08 85 0090 R                   OR      @DSK_STATE[DI],AL       ; REPLACE WITH COMPATIBLE MODE
04A1                        XO_OUT:
04A1 C3                             RET
04A2                        XLAT_OLD        ENDP


                            ;----------------------------------------------------------------
                            ; 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 :
                            ;----------------------------------------------------------------
04A2                        RD_WR_VF        PROC    NEAR
04A2 50                             PUSH    AX                      ; SAVE DMA, NEC PARAMETERS
04A3 E8 0403 R                      CALL    XLAT_NEW                ; TRANSLATE STATE TO PRESENT ARCH.
04A6 E8 055E R                      CALL    SETUP_STATE             ; INITIALIZE START AND END RATE
04A9 58                             POP     AX                      ; RESTORE READ/WRITE/VERIFY

04AA                        DO_AGAIN:
```

```
04AA 50                         PUSH    AX                      ; SAVE READ/WRITE/VERIFY PARAMETER
04AB E8 05DD R                  CALL    MED_CHANGE              ; MEDIA CHANGE AND RESET IF CHANGED
04AE 58                         POP     AX                      ; RESTORE READ/WRITE/VERIFY
                        ;       JC      RWV_END                 ; MEDIA CHANGE ERROR OR TIME-OUT
04AF 73 03                      JNC     RWV
04B1 E9 054F R                  JMP     RWV_END
04B4                    RWV:
04B4 50                         PUSH    AX                      ; SAVE READ/WRITE/VERIFY PARAMETER

04B5 8A B5 0090 R               MOV     DH,@DSK_STATE[DI]       ; GET RATE STATE OF THIS DRIVE
04B9 80 E6 C0                   AND     DH,RATE_MSK             ; KEEP ONLY RATE
04BC E8 0888 R                  CALL    CMOS_TYPE               ; RETURN DRIVE TYPE IN (AL)
04BF 72 46                      JC      RWV_ASSUME              ; ERROR IN CMOS
04C1 3C 01                      CMP     AL,1                    ; 40 TRACK DRIVE?
04C3 75 0B                      JNE     RWV_1                   ; NO, BYPASS CMOS VALIDITY CHECK
04C5 F6 85 0090 R 01            TEST    @DSK_STATE[DI],TRK_CAPA ; CHECK FOR 40 TRACK DRIVE
04CA 74 0F                      JZ      RWV_2                   ; YES, CMOS IS CORRECT
04CC B0 02                      MOV     AL,2                    ; CHANGE TO 1.2 M
04CE EB 0B                      JMP     SHORT RWV_2             ; CONTINUE
04D0                    RWV_1:
04D0 72 09                      JB      RWV_2                   ; NO DRIVE SPECIFIED, CONTINUE
04D2 F6 85 0090 R 01            TEST    @DSK_STATE[DI],TRK_CAPA ; IS IT REALLY 40 TRACK?
04D7 75 02                      JNZ     RWV_2                   ; NO, 80 TRACK
04D9 B0 01                      MOV     AL,1                    ; IT'S 40 TRACK, FIX CMOS VALUE
04DB                    RWV_2:

04DB 0A C0                      OR      AL,AL                   ; TEST FOR NO DRIVE
04DD 74 28                      JZ      RWV_ASSUME              ; ASSUME TYPE, USE MAX TRACK
04DF E8 03AC R                  CALL    DR_TYPE_CHECK           ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
04E2 72 23                      JC      RWV_ASSUME              ; TYPE NOT IN TABLE (BAD CMOS)

                        ;--- SEARCH FOR MEDIA/DRIVE PARAMETER TABLE

04E4 57                         PUSH    DI                      ; SAVE DRIVE #
04E5 33 DB                      XOR     BX,BX                   ; BX = INDEX TO DR_TYPE TABLE
04E7 B9 0006                    MOV     CX,DR_CNT               ; CX = LOOP COUNT
04EA                    RWV_DR_SEARCH:
04EA 2E: 8A A7 0000 R           MOV     AH,CS:DR_TYPE[BX]       ; GET DRIVE TYPE
04EF 80 E4 7F                   AND     AH,BIT7OFF              ; MASK OUT MSB
04F2 3A C4                      CMP     AL,AH                   ; DRIVE TYPE MATCH ?
04F4 75 0B                      JNE     RWV_NXT_MD              ; NO, CHECK NEXT DRIVE TYPE
04F6                    RWV_DR_FND:
04F6 2E: 8B BF 0001 R           MOV     DI,WORD PTR CS:DR_TYPE[BX+1]    ; DI = MEDIA/DRIVE PARAMETER TABLE
04FB                    RWV_MD_SEARCH:
04FB 2E: 3A 75 0C               CMP     DH,CS:[DI].MD_RATE      ; MATCH ?
04FF 74 16                      JE      RWV_MD_FND              ; YES, GO GET 1ST SPECIFY BYTE
0501                    RWV_NXT_MD:
0501 83 C3 03                   ADD     BX,3                    ; CHECK NEXT DRIVE TYPE
0504 E2 E4                      LOOP    RWV_DR_SEARCH
0506 5F                         POP     DI                      ; RESTORE DRIVE #

                        ;--- ASSUME PRIMARY DRIVE IS INSTALLED AS SHIPPED

0507                    RWV_ASSUME:
0507 BB 0012 R                  MOV     BX,OFFSET MD_TBL1       ; POINT TO 40 TK 250 KBS
050A F6 85 0090 R 01            TEST    @DSK_STATE[DI],TRK_CAPA ; TEST FOR 80 TRACK
050F 74 09                      JZ      RWV_MD_FND1             ; MUST BE 40 TRACK
0511 BB 002C R                  MOV     BX,OFFSET MD_TBL3       ; POINT TO 80 TK 500 KBS
0514 EB 04 90                   JMP     RWV_MD_FND1             ; GO SET SPECIFY PARAMETERS

                        ;--- CS:BX POINTS TO MEDIA/DRIVE PARAMETER TABLE

0517                    RWV_MD_FND:

0517 8B DF                      MOV     BX,DI                   ; BX = MEDIA/DRIVE PARAMETER TABLE
0519 5F                         POP     DI                      ; RESTORE DRIVE #
051A                    RWV_MD_FND1:

                        ;--- SEND THE SPECIFY COMMAND TO THE CONTROLLER

051A E8 03E9 R                  CALL    SEND_SPEC_MD
051D E8 0631 R                  CALL    CHK_LASTRATE            ; ZF=1 ATTEMPT RATE IS SAME AS LAST RATE
0520 74 03                      JZ      RWV_DBL                 ; YES, SKIP SEND RATE COMMAND
0522 E8 0618 R                  CALL    SEND_RATE               ; SEND DATA RATE TO NEC


0525                    RWV_DBL:
0525 53                         PUSH    BX                      ; SAVE MEDIA/DRIVE PARAM ADDRESS
0526 E8 080F R                  CALL    SETUP_DBL               ; CHECK FOR DOUBLE STEP
0529 5B                         POP     BX                      ; RESTORE ADDRESS
052A 72 1A                      JC      CHK_RET                 ; ERROR FROM READ ID, POSSIBLE RETRY
052C 58                         POP     AX                      ; RESTORE NEC,DMA COMMAND
052D 50                         PUSH    AX                      ; SAVE NEC COMMAND
052E 53                         PUSH    BX                      ; SAVE MEDIA/DRIVE PARAM ADDRESS
052F E8 0641 R                  CALL    DMA_SETUP               ; SET UP THE DMA
0532 5B                         POP     BX                      ; RESTORE ADDRESS
0533 58                         POP     AX                      ; RESTORE NEC COMMAND
0534 72 1F                      JC      RWV_BAC                 ; CHECK FOR DMA BOUNDARY ERROR
0536 50                         PUSH    AX                      ; SAVE NEC COMMAND
0537 53                         PUSH    BX                      ; SAVE MEDIA/DRIVE PARAM ADDRESS
0538 E8 06A1 R                  CALL    NEC_INIT                ; INITIALIZE NEC
053B 5B                         POP     BX                      ; RESTORE ADDRESS
053C 72 08                      JC      CHK_RET                 ; ERROR - EXIT
053E E8 06C6 R                  CALL    RWV_COM                 ; OP CODE COMMON TO READ/WRITE/VERIFY
0541 72 03                      JC      CHK_RET                 ; ERROR - EXIT
0543 E8 06FC R                  CALL    NEC_TERM                ; TERMINATE, GET STATUS, ETC.

0546                    CHK_RET:
0546 E8 0786 R                  CALL    RETRY                   ; CHECK FOR, SETUP RETRY
0549 58                         POP     AX                      ; RESTORE READ/WRITE/VERIFY PARAMETER
054A 73 03                      JNC     RWV_END                 ; CY = 0 NO RETRY
054C E9 04AA R                  JMP     DO_AGAIN                ; CY = 1 MEANS RETRY

054F                    RWV_END:
054F E8 074E R                  CALL    DSTATE                  ; ESTABLISH STATE IF SUCCESSFUL
0552 E8 07C8 R                  CALL    NUM_TRANS               ; AL = NUMBER TRANSFERRED

0555                    RWV_BAC:                                ; BAD DMA ERROR ENTRY
0555 50                         PUSH    AX                      ; SAVE NUMBER TRANSFERRED
0556 E8 0429 R                  CALL    XLAT_OLD                ; TRANSLATE STATE TO COMPATIBLE MODE
0559 58                         POP     AX                      ; RESTORE NUMBER TRANSFERRED
055A E8 07F5 R                  CALL    SETUP_END               ; VARIOUS CLEANUPS
055D C3                         RET
055E                    RD_WR_VF        ENDP
```

SECTION 5

```
                             ;-------------------------------------------------------------
                             ; SETUP_STATE:  INITIALIZES START AND END RATES.            :
                             ;-------------------------------------------------------------
055E                         SETUP_STATE      PROC    NEAR               ;
055E F6 85 0090 R 10                  TEST    @DSK_STATE[DI],MED_DET   ; MEDIA DETERMINED ?
0563 75 29                            JNZ     J1C                      ; NO STATES IF DETERMINED
0565 B8 4080                          MOV     AX,RATE_300*H+RATE_250   ; AH = START RATE,  AL = END RATE
0568 F6 85 0090 R 04                  TEST    @DSK_STATE[DI],DRV_DET   ; DRIVE ?
056D 74 0A                            JZ      AX_SET                   ; DO NOT KNOW DRIVE
056F F6 85 0090 R 02                  TEST    @DSK_STATE[DI],FMT_CAPA  ; MULTI-RATE ?
0574 75 03                            JNZ     AX_SET                   ; JUMP IF YES
0576 B8 8080                          MOV     AX,RATE_250*X            ; START & END RATE = 250 FOR 360 DRIVE

0579                         AX_SET:
0579 80 A5 0090 R 1F                  AND     @DSK_STATE[DI],NOT RATE_MSK+DBL_STEP ; TURN OFF THE RATE
057E 08 A5 0090 R                     OR      @DSK_STATE[DI],AH        ; RATE FIRST TO TRY
0582 80 26 008B R F3                  AND     @LASTRATE,NOT STRT_MSK   ; ERASE LAST TO TRY RATE BITS
0587 C0 C8 04                         ROR     AL,4                     ; TO OPERATION LAST RATE LOCATION
058A 08 06 008B R                     OR      @LASTRATE,AL             ; LAST RATE
058E                         J1C:
058E C3                               RET
058F                         SETUP_STATE      ENDP
                             ;-------------------------------------------------------------
                             ;  FMT_INIT: ESTABLISH STATE IF UNESTABLISHED AT FORMAT TIME.   :
                             ;-------------------------------------------------------------
058F                         FMT_INIT         PROC    NEAR
058F F6 85 0090 R 10                  TEST    @DSK_STATE[DI],MED_DET   ; IS MEDIA ESTABLISHED
0594 75 42                            JNZ     FI_OUT                   ; IF SO RETURN
0596 E8 0888 R                        CALL    CMOS_TYPE                ; RETURN DRIVE TYPE IN AL
0599 72 3E                            JC      CL_DRV                   ; ERROR IN CMOS ASSUME NO DRIVE
059B FE C8                            DEC     AL                       ; MAKE ZERO ORIGIN
059D 78 3A                            JS      CL_DRV                   ; NO DRIVE IF AL 0
059F 8A A5 0090 R                     MOV     AH,@DSK_STATE[DI]        ; AH = CURRENT STATE
05A3 80 E4 0F                         AND     AH,NOT MED_DET+DBL_STEP+RATE_MSK      ; CLEAR
05A6 0A C0                            OR      AL,AL                    ; CHECK FOR 360
05A8 75 05                            JNZ     N_360                    ; IF 360 WILL BE 0
05AA 80 CC 90                         OR      AH,MED_DET+RATE_250      ; ESTABLISH MEDIA
05AD EB 25                            JMP     SHORT SKP_STATE          ; SKIP OTHER STATE PROCESSING

05AF                         N_360:
05AF FE C8                            DEC     AL                       ; 1.2 M DRIVE
05B1 75 05                            JNZ     N_12                     ; JUMP IF NOT
05B3 80 CC 10                FI_RATE:OR       AH,MED_DET+RATE_500      ; SET FORMAT RATE
05B6 EB 1C                            JMP     SHORT SKP_STATE          ; SKIP OTHER STATE PROCESSING

05B8                         N_12:
05B8 FE C8                            DEC     AL                       ; CHECK FOR TYPE 3
05BA 75 0F                            JNZ     N_720                    ; JUMP IF NOT
05BC F6 C4 04                         TEST    AH,DRV_DET               ; IS DRIVE DETERMINED
05BF 74 10                            JZ      ISNT_12                  ; TREAT AS NON 1.2 DRIVE
05C1 F6 C4 02                         TEST    AH,FMT_CAPA              ; IS 1.2M
05C4 74 0B                            JZ      ISNT_12                  ; JUMP IF NOT
05C6 80 CC 50                         OR      AH,MED_DET+RATE_300      ; RATE 300
05C9 EB 09                            JMP     SHORT SKP_STATE          ; CONTINUE

05CB                         N_720:
05CB FE C8                            DEC     AL                       ; CHECK FOR TYPE 4
05CD 75 0A                            JNZ     CL_DRV                   ; NO DRIVE, CMOS BAD
05CF EB E2                            JMP     SHORT FI_RATE

05D1                         ISNT_12:
05D1 80 CC 90                         OR      AH,MED_DET+RATE_250      ; MUST BE RATE 250

05D4                         SKP_STATE:
05D4 88 A5 0090 R                     MOV     @DSK_STATE[DI],AH        ; STORE AWAY

05D8                         FI_OUT:
05D8 C3                               RET

05D9                         CL_DRV:
05D9 32 E4                            XOR     AH,AH                    ; CLEAR STATE
05DB EB F7                            JMP     SHORT SKP_STATE          ; SAVE IT
05DD                         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                :
                             ;-------------------------------------------------------------
05DD                         MED_CHANGE       PROC    NEAR
05DD E8 0AC4 R                        CALL    READ_DSKCHNG             ; READ DISK CHANGE LINE STATE
05E0 74 34                            JZ      MC_OUT                   ; BYPASS HANDLING DISK CHANGE LINE
05E2 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).

05E7 8B CF                            MOV     CX,DI                    ; CL = DRIVE #
05E9 B0 01                            MOV     AL,1                     ; MOTOR ON BIT MASK
05EB D2 E0                            SHL     AL,CL                    ; TO APPROPRIATE POSITION
05ED F6 D0                            NOT     AL                       ; KEEP ALL BUT MOTOR ON
05EF FA                               CLI                              ; NO INTERRUPTS
05F0 20 06 003F R                     AND     @MOTOR_STATUS,AL         ; TURN MOTOR OFF INDICATOR
05F4 FB                               STI                              ; INTERRUPTS ENABLED
05F5 E8 08B6 R                        CALL    MOTOR_ON                 ; TURN MOTOR ON

                             ;-----   THIS SEQUENCE OF SEEKS IS USED TO RESET DISKETTE CHANGE SIGNAL

05F8 E8 00E7 R                        CALL    DISK_RESET               ; RESET NEC
05FB B5 01                            MOV     CH,01H                   ; MOVE TO CYLINDER 1
05FD E8 09C0 R                        CALL    SEEK                     ; ISSUE SEEK
0600 32 ED                            XOR     CH,CH                    ; MOVE TO CYLINDER 0
0602 E8 09C0 R                        CALL    SEEK                     ; ISSUE SEEK
0605 C6 06 0041 R 06                  MOV     @DSKETTE_STATUS,MEDIA_CHANGE     ; STORE IN STATUS

060A E8 0AC4 R               OK1:     CALL    READ_DSKCHNG             ; CHECK MEDIA CHANGED AGAIN
060D 74 05                            JZ      OK2                      ; IF ACTIVE, NO DISKETTE, TIMEOUT

060F C6 06 0041 R 80         OK4:     MOV     @DSKETTE_STATUS,TIME_OUT; TIMEOUT IF DRIVE EMPTY

0614 F9                      OK2:     STC                              ; MEDIA CHANGED, SET CY
```

```
0615 C3                         RET
0616                    MC_OUT:
0616 F8                         CLC                     ; NO MEDIA CHANGED, CLEAR CY
0617 C3                         RET
0618                    MED_CHANGE      ENDP
                        ;-------------------------------------------------------------
                        ; SEND_RATE                                                  :
                        ;       SENDS DATA RATE COMMAND TO NEC                       :
                        ; ON ENTRY:     DI = DRIVE #                                 :
                        ; ON EXIT:      NONE                                         :
                        ; REGISTERS ALTERED: DX                                      :
                        ;-------------------------------------------------------------
0618                    SEND_RATE       PROC    NEAR            ;

0618 50                         PUSH    AX                      ; SAVE REG.
0619 80 26 008B R 3F            AND     @LASTRATE,NOT SEND_MSK  ; ELSE CLEAR LAST RATE ATTEMPTED
061E 8A 85 0090 R               MOV     AL,@DSK_STATE[DI]       ; GET RATE STATE OF THIS DRIVE
0622 24 C0                      AND     AL,SEND_MSK             ; KEEP ONLY RATE BITS
0624 08 06 008B R               OR      @LASTRATE,AL            ; SAVE NEW RATE FOR NEXT CHECK
0628 C0 C0 02                   ROL     AL,2                    ; MOVE TO BIT OUTPUT POSITIONS
062B BA 03F7                    MOV     DX,03F7H                ; OUTPUT NEW DATA RATE
062E EE                         OUT     DX,AL

062F 58                         POP     AX                      ; RESTORE REG.
0630 C3                         RET
0631                    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                                    :
                        ;-------------------------------------------------------------
0631                    CHK_LASTRATE    PROC    NEAR
0631 50                         PUSH    AX                      ; SAVE REG
0632 8A 26 008B R               MOV     AH,@LASTRATE            ; GET LAST DATA RATE SELECTED
0636 8A 85 0090 R               MOV     AL,@DSK_STATE[DI]       ; GET RATE STATE OF THIS DRIVE
063A 25 C0C0                    AND     AX,SEND_MSK*X           ; KEEP ONLY RATE BITS OF BOTH
063D 3A C4                      CMP     AL,AH                   ; COMPARE TO PREVIOUSLY TRIED
                                                                ; ZF = 1  RATE IS THE SAME
063F 58                         POP     AX                      ; RESTORE REG.
0640 C3                         RET
0641                    CHK_LASTRATE    ENDP

                        SUBTTL (DSK3.ASM)
```

SECTION 5

```
                           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 :
                           ;-------------------------------------------------------------
0641                       DMA_SETUP    PROC    NEAR
0641 FA                            CLI                         ; DISABLE INTERRUPTS DURING DMA SET-UP
0642 E6 0C                         OUT     DMA+12,AL           ; SET THE FIRST/LAST F/F
0644 EB 00                         JMP     $+2                 ; WAIT FOR I/O
0646 E6 0B                         OUT     DMA+11,AL           ; OUTPUT THE MODE BYTE
0648 3C 42                         CMP     AL,42H              ; DMA VERIFY COMMAND
064A 75 04                         JNE     NOT_VERF            ; NO
064C 33 C0                         XOR     AX,AX               ; START ADDRESS
064E EB 10                         JMP     SHORT J33
0650                       NOT_VERF:
0650 8C C0                         MOV     AX,ES               ; GET THE ES VALUE
0652 C1 C0 04                      ROL     AX,4                ; ROTATE LEFT
0655 8A E8                         MOV     CH,AL               ; GET HIGHEST NIBBLE OF ES TO CH
0657 24 F0                         AND     AL,11110000B        ; ZERO THE LOW NIBBLE FROM SEGMENT
0659 03 46 02                      ADD     AX,[BP+2]           ; TEST FOR CARRY FROM ADDITION
065C 73 02                         JNC     J33
065E FE C5                         INC     CH                  ; CARRY MEANS HIGH 4 BITS MUST BE INC
0660                       J33:
0660 50                            PUSH    AX                  ; SAVE START ADDRESS
0661 E6 04                         OUT     DMA+4,AL            ; OUTPUT LOW ADDRESS
0663 EB 00                         JMP     $+2                 ; WAIT FOR I/O
0665 8A C4                         MOV     AL,AH
0667 E6 04                         OUT     DMA+4,AL            ; OUTPUT HIGH ADDRESS
0669 8A C5                         MOV     AL,CH               ; GET HIGH 4 BITS
066B EB 00                         JMP     $+2                 ; I/O WAIT STATE
066D 24 0F                         AND     AL,00001111B
066F E6 81                         OUT     081H,AL             ; OUTPUT HIGH 4 BITS TO PAGE REGISTER

                           ;-----  DETERMINE COUNT

0671 8B C6                         MOV     AX,SI               ; AL = # OF SECTORS
0673 86 C4                         XCHG    AL,AH               ; AH = # OF SECTORS
0675 2A C0                         SUB     AL,AL               ; AL = 0, AX = # OF SECTORS * 256
0677 D1 E8                         SHR     AX,1                ; AX = # SECTORS * 128
0679 50                            PUSH    AX                  ; SAVE # OF SECTORS * 128
067A B2 03                         MOV     DL,3                ; GET BYTES/SECTOR PARAMETER
067C E8 08A1 R                     CALL    GET_PARM            ; "
067F 8A CC                         MOV     CL,AH               ; SHIFT COUNT (0=128, 1=256 ETC)
0681 58                            POP     AX                  ; AX = # OF SECTORS * 128
0682 D3 E0                         SHL     AX,CL               ; SHIFT BY PARAMETER VALUE
0684 48                            DEC     AX                  ;-1 FOR DMA VALUE
0685 50                            PUSH    AX                  ; SAVE COUNT VALUE
0686 E6 05                         OUT     DMA+5,AL            ; LOW BYTE OF COUNT
0688 EB 00                         JMP     $+2                 ; WAIT FOR I/O
068A 8A C4                         MOV     AL,AH
068C E6 05                         OUT     DMA+5,AL            ; HIGH BYTE OF COUNT
068E FB                            STI                         ; RE-ENABLE INTERRUPTS
068F 59                            POP     CX                  ; RECOVER COUNT VALUE
0690 58                            POP     AX                  ; RECOVER ADDRESS VALUE
0691 03 C1                         ADD     AX,CX               ; ADD, TEST FOR 64K OVERFLOW
0693 B0 02                         MOV     AL,2                ; MODE FOR 8237
0695 EB 00                         JMP     $+2                 ; WAIT FOR I/O
0697 E6 0A                         OUT     DMA+10,AL           ; INITIALIZE THE DISKETTE CHANNEL

0699 73 05                         JNC     NO_BAD              ; CHECK FOR ERROR
069B C6 06 0041 R 09               MOV     @DSKETTE_STATUS,DMA_BOUNDARY    ; SET ERROR

06A0                       NO_BAD:
06A0 C3                            RET                         ; CY SET BY ABOVE IF ERROR
06A1                       DMA_SETUP    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 :
                           ;-------------------------------------------------------------
06A1                       NEC_INIT     PROC    NEAR
06A1 50                            PUSH    AX                  ; SAVE NEC COMMAND
06A2 E8 08B6 R                     CALL    MOTOR_ON            ; TURN MOTOR ON FOR SPECIFIC DRIVE

                           ;-----  DO THE SEEK OPERATION

06A5 8A 6E 01                      MOV     CH,[BP+1]           ; CH = TRACK #
06A8 E8 09C0 R                     CALL    SEEK                ; MOVE TO CORRECT TRACK
06AB 58                            POP     AX                  ; RECOVER COMMAND
06AC 72 17                         JC      ER_1                ; ERROR ON SEEK
06AE BB 06C5 R                     MOV     BX,OFFSET ER_1      ; LOAD ERROR ADDRESS
06B1 53                            PUSH    BX                  ; PUSH NEC_OUT ERROR RETURN

                           ;-----  SEND OUT THE PARAMETERS TO THE CONTROLLER

06B2 E8 0994 R                     CALL    NEC_OUTPUT          ; OUTPUT THE OPERATION COMMAND
06B5 8B C6                         MOV     AX,SI               ; AH = HEAD #
06B7 8B DF                         MOV     BX,DI               ; BL = DRIVE #
06B9 C0 E4 02                      SAL     AH,2                ; MOVE IT TO BIT 2
06BC 80 E4 04                      AND     AH,00000100B        ; ISOLATE THAT BIT
06BF 0A E3                         OR      AH,BL               ; OR IN THE DRIVE NUMBER
06C1 E8 0994 R                     CALL    NEC_OUTPUT          ; FALL THRU CY SET IF ERROR
06C4 5B                            POP     BX                  ; THROW AWAY ERROR RETURN
06C5                       ER_1:
06C5 C3                            RET
06C6                       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 :
                           ;-------------------------------------------------------------
```

```
06C6                       RWV_COM PROC   NEAR
06C6 B8 06FB R                     MOV    AX,OFFSET ER_2       ; LOAD ERROR ADDRESS
06C9 50                            PUSH   AX                   ; PUSH NEC_OUT ERROR RETURN
06CA 8A 66 01                      MOV    AH,[BP+1]            ; OUTPUT TRACK #
06CD E8 0994 R                     CALL   NEC_OUTPUT
06D0 8B C6                         MOV    AX,SI                ; OUTPUT HEAD #
06D2 E8 0994 R                     CALL   NEC_OUTPUT
06D5 8A 66 00                      MOV    AH,[BP]              ; OUTPUT SECTOR #
06D8 E8 0994 R                     CALL   NEC_OUTPUT
06DB B2 03                         MOV    DL,3                 ; BYTES/SECTOR PARAMETER FROM BLOCK
06DD E8 08A1 R                     CALL   GET_PARM             ; . TO THE NEC
06E0 E8 0994 R                     CALL   NEC_OUTPUT           ; OUTPUT TO CONTROLLER
06E3 B2 04                         MOV    DL,4                 ; EOT PARAMETER FROM BLOCK
06E5 E8 08A1 R                     CALL   GET_PARM             ; . TO THE NEC
06E8 E8 0994 R                     CALL   NEC_OUTPUT           ; OUTPUT TO CONTROLLER

06EB 2E: 8A 67 05                  MOV    AH,CS:[BX].MD_GAP     ; GET GAP LENGTH

06EF                       R15:
06EF E8 0994 R                     CALL   NEC_OUTPUT
06F2 B2 06                         MOV    DL,6                 ; DTL PARAMETER FROM BLOCK
06F4 E8 08A1 R                     CALL   GET_PARM             ;  TO THE NEC
06F7 E8 0994 R                     CALL   NEC_OUTPUT           ; OUTPUT TO CONTROLLER
06FA 58                            POP    AX                   ; THROW AWAY ERROR EXIT
06FB                       ER_2:
06FB C3                            RET
06FC                       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 :
                           ;-------------------------------------------------------------
06FC                       NEC_TERM        PROC    NEAR

                           ;-----  LET THE OPERATION HAPPEN

06FC 56                            PUSH   SI                   ; SAVE HEAD #, # OF SECTORS
06FD E8 0A5D R                     CALL   WAIT_INT             ; WAIT FOR THE INTERRUPT
0700 9C                            PUSHF
0701 E8 0A85 R                     CALL   RESULTS              ; GET THE NEC STATUS
0704 72 45                         JC     SET_END_POP
0706 9D                            POPF
0707 72 3A                         JC     SET_END              ; LOOK FOR ERROR

                           ;-----  CHECK THE RESULTS RETURNED BY THE CONTROLLER

0709 FC                            CLD                         ; SET THE CORRECT DIRECTION
070A BE 0042 R                     MOV    SI,OFFSET @NEC_STATUS ; POINT TO STATUS FIELD
070D AC                            LODS   @NEC_STATUS          ; GET ST0
070E 24 C0                         AND    AL,11000000B         ; TEST FOR NORMAL TERMINATION
0710 74 31                         JZ     SET_END              ;
0712 3C 40                         CMP    AL,01000000B         ; TEST FOR ABNORMAL TERMINATION
0714 75 27                         JNZ    J18                  ; NOT ABNORMAL, BAD NEC

                           ;-----  ABNORMAL TERMINATION, FIND OUT WHY

0716 AC                            LODS   @NEC_STATUS          ; GET ST1
0717 D0 E0                         SAL    AL,1                 ; TEST FOR EOT FOUND
0719 B4 04                         MOV    AH,RECORD_NOT_FND
071B 72 22                         JC     J19
071D C0 E0 02                      SAL    AL,2
0720 B4 10                         MOV    AH,BAD_CRC
0722 72 1B                         JC     J19
0724 D0 E0                         SAL    AL,1                 ; TEST FOR DMA OVERRUN
0726 B4 08                         MOV    AH,BAD_DMA
0728 72 15                         JC     J19
072A C0 E0 02                      SAL    AL,2                 ; TEST FOR RECORD NOT FOUND
072D B4 04                         MOV    AH,RECORD_NOT_FND
072F 72 0E                         JC     J19
0731 D0 E0                         SAL    AL,1
0733 B4 03                         MOV    AH,WRITE_PROTECT     ; TEST FOR WRITE_PROTECT
0735 72 08                         JC     J19
0737 D0 E0                         SAL    AL,1                 ; TEST MISSING ADDRESS MARK
0739 B4 02                         MOV    AH,BAD_ADDR_MARK
073B 72 02                         JC     J19

                           ;-----  NEC MUST HAVE FAILED
073D                       J18:
073D B4 20                         MOV    AH,BAD_NEC
073F                       J19:
073F 08 26 0041 R                  OR     @DSKETTE_STATUS,AH
0743                       SET_END:
0743 80 3E 0041 R 01               CMP    @DSKETTE_STATUS,1    ; SET ERROR CONDITION
0748 F5                            CMC                         ;
0749 5E                            POP    SI                   ; RESTORE HEAD #, # OF SECTORS
074A C3                            RET

074B                       SET_END_POP:
074B 9D                            POPF
074C EB F5                         JMP    SHORT SET_END
074E                       NEC_TERM        ENDP
                           ;-------------------------------------------------------------
                           ; DSTATE:       ESTABLISH STATE UPON SUCCESSFUL OPERATION.     :
                           ;-------------------------------------------------------------
074E                       DSTATE  PROC   NEAR
074E 80 3E 0041 R 00               CMP    @DSKETTE_STATUS,0    ; CHECK FOR ERROR
0753 75 30                         JNZ    SETBAC               ; IF ERROR JUMP
0755 80 8D 0090 R 10               OR     @DSK_STATE[DI],MED_DET ; NO ERROR, MARK MEDIA AS DETERMINED
075A F6 85 0090 R 04               TEST   @DSK_STATE[DI],DRV_DET ; DRIVE DETERMINED ?
075F 75 24                         JNZ    SETBAC               ; IF DETERMINED NO TRY TO DETERMINE
0761 8A 85 0090 R                  MOV    AL,@DSK_STATE[DI]    ; LOAD STATE
0765 24 C0                         AND    AL,RATE_MSK          ; KEEP ONLY RATE
0767 3C 80                         CMP    AL,RATE_250          ; RATE 250 ?
0769 75 15                         JNE    M_12                 ; NO, MUST BE 1.2M OR 1.44M DRV

                           ;--- CHECK IF IT IS 1.44M

076B E8 0888 R                     CALL   CMOS_TYPE            ; RETURN DRIVE TYPE IN (AL)
076E 72 10                         JC     M_12                 ; CMOS BAD
0770 3C 04                         CMP    AL,04                ; 1.44MB DRIVE ?
0772 74 0C                         JE     M_12                 ; YES
```

SECTION 5

```
0774                          M_720:
0774 80 A5 0090 R FD                  AND     @DSK_STATE[DI],NOT FMT_CAPA ; TURN OFF FORMAT CAPA
0779 80 8D 0090 R 04                  OR      @DSK_STATE[DI],DRV_DET   ; MARK DRIVE DETERMINED
077E EB 05                            JMP     SHORT SETBAC             ; BACK

0780                          M_12:
0780 80 8D 0090 R 06                  OR      @DSK_STATE[DI],DRV_DET+FMT_CAPA   ; TURN ON DETERMINED & FMT CAPA

0785                          SETBAC:
0785 C3                               RET
0786                          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        :
                              ;--------------------------------------------------------------
0786                          RETRY   PROC    NEAR
0786 80 3E 0041 R 00                  CMP     @DSKETTE_STATUS,0        ; GET STATUS OF OPERATION
078B 74 39                            JZ      NO_RETRY                 ; SUCCESSFUL OPERATION
078D 80 3E 0041 R 80                  CMP     @DSKETTE_STATUS,TIME_OUT        ; IF TIME OUT NO RETRY
0792 74 32                            JZ      NO_RETRY                 ;
0794 8A A5 0090 R                     MOV     AH,@DSK_STATE[DI]        ; GET MEDIA STATE OF DRIVE
0798 F6 C4 10                         TEST    AH,MED_DET               ; ESTABLISHED/DETERMINED ?
079B 75 29                            JNZ     NO_RETRY                 ; IF ESTABLISHED STATE THEN TRUE ERROR
079D 80 E4 C0                         AND     AH,RATE_MSK              ; ISOLATE RATE
07A0 8A 2E 008B R                     MOV     CH,@LASTRATE             ; GET START OPERATION STATE
07A4 C0 C5 04                         ROL     CH,4                     ; TO CORRESPONDING BITS
07A7 80 E5 C0                         AND     CH,RATE_MSK              ; ISOLATE RATE BITS
07AA 3A EC                            CMP     CH,AH                    ; ALL RATES TRIED
07AC 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)

07AE 80 FC 01                         CMP     AH,RATE_500+1            ; SET CY FOR RATE 500
07B1 D0 DC                            RCR     AH,1                     ; TO NEXT STATE
07B3 80 E4 C0                         AND     AH,RATE_MSK              ; KEEP ONLY RATE BITS
07B6 80 A5 0090 R 1F                  AND     @DSK_STATE[DI],NOT RATE_MSK+DBL_STEP     ; RATE, DBL STEP OFF
07BB 08 A5 0090 R                     OR      @DSK_STATE[DI],AH        ; TURN ON NEW RATE
07BF C6 06 0041 R 00                  MOV     @DSKETTE_STATUS,0        ; RESET STATUS FOR RETRY
07C4 F9                               STC                              ; SET CARRY FOR RETRY
07C5 C3                               RET                              ; RETRY RETURN

07C6                          NO_RETRY:
07C6 F8                               CLC                              ; CLEAR CARRY NO RETRY
07C7 C3                               RET                              ; NO RETRY RETURN
07C8                          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             :
                              ;--------------------------------------------------------------
07C8                          NUM_TRANS       PROC    NEAR
07C8 32 C0                            XOR     AL,AL                    ; CLEAR FOR ERROR
07CA 80 3E 0041 R 00                  CMP     @DSKETTE_STATUS,0        ; CHECK FOR ERROR
07CF 75 23                            JNZ     NT_OUT                   ; IF ERROR 0 TRANSFERRED
07D1 B2 04                            MOV     DL,4                     ; SECTORS/TRACK OFFSET TO DL
07D3 E8 08A1 R                        CALL    GET_PARM                 ; AH = SECTORS/TRACK
07D6 8A 1E 0047 R                     MOV     BL,@NEC_STATUS+5         ; GET ENDING SECTOR
07DA 8B CE                            MOV     CX,SI                    ; CH = HEAD # STARTED
07DC 3A 2E 0046 R                     CMP     CH,@NEC_STATUS+4         ; GET HEAD ENDED UP ON
07E0 75 0B                            JNZ     DIF_HD                   ; IF ON SAME HEAD, THEN NO ADJUST

07E2 8A 2E 0045 R                     MOV     CH,@NEC_STATUS+3         ; GET TRACK ENDED UP ON
07E6 3A 6E 01                         CMP     CH,[BP+1]                ; IS IT ASKED FOR TRACK
07E9 74 04                            JZ      SAME_TRK                 ; IF SAME TRACK NO INCREASE

07EB 02 DC                            ADD     BL,AH                    ; ADD SECTORS/TRACK
07ED                          DIF_HD:
07ED 02 DC                            ADD     BL,AH                    ; ADD SECTORS/TRACK
07EF                          SAME_TRK:
07EF 2A 5E 00                         SUB     BL,[BP]                  ; SUBTRACT START FROM END
07F2 8A C3                            MOV     AL,BL                    ; TO AL

07F4                          NT_OUT:
07F4 C3                               RET
07F5                          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    :
                              ;--------------------------------------------------------------
07F5                          SETUP_END       PROC    NEAR
07F5 B2 02                            MOV     DL,2                     ; GET THE MOTOR WAIT PARAMETER
07F7 50                               PUSH    AX                       ; SAVE NUMBER TRANSFERRED
07F8 E8 08A1 R                        CALL    GET_PARM
07FB 88 26 0040 R                     MOV     @MOTOR_COUNT,AH          ; STORE UPON RETURN
07FF 58                               POP     AX                       ; RESTORE NUMBER TRANSFERRED
0800 8A 26 0041 R                     MOV     AH,@DSKETTE_STATUS       ; GET STATUS OF OPERATION
0804 0A E4                            OR      AH,AH                    ; CHECK FOR ERROR
0806 74 02                            JZ      NUN_ERR                  ; NO ERROR
0808 32 C0                            XOR     AL,AL                    ; CLEAR NUMBER RETURNED

080A                          NUN_ERR:
080A 80 FC 01                         CMP     AH,1                     ; SET THE CARRY FLAG TO INDICATE
080D F5                               CMC                              ; SUCCESS OR FAILURE
080E C3                               RET
080F                          SETUP_END       ENDP
                              ;--------------------------------------------------------------
                              ; SETUP_DBL                                                    :
                              ;       CHECK DOUBLE STEP.                                     :
                              ; ON ENTRY:                                                    :
```

```
                                     ;              DI = DRIVE                                       :
                                     ; ON EXIT :    CY = 1 MEANS ERROR                               :
                                     ;---------------------------------------------------------------
 080F                                SETUP_DBL      PROC    NEAR
 080F 8A A5 0090 R                          MOV     AH,@DSK_STATE[DI]        ; ACCESS STATE
 0813 F6 C4 10                              TEST    AH,MED_DET               ; ESTABLISHED STATE ?
 0816 75 59                                 JNZ     NO_DBL                   ; IF ESTABLISHED THEN DOUBLE DONE

                                     ;-----  CHECK FOR TRACK 0 TO SPEED UP ACKNOWLEDGE OF UNFORMATTED DISKETTE

 0818 C6 06 003E R 00                       MOV     @SEEK_STATUS,0           ; SET RECALIBRATE REQUIRED ON ALL DRIVES
 081D E8 08B6 R                             CALL    MOTOR_ON                 ; ENSURE MOTOR STAY ON
 0820 B5 00                                 MOV     CH,0                     ; LOAD TRACK 0
 0822 E8 09C0 R                             CALL    SEEK                     ; SEEK TO TRACK 0
 0825 E8 0873 R                             CALL    READ_ID                  ; READ ID FUNCTION
 0828 72 32                                 JC      SD_ERR                   ; IF ERROR NO TRACK 0

                                     ;-----  INITIALIZE START AND MAX TRACKS (TIMES 2 FOR BOTH HEADS)

 082A B9 0450                               MOV     CX,0450H                 ; START, MAX TRACKS
 082D F6 85 0090 R 01                       TEST    @DSK_STATE[DI],TRK_CAPA  ; TEST FOR 80 TRACK CAPABILITY
 0832 74 02                                 JZ      CNT_OK                   ; IF NOT COUNT IS SETUP
 0834 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.

 0836                                CNT_OK:
 0836 51                                    PUSH    CX                       ; SAVE TRACK, COUNT
 0837 C6 06 0041 R 00                       MOV     @DISKETTE_STATUS,0       ; CLEAR STATUS, EXPECT ERRORS
 083C 33 C0                                 XOR     AX,AX                    ; CLEAR AX
 083E D0 ED                                 SHR     CH,1                     ; HALVE TRACK, CY = HEAD
 0840 C0 D0 03                              RCL     AL,3                     ; AX = HEAD IN CORRECT BIT
 0843 50                                    PUSH    AX                       ; SAVE HEAD
 0844 E8 09C0 R                             CALL    SEEK                     ; SEEK TO TRACK
 0847 58                                    POP     AX                       ; RESTORE HEAD
 0848 0B F8                                 OR      DI,AX                    ; DI = HEAD OR'ED DRIVE
 084A E8 0873 R                             CALL    READ_ID                  ; READ ID HEAD 0
 084D 9C                                    PUSHF                            ; SAVE RETURN FROM READ_ID
 084E 81 E7 00FB                            AND     DI,11111011B             ; TURN OFF HEAD 1 BIT
 0852 9D                                    POPF                             ; RESTORE ERROR RETURN
 0853 59                                    POP     CX                       ; RESTORE COUNT
 0854 73 08                                 JNC     DO_CHK                   ; IF OK, ASKED = RETURNED TRACK ?
 0856 FE C5                                 INC     CH                       ; INC FOR NEXT TRACK
 0858 3A E9                                 CMP     CH,CL                    ; REACHED MAXIMUM YET
 085A 75 DA                                 JNZ     CNT_OK                   ; CONTINUE TILL ALL TRIED

                                     ;-----  FALL THRU, READ ID FAILED FOR ALL TRACKS

 085C                                SD_ERR:
 085C F9                                    STC                              ; SET CARRY FOR ERROR
 085D C3                                    RET                              ; SETUP_DBL ERROR EXIT

 085E                                DO_CHK:
 085E 8A 0E 0045 R                          MOV     CL,@NEC_STATUS+3         ; LOAD RETURNED TRACK
 0862 88 8D 0094 R                          MOV     @DSK_TRK[DI],CL          ; STORE TRACK NUMBER
 0866 D0 ED                                 SHR     CH,1                     ; HALVE TRACK
 0868 3A E9                                 CMP     CH,CL                    ; IS IT THE SAME AS ASKED FOR TRACK
 086A 74 05                                 JZ      NO_DBL                   ; IF SAME THEN NO DOUBLE STEP
 086C 80 8D 0090 R 20                       OR      @DSK_STATE[DI],DBL_STEP  ; TURN ON DOUBLE STEP REQUIRED

 0871                                NO_DBL:
 0871 F8                                    CLC                              ; CLEAR ERROR FLAG
 0872 C3                                    RET
 0873                                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            :
                                     ;              @DISKETTE_STATUS, CY REFLECT STATUS OF OPERATION :
                                     ;---------------------------------------------------------------
 0873                                READ_ID PROC   NEAR
 0873 B8 0887 R                             MOV     AX,OFFSET ER_3           ; MOVE NEC OUTPUT ERROR ADDRESS
 0876 50                                    PUSH    AX
 0877 B4 4A                                 MOV     AH,4AH                   ; READ ID COMMAND
 0879 E8 0994 R                             CALL    NEC_OUTPUT               ; TO CONTROLLER
 087C 8B C7                                 MOV     AX,DI                    ; DRIVE # TO AH, HEAD 0
 087E 8A E0                                 MOV     AH,AL
 0880 E8 0994 R                             CALL    NEC_OUTPUT               ; TO CONTROLLER
 0883 E8 06FC R                             CALL    NEC_TERM                 ; WAIT FOR OPERATION, GET STATUS
 0886 58                                    POP     AX                       ; THROW AWAY ERROR ADDRESS
 0887                                ER_3:
 0887 C3                                    RET
 0888                                READ_ID ENDP
                                     ;---------------------------------------------------------------
                                     ; CMOS_TYPE                                                     :
                                     ;      RETURNS DISKETTE TYPE FROM CMOS                          :
                                     ;                                                               :
                                     ; ON ENTRY:    DI = DRIVE #                                     :
                                     ;                                                               :
                                     ; ON EXIT:     AL = TYPE; CY REFLECTS STATUS                    :
                                     ;---------------------------------------------------------------
 0888                                CMOS_TYPE      PROC    NEAR
 0888 B0 0E                                 MOV     AL,CMOS_DIAG             ; CMOS DIAGNOSTIC STATUS BYTE ADDRESS
 088A E8 0000 E                             CALL    CMOS_READ                ; GET CMOS STATUS
 088D A8 C0                                 TEST    AL,BAD_BAT+BAD_CKSUM     ; BATTERY GOOD AND CHECKSUM VALID ?
 088F F9                                    STC                              ; SET CY = 1 INDICATING ERROR FOR RETURN
 0890 75 0E                                 JNZ     BAD_CM                   ; ERROR IF EITHER BIT ON

 0892 B0 10                                 MOV     AL,CMOS_DISKETTE         ; ADDRESS OF DISKETTE BYTE IN CMOS
 0894 E8 0000 E                             CALL    CMOS_READ                ; GET DISKETTE BYTE
 0897 0B FF                                 OR      DI,DI                    ; SEE WHICH DRIVE IN QUESTION
 0899 75 03                                 JNZ     TB                       ; IF DRIVE 1, DATA IN LOW NIBBLE

 089B C0 C8 04                              ROR     AL,4                     ; EXCHANGE NIBBLES IF SECOND DRIVE
 089E                                TB:
 089E 24 0F                                 AND     AL,00FH                  ; KEEP ONLY DRIVE DATA, RESET CY = 0

 08A0                                BAD_CM:

 08A0 C3                                    RET
```

SECTION 5

```
08A1                    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                                        :
                        ;---------------------------------------------------------------------
08A1                    GET_PARM        PROC    NEAR
08A1 1E                         PUSH    DS
08A2 56                         PUSH    SI
08A3 2B C0                      SUB     AX,AX                   ; DS = 0 , BIOS DATA AREA
08A5 8E D8                      MOV     DS,AX
08A7 87 D3                      XCHG    DX,BX                   ; BL = INDEX
08A9 2A FF                      SUB     BH,BH                   ; BX = INDEX
                                ASSUME  DS:ABS0
08AB C5 36 0078 R               LDS     SI,@DISK_POINTER        ; POINT TO BLOCK
08AF 8A 20                      MOV     AH,[SI+BX]              ; GET THE WORD
08B1 87 D3                      XCHG    DX,BX                   ; RESTORE BX
08B3 5E                         POP     SI
08B4 1F                         POP     DS
08B5 C3                         RET
                                ASSUME  DS:DATA
08B6                    GET_PARM        ENDP
                        ;------------------------------------------------------------------------------
                        ; MOTOT_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                                            :
                        ;------------------------------------------------------------------------------
08B6                    MOTOR_ON        PROC    NEAR
08B6 53                         PUSH    BX                      ; SAVE REG.
08B7 E8 0901 R                  CALL    TURN_ON                 ; TURN ON MOTOR
08BA 72 43                      JC      MOT_IS_ON               ; IF CY=1 NO WAIT
08BC E8 0429 R                  CALL    XLAT_OLD                ; TRANSLATE STATE TO COMPATIBLE MODE
08BF B8 90FD                    MOV     AX,090FDH               ; LOAD WAIT CODE & TYPE
08C2 CD 15                      INT     15H                     ; TELL OPERATING SYSTEM ABOUT TO DO WAIT
08C4 9C                         PUSHF                           ; SAVE CY FOR TEST
08C5 E8 0403 R                  CALL    XLAT_NEW                ; TRANSLATE STATE TO PRESENT ARCH.
08C8 9D                         POPF                            ; RESTORE CY FOR TEST
08C9 73 05                      JNC     M_WAIT                  ; BYPASS LOOP IF OP SYSTEM HANDLED WAIT
08CB E8 0901 R                  CALL    TURN_ON                 ; CHECK AGAIN IF MOTOR ON
08CE 72 2F                      JC      MOT_IS_ON               ; IF NO WAIT MEANS IT IS ON

08D0                    M_WAIT:
08D0 B2 0A                      MOV     DL,10                   ; GET THE MOTOR WAIT PARAMETER
08D2 E8 08A1 R                  CALL    GET_PARM
08D5 8A C4                      MOV     AL,AH                   ; AL = MOTOR WAIT PARAMETER
08D7 32 E4                      XOR     AH,AH                   ; AX = MOTOR WAIT PARAMETER
08D9 3C 08                      CMP     AL,8                    ; SEE IF AT LEAST A SECOND IS SPECIFIED
08DB 73 02                      JAE     GP2                     ; IF YES, CONTINUE
08DD B0 08                      MOV     AL,8                    ; ONE SECOND WAIT FOR MOTOR START UP

                        ;-----  AX CONTAINS NUMBER OF 1/8 SECONDS (125000 MICROSECONDS) TO WAIT

08DF 50                 GP2:    PUSH    AX                      ; SAVE WAIT PARAMETER
08E0 BA F424                    MOV     DX,62500                ; LOAD LARGEST POSSIBLE MULTIPLIER
08E3 F7 E2                      MUL     DX                      ; MULTIPLY BY HALF OF WHAT'S NECESSARY
08E5 8B CA                      MOV     CX,DX                   ; CX = HIGH WORD
08E7 8B D0                      MOV     DX,AX                   ; CX,DX = 1/2 * (# OF MICROSECONDS)
08E9 F8                         CLC                             ; CLEAR CARRY FOR ROTATE
08EA D1 D2                      RCL     DX,1                    ; DOUBLE LOW WORD, CY CONTAINS OVERFLOW
08EC D1 D1                      RCL     CX,1                    ; DOUBLE HI, INCLUDING LOW WORD OVERFLOW
08EE B4 86                      MOV     AH,86H                  ; LOAD WAIT CODE
08F0 CD 15                      INT     15H                     ; PERFORM WAIT
08F2 58                         POP     AX                      ; RESTORE WAIT PARAMETER
08F3 73 0A                      JNC     MOT_IS_ON               ; CY MEANS WAIT COULD NOT BE DONE

                        ;-----  FOLLOWING LOOPS REQUIRED WHEN RTC WAIT FUNCTION IS ALREADY IN USE

08F5                    J13:                                    ;       WAIT FOR 1/8 SECOND PER (AL)
08F5 B9 205E                    MOV     CX,8286                 ; COUNT FOR 1/8 SECOND AT 15.085737 US
08F8 E8 0000 E                  CALL    WAITF                   ; GO TO FIXED WAIT ROUTINE
08FB FE C8                      DEC     AL                      ; DECREMENT TIME VALUE
08FD 75 F6                      JNZ     J13                     ; ARE WE DONE YET

08FF                    MOT_IS_ON:
08FF 5B                         POP     BX                      ; RESTORE REG.
0900 C3                         RET
0901                    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                                  :
                        ;---------------------------------------------------------------------
0901                    TURN_ON PROC    NEAR
0901 8B DF                      MOV     BX,DI                   ; BX = DRIVE #
0903 8A CB                      MOV     CL,BL                   ; CL = DRIVE #
0905 C0 C3 04                   ROL     BL,4                    ; BL = DRIVE SELECT
```

```
0908 FA                          CLI                                   ; NO INTERRUPTS WHILE DETERMINING STATUS
0909 C6 06 0040 R FF             MOV     @MOTOR_COUNT,0FFH             ; ENSURE MOTOR STAYS ON FOR OPERATION
090E A0 003F R                   MOV     AL,@MOTOR_STATUS              ; GET DIGITAL OUTPUT REGISTER REFLECTION
0911 24 30                       AND     AL,00110000B                  ; KEEP ONLY DRIVE SELECT BITS
0913 B4 01                       MOV     AH,1                          ; MASK FOR DETERMINING MOTOR BIT
0915 D2 E4                       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

0917 3A C3                       CMP     AL,BL                         ; REQUESTED DRIVE ALREADY SELECTED ?
0919 75 06                       JNZ     TURN_IT_ON                    ; IF NOT SELECTED JUMP
091B 84 26 003F R                TEST    AH,@MOTOR_STATUS              ; TEST MOTOR ON BIT
091F 75 2C                       JNZ     NO_MOT_WAIT                   ; JUMP IF MOTOR ON AND SELECTED

0921                    TURN_IT_ON:
0921 0A E3                       OR      AH,BL                         ; AH = DRIVE SELECT AND MOTOR ON
0923 8A 3E 003F R                MOV     BH,@MOTOR_STATUS              ; SAVE COPY OF @MOTOR_STATUS BEFORE
0927 80 E7 0F                    AND     BH,00001111B                  ; KEEP ONLY MOTOR BITS
092A 80 26 003F R CF             AND     @MOTOR_STATUS,11001111B       ; CLEAR OUT DRIVE SELECT
092F 08 26 003F R                OR      @MOTOR_STATUS,AH              ; OR IN DRIVE SELECTED AND MOTOR ON
0933 A0 003F R                   MOV     AL,@MOTOR_STATUS              ; GET DIGITAL OUTPUT REGISTER REFLECTION
0936 8A D8                       MOV     BL,AL                         ; BL=@MOTOR_STATUS AFTER, BH=BEFORE
0938 80 E3 0F                    AND     BL,00001111B                  ; KEEP ONLY MOTOR BITS
093B FB                          STI                                   ; ENABLE INTERRUPTS AGAIN
093C 24 3F                       AND     AL,00111111B                  ; STRIP AWAY UNWANTED BITS
093E C0 C0 04                    ROL     AL,4                          ; PUT BITS IN DESIRED POSITIONS
0941 0C 0C                       OR      AL,00001100B                  ; NO RESET, ENABLE DMA/INTERRUPT
0943 BA 03F2                     MOV     DX,03F2H                      ; SELECT DRIVE AND TURN ON MOTOR
0946 EE                          OUT     DX,AL                         ; "
0947 3A DF                       CMP     BL,BH                         ; NEW MOTOR TURNED ON ?
0949 74 02                       JZ      NO_MOT_WAIT                   ; NO WAIT REQUIRED IF JUST SELECT
094B F8                          CLC                                   ; SET CARRY MEANING WAIT
094C C3                          RET

094D                    NO_MOT_WAIT:
094D F9                          STC                                   ; SET NO WAIT REQUIRED
094E FB                          STI                                   ; INTERRUPTS BACK ON
094F C3                          RET
0950                    TURN_ON ENDP
                        ;-------------------------------------------------------------
                        ; HD_WAIT                                                    :
                        ;        WAIT FOR HEAD SETTLE TIME.                          :
                        ;                                                            :
                        ; ON ENTRY:      DI : DRIVE #                                :
                        ;                                                            :
                        ; ON EXIT:       AX,BX,CX,DX DESTROYED                       :
                        ;-------------------------------------------------------------
0950                    HD_WAIT          PROC    NEAR
0950 B2 09                       MOV     DL,9                          ; GET HEAD SETTLE PARAMETER
0952 E8 08A1 R                   CALL    GET_PARM                      ; "
0955 F6 06 003F R 80             TEST    @MOTOR_STATUS,10000000B       ; SEE IF A WRITE OPERATION
095A 74 14                       JZ      ISNT_WRITE                    ; IF NOT, DO NOT ENFORCE ANY VALUES
095C 0A E4                       OR      AH,AH                         ; CHECK FOR ANY WAIT?
095E 75 14                       JNZ     DO_WAT                        ; IF THERE DO NOT ENFORCE
0960 B4 0F                       MOV     AH,HD12_SETTLE                ; LOAD 1.2M HEAD SETTLE MINIMUM
0962 8A 85 0090 R                MOV     AL,@DSK_STATE[DI]             ; LOAD STATE
0966 24 C0                       AND     AL,RATE_MSK                   ; KEEP ONLY RATE
0968 3C 80                       CMP     AL,RATE_250                   ; 1.2 M DRIVE ?
096A 75 08                       JNZ     DO_WAT                        ; DEFAULT HEAD SETTLE LOADED

096C B4 14              GP3:     MOV     AH,HD320_SETTLE               ; USE 320/360 HEAD SETTLE
096E EB 04                       JMP     SHORT DO_WAT                  ; "

0970                    ISNT_WRITE:
0970 0A E4                       OR      AH,AH                         ; CHECK FOR NO WAIT
0972 74 1F                       JZ      HW_DONE                       ; IF NOT WRITE AND 0 ITS OK

                        ;-----  AH CONTAINS NUMBER OF MILLISECONDS TO WAIT

0974                    DO_WAT:
0974 8A C4                       MOV     AL,AH                         ; AL = # MILLISECONDS
0976 32 E4                       XOR     AH,AH                         ; AX = # MILLISECONDS
0978 50                          PUSH    AX                            ; SAVE HEAD SETTLE PARAMETER
0979 BA 03E8                     MOV     DX,1000                       ; SET UP FOR MULTIPLY TO MICROSECONDS
097C F7 E2                       MUL     DX                            ; DX,AX = # MICROSECONDS
097E 8B CA                       MOV     CX,DX                         ; CX,AX = # MICROSECONDS
0980 8B D0                       MOV     DX,AX                         ; CX,DX = # MICROSECONDS
0982 B4 86                       MOV     AH,86H                        ; LOAD WAIT CODE
0984 CD 15                       INT     15H                           ; PERFORM WAIT
0986 58                          POP     AX                            ; RESTORE HEAD SETTLE PARAMETER
0987 73 0A                       JNC     HW_DONE                       ; CHECK FOR EVENT WAIT ACTIVE

0989                    J29:                                           ;       1 MILLISECOND LOOP
0989 B9 0042                     MOV     CX,66                         ; COUNT AT 15.085737 US PER COUNT
098C E8 0000 E                   CALL    WAITF                         ; DELAY FOR 1 MILLISECOND
098F FE C8                       DEC     AL                            ; DECREMENT THE COUNT
0991 75 F6                       JNZ     J29                           ; DO AL MILLISECOND # OF TIMES
0993                    HW_DONE:
0993 C3                          RET
0994                    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                                  :
                        ;-------------------------------------------------------------
0994                    NEC_OUTPUT       PROC    NEAR
0994 53                          PUSH    BX                            ; SAVE REG.
0995 BA 03F4                     MOV     DX,03F4H                      ; STATUS PORT
```

SECTION 5

```
0998 B3 02                      MOV     BL,2                    ; HIGH ORDER COUNTER
099A 33 C9                      XOR     CX,CX                   ; COUNT FOR TIME OUT

099C EC                 J23:    IN      AL,DX                   ; GET STATUS
099D 24 C0                      AND     AL,11000000B            ; KEEP STATUS AND DIRECTION
099F 3C 80                      CMP     AL,10000000B            ; STATUS 1 AND DIRECTION 0 ?
09A1 74 0F                      JZ      J27                     ; STATUS AND DIRECTION OK
09A3 E2 F7                      LOOP    J23                     ; CONTINUE TILL CX EXHAUSTED

09A5 FE CB                      DEC     BL                      ; DECREMENT COUNTER
09A7 75 F3                      JNZ     J23                     ; REPEAT TILL DELAY FINISHED, CX = 0

                        ;-----  FALL THRU TO ERROR RETURN

09A9 80 0E 0041 R 80            OR      @DISKETTE_STATUS,TIME_OUT
09AE 5B                         POP     BX                      ; RESTORE REG.
09AF 58                         POP     AX                      ; DISCARD THE RETURN ADDRESS
09B0 F9                         STC                             ; INDICATE ERROR TO CALLER
09B1 C3                         RET

                        ;-----  DIRECTION AND STATUS OK; OUTPUT BYTE

09B2                    J27:
09B2 8A C4                      MOV     AL,AH                   ; GET BYTE TO OUTPUT
09B4 42                         INC     DX                      ; DATA PORT = STATUS PORT + 1
09B5 EE                         OUT     DX,AL                   ; OUTPUT THE BYTE

09B6 9C                         PUSHF                           ; SAVE FLAGS
09B7 B9 0003                    MOV     CX,3                    ; 30 TO 45 MICROSECOND WAIT FOR
09BA E8 0000 E                  CALL    WAITF                   ; NEC FLAGS UPDATE CYCLE
09BD 9D                         POPF                            ; RESTORE FLAGS FOR EXIT
09BE 5B                         POP     BX                      ; RESTORE REG.
09BF C3                         RET                             ; CY = 0 FROM TEST INSTRUCTION
09C0                    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:      @DISKETTE_STATUS, CY REFLECT STATUS OF OPERATION.:
                        ;               AX,BX,CX,DX DESTROYED                           :
                        ;---------------------------------------------------------------
09C0                    SEEK    PROC    NEAR
09C0 8B DF                      MOV     BX,DI                   ; BX = DRIVE #
09C2 B0 01                      MOV     AL,1                    ; ESTABLISH MASK FOR RECALIBRATE TEST
09C4 86 CB                      XCHG    CL,BL                   ; GET DRIVE VALUE INTO CL
09C6 D2 C0                      ROL     AL,CL                   ; SHIFT MASK BY THE DRIVE VALUE
09C8 86 CB                      XCHG    CL,BL                   ; RECOVER TRACK VALUE
09CA 84 06 003E R               TEST    AL,@SEEK_STATUS         ; TEST FOR RECALIBRATE REQUIRED
09CE 75 1C                      JNZ     J28A                    ; JUMP IF RECALIBRATE NOT REQUIRED

09D0 08 06 003E R               OR      @SEEK_STATUS,AL         ; TURN ON THE NO RECALIBRATE BIT IN FLAG
09D4 E8 0A1F R                  CALL    RECAL                   ; RECALIBRATE DRIVE
09D7 73 0A                      JNC     AFT_RECAL               ; RECALIBRATE DONE

                        ;-----  ISSUE RECALIBRATE FOR 80 TRACK DISKETTES

09D9 C6 06 0041 R 00            MOV     @DISKETTE_STATUS,0      ; CLEAR OUT INVALID STATUS
09DE E8 0A1F R                  CALL    RECAL                   ; RECALIBRATE DRIVE
09E1 72 3B                      JC      RB                      ; IF RECALIBRATE FAILS TWICE THEN ERROR

09E3                    AFT_RECAL:
09E3 C6 85 0094 R 00            MOV     @DSK_TRK[DI],0          ; SAVE NEW CYLINDER AS PRESENT POSITION
09E8 0A ED                      OR      CH,CH                   ; CHECK FOR SEEK TO TRACK 0
09EA 74 2D                      JZ      DO_WAIT                 ; HEAD SETTLE, CY = 0 IF JUMP

                        ;-----  DRIVE IS IN SYNCHRONIZATION WITH CONTROLLER, SEEK TO TRACK

09EC F6 85 0090 R 20    J28A:   TEST    @DSK_STATE[DI],DBL_STEP ; CHECK FOR DOUBLE STEP REQUIRED
09F1 74 02                      JZ      R7                      ; SINGLE STEP REQUIRED BYPASS DOUBLE
09F3 D0 E5                      SHL     CH,1                    ; DOUBLE NUMBER OF STEP TO TAKE

09F5 3A AD 0094 R       R7:     CMP     CH,@DSK_TRK[DI]         ; SEE IF ALREADY AT THE DESIRED TRACK
09F9 74 23                      JE      RB                      ; IF YES, DO NOT NEED TO SEEK

09FB BA 0A1E R                  MOV     DX,OFFSET NEC_ERR       ; LOAD RETURN ADDRESS
09FE 52                         PUSH    DX                      ; ON STACK FOR NEC_OUTPUT ERROR
09FF 88 AD 0094 R               MOV     @DSK_TRK[DI],CH         ; SAVE NEW CYLINDER AS PRESENT POSITION
0A03 B4 0F                      MOV     AH,0FH                  ; SEEK COMMAND TO NEC
0A05 E8 0994 R                  CALL    NEC_OUTPUT
0A08 8B DF                      MOV     BX,DI                   ; BX = DRIVE #
0A0A 8A E3                      MOV     AH,BL                   ; OUTPUT DRIVE NUMBER
0A0C E8 0994 R                  CALL    NEC_OUTPUT
0A0F 8A A5 0094 R               MOV     AH,@DSK_TRK[DI]         ; GET CYLINDER NUMBER
0A13 E8 0994 R                  CALL    NEC_OUTPUT
0A16 E8 0A36 R                  CALL    CHK_STAT_2              ; ENDING INTERRUPT AND SENSE STATUS

                        ;-----  WAIT FOR HEAD SETTLE

0A19                    DO_WAIT:
0A19 9C                         PUSHF                           ; SAVE STATUS
0A1A E8 0950 R                  CALL    HD_WAIT                 ; WAIT FOR HEAD SETTLE TIME
0A1D 9D                         POPF                            ; RESTORE STATUS
0A1E                    RB:
0A1E                    NEC_ERR:
0A1E C3                         RET                             ; RETURN TO CALLER
0A1F                    SEEK    ENDP
                        ;---------------------------------------------------------------
                        ; RECAL                                                         :
                        ;       RECALIBRATE DRIVE                                       :
                        ;                                                               :
                        ; ON ENTRY      DI = DRIVE #                                    :
                        ;                                                               :
                        ; ON EXIT:      CY  REFLECTS STATUS OF OPERATION.               :
                        ;---------------------------------------------------------------
0A1F                    RECAL   PROC    NEAR
0A1F 51                         PUSH    CX
0A20 B8 0A34 R                  MOV     AX,OFFSET RC_BACK       ; LOAD NEC_OUTPUT ERROR
0A23 50                         PUSH    AX
```

```
0A24 B4 07                  MOV     AH,07H                  ; RECALIBRATE COMMAND
0A26 E8 0994 R              CALL    NEC_OUTPUT
0A29 8B DF                  MOV     BX,DI                   ; BX = DRIVE #
0A2B 8A E3                  MOV     AH,BL
0A2D E8 0994 R              CALL    NEC_OUTPUT              ; OUTPUT THE DRIVE NUMBER
0A30 E8 0A36 R              CALL    CHK_STAT_2              ; GET THE INTERRUPT AND SENSE INT STATUS
0A33 58                     POP     AX                      ; THROW AWAY ERROR
0A34                RC_BACK:
0A34 59                     POP     CX
0A35 C3                     RET
0A36                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.:
                    ;---------------------------------------------------------------
0A36                CHK_STAT_2      PROC    NEAR
0A36 B8 0A54 R              MOV     AX,OFFSET CS_BACK       ; LOAD NEC_OUTPUT ERROR ADDRESS
0A39 50                     PUSH    AX
0A3A E8 0A5D R              CALL    WAIT_INT                ; WAIT FOR THE INTERRUPT
0A3D 72 14                  JC      J34                     ; IF ERROR, RETURN IT
0A3F B4 08                  MOV     AH,08H                  ; SENSE INTERRUPT STATUS COMMAND
0A41 E8 0994 R              CALL    NEC_OUTPUT
0A44 E8 0A85 R              CALL    RESULTS                 ; READ IN THE RESULTS
0A47 72 0A                  JC      J34
0A49 A0 0042 R              MOV     AL,@NEC_STATUS          ; GET THE FIRST STATUS BYTE
0A4C 24 60                  AND     AL,01100000B            ; ISOLATE THE BITS
0A4E 3C 60                  CMP     AL,01100000B            ; TEST FOR CORRECT VALUE
0A50 74 03                  JZ      J35                     ; IF ERROR, GO MARK IT
0A52 F8                     CLC                             ; GOOD RETURN
0A53                J34:
0A53 58                     POP     AX                      ; THROW AWAY ERROR RETURN
0A54                CS_BACK:
0A54 C3                     RET

0A55                J35:
0A55 80 0E 0041 R 40        OR      @DSKETTE_STATUS,BAD_SEEK
0A5A F9                     STC                             ; ERROR RETURN CODE
0A5B EB F6                  JMP     SHORT J34
0A5D                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.:
                    ;---------------------------------------------------------------
0A5D                WAIT_INT        PROC    NEAR
0A5D FB                     STI                             ; TURN ON INTERRUPTS, JUST IN CASE
0A5E F8                     CLC                             ; CLEAR TIMEOUT INDICATOR
0A5F B8 9001                MOV     AX,09001H               ; LOAD WAIT CODE AND TYPE
0A62 CD 15                  INT     15H                     ; PERFORM OTHER FUNCTION
0A64 72 11                  JC      J36A                    ; BYPASS TIMING LOOP IF TIMEOUT DONE
0A66 B3 04                  MOV     BL,4                    ; CLEAR THE COUNTERS
0A68 33 C9                  XOR     CX,CX                   ; FOR 2 SECOND WAIT
0A6A                J36:
0A6A F6 06 003E R 80        TEST    @SEEK_STATUS,INT_FLAG   ; TEST FOR INTERRUPT OCCURRING
0A6F 75 0C                  JNZ     J37
0A71 E2 F7                  LOOP    J36                     ; COUNT DOWN WHILE WAITING
0A73 FE CB                  DEC     BL                      ; SECOND LEVEL COUNTER
0A75 75 F3                  JNZ     J36

0A77 80 0E 0041 R 80 J36A:  OR      @DSKETTE_STATUS,TIME_OUT        ; NOTHING HAPPENED
0A7C F9                     STC                             ; ERROR RETURN
0A7D                J37:
0A7D 9C                     PUSHF                           ; SAVE CURRENT CARRY
0A7E 80 26 003E R 7F        AND     @SEEK_STATUS,NOT INT_FLAG       ; TURN OFF INTERRUPT FLAG
0A83 9D                     POPF                            ; RECOVER CARRY
0A84 C3                     RET                             ; GOOD RETURN CODE
0A85                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                          :
                    ;---------------------------------------------------------------
0A85                RESULTS PROC    NEAR
0A85 57                     PUSH    DI
0A86 BF 0042 R              MOV     DI,OFFSET @NEC_STATUS   ; POINTER TO DATA AREA
0A89 B3 07                  MOV     BL,7                    ; MAX STATUS BYTES
0A8B BA 03F4                MOV     DX,03F4H                ; STATUS PORT

                    ;-----  WAIT FOR REQUEST FOR MASTER

0A8E B7 02          R10:    MOV     BH,2                    ; HIGH ORDER COUNTER
0A90 33 C9                  XOR     CX,CX                   ; COUNTER
0A92                J39:                                    ; WAIT FOR MASTER
0A92 EC                     IN      AL,DX                   ; GET STATUS
0A93 24 C0                  AND     AL,11000000B            ; KEEP ONLY STATUS AND DIRECTION
0A95 3C C0                  CMP     AL,11000000B            ; STATUS 1 AND DIRECTION 1 ?
0A97 74 0E                  JZ      J42                     ; STATUS AND DIRECTION OK
0A99 E2 F7                  LOOP    J39                     ; LOOP TILL TIMEOUT

0A9B FE CF                  DEC     BH                      ; DECREMENT HIGH ORDER COUNTER
0A9D 75 F3                  JNZ     J39                     ; REPEAT TILL DELAY DONE

0A9F 80 0E 0041 R 80        OR      @DSKETTE_STATUS,TIME_OUT
0AA4 F9                     STC                             ; SET ERROR RETURN
0AA5 EB 1B                  JMP     SHORT POPRES            ; POP REGISTERS AND RETURN

                    ;-----  READ IN THE STATUS

0AA7                J42:
                    ;       JMP     $+2                     ; I/O DELAY
0AA7 42                     INC     DX                      ; POINT AT DATA PORT
0AA8 EC                     IN      AL,DX                   ; GET THE DATA
0AA9 88 05                  MOV     [DI],AL                 ; STORE THE BYTE
```

SECTION 5

```
0AAB 47                      INC     DI                      ; INCREMENT THE POINTER
0AAC B9 0003                 MOV     CX,3                    ; MINIMUM 24 MICROSECONDS FOR NEC
0AAF E8 0000 E               CALL    WAITF                   ; WAIT 30 TO 45 MICROSECONDS
0AB2 4A                      DEC     DX                      ; POINT AT STATUS PORT
0AB3 EC                      IN      AL,DX                   ; GET STATUS
0AB4 A8 10                   TEST    AL,00010000B            ; TEST FOR NEC STILL BUSY
0AB6 74 0A                   JZ      POPRES                  ; RESULTS DONE ?
0AB8 FE CB                   DEC     BL                      ; DECREMENT THE STATUS COUNTER
0ABA 75 D2                   JNZ     R10                     ; GO BACK FOR MORE
0ABC 80 0E 0041 R 20         OR      @DSKETTE_STATUS,BAD_NEC ; TOO MANY STATUS BYTES
0AC1 F9                      STC                             ; SET ERROR FLAG

                        ;-----  RESULT OPERATION IS DONE

0AC2                    POPRES:
0AC2 5F                      POP     DI
0AC3 C3                      RET                             ; RETURN WITH CARRY SET
0AC4                    RESULTS ENDP
                        ;----------------------------------------------------------
                        ; 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                         :
                        ;----------------------------------------------------------
0AC4                    READ_DSKCHNG    PROC    NEAR
0AC4 E8 08B6 R               CALL    MOTOR_ON                ; TURN ON THE MOTOR IF OFF
0AC7 BA 03F7                 MOV     DX,03F7H                ; ADDRESS DIGITAL INPUT REGISTER
0ACA EC                      IN      AL,DX                   ; INPUT DIGITAL INPUT REGISTER
0ACB A8 80                   TEST    AL,DSK_CHG              ; CHECK FOR DISK CHANGE LINE ACTIVE
0ACD C3                      RET                             ; RETURN TO CALLER WITH ZERO FLAG SET
0ACE                    READ_DSKCHNG    ENDP
                        ;----------------------------------------------------------
                        ; DRIVE_DET                                                :
                        ;       DETERMINES WHETHER DRIVE IS 80 OR 40 TRACKS AND    :
                        ;       UPDATES STATE INFORMATION ACCORDINGLY.             :
                        ;                                                          :
                        ; ON ENTRY:     DI = DRIVE #                               :
                        ;----------------------------------------------------------
0ACE                    DRIVE_DET       PROC    NEAR
0ACE E8 08B6 R               CALL    MOTOR_ON                ; TURN ON MOTOR IF NOT ALREADY ON
0AD1 E8 0A1F R               CALL    RECAL                   ; RECALIBRATE DRIVE
0AD4 72 3C                   JC      DD_BAC                  ; ASSUME NO DRIVE PRESENT
0AD6 B5 30                   MOV     CH,TRK_SLAP             ; SEEK TO TRACK 48
0AD8 E8 09C0 R               CALL    SEEK                    ; "
0ADB 72 35                   JC      DD_BAC                  ; ERROR NO DRIVE
0ADD B5 0B                   MOV     CH,QUIET_SEEK+1         ; SEEK TO TRACK 10
0ADF                    SK_GIN:
0ADF FE CD                   DEC     CH                      ; DECREMENT TO NEXT TRACK
0AE1 51                      PUSH    CX                      ; SAVE TRACK
0AE2 E8 09C0 R               CALL    SEEK                    ; "
0AE5 72 2C                   JC      POP_BAC                 ; POP AND RETURN
0AE7 B8 0B13 R               MOV     AX,OFFSET POP_BAC       ; LOAD NEC OUTPUT ERROR ADDRESS
0AEA 50                      PUSH    AX                      ; "
0AEB B4 04                   MOV     AH,SENSE_DRV_ST         ; SENSE DRIVE STATUS COMMAND BYTE
0AED E8 0994 R               CALL    NEC_OUTPUT              ; OUTPUT TO NEC
0AF0 8B C7                   MOV     AX,DI                   ; AL = DRIVE
0AF2 8A E0                   MOV     AH,AL                   ; AH = DRIVE
0AF4 E8 0994 R               CALL    NEC_OUTPUT              ; OUTPUT TO NEC
0AF7 E8 0A85 R               CALL    RESULTS                 ; GO GET STATUS
0AFA 58                      POP     AX                      ; THROW AWAY ERROR ADDRESS
0AFB 59                      POP     CX                      ; RESTORE TRACK
0AFC F6 06 0042 R 10         TEST    @NEC_STATUS,HOME        ; TRACK 0 ?
0B01 74 DC                   JZ      SK_GIN                  ; GO TILL TRACK 0
0B03 0A ED                   OR      CH,CH                   ; IS HOME AT TRACK 0 ?
0B05 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.

0B07 80 8D 0090 R 94         OR      @DSK_STATE[DI],DRV_DET+MED_DET+RATE_250
0B0C C3                      RET                             ; ALL INFORMATION SET

0B0D                    IS_80:
0B0D 80 8D 0090 R 01         OR      @DSK_STATE[DI],TRK_CAPA ; SETUP 80 TRACK CAPABILITY
0B12                    DD_BAC:
0B12 C3                      RET

0B13                    POP_BAC:
0B13 59                      POP     CX                      ; THROW AWAY
0B14 C3                      RET

0B15                    DRIVE_DET       ENDP
                        ;----------------------------------------------------------
                        ; DISK_INT                                                 :
                        ;       THIS ROUTINE HANDLES THE DISKETTE INTERRUPT.       :
                        ;                                                          :
                        ; ON EXIT:      THE INTERRUPT FLAG IS SET IN @SEEK_STATUS. :
                        ;----------------------------------------------------------
0B15                    DISK_INT_1      PROC    FAR          ; ENTRY POINT FOR ORG 0EF57H
0B15 50                      PUSH    AX                      ; SAVE WORK REGISTER
0B16 1E                      PUSH    DS                      ; SAVE REGISTERS
0B17 E8 0000 E               CALL    DDS                     ; SETUP DATA ADDRESSING
0B1A 80 0E 003E R 80         OR      @SEEK_STATUS,INT_FLAG   ; TURN ON INTERRUPT OCCURRED
0B1F 1F                      POP     DS                      ; RESTORE USER (DS)
0B20 B0 20                   MOV     AL,EOI                  ; END OF INTERRUPT MARKER
0B22 E6 20                   OUT     INTA00,AL               ; INTERRUPT CONTROL PORT
0B24 FB                      STI                             ; RE-ENABLE INTERRUPTS
0B25 B8 9101                 MOV     AX,09101H               ; INTERRUPT POST CODE AND TYPE
0B28 CD 15                   INT     15H                     ; GO PERFORM OTHER TASK
0B2A 58                      POP     AX                      ; RECOVER REGISTER
0B2B CF                      IRET                            ; RETURN FROM INTERRUPT
0B2C                    DISK_INT_1      ENDP
                        ;----------------------------------------------------------
                        ; DSKETTE_SETUP                                            :
                        ;       THIS ROUTINE DOES A PRELIMINARY CHECK TO SEE WHAT TYPE :
                        ;       OF DISKETTE DRIVES ARE ATTACH TO THE SYSTEM.       :
                        ;----------------------------------------------------------
```

```
0B2C                        DSKETTE_SETUP PROC      NEAR
0B2C 50                             PUSH    AX                      ; SAVE REGISTERS
0B2D 53                             PUSH    BX
0B2E 51                             PUSH    CX
0B2F 52                             PUSH    DX
0B30 57                             PUSH    DI
0B31 1E                             PUSH    DS
0B32 E8 0000 E                      CALL    DDS                     ; POINT DATA SEGMENT TO BIOS DATA AREA
0B35 80 0E 00A0 R 01                OR      @RTC_WAIT_FLAG,01       ; NO RTC WAIT, FORCE USE OF LOOP
0B3A 33 FF                          XOR     DI,DI                   ; INITIALIZE DRIVE POINTER
0B3C C7 06 0090 R 0000              MOV     WORD PTR @DSK_STATE,0   ; INITIALIZE STATES
0B42 80 26 008B R 33                AND     @LASTRATE,NOT STRT_MSK+SEND_MSK  ; CLEAR START & SEND
0B47 80 0E 008B R C0                OR      @LASTRATE,SEND_MSK      ; INITIALIZE SENT TO IMPOSSIBLE
0B4C C6 06 003E R 00                MOV     @SEEK_STATUS,0          ; INDICATE RECALIBRATE NEEDED
0B51 C6 06 0040 R 00                MOV     @MOTOR_COUNT,0          ; INITIALIZE MOTOR COUNT
0B56 C6 06 003F R 00                MOV     @MOTOR_STATUS,0         ; INITIALIZE DRIVES TO OFF STATE
0B5B C6 06 0041 R 00                MOV     @DSKETTE_STATUS,0       ; NO ERRORS

0B60                        SUP0:
0B60 E8 0ACE R                      CALL    DRIVE_DET               ; DETERMINE DRIVE
0B63 E8 0429 R                      CALL    XLAT_OLD                ; TRANSLATE STATE TO COMPATIBLE MODE
0B66 47                             INC     DI                      ; POINT TO NEXT DRIVE
0B67 83 FF 02                       CMP     DI,MAX_DRV              ; SEE IF DONE
0B6A 75 F4                          JNZ     SUP0                    ; REPEAT FOR EACH DRIVE
0B6C C6 06 003E R 00                MOV     @SEEK_STATUS,0          ; FORCE RECALIBRATE
0B71 80 26 00A0 R FE                AND     @RTC_WAIT_FLAG,0FEH     ; ALLOW FOR RTC WAIT
0B76 E8 07F5 R                      CALL    SETUP_END               ; VARIOUS CLEANUPS
0B79 1F                             POP     DS                      ; RESTORE CALLERS RESISTERS
0B7A 5F                             POP     DI
0B7B 5A                             POP     DX
0B7C 59                             POP     CX
0B7D 5B                             POP     BX
0B7E 58                             POP     AX
0B7F C3                             RET
0B80                        DSKETTE_SETUP ENDP
0B80                        CODE    ENDS
                                    END
```

```
                PAGE 118,123
                TITLE DISK ----- 11/15/85  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 PAIR 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
 = 0005         BAD_RESET       EQU     05H             ; 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
```

```
                                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 @HF1_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
```

SECTION 5

```
                              ;----- 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
```

```
                                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:                                     ;       SAVE REGISTERS DURING OPERATION
 01D1 C8 0008 00                        ENTER    8,0                    ; SAVE (BP) AND MAKE ROOM FOR @CMD_BLOCK
 01D5 53                                PUSH     BX                     ;  IN THE STACK. THE COMMAND BLOCK IS:
 01D6 51                                PUSH     CX                     ;   @CMD_BLOCK == BYTE PTR [BP]-8
 01D7 52                                PUSH     DX
 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 0225 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                           ; ADJUST (SP) AND RESTORE (BP)
 01F8 CA 0002                           RET      2                      ; THROW AWAY SAVED FLAGS
 01FB                           DISK_IO ENDP

 01FB                           M1      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                         M1L     EQU      $-M1

 0225                           DISK_IO_CONT     PROC     NEAR
 0225 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
```

SECTION 5

```
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,M1L                   ; 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 M1]
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     COMMANDI
 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
```

SECTION 5

```
                                  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 80 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      COMMANDI
 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      COMMANDO
 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 0630 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

                                ;----------------------------------------
                                ; COMMANDI                              :
                                ;       REPEATEDLY INPUTS DATA TILL     :
                                ;       NSECTOR RETURNS ZERO            :
                                ;----------------------------------------
04C6                            COMMANDI:
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_I1:
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_I2: 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_I2
04FA E8 0630 R                  CMD_I3: 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_I1
0504                            CMD_ABORT:
0504                            TM_OUT:
0504 C3                                 RET
```

SECTION 5

```
                         PAGE
                         ;----------------------------------------
                         ; COMMAND0                               :
                         ;       REPEATEDLY OUTPUTS DATA TILL     :
                         ;       NSECTOR RETURNS ZERO             :
                         ;----------------------------------------
0505                     COMMAND0:
0505 E8 06A1 R                   CALL    CHECK_DMA              ; CHECK 64K BOUNDARY ERROR
0508 72 FA                       JC      CMD_ABORT
050A 8B F3               CMD_OF: MOV     SI,BX
050C E8 055C R                   CALL    COMMAND                ; OUTPUT COMMAND
050F 75 F3                       JNZ     CMD_ABORT
0511 E8 061A R                   CALL    WAIT_DRQ               ; WAIT FOR DATA REQUEST
0514 72 EE                       JC      TM_OUT                 ; TOO LONG
0516 1E                  CMD_01: PUSH    DS
0517 06                          PUSH    ES                     ; MOVE ES TO DS
0518 1F                          POP     DS
0519 B9 0100                     MOV     CX,256D                ; PUT THE DATA OUT TO THE CARD
051C BA 01F0                     MOV     DX,HF_PORT
051F FA                          CLI
0520 FC                          CLD
0521 F3/ 6F                      REP     OUTSW
0523 FB                          STI
0524 1F                          POP     DS                     ; RESTORE DS
0525 F6 46 FE 02                 TEST    @CMD_BLOCK+6,ECC_MODE  ; CHECK FOR NORMAL OUTPUT
0529 74 12                       JZ      CMD_03
052B E8 061A R                   CALL    WAIT_DRQ               ; WAIT FOR DATA REQUEST
052E 72 D4                       JC      TM_OUT
0530 BA 01F0                     MOV     DX,HF_PORT
0533 B9 0004                     MOV     CX,4                   ; OUTPUT THE ECC BYTES
0536 26: 8A 04           CMD_02: MOV     AL,ES:BYTE PTR [SI]
0539 EE                          OUT     DX,AL
053A 46                          INC     SI
053B E2 F9                       LOOP    CMD_02
053D                     CMD_03:
053D E8 05C2 R                   CALL    WAIT                   ; WAIT FOR SECTOR COMPLETE INTERRUPT
0540 75 C2                       JNZ     TM_OUT                 ; ERROR RETURNED
0542 E8 0630 R                   CALL    CHECK_STATUS
0545 75 BD                       JNZ     CMD_ABORT
0547 F6 06 008C R 08             TEST    @HF_STATUS,ST_DRQ      ; CHECK FOR MORE
054C 75 C8                       JNZ     SHORT CMD_01
054E BA 01F2                     MOV     DX,HF_PORT+2           ; CHECK RESIDUAL SECTOR COUNT
0551 EC                          IN      AL,DX                  ;
0552 A8 FF                       TEST    AL,0FFH                ;
0554 74 05                       JZ      CMD_04                 ; COUNT = 0  OK
0556 C6 06 0074 R BB             MOV     @DISK_STATUS1,UNDEF_ERR ; OPERATION ABORTED - PARTIAL TRANSFER
055B                     CMD_04:
055B C3                          RET

                         ;----------------------------------------------------
                         ; COMMAND                                            :
                         ;       THIS ROUTINE OUTPUTS THE COMMAND BLOCK       :
                         ; OUTPUT                                             :
                         ;       BL = STATUS                                  :
                         ;       BH = ERROR REGISTER                          :
                         ;----------------------------------------------------

055C                     COMMAND PROC    NEAR
055C 53                          PUSH    BX                     ; WAIT FOR SEEK COMPLETE AND READY
055D B9 0600                     MOV     CX,DELAY_2             ; SET INITIAL DELAY BEFORE TEST
0560                     COMMAND1:
0560 51                          PUSH    CX                     ; SAVE LOOP COUNT
0561 E8 044F R                   CALL    TST_RDY                ; CHECK DRIVE READY
0564 59                          POP     CX
0565 74 0B                       JZ      COMMAND2               ; DRIVE IS READY
0567 80 3E 0074 R 80             CMP     @DISK_STATUS1,TIME_OUT ; TST_RDY TIMED OUT--GIVE UP
056C 74 48                       JZ      CMD_TIMEOUT
056E E2 F0                       LOOP    COMMAND1               ; KEEP TRYING FOR A WHILE
0570 EB 49                       JMP     SHORT COMMAND4         ; ITS NOT GOING TO GET READY
0572                     COMMAND2:
0572 5B                          POP     BX
0573 57                          PUSH    DI
0574 C6 06 008E R 00             MOV     @HF_INT_FLAG,0         ; RESET INTERRUPT FLAG
0579 FA                          CLI                            ; INHIBIT INTERRUPTS WHILE CHANGING MASK
057A E4 A1                       IN      AL,INTB01              ; TURN ON SECOND INTERRUPT CHIP
057C 24 BF                       AND     AL,0BFH
057E EB 00                       JMP     $+2
0580 E6 A1                       OUT     INTB01,AL
0582 E4 21                       IN      AL,INTA01              ; LET INTERRUPTS PASS THRU TO
0584 24 FB                       AND     AL,0FBH                ;  SECOND CHIP
0586 EB 00                       JMP     $+2
0588 E6 21                       OUT     INTA01,AL
058A FB                          STI
058B 33 FF                       XOR     DI,DI                  ; INDEX THE COMMAND TABLE
058D BA 01F1                     MOV     DX,HF_PORT+1           ; DISK ADDRESS
0590 F6 06 0076 R C0             TEST    @CONTROL_BYTE,0C0H     ; CHECK FOR RETRY SUPPRESSION
0595 74 11                       JZ      COMMAND3
0597 8A 46 FE                    MOV     AL,@CMD_BLOCK+6        ; YES-GET OPERATION CODE
059A 24 F0                       AND     AL,0F0H                ; GET RID OF MODIFIERS
059C 3C 20                       CMP     AL,20H                 ; 20H-40H IS READ, WRITE, VERIFY
059E 72 08                       JB      COMMAND3
05A0 3C 40                       CMP     AL,40H
05A2 77 04                       JA      COMMAND3
05A4 80 4E FE 01                 OR      @CMD_BLOCK+6,NO_RETRIES ; VALID OPERATION FOR RETRY SUPPRESS
05A8                     COMMAND3:
05A8 8A 43 F8                    MOV     AL,[@CMD_BLOCK+DI]     ; GET THE COMMAND STRING BYTE
05AB EE                          OUT     DX,AL                  ; GIVE IT TO CONTROLLER
05AC 47                          INC     DI                     ; NEXT BYTE IN COMMAND BLOCK
05AD 42                          INC     DX                     ; NEXT DISK ADAPTER REGISTER
05AE 81 FA 01F8                  CMP     DX,HF_PORT+8           ; ALL DONE?
05B2 75 F4                       JNZ     COMMAND3               ; NO--GO DO NEXT ONE
05B4 5F                          POP     DI
05B5 C3                          RET                            ; ZERO FLAG IS SET
05B6                     CMD_TIMEOUT:
05B6 C6 06 0074 R 20             MOV     @DISK_STATUS1,BAD_CNTLR
05BB                     COMMAND4:
05BB 5B                          POP     BX
05BC 80 3E 0074 R 00             CMP     @DISK_STATUS1,0        ; SET CONDITION CODE FOR CALLER
05C1 C3                          RET
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
```

SECTION 5

```
                                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                  DB      '11/15/85'              ; RELEASE MARKER
     38 35
06FD                            CODE    ENDS
                                        END
```

```
                                 PAGE 118,123
                                 TITLE KYBD ----- 11/15/85  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   K9:BYTE
                                          EXTRN   K10:BYTE
                                          EXTRN   K11:BYTE
                                          EXTRN   K12:BYTE
                                 ;        EXTRN   K13: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)= 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         :
                                 ;        (AH)= 11H  EXTENDED ASCII STATUS FOR THE ENHANCED KEYBOARD,          :
                                 ;                   OTHERWISE SAME AS FUNCTION AH=2                           :
                                 ;        (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     KIO1                    ;
 001C E9 00A4 R                           JMP     K500                    ; KEYBOARD WRITE
 001F 80 EC 0B                   KIO1:    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    KIO_E_XLAT      ; ROUTINE TO XLATE FOR EXTENDED CALLS
 0036 EB F4                               JMP     KIO_EXIT        ; GIVE IT TO THE CALLER

 0038 E8 00C7 R                  K1:      CALL    K1S             ; GET A CHARACTER FROM THE BUFFER
 003B E8 0130 R                           CALL    KIO_S_XLAT      ; ROUTINE TO XLATE FOR STANDARD CALLS
 003E 72 F8                               JC      K1              ; CARRY SET MEANS THROW CODE AWAY
 0040 EB EA                      K1A:     JMP     KIO_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    KIO_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    KIO_S_XLAT      ; ROUTINE TO XLATE FOR STANDARD CALLS
```

SECTION 5

```
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     KIO_EXIT         ; RETURN TO CALLER

                        ;------ SET TYPAMATIC RATE AND DELAY

0083 3C 05              K300:   CMP     AL,5             ; CORRECT FUNCTION CALL?
0085 75 A5                      JNE     KIO_EXIT         ;  NO, RETURN
0087 F6 C3 E0                   TEST    BL,0E0h          ; TEST FOR OUT-OF-RANGE RATE
008A 75 A0                      JNZ     KIO_EXIT         ; RETURN IF SO
008C F6 C7 FC                   TEST    BH,0FCh          ; TEST FOR OUT-OF-RANGE DELAY
008F 75 9B                      JNZ     KIO_EXIT         ; RETURN IF SO
0091 B0 F3                      MOV     AL,KB_TYPA_RD    ; COMMAND FOR TYPAMATIC RATE/DELAY
0093 E8 0644 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 0644 R                  CALL    SND_DATA         ; SEND TO KEYBOARD
00A2 EB 88                      JMP     KIO_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 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     KIO_EXIT         ; RETURN TO CALLER WITH STATUS IN AL

00C7                    KEYBOARD_IO_1   ENDP
                        ;------ 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 06D1 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 0693 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 06D1 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 0680 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              KIO_E_XLAT:
0125 3C F0                 CMP     AL,0F0h            ; IS IT ONE OF THE FILL-INs?
0127 75 06                 JNE     KIO_E_RET          ;  NO, PASS IT ON
0129 0A E4                 OR      AH,AH              ; AH = 0 IS SPECIAL CASE
012B 74 02                 JZ      KIO_E_RET          ;  PASS THIS ON UNCHANGED
012D 32 C0                 XOR     AL,AL              ;  OTHERWISE SET AL = 0
012F              KIO_E_RET:
012F C3                    RET                        ; GO BACK


                  ;------ ROUTINE TO TRANSLATE SCAN CODE PAIRS FOR STANDARD CALLS

0130              KIO_S_XLAT:
0130 80 FC E0              CMP     AH,0E0h            ; IS IT KEYPAD ENTER OR / ?
0133 75 12                 JNE     KIO_S2             ;  NO, CONTINUE
0135 3C 0D                 CMP     AL,0Dh             ; KEYPAD ENTER CODE?
0137 74 09                 JE      KIO_S1             ;  YES, MASSAGE A BIT
0139 3C 0A                 CMP     AL,0Ah             ; CTRL KEYPAD ENTER CODE?
013B 74 05                 JE      KIO_S1             ;  YES, MASSAGE THE SAME
013D B4 35                 MOV     AH,35h             ;  NO, MUST BE KEYPAD /
013F EB 23 90              JMP     KIO_USE            ; GIVE TO CALLER
0142 B4 1C        KIO_S1:  MOV     AH,1Ch             ; CONVERT TO COMPATIBLE OUTPUT
0144 EB 1E 90              JMP     KIO_USE            ; GIVE TO CALLER

0147 80 FC 84     KIO_S2:  CMP     AH,84h             ; IS IT ONE OF THE EXTENDED ONES?
014A 77 1A                 JA      KIO_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     KIO_S3             ;  NO, TRY LAST TEST
0150 0A E4                 OR      AH,AH              ; AH = 0 IS SPECIAL CASE
0152 74 10                 JZ      KIO_USE            ;  PASS THIS ON UNCHANGED
0154 EB 10 90              JMP     KIO_DIS            ;  THROW AWAY THE REST

0157 3C E0        KIO_S3:  CMP     AL,0E0h            ; IS IT AN EXTENSION OF A PREVIOUS ONE?
0159 75 09                 JNE     KIO_USE            ;  NO, MUST BE A STANDARD CODE
015B 0A E4                 OR      AH,AH              ; AH = 0 IS SPECIAL CASE
015D 74 05                 JZ      KIO_USE            ; JUMP IF AH = 0
015F 32 C0                 XOR     AL,AL              ; CONVERT TO COMPATIBLE OUTPUT
0161 EB 01 90              JMP     KIO_USE            ; PASS IT ON TO CALLER

0164              KIO_USE:
0164 F8                    CLC                        ; CLEAR CARRY TO INDICATE GOOD CODE
0165 C3                    RET                        ; RETURN
0166              KIO_DIS:
0166 F9                    STC                        ; SET CARRY TO INDICATE DISCARD CODE
0167 C3                    RET                        ; RETURN
                  ;----------------------------------------------------------------
                  ;        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 0635 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 0399 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 0399 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 0399 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 06D1 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 0680 R                  CALL    SND_LED                 ; GO TURN ON MODE INDICATORS
01CA 58                 UP0:    POP     AX                      ; RESTORE DATA IN
                        ;---------------------------------------------------------------------
                        ;               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 0626 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,ID_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 85                      CMP     AL,ID_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 0680 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 0399 R          ID_EX:  JMP     K26                     ; EXIT
0213                    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 039E 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 5E                      JNE     K16A                    ; NO, CONTINUE KEY PROCESSING
023B EB 42                      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 16                      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 2F                      JNE    K16B                    ; NO, THROW AWAY & RESET FLAG
0250 F6 C4 80                   TEST   AH,80H                  ; YES, IS IT THE BREAK OF THE KEY?
0253 75 2A                      JNZ    K16B                    ;  YES, THROW THIS AWAY, TOO
0255 E9 04D4 R                  JMP    K39P                    ;  NO, THIS IS THE REAL PAUSE STAT
                        ;------ TEST FOR SYSTEM KEY

0258                    T_SYS_KEY:
0258 3C 54                      CMP    AL,SYS_KEY              ; IS IT THE SYSTEM KEY?
025A 75 3D                      JNE    K16A                    ; CONTINUE IF NOT

025C F6 C4 80                   TEST   AH,080H                 ; CHECK IF THIS A BREAK CODE
025F 75 21                      JNZ    K16C                    ; DON'T TOUCH SYSTEM INDICATOR IF TRUE

0261 F6 06 0018 R 04            TEST   @KB_FLAG_1,SYS_SHIFT    ; SEE IF IN SYSTEM KEY HELD DOWN
0266 75 17                      JNZ    K16B                    ; IF YES, DON'T PROCESS SYSTEM INDICATOR

0268 80 0E 0018 R 04            OR     @KB_FLAG_1,SYS_SHIFT    ; INDICATE SYSTEM KEY DEPRESSED
026D B0 20                      MOV    AL,EOI                  ; END OF INTERRUPT COMMAND
026F E6 20                      OUT    020H,AL                 ; SEND COMMAND TO INTERRUPT CONTROL PORT
                                                               ; INTERRUPT-RETURN-NO-EOI
0271 B0 AE                      MOV    AL,ENA_KBD              ; INSURE KEYBOARD IS ENABLED
0273 E8 0635 R                  CALL   SHIP_IT                 ; EXECUTE ENABLE
0276 B8 8500                    MOV    AX,08500H               ; FUNCTION VALUE FOR MAKE OF SYSTEM KEY
0279 FB                         STI                            ; MAKE SURE INTERRUPTS ENABLED
027A CD 15                      INT    15H                     ; USER INTERRUPT
027C E9 03A8 R                  JMP    K27A                    ; END PROCESSING

027F E9 0399 R          K16B:   JMP    K26                     ; IGNORE SYSTEM KEY

0282 80 26 0018 R FB    K16C:   AND    @KB_FLAG_1,NOT SYS_SHIFT ;TURN OFF SHIFT KEY HELD DOWN
0287 B0 20                      MOV    AL,EOI                  ; END OF INTERRUPT COMMAND
0289 E6 20                      OUT    020H,AL                 ; SEND COMMAND TO INTERRUPT CONTROL PORT
                                                               ; INTERRUPT-RETURN-NO-EOI
028B B0 AE                      MOV    AL,ENA_KBD              ; INSURE KEYBOARD IS ENABLED
028D E8 0635 R                  CALL   SHIP_IT                 ; EXECUTE ENABLE
0290 B8 8501                    MOV    AX,08501H               ; FUNCTION VALUE FOR BREAK OF SYSTEM KEY
0293 FB                         STI                            ; MAKE SURE INTERRUPTS ENABLED
0294 CD 15                      INT    15H                     ; USER INTERRUPT
0296 E9 03A8 R                  JMP    K27A                    ; IGNORE SYSTEM KEY
                        ;------ TEST FOR SHIFT KEYS

0299 8A 1E 0017 R       K16A:   MOV    BL,@KB_FLAG             ; PUT STATE FLAGS IN BL
029D BF 0000 E                  MOV    DI,OFFSET K6            ; SHIFT KEY TABLE
02A0 B9 0000 E                  MOV    CX,OFFSET K6L           ; LENGTH
02A3 F2/ AE                     REPNE  SCASB                   ; LOOK THROUGH THE TABLE FOR A MATCH
02A5 8A C4                      MOV    AL,AH                   ; RECOVER SCAN CODE
02A7 74 03                      JE     K17                     ; JUMP IF MATCH FOUND
02A9 E9 0385 R                  JMP    K25                     ; IF NO MATCH, THEN SHIFT NOT FOUND

                        ;------ SHIFT KEY FOUND

02AC 81 EF 0001 E       K17:    SUB    DI,OFFSET K6+1          ; ADJUST PTR TO SCAN CODE MTCH
02B0 2E: 8A A5 0000 E           MOV    AH,CS:K7[DI]            ; GET MASK INTO AH
02B5 B1 02                      MOV    CL,2                    ; SET UP COUNT FOR FLAG SHIFTS
02B7 A8 80                      TEST   AL,80H                  ; TEST FOR BREAK KEY
02B9 74 03                      JZ     K17C
02BB EB 78 90                   JMP    K23                     ; JUMP IF BREAK

                        ;------ SHIFT MAKE FOUND, DETERMINE SET OR TOGGLE

02BE 80 FC 10           K17C:   CMP    AH,SCROLL_SHIFT
02C1 73 21                      JAE    K18                     ; IF SCROLL SHIFT OR ABOVE, TOGGLE KEY

                        ;------ PLAIN SHIFT KEY, SET SHIFT ON

02C3 08 26 0017 R               OR     @KB_FLAG,AH             ; TURN ON SHIFT BIT
02C7 F6 C4 0C                   TEST   AH,CTL_SHIFT+ALT_SHIFT  ; IS IT ALT OR CTRL?
02CA 75 03                      JNZ    K17D                    ;  YES, MORE FLAGS TO SET
02CC E9 0399 R                  JMP    K26                     ;  NO, INTERRUPT_RETURN
02CF F6 C7 02           K17D:   TEST   BH,LC_E0                ; IS THIS ONE OF THE NEW KEYS?
02D2 74 07                      JZ     K17E                    ;  NO, JUMP
02D4 08 26 0096 R               OR     @KB_FLAG_3,AH           ; SET BITS FOR RIGHT CTRL, ALT
02D8 E9 0399 R                  JMP    K26                     ; INTERRUPT_RETURN
02DB D2 EC              K17E:   SHR    AH,CL                   ; MOVE FLAG BITS TWO POSITIONS
02DD 08 26 0018 R               OR     @KB_FLAG_1,AH           ; SET BITS FOR LEFT CTRL, ALT
02E1 E9 0399 R                  JMP    K26                     ; INTERRUPT_RETURN

                        ;------ TOGGLED SHIFT KEY, TEST FOR 1ST MAKE OR NOT

02E4                    K18:                                   ; SHIFT-TOGGLE
02E4 F6 C3 04                   TEST   BL,CTL_SHIFT            ; CHECK CTL SHIFT STATE
02E7 74 03                      JZ     K18A                    ; JUMP IF NOT CTL STATE
02E9 E9 0385 R                  JMP    K25                     ; JUMP IF CTL STATE
02EC 3C 52              K18A:   CMP    AL,INS_KEY              ; CHECK FOR INSERT KEY
02EE 75 21                      JNE    K22                     ; JUMP IF NOT INSERT KEY
02F0 F6 C3 08                   TEST   BL,ALT_SHIFT            ; CHECK FOR ALTERNATE SHIFT
02F3 74 03                      JZ     K18B                    ; JUMP IF NOT ALTERNATE SHIFT
02F5 E9 0385 R                  JMP    K25                     ; JUMP IF ALTERNATE SHIFT
02F8 F6 C7 02           K18B:   TEST   BH,LC_E0                ; IS THIS THE NEW INSERT KEY?
02FB 75 14                      JNZ    K22                     ; YES, THIS ONE'S NEVER A "0"
02FD F6 C3 20           K19:    TEST   BL,NUM_STATE            ; CHECK FOR BASE STATE
0300 75 0A                      JNZ    K21                     ; JUMP IF NUM LOCK IS ON
0302 F6 C3 03                   TEST   BL,LEFT_SHIFT+RIGHT_SHIFT  ; TEST FOR SHIFT STATE
0305 74 0A                      JZ     K22                        ; JUMP IF BASE STATE
0307 8A E0              K20:    MOV    AH,AL                   ; PUT SCAN CODE BACK IN AH
0309 EB 7A 90                   JMP    K25                     ; NUMERAL "0", STNDRD. PROCESSING

030C F6 C3 03           K21:    TEST   BL,LEFT_SHIFT+RIGHT_SHIFT  ; MIGHT BE NUMERIC
030F 74 F6                      JZ     K20                        ; IS NUMERIC, STD. PROC.

0311                    K22:                                   ; SHIFT TOGGLE KEY HIT; PROCESS IT
0311 84 26 0018 R               TEST   AH,@KB_FLAG_1           ; IS KEY ALREADY DEPRESSED?
0315 74 03                      JZ     K22A
0317 E9 0399 R                  JMP    K26                     ; JUMP IF KEY ALREADY DEPRESSED
031A 08 26 0018 R       K22A:   OR     @KB_FLAG_1,AH           ; INDICATE THAT THE KEY IS DEPRESSED
031E 30 26 0017 R               XOR    @KB_FLAG,AH             ; TOGGLE THE SHIFT STATE

                        ;------ TOGGLE LED IF CAPS, NUM, OR SCROLL KEY DEPRESSED

0322 F6 C4 70                   TEST   AH,CAPS_SHIFT+NUM_SHIFT+SCROLL_SHIFT ; SHIFT TOGGLE?
0325 74 05                      JZ     K22B                    ; GO IF NOT
```

SECTION 5

```
0327 50                          PUSH    AX                      ; SAVE SCAN CODE AND SHIFT MASK
0328 E8 0680 R                   CALL    SND_LED                 ; GO TURN MODE INDICATORS ON
032B 58                          POP     AX                      ; RESTORE SCAN CODE

032C 3C 52              K22B:    CMP     AL,INS_KEY              ; TEST FOR 1ST MAKE OF INSERT KEY
032E 75 69                       JNE     K26                     ; JUMP IF NOT INSERT KEY
0330 8A E0                       MOV     AH,AL                   ; SCAN CODE IN BOTH HALVES OF AX
0332 EB 7F 90                    JMP     K28                     ; FLAGS UPDATED, PROC. FOR BUFFER

                        ;------ BREAK SHIFT FOUND

0335                    K23:                                     ; BREAK-SHIFT-FOUND
0335 80 FC 10                    CMP     AH,SCROLL_SHIFT         ; IS THIS A TOGGLE KEY?
0338 F6 D4                       NOT     AH                      ; INVERT MASK
033A 73 43                       JAE     K24                     ; YES, HANDLE BREAK TOGGLE
033C 20 26 0017 R                AND     @KB_FLAG,AH             ; TURN OFF SHIFT BIT
0340 80 FC FB                    CMP     AH,NOT CTL_SHIFT        ; IS THIS ALT OR CTL?
0343 77 26                       JA      K23D                    ;  NO, ALL DONE

0345 F6 C7 02                    TEST    BH,LC_E0                ; 2ND ALT OR CTL?
0348 74 06                       JZ      K23A                    ; NO, HANDLE NORMALLY
034A 20 26 0096 R                AND     @KB_FLAG_3,AH           ; RESET BIT FOR RIGHT ALT OR CTL
034E EB 06                       JMP     SHORT K23B              ; CONTINUE
0350 D2 FC              K23A:    SAR     AH,CL                   ; MOVE THE MASK BIT TWO POSITIONS
0352 20 26 0018 R                AND     @KB_FLAG_1,AH           ; RESET BIT FOR LEFT ALT OR CTL
0356 8A E0              K23B:    MOV     AH,AL                   ; SAVE SCAN CODE
0358 A0 0096 R                   MOV     AL,@KB_FLAG_3           ; GET RIGHT ALT & CTRL FLAGS
035B D2 E8                       SHR     AL,CL                   ; MOVE TO BITS 1 & 0
035D 0A 06 0018 R                OR      AL,@KB_FLAG_1           ; PUT IN LEFT ALT & CTL FLAGS
0361 D2 E0                       SHL     AL,CL                   ; MOVE BACK TO BITS 3 & 2
0363 24 0C                       AND     AL,ALT_SHIFT+CTL_SHIFT  ; FILTER OUT OTHER GARBAGE
0365 08 06 0017 R                OR      @KB_FLAG,AL             ; PUT RESULT IN THE REAL FLAGS
0369 8A C4                       MOV     AL,AH                   ; RECOVER SAVED SCAN CODE

036B 3C B8              K23D:    CMP     AL,ALT_KEY+80H          ; IS THIS ALTERNATE SHIFT RELEASE
036D 75 2A                       JNE     K26                     ; INTERRUPT_RETURN

                        ;------ ALTERNATE SHIFT KEY RELEASED, GET THE VALUE INTO BUFFER

036F A0 0019 R                   MOV     AL,@ALT_INPUT
0372 B4 00                       MOV     AH,0                    ; SCAN CODE OF 0
0374 88 26 0019 R                MOV     @ALT_INPUT,AH           ; ZERO OUT THE FIELD
0378 3C 00                       CMP     AL,0                    ; WAS THE INPUT = 0?
037A 74 1D                       JE      K26                     ; INTERRUPT_RETURN
037C E9 05FA R                   JMP     K61                     ; IT WASN'T, SO PUT IN BUFFER

037F                    K24:                                     ; BREAK-TOGGLE
037F 20 26 0018 R                AND     @KB_FLAG_1,AH           ; INDICATE NO LONGER DEPRESSED
0383 EB 14                       JMP     SHORT K26               ; INTERRUPT_RETURN

                        ;------ TEST FOR HOLD STATE
                                                                 ; AL, AH = SCAN CODE
0385                    K25:                                     ; NO-SHIFT-FOUND
0385 3C 80                       CMP     AL,80H                  ; TEST FOR BREAK KEY
0387 73 10                       JAE     K26                     ; NOTHING FOR BREAK CHARS FROM HERE ON
0389 F6 06 0018 R 08             TEST    @KB_FLAG_1,HOLD_STATE   ; ARE WE IN HOLD STATE
038E 74 23                       JZ      K28                     ; BRANCH AROUND TEST IF NOT
0390 3C 45                       CMP     AL,NUM_KEY
0392 74 05                       JE      K26                     ; CAN'T END HOLD ON NUM_LOCK
0394 80 26 0018 R F7             AND     @KB_FLAG_1,NOT HOLD_STATE  ; TURN OFF THE HOLD STATE BIT

0399                    K26:
0399 80 26 0096 R FC             AND     @KB_FLAG_3,NOT LC_E0+LC_E1 ; RESET LAST CHAR H.C. FLAG

039E                    K26A:                                    ; INTERRUPT-RETURN
039E FA                          CLI                             ; TURN OFF INTERRUPTS
039F B0 20                       MOV     AL,EOI                  ; END OF INTERRUPT COMMAND
03A1 E6 20                       OUT     020H,AL                 ; SEND COMMAND TO INTERRUPT CONTROL PORT

03A3                    K27:                                     ; INTERRUPT-RETURN-NO-EOI
03A3 B0 AE                       MOV     AL,ENA_KBD              ; INSURE KEYBOARD IS ENABLED
03A5 E8 0635 R                   CALL    SHIP_IT                 ; EXECUTE ENABLE

03A8 FA                 K27A:    CLI                             ; DISABLE INTERRUPTS
03A9 07                          POP     ES                      ; RESTORE REGISTERS
03AA 1F                          POP     DS                      ; •
03AB 5F                          POP     DI                      ; •
03AC 5E                          POP     SI                      ; •
03AD 5A                          POP     DX                      ; •
03AE 59                          POP     CX                      ; •
03AF 5B                          POP     BX                      ; •
03B0 58                          POP     AX                      ; •
03B1 5D                          POP     BP                      ; •
03B2 CF                          IRET                            ; RETURN, INTERRUPTS BACK ON WITH FLAG CHA
                   NGE
                        ;------ NOT IN HOLD STATE
                                                                 ; AL, AH = SCAN CODE (ALL MAKES)
03B3                    K28:                                     ; NO-HOLD-STATE
03B3 3C 58                       CMP     AL,88                   ; TEST FOR OUT-OF-RANGE SCAN CODES
03B5 77 E2                       JA      K26                     ; IGNORE IF OUT-OF-RANGE

03B7 F6 C3 08                    TEST    BL,ALT_SHIFT            ; ARE WE IN ALTERNATE SHIFT?
03BA 74 0C                       JZ      K28A                    ; JUMP IF NOT ALTERNATE

03BC F6 C7 10                    TEST    BH,KBX                  ; IS THIS THE ENHANCED KEYBOARD?
03BF 74 0A                       JZ      K29                     ; NO, ALT STATE IS REAL

03C1 F6 06 0018 R 04             TEST    @KB_FLAG_1,SYS_SHIFT    ; YES, IS SYSREQ KEY DOWN?
03C6 74 03                       JZ      K29                     ;  NO, ALT STATE IS REAL
03C8 E9 049C R          K28A:    JMP     K38                     ;  YES, THIS IS PHONY ALT STATE
                                                                 ;        DUE TO PRESSING SYSREQ

                        ;------ TEST FOR RESET KEY SEQUENCE (CTL ALT DEL)

03CB                    K29:                                     ; TEST-RESET
03CB F6 C3 04                    TEST    BL,CTL_SHIFT            ; ARE WE IN CONTROL SHIFT ALSO?
03CE 74 31                       JZ      K31                     ; NO_RESET
03D0 3C 53                       CMP     AL,DEL_KEY              ; SHIFT STATE IS THERE, TEST KEY
03D2 75 2D                       JNE     K31                     ; NO_RESET,  IGNORE

                        ;------ CTL-ALT-DEL HAS BEEN FOUND, DO I/O CLEANUP

03D4 C7 06 0072 R 1234           MOV     @RESET_FLAG,1234H       ; SET FLAG FOR RESET FUNCTION
03DA E9 0000 E                   JMP     START_1                 ; JUMP TO POWER ON DIAGNOSTICS
```

```
                               ;----------------------  TABLES FOR ALT CASE  -----------------------
                               ;------ ALT-INPUT-TABLE
03DD                           K30      LABEL   BYTE
03DD 52 4F 50 51 4B                     DB      82,79,80,81,75
03E2 4C 4D 47 48 49                     DB      76,77,71,72,73          ; 10 NUMBERS ON KEYPAD
                               ;------ SUPER-SHIFT-TABLE
03E7 10 11 12 13 14 15                  DB      16,17,18,19,20,21       ; A-Z TYPEWRITER CHARS
03ED 16 17 18 19 1E 1F                  DB      22,23,24,25,30,31
03F3 20 21 22 23 24 25                  DB      32,33,34,35,36,37
03F9 26 2C 2D 2E 2F 30                  DB      38,44,45,46,47,48
03FF 31 32                              DB      49,50

                               ;------ IN ALTERNATE SHIFT, RESET NOT FOUND

0401                           K31:                                     ; NO-RESET
0401 3C 39                              CMP     AL,57                   ; TEST FOR SPACE KEY
0403 75 05                              JNE     K311                    ; NOT THERE
0405 B0 20                              MOV     AL,' '                  ; SET SPACE CHAR
0407 E9 05EE R                          JMP     K57                     ; BUFFER_FILL
040A                           K311:
040A 3C 0F                              CMP     AL,15                   ; TEST FOR TAB KEY
040C 75 06                              JNE     K312                    ; NOT THERE
040E B8 A500                            MOV     AX,0A500h               ; SET SPECIAL CODE FOR ALT-TAB
0411 E9 05EE R                          JMP     K57                     ; BUFFER_FILL
0414                           K312:
0414 3C 4A                              CMP     AL,74                   ; TEST FOR KEYPAD -
0416 74 79                              JE      K37B                    ; GO PROCESS
0418 3C 4E                              CMP     AL,78                   ; TEST FOR KEYPAD +
041A 74 75                              JE      K37B                    ; GO PROCESS

                               ;------ LOOK FOR KEY PAD ENTRY

041C                           K32:                                     ; ALT-KEY-PAD
041C BF 03DD R                          MOV     DI,OFFSET K30           ; ALT-INPUT-TABLE
041F B9 000A                            MOV     CX,10                   ; LOOK FOR ENTRY USING KEYPAD
0422 F2/ AE                             REPNE   SCASB                   ; LOOK FOR MATCH
0424 75 18                              JNE     K33                     ; NO_ALT_KEYPAD
0426 F6 C7 02                           TEST    BH,LC_E0                ; IS THIS ONE OF THE NEW KEYS?
0429 75 6B                              JNZ     K37C                    ;  YES, JUMP, NOT NUMPAD KEY
042B 81 EF 03DE R                       SUB     DI,OFFSET K30+1         ; DI NOW HAS ENTRY VALUE
042F A0 0019 R                          MOV     AL,@ALT_INPUT           ; GET THE CURRENT BYTE
0432 B4 0A                              MOV     AH,10                   ; MULTIPLY BY 10
0434 F6 E4                              MUL     AH
0436 03 C7                              ADD     AX,DI                   ; ADD IN THE LATEST ENTRY
0438 A2 0019 R                          MOV     @ALT_INPUT,AL           ; STORE IT AWAY
043B E9 0399 R                 K32A:    JMP     K26                     ; THROW AWAY THAT KEYSTROKE

                               ;------ LOOK FOR SUPERSHIFT ENTRY

043E                           K33:                                     ; NO-ALT-KEYPAD
043E C6 06 0019 R 00                    MOV     @ALT_INPUT,0            ; ZERO ANY PREVIOUS ENTRY INTO INPUT
0443 B9 001A                            MOV     CX,26                   ; DI,ES ALREADY POINTING
0446 F2/ AE                             REPNE   SCASB                   ; LOOK FOR MATCH IN ALPHABET
0448 74 42                              JE      K37A                    ; MATCH FOUND, GO FILL THE BUFFER

                               ;------ LOOK FOR TOP ROW OF ALTERNATE SHIFT

044A                           K34:                                     ; ALT-TOP-ROW
044A 3C 02                              CMP     AL,2                    ; KEY WITH '1' ON IT
044C 72 43                              JB      K37B                    ; MUST BE ESCAPE
044E 3C 0D                              CMP     AL,13                   ; IS IT IN THE REGION
0450 77 05                              JA      K35                     ;  NO, ALT-SOMETHING ELSE
0452 80 C4 76                           ADD     AH,118                  ; CONVERT PSEUDO SCAN CODE TO RANGE
0455 EB 35                              JMP     SHORT K37A              ; GO FILL THE BUFFER

                               ;------ TRANSLATE ALTERNATE SHIFT PSEUDO SCAN CODES

0457                           K35:                                     ; ALT-FUNCTION
0457 3C 57                              CMP     AL,F11_M                ; IS IT F11?
0459 72 09                              JB      K35A                    ;  NO, BRANCH
045B 3C 58                              CMP     AL,F12_M                ; IS IT F12?
045D 77 05                              JA      K35A                    ;  NO, BRANCH
045F 80 C4 34                           ADD     AH,52                   ; CONVERT TO PSEUDO SCAN CODE
0462 EB 28                              JMP     SHORT K37A              ; GO FILL THE BUFFER

0464 F6 C7 02                  K35A:    TEST    BH,LC_E0                ; DO WE HAVE ONE OF THE NEW KEYS?
0467 74 18                              JZ      K37                     ;  NO, JUMP
0469 3C 1C                              CMP     AL,28                   ; TEST FOR KEYPAD ENTER
046B 75 06                              JNE     K35B                    ; NOT THERE
046D B8 A600                            MOV     AX,0A600h               ; SPECIAL CODE
0470 E9 05EE R                          JMP     K57                     ; BUFFER FILL
0473 3C 53                     K35B:    CMP     AL,83                   ; TEST FOR DELETE KEY
0475 74 1F                              JE      K37C                    ; HANDLE WITH OTHER EDIT KEYS
0477 3C 35                              CMP     AL,53                   ; TEST FOR KEYPAD /
0479 75 C0                              JNE     K32A                    ; NOT THERE, NO OTHER E0 SPECIALS
047B B8 A400                            MOV     AX,0A400h               ; SPECIAL CODE
047E E9 05EE R                          JMP     K57                     ; BUFFER FILL

0481 3C 3B                     K37:     CMP     AL,59                   ; TEST FOR FUNCTION KEYS (F1)
0483 72 0C                              JB      K37B                    ;  NO FN, HANDLE W/OTHER EXTENDED
0485 3C 44                              CMP     AL,68                   ; IN KEYPAD REGION?
                                                                        ; OR NUMLOCK, SCROLLOCK?
0487 77 B2                              JA      K32A                    ; IF SO, IGNORE
0489 80 C4 2D                           ADD     AH,45                   ; CONVERT TO PSEUDO SCAN CODE

048C B0 00                     K37A:    MOV     AL,0                    ; ASCII CODE OF ZERO
048E E9 05EE R                          JMP     K57                     ; PUT IT IN THE BUFFER

0491 B0 F0                     K37B:    MOV     AL,0F0h                 ; USE SPECIAL ASCII CODE
0493 E9 05EE R                          JMP     K57                     ; PUT IT IN THE BUFFER

0496 04 50                     K37C:    ADD     AL,80                   ; CONVERT SCAN CODE (EDIT KEYS)
0498 8A E0                              MOV     AH,AL                   ; (SCAN CODE NOT IN AH FOR INSERT)
049A EB F0                              JMP     K37A                    ; PUT IT IN THE BUFFER
                               ;------ NOT IN ALTERNATE SHIFT

049C                           K38:                                     ; NOT-ALT-SHIFT
                                                                        ; BL STILL HAS SHIFT FLAGS
049C F6 C3 04                           TEST    BL,CTL_SHIFT            ; ARE WE IN CONTROL SHIFT?
049F 75 03                              JNZ     K38A                    ;  YES, START PROCESSING
04A1 E9 052E R                          JMP     K44                     ; NOT-CTL-SHIFT

                               ;------ CONTROL SHIFT, TEST SPECIAL CHARACTERS
```

SECTION 5

```

                          ;------ TEST FOR BREAK

04A4 3C 46                K38A:   CMP     AL,SCROLL_KEY           ; TEST FOR BREAK
04A6 75 23                        JNE     K39                     ; JUMP, NO-BREAK
04A8 F6 C7 10                     TEST    BH,KBX                  ; IS THIS THE ENHANCED KEYBOARD?
04AB 74 05                        JZ      K38B                    ;  NO, BREAK IS VALID
04AD F6 C7 02                     TEST    BH,LC_E0                ;  YES, WAS LAST CODE AN E0?
04B0 74 19                        JZ      K39                     ;   NO-BREAK, TEST FOR PAUSE

04B2 8B 1E 001A R         K38B:   MOV     BX,@BUFFER_HEAD         ; RESET BUFFER TO EMPTY
04B6 89 1E 001C R                 MOV     @BUFFER_TAIL,BX         ;
04BA C6 06 0071 R 80              MOV     @BIOS_BREAK,80H         ; TURN ON BIOS_BREAK BIT

                          ;-------- ENABLE KEYBOARD

04BF B0 AE                        MOV     AL,ENA_KBD              ; ENABLE KEYBOARD
04C1 E8 0635 R                    CALL    SHIP_IT                 ; EXECUTE ENABLE
04C4 CD 1B                        INT     1BH                     ; BREAK INTERRUPT VECTOR
04C6 2B C0                        SUB     AX,AX                   ; PUT OUT DUMMY CHARACTER
04C8 E9 05EE R                    JMP     K57                     ; BUFFER_FILL

                          ;-------- TEST FOR PAUSE

04CB                      K39:                                    ; NO-BREAK
04CB F6 C7 10                     TEST    BH,KBX                  ; IS THIS THE ENHANCED KEYBOARD?
04CE 75 2A                        JNZ     K41                     ; YES, THEN THIS CAN'T BE PAUSE
04D0 3C 45                        CMP     AL,NUM_KEY              ; LOOK FOR PAUSE KEY
04D2 75 26                        JNE     K41                     ; NO-PAUSE
04D4 80 0E 0018 R 08      K39P:   OR      @KB_FLAG_1,HOLD_STATE   ; TURN ON THE HOLD FLAG

                          ;-------- ENABLE KEYBOARD

04D9 B0 AE                        MOV     AL,ENA_KBD              ; ENABLE KEYBOARD
04DB E8 0635 R                    CALL    SHIP_IT                 ; EXECUTE ENABLE
04DE B0 20                K39A:   MOV     AL,EOI                  ; END OF INTERRUPT TO CONTROL PORT
04E0 E6 20                        OUT     020H,AL                 ; ALLOW FURTHER KEYSTROKE INTS

                          ;------ DURING PAUSE INTERVAL, TURN CRT BACK ON

04E2 80 3E 0049 R 07              CMP     @CRT_MODE,7             ; IS THIS BLACK AND WHITE CARD
04E7 74 07                        JE      K40                     ; YES, NOTHING TO DO
04E9 BA 03D8                      MOV     DX,03D8H                ; PORT FOR COLOR CARD
04EC A0 0065 R                    MOV     AL,@CRT_MODE_SET        ; GET THE VALUE OF THE CURRENT MODE
04EF EE                           OUT     DX,AL                   ; SET THE CRT MODE, SO THAT CRT IS ON
04F0                      K40:                                    ; PAUSE-LOOP
04F0 F6 06 0018 R 08              TEST    @KB_FLAG_1,HOLD_STATE
04F5 75 F9                        JNZ     K40                     ; LOOP UNTIL FLAG TURNED OFF
04F7 E9 03A3 R                    JMP     K27                     ; INTERRUPT_RETURN_NO_EOI

                          ;------ TEST SPECIAL CASE KEY 55

04FA                      K41:                                    ; NO-PAUSE
04FA 3C 37                        CMP     AL,55                   ; TEST FOR */PRTSC KEY
04FC 75 10                        JNE     K42                     ; NOT-KEY-55
04FE F6 C7 10                     TEST    BH,KBX                  ; IS THIS THE ENHANCED KEYBOARD?
0501 74 05                        JZ      K41A                    ;  NO, CTL-PRTSC IS VALID
0503 F6 C7 02                     TEST    BH,LC_E0                ;  YES, WAS LAST CODE AN E0?
0506 74 20                        JZ      K42B                    ;   NO, TRANSLATE TO A FUNCTION
0508 B8 7200              K41A:   MOV     AX,114*256              ; START/STOP PRINTING SWITCH
050B E9 05EE R                    JMP     K57                     ; BUFFER_FILL

                          ;------ SET UP TO TRANSLATE CONTROL SHIFT

050E                      K42:                                    ; NOT-KEY-55
050E 3C 0F                        CMP     AL,15                   ; IS IT THE TAB KEY?
0510 74 16                        JE      K42B                    ;  YES, XLATE TO FUNCTION CODE
0512 3C 35                        CMP     AL,53                   ; IS IT THE / KEY?
0514 75 0B                        JNE     K42A                    ;  NO, NO MORE SPECIAL CASES
0516 F6 C7 02                     TEST    BH,LC_E0                ;  YES, IS IT FROM THE KEYPAD?
0519 74 06                        JZ      K42A                    ;   NO, JUST TRANSLATE
051B B8 9500                      MOV     AX,9500h                ;   YES, SPECIAL CODE FOR THIS ONE
051E E9 05EE R                    JMP     K57                     ;   BUFFER FILL

0521 BB 0000 E            K42A:   MOV     BX,OFFSET K8            ; SET UP TO TRANSLATE CTL
0524 3C 3B                        CMP     AL,59                   ; IS IT IN CHARACTER TABLE?
0526 72 5E                        JB      K45F                    ;  YES, GO TRANSLATE CHAR
0528 BB 0000 E            K42B:   MOV     BX,OFFSET K8            ; SET UP TO TRANSLATE CTL
052B E9 05DD R                    JMP     K64                     ;  NO, GO TRANSLATE_SCAN
                          ;------ NOT IN CONTROL SHIFT

052E 3C 37                K44:    CMP     AL,55                   ; PRINT SCREEN KEY?
0530 75 26                        JNE     K45                     ; NOT-PRINT-SCREEN
0532 F6 C7 10                     TEST    BH,KBX                  ; IS THIS ENHANCED KEYBOARD?
0535 74 07                        JZ      K44A                    ; NO, TEST FOR SHIFT STATE
0537 F6 C7 02                     TEST    BH,LC_E0                ; YES, LAST CODE A MARKER?
053A 75 07                        JNZ     K44B                    ;  YES, IS PRINT SCREEN
053C EB 3B                        JMP     SHORT K45C              ;  NO, XLATE TO "*" CHARACTER
053E F6 C3 03             K44A:   TEST    BL,LEFT_SHIFT+RIGHT_SHIFT ;NOT 101 KBD, SHIFT KEY DOWN?
0541 74 36                        JZ      K45C                    ; NO, XLATE TO "*" CHARACTER

                          ;------ ISSUE INTERRUPT TO PERFORM PRINT SCREEN FUNCTION
0543 B0 AE                K44B:   MOV     AL,ENA_KBD              ; INSURE KEYBOARD IS ENABLED
0545 E8 0635 R                    CALL    SHIP_IT                 ; EXECUTE ENABLE
0548 B0 20                        MOV     AL,EOI                  ; END OF CURRENT INTERRUPT
054A E6 20                        OUT     020H,AL                 ;  SO FURTHER THINGS CAN HAPPEN
054C 55                           PUSH    BP                      ; SAVE POINTER
054D CD 05                        INT     5H                      ; ISSUE PRINT SCREEN INTERRUPT
054F 5D                           POP     BP                      ; RESTORE POINTER
0550 80 26 0096 R FC              AND     @KB_FLAG_3,NOT LC_E0+LC_E1 ;ZERO OUT THESE FLAGS
0555 E9 03A3 R                    JMP     K27                     ; GO BACK WITHOUT EOI OCCURRING


                          ;------ HANDLE THE IN-CORE KEYS
0558                      K45:                                    ; NOT-PRINT-SCREEN
0558 3C 3A                        CMP     AL,58                   ; TEST FOR IN-CORE AREA
055A 77 2C                        JA      K46                     ; JUMP IF NOT

055C 3C 35                        CMP     AL,53                   ; IS THIS THE "/" KEY?
055E 75 05                        JNE     K45A                    ;  NO, JUMP
0560 F6 C7 02                     TEST    BH,LC_E0                ; WAS LAST CODE THE MARKER?
0563 75 14                        JNZ     K45C                    ;  YES, TRANSLATE TO CHARACTER

0565 B9 001A              K45A:   MOV     CX,26                   ; LENGTH OF SEARCH
```

```
0568 BF 03E7 R                   MOV    DI,OFFSET K30+10          ; POINT TO TABLE OF A-Z CHARS
056B F2/ AE                      REPNE  SCASB                     ; IS THIS A LETTER KEY?
056D 75 05                       JNE    K45B                      ;  NO, SYMBOL KEY
056F F6 C3 40                    TEST   BL,CAPS_STATE             ; ARE WE IN CAPS_LOCK?
0572 75 0A                       JNZ    K45D                      ; TEST FOR SURE
0574 F6 C3 03            K45B:   TEST   BL,LEFT_SHIFT+RIGHT_SHIFT ; ARE WE IN SHIFT STATE?
0577 75 0A                       JNZ    K45E                      ; YES, UPPERCASE
                                                                  ; NO, LOWERCASE
0579 BB 0000 E           K45C:   MOV    BX,OFFSET K10             ; TRANSLATE TO LOWERCASE LETTERS
057C EB 50                       JMP    SHORT K56
057E                     K45D:                                    ; ALMOST-CAPS-STATE
057E F6 C3 03                    TEST   BL,LEFT_SHIFT+RIGHT_SHIFT ; CL ON. IS SHIFT ON, TOO?
0581 75 F6                       JNZ    K45C                      ; SHIFTED TEMP OUT OF CAPS STATE
0583 BB 0000 E           K45E:   MOV    BX,OFFSET K11             ; TRANSLATE TO UPPERCASE LETTERS
0586 EB 46               K45F:   JMP    SHORT K56

                         ;------ TEST FOR KEYS F1 - F10
0588                     K46:                                     ; NOT IN-CORE AREA
0588 3C 44                       CMP    AL,68                     ; TEST FOR F1 - F10
058A 77 02                       JA     K47                       ; JUMP IF NOT
058C EB 36                       JMP    SHORT K53                 ; YES, GO DO FN KEY PROCESS

                         ;------ HANDLE THE NUMERIC PAD KEYS
058E                     K47:                                     ; NOT F1 - F10
058E 3C 53                       CMP    AL,83                     ; TEST FOR NUMPAD KEYS
0590 77 2C                       JA     K52                       ; JUMP IF NOT

                         ;------ KEYPAD KEYS, MUST TEST NUM LOCK FOR DETERMINATION
0592 3C 4A               K48:    CMP    AL,74                     ; SPECIAL CASE FOR MINUS
0594 74 ED                       JE     K45E                      ; GO TRANSLATE
0596 3C 4E                       CMP    AL,78                     ; SPECIAL CASE FOR PLUS
0598 74 E9                       JE     K45E                      ; GO TRANSLATE
059A F6 C7 02                    TEST   BH,LC_E0                  ; IS THIS ONE OF THE NEW KEYS?
059D 75 0A                       JNZ    K49                       ;  YES, TRANSLATE TO BASE STATE

059F F6 C3 20                    TEST   BL,NUM_STATE              ; ARE WE IN NUM_LOCK?
05A2 75 13                       JNZ    K50                       ; TEST FOR SURE
05A4 F6 C3 03                    TEST   BL,LEFT_SHIFT+RIGHT_SHIFT  ; ARE WE IN SHIFT STATE?
05A7 75 13                       JNZ    K51                        ; IF SHIFTED, REALLY NUM STATE

                         ;------ BASE CASE FOR KEYPAD
05A9 3C 4C               K49:    CMP    AL,76                     ; SPECIAL CASE FOR BASE STATE 5
05AB 75 05                       JNE    K49A                      ; CONTINUE IF NOT KEYPAD 5
05AD B0 F0                       MOV    AL,0F0h                   ; SPECIAL ASCII CODE
05AF EB 3D 90                    JMP    K57                       ; BUFFER FILL
05B2 BB 0000 E           K49A:   MOV    BX,OFFSET K10             ; BASE CASE TABLE
05B5 EB 26                       JMP    SHORT K64                 ; CONVERT TO PSEUDO SCAN

                         ;------ MIGHT BE NUM LOCK, TEST SHIFT STATUS
05B7 F6 C3 03            K50:    TEST   BL,LEFT_SHIFT+RIGHT_SHIFT        ;ALMOST-NUM-STATE
05BA 75 ED                       JNZ    K49                       ; SHIFTED TEMP OUT OF NUM STATE
05BC EB C5               K51:    JMP    SHORT K45E                ; REALLY_NUM_STATE

                         ;------ TEST FOR THE NEW KEY ON WT KEYBOARDS
05BE                     K52:                                     ; NOT A NUMPAD KEY
05BE 3C 56                       CMP    AL,86                     ; IS IT THE NEW WT KEY?
05C0 75 02                       JNE    K53                       ; JUMP IF NOT
05C2 EB B0                       JMP    SHORT K45B                ; HANDLE WITH REST OF LETTER KEYS

                         ;------ MUST BE F11 OR F12
                                                                  ; F1 - F10 COME HERE, TOO
05C4 F6 C3 03            K53:    TEST   BL,LEFT_SHIFT+RIGHT_SHIFT ;TEST SHIFT STATE
05C7 74 E0                       JZ     K49                        ; JUMP, LOWERCASE PSEUDO SC'S

05C9 BB 0000 E                   MOV    BX,OFFSET K11             ; UPPER CASE PSEUDO SCAN CODES
05CC EB 0F                       JMP    SHORT K64                 ; TRANSLATE_SCAN
                         ;------ TRANSLATE THE CHARACTER

05CE                     K56:                                     ; TRANSLATE-CHAR
05CE FE C8                       DEC    AL                        ; CONVERT ORIGIN
05D0 2E: D7                      XLAT   CS:K11                    ; CONVERT THE SCAN CODE TO ASCII
05D2 F6 06 0096 R 02             TEST   @KB_FLAG_3,LC_E0          ; IS THIS A NEW KEY?
05D7 74 15                       JZ     K57                       ;  NO, GO FILL BUFFER
05D9 B4 E0                       MOV    AH,MC_E0                  ;  YES, PUT SPECIAL MARKER IN AH
05DB EB 11                       JMP    SHORT K57                 ; PUT IT INTO THE BUFFER

                         ;------ TRANSLATE SCAN FOR PSEUDO SCAN CODES

05DD                     K64:                                     ; TRANSLATE-SCAN-ORGD
05DD FE C8                       DEC    AL                        ; CONVERT ORIGIN
05DF 2E: D7                      XLAT   CS:K8                     ; CTL TABLE SCAN
05E1 8A E0                       MOV    AH,AL                     ; PUT VALUE INTO AH
05E3 B0 00                       MOV    AL,0                      ; ZERO ASCII CODE
05E5 F6 06 0096 R 02             TEST   @KB_FLAG_3,LC_E0          ; IS THIS A NEW KEY?
05EA 74 02                       JZ     K57                       ;  NO, GO FILL BUFFER
05EC B0 E0                       MOV    AL,MC_E0                  ;  YES, PUT SPECIAL MARKER IN AL

                         ;------ PUT CHARACTER INTO BUFFER

05EE                     K57:                                     ; BUFFER-FILL
05EE 3C FF                       CMP    AL,-1                     ; IS THIS AN IGNORE CHAR
05F0 74 05                       JE     K59                       ; YES, DO NOTHING WITH IT
05F2 80 FC FF                    CMP    AH,-1                     ; LOOK FOR -1 PSEUDO SCAN
05F5 75 03                       JNE    K61                       ; NEAR_INTERRUPT_RETURN

05F7                     K59:                                     ; NEAR-INTERRUPT-RETURN
05F7 E9 0399 R                   JMP    K26                       ; INTERRUPT_RETURN

05FA                     K61:
05FA 8B 1E 001C R                MOV    BX,@BUFFER_TAIL           ; GET THE END POINTER TO THE BUFFER
05FE 8B F3                       MOV    SI,BX                     ; SAVE THE VALUE
0600 E8 0168 R                   CALL   K4                        ; ADVANCE THE TAIL
0603 3B 1E 001A R                CMP    BX,@BUFFER_HEAD           ; HAS THE BUFFER WRAPPED AROUND
0607 74 1D                       JE     K62                       ; BUFFER_FULL_BEEP
0609 89 04                       MOV    [SI],AX                   ; STORE THE VALUE
060B 89 1E 001C R                MOV    @BUFFER_TAIL,BX           ; MOVE THE POINTER UP
060F FA                          CLI                              ; TURN OFF INTERRUPTS
```

SECTION 5

```
0610 B0 20                          MOV     AL,EOI                  ; END OF INTERRUPT COMMAND
0612 E6 20                          OUT     INTA00,AL               ; SEND COMMAND TO INTERRUPT CONTROL PORT
0614 B0 AE                          MOV     AL,ENA_KBD              ; INSURE KEYBOARD IS ENABLED
0616 E8 0635 R                      CALL    SHIP_IT                 ; EXECUTE ENABLE
0619 B8 9102                        MOV     AX,09102H               ; MOVE IN POST CODE & TYPE
061C CD 15                          INT     15H                     ; PERFORM OTHER FUNCTION
061E 80 26 0096 R FC                AND     @KB_FLAG_3,NOT LC_E0+LC_E1 ; RESET LAST CHAR H.C. FLAG
0623 E9 03A8 R                      JMP     K27A                    ; INTERRUPT_RETURN

                            ;------ BUFFER IS FULL SOUND THE BEEPER

0626                        K62:
0626 B0 20                          MOV     AL,EOI                  ; ENABLE INTERRUPT CONTROLLER CHIP
0628 E6 20                          OUT     INTA00,AL               ;
062A B9 02A6                        MOV     CX,678                  ; DIVISOR FOR 1760 HZ
062D B3 04                          MOV     BL,4                    ; SHORT BEEP COUNT (1/16 + 1/64 DELAY)
062F E8 0000 E                      CALL    BEEP                    ; GO TO COMMON BEEP HANDLER
0632 E9 03A3 R                      JMP     K27                     ; EXIT

0635                        KB_INT_1        ENDP
                            ;---------------------------------------------------------------
                            ;                                                              -
                            ;       SHIP_IT                                                -
                            ;                                                              -
                            ;       THIS ROUTINE HANDLES TRANSMISSION OF COMMAND AND DATA BYTES -
                            ;       TO THE KEYBOARD CONTROLLER.                            -
                            ;                                                              -
                            ;---------------------------------------------------------------
0635                        SHIP_IT PROC    NEAR
0635 50                             PUSH    AX                      ; SAVE DATA TO SEND

                            ;------- WAIT FOR COMMAND TO BE ACCEPTED
0636 FA                             CLI                             ; DISABLE INTERRUPTS TILL DATA SENT
0637 2B C9                          SUB     CX,CX                   ; CLEAR TIMEOUT COUNTER
0639                        S10:
0639 E4 64                          IN      AL,STATUS_PORT          ; READ KEYBOARD CONTROLLER STATUS
063B A8 02                          TEST    AL,INPT_BUF_FULL        ; CHECK FOR ITS INPUT BUFFER BUSY
063D E0 FA                          LOOPNZ  S10                     ; WAIT FOR COMMAND TO BE ACCEPTED

063F 58                             POP     AX                      ; GET DATA TO SEND
0640 E6 64                          OUT     STATUS_PORT,AL          ; SEND TO KEYBOARD CONTROLLER
0642 FB                             STI                             ; ENABLE INTERRUPTS AGAIN
0643 C3                             RET                             ; RETURN TO CALLER
0644                        SHIP_IT ENDP
                            ;---------------------------------------------------------------
                            ;                                                              -
                            ;       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                        -
                            ;                                                              -
                            ;---------------------------------------------------------------

0644                        SND_DATA PROC   NEAR
0644 50                             PUSH    AX                      ; SAVE REGISTERS
0645 53                             PUSH    BX                      ; *
0646 51                             PUSH    CX
0647 8A F8                          MOV     BH,AL                   ; SAVE TRANSMITTED BYTE FOR RETRIES
0649 B3 03                          MOV     BL,3                    ; LOAD RETRY COUNT
064B FA                     SD0:    CLI                             ; DISABLE INTERRUPTS
064C 80 26 0097 R CF                AND     @KB_FLAG_2,NOT (KB_FE+KB_FA) ; CLEAR ACK AND RESEND FLAGS

                            ;------- WAIT FOR COMMAND TO BE ACCEPTED

0651 2B C9                          SUB     CX,CX
0653 E4 64                  SD5:    IN      AL,STATUS_PORT
0655 A8 02                          TEST    AL,INPT_BUF_FULL
0657 E0 FA                          LOOPNZ  SD5                     ; WAIT FOR COMMAND TO BE ACCEPTED

0659 8A C7                          MOV     AL,BH                   ; REESTABLISH BYTE TO TRANSMIT
065B E6 60                          OUT     PORT_A,AL               ; SEND BYTE
065D FB                             STI                             ; ENABLE INTERRUPTS
065E B9 1A00                        MOV     CX,01A00H               ; LOAD COUNT FOR 10mS+
0661 F6 06 0097 R 30        SD1:    TEST    @KB_FLAG_2,KB_FE+KB_FA  ; SEE IF EITHER BIT SET
0666 75 0D                          JNZ     SD3                     ; IF SET, SOMETHING RECEIVED GO PROCESS
0668 E2 F7                          LOOP    SD1                     ; OTHERWISE WAIT

066A FE CB                  SD2:    DEC     BL                      ; DECREMENT RETRY COUNT
066C 75 DD                          JNZ     SD0                     ; RETRY TRANSMISSION
066E 80 0E 0097 R 80                OR      @KB_FLAG_2,KB_ERR       ; TURN ON TRANSMIT ERROR FLAG
0673 EB 07                          JMP     SHORT SD4               ; RETRIES EXHAUSTED FORGET TRANSMISSION

0675 F6 06 0097 R 10        SD3:    TEST    @KB_FLAG_2,KB_FA        ; SEE IF THIS IS AN ACKNOWLEDGE
067A 74 EE                          JZ      SD2                     ; IF NOT, GO RESEND

067C 59                     SD4:    POP     CX                      ; RESTORE REGISTERS
067D 5B                             POP     BX                      ;
067E 58                             POP     AX                      ; *
067F C3                             RET                             ; RETURN, GOOD TRANSMISSION
0680                        SND_DATA ENDP
                            ;---------------------------------------------------------------
                            ;                                                              -
                            ;       SND_LED                                                -
                            ;                                                              -
                            ;       THIS ROUTINE TURNS ON THE MODE INDICATORS.             -
                            ;                                                              -
                            ;---------------------------------------------------------------
0680                        SND_LED PROC    NEAR
0680 FA                             CLI                             ; TURN OFF INTERRUPTS
0681 F6 06 0097 R 40                TEST    @KB_FLAG_2,KB_PR_LED    ; CHECK FOR MODE INDICATOR UPDATE
0686 75 47                          JNZ     SL1                     ; DONT UPDATE AGAIN IF UPDATE UNDERWAY
                            ;
0688 80 0E 0097 R 40                OR      @KB_FLAG_2,KB_PR_LED    ; TURN ON UPDATE IN PROCESS
068D B0 20                          MOV     AL,EOI                  ; END OF INTERRUPT COMMAND
068F E6 20                          OUT     020H,AL                 ; SEND COMMAND TO INTERRUPT CONTROL PORT
0691 EB 0D                          JMP     SHORT SL0               ; GO SEND MODE INDICATOR COMMAND
                            ;
0693                        SND_LED1:
0693 FA                             CLI                             ; TURN OFF INTERRUPTS
0694 F6 06 0097 R 40                TEST    @KB_FLAG_2,KB_PR_LED    ; CHECK FOR MODE INDICATOR UPDATE
0699 75 34                          JNZ     SL1                     ; DONT UPDATE AGAIN IF UPDATE UNDERWAY
                            ;
069B 80 0E 0097 R 40                OR      @KB_FLAG_2,KB_PR_LED    ; TURN ON UPDATE IN PROCESS
```

```
06A0 B0 ED                 SL0:    MOV     AL,LED_CMD              ; LED CMD BYTE
06A2 E8 0644 R                     CALL    SND_DATA                ; SEND DATA TO KEYBOARD
06A5 FA                            CLI                             ;
06A6 E8 06D1 R                     CALL    MAKE_LED                ; GO FORM INDICATOR DATA BYTE
06A9 80 26 0097 R F8               AND     @KB_FLAG_2,0F8H         ; CLEAR MODE INDICATOR BITS
06AE 08 06 0097 R                  OR      @KB_FLAG_2,AL           ; SAVE PRESENT INDICATORS FOR NEXT TIME
06B2 F6 06 0097 R 80               TEST    @KB_FLAG_2,KB_ERR       ; TRANSMIT ERROR DETECTED
06B7 75 0B                         JNZ     SL2                     ;  YES, BYPASS SECOND BYTE TRANSMISSION
                           ;
06B9 E8 0644 R                     CALL    SND_DATA                ; SEND DATA TO KEYBOARD
06BC FA                            CLI                             ; TURN OFF INTERRUPTS
06BD F6 06 0097 R 80               TEST    @KB_FLAG_2,KB_ERR       ; TRANSMIT ERROR DETECTED
06C2 74 06                         JZ      SL3                     ; IF NOT, DONT SEND AN ENABLE COMMAND
                           ;
06C4 B0 F4                 SL2:    MOV     AL,KB_ENABLE            ; GET KEYBOARD CSA ENABLE COMMAND
06C6 E8 0644 R                     CALL    SND_DATA                ; SEND DATA TO KEYBOARD
06C9 FA                            CLI                             ; TURN OFF INTERRUPTS
06CA 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
06CF FB                    SL1:    STI                             ; ENABLE INTERRUPTS
06D0 C3                            RET                             ; RETURN TO CALLER
06D1                       SND_LED ENDP
                           ;---------------------------------------------------------------
                           ;                                                              -
                           ;       MAKE_LED                                               -
                           ;                                                              -
                           ;       THIS ROUTINE FORMS THE DATA BYTE NECESSARY TO TURN ON/OFF   -
                           ;         THE MODE INDICATORS                                  -
                           ;                                                              -
                           ;---------------------------------------------------------------
06D1                       MAKE_LED PROC   NEAR
06D1 51                            PUSH    CX                      ; SAVE CX
06D2 A0 0017 R                     MOV     AL,@KB_FLAG             ; GET CAPS & NUM LOCK INDICATORS
06D5 24 70                         AND     AL,CAPS_STATE+NUM_STATE+SCROLL_STATE ; ISOLATE INDICATORS
06D7 B1 04                         MOV     CL,4                    ; SHIFT COUNT
06D9 D2 C0                         ROL     AL,CL                   ; SHIFT BITS OVER TO TURN ON INDICATORS
06DB 24 07                         AND     AL,07H                  ; MAKE SURE ONLY MODE BITS ON
06DD 59                            POP     CX
06DE C3                            RET                             ; RETURN TO CALLER
06DF                       MAKE_LED ENDP

06DF                       CODE    ENDS
                                   END
```

```
                                PAGE 118,123
                                TITLE PRT ------ 11/15/85  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 BITS 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      B80                     ;  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                           B20:
 0030 50                                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

 003E                           B25:                                    ; ADJUST OUTER LOOP COUNT
 003E 2A FF                             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:                                    ;       SEND STROBE PULSE
005A 5B                        POP     BX                      ; RESTORE (BX) WITH TIMEOUT COUNT
005B B0 0D                     MOV     AL,0DH                  ; SET THE STROBE LOW (BIT ON)
005D 42                        INC     DX                      ; OUTPUT STROBE TO CONTROL PORT
005E FA                        CLI                             ; PREVENT INTERRUPT PULSE STRETCHING
005F EE                        OUT     DX,AL                   ; OUTPUT STROBE BIT   > 1us  < 5us
0060 EB 00                     JMP     $+2                     ; I/O DELAY TO ALLOW FOR LINE LOADING
0062 EB 00                     JMP     $+2                     ;  AND FOR CORRECT PULSE WIDTH
0064 B0 0C                     MOV     AL,0CH                  ; SET THE -STROBE HIGH
0066 EE                        OUT     DX,AL
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

008F                   PRINTER_IO_1    ENDP

008F                           CODE    ENDS
                               END
```

SECTION 5

```
                                PAGE 118,123
                                TITLE RS232 ---- 11/15/85  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 (0,1 ALLOWED)            :
                                ;                                                                             :
                                ; DATA AREA @RS232_BASE CONTAINS THE BASE ADDRESS OF THE 8250 ON THE CARD     :
                                ;       LOCATION 400H CONTAINS UP TO 4 RS232 ADDRESSES POSSIBLE               :
                                ;       DATA AREA 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 D1 EA                             SHR     DX,1                    ; TEST PARAMETER
 000D 75 20                             JNZ     A3                      ; RETURN IF NOT IN RANGE
 000F D1 E6                             SHL     SI,1                    ; WORD OFFSET
 0011 E8 0000 E                         CALL    DDS
 0014 8B 94 0000 R                      MOV     DX,@RS232_BASE[SI]      ; GET BASE ADDRESS
 0018 0B D2                             OR      DX,DX                   ; TEST FOR 0 BASE ADDRESS
 001A 74 13                             JZ      A3                      ; RETURN
 001C 0A E4                             OR      AH,AH                   ; TEST FOR (AH)= 00H
 001E 74 16                             JZ      A4                      ; COMMO INITIALIZATION
 0020 FE CC                             DEC     AH                      ; TEST FOR (AH)= 01H
 0022 74 4B                             JZ      A5                      ; SEND (AL)
 0024 FE CC                             DEC     AH                      ; TEST FOR (AH)= 02H
 0026 74 70                             JZ      A12                     ; RECEIVE INTO (AL)
 0028                           A2:
 0028 FE CC                             DEC     AH                      ; TEST FOR (AH)= 03H
 002A 75 03                             JNZ     A3
 002C E9 00BA R                         JMP     A18                     ; COMMUNICATION STATUS
 002F                           A3:                                     ; RETURN FROM RS232
 002F 5B                                POP     BX
 0030 59                                POP     CX
 0031 5F                                POP     DI
 0032 5E                                POP     SI
 0033 5A                                POP     DX
```

```
0034 1F                        POP     DS
0035 CF                        IRET                        ; RETURN TO CALLER, NO ACTION
                        PAGE
                        ;-----  INITIALIZE THE COMMUNICATIONS PORT

0036                    A4:
0036 8A E0                     MOV     AH,AL               ; SAVE INITIALIZATION PARAMETERS IN (AH)
0038 83 C2 03                  ADD     DX,3                ; POINT TO 8250 CONTROL REGISTER
003B B0 80                     MOV     AL,80H
003D EE                        OUT     DX,AL               ; SET DLAB=1

                        ;-----  DETERMINE BAUD RATE DIVISOR

003E 8A D4                     MOV     DL,AH               ; GET PARAMETERS TO (DL)
0040 B1 04                     MOV     CL,4
0042 D2 C2                     ROL     DL,CL
0044 81 E2 000E                AND     DX,0EH              ; ISOLATE THEM
0048 BF 0000 E                 MOV     DI,OFFSET A1        ; BASE OF TABLE
004B 03 FA                     ADD     DI,DX               ; PUT INTO INDEX REGISTER
004D 8B 94 0000 R              MOV     DX,@RS232_BASE[SI]  ; POINT TO HIGH ORDER OF DIVISOR
0051 42                        INC     DX
0052 2E: 8A 45 01              MOV     AL,CS:[DI]+1        ; GET HIGH ORDER OF DIVISOR
0056 EE                        OUT     DX,AL               ; SET ms OF DIVISOR TO 0
0057 4A                        DEC     DX
0058 EB 00                     JMP     $+2                 ; I/O DELAY
005A 2E: 8A 05                 MOV     AL,CS:[DI]          ; GET LOW ORDER OF DIVISOR
005D EE                        OUT     DX,AL               ; SET LOW OF DIVISOR
005E 83 C2 03                  ADD     DX,3
0061 8A C4                     MOV     AL,AH               ; GET PARAMETERS BACK
0063 24 1F                     AND     AL,01FH             ; STRIP OFF THE BAUD BITS
0065 EE                        OUT     DX,AL               ; LINE CONTROL TO 8 BITS
0066 4A                        DEC     DX
0067 4A                        DEC     DX
0068 EB 00                     JMP     $+2                 ; I/O DELAY
006A B0 00                     MOV     AL,0
006C EE                        OUT     DX,AL               ; INTERRUPT ENABLES ALL OFF
006D EB 4B                     JMP     SHORT A18           ; COM_STATUS

                        ;-----  SEND CHARACTER IN (AL) OVER COMMO LINE

006F                    A5:
006F 50                        PUSH    AX                  ; SAVE CHAR TO SEND
0070 83 C2 04                  ADD     DX,4                ; MODEM CONTROL REGISTER
0073 B0 03                     MOV     AL,3                ; DTR AND RTS
0075 EE                        OUT     DX,AL               ; DATA TERMINAL READY, REQUEST TO SEND
0076 42                        INC     DX                  ; MODEM STATUS REGISTER
0077 42                        INC     DX
0078 B7 30                     MOV     BH,30H              ; DATA SET READY & CLEAR TO SEND
007A E8 00C9 R                 CALL    WAIT_FOR_STATUS     ; ARE BOTH TRUE
007D 74 08                     JE      A9                  ; YES, READY TO TRANSMIT CHAR
007F                    A7:
007F 59                        POP     CX
0080 8A C1                     MOV     AL,CL               ; RELOAD DATA BYTE
0082                    A8:
0082 80 CC 80                  OR      AH,80H              ; INDICATE TIME OUT
0085 EB A8                     JMP     A3                  ; RETURN

0087                    A9:                                 ; CLEAR_TO_SEND
0087 4A                        DEC     DX                  ; LINE STATUS REGISTER
0088                    A10:                                ; WAIT_SEND
0088 B7 20                     MOV     BH,20H              ; IS TRANSMITTER READY
008A E8 00C9 R                 CALL    WAIT_FOR_STATUS     ; TEST FOR TRANSMITTER READY
008D 75 F0                     JNZ     A7                  ; RETURN WITH TIME OUT SET
008F                    A11:                                ; OUT_CHAR
008F 83 EA 05                  SUB     DX,5                ; DATA PORT
0092 59                        POP     CX                  ; RECOVER IN CX TEMPORARILY
0093 8A C1                     MOV     AL,CL               ; MOVE CHAR TO AL FOR OUT, STATUS IN AH
0095 EE                        OUT     DX,AL               ; OUTPUT CHARACTER
0096 EB 97                     JMP     A3                  ; RETURN

                        ;-----  RECEIVE CHARACTER FROM COMMO LINE

0098                    A12:
0098 83 C2 04                  ADD     DX,4                ; MODEM CONTROL REGISTER
009B B0 01                     MOV     AL,1                ; DATA TERMINAL READY
009D EE                        OUT     DX,AL
009E 42                        INC     DX                  ; MODEM STATUS REGISTER
009F 42                        INC     DX
00A0                    A13:                                ; WAIT_DSR
00A0 B7 20                     MOV     BH,20H              ; DATA_SET READY
00A2 E8 00C9 R                 CALL    WAIT_FOR_STATUS     ; TEST FOR DSR
00A5 75 DB                     JNZ     A8                  ; RETURN WITH ERROR
00A7                    A15:                                ; WAIT_DSR_END
00A7 4A                        DEC     DX                  ; LINE_STATUS REGISTER
00A8                    A16:                                ; WAIT_RECV
00A8 B7 01                     MOV     BH,1                ; RECEIVE BUFFER FULL
00AA E8 00C9 R                 CALL    WAIT_FOR_STATUS     ; TEST FOR RECEIVE BUFFER FULL
00AD 75 D3                     JNZ     A8                  ; SET TIME OUT ERROR
00AF                    A17:                                ; GET_CHAR
00AF 80 E4 1E                  AND     AH,00011110B        ; TEST FOR ERROR CONDITIONS ON RECEIVE

00B2 8B 94 0000 R              MOV     DX,@RS232_BASE[SI]  ; DATA PORT
00B6 EC                        IN      AL,DX               ; GET CHARACTER FROM LINE
00B7 E9 002F R                 JMP     A3                  ; RETURN

                        ;-----  COMMO PORT STATUS ROUTINE

00BA                    A18:
00BA 8B 94 0000 R              MOV     DX,@RS232_BASE[SI]
00BE 83 C2 05                  ADD     DX,5                ; CONTROL PORT
00C1 EC                        IN      AL,DX               ; GET LINE CONTROL STATUS
00C2 8A E0                     MOV     AH,AL               ; PUT IN (AH) FOR RETURN
00C4 42                        INC     DX                  ; POINT TO MODEM STATUS REGISTER
00C5 EC                        IN      AL,DX               ; GET MODEM CONTROL STATUS
00C6 E9 002F R                 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          :
                                ;----------------------------------------

 00C9                           WAIT_FOR_STATUS PROC    NEAR
 00C9 8A 9D 007C R                      MOV     BL,@RS232_TIM_OUT[DI]   ; LOAD OUTER LOOP COUNT

                                ;-----  ADJUST OUTER LOOP COUNT

 00CD 55                                PUSH    BP                      ; SAVE (BP)
 00CE 53                                PUSH    BX                      ; SAVE (BX)
 00CF 5D                                POP     BP                      ; USE BP FOR OUTER LOOP COUNT
 00D0 81 E5 00FF                        AND     BP,00FFH                ; STRIP HIGH BITS
 00D4 D1 D5                             RCL     BP,1                    ; MULTIPLY OUTER COUNT BY 4
 00D6 D1 D5                             RCL     BP,1
 00D8                           WFS0:
 00D8 2B C9                             SUB     CX,CX
 00DA                           WFS1:
 00DA EC                                IN      AL,DX                   ; GET STATUS
 00DB 8A E0                             MOV     AH,AL                   ; MOVE TO (AH)
 00DD 22 C7                             AND     AL,BH                   ; ISOLATE BITS TO TEST
 00DF 3A C7                             CMP     AL,BH                   ; EXACTLY = TO MASK
 00E1 74 07                             JE      WFS_END                 ; RETURN WITH ZERO FLAG ON

 00E3 E2 F5                             LOOP    WFS1                    ; TRY AGAIN

 00E5 4D                                DEC     BP
 00E6 75 F0                             JNZ     WFS0

 00E8 0A FF                             OR      BH,BH                   ; SET ZERO FLAG OFF
 00EA                           WFS_END:
 00EA 5D                                POP     BP                      ; RESTORE (BP)
 00EB C3                                RET

 00EC                           WAIT_FOR_STATUS ENDP

 00EC                           RS232_IO_1      ENDP

 00EC                           CODE    ENDS
                                        END
```

```
                                PAGE 118,123
                                TITLE VIDEO1 --- 11/15/85  VIDEO DISPLAY BIOS
                                .286C
                                .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               ; SPEEKER 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.                  :
                                ;                                                                             :
```

SECTION 5

```
                           ;    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 0067 R               M1      DW      OFFSET  SET_MODE        ;  TABLE OF ROUTINES WITHIN VIDEO I/O
 0002 014E R                       DW      OFFSET  SET_CTYPE
 0004 0173 R                       DW      OFFSET  SET_CPOS
 0006 019B R                       DW      OFFSET  READ_CURSOR
 0008 0791 R                       DW      OFFSET  READ_LPEN
 000A 01B2 R                       DW      OFFSET  ACT_DISP_PAGE
 000C 021F R                       DW      OFFSET  SCROLL_UP
 000E 02BE R                       DW      OFFSET  SCROLL_DOWN
 0010 0310 R                       DW      OFFSET  READ_AC_CURRENT
 0012 036A R                       DW      OFFSET  WRITE_AC_CURRENT
 0014 039C R                       DW      OFFSET  WRITE_C_CURRENT
 0016 01D6 R                       DW      OFFSET  SET_COLOR
 0018 045D R                       DW      OFFSET  WRITE_DOT
 001A 044C R                       DW      OFFSET  READ_DOT
 001C 070A R                       DW      OFFSET  WRITE_TTY
 001E 01FC R                       DW      OFFSET  VIDEO_STATE
 0020 0145 R                       DW      OFFSET  VIDEO_RETURN    ; RESERVED
 0022 0145 R                       DW      OFFSET  VIDEO_RETURN    ; RESERVED
 0024 0145 R                       DW      OFFSET  VIDEO_RETURN    ; RESERVED
 0026 03C9 R                       DW      OFFSET  WRITE_STRING    ; CASE 19H, WRITE STRING
 = 0028                    M1L     EQU     $-M1

 0028                      VIDEO_IO_1      PROC    NEAR            ;       ENTRY POINT FOR ORG 0F065H
 0028 FB                           STI                             ; INTERRUPTS BACK ON
 0029 FC                           CLD                             ; SET DIRECTION FORWARD
 002A 06                           PUSH    ES
 002B 1E                           PUSH    DS                      ; SAVE WORK AND PARAMETER REGISTERS
 002C 52                           PUSH    DX
 002D 51                           PUSH    CX
 002E 53                           PUSH    BX
 002F 56                           PUSH    SI
 0030 57                           PUSH    DI
 0031 55                           PUSH    BP
 0032 E8 0000 E                    CALL    DDS                     ; POINT DS: TO DATA SEGMENT
 0035 BE B800                      MOV     SI,0B800H               ; GET SEGMENT FOR COLOR CARD
 0038 8B 3E 0010 R                 MOV     DI,@EQUIP_FLAG          ; GET EQUIPMENT FLAGS SETTING
 003C 81 E7 0030                   AND     DI,30H                  ; ISOLATE CRT SWITCHES
 0040 83 FF 30                     CMP     DI,30H                  ; IS SETTING FOR BW CARD?
 0043 75 03                        JNE     M2                      ; SKIP IF NOT BW CARD
 0045 BE B000                      MOV     SI,0B000H               ; ELSE GET SEGMENT FOR BW CARD
 0048                      M2:
 0048 80 FC 13                     CMP     AH,13H                  ; TEST FOR WRITE STRING OPERATION
 004B 74 02                        JE      M3                      ; SKIP IF ES:BP VALID AS PASSED
```

```
004D 8E C6                      MOV     ES,SI                   ; SET UP TO POINT AT VIDEO MEMORY AREAS
004F                    M3:
004F 8B F0                      MOV     SI,AX                   ; MOVE COMMAND TO LOOK UP REGISTER
0051 C1 EE 08                   SHR     SI,8                    ; SHIFT COMMAND TO FORM BYTE OFFSET
0054 D1 E6                      SAL     SI,1                    ; TIMES 2 FOR WORD TABLE LOOKUP
0056 83 FE 28                   CMP     SI,MIL                  ; TEST FOR WITHIN TABLE RANGE
0059 73 09                      JNB     M4                      ; BRANCH TO EXIT IF NOT A VALID COMMAND

005B 8A 26 0049 R               MOV     AH,@CRT_MODE            ; MOVE CURRENT MODE INTO AH
005F 2E: FF A4 0000 R           JMP     WORD PTR CS:[SI+OFFSET MI]      ; GO TO SELECTED FUNCTION

0064                    M4:                                     ; COMMAND NOT VALID
0064 E9 0145 R                  JMP     VIDEO_RETURN            ; DO NOTHING IF NOT IN VALID RANGE
0067                    VIDEO_IO_1      ENDP
                        ;---------------------------------------------------------
                        ; SET_MODE                                              :
                        ;       THIS ROUTINE INITIALIZES THE ATTACHMENT TO      :
                        ;       THE SELECTED MODE.  THE SCREEN IS BLANKED.      :
                        ; INPUT                                                 :
                        ;       (AL) = MODE SELECTED (RANGE 0-7)                :
                        ; OUTPUT                                                :
                        ;       NONE                                            :
                        ;---------------------------------------------------------
0067                    SET_MODE        PROC    NEAR
0067 BA 03D4                    MOV     DX,03D4H                ; ADDRESS OF COLOR CARD
006A 8B 3E 0010 R               MOV     DI,@EQUIP_FLAG          ; GET EQUIPMENT FLAGS SETTING
006E 81 E7 0030                 AND     DI,30H                  ; ISOLATE CRT SWITCHES
0072 83 FF 30                   CMP     DI,30H                  ; IS BW CARD INSTALLED AS PRIMARY
0075 75 06                      JNE     M8C                     ; SKIP AND CHECK IF COLOR
0077 B0 07                      MOV     AL,7                    ; ELSE INDICATE INTERNAL BW CARD MODE
0079 B2 B4                      MOV     DL,0B4H                 ; SET ADDRESS OF BW (MONOCHROME) CARD
007B EB 0D                      JMP     SHORT M8                ; CONTINUE WITH FORCED MODE 7
007D                    M8C:
007D 3C 07                      CMP     AL,7                    ; CHECK FOR VALID COLOR MODES 0-6
007F 72 09                      JB      M8                      ; CONTINUE IF BELOW MODE 7
0081 B0 00                      MOV     AL,0                    ; FORCE DEFAULT 40x25 BW MODE
0083 83 FF 20                   CMP     DI,20H                  ; CHECK FOR @EQUIP_FLAG AT 80x25 BW
0086 74 02                      JE      M8                      ; CONTINUE WITH MODE 0 IF NOT
0088 B0 02                      MOV     AL,2                    ; ELSE FORCE MODE 2
008A                    M8:
008A A2 0049 R                  MOV     @CRT_MODE,AL            ; SAVE MODE IN GLOBAL VARIABLE
008D 89 16 0063 R               MOV     @ADDR_6845,DX           ; SAVE ADDRESS OF BASE
0091 C6 06 0084 R 18            MOV     @ROWS,25-1              ; INITIALIZE DEFAULT ROW COUNT OF 25
0096 1E                         PUSH    DS                      ; SAVE POINTER TO DATA SEGMENT
0097 50                         PUSH    AX                      ; SAVE MODE NUMBER (AL)
0098 98                         CBW                             ; CLEAR HIGH BYTE OF MODE
0099 8B F0                      MOV     SI,AX                   ; SET TABLE POINTER, INDEXED BY MODE
009B 2E: 8A 84 0000 E           MOV     AL,CS:[SI + OFFSET M7]  ; GET THE MODE SET VALUE FROM TABLE
00A0 A2 0065 R                  MOV     @CRT_MODE_SET,AL        ; SAVE THE MODE SET VALUE
00A3 24 37                      AND     AL,037H                 ; VIDEO OFF, SAVE HIGH RESOLUTION BIT
00A5 52                         PUSH    DX                      ; SAVE OUTPUT PORT VALUE
00A6 83 C2 04                   ADD     DX,4                    ; POINT TO CONTROL REGISTER
00A9 EE                         OUT     DX,AL                   ; RESET VIDEO TO OFF TO SUPPRESS ROLLING
00AA 5A                         POP     DX                      ; BACK TO BASE REGISTER
                                ASSUME  DS:ABS0
00AB 2B DB                      SUB     BX,BX                   ; SET UP FOR ABS0 SEGMENT
00AD 8E DB                      MOV     DS,BX                   ; ESTABLISH VECTOR TABLE ADDRESSING
00AF C5 1E 0074 R               LDS     BX,@PARM_PTR            ; GET POINTER TO VIDEO PARMS
                                ASSUME  DS:CODE
00B3 58                         POP     AX                      ; RECOVER MODE NUMBER IN (AL)
00B4 B9 0010                    MOV     CX,16                   ; LENGTH OF EACH ROW OF TABLE
00B7 3C 02                      CMP     AL,2                    ; DETERMINE WHICH ONE TO USE
00B9 72 0E                      JC      M9                      ; MODE IS 0 OR 1
00BB 03 D9                      ADD     BX,CX                   ; NEXT ROW OF INITIALIZATION TABLE
00BD 3C 04                      CMP     AL,4
00BF 72 08                      JC      M9                      ; MODE IS 2 OR 3
00C1 03 D9                      ADD     BX,CX                   ; MOVE TO GRAPHICS ROW OF INIT_TABLE
00C3 3C 07                      CMP     AL,7
00C5 72 02                      JC      M9                      ; MODE IS 4,5, OR 6
00C7 03 D9                      ADD     BX,CX                   ; MOVE TO BW CARD ROW OF INIT_TABLE

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

00C9                    M9:                                     ; OUT_INIT
00C9 50                         PUSH    AX                      ; SAVE MODE IN (AL)
00CA 8B 47 0A                   MOV     AX,[BX+10]              ; GET THE CURSOR MODE FROM THE TABLE
00CD 86 E0                      XCHG    AH,AL                   ; PUT CURSOR MODE IN CORRECT POSITION
00CF 1E                         PUSH    DS                      ; SAVE TABLE SEGMENT POINTER
                                ASSUME  DS:DATA
00D0 E8 0000 E                  CALL    DDS                     ; POINT DS TO DATA SEGMENT
00D3 A3 0060 R                  MOV     @CURSOR_MODE,AX         ; PLACE INTO BIOS DATA SAVE AREA
                                ASSUME  DS:CODE
00D6 1F                         POP     DS                      ; RESTORE THE TABLE SEGMENT POINTER
00D7 32 E4                      XOR     AH,AH                   ; AH IS REGISTER NUMBER DURING LOOP

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

00D9                    M10:                                    ;  INITIALIZATION LOOP
00D9 8A C4                      MOV     AL,AH                   ; GET 6845 REGISTER NUMBER
00DB EE                         OUT     DX,AL
00DC 42                         INC     DX                      ; POINT TO DATA PORT
00DD FE C4                      INC     AH                      ; NEXT REGISTER VALUE
00DF 8A 07                      MOV     AL,[BX]                 ; GET TABLE VALUE
00E1 EE                         OUT     DX,AL                   ; OUT TO CHIP
00E2 43                         INC     BX                      ; NEXT IN TABLE
00E3 4A                         DEC     DX                      ; BACK TO POINTER REGISTER
00E4 E2 F3                      LOOP    M10                     ; DO THE WHOLE TABLE
00E6 58                         POP     AX                      ; GET MODE BACK INTO (AL)
00E7 1F                         POP     DS                      ; RECOVER SEGMENT VALUE
                                ASSUME  DS:DATA

                        ;-----  FILL REGEN AREA WITH BLANK

00E8 33 FF                      XOR     DI,DI                   ; SET UP POINTER FOR REGEN
00EA 89 3E 004E R               MOV     @CRT_START,DI           ; START ADDRESS SAVED IN GLOBAL
00EE C6 06 0062 R 00            MOV     @ACTIVE_PAGE,0          ; SET PAGE VALUE
00F3 B9 2000                    MOV     CX,8192                 ; NUMBER OF WORDS IN COLOR CARD
00F6 3C 04                      CMP     AL,4                    ; TEST FOR GRAPHICS
00F8 72 0A                      JC      M12                     ; NO_GRAPHICS_INIT
00FA 3C 07                      CMP     AL,7                    ; TEST FOR BW CARD
00FC 74 04                      JE      M11                     ; BW_CARD_INIT
00FE 33 C0                      XOR     AX,AX                   ; FILL FOR GRAPHICS MODE
0100 EB 05                      JMP     SHORT M13               ; CLEAR_BUFFER
0102                    M11:                                    ; BW_CARD_INIT
```

SECTION 5

```
0102 B5 08                      MOV     CH,08H                  ; BUFFER SIZE ON BW CARD (2048)
0104                    M12:                                    ; NO_GRAPHICS_INIT
0104 B8 0720                    MOV     AX,' '+7*H              ; FILL CHAR FOR ALPHA + ATTRIBUTE
0107                    M13:                                    ; CLEAR_BUFFER
0107 F3/ AB                     REP     STOSW                   ; FILL THE REGEN BUFFER WITH BLANKS

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

0109 8B 16 0063 R               MOV     DX,@ADDR_6845           ; PREPARE TO OUTPUT TO VIDEO ENABLE PORT
010D 83 C2 04                   ADD     DX,4                    ; POINT TO THE MODE CONTROL REGISTER
0110 A0 0065 R                  MOV     AL,@CRT_MODE_SET        ; GET THE MODE SET VALUE
0113 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

0114 2E: 8A 84 0000 E           MOV     AL,CS:[SI + OFFSET M6]  ; GET NUMBER OF COLUMNS ON THIS SCREEN
0119 98                         CBW                             ; CLEAR HIGH BYTE
011A A3 004A R                  MOV     @CRT_COLS,AX            ; INITIALIZE NUMBER OF COLUMNS COUNT

                        ;-----  SET CURSOR POSITIONS

011D 81 E6 000E                 AND     SI,000EH                ; WORD OFFSET INTO CLEAR LENGTH TABLE
0121 2E: 8B 84 0000 E           MOV     AX,CS:[SI + OFFSET M5]  ; LENGTH TO CLEAR
0126 A3 004C R                  MOV     @CRT_LEN,AX             ; SAVE LENGTH OF CRT -- NOT USED FOR BW
0129 B9 0008                    MOV     CX,8                    ; CLEAR ALL CURSOR POSITIONS
012C BF 0050 R                  MOV     DI,OFFSET @CURSOR_POSN
012F 1E                         PUSH    DS                      ; ESTABLISH SEGMENT
0130 07                         POP     ES                      ;   ADDRESSING
0131 33 C0                      XOR     AX,AX
0133 F3/ AB                     REP     STOSW                   ; FILL WITH ZEROES

                        ;-----  SET UP OVERSCAN REGISTER

0135 42                         INC     DX                      ; SET OVERSCAN PORT TO A DEFAULT
0136 B0 30                      MOV     AL,30H                  ; 30H VALUE FOR ALL MODES EXCEPT 640X200
0138 80 3E 0049 R 06            CMP     @CRT_MODE,6             ; SEE IF THE MODE IS 640X200 BW
013D 75 02                      JNZ     M14                     ; IF NOT 640X200, THEN GO TO REGULAR
013F B0 3F                      MOV     AL,3FH                  ; IF IT IS 640X200, THEN PUT IN 3FH
0141                    M14:
0141 EE                         OUT     DX,AL                   ; OUTPUT THE CORRECT VALUE TO 3D9 PORT
0142 A2 0066 R                  MOV     @CRT_PALETTE,AL         ; SAVE THE VALUE FOR FUTURE USE

                        ;-----  NORMAL RETURN FROM ALL VIDEO RETURNS

0145                    VIDEO_RETURN:
0145 5D                         POP     BP
0146 5F                         POP     DI
0147 5E                         POP     SI
0148 5B                         POP     BX
0149                    M15:                                    ; VIDEO_RETURN_C
0149 59                         POP     CX
014A 5A                         POP     DX
014B 1F                         POP     DS
014C 07                         POP     ES                      ; RECOVER SEGMENTS
014D CF                         IRET                            ; ALL DONE
014E                    SET_MODE        ENDP
                        ;-----------------------------------------------
                        ; SET_CTYPE
                        ;       THIS ROUTINE SETS THE CURSOR VALUE
                        ; INPUT
                        ;       (CX) HAS CURSOR VALUE CH-START LINE, CL-STOP LINE
                        ; OUTPUT
                        ;       NONE
                        ;-----------------------------------------------
014E                    SET_CTYPE       PROC    NEAR
014E B4 0A                      MOV     AH,10                   ; 6845 REGISTER FOR CURSOR SET
0150 89 0E 0060 R               MOV     @CURSOR_MODE,CX         ; SAVE IN DATA AREA
0154 E8 0159 R                  CALL    M16                     ; OUTPUT CX REGISTER
0157 EB EC                      JMP ^   VIDEO_RETURN

                        ;-----  THIS ROUTINE OUTPUTS THE CX REGISTER TO THE 6845 REGISTERS NAMED IN (AH)


0159                    M16:
0159 8B 16 0063 R               MOV     DX,@ADDR_6845           ; ADDRESS REGISTER
015D 8A C4                      MOV     AL,AH                   ; GET VALUE
015F EE                         OUT     DX,AL                   ; REGISTER SET
0160 42                         INC     DX                      ; DATA REGISTER
0161 EB 00                      JMP     $+2                     ; I/O DELAY
0163 8A C5                      MOV     AL,CH                   ; DATA
0165 EE                         OUT     DX,AL
0166 4A                         DEC     DX
0167 8A C4                      MOV     AL,AH
0169 FE C0                      INC     AL                      ; POINT TO OTHER DATA REGISTER
016B EE                         OUT     DX,AL                   ; SET FOR SECOND REGISTER
016C 42                         INC     DX
016D EB 00                      JMP     $+2                     ; I/O DELAY
016F 8A C1                      MOV     AL,CL                   ; SECOND DATA VALUE
0171 EE                         OUT     DX,AL
0172 C3                         RET                             ; ALL DONE
0173                    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
                        ;------------------------------------------
0173                    SET_CPOS        PROC    NEAR
0173 8A C7                      MOV     AL,BH                   ; MOVE PAGE NUMBER TO WORK REGISTER
0175 98                         CBW                             ; CONVERT PAGE TO WORD VALUE
0176 D1 E0                      SAL     AX,1                    ; WORD OFFSET
0178 96                         XCHG    AX,SI                   ; USE INDEX REGISTER
0179 89 94 0050 R               MOV     [SI+OFFSET @CURSOR_POSN],DX     ; SAVE THE POINTER
017D 38 3E 0062 R               CMP     @ACTIVE_PAGE,BH
0181 75 05                      JNZ     M17                     ; SET_CPOS_RETURN
0183 8B C2                      MOV     AX,DX                   ; GET ROW/COLUMN TO AX
0185 E8 018A R                  CALL    M18                     ; CURSOR_SET
0188                    M17:                                    ; SET_CPOS_RETURN
```

```
0188 EB BB                              JMP     VIDEO_RETURN
018A                            SET_CPOS        ENDP

                                ;----- SET CURSOR POSITION, AX HAS ROW/COLUMN FOR CURSOR

018A                            M18     PROC    NEAR
018A E8 020E R                          CALL    POSITION                ; DETERMINE LOCATION IN REGEN BUFFER
018D 8B C8                              MOV     CX,AX
018F 03 0E 004E R                       ADD     CX,@CRT_START           ; ADD IN THE START ADDRESS FOR THIS PAGE
0193 D1 F9                              SAR     CX,1                    ; DIVIDE BY 2 FOR CHAR ONLY COUNT
0195 B4 0E                              MOV     AH,14                   ; REGISTER NUMBER FOR CURSOR
0197 E8 0159 R                          CALL    M16                     ; OUTPUT THE VALUE TO THE 6845
019A C3                                 RET
019B                            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
                                ;-----------------------------------------
019B                            READ_CURSOR     PROC    NEAR
019B 8A DF                              MOV     BL,BH
019D 32 FF                              XOR     BH,BH
019F D1 E3                              SAL     BX,1                    ; WORD OFFSET
01A1 8B 97 0050 R                       MOV     DX,[BX+OFFSET @CURSOR_POSN]
01A5 8B 0E 0060 R                       MOV     CX,@CURSOR_MODE
01A9 5D                                 POP     BP
01AA 5F                                 POP     DI
01AB 5E                                 POP     SI
01AC 5B                                 POP     BX
01AD 58                                 POP     AX                      ; DISCARD SAVED CX AND DX
01AE 58                                 POP     AX
01AF 1F                                 POP     DS
01B0 07                                 POP     ES
01B1 CF                                 IRET
01B2                            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
                                ;--------------------------------------------------
01B2                            ACT_DISP_PAGE   PROC    NEAR
01B2 A2 0062 R                          MOV     @ACTIVE_PAGE,AL         ; SAVE ACTIVE PAGE VALUE
01B5 8B 0E 004C R                       MOV     CX,@CRT_LEN             ; GET SAVED LENGTH OF REGEN BUFFER
01B9 98                                 CBW                             ; CONVERT AL TO WORD
01BA 50                                 PUSH    AX                      ; SAVE PAGE VALUE
01BB F7 E1                              MUL     CX                      ; DISPLAY PAGE TIMES REGEN LENGTH
01BD A3 004E R                          MOV     @CRT_START,AX           ; SAVE START ADDRESS FOR LATER
01C0 8B C8                              MOV     CX,AX                   ; START ADDRESS TO CX
01C2 D1 F9                              SAR     CX,1                    ; DIVIDE BY 2 FOR 6845 HANDLING
01C4 B4 0C                              MOV     AH,12                   ; 6845 REGISTER FOR START ADDRESS
01C6 E8 0159 R                          CALL    M16
01C9 5B                                 POP     BX                      ; RECOVER PAGE VALUE
01CA D1 E3                              SAL     BX,1                    ; *2 FOR WORD OFFSET
01CC 8B 87 0050 R                       MOV     AX,[BX + OFFSET @CURSOR_POSN]   ; GET CURSOR FOR THIS PAGE
01D0 E8 018A R                          CALL    M18                     ; SET THE CURSOR POSITION
01D3 E9 0145 R                          JMP     VIDEO_RETURN
01D6                            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
                                ;----------------------------------------------
01D6                            SET_COLOR       PROC    NEAR
01D6 8B 16 0063 R                       MOV     DX,@ADDR_6845           ; I/O PORT FOR PALETTE
01DA 83 C2 05                           ADD     DX,5                    ; OVERSCAN PORT
01DD A0 0066 R                          MOV     AL,@CRT_PALETTE         ; GET THE CURRENT PALETTE VALUE
01E0 0A FF                              OR      BH,BH                   ; IS THIS COLOR 0?
01E2 75 0E                              JNZ     M20                     ; OUTPUT COLOR 1

                                ;----- HANDLE COLOR 0 BY SETTING THE BACKGROUND COLOR

01E4 24 E0                              AND     AL,0E0H                 ; TURN OFF LOW 5 BITS OF CURRENT
01E6 80 E3 1F                           AND     BL,01FH                 ; TURN OFF HIGH 3 BITS OF INPUT VALUE
01E9 0A C3                              OR      AL,BL                   ; PUT VALUE INTO REGISTER
01EB                            M19:                                    ; OUTPUT THE PALETTE
01EB EE                                 OUT     DX,AL                   ; OUTPUT COLOR SELECTION TO 3D9 PORT
01EC A2 0066 R                          MOV     @CRT_PALETTE,AL         ; SAVE THE COLOR VALUE
01EF E9 0145 R                          JMP     VIDEO_RETURN

                                ;----- HANDLE COLOR 1 BY SELECTING THE PALETTE TO BE USED

01F2                            M20:
01F2 24 DF                              AND     AL,0DFH                 ; TURN OFF PALETTE SELECT BIT
01F4 D0 EB                              SHR     BL,1                    ; TEST THE LOW ORDER BIT OF BL
01F6 73 F3                              JNC     M19                     ; ALREADY DONE
01F8 0C 20                              OR      AL,20H                  ; TURN ON PALETTE SELECT BIT
01FA EB EF                              JMP     M19                     ; GO DO IT
01FC                            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
```

```
                                ;-------------------------------------------
 01FC                           VIDEO_STATE     PROC    NEAR
 01FC 8A 26 004A R                      MOV     AH,BYTE PTR @CRT_COLS   ; GET NUMBER OF COLUMNS
 0200 A0 0049 R                         MOV     AL,@CRT_MODE            ; CURRENT MODE
 0203 8A 3E 0062 R                      MOV     BH,@ACTIVE_PAGE         ; GET CURRENT ACTIVE PAGE
 0207 5D                                POP     BP                      ; RECOVER REGISTERS
 0208 5F                                POP     DI
 0209 5E                                POP     SI
 020A 59                                POP     CX                      ; DISCARD SAVED BX
 020B E9 0149 R                         JMP     M15                     ; RETURN TO CALLER
 020E                           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
                                ;---------------------------------------
 020E                           POSITION        PROC    NEAR
 020E 53                                PUSH    BX                      ; SAVE REGISTER
 020F 8B D8                             MOV     BX,AX
 0211 8A C4                             MOV     AL,AH                   ; ROWS TO AL
 0213 F6 26 004A R                      MUL     BYTE PTR @CRT_COLS      ; DETERMINE BYTES TO ROW
 0217 32 FF                             XOR     BH,BH
 0219 03 C3                             ADD     AX,BX                   ; ADD IN COLUMN VALUE
 021B D1 E0                             SAL     AX,1                    ; * 2 FOR ATTRIBUTE BYTES
 021D 5B                                POP     BX
 021E C3                                RET
 021F                           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
 021F                           SCROLL_UP       PROC    NEAR

 021F E8 02FB R                         CALL    TEST_LINE_COUNT
 0222 80 FC 04                          CMP     AH,4                    ; TEST FOR GRAPHICS MODE
 0225 72 08                             JC      N1                      ; HANDLE SEPARATELY
 0227 80 FC 07                          CMP     AH,7                    ; TEST FOR BW CARD
 022A 74 03                             JE      N1
 022C E9 04BA R                         JMP     GRAPHICS_UP
 022F                           N1:                                     ; UP_CONTINUE
 022F 53                                PUSH    BX                      ; SAVE FILL ATTRIBUTE IN BH
 0230 8B C1                             MOV     AX,CX                   ; UPPER LEFT POSITION
 0232 E8 026C R                         CALL    SCROLL_POSITION         ; DO SETUP FOR SCROLL
 0235 74 31                             JZ      N7                      ; BLANK_FIELD
 0237 03 F0                             ADD     SI,AX                   ; FROM ADDRESS
 0239 8A E6                             MOV     AH,DH                   ; # ROWS IN BLOCK
 023B 2A E3                             SUB     AH,BL                   ; # ROWS TO BE MOVED
 023D                           N2:                                     ; ROW_LOOP
 023D E8 02AE R                         CALL    N10                     ; MOVE ONE ROW
 0240 03 F5                             ADD     SI,BP
 0242 03 FD                             ADD     DI,BP                   ; POINT TO NEXT LINE IN BLOCK
 0244 FE CC                             DEC     AH                      ; COUNT OF LINES TO MOVE
 0246 75 F5                             JNZ     N2                      ; ROW_LOOP
 0248                           N3:                                     ; CLEAR_ENTRY
 0248 58                                POP     AX                      ; RECOVER ATTRIBUTE IN AH
 0249 B0 20                             MOV     AL,' '                  ; FILL WITH BLANKS
 024B                           N4:                                     ; CLEAR_LOOP
 024B E8 02B7 R                         CALL    N11                     ; CLEAR THE ROW
 024E 03 FD                             ADD     DI,BP                   ; POINT TO NEXT LINE
 0250 FE CB                             DEC     BL                      ; COUNTER OF LINES TO SCROLL
 0252 75 F7                             JNZ     N4                      ; CLEAR_LOOP
 0254                           N5:                                     ; SCROLL_END
 0254 E8 0000 E                         CALL    DDS
 0257 80 3E 0049 R 07                   CMP     @CRT_MODE,7             ; IS THIS THE BLACK AND WHITE CARD
 025C 74 07                             JE      N6                      ; IF SO, SKIP THE MODE RESET
 025E A0 0065 R                         MOV     AL,@CRT_MODE_SET        ; GET THE VALUE OF THE MODE SET
 0261 BA 03D8                           MOV     DX,03D8H                ; ALWAYS SET COLOR CARD PORT
 0264 EE                                OUT     DX,AL
 0265                           N6:                                     ; VIDEO_RET_HERE
 0265 E9 0145 R                         JMP     VIDEO_RETURN
 0268                           N7:                                     ; BLANK_FIELD
 0268 8A DE                             MOV     BL,DH                   ; GET ROW COUNT
 026A EB DC                             JMP     N3                      ; GO CLEAR THAT AREA
 026C                           SCROLL_UP       ENDP

                                ;-----  HANDLE COMMON SCROLL SET UP HERE

 026C                           SCROLL_POSITION PROC    NEAR
 026C E8 020E R                         CALL    POSITION                ; CONVERT TO REGEN POINTER
 026F 03 06 004E R                      ADD     AX,@CRT_START           ; OFFSET OF ACTIVE PAGE
 0273 8B F8                             MOV     DI,AX                   ; TO ADDRESS FOR SCROLL
 0275 8B F0                             MOV     SI,AX                   ; FROM ADDRESS FOR SCROLL
 0277 2B D1                             SUB     DX,CX                   ; DX = #ROWS, #COLS IN BLOCK
 0279 FE C6                             INC     DH
 027B FE C2                             INC     DL                      ; INCREMENT FOR 0 ORIGIN
 027D 32 ED                             XOR     CH,CH                   ; SET HIGH BYTE OF COUNT TO ZERO
 027F 8B 2E 004A R                      MOV     BP,@CRT_COLS            ; GET NUMBER OF COLUMNS IN DISPLAY
 0283 03 ED                             ADD     BP,BP                   ; TIMES 2 FOR ATTRIBUTE BYTE
 0285 8A C3                             MOV     AL,BL                   ; GET LINE COUNT
 0287 F6 26 004A R                      MUL     BYTE PTR @CRT_COLS      ; DETERMINE OFFSET TO FROM ADDRESS
 028B 03 C0                             ADD     AX,AX                   ; *2 FOR ATTRIBUTE BYTE
 028D 50                                PUSH    AX                      ; SAVE LINE COUNT
 028E A0 0049 R                         MOV     AL,@CRT_MODE            ; GET CURRENT MODE
 0291 06                                PUSH    ES                      ; ESTABLISH ADDRESSING TO REGEN BUFFER
 0292 1F                                POP     DS                      ;  FOR BOTH POINTERS
 0293 3C 02                             CMP     AL,2                    ; TEST FOR COLOR CARD SPECIAL CASES HERE
 0295 72 13                             JB      N9                      ; HAVE TO HANDLE 80X25 SEPARATELY
 0297 3C 03                             CMP     AL,3
```

```
0299 77 0F                      JA      N9
                        ;-----                                  ; 80X25 COLOR CARD SCROLL
029B 52                         PUSH    DX
029C BA 03DA                    MOV     DX,3DAH                 ; GUARANTEED TO BE COLOR CARD HERE
029F                    N8:                                     ; WAIT_DISP_ENABLE
029F EC                         IN      AL,DX                   ; GET PORT
02A0 A8 08                      TEST    AL,RVRT                 ; WAIT FOR VERTICAL RETRACE
02A2 74 FB                      JZ      N8                      ; WAIT_DISP_ENABLE
02A4 B0 25                      MOV     AL,25H
02A6 B2 D8                      MOV     DL,0D8H                 ; ADDRESS CONTROL PORT
02A8 EE                         OUT     DX,AL                   ; TURN OFF VIDEO DURING VERTICAL RETRACE
02A9 5A                         POP     DX
02AA                    N9:
02AA 58                         POP     AX                      ; RESTORE LINE COUNT
02AB 0A DB                      OR      BL,BL                   ; 0 SCROLL MEANS BLANK FIELD
02AD C3                         RET                             ; RETURN WITH FLAGS SET
02AE                    SCROLL_POSITION ENDP

                        ;-----  MOVE_ROW
02AE                    N10     PROC    NEAR
02AE 8A CA                      MOV     CL,DL                   ; GET # OF COLS TO MOVE
02B0 56                         PUSH    SI
02B1 57                         PUSH    DI                      ; SAVE START ADDRESS
02B2 F3/ A5                     REP     MOVSW                   ; MOVE THAT LINE ON SCREEN
02B4 5F                         POP     DI
02B5 5E                         POP     SI                      ; RECOVER ADDRESSES
02B6 C3                         RET
02B7                    N10     ENDP

                        ;-----  CLEAR_ROW
02B7                    N11     PROC    NEAR
02B7 8A CA                      MOV     CL,DL                   ; GET # COLUMNS TO CLEAR
02B9 57                         PUSH    DI
02BA F3/ AB                     REP     STOSW                   ; STORE THE FILL CHARACTER
02BC 5F                         POP     DI
02BD C3                         RET
02BE                    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
                        ;----------------------------------------------
02BE                    SCROLL_DOWN     PROC    NEAR
02BE FD                         STD                             ; DIRECTION FOR SCROLL DOWN
02BF E8 02FB R                  CALL    TEST_LINE_COUNT         ;
02C2 80 FC 04                   CMP     AH,4                    ; TEST FOR GRAPHICS
02C5 72 08                      JC      N12
02C7 80 FC 07                   CMP     AH,7                    ; TEST FOR BW CARD
02CA 74 03                      JE      N12
02CC E9 0511 R                  JMP     GRAPHICS_DOWN
02CF                    N12:                                    ; CONTINUE_DOWN
02CF 53                         PUSH    BX                      ; SAVE ATTRIBUTE IN BH
02D0 8B C2                      MOV     AX,DX                   ; LOWER RIGHT CORNER
02D2 E8 026C R                  CALL    SCROLL_POSITION         ; GET REGEN LOCATION
02D5 74 20                      JZ      N16
02D7 2B F0                      SUB     SI,AX                   ; SI IS FROM ADDRESS
02D9 8A E6                      MOV     AH,DH                   ; GET TOTAL # ROWS
02DB 2A E3                      SUB     AH,BL                   ; COUNT TO MOVE IN SCROLL
02DD                    N13:
02DD E8 02AE R                  CALL    N10                     ; MOVE ONE ROW
02E0 2B F5                      SUB     SI,BP
02E2 2B FD                      SUB     DI,BP
02E4 FE CC                      DEC     AH
02E6 75 F5                      JNZ     N13
02E8                    N14:
02E8 58                         POP     AX                      ; RECOVER ATTRIBUTE IN AH
02E9 B0 20                      MOV     AL,' '
02EB                    N15:
02EB E8 02B7 R                  CALL    N11                     ; CLEAR ONE ROW
02EE 2B FD                      SUB     DI,BP                   ; GO TO NEXT ROW
02F0 FE CB                      DEC     BL
02F2 75 F7                      JNZ     N15
02F4 E9 0254 R                  JMP     N5                      ; SCROLL_END
02F7                    N16:
02F7 8A DE                      MOV     BL,DH
02F9 EB ED                      JMP     N14
02FB                    SCROLL_DOWN     ENDP

                        ;-----  IF  AMOUNT OF LINES TO BE SCROLLED = AMOUNT OF LINES IN WINDOW
                        ;-----   THEN  ADJUST AL; ELSE  RETURN;

02FB                    TEST_LINE_COUNT PROC    NEAR

02FB 8A D8                      MOV     BL,AL                   ; SAVE LINE COUNT IN BL
02FD 0A C0                      OR      AL,AL                   ; TEST IF AL IS ALREADY ZERO
02FF 74 0E                      JZ      BL_SET                  ; IF IT IS THEN RETURN...
0301 50                         PUSH    AX                      ; SAVE AX
0302 8A C6                      MOV     AL,DH                   ; SUBTRACT LOWER ROW FROM UPPER ROW
0304 2A C5                      SUB     AL,CH
0306 FE C0                      INC     AL                      ; ADJUST DIFFERENCE BY 1
0308 3A C3                      CMP     AL,BL                   ; LINE COUNT = AMOUNT OF ROWS IN WINDOW?
030A 58                         POP     AX                      ; RESTORE AX
030B 75 02                      JNE     BL_SET                  ; IF NOT THEN WE'RE ALL SET
030D 2A DB                      SUB     BL,BL                   ; OTHERWISE SET BL TO ZERO
030F                    BL_SET:
030F C3                         RET                             ; RETURN
0310                    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

 0310                           READ_AC_CURRENT PROC    NEAR
 0310  80 FC 04                         CMP     AH,4                    ; IS THIS GRAPHICS
 0313  72 08                            JC      P10

 0315  80 FC 07                         CMP     AH,7                    ; IS THIS BW CARD
 0318  74 03                            JE      P10

 031A  E9 064A R                        JMP     GRAPHICS_READ
 031D                           P10:                                    ;  READ_AC_CONTINUE
 031D  E8 0339 R                        CALL    FIND_POSITION           ; GET REGEN LOCATION AND PORT ADDRESS
 0320  8B F7                            MOV     SI,DI                   ; ESTABLISH ADDRESSING IN SI
 0322  06                               PUSH    ES                      ; GET REGEN SEGMENT FOR QUICK ACCESS
 0323  1F                               POP     DS

                                ;-----  WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80

 0324  0A DB                            OR      BL,BL                   ; CHECK MODE FLAG FOR COLOR CARD IN 80
 0326  75 0D                            JNZ     P13                     ; ELSE SKIP RETRACE WAIT - DO FAST READ
 0328                           P11:                                    ;  WAIT FOR HORZ RETRACE LOW OR VERTICAL
 0328  FB                               STI                             ; ENABLE INTERRUPTS FIRST
 0329  90                               NOP                             ; ALLOW FOR SMALL INTERRUPT WINDOW
 032A  FA                               CLI                             ; BLOCK INTERRUPTS FOR SINGLE LOOP
 032B  EC                               IN      AL,DX                   ; GET STATUS FROM THE ADAPTER
 032C  A8 01                            TEST    AL,RHRZ                 ; IS HORIZONTAL RETRACE LOW
 032E  75 F8                            JNZ     P11                     ; WAIT UNTIL IT IS
 0330                           P12:                                    ;  NOW WAIT FOR EITHER RETRACE HIGH
 0330  EC                               IN      AL,DX                   ; GET STATUS
 0331  A8 09                            TEST    AL,RVRT+RHRZ            ; IS HORIZONTAL OR VERTICAL RETRACE HIGH
 0333  74 FB                            JZ      P12                     ; WAIT UNTIL EITHER IS ACTIVE
 0335                           P13:
 0335  AD                               LODSW                           ; GET THE CHARACTER AND ATTRIBUTE
 0336  E9 0145 R                        JMP     VIDEO_RETURN            ;  EXIT WITH (AX)

 0339                           READ_AC_CURRENT ENDP


 0339                           FIND_POSITION   PROC    NEAR            ;       SETUP FOR BUFFER READ OR WRITE
 0339  86 E3                            XCHG    AH,BL                   ; SWAP MODE TYPE WITH ATTRIBUTE
 033B  8B E8                            MOV     BP,AX                   ; SAVE CHARACTER/ATTR IN (BP) REGISTER
 033D  80 EB 02                         SUB     BL,2                    ; CONVERT DISPLAY MODE TYPE TO A
 0340  D0 EB                            SHR     BL,1                    ;  ZERO VALUE FOR COLOR IN 80 COLUMN
 0342  8B F3                            MOV     SI,BX                   ;  AND SAVE (2 OR 3 --> ZERO)
 0344  8A DF                            MOV     BL,BH                   ; MOVE DISPLAY PAGE TO LOW BYTE
 0346  32 FF                            XOR     BH,BH                   ; CLEAR HIGH BYTE OF COUNT/BYTE OFFSET
 0348  8B FB                            MOV     DI,BX                   ; MOVE DISPLAY PAGE (COUNT) TO WORK REG
 034A  D1 E7                            SAL     DI,1                    ; TIMES 2 FOR WORD OFFSET
 034C  8B 85 0050 R                     MOV     AX,[DI+OFFSET @CURSOR_POSN]     ; GET ROW/COLUMN OF THAT PAGE
 0350  74 09                            JZ      P21                     ; SKIP BUFFER ADJUSTMENT IF PAGE ZERO

 0352  33 FF                            XOR     DI,DI                   ; ELSE SET BUFFER START ADDRESS TO ZERO
 0354                           P20:
 0354  03 3E 004C R                     ADD     DI,@CRT_LEN             ; ADD LENGTH OF BUFFER FOR ONE PAGE
 0358  4B                               DEC     BX                      ; DECREMENT PAGE COUNT
 0359  75 F9                            JNZ^    P20                     ; LOOP TILL PAGE COUNT EXHAUSTED
 035B                           P21:
 035B  E8 020E R                        CALL    POSITION                ; DETERMINE LOCATION IN REGEN IN PAGE
 035E  03 F8                            ADD     DI,AX                   ; ADD LOCATION TO START OF REGEN PAGE
 0360  8B 16 0063 R                     MOV     DX,@ADDR_6845           ; GET BASE ADDRESS OF ACTIVE DISPLAY
 0364  83 C2 06                         ADD     DX,6                    ; POINT AT STATUS PORT
 0367  8B DE                            MOV     BX,SI                   ; RECOVER CONVERTED MODE TYPE IN (BL)
 0369  C3                               RET                             ;  BP= ATTRIBUTE/CHARACTER (FROM BL/AL)
                                                                        ;  DI= POSITION (OFFSET IN REGEN BUFFER)
                                                                        ;  DX= STATUS PORT ADDRESS OF ADAPTER
 036A                           FIND_POSITION   ENDP                    ;  BL= MODE FLAG (ZERO FOR 80X25 COLOR)
```

```
                        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                                  :
                        ;---------------------------------------------------------------------

036A                    WRITE_AC_CURRENT        PROC    NEAR
036A 80 FC 04                   CMP     AH,4                    ; IS THIS GRAPHICS
036D 72 08                      JC      P30
036F 80 FC 07                   CMP     AH,7                    ; IS THIS BW CARD
0372 74 03                      JE      P30
0374 E9 0599 R                  JMP     GRAPHICS_WRITE
0377                    P30:                                    ;  WRITE_AC_CONTINUE
0377 E8 0339 R                  CALL    FIND_POSITION           ; GET REGEN LOCATION AND PORT ADDRESS
                                                                ; ADDRESS IN (DI) REGISTER
037A 0A DB                      OR      BL,BL                   ; CHECK MODE FLAG FOR COLOR CARD AT 80
037C 74 06                      JZ      P32                     ; SKIP TO RETRACE WAIT IF COLOR AT 80

037E 95                         XCHG    AX,BP                   ; GET THE ATTR/CHAR SAVED FOR FAST WRITE
037F F3/ AB                     REP     STOSW                   ; STRING WRITE THE ATTRIBUTE & CHARACTER
0381 EB 16                      JMP     SHORT  P35              ; EXIT FAST WRITE ROUTINE

                        ;-----  WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80

0383                    P31:                                    ;  LOOP FOR EACH ATTR/CHAR WRITE
0383 95                         XCHG    BP,AX                   ; PLACE ATTR/CHAR BACK IN SAVE REGISTER
0384                    P32:                                    ;  WAIT FOR HORZ RETRACE LOW OR VERTICAL
0384 FB                         STI                             ; ENABLE INTERRUPTS FIRST
0385 90                         NOP                             ; ALLOW FOR INTERRUPT WINDOW
0386 FA                         CLI                             ; BLOCK INTERRUPTS FOR SINGLE LOOP
0387 EC                         IN      AL,DX                   ; GET STATUS FROM THE ADAPTER
0388 A8 08                      TEST    AL,RVRT                 ; CHECK FOR VERTICAL RETRACE FIRST
038A 75 09                      JNZ     P34                     ; DO FAST WRITE NOW IF VERTICAL RETRACE
038C A8 01                      TEST    AL,RHRZ                 ; IS HORIZONTAL RETRACE LOW THEN
038E 75 F4                      JNZ     P32                     ; WAIT UNTIL IT IS
0390                    P33:                                    ;  WAIT FOR EITHER RETRACE HIGH
0390 EC                         IN      AL,DX                   ; GET STATUS AGAIN
0391 A8 09                      TEST    AL,RVRT+RHRZ            ; IS HORIZONTAL OR VERTICAL RETRACE HIGH
0393 74 FB                      JZ      P33                     ; WAIT UNTIL EITHER IS ACTIVE
0395                    P34:
0395 95                         XCHG    AX,BP                   ; GET THE ATTR/CHAR SAVED IN (BP)
0396 AB                         STOSW                           ; WRITE THE ATTRIBUTE AND CHARACTER
0397 E2 EA                      LOOP    P31                     ; AS MANY TIMES AS REQUESTED - TILL CX=0
0399                    P35:
0399 E9 0145 R                  JMP     VIDEO_RETURN            ; EXIT

039C                    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                                  :
                        ;---------------------------------------------------------------------

039C                    WRITE_C_CURRENT PROC    NEAR
039C 80 FC 04                   CMP     AH,4                    ; IS THIS GRAPHICS
039F 72 08                      JC      P40
03A1 80 FC 07                   CMP     AH,7                    ; IS THIS BW CARD
03A4 74 03                      JE      P40
03A6 E9 0599 R                  JMP     GRAPHICS_WRITE
03A9                    P40:
03A9 E8 0339 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

03AC                    P41:                                    ;  WAIT FOR HORZ RETRACE LOW OR VERTICAL
03AC FB                         STI                             ; ENABLE INTERRUPTS FIRST
03AD 0A DB                      OR      BL,BL                   ; CHECK MODE FLAG FOR COLOR CARD IN 80
03AF 75 0F                      JNZ     P43                     ; ELSE SKIP RETRACE WAIT - DO FAST WRITE
03B1 FA                         CLI                             ; BLOCK INTERRUPTS FOR SINGLE LOOP
03B2 EC                         IN      AL,DX                   ; GET STATUS FROM THE ADAPTER
03B3 A8 08                      TEST    AL,RVRT                 ; CHECK FOR VERTICAL RETRACE FIRST
03B5 75 09                      JNZ     P43                     ; DO FAST WRITE NOW IF VERTICAL RETRACE
03B7 A8 01                      TEST    AL,RHRZ                 ; IS HORIZONTAL RETRACE LOW THEN
03B9 75 F1                      JNZ     P41                     ; WAIT UNTIL IT IS
03BB                    P42:                                    ;  WAIT FOR EITHER RETRACE HIGH
03BB EC                         IN      AL,DX                   ; GET STATUS AGAIN
03BC A8 09                      TEST    AL,RVRT+RHRZ            ; IS HORIZONTAL OR VERTICAL RETRACE HIGH
03BE 74 FB                      JZ      P42                     ; WAIT UNTIL EITHER RETRACE ACTIVE
03C0                    P43:
03C0 8B C5                      MOV     AX,BP                   ; GET THE CHARACTER SAVE IN (BP)
03C2 AA                         STOSB                           ; PUT THE CHARACTER INTO REGEN BUFFER
03C3 47                         INC     DI                      ; BUMP POINTER PAST ATTRIBUTE
03C4 E2 E6                      LOOP    P41                     ;  AS MANY TIMES AS REQUESTED

03C6 E9 0145 R                  JMP     VIDEO_RETURN

03C9                    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                                                :
                                 ;       (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   :
                                 ;       (ES) = SOURCE STRING SEGMENT                                       :
                                 ;       (BP) = SOURCE STRING OFFSET                                        :
                                 ; OUTPUT                                                                   :
                                 ;       NONE                                                               :
                                 ;--------------------------------------------------------------------------
 03C9                            WRITE_STRING    PROC    NEAR
 03C9 3C 04                              CMP     AL,04                   ; TEST FOR INVALID WRITE STRING OPTION
 03CB 73 7C                              JNB     P59                     ; IF OPTION INVALID THEN RETURN
 03CD E3 7A                              JCXZ    P59                     ; IF ZERO LENGTH STRING THEN RETURN

 03CF 8B F3                              MOV     SI,BX                   ; GET CURRENT CURSOR PAGE
 03D1 C1 EE 08                           SHR     SI,8                    ; CLEAR HIGH BYTE
 03D4 D1 E6                              SAL     SI,1                    ; CONVERT TO PAGE OFFSET  (SI= PAGE)
 03D6 FF B4 0050 R                       PUSH    [SI+OFFSET @CURSOR_POSN]; SAVE CURRENT CURSOR POSITION IN STACK
 03DA 50                                 PUSH    AX                      ; SAVE WRITE STRING OPTION
 03DB B8 0200                            MOV     AX,0200H                ; SET NEW CURSOR POSITION
 03DE CD 10                              INT     10H
 03E0 58                                 POP     AX                      ; RESTORE WRITE STRING OPTION
 03E1                            P50:
 03E1 51                                 PUSH    CX
 03E2 53                                 PUSH    BX
 03E3 50                                 PUSH    AX
 03E4 86 E0                              XCHG    AH,AL                   ; PUT THE WRITE STRING OPTION INTO (AH)
 03E6 26: 8A 46 00                       MOV     AL,ES:[BP]              ; GET CHARACTER FROM INPUT STRING
 03EA 45                                 INC     BP                      ; BUMP POINTER TO CHARACTER

                                 ;-----  TEST FOR SPECIAL CHARACTER'S

 03EB 3C 08                              CMP     AL,08H                  ; IS IT A BACKSPACE
 03ED 74 0C                              JE      P51                     ; BACK_SPACE
 03EF 3C 0D                              CMP     AL,CR                   ; IS IT CARRIAGE RETURN
 03F1 74 08                              JE      P51                     ; CAR_RET
 03F3 3C 0A                              CMP     AL,LF                   ; IS IT A LINE FEED
 03F5 74 04                              JE      P51                     ; LINE_FEED
 03F7 3C 07                              CMP     AL,07H                  ; IS IT A BELL
 03F9 75 0D                              JNE     P52                     ; IF NOT THEN DO WRITE CHARACTER
 03FB                            P51:
 03FB B4 0E                              MOV     AH,0EH                  ; TTY_CHARACTER WRITE
 03FD CD 10                              INT     10H                       ; WRITE TTY CHARACTER TO THE CRT
 03FF 8B 94 0050 R                       MOV     DX,[SI+OFFSET @CURSOR_POSN] ; GET CURRENT CURSOR POSITION
 0403 58                                 POP     AX                        ; RESTORE REGISTERS
 0404 5B                                 POP     BX
 0405 59                                 POP     CX
 0406 EB 2E                              JMP     SHORT P54               ; GO SET CURSOR POSITION AND CONTINUE
 0408                            P52:
 0408 B9 0001                            MOV     CX,1                    ; SET CHARACTER WRITE AMOUNT TO ONE
 040B 80 FC 02                           CMP     AH,2                    ; IS THE ATTRIBUTE IN THE STRING
 040E 72 05                              JB      P53                     ; IF NOT THEN SKIP
 0410 26: 8A 5E 00                       MOV     BL,ES:[BP]              ; ELSE GET NEW ATTRIBUTE
 0414 45                                 INC     BP                      ; BUMP STRING POINTER
 0415                            P53:
 0415 B4 09                              MOV     AH,09H                  ;  GOT_CHARACTER
 0417 CD 10                              INT     10H                     ; WRITE CHARACTER TO THE CRT
 0419 58                                 POP     AX                      ; RESTORE REGISTERS
 041A 5B                                 POP     BX
 041B 59                                 POP     CX
 041C FE C2                              INC     DL                      ; INCREMENT COLUMN COUNTER
 041E 3A 16 004A R                       CMP     DL,BYTE PTR @CRT_COLS   ; IF COLS ARE WITHIN RANGE FOR THIS MODE
 0422 72 12                              JB      P54                     ;     THEN GO TO COLUMNS SET
 0424 FE C6                              INC     DH                      ; BUMP ROW COUNTER BY ONE
 0426 2A D2                              SUB     DL,DL                   ; SET COLUMN COUNTER TO ZERO
 0428 80 FE 19                           CMP     DH,25                   ; IF ROWS ARE LESS THAN 25 THEN
 042B 72 09                              JB      P54                     ;    GO TO ROWS_COLUMNS_SET

 042D 50                                 PUSH    AX                      ; ELSE SCROLL SCREEN
 042E B8 0E0A                            MOV     AX,0E0AH                ; DO SCROLL ONE LINE
 0431 CD 10                              INT     10H                     ; RESET ROW COUNTER TO 24
 0433 FE CE                              DEC     DH
 0435 58                                 POP     AX                      ; RESTORE REGISTERS
 0436                            P54:                                    ;  ROW_COLUMNS_SET
 0436 50                                 PUSH    AX                      ; SAVE WRITE STRING OPTION
 0437 B8 0200                            MOV     AX,0200H                ; SET NEW CURSOR POSITION COMMAND
 043A CD 10                              INT     10H                     ; ESTABLISH NEW CURSOR POSITION
 043C 58                                 POP     AX
 043D E2 A2                              LOOP    P50                     ; DO IT ONCE MORE UNTIL (CX) = ZERO

 043F 5A                                 POP     DX                      ; RESTORE OLD CURSOR COORDINATES
 0440 A8 01                              TEST    AL,01H                  ; IF CURSOR WAS NOT TO BE MOVED THEN
 0442 75 05                              JNZ     P59                     ;  THEN EXIT WITHOUT RESETTING OLD VALUE
 0444 B8 0200                            MOV     AX,0200H                ; ELSE RESTORE OLD CURSOR POSITION
 0447 CD 10                              INT     10H
 0449                            P59:                                    ;  DONE - EXIT WRITE STRING
 0449 E9 0145 R                          JMP     VIDEO_RETURN            ; RETURN TO CALLER

 044C                            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
 044C                           READ_DOT        PROC    NEAR
 044C E8 0480 R                         CALL    R3                      ; DETERMINE BYTE POSITION OF DOT
 044F 26: 8A 04                         MOV     AL,ES:[SI]              ; GET THE BYTE
 0452 22 C4                             AND     AL,AH                   ; MASK OFF THE OTHER BITS IN THE BYTE
 0454 D2 E0                             SHL     AL,CL                   ; LEFT JUSTIFY THE VALUE
 0456 8A CE                             MOV     CL,DH                   ; GET NUMBER OF BITS IN RESULT
 0458 D2 C0                             ROL     AL,CL                   ; RIGHT JUSTIFY THE RESULT
 045A E9 0145 R                         JMP     VIDEO_RETURN            ; RETURN FROM VIDEO I/O
 045D                           READ_DOT        ENDP

 045D                           WRITE_DOT       PROC    NEAR
 045D 50                                PUSH    AX                      ; SAVE DOT VALUE
 045E 50                                PUSH    AX                      ;  TWICE
 045F E8 0480 R                         CALL    R3                      ; DETERMINE BYTE POSITION OF THE DOT
 0462 D2 E8                             SHR     AL,CL                   ; SHIFT TO SET UP THE BITS FOR OUTPUT
 0464 22 C4                             AND     AL,AH                   ; STRIP OFF THE OTHER BITS
 0466 26: 8A 0C                         MOV     CL,ES:[SI]              ; GET THE CURRENT BYTE
 0469 5B                                POP     BX                      ; RECOVER XOR FLAG
 046A F6 C3 80                          TEST    BL,80H                  ; IS IT ON
 046D 75 0D                             JNZ     R2                      ; YES, XOR THE DOT
 046F F6 D4                             NOT     AH                      ; SET MASK TO REMOVE THE INDICATED BITS
 0471 22 CC                             AND     CL,AH
 0473 0A C1                             OR      AL,CL                   ; OR IN THE NEW VALUE OF THOSE BITS
 0475                           R1:                                     ; FINISH_DOT
 0475 26: 88 04                         MOV     ES:[SI],AL              ; RESTORE THE BYTE IN MEMORY
 0478 58                                POP     AX
 0479 E9 0145 R                         JMP     VIDEO_RETURN            ; RETURN FROM VIDEO I/O
 047C                           R2:                                     ; XOR_DOT
 047C 32 C1                             XOR     AL,CL                   ; EXCLUSIVE OR THE DOTS
 047E EB F5                             JMP     R1                      ; FINISH UP THE WRITING
 0480                           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
                                ;-----------------------------------------------
 0480                           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 )

 0480 93                                XCHG    AX,BX                   ; WILL SAVE AL AND AH DURING OPERATION
 0481 B0 28                             MOV     AL,40
 0483 F6 E2                             MUL     DL                      ; AX= ADDRESS OF START OF INDICATED ROW
 0485 A8 08                             TEST    AL,008H                 ; TEST FOR EVEN/ODD ROW CALCULATED
 0487 74 03                             JZ      R4                      ; JUMP IF EVEN ROW
 0489 05 1FD8                           ADD     AX,2000H-40             ; OFFSET TO LOCATION OF ODD ROWS ADJUST
 048C                           R4:                                     ; EVEN_ROW
 048C 96                                XCHG    SI,AX                   ; MOVE POINTER TO SI
 048D 93                                XCHG    AX,BX                   ; RECOVER AL AND AH VALUES
 048E 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 )

 0490 BB 02C0                           MOV     BX,2C0H
 0493 B9 0302                           MOV     CX,302H                 ; SET PARMS FOR MED RES
 0496 80 3E 0049 R 06                   CMP     @CRT_MODE,6
 049B 72 06                             JC      R5                      ; HANDLE IF MED RES
 049D BB 0180                           MOV     BX,180H
 04A0 B9 0703                           MOV     CX,703H                 ; SET PARMS FOR HIGH RES

                                ;-----  DETERMINE BIT OFFSET IN BYTE FROM COLUMN MASK
 04A3                           R5:
 04A3 22 EA                             AND     CH,DL                   ; ADDRESS OF PEL WITHIN BYTE TO CH

                                ;-----  DETERMINE BYTE OFFSET FOR THIS LOCATION IN COLUMN

 04A5 D3 EA                             SHR     DX,CL                   ; SHIFT BY CORRECT AMOUNT
 04A7 03 F2                             ADD     SI,DX                   ; INCREMENT THE POINTER
 04A9 8A F7                             MOV     DH,BH                   ; GET THE # OF BITS IN RESULT TO DH

                                ;-----  MULTIPLY BH (VALID BITS IN BYTE) BY CH (BIT OFFSET)

 04AB 2A C9                             SUB     CL,CL                   ; ZERO INTO STORAGE LOCATION
 04AD                           R6:
 04AD D0 C8                             ROR     AL,1                    ; LEFT JUSTIFY VALUE IN AL (FOR WRITE)
 04AF 02 CD                             ADD     CL,CH                   ; ADD IN THE BIT OFFSET VALUE
 04B1 FE CF                             DEC     BH                      ; LOOP CONTROL
 04B3 75 F8                             JNZ     R6                      ; ON EXIT, CL HAS COUNT TO RESTORE BITS
 04B5 8A E3                             MOV     AH,BL                   ;  GET MASK TO AH
 04B7 D2 EC                             SHR     AH,CL                   ;  MOVE THE MASK TO CORRECT LOCATION
 04B9 C3                                RET                             ;  RETURN WITH EVERYTHING SET UP
```

```
04BA                          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
                              ;--------------------------------------------------------
04BA                          GRAPHICS_UP     PROC    NEAR
04BA 8A D8                            MOV     BL,AL                   ; SAVE LINE COUNT IN BL
04BC 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

04BE E8 06F8 R                        CALL    GRAPH_POSN
04C1 8B F8                            MOV     DI,AX                   ; SAVE RESULT AS DESTINATION ADDRESS

                              ;-----  DETERMINE SIZE OF WINDOW

04C3 2B D1                            SUB     DX,CX
04C5 81 C2 0101                       ADD     DX,101H                 ; ADJUST VALUES
04C9 C0 E6 02                         SAL     DH,2                    ; MULTIPLY ROWS BY 4 AT 8 VERT DOTS/CHAR
                                                                      ;  AND EVEN/ODD ROWS
                              ;-----  DETERMINE CRT MODE

04CC 80 3E 0049 R 06                  CMP     @CRT_MODE,6             ; TEST FOR MEDIUM RES
04D1 73 04                            JNC     R7                      ; FIND_SOURCE

                              ;-----  MEDIUM RES UP
04D3 D0 E2                            SAL     DL,1                    ; # COLUMNS * 2, SINCE 2 BYTES/CHAR
04D5 D1 E7                            SAL     DI,1                    ; OFFSET *2 SINCE 2 BYTES/CHAR

                              ;-----  DETERMINE THE SOURCE ADDRESS IN THE BUFFER
04D7                          R7:                                     ; FIND_SOURCE
04D7 06                               PUSH    ES                      ; GET SEGMENTS BOTH POINTING TO REGEN
04D8 1F                               POP     DS
04D9 2A ED                            SUB     CH,CH                   ; ZERO TO HIGH OF COUNT REGISTER
04DB C0 E3 02                         SAL     BL,2                    ; MULTIPLY NUMBER OF LINES BY 4
04DE 74 2D                            JZ      R11                     ; IF ZERO, THEN BLANK ENTIRE FIELD
04E0 8A C3                            MOV     AL,BL                   ; GET NUMBER OF LINES IN AL
04E2 B4 50                            MOV     AH,80                   ; 80 BYTES/ROW
04E4 F6 E4                            MUL     AH                      ; DETERMINE OFFSET TO SOURCE
04E6 8B F7                            MOV     SI,DI                   ; SET UP SOURCE
04E8 03 F0                            ADD     SI,AX                   ;  ADD IN OFFSET TO IT
04EA 8A E6                            MOV     AH,DH                   ; NUMBER OF ROWS IN FIELD
04EC 2A E3                            SUB     AH,BL                   ; DETERMINE NUMBER TO MOVE

                              ;-----  LOOP THROUGH, MOVING ONE ROW AT A TIME, BOTH EVEN AND ODD FIELDS
04EE                          R8:                                     ; ROW_LOOP
04EE E8 056F R                        CALL    R17                     ; MOVE ONE ROW
04F1 81 EE 1FB0                       SUB     SI,2000H-80             ; MOVE TO NEXT ROW
04F5 81 EF 1FB0                       SUB     DI,2000H-80
04F9 FE CC                            DEC     AH                      ; NUMBER OF ROWS TO MOVE
04FB 75 F1                            JNZ     R8                      ; CONTINUE TILL ALL MOVED

                              ;-----  FILL IN THE VACATED LINE(S)
04FD                          R9:                                     ; CLEAR_ENTRY
04FD 8A C7                            MOV     AL,BH                   ; ATTRIBUTE TO FILL WITH
04FF                          R10:
04FF E8 0588 R                        CALL    R18                     ; CLEAR THAT ROW
0502 81 EF 1FB0                       SUB     DI,2000H-80             ; POINT TO NEXT LINE
0506 FE CB                            DEC     BL                      ; NUMBER OF LINES TO FILL
0508 75 F5                            JNZ^    R10                     ; CLEAR_LOOP
050A E9 0145 R                        JMP     VIDEO_RETURN            ; EVERYTHING DONE

050D                          R11:                                    ; BLANK_FIELD
050D 8A DE                            MOV     BL,DH                   ; SET BLANK COUNT TO EVERYTHING IN FIELD
050F EB EC                            JMP     R9                      ; CLEAR THE FIELD
0511                          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
                              ;-----------------------------------------------------

0511                          GRAPHICS_DOWN   PROC    NEAR
0511 FD                               STD                             ; SET DIRECTION
0512 8A D8                            MOV     BL,AL                   ; SAVE LINE COUNT IN BL
0514 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

0516 E8 06F8 R                        CALL    GRAPH_POSN
0519 8B F8                            MOV     DI,AX                   ; SAVE RESULT AS DESTINATION ADDRESS

                              ;-----  DETERMINE SIZE OF WINDOW

051B 2B D1                            SUB     DX,CX
051D 81 C2 0101                       ADD     DX,101H                 ; ADJUST VALUES
0521 C0 E6 02                         SAL     DH,2                    ; MULTIPLY ROWS BY 4 AT 8 VERT DOTS/CHAR
                                                                      ;  AND EVEN/ODD ROWS
                              ;-----  DETERMINE CRT MODE

0524 80 3E 0049 R 06                  CMP     @CRT_MODE,6             ; TEST FOR MEDIUM RES
0529 73 05                            JNC     R12                     ; FIND_SOURCE_DOWN
```

```
                               ;----- MEDIUM RES DOWN
052B D0 E2                            SAL     DL,1                  ; # COLUMNS * 2, SINCE 2 BYTES/CHAR
052D D1 E7                            SAL     DI,1                  ; OFFSET *2 SINCE 2 BYTES/CHAR
052F 47                               INC     DI                    ; POINT TO LAST BYTE

                               ;----- DETERMINE THE SOURCE ADDRESS IN THE BUFFER
0530                           R12:                                 ; FIND_SOURCE_DOWN
0530 06                               PUSH    ES                    ; BOTH_SEGMENTS TO REGEN
0531 1F                               POP     DS
0532 2A ED                            SUB     CH,CH                 ; ZERO TO HIGH OF COUNT REGISTER
0534 81 C7 00F0                       ADD     DI,240                ; POINT TO LAST ROW OF PIXELS
0538 C0 E3 02                         SAL     BL,2                  ; MULTIPLY NUMBER OF LINES BY 4
053B 74 2E                            JZ      R16                   ; IF ZERO, THEN BLANK ENTIRE FIELD
053D 8A C3                            MOV     AL,BL                 ; GET NUMBER OF LINES IN AL
053F B4 50                            MOV     AH,80                 ; 80 BYTES/ROW
0541 F6 E4                            MUL     AH                    ; DETERMINE OFFSET TO SOURCE
0543 8B F7                            MOV     SI,DI                 ; SET UP SOURCE
0545 2B F0                            SUB     SI,AX                 ;  SUBTRACT THE OFFSET
0547 8A E6                            MOV     AH,DH                 ; NUMBER OF ROWS IN FIELD
0549 2A E3                            SUB     AH,BL                 ; DETERMINE NUMBER TO MOVE

                               ;----- LOOP THROUGH, MOVING ONE ROW AT A TIME, BOTH EVEN AND ODD FIELDS
054B                           R13:                                 ; ROW_LOOP_DOWN
054B E8 056F R                        CALL    R17                   ; MOVE ONE ROW
054E 81 EE 2050                       SUB     SI,2000H+80           ; MOVE TO NEXT ROW
0552 81 EF 2050                       SUB     DI,2000H+80
0556 FE CC                            DEC     AH                    ; NUMBER OF ROWS TO MOVE
0558 75 F1                            JNZ     R13                   ; CONTINUE TILL ALL MOVED

                               ;----- FILL IN THE VACATED LINE(S)
055A                           R14:                                 ; CLEAR_ENTRY_DOWN
055A 8A C7                            MOV     AL,BH                 ; ATTRIBUTE TO FILL WITH
055C                           R15:                                 ; CLEAR_LOOP_DOWN
055C E8 0588 R                        CALL    R18                   ; CLEAR A ROW
055F 81 EF 2050                       SUB     DI,2000H+80           ; POINT TO NEXT LINE
0563 FE CB                            DEC     BL                    ; NUMBER OF LINES TO FILL
0565 75 F5                            JNZ     R15                   ; CLEAR_LOOP_DOWN
0567 FC                               CLD                           ; RESET THE DIRECTION FLAG
0568 E9 0145 R                        JMP     VIDEO_RETURN          ; EVERYTHING DONE

056B                           R16:                                 ; BLANK_FIELD_DOWN
056B 8A DE                            MOV     BL,DH                 ; SET BLANK COUNT TO EVERYTHING IN FIELD
056D EB EB                            JMP     R14                   ; CLEAR THE FIELD
056F                           GRAPHICS_DOWN  ENDP

                               ;----- ROUTINE TO MOVE ONE ROW OF INFORMATION

056F                           R17    PROC    NEAR
056F 8A CA                            MOV     CL,DL                 ; NUMBER OF BYTES IN THE ROW
0571 56                               PUSH    SI
0572 57                               PUSH    DI                    ; SAVE POINTERS
0573 F3/ A4                           REP     MOVSB                 ; MOVE THE EVEN FIELD
0575 5F                               POP     DI
0576 5E                               POP     SI
0577 81 C6 2000                       ADD     SI,2000H
057B 81 C7 2000                       ADD     DI,2000H              ; POINT TO THE ODD FIELD
057F 56                               PUSH    SI
0580 57                               PUSH    DI                    ; SAVE THE POINTERS
0581 8A CA                            MOV     CL,DL                 ; COUNT BACK
0583 F3/ A4                           REP     MOVSB                 ; MOVE THE ODD FIELD
0585 5F                               POP     DI
0586 5E                               POP     SI                    ; POINTERS BACK
0587 C3                               RET                           ; RETURN TO CALLER
0588                           R17    ENDP

                               ;----- CLEAR A SINGLE ROW

0588                           R18    PROC    NEAR
0588 8A CA                            MOV     CL,DL                 ; NUMBER OF BYTES IN FIELD
058A 57                               PUSH    DI                    ; SAVE POINTER
058B F3/ AA                           REP     STOSB                 ; STORE THE NEW VALUE
058D 5F                               POP     DI                    ; POINTER BACK
058E 81 C7 2000                       ADD     DI,2000H              ; POINT TO ODD FIELD
0592 57                               PUSH    DI
0593 8A CA                            MOV     CL,DL
0595 F3/ AA                           REP     STOSB                 ; FILL THE ODD FIELD
0597 5F                               POP     DI
0598 C3                               RET                           ; RETURN TO CALLER
0599                           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
0599                           GRAPHICS_WRITE PROC    NEAR
0599 B4 00                            MOV     AH,0                  ; ZERO TO HIGH OF CODE POINT
059B 50                               PUSH    AX                    ; SAVE CODE POINT VALUE

```

SECTION 5

```
                                ;-----  DETERMINE POSITION IN REGEN BUFFER TO PUT CODE POINTS

059C E8 06F5 R                          CALL    S26                     ; FIND LOCATION IN REGEN BUFFER
059F 8B F8                              MOV     DI,AX                   ; REGEN POINTER IN DI

                                ;-----  DETERMINE REGION TO GET CODE POINTS FROM

05A1 58                                 POP     AX                      ; RECOVER CODE POINT
05A2 3C 80                              CMP     AL,80H                  ; IS IT IN SECOND HALF
05A4 73 06                              JAE     S1                      ; YES

                                ;-----  IMAGE IS IN FIRST HALF, CONTAINED IN ROM

05A6 BE 0000 E                          MOV     SI,OFFSET CRT_CHAR_GEN  ; OFFSET OF IMAGES
05A9 0E                                 PUSH    CS                      ; SAVE SEGMENT ON STACK
05AA EB 18                              JMP     SHORT S2                ; DETERMINE_MODE

                                ;-----  IMAGE IS IN SECOND HALF, IN USER MEMORY

05AC                            S1:                                     ; EXTEND_CHAR
05AC 2C 80                              SUB     AL,80H                  ; ZERO ORIGIN FOR SECOND HALF
05AE 1E                                 PUSH    DS                      ; SAVE DATA POINTER
05AF 2B F6                              SUB     SI,SI
05B1 8E DE                              MOV     DS,SI                   ; ESTABLISH VECTOR ADDRESSING
                                        ASSUME  DS:ABS0
05B3 C5 36 007C R                       LDS     SI,@EXT_PTR             ; GET THE OFFSET OF THE TABLE
05B7 8C DA                              MOV     DX,DS                   ; GET THE SEGMENT OF THE TABLE
                                        ASSUME  DS:DATA
05B9 1F                                 POP     DS                      ; RECOVER DATA SEGMENT
05BA 52                                 PUSH    DX                      ; SAVE TABLE SEGMENT ON STACK
05BB 0B D6                              OR      DX,SI                   ; CHECK FOR VALID TABLE DEFINED
05BD 75 05                              JNZ     S2                      ; CONTINUE IF DS:SI NOT 0000:0000

05BF 58                                 POP     AX                      ; ELSE SET (AX)= 0000 FOR "NULL"
05C0 BE 0000 E                          MOV     SI,OFFSET CRT_CHAR_GEN  ; POINT TO DEFAULT TABLE OFFSET
05C3 0E                                 PUSH    CS                      ;  IN THE CODE SEGMENT

                                ;-----  DETERMINE GRAPHICS MODE IN OPERATION

05C4                            S2:                                     ;       DETERMINE_MODE
05C4 C1 E0 03                           SAL     AX,3                    ; MULTIPLY CODE POINT VALUE BY 8
05C7 03 F0                              ADD     SI,AX                   ; SI HAS OFFSET OF DESIRED CODES
05C9 80 3E 0049 R 06                    CMP     @CRT_MODE,6
05CE 1F                                 POP     DS                      ; RECOVER TABLE POINTER SEGMENT
05CF 72 2C                              JC      S7                      ; TEST FOR MEDIUM RESOLUTION MODE

                                ;-----  HIGH RESOLUTION MODE
05D1                            S3:                                     ; HIGH_CHAR
05D1 57                                 PUSH    DI                      ; SAVE REGEN POINTER
05D2 56                                 PUSH    SI                      ; SAVE CODE POINTER
05D3 B6 04                              MOV     DH,4                    ; NUMBER OF TIMES THROUGH LOOP
05D5                            S4:
05D5 AC                                 LODSB                           ; GET BYTE FROM CODE POINTS
05D6 F6 C3 80                           TEST    BL,80H                  ; SHOULD WE USE THE FUNCTION
05D9 75 16                              JNZ     S6                      ;  TO PUT CHAR IN
05DB AA                                 STOSB                           ; STORE IN REGEN BUFFER
05DC AC                                 LODSB
05DD                            S5:
05DD 26: 88 85 1FFF                     MOV     ES:[DI+2000H-1],AL      ; STORE IN SECOND HALF
05E2 83 C7 4F                           ADD     DI,79                   ; MOVE TO NEXT ROW IN REGEN
05E5 FE CE                              DEC     DH                      ; DONE WITH LOOP
05E7 75 EC                              JNZ     S4
05E9 5E                                 POP     SI
05EA 5F                                 POP     DI                      ; RECOVER REGEN POINTER
05EB 47                                 INC     DI                      ; POINT TO NEXT CHAR POSITION
05EC E2 E3                              LOOP    S3                      ; MORE CHARS TO WRITE
05EE E9 0145 R                          JMP     VIDEO_RETURN

05F1                            S6:
05F1 26: 32 05                          XOR     AL,ES:[DI]              ; EXCLUSIVE OR WITH CURRENT
05F4 AA                                 STOSB                           ; STORE THE CODE POINT
05F5 AC                                 LODSB                           ; AGAIN FOR ODD FIELD
05F6 26: 32 85 1FFF                     XOR     AL,ES:[DI+2000H-1]
05FB EB E0                              JMP     S5                      ; BACK TO MAINSTREAM

                                ;-----  MEDIUM RESOLUTION WRITE
05FD                            S7:                                     ;  MED_RES_WRITE
05FD 8A D3                              MOV     DL,BL                   ; SAVE HIGH COLOR BIT
05FF D1 E7                              SAL     DI,1                    ; OFFSET*2 SINCE 2 BYTES/CHAR
                                                                        ; EXPAND BL TO FULL WORD OF COLOR
0601 80 E3 03                           AND     BL,3                    ; ISOLATE THE COLOR BITS ( LOW 2 BITS )
0604 B0 55                              MOV     AL,055H                 ; GET BIT CONVERSION MULTIPLIER
0606 F6 E3                              MUL     BL                      ; EXPAND 2 COLOR BITS TO 4 REPLICATIONS
0608 8A D8                              MOV     BL,AL                   ; PLACE BACK IN WORK REGISTER
060A 8A F8                              MOV     BH,AL                   ; EXPAND TO 8 REPLICATIONS OF COLOR BITS
060C                            S8:                                     ;  MED_CHAR
060C 57                                 PUSH    DI                      ; SAVE REGEN POINTER
060D 56                                 PUSH    SI                      ; SAVE THE CODE POINTER
060E B6 04                              MOV     DH,4                    ; NUMBER OF LOOPS
0610                            S9:
0610 AC                                 LODSB                           ; GET CODE POINT
0611 E8 06CD R                          CALL    S21                     ; DOUBLE UP ALL THE BITS
0614 23 C3                              AND     AX,BX                   ; CONVERT TO FOREGROUND COLOR ( 0 BACK )
0616 86 E0                              XCHG    AH,AL                   ; SWAP HIGH/LOW BYTES FOR WORD MOVE
0618 F6 C2 80                           TEST    DL,80H                  ; IS THIS XOR FUNCTION
061B 74 03                              JZ      S10                     ; NO, STORE IT IN AS IT IS
061D 26: 33 05                          XOR     AX,ES:[DI]              ; DO FUNCTION WITH LOW/HIGH
0620                            S10:                                    ;
0620 26: 89 05                          MOV     ES:[DI],AX              ; STORE FIRST BYTE HIGH, SECOND LOW
0623 AC                                 LODSB                           ; GET CODE POINT
0624 E8 06CD R                          CALL    S21
0627 23 C3                              AND     AX,BX                   ; CONVERT TO COLOR
0629 86 E0                              XCHG    AH,AL                   ; SWAP HIGH/LOW BYTES FOR WORD MOVE
062B F6 C2 80                           TEST    DL,80H                  ; AGAIN, IS THIS XOR FUNCTION
062E 74 05                              JZ      S11                     ; NO, JUST STORE THE VALUES
0630 26: 33 85 2000                     XOR     AX,ES:[DI+2000H]        ; FUNCTION WITH FIRST HALF LOW
0635                            S11:
0635 26: 89 85 2000                     MOV     ES:[DI+2000H],AX        ; STORE SECOND PORTION HIGH
063A 83 C7 50                           ADD     DI,80                   ; POINT TO NEXT LOCATION
063D FE CE                              DEC     DH
063F 75 CF                              JNZ     S9                      ; KEEP GOING
0641 5E                                 POP     SI                      ; RECOVER CODE POINTER
0642 5F                                 POP     DI                      ; RECOVER REGEN POINTER
0643 47                                 INC     DI                      ; POINT TO NEXT CHAR POSITION
```

```
0644 47                         INC     DI
0645 E2 C5                      LOOP    S8                      ; MORE TO WRITE
0647 E9 0145 R                  JMP     VIDEO_RETURN
064A                    GRAPHICS_WRITE  ENDP
                        ;-----------------------------------------
                        ; GRAPHICS READ
                        ;-----------------------------------------
064A                    GRAPHICS_READ   PROC    NEAR
064A E8 06F5 R                  CALL    S26                     ; CONVERTED TO OFFSET IN REGEN
064D 8B F0                      MOV     SI,AX                   ; SAVE IN SI
064F 83 EC 08                   SUB     SP,8                    ; ALLOCATE SPACE FOR THE READ CODE POINT
0652 8B EC                      MOV     BP,SP                   ; POINTER TO SAVE AREA

                        ;-----  DETERMINE GRAPHICS MODES

0654 80 3E 0049 R 06            CMP     @CRT_MODE,6
0659 06                         PUSH    ES
065A 1F                         POP     DS                      ; POINT TO REGEN SEGMENT
065B 72 19                      JC      S13                     ; MEDIUM RESOLUTION

                        ;-----  HIGH RESOLUTION READ

                        ;-----  GET VALUES FROM REGEN BUFFER AND CONVERT TO CODE POINT
065D B6 04                      MOV     DH,4                    ; NUMBER OF PASSES
065F                    S12:
065F 8A 04                      MOV     AL,[SI]                 ; GET FIRST BYTE
0661 88 46 00                   MOV     [BP],AL                 ; SAVE IN STORAGE AREA
0664 45                         INC     BP                      ; NEXT LOCATION
0665 8A 84 2000                 MOV     AL,[SI+2000H]           ; GET LOWER REGION BYTE
0669 88 46 00                   MOV     [BP],AL                 ; ADJUST AND STORE
066C 45                         INC     BP
066D 83 C6 50                   ADD     SI,80                   ; POINTER INTO REGEN
0670 FE CE                      DEC     DH                      ; LOOP CONTROL
0672 75 EB                      JNZ     S12                     ; DO IT SOME MORE
0674 EB 16                      JMP     SHORT S15               ; GO MATCH THE SAVED CODE POINTS

                        ;-----  MEDIUM RESOLUTION READ
0676                    S13:                                    ; MED_RES_READ
0676 D1 E6                      SAL     SI,1                    ; OFFSET*2 SINCE 2 BYTES/CHAR
0678 B6 04                      MOV     DH,4                    ; NUMBER OF PASSES
067A                    S14:
067A E8 06DC R                  CALL    S23                     ; GET BYTES FROM REGEN INTO SINGLE SAVE
067D 81 C6 1FFE                 ADD     SI,2000H-2              ; GO TO LOWER REGION
0681 E8 06DC R                  CALL    S23                     ; GET THIS PAIR INTO SAVE
0684 81 EE 1FB2                 SUB     SI,2000H-80+2           ; ADJUST POINTER BACK INTO UPPER
0688 FE CE                      DEC     DH
068A 75 EE                      JNZ     S14                     ; KEEP GOING UNTIL ALL 8 DONE

                        ;-----  SAVE AREA HAS CHARACTER IN IT, MATCH IT
068C                    S15:                                    ; FIND_CHAR
068C BF 0000 E                  MOV     DI,OFFSET CRT_CHAR_GEN  ; ESTABLISH ADDRESSING
068F 0E                         PUSH    CS
0690 07                         POP     ES                      ; CODE POINTS IN CS
0691 83 ED 08                   SUB     BP,8                    ; ADJUST POINTER TO START OF SAVE AREA
0694 8B F5                      MOV     SI,BP
0696 FC                         CLD                             ; ENSURE DIRECTION
0697 B0 00                      MOV     AL,0                    ; CURRENT CODE POINT BEING MATCHED
0699                    S16:
0699 16                         PUSH    SS                      ; ESTABLISH ADDRESSING TO STACK
069A 1F                         POP     DS                      ; FOR THE STRING COMPARE
069B BA 0080                    MOV     DX,128                  ; NUMBER TO TEST AGAINST
069E                    S17:
069E 56                         PUSH    SI                      ; SAVE SAVE AREA POINTER
069F 57                         PUSH    DI                      ; SAVE CODE POINTER
06A0 B9 0004                    MOV     CX,4                    ; NUMBER OF WORDS TO MATCH
06A3 F3/ A7                     REPE    CMPSW                   ; COMPARE THE 8 BYTES AS WORDS
06A5 5F                         POP     DI                      ; RECOVER THE POINTERS
06A6 5E                         POP     SI
06A7 74 1E                      JZ      S18                     ; IF ZERO FLAG SET, THEN MATCH OCCURRED
06A9 FE C0                      INC     AL                      ; NO MATCH, MOVE ON TO NEXT
06AB 83 C7 08                   ADD     DI,8                    ; NEXT CODE POINT
06AE 4A                         DEC     DX                      ; LOOP CONTROL
06AF 75 ED                      JNZ     S17                     ; DO ALL OF THEM

                        ;-----  CHAR NOT MATCHED, MIGHT BE IN USER SUPPLIED SECOND HALF

06B1 3C 00                      CMP     AL,0                    ;  AL<> 0 IF ONLY 1ST HALF SCANNED
06B3 74 12                      JE      S18                     ;  IF = 0, THEN ALL HAS BEEN SCANNED
06B5 2B C0                      SUB     AX,AX
06B7 8E D8                      MOV     DS,AX                   ;  ESTABLISH ADDRESSING TO VECTOR
                                ASSUME  DS:ABS0
06B9 C4 3E 007C R               LES     DI,@EXT_PTR             ;  GET POINTER
06BD 8C C0                      MOV     AX,ES                   ;  SEE IF THE POINTER REALLY EXISTS
06BF 0B C7                      OR      AX,DI                   ; IF ALL 0, THEN DOESN'T EXIST
06C1 74 04                      JZ      S18                     ; NO SENSE LOOKING
06C3 B0 80                      MOV     AL,128                  ; ORIGIN FOR SECOND HALF
06C5 EB D2                      JMP     S16                     ; GO BACK AND TRY FOR IT
                                ASSUME  DS:DATA

                        ;-----  CHARACTER IS FOUND ( AL=0 IF NOT FOUND )
06C7                    S18:
06C7 83 C4 08                   ADD     SP,8                    ; READJUST THE STACK, THROW AWAY SAVE
06CA E9 0145 R                  JMP     VIDEO_RETURN            ; ALL DONE
06CD                    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
                        ;-----------------------------------------
06CD                    S21     PROC    NEAR
06CD 51                         PUSH    CX                      ; SAVE REGISTER
06CE B9 0008                    MOV     CX,8                    ; SHIFT COUNT REGISTER FOR ONE BYTE
06D1                    S22:
06D1 D0 C8                      ROR     AL,1                    ; SHIFT BITS, LOW BIT INTO CARRY FLAG
06D3 D1 DD                      RCR     BP,1                    ; MOVE CARRY FLAG (LOW BIT) INTO RESULTS
06D5 D1 FD                      SAR     BP,1                    ; SIGN EXTEND HIGH BIT (DOUBLE IT)
06D7 E2 F8                      LOOP    S22                     ; REPEAT FOR ALL 8 BITS

06D9 95                         XCHG    AX,BP                   ; MOVE RESULTS TO PARAMETER REGISTER
06DA 59                         POP     CX                      ; RECOVER REGISTER
06DB C3                         RET                             ; ALL DONE
06DC                    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
                                ;----------------------------------------------------
06DC                            S23     PROC    NEAR
06DC AD                                 LODSW                           ; GET FIRST BYTE AND SECOND BYTES
06DD 86 C4                              XCHG    AL,AH                   ; SWAP FOR COMPARE
06DF B9 C000                            MOV     CX,0C000H               ; 2 BIT MASK TO TEST THE ENTRIES
06E2 B2 00                              MOV     DL,0                    ; RESULT REGISTER
06E4                            S24:
06E4 85 C1                              TEST    AX,CX                   ; IS THIS SECTION BACKGROUND?
06E6 74 01                              JZ      S25                     ; IF ZERO, IT IS BACKGROUND (CARRY=0)
06E8 F9                                 STC                             ; WASN'T, SO SET CARRY
06E9                            S25:
06E9 D0 D2                              RCL     DL,1                    ; MOVE THAT BIT INTO THE RESULT
06EB C1 E9 02                           SHR     CX,2                    ; MOVE THE MASK TO THE RIGHT BY 2 BITS
06EE 73 F4                              JNC     S24                     ; DO IT AGAIN IF MASK DIDN'T FALL OUT
06F0 88 56 00                           MOV     [BP],DL                 ; STORE RESULT IN SAVE AREA
06F3 45                                 INC     BP                      ; ADJUST POINTER
06F4 C3                                 RET                             ; ALL DONE
06F5                            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
                                ;----------------------------------------
06F5                            S26     PROC    NEAR
06F5 A1 0050 R                          MOV     AX,@CURSOR_POSN         ; GET CURRENT CURSOR
06F8                            GRAPH_POSN      LABEL   NEAR
06F8 53                                 PUSH    BX                      ; SAVE REGISTER
06F9 8B D8                              MOV     BX,AX                   ; SAVE A COPY OF CURRENT CURSOR
06FB 8A C4                              MOV     AL,AH                   ; GET ROWS TO AL
06FD F6 26 004A R                       MUL     BYTE PTR @CRT_COLS      ; MULTIPLY BY BYTES/COLUMN
0701 C1 E0 02                           SHL     AX,2                    ; MULTIPLY * 4 SINCE 4 ROWS/BYTE
0704 2A FF                              SUB     BH,BH                   ; ISOLATE COLUMN VALUE
0706 03 C3                              ADD     AX,BX                   ; DETERMINE OFFSET
0708 5B                                 POP     BX                      ; RECOVER POINTER
0709 C3                                 RET                             ; ALL DONE
070A                            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                                                :
                                ;-------------------------------------------------------------------------
                                        ASSUME  DS:DATA
070A                            WRITE_TTY       PROC    NEAR
070A 50                                 PUSH    AX                      ; SAVE REGISTERS
070B 50                                 PUSH    AX                      ; SAVE CHARACTER TO WRITE
070C B4 03                              MOV     AH,03H
070E 8A 3E 0062 R                       MOV     BH,@ACTIVE_PAGE         ; GET CURRENT PAGE SETTING
0712 CD 10                              INT     10H                     ; READ THE CURRENT CURSOR POSITION
0714 58                                 POP     AX                      ; RECOVER CHARACTER

                                ;-----  DX NOW HAS THE CURRENT CURSOR POSITION

0715 3C 0D                              CMP     AL,CR                   ; IS IT CARRIAGE RETURN OR CONTROL
0717 76 46                              JBE     U8                      ; GO TO CONTROL CHECKS IF IT IS

                                ;-----  WRITE THE CHAR TO THE SCREEN
0719                            U0:
0719 B4 0A                              MOV     AH,0AH                  ; WRITE CHARACTER ONLY COMMAND
071B B9 0001                            MOV     CX,1                    ; ONLY ONE CHARACTER
071E CD 10                              INT     10H                     ; WRITE THE CHARACTER

                                ;-----  POSITION THE CURSOR FOR NEXT CHAR

0720 FE C2                              INC     DL
0722 3A 16 004A R                       CMP     DL,BYTE PTR @CRT_COLS   ; TEST FOR COLUMN OVERFLOW
0726 75 33                              JNZ     U7                      ;  SET_CURSOR
0728 B2 00                              MOV     DL,0                    ; COLUMN FOR CURSOR
072A 80 FE 18                           CMP     DH,25-1                 ; CHECK FOR LAST ROW
072D 75 2A                              JNZ     U6                      ;  SET_CURSOR_INC

                                ;-----  SCROLL REQUIRED
072F                            U1:
072F B4 02                              MOV     AH,02H
0731 CD 10                              INT     10H                     ; SET THE CURSOR

                                ;-----  DETERMINE VALUE TO FILL WITH DURING SCROLL

0733 A0 0049 R                          MOV     AL,@CRT_MODE            ; GET THE CURRENT MODE
0736 3C 04                              CMP     AL,4
0738 72 06                              JC      U2                      ; READ-CURSOR
```

```
073A 3C 07                    CMP     AL,7
073C B7 00                    MOV     BH,0                    ; FILL WITH BACKGROUND
073E 75 06                    JNE     U3                      ; SCROLL-UP
0740                  U2:                                     ; READ-CURSOR
0740 B4 08                    MOV     AH,08H                  ; GET READ CURSOR COMMAND
0742 CD 10                    INT     10H                     ; READ CHAR/ATTR AT CURRENT CURSOR
0744 8A FC                    MOV     BH,AH                   ; STORE IN BH
0746                  U3:                                     ; SCROLL-UP
0746 B8 0601                  MOV     AX,0601H                ; SCROLL ONE LINE
0749 2B C9                    SUB     CX,CX                   ; UPPER LEFT CORNER
074B B6 18                    MOV     DH,25-1                 ; LOWER RIGHT ROW
074D 8A 16 004A R             MOV     DL,BYTE PTR @CRT_COLS   ; LOWER RIGHT COLUMN
0751 FE CA                    DEC     DL
0753                  U4:                                     ; VIDEO-CALL-RETURN
0753 CD 10                    INT     10H                     ; SCROLL UP THE SCREEN
0755                  U5:                                     ; TTY-RETURN
0755 58                       POP     AX                      ; RESTORE THE CHARACTER
0756 E9 0145 R                JMP     VIDEO_RETURN            ; RETURN TO CALLER

0759                  U6:                                     ; SET-CURSOR-INC
0759 FE C6                    INC     DH                      ; NEXT ROW
075B                  U7:                                     ; SET-CURSOR
075B B4 02                    MOV     AH,02H
075D EB F4                    JMP     U4                      ; ESTABLISH THE NEW CURSOR

                      ;-----  CHECK FOR CONTROL CHARACTERS
075F                  U8:
075F 74 13                    JE      U9                      ; WAS IT A CARRIAGE RETURN
0761 3C 0A                    CMP     AL,LF                   ; IS IT A LINE FEED
0763 74 13                    JE      U10                     ; GO TO LINE FEED
0765 3C 07                    CMP     AL,07H                  ; IS IT A BELL
0767 74 16                    JE      U11                     ; GO TO BELL
0769 3C 08                    CMP     AL,08H                  ; IS IT A BACKSPACE
076B 75 AC                    JNE     U0                      ; IF NOT A CONTROL, DISPLAY IT

                      ;-----  BACK SPACE FOUND

076D 0A D2                    OR      DL,DL                   ; IS IT ALREADY AT START OF LINE
076F 74 EA                    JE      U7                      ; SET_CURSOR
0771 4A                       DEC     DX                      ; NO -- JUST MOVE IT BACK
0772 EB E7                    JMP     U7                      ; SET_CURSOR

                      ;-----  CARRIAGE RETURN FOUND

0774                  U9:
0774 B2 00                    MOV     DL,0                    ; MOVE TO FIRST COLUMN
0776 EB E3                    JMP     U7                      ; SET_CURSOR

                      ;-----  LINE FEED FOUND

0778                  U10:
0778 80 FE 18                 CMP     DH,25-1                 ; BOTTOM OF SCREEN
077B 75 DC                    JNE     U6                      ; YES, SCROLL THE SCREEN
077D EB B0                    JMP     U1                      ; NO, JUST SET THE CURSOR

                      ;-----  BELL FOUND

077F                  U11:
077F B9 0533                  MOV     CX,1331                 ; DIVISOR FOR 896 HZ TONE
0782 B3 1F                    MOV     BL,31                   ; SET COUNT FOR 31/64 SECOND FOR BEEP
0784 E8 0000 E                CALL    BEEP                    ; SOUND THE POD BELL
0787 EB CC                    JMP     U5                      ; TTY_RETURN
0789                  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
0789 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

0791                  READ_LPEN       PROC    NEAR
0791 B4 00                    MOV     AH,0                    ; SET NO LIGHT PEN RETURN CODE
0793 8B 16 0063 R             MOV     DX,@ADDR_6845           ; GET BASE ADDRESS OF 6845
0797 83 C2 06                 ADD     DX,6                    ; POINT TO STATUS REGISTER
079A EC                       IN      AL,DX                   ; GET STATUS REGISTER
079B A8 04                    TEST    AL,004H                 ; TEST LIGHT PEN SWITCH
079D 74 03                    JZ      V6_A                    ; GO IF YES
079F E9 0823 R                JMP     V6                      ; NOT SET, RETURN

                      ;-----  NOW TEST FOR LIGHT PEN TRIGGER

07A2 A8 02            V6_A:   TEST    AL,2                    ; TEST LIGHT PEN TRIGGER
07A4 75 03                    JNZ     V7A                     ; RETURN WITHOUT RESETTING TRIGGER
07A6 E9 082D R                JMP     V7

                      ;-----  TRIGGER HAS BEEN SET, READ THE VALUE IN

07A9                  V7A:
07A9 B4 10                    MOV     AH,16                   ; LIGHT PEN REGISTERS ON 6845

                      ;-----  INPUT REGISTERS POINTED TO BY AH, AND CONVERT TO ROW COLUMN IN (DX)

07AB 8B 16 0063 R             MOV     DX,@ADDR_6845           ; ADDRESS REGISTER FOR 6845
07AF 8A C4                    MOV     AL,AH                   ; REGISTER TO READ
07B1 EE                       OUT     DX,AL                   ; SET IT UP
07B2 EB 00                    JMP     $+2                     ; I/O DELAY
07B4 42                       INC     DX                      ; DATA REGISTER
07B5 EC                       IN      AL,DX                   ; GET THE VALUE
07B6 8A E8                    MOV     CH,AL                   ; SAVE IN CX
07B8 4A                       DEC     DX                      ; ADDRESS REGISTER
07B9 FE C4                    INC     AH
07BB 8A C4                    MOV     AL,AH                   ; SECOND DATA REGISTER
```

```
07BD EE                           OUT   DX,AL
07BE 42                           INC   DX                 ; POINT TO DATA REGISTER
07BF EB 00                        JMP   $+2                ; I/O DELAY
07C1 EC                           IN    AL,DX              ; GET SECOND DATA VALUE
07C2 8A E5                        MOV   AH,CH              ; AX HAS INPUT VALUE

                          ;-----  AX HAS THE VALUE READ IN FROM THE 6845

07C4 8A 1E 0049 R                 MOV   BL,@CRT_MODE
07C8 2A FF                        SUB   BH,BH              ; MODE VALUE TO BX
07CA 2E: 8A 9F 0789 R             MOV   BL,CS:V1[BX]       ; DETERMINE AMOUNT TO SUBTRACT
07CF 2B C3                        SUB   AX,BX              ; TAKE IT AWAY
07D1 8B 1E 004E R                 MOV   BX,@CRT_START
07D5 D1 EB                        SHR   BX,1
07D7 2B C3                        SUB   AX,BX              ; CONVERT TO CORRECT PAGE ORIGIN
07D9 79 02                        JNS   V2                 ; IF POSITIVE, DETERMINE MODE
07DB 2B C0                        SUB   AX,AX              ; <0 PLAYS AS 0

                          ;-----  DETERMINE MODE OF OPERATION

07DD                      V2:                              ; DETERMINE_MODE
07DD B1 03                        MOV   CL,3               ; SET *8 SHIFT COUNT
07DF 80 3E 0049 R 04              CMP   @CRT_MODE,4        ; DETERMINE IF GRAPHICS OR ALPHA
07E4 72 29                        JB    V4                 ; ALPHA_PEN
07E6 80 3E 0049 R 07              CMP   @CRT_MODE,7
07EB 74 22                        JE    V4                 ; ALPHA_PEN

                          ;-----  GRAPHICS MODE

07ED B2 28                        MOV   DL,40              ; DIVISOR FOR GRAPHICS
07EF F6 F2                        DIV   DL                 ; DETERMINE ROW(AL) AND COLUMN(AH)
                                                           ;  AL RANGE 0-99, AH RANGE 0-39
                          ;-----  DETERMINE GRAPHIC ROW POSITION

07F1 8A E8                        MOV   CH,AL              ; SAVE ROW VALUE IN CH
07F3 02 ED                        ADD   CH,CH              ; *2 FOR EVEN/ODD FIELD
07F5 8A DC                        MOV   BL,AH              ; COLUMN VALUE TO BX
07F7 2A FF                        SUB   BH,BH              ; MULTIPLY BY 8 FOR MEDIUM RES
07F9 80 3E 0049 R 06              CMP   @CRT_MODE,6        ; DETERMINE MEDIUM OR HIGH RES
07FE 75 04                        JNE   V3                 ; NOT_HIGH_RES
0800 B1 04                        MOV   CL,4               ; SHIFT VALUE FOR HIGH RES
0802 D0 E4                        SAL   AH,1               ; COLUMN VALUE TIMES 2 FOR HIGH RES
0804                      V3:                              ; NOT_HIGH_RES
0804 D3 E3                        SHL   BX,CL              ; MULTIPLY *16 FOR HIGH RES

                          ;-----  DETERMINE ALPHA CHAR POSITION

0806 8A D4                        MOV   DL,AH              ; COLUMN VALUE FOR RETURN
0808 8A F0                        MOV   DH,AL              ; ROW VALUE
080A C0 EE 02                     SHR   DH,2               ; DIVIDE BY 4 FOR VALUE IN 0-24 RANGE
080D EB 12                        JMP   SHORT V5           ; LIGHT_PEN_RETURN_SET

                          ;-----  ALPHA MODE ON LIGHT PEN

080F                      V4:                              ; ALPHA_PEN
080F F6 36 004A R                 DIV   BYTE PTR @CRT_COLS ; DETERMINE ROW,COLUMN VALUE
0813 8A F0                        MOV   DH,AL              ; ROWS TO DH
0815 8A D4                        MOV   DL,AH              ; COLS TO DL
0817 D2 E0                        SAL   AL,CL              ; MULTIPLY ROWS * 8
0819 8A E8                        MOV   CH,AL              ; GET RASTER VALUE TO RETURN REGISTER
081B 8A DC                        MOV   BL,AH              ; COLUMN VALUE
081D 32 FF                        XOR   BH,BH              ;  TO BX
081F D3 E3                        SAL   BX,CL
0821                      V5:                              ; LIGHT_PEN_RETURN_SET
0821 B4 01                        MOV   AH,1               ; INDICATE EVERY THING SET
0823                      V6:                              ; LIGHT_PEN_RETURN
0823 52                           PUSH  DX                 ; SAVE RETURN VALUE (IN CASE)
0824 8B 16 0063 R                 MOV   DX,@ADDR_6845      ; GET BASE ADDRESS
0828 83 C2 07                     ADD   DX,7               ; POINT TO RESET PARM
082B EE                           OUT   DX,AL              ; ADDRESS, NOT DATA, IS IMPORTANT
082C 5A                           POP   DX                 ; RECOVER VALUE
082D                      V7:                              ; RETURN_NO_RESET
082D 5D                           POP   BP
082E 5F                           POP   DI
082F 5E                           POP   SI
0830 1F                           POP   DS                 ; DISCARD SAVED BX,CX,DX
0831 1F                           POP   DS
0832 1F                           POP   DS
0833 1F                           POP   DS
0834 07                           POP   ES
0835 CF                           IRET
0836                      READ_LPEN     ENDP
0836                      CODE    ENDS
                                  END
```

```
                              PAGE 118,121
                              TITLE BIOS ----- 11/15/85  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
```

SECTION 5

```
                                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 0D                             MOV     AL,CMOS_REG_D           ; 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 8D                             MOV     AL,CMOS_REG_D+NMI       ; MASK TRAP (NMI) INTERRUPTS OFF
 0025 E6 70                             OUT     CMOS_PORT,AL
 0027 B0 AD                             MOV     AL,DIS_KBD              ; DISABLE THE KEYBOARD
 0029 E8 0000 E                         CALL    C8042                   ; SEND COMMAND TO ADAPTER
 002C E8 0000 E                         CALL    DDS                     ; ADDRESS DATA SEGMENT
 002F B4 00                             MOV     AH,0                    ; INITIALIZE AND SET MODE FOR VIDEO
 0031 A0 0049 R                         MOV     AL,@CRT_MODE            ; GET CURRENT MODE
 0034 CD 10                             INT     10H                     ; CALL VIDEO_IO TO CLEAR SCREEN

                                ;-----  DISPLAY "PARITY CHECK ?" ERROR MESSAGES

 0036 58                                POP     AX                      ; RECOVER INITIAL CHECK STATUS
 0037 BE 0000 E                         MOV     SI,OFFSET D1            ; PLANAR ERROR, ADDRESS "PARITY CHECK 1"
 003A A8 80                             TEST    AL,PARITY_CHECK         ; CHECK FOR PLANAR ERROR
 003C 74 05                             JZ      NMI_2                   ; SKIP IF NOT

 003E 50                                PUSH    AX                      ; SAVE STATUS
 003F E8 0000 E                         CALL    P_MSG                   ; DISPLAY "PARITY CHECK 1" MESSAGE
 0042 58                                POP     AX                      ; AND RECOVER STATUS
 0043                           NMI_2:
 0043 BE 0000 E                         MOV     SI,OFFSET D2            ; ADDRESS OF "PARITY CHECK 2" MESSAGE
 0046 A8 40                             TEST    AL,IO_CHECK             ; I/O PARITY CHECK ?
 0048 74 03                             JZ      NMI_3                   ; SKIP IF CORRECT ERROR DISPLAYED
 004A E8 0000 E                         CALL    P_MSG                   ; DISPLAY "PARITY CHECK 2" ERROR

                                ;-----  TEST FOR HOT NMI ON PLANAR PARITY LINE

 004D                           NMI_3:
 004D E4 61                             IN      AL,PORT_B
 004F 0C 0C                             OR      AL,RAM_PAR_OFF          ; TOGGLE PARITY CHECK ENABLES
 0051 E6 61                             OUT     PORT_B,AL
 0053 24 F3                             AND     AL,RAM_PAR_ON           ; TO CLEAR THE PENDING CHECK
 0055 E6 61                             OUT     PORT_B,AL

 0057 FC                                CLD                             ; SET DIRECTION FLAG TO INCREMENT
 0058 2B D2                             SUB     DX,DX                   ; POINT (DX) AT START OF REAL MEMORY
 005A 2B F6                             SUB     SI,SI                   ; SET (SI) TO START OF (DS:)
 005C E4 61                             IN      AL,PORT_B               ; READ CURRENT PARITY CHECK LATCH
 005E A8 C0                             TEST    AL,PARITY_ERR           ; CHECK FOR HOT NMI SOURCE
 0060 75 19                             JNZ     NMI_5                   ; SKIP IF ERROR NOT RESET (DISPLAY ???)

                                ;-----  SEE IF LOCATION THAT CAUSED PARITY CHECK CAN BE FOUND IN BASE MEMORY

 0062 8B 1E 0013 R                      MOV     BX,@MEMORY_SIZE         ; GET BASE MEMORY SIZE WORD
 0066                           NMI_4:
 0066 8E DA                             MOV     DS,DX                   ; POINT TO 64K SEGMENT
 0068 B9 8000                           MOV     CX,4000H*2              ; SET WORD COUNT FOR 64 KB SCAN
 006B F3/ AD                            REP     LODSW                   ; READ 64 KB OF MEMORY
 006D E4 61                             IN      AL,PORT_B               ; READ PARITY CHECK LATCHES
 006F A8 C0                             TEST    AL,PARITY_ERR           ; CHECK FOR ANY PARITY ERROR PENDING
 0071 75 10                             JNZ     NMI_6                   ; GO PRINT SEGMENT ADDRESS IF ERROR

 0073 80 C6 10                          ADD     DH,010H                 ; POINT TO NEXT 64K BLOCK
 0076 83 EB 40                          SUB     BX,16D*4                ; DECREMENT COUNT OF 1024 BYTE SEGMENTS
 0079 77 EB                             JA      NMI_4                   ; LOOP TILL ALL 64K SEGMENTS DONE
 007B                           NMI_5:
 007B BE 0000 E                         MOV     SI,OFFSET D2A           ; PRINT ROW OF ????? IF PARITY
 007E E8 0000 E                         CALL    P_MSG                   ; CHECK COULD NOT BE RE-CREATED
 0081 FA                                CLI
 0082 F4                                HLT                             ; HALT SYSTEM

 0083                           NMI_6:
 0083 E8 0000 E                         CALL    PRT_SEG                 ; PRINT SEGMENT VALUE (IN DX)
 0086 B0 28                             MOV     AL,'('                  ; PRINT (S)
 0088 E8 0000 E                         CALL    PRT_HEX
 008B B0 53                             MOV     AL,'S'
 008D E8 0000 E                         CALL    PRT_HEX
 0090 B0 29                             MOV     AL,')'
 0092 E8 0000 E                         CALL    PRT_HEX
 0095 FA                                CLI                             ; HALT SYSTEM
 0096 F4                                HLT

 0097                           NMI_INT_1 ENDP

 0097                           CODE    ENDS
                                        END
```

```
                              PAGE 118,123
                              TITLE BIOS1 ---- 11/15/85  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                :
                              ;                                                                           :
```

SECTION 5

```
                              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 4E                         JB      C1                   ; RETURN IF 00-7FH
 0006 80 FC C0                      CMP     AH,0C0H              ; CHECK FOR CONFIGURATION PARAMETERS
 0009 74 51                         JE      CONF_PARMS
 000B 80 EC 80                      SUB     AH,080H              ; BASE ON 0
 000E 0A E4                         OR      AH,AH
 0010 74 48                         JZ      DEV_OPEN             ; DEVICE OPEN
 0012 FE CC                         DEC     AH
 0014 74 44                         JZ      DEV_CLOSE            ; DEVICE CLOSE
 0016 FE CC                         DEC     AH
 0018 74 40                         JZ      PROG_TERM            ; PROGRAM TERMINATION
 001A FE CC                         DEC     AH
 001C 74 47                         JZ      EVENT_WAIT           ; EVENT WAIT
 001E FE CC                         DEC     AH
 0020 75 03                         JNZ     NOT_JOYSTICK
 0022 E9 00D0 R                     JMP     JOY_STICK            ; JOYSTICK BIOS
 0025                         NOT_JOYSTICK:
 0025 FE CC                         DEC     AH
 0027 74 31                         JZ      SYS_REQ              ; SYSTEM REQUEST KEY
 0029 FE CC                         DEC     AH
 002B 74 07                         JZ      C1_A                 ; WAIT
 002D FE CC                         DEC     AH
 002F 75 06                         JNZ     C1_B
 0031 E9 01CA R                     JMP     BLOCKMOVE            ; MOVE BLOCK

 0034 E9 016A R               C1_A: JMP     WAIT                 ; WAIT

 0037 FE CC                   C1_B: DEC     AH

 0039 75 03                         JNZ     C1_C
 003B E9 03EE R                     JMP     EXT_MEMORY           ; GO GET THE EXTENDED MEMORY

 003E FE CC                   C1_C: DEC     AH
 0040 75 03                         JNZ     C1_D                 ; CHECK FOR FUNCTION 89H
 0042 E9 03FA R                     JMP     SET_VMODE            ; SWAP TO VIRTUAL MODE

 0045 80 EC 07                C1_D: SUB     AH,7                 ; CHECK FOR FUNCTION 90H
 0048 75 03                         JNZ     C1_E                 ; GO IF NOT
 004A E9 0483 R                     JMP     DEVICE_BUSY

 004D FE CC                   C1_E: DEC     AH                   ; CHECK FOR FUNCTION 8BH
 004F 75 03                         JNZ     C1                   ; GO IF NOT
 0051 E9 0487 R                     JMP     INT_COMPLETE

 0054 B4 86                   C1:   MOV     AH,86H               ; SET BAD COMMAND
 0056 F9                            STC                          ; SET CARRY FLAG ON
 0057                         C1_F:
 0057 CA 0002                       RET     2                    ; FAR RETURN EXIT FROM ROUTINES


 005A                         DEV_OPEN:                          ; NULL HANDLERS

 005A                         DEV_CLOSE:

 005A                         PROG_TERM:

 005A                         SYS_REQ:
 005A EB FB                         JMP     C1_F                 ; RETURN
 005C                         CASSETTE_IO_1   ENDP

 005C                         CONF_PARMS      PROC    NEAR
 005C 0E                            PUSH    CS                   ; GET CODE SEGMENT
 005D 07                            POP     ES                   ; PLACE IN SELECTOR POINTER
 005E BB 0000 E                     MOV     BX,OFFSET CONF_TBL   ; GET OFFSET OF PARAMETER TABLE
 0061 32 E4                         XOR     AH,AH                ; CLEAR AH AND SET CARRY OFF
 0063 EB F2                         JMP     C1_F                 ; EXIT THROUGH COMMON RETURN
 0065                         CONF_PARMS      ENDP

 0065                         EVENT_WAIT      PROC    NEAR
                                    ASSUME  DS:DATA
 0065 1E                            PUSH    DS                   ; SAVE
 0066 E8 0000 E                     CALL    DDS
 0069 0A C0                         OR      AL,AL
 006B 74 08                         JZ      EVENT_WAIT_2         ; GO IF ZERO
 006D FE C8                         DEC     AL                   ; CHECK IF 1
 006F 74 45                         JZ      EVENT_WAIT_3
 0071 1F                            POP     DS                   ; RESTORE DATA SEGMENT
 0072 F9                            STC                          ; SET CARRY
 0073 EB E2                         JMP     C1_F                 ; EXIT

 0075                         EVENT_WAIT_2:
 0075 FA                            CLI                          ; NO INTERRUPTS ALLOWED
 0076 F6 06 00A0 R 01               TEST    @RTC_WAIT_FLAG,01    ; CHECK FOR FUNCTION ACTIVE
 007B 74 05                         JZ      EVENT_WAIT_1
 007D FB                            STI                          ; ENABLE INTERRUPTS
 007E 1F                            POP     DS
 007F F9                            STC                          ; SET ERROR
 0080 EB D5                         JMP     C1_F                 ; RETURN
```

```
0082                          EVENT_WAIT_1:
0082 E4 A1                            IN      AL,INTB01               ; ENSURE INTERRUPT UNMASKED
0084 EB 00                            JMP     $+2
0086 24 FE                            AND     AL,0FEH
0088 E6 A1                            OUT     INTB01,AL
008A 8C 06 009A R                     MOV     @USER_FLAG_SEG,ES       ; SET UP TRANSFER TABLE
008E 89 1E 0098 R                     MOV     @USER_FLAG,BX
0092 89 0E 009E R                     MOV     @RTC_HIGH,CX
0096 89 16 009C R                     MOV     @RTC_LOW,DX
009A C6 06 00A0 R 01                  MOV     @RTC_WAIT_FLAG,01       ; SET ON FUNCTION ACTIVE SWITCH
009F B0 0B                            MOV     AL,CMOS_REG_B           ; ENABLE PIE
00A1 E8 0000 E                        CALL    CMOS_READ               ; READ CMOS LOCATION
00A4 24 7F                            AND     AL,07FH                 ; CLEAR SET
00A6 0C 40                            OR      AL,040H                 ; ENABLE PIE
00A8 50                               PUSH    AX                      ; SAVE AH
00A9 8A E0                            MOV     AH,AL                   ; PLACE DATA INTO DATA REGISTER
00AB B0 0B                            MOV     AL,CMOS_REG_B           ; ADDRESS ALARM REGISTER
00AD E8 0000 E                        CALL    CMOS_WRITE              ; PLACE DATA IN AH INTO ALARM REGISTER
00B0 58                               POP     AX                      ; RESTORE AH
00B1 1F                               POP     DS
00B2 FB                               STI                             ; ENABLE INTERRUPTS
00B3 F8                               CLC                             ; CLEAR CARRY
00B4 EB A1                            JMP     C1_F

                              ;-----  CANCEL

00B6                          EVENT_WAIT_3:
00B6 50                               PUSH    AX                      ; SAVE
00B7 FA                               CLI                             ; DISABLE INTERRUPTS
00B8 B8 0B0B                          MOV     AX,X*CMOS_REG_B         ; TURN OFF PIE
00BB E8 0000 E                        CALL    CMOS_READ               ; GET ALARM REGISTER
00BE 24 BF                            AND     AL,0BFH                 ; CLEAR PIE
00C0 86 E0                            XCHG    AH,AL                   ; PLACE INTO WRITE REGISTER
00C2 E8 0000 E                        CALL    CMOS_WRITE              ; WRITE BACK TO ALARM REGISTER
00C5 58                               POP     AX                      ; RESTORE AH
00C6 C6 06 00A0 R 00                  MOV     @RTC_WAIT_FLAG,0        ; SET FUNCTION ACTIVE FLAG OFF
00CB FB                               STI                             ; ENABLE INTERRUPTS
00CC 1F                               POP     DS                      ; RESTORE DATA SEGMENT
00CD F8                               CLC                             ; SET CARRY OFF
00CE EB 87                            JMP     C1_F                    ; RETURN

00D0                          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         :
                              ;-------------------------------------------------------------

00D0                          JOY_STICK       PROC    NEAR
00D0 FB                               STI                             ; INTERRUPTS BACK ON
00D1 8B C2                            MOV     AX,DX                   ; GET SUB FUNCTION CODE
00D3 BA 0201                          MOV     DX,201H                 ; ADDRESS OF PORT
00D6 0A C0                            OR      AL,AL
00D8 74 0B                            JZ      JOY_2                   ; READ SWITCHES
00DA FE C8                            DEC     AL
00DC 74 0C                            JZ      JOY_3                   ; READ RESISTIVE INPUTS
00DE E9 0054 R                        JMP     C1                      ; GO TO ERROR RETURN
00E1                          JOY_1:
00E1 FB                               STI
00E2 E9 0057 R                        JMP     C1_F                    ; GO TO COMMON RETURN

00E5                          JOY_2:
00E5 EC                               IN      AL,DX
00E6 24 F0                            AND     AL,0F0H                 ; STRIP UNWANTED BITS OFF
00E8 EB F7                            JMP     JOY_1                   ; FINISHED

00EA                          JOY_3:
00EA B3 01                            MOV     BL,1
00EC E8 0108 R                        CALL    TEST_CORD
00EF 51                               PUSH    CX                      ; SAVE A(X) VALUE
00F0 B3 02                            MOV     BL,2
00F2 E8 0108 R                        CALL    TEST_CORD
00F5 51                               PUSH    CX                      ; SAVE A(Y) VALUE
00F6 B3 04                            MOV     BL,4
00F8 E8 0108 R                        CALL    TEST_CORD
00FB 51                               PUSH    CX                      ; SAVE B(X) VALUE
00FC B3 08                            MOV     BL,8
00FE E8 0108 R                        CALL    TEST_CORD
0101 8B D1                            MOV     DX,CX                   ; SAVE B(Y) VALUE
0103 59                               POP     CX                      ; GET B(X) VALUE
0104 5B                               POP     BX                      ; GET A(Y) VALUE
0105 58                               POP     AX                      ; GET A(X) VALUE
0106 EB D9                            JMP     JOY_1                   ; FINISHED - RETURN

0108                          TEST_CORD       PROC    NEAR
0108 52                               PUSH    DX                      ; SAVE
0109 FA                               CLI                             ; BLOCK INTERRUPTS WHILE READING
010A B0 00                            MOV     AL,0                    ; SET UP TO LATCH TIMER 0
010C E6 43                            OUT     TIMER+3,AL
010E EB 00                            JMP     $+2
0110 E4 40                            IN      AL,TIMER                ; READ LOW BYTE OF TIMER 0
0112 EB 00                            JMP     $+2
0114 8A E0                            MOV     AH,AL
0116 E4 40                            IN      AL,TIMER                ; READ HIGH BYTE OF TIMER 0
0118 86 E0                            XCHG    AH,AL                   ; REARRANGE TO HIGH,LOW
011A 50                               PUSH    AX                      ; SAVE
011B B9 04FF                          MOV     CX,4FFH                 ; SET COUNT
011E EE                               OUT     DX,AL                   ; FIRE TIMER
011F EB 00                            JMP     $+2
0121                          TEST_CORD_1:
0121 EC                               IN      AL,DX                   ; READ VALUES
0122 84 C3                            TEST    AL,BL                   ; HAS PULSE ENDED?
0124 E0 FB                            LOOPNZ  TEST_CORD_1
```

```
0126 83 F9 00                   CMP     CX,0
0129 59                         POP     CX                      ; ORIGINAL COUNT
012A 75 04                      JNZ     SHORT TEST_CORD_2
012C 2B C9                      SUB     CX,CX                   ; SET 0 COUNT FOR RETURN
012E EB 28                      JMP     SHORT TEST_CORD_3       ; EXIT WITH COUNT = 0
0130                    TEST_CORD_2:
0130 B0 00                      MOV     AL,0                    ; SET UP TO LATCH TIMER 0
0132 E6 43                      OUT     TIMER+3,AL
0134 EB 00                      JMP     $+2
0136 E4 40                      IN      AL,TIMER                ; READ LOW BYTE OF TIMER 0
0138 8A E0                      MOV     AH,AL
013A EB 00                      JMP     $+2
013C E4 40                      IN      AL,TIMER                ; READ HIGH BYTE OF TIMER 0
013E 86 E0                      XCHG    AH,AL                   ; REARRANGE TO HIGH,LOW

0140 3B C8                      CMP     CX,AX                   ; CHECK FOR COUNTER WRAP
0142 73 0B                      JAE     TEST_CORD_4             ; GO IF NO
0144 52                         PUSH    DX
0145 BA FFFF                    MOV     DX,-1

0148 2B D0                      SUB     DX,AX                   ; ADJUST FOR WRAP
014A 03 CA                      ADD     CX,DX
014C 5A                         POP     DX
014D EB 02                      JMP     SHORT TEST_CORD_5

014F                    TEST_CORD_4:
014F 2B C8                      SUB     CX,AX
0151                    TEST_CORD_5:
0151 81 E1 1FF0                 AND     CX,1FF0H                ; ADJUST
0155 C1 E9 04                   SHR     CX,4

0158                    TEST_CORD_3:
0158 FB                         STI                             ; INTERRUPTS BACK ON
0159 BA 0201                    MOV     DX,201H                 ; FLUSH OTHER INPUTS
015C 51                         PUSH    CX
015D 50                         PUSH    AX
015E B9 04FF                    MOV     CX,4FFH                 ; COUNT
0161                    TEST_CORD_6:
0161 EC                         IN      AL,DX
0162 A8 0F                      TEST    AL,0FH
0164 E0 FB                      LOOPNZ  TEST_CORD_6

0166 58                         POP     AX
0167 59                         POP     CX
0168 5A                         POP     DX                      ; SET COUNT

0169 C3                         RET                             ; RETURN

016A                    TEST_CORD       ENDP
016A                    JOY_STICK       ENDP

016A                    WAIT    PROC    NEAR
016A 1E                         PUSH    DS                      ; SAVE
016B E8 0000 E                  CALL    DDS
016E F6 06 00A0 R 01            TEST    @RTC_WAIT_FLAG,01       ; TEST FOR FUNCTION ACTIVE
0173 74 05                      JZ      WAIT_1
0175 1F                         POP     DS
0176 F9                         STC                             ; SET ERROR
0177 E9 0057 R                  JMP     C1_F                    ; RETURN
017A                    WAIT_1:
017A FA                         CLI                             ; NO INTERRUPTS ALLOWED
017B E4 A1                      IN      AL,INTB01               ; ENSURE INTERRUPT UNMASKED
017D EB 00                      JMP     $+2
017F 24 FE                      AND     AL,0FEH
0181 E6 A1                      OUT     INTB01,AL
0183 8C 1E 009A R               MOV     @USER_FLAG_SEG,DS       ; SET UP TRANSFER TABLE
0187 C7 06 0098 R 00A0 R        MOV     @USER_FLAG,OFFSET @RTC_WAIT_FLAG
018D 89 0E 009E R               MOV     @RTC_HIGH,CX
0191 89 16 009C R               MOV     @RTC_LOW,DX
0195 C6 06 00A0 R 01            MOV     @RTC_WAIT_FLAG,01       ; SET ON FUNCTION ACTIVE SWITCH
019A 50                         PUSH    AX                      ; SAVE (AH)
019B B8 0B0B                    MOV     AX,X*CMOS_REG_B         ; ENABLE PIE
019E E8 0000 E                  CALL    CMOS_READ               ; READ ALARM BYTE
01A1 24 7F                      AND     AL,07FH                 ; CLEAR SIT BIT
01A3 0C 40                      OR      AL,040H                 ; ENABLE PIE BIT
01A5 86 E0                      XCHG    AH,AL                   ; DATA TO WORK REGISTER
01A7 E8 0000 E                  CALL    CMOS_WRITE              ; WRITE NEW ALARM BYTE
01AA 58                         POP     AX                      ; RESTORE (AH)

                        ;-----  WAIT TILL RTC TIMEOUT POSTED   (WITH ERROR TIMEOUT)

01AB FB                         STI                             ; ENABLE INTERRUPTS
01AC 51                         PUSH    CX
01AD 52                         PUSH    DX                      ; SAVE CALLERS PARAMETERS
01AE 87 D1                      XCHG    DX,CX                   ; SWAP COUNT WORK REGISTERS
01B0                    WAIT_2:
01B0 F6 06 00A0 R 80            TEST    @RTC_WAIT_FLAG,080H     ; CHECK FOR END OF WAIT - CLEAR CARRY
01B5 E1 F9                      LOOPZ   WAIT_2                  ; DECREMENT TIMEOUT DELAY TILL WAIT END
01B7 75 05                      JNZ     WAIT_9                  ; EXIT IF RTC TIMER WAIT ENDED FLAG SET
01B9 83 EA 01                   SUB     DX,1                    ; DECREMENT ERROR TIMEOUT COUNTER
01BC 73 F2                      JNC     WAIT_2                  ; LOOP TILL COUNTERS TIMEOUT
01BE                    WAIT_9:
01BE C6 06 00A0 R 00            MOV     @RTC_WAIT_FLAG,0        ; SET FUNCTION INACTIVE
01C3 5A                         POP     DX
01C4 59                         POP     CX                      ; RESTORE CALLERS PARAMETERS
01C5 1F                         POP     DS
01C6 F8                         CLC                             ; CLEAR CARRY FLAG
01C7 E9 0057 R                  JMP     C1_F

01CA                    WAIT    ENDP
```

```
                                 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

 01CA                            BLOCKMOVE    PROC     NEAR

 01CA FC                                 CLD                        ; SET DIRECTION FORWARD
 01CB 60                                 PUSHA                      ; SAVE GENERAL PURPOSE REGISTERS
 01CC 06                                 PUSH    ES                 ; SAVE USERS EXTRA SEGMENT
 01CD 1E                                 PUSH    DS                 ; SAVE USERS DATA SEGMENT

                                 ;-----  SAVE THE CALLING ROUTINE"S STACK

 01CE E8 0000 E                          CALL    DDS                ; SET DS TO DATA AREA
```

SECTION 5

```
01D1 8C 16 0069 R                      MOV     @IO_ROM_SEG,SS          ; SAVE USERS STACK SEGMENT
01D5 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
01D9 8C C0                             MOV     AX,ES                   ; GET THE GDT DATA SEGMENT
01DB 8E D8                             MOV     DS,AX                   ; MOVE THE GDT SEGMENT POINTER TO (DS)
01DD 8A F4                             MOV     DH,AH                   ; BUILD HIGH BYTE OF THE 24 BIT ADDRESS
01DF C0 EE 04                          SHR     DH,4                    ; USE ONLY HIGH NIBBLE SHIFT - RIGHT 4
01E2 C1 E0 04                          SHL     AX,4                    ; STRIP HIGH NIBBLE FROM (AX)
01E5 03 C6                             ADD     AX,SI                   ; ADD THE GDT OFFSET TO DEVELOP LOW WORD
01E7 80 D6 00                          ADC     DH,0                    ; ADJUST HIGH BYTE IF CARRY FROM LOW

                              ;-----  SET THE GDT_LOC

01EA C7 44 08 FFFF                     MOV     [SI].CGDT_LOC.SEG_LIMIT,MAX_SEG_LEN
01EF 89 44 0A                          MOV     [SI].CGDT_LOC.BASE_LO_WORD,AX    ; SET THE LOW WORD
01F2 88 74 0C                          MOV     [SI].CGDT_LOC.BASE_HI_BYTE,DH    ; SET THE HIGH BYTE
01F5 C7 44 0E 0000                     MOV     [SI].CGDT_LOC.DATA_RESERVED,0    ; RESERVED

                              ;-----  SET UP THE CODE SEGMENT DESCRIPTOR

01FA C7 44 20 FFFF                     MOV     [SI].BIOS_CS.SEG_LIMIT,MAX_SEG_LEN
01FF C7 44 22 0000                     MOV     [SI].BIOS_CS.BASE_LO_WORD,CSEG@_LO     ; LOW WORD OF (CS)= 0
0204 C6 44 24 0F                       MOV     [SI].BIOS_CS.BASE_HI_BYTE,CSEG@_HI     ; HIGH BYTE OF (CS)= 0FH
0208 C6 44 25 9B                       MOV     [SI].BIOS_CS.DATA_ACC_RIGHTS,CPL0_CODE_ACCESS
020C 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) )

0211 8C D0                             MOV     AX,SS                   ; GET THE CURRENT STACK SEGMENT
0213 8A F4                             MOV     DH,AH                   ; FORM HIGH BYTE OF 24 BIT ADDRESS
0215 C0 EE 04                          SHR     DH,4                    ; FORM HIGH BYTE - SHIFT RIGHT 4
0218 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

021B C7 44 28 FFFF                     MOV     [SI].TEMP_SS.SEG_LIMIT,MAX_SEG_LEN   ; SET THE SS SEGMENT LIMIT
0220 89 44 2A                          MOV     [SI].TEMP_SS.BASE_LO_WORD,AX         ; SET THE LOW WORD
0223 88 74 2C                          MOV     [SI].TEMP_SS.BASE_HI_BYTE,DH         ; SET THE HIGH BYTE
0226 C6 44 2D 93                       MOV     [SI].TEMP_SS.DATA_ACC_RIGHTS,CPL0_DATA_ACCESS  ; SET CPL 0

                              ;-----  GATE ADDRESS BIT 20 ON  (DISABLE INTERRUPTS)

022A B4 DF                             MOV     AH,ENABLE_BIT20         ; GET ENABLE MASK
022C E8 03CC R                         CALL    GATE_A20                ; ENABLE A20 AND CLEAR INTERRUPTS
022F 3C 00                             CMP     AL,0                    ; WAS THE COMMAND ACCEPTED?
0231 74 06                             JZ      BL4                     ; GO IF YES

0233 B0 03                             MOV     AL,03H                  ; SET THE ERROR FLAG IF NOT
0235 E6 80                             OUT     MFG_PORT,AL
0237 EB 51                             JMP     SHORT SHUT9             ; EARLY ERROR EXIT

                              ;-----  SET SHUTDOWN RETURN ADDRESS AND DISABLE NMI
0239                          BL4:
0239 B8 098F                           MOV     AX,9*H+CMOS_SHUT_DOWN+NMI      ; SET THE SHUTDOWN BYTE LOCATION
023C E8 0000 E                         CALL    CMOS_WRITE                     ; TO SHUT DOWN 9 AND DISABLE NMI

                              ;-----  CLEAR EXCEPTION ERROR FLAG

023F 2A C0                             SUB     AL,AL
0241 E6 80                             OUT     MFG_PORT,AL             ; SET ERROR FLAG LOCATION TO 0

                              ;-----  LOAD THE IDT AND GDT

0243 BD 02C6 R                         MOV     BP,OFFSET ROM_IDT_LOC
                                       SEGOV   CS                      ; LOAD THE IDT
0246 2E                   +            DB      02EH
                                       LIDT    [BP]                    ;   REGISTER FROM THIS AREA
0247 0F                   +            DB      00FH
0248                      + ??0001     LABEL   BYTE
0248 8B 5E 00             +            MOV     BX,WORD PTR [BP]
024B                      + ??0002     LABEL   BYTE
0248                      +            ORG     OFFSET CS:??0001
0248 01                   +            DB      001H
024B                      +            ORG     OFFSET CS:??0002

                                       LGDT    [SI].CGDT_LOC           ; LOAD GLOBAL DESCRIPTOR TABLE REGISTER
024B 0F                   +            DB      00FH
024C                      + ??0003     LABEL   BYTE
024C 8B 54 08             +            MOV     DX,WORD PTR [SI].CGDT_LOC
024F                      + ??0004     LABEL   BYTE
024C                      +            ORG     OFFSET CS:??0003
024C 01                   +            DB      001H
024F                      +            ORG     OFFSET CS:??0004


                              ;-----  SWITCH TO VIRTUAL MODE

024F B8 0001                           MOV     AX,VIRTUAL_ENABLE       ; MACHINE STATUS WORD NEEDED TO
                                       LMSW    AX                      ;  SWITCH TO VIRTUAL MODE
0252 0F 01 F0             +            DB      00FH,001H,0F0H
0255 EA                                 DB     0EAH                    ; PURGE PRE-FETCH QUEUE WITH FAR JUMP
0256 025A R                             DW     OFFSET VIRT             ;  - TO OFFSET
0258 0020                               DW     BIOS_CS                 ;  - IN SEGMENT -PROTECTED MODE SELECTOR
025A                          VIRT:

                              ;-----  IN PROTECTED MODE - SETUP STACK SELECTOR AND SOURCE/TARGET SELECTORS

025A B8 0028                           MOV     AX,TEMP_SS              ; USER'S SS+SP IS NOT A DESCRIPTOR
025D 8E D0                             MOV     SS,AX                   ; LOAD STACK SELECTOR
025F B8 0010                           MOV     AX,SOURCE               ; GET THE SOURCE ENTRY
0262 8E D8                             MOV     DS,AX                   ; LOAD SOURCE SELECTOR
0264 B8 0018                           MOV     AX,TARGET               ; GET THE TARGET ENTRY
0267 8E C0                             MOV     ES,AX                   ; LOAD TARGET SELECTOR
0269 2B F6                             SUB     SI,SI                   ; SET SOURCE INDEX REGISTER TO ZERO
026B 2B FF                             SUB     DI,DI                   ; SET TARGET INDEX REGISTER TO ZERO

026D F3/ A5                            REP     MOVSW                   ; MOVE THE BLOCK COUNT PASSED IN (CX)

                              ;-----  CHECK FOR MEMORY PARITY BEFORE SHUTDOWN
```

```

026F E4 61                     IN      AL,PORT_B               ; GET THE PARITY LATCHES
0271 24 C0                     AND     AL,PARITY_ERR           ; STRIP UNWANTED BITS
0273 74 12                     JZ      DONE1                   ; GO IF NO PARITY ERROR

                        ;-----  CLEAR PARITY BEFORE SHUTDOWN

0275 8B 05                     MOV     AX,DS:[DI]              ; FETCH CURRENT SOURCE DATA
0277 89 05                     MOV     DS:[DI],AX              ; WRITE IT BACK
0279 B0 01                     MOV     AL,01                   ; SET PARITY CHECK ERROR = 01
027B E6 80                     OUT     MFG_PORT,AL
027D E4 61                     IN      AL,PORT_B
027F 0C 0C                     OR      AL,RAM_PAR_OFF          ; TOGGLE PARITY CHECK LATCHES
0281 E6 61                     OUT     PORT_B,AL               ;  TO CLEAR THE PENDING ERROR
0283 24 F3                     AND     AL,RAM_PAR_ON           ;  AND ENABLE CHECKING
0285 E6 61                     OUT     PORT_B,AL

                        ;-----  CAUSE A SHUTDOWN

0287                    DONE1:
0287 E9 0000 E                 JMP     PROC_SHUTDOWN           ; GO RESET PROCESSOR AND SHUTDOWN

                               ;=====================
                        ;------  RETURN FROM SHUTDOWN
                               ;=====================
028A                    SHUT9:                                 ;       RESTORE USERS STACK
                               ASSUME  DS:DATA
028A B8 ---- R                 MOV     AX,DATA                 ; SET DS TO DATA AREA
028D 8E D8                     MOV     DS,AX
028F 8E 16 0069 R              MOV     SS,@IO_ROM_SEG          ; GET USER STACK SEGMENT
0293 8B 26 0067 R              MOV     SP,@IO_ROM_INIT         ; GET USER STACK POINTER

                        ;-----  GATE ADDRESS BIT 20 OFF

0297 B4 DD                     MOV     AH,DISABLE_BIT20        ; DISABLE MASK
0299 E8 03CC R                 CALL    GATE_A20                ; GATE ADDRESS 20 LINE OFF
029C 3C 00                     CMP     AL,0                    ; COMMAND ACCEPTED?
029E 74 0A                     JZ      DONE3                   ; GO IF YES

02A0 E4 80                     IN      AL,MFG_PORT             ; CHECK FOR ANY OTHER ERROR FIRST
02A2 3C 00                     CMP     AL,0                    ; WAS THERE AN ERROR?
02A4 75 04                     JNZ     DONE3                   ; REPORT FIRST ERROR IF YES
02A6 B0 03                     MOV     AL,03H                  ; ELSE SET GATE A20 ERROR FLAG
02A8 E6 80                     OUT     MFG_PORT,AL

                        ;-----  RESTORE THE USERS REGISTERS AND SET RETURN CODES

02AA                    DONE3:
02AA B8 000D                   MOV     AX,CMOS_REG_D           ; CLEAR (AH) TO ZERO AND (AL) TO DEFAULT
02AD E6 70                     OUT     CMOS_PORT,AL            ; ENABLE NMI INTERRUPTS

02AF 1F                        POP     DS                      ; RESTORE USER DATA SEGMENT
02B0 07                        POP     ES                      ; RESTORE USER EXTRA SEGMENT
02B1 E4 80                     IN      AL,MFG_PORT             ; GET THE ENDING STATUS RETURN CODE
02B3 8B EC                     MOV     BP,SP                   ; POINT TO REGISTERS IN THE STACK
02B5 88 46 0F                  MOV     [BP+15],AL              ; PLACE ERROR CODE INTO STACK AT (AH)
02B8 3A E0                     CMP     AH,AL                   ; SET THE ZF & CY FLAGS WITH RETURN CODE
02BA 61                        POPA                            ; RESTORE THE GENERAL PURPOSE REGISTERS
02BB FB                        STI                             ; TURN INTERRUPTS ON
02BC                    DONE4  PROC    FAR
02BC CA 0002                   RET     2                       ; RETURN WITH FLAGS SET --  (AH)= CODE
02BF                    DONE4  ENDP                            ;  (CY=0,ZF=1)= OK    (CY=1,ZF=0)= ERROR

                        ;-----  BLOCK MOVE EXCEPTION INTERRUPT HANDLER

02BF                    EX_INT:
02BF B0 02                     MOV     AL,02H                  ; GET EXCEPTION ERROR CODE
02C1 E6 80                     OUT     MFG_PORT,AL             ; SET EXCEPTION INTERRUPT OCCURRED FLAG
02C3 E9 0000 E                 JMP     PROC_SHUTDOWN           ; CAUSE A EARLY SHUTDOWN

                        ;-----  ROM IDT LOCATION

02C6                    ROM_IDT_LOC:
02C6 0100                      DW      ROM_IDT_END-ROM_IDT     ; LENGTH OF ROM IDT TABLE
02C8 02CC R                    DW      ROM_IDT                 ; LOW WORD OF BASE ADDRESS
02CA 0F                        DB      CSEG@_HI                ; HIGH BYTE OF BASE ADDRESS
02CB 00                        DB      0                       ; RESERVED

                        ;-----  THE ROM EXCEPTION INTERRUPT VECTOR GATES FOR BLOCK MOVE

02CC                    ROM_IDT:                               ;       EXCEPTION 00
02CC 02BF R                    DW      EX_INT                  ; DESTINATION OFFSET
02CE 0020                      DW      BIOS_CS                 ; DESTINATION SEGMENT SELECTOR
02D0 00                        DB      0                       ; WORD COPY COUNT
02D1 87                        DB      TRAP_GATE               ; GATE TYPE - ACCESS RIGHTS BYTE
02D2 0000                      DW      0                       ; RESERVED
                                                               ;       EXCEPTION 01
02D4 02BF R                    DW      EX_INT                  ; DESTINATION OFFSET
02D6 0020                      DW      BIOS_CS                 ; DESTINATION SEGMENT SELECTOR
02D8 00                        DB      0                       ; WORD COPY COUNT
02D9 87                        DB      TRAP_GATE               ; GATE TYPE - ACCESS RIGHTS BYTE
02DA 0000                      DW      0                       ; RESERVED
                                                               ;       EXCEPTION 02
02DC 02BF R                    DW      EX_INT                  ; DESTINATION OFFSET
02DE 0020                      DW      BIOS_CS                 ; DESTINATION SEGMENT SELECTOR
02E0 00                        DB      0                       ; WORD COPY COUNT
02E1 87                        DB      TRAP_GATE               ; GATE TYPE - ACCESS RIGHTS BYTE
02E2 0000                      DW      0                       ; RESERVED
                                                               ;       EXCEPTION 03
02E4 02BF R                    DW      EX_INT                  ; DESTINATION OFFSET
02E6 0020                      DW      BIOS_CS                 ; DESTINATION SEGMENT SELECTOR
02E8 00                        DB      0                       ; WORD COPY COUNT
02E9 87                        DB      TRAP_GATE               ; GATE TYPE - ACCESS RIGHTS BYTE
02EA 0000                      DW      0                       ; RESERVED
                                                               ;       EXCEPTION 04
02EC 02BF R                    DW      EX_INT                  ; DESTINATION OFFSET
02EE 0020                      DW      BIOS_CS                 ; DESTINATION SEGMENT SELECTOR
02F0 00                        DB      0                       ; WORD COPY COUNT
02F1 87                        DB      TRAP_GATE               ; GATE TYPE - ACCESS RIGHTS BYTE
02F2 0000                      DW      0                       ; RESERVED
                                                               ;       EXCEPTION 05
02F4 02BF R                    DW      EX_INT                  ; DESTINATION OFFSET
02F6 0020                      DW      BIOS_CS                 ; DESTINATION SEGMENT SELECTOR
02F8 00                        DB      0                       ; WORD COPY COUNT
```

```
02F9 87              DB      TRAP_GATE       ; GATE TYPE - ACCESS RIGHTS BYTE
02FA 0000            DW      0               ; RESERVED
                                             ;        EXCEPTION 06
02FC 02BF R          DW      EX_INT          ; DESTINATION OFFSET
02FE 0020            DW      BIOS_CS         ; DESTINATION SEGMENT SELECTOR
0300 00              DB      0               ; WORD COPY COUNT
0301 87              DB      TRAP_GATE       ; GATE TYPE - ACCESS RIGHTS BYTE
0302 0000            DW      0               ; RESERVED
                                             ;        EXCEPTION 07
0304 02BF R          DW      EX_INT          ; DESTINATION OFFSET
0306 0020            DW      BIOS_CS         ; DESTINATION SEGMENT SELECTOR
0308 00              DB      0               ; WORD COPY COUNT
0309 87              DB      TRAP_GATE       ; GATE TYPE - ACCESS RIGHTS BYTE
030A 0000            DW      0               ; RESERVED
                                             ;        EXCEPTION 08
030C 02BF R          DW      EX_INT          ; DESTINATION OFFSET
030E 0020            DW      BIOS_CS         ; DESTINATION SEGMENT SELECTOR
0310 00              DB      0               ; WORD COPY COUNT
0311 87              DB      TRAP_GATE       ; GATE TYPE - ACCESS RIGHTS BYTE
0312 0000            DW      0               ; RESERVED
                                             ;        EXCEPTION 09
0314 02BF R          DW      EX_INT          ; DESTINATION OFFSET
0316 0020            DW      BIOS_CS         ; DESTINATION SEGMENT SELECTOR
0318 00              DB      0               ; WORD COPY COUNT
0319 87              DB      TRAP_GATE       ; GATE TYPE - ACCESS RIGHTS BYTE
031A 0000            DW      0               ; RESERVED
                                             ;        EXCEPTION 10
031C 02BF R          DW      EX_INT          ; DESTINATION OFFSET
031E 0020            DW      BIOS_CS         ; DESTINATION SEGMENT SELECTOR
0320 00              DB      0               ; WORD COPY COUNT
0321 87              DB      TRAP_GATE       ; GATE TYPE - ACCESS RIGHTS BYTE
0322 0000            DW      0               ; RESERVED
                                             ;        EXCEPTION 11
0324 02BF R          DW      EX_INT          ; DESTINATION OFFSET
0326 0020            DW      BIOS_CS         ; DESTINATION SEGMENT SELECTOR
0328 00              DB      0               ; WORD COPY COUNT
0329 87              DB      TRAP_GATE       ; GATE TYPE - ACCESS RIGHTS BYTE
032A 0000            DW      0               ; RESERVED
                                             ;        EXCEPTION 12
032C 02BF R          DW      EX_INT          ; DESTINATION OFFSET
032E 0020            DW      BIOS_CS         ; DESTINATION SEGMENT SELECTOR
0330 00              DB      0               ; WORD COPY COUNT
0331 87              DB      TRAP_GATE       ; GATE TYPE - ACCESS RIGHTS BYTE
0332 0000            DW      0               ; RESERVED
                                             ;        EXCEPTION 13
0334 02BF R          DW      EX_INT          ; DESTINATION OFFSET
0336 0020            DW      BIOS_CS         ; DESTINATION SEGMENT SELECTOR
0338 00              DB      0               ; WORD COPY COUNT
0339 87              DB      TRAP_GATE       ; GATE TYPE - ACCESS RIGHTS BYTE
033A 0000            DW      0               ; RESERVED
                                             ;        EXCEPTION 14
033C 02BF R          DW      EX_INT          ; DESTINATION OFFSET
033E 0020            DW      BIOS_CS         ; DESTINATION SEGMENT SELECTOR
0340 00              DB      0               ; WORD COPY COUNT
0341 87              DB      TRAP_GATE       ; GATE TYPE - ACCESS RIGHTS BYTE
0342 0000            DW      0               ; RESERVED
                                             ;        EXCEPTION 15
0344 02BF R          DW      EX_INT          ; DESTINATION OFFSET
0346 0020            DW      BIOS_CS         ; DESTINATION SEGMENT SELECTOR
0348 00              DB      0               ; WORD COPY COUNT
0349 87              DB      TRAP_GATE       ; GATE TYPE - ACCESS RIGHTS BYTE
034A 0000            DW      0               ; RESERVED
                                             ;        EXCEPTION 16
034C 02BF R          DW      EX_INT          ; DESTINATION OFFSET
034E 0020            DW      BIOS_CS         ; DESTINATION SEGMENT SELECTOR
0350 00              DB      0               ; WORD COPY COUNT
0351 87              DB      TRAP_GATE       ; GATE TYPE - ACCESS RIGHTS BYTE
0352 0000            DW      0               ; RESERVED
                                             ;        EXCEPTION 17
0354 02BF R          DW      EX_INT          ; DESTINATION OFFSET
0356 0020            DW      BIOS_CS         ; DESTINATION SEGMENT SELECTOR
0358 00              DB      0               ; WORD COPY COUNT
0359 87              DB      TRAP_GATE       ; GATE TYPE - ACCESS RIGHTS BYTE
035A 0000            DW      0               ; RESERVED
                                             ;        EXCEPTION 18
035C 02BF R          DW      EX_INT          ; DESTINATION OFFSET
035E 0020            DW      BIOS_CS         ; DESTINATION SEGMENT SELECTOR
0360 00              DB      0               ; WORD COPY COUNT
0361 87              DB      TRAP_GATE       ; GATE TYPE - ACCESS RIGHTS BYTE
0362 0000            DW      0               ; RESERVED
                                             ;        EXCEPTION 19
0364 02BF R          DW      EX_INT          ; DESTINATION OFFSET
0366 0020            DW      BIOS_CS         ; DESTINATION SEGMENT SELECTOR
0368 00              DB      0               ; WORD COPY COUNT
0369 87              DB      TRAP_GATE       ; GATE TYPE - ACCESS RIGHTS BYTE
036A 0000            DW      0               ; RESERVED
                                             ;        EXCEPTION 20
036C 02BF R          DW      EX_INT          ; DESTINATION OFFSET
036E 0020            DW      BIOS_CS         ; DESTINATION SEGMENT SELECTOR
0370 00              DB      0               ; WORD COPY COUNT
0371 87              DB      TRAP_GATE       ; GATE TYPE - ACCESS RIGHTS BYTE
0372 0000            DW      0               ; RESERVED
                                             ;        EXCEPTION 21
0374 02BF R          DW      EX_INT          ; DESTINATION OFFSET
0376 0020            DW      BIOS_CS         ; DESTINATION SEGMENT SELECTOR
0378 00              DB      0               ; WORD COPY COUNT
0379 87              DB      TRAP_GATE       ; GATE TYPE - ACCESS RIGHTS BYTE
037A 0000            DW      0               ; RESERVED
                                             ;        EXCEPTION 22
037C 02BF R          DW      EX_INT          ; DESTINATION OFFSET
037E 0020            DW      BIOS_CS         ; DESTINATION SEGMENT SELECTOR
0380 00              DB      0               ; WORD COPY COUNT
0381 87              DB      TRAP_GATE       ; GATE TYPE - ACCESS RIGHTS BYTE
0382 0000            DW      0               ; RESERVED
                                             ;        EXCEPTION 23
0384 02BF R          DW      EX_INT          ; DESTINATION OFFSET
0386 0020            DW      BIOS_CS         ; DESTINATION SEGMENT SELECTOR
0388 00              DB      0               ; WORD COPY COUNT
0389 87              DB      TRAP_GATE       ; GATE TYPE - ACCESS RIGHTS BYTE
038A 0000            DW      0               ; RESERVED
                                             ;        EXCEPTION 24
038C 02BF R          DW      EX_INT          ; DESTINATION OFFSET
038E 0020            DW      BIOS_CS         ; DESTINATION SEGMENT SELECTOR
0390 00              DB      0               ; WORD COPY COUNT
```

```
0391 87                 DB      TRAP_GATE       ; GATE TYPE - ACCESS RIGHTS BYTE
0392 0000               DW      0               ; RESERVED
                                                ;       EXCEPTION 25
0394 02BF R             DW      EX_INT          ; DESTINATION OFFSET
0396 0020               DW      BIOS_CS         ; DESTINATION SEGMENT SELECTOR
0398 00                 DB      0               ; WORD COPY COUNT
0399 87                 DB      TRAP_GATE       ; GATE TYPE - ACCESS RIGHTS BYTE
039A 0000               DW      0               ; RESERVED
                                                ;       EXCEPTION 26
039C 02BF R             DW      EX_INT          ; DESTINATION OFFSET
039E 0020               DW      BIOS_CS         ; DESTINATION SEGMENT SELECTOR
03A0 00                 DB      0               ; WORD COPY COUNT
03A1 87                 DB      TRAP_GATE       ; GATE TYPE - ACCESS RIGHTS BYTE
03A2 0000               DW      0               ; RESERVED
                                                ;       EXCEPTION 27
03A4 02BF R             DW      EX_INT          ; DESTINATION OFFSET
03A6 0020               DW      BIOS_CS         ; DESTINATION SEGMENT SELECTOR
03A8 00                 DB      0               ; WORD COPY COUNT
03A9 87                 DB      TRAP_GATE       ; GATE TYPE - ACCESS RIGHTS BYTE
03AA 0000               DW      0               ; RESERVED
                                                ;       EXCEPTION 28
03AC 02BF R             DW      EX_INT          ; DESTINATION OFFSET
03AE 0020               DW      BIOS_CS         ; DESTINATION SEGMENT SELECTOR
03B0 00                 DB      0               ; WORD COPY COUNT
03B1 87                 DB      TRAP_GATE       ; GATE TYPE - ACCESS RIGHTS BYTE
03B2 0000               DW      0               ; RESERVED
                                                ;       EXCEPTION 29
03B4 02BF R             DW      EX_INT          ; DESTINATION OFFSET
03B6 0020               DW      BIOS_CS         ; DESTINATION SEGMENT SELECTOR
03B8 00                 DB      0               ; WORD COPY COUNT
03B9 87                 DB      TRAP_GATE       ; GATE TYPE - ACCESS RIGHTS BYTE
03BA 0000               DW      0               ; RESERVED
                                                ;       EXCEPTION 30
03BC 02BF R             DW      EX_INT          ; DESTINATION OFFSET
03BE 0020               DW      BIOS_CS         ; DESTINATION SEGMENT SELECTOR
03C0 00                 DB      0               ; WORD COPY COUNT
03C1 87                 DB      TRAP_GATE       ; GATE TYPE - ACCESS RIGHTS BYTE
03C2 0000               DW      0               ; RESERVED
                                                ;       EXCEPTION 31
03C4 02BF R             DW      EX_INT          ; DESTINATION OFFSET
03C6 0020               DW      BIOS_CS         ; DESTINATION SEGMENT SELECTOR
03C8 00                 DB      0               ; WORD COPY COUNT
03C9 87                 DB      TRAP_GATE       ; GATE TYPE - ACCESS RIGHTS BYTE
03CA 0000               DW      0               ; RESERVED
03CC            ROM_IDT_END:

03CC            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.                 :
                                ;---------------------------------------------------------------------------
 03CC                           GATE_A20        PROC
 03CC 51                                PUSH    CX                      ; SAVE USERS (CX)
 03CD FA                                CLI                             ; DISABLE INTERRUPTS WHILE USING 8042
 03CE E8 03E5 R                         CALL    EMPTY_8042              ; INSURE 8042 INPUT BUFFER EMPTY
 03D1 75 10                             JNZ     GATE_A20_RETURN         ; EXIT IF 8042 UNABLE TO ACCEPT COMMAND
 03D3 B0 D1                             MOV     AL,0D1H                 ; 8042 COMMAND TO WRITE OUTPUT PORT
 03D5 E6 64                             OUT     STATUS_PORT,AL          ; OUTPUT COMMAND TO 8042
 03D7 E8 03E5 R                         CALL    EMPTY_8042              ; WAIT FOR 8042 TO ACCEPT COMMAND
 03DA 75 07                             JNZ     GATE_A20_RETURN         ; EXIT IF 8042 UNABLE TO ACCEPT COMMAND
 03DC 8A C4                             MOV     AL,AH                   ; 8042 PORT DATA
 03DE E6 60                             OUT     PORT_A,AL               ; OUTPUT PORT DATA TO 8042
 03E0 E8 03E5 R                         CALL    EMPTY_8042              ; WAIT FOR 8042 TO ACCEPT PORT DATA

                                ;-----  8042 OUTPUT WILL SWITCH WITHIN 20 MICRO SECONDS OF ACCEPTING PORT DATA

 03E3                           GATE_A20_RETURN:
 03E3 59                                POP     CX                      ; RESTORE USERS (CX)
 03E4 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                                              :
                                ;---------------------------------------------------------------------------
 03E5                           EMPTY_8042:
 03E5 2B C9                             SUB     CX,CX                   ; (CX)=0, WILL BE USED AS TIME OUT VALUE
 03E7                           EMPTY_L:
 03E7 E4 64                             IN      AL,STATUS_PORT          ; READ 8042 STATUS PORT
 03E9 24 02                             AND     AL,INPT_BUF_FULL        ; TEST INPUT BUFFER FULL FLAG (BIT 1)
 03EB E0 FA                             LOOPNZ  EMPTY_L                 ; LOOP UNTIL BUFFER EMPTY OR TIME OUT
 03ED C3                                RET
 03EE                           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.                        :
                                ;                                                                          :
                                ;---------------------------------------------------------------------------

 03EE                           EXT_MEMORY      PROC

 03EE B8 3031                           MOV     AX,CMOS_U_M_S_LO*H+CMOS_U_M_S_HI ; ADDRESS HIGH/LOW BYTES
 03F1 E8 0000 E                         CALL    CMOS_READ               ; GET THE HIGH BYTE OF I/O MEMORY
 03F4 86 C4                             XCHG    AL,AH                   ; PUT HIGH BYTE IN POSITION (AH)
 03F6 E8 0000 E                         CALL    CMOS_READ               ; GET THE LOW BYTE OF I/O MEMORY
 03F9 CF                                IRET                            ; RETURN TO USER

 03FA                           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.        ;
                                ;----------------------------------------------------------------------------
```

SECTION 5

```
                                 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 ????????????????                           DQ      ?                ; FIRST DESCRIPTOR NOT ACCESSIBLE
0008 ????????????????            GDTPTR         DQ      ?                ; GDT DESCRIPTOR
0010 ????????????????            IDTPTR         DQ      ?                ; IDT DESCRIPTOR
0018 ????????????????            USER_DS        DQ      ?                ; USER DATA SEGMENT DESCRIPTOR
0020 ????????????????            USER_ES        DQ      ?                ; USER EXTRA SEGMENT DESCRIPTOR
0028 ????????????????            USER_SS        DQ      ?                ; USER STACK SEGMENT DESCRIPTOR
0030 ????????????????            USER_CS        DQ      ?                ; USER CODE SEGMENT DESCRIPTOR
0038 ????????????????            BIO_CS         DQ      ?                ; TEMPORARY BIOS DESCRIPTOR
0040                             VIRTUAL_ENABLE_GDT_DEF  ENDS

                                         ASSUME  DS:DATA

03FA                             X_VIRTUAL       PROC    FAR
03FA                             SET_VMODE:

                                 ;-----  ENABLE ADDRESS LATCH BIT 20

03FA FA                                  CLI                             ; NO INTERRUPTS ALLOWED
03FB B4 DF                               MOV     AH,ENABLE_BIT20         ; ENABLE BIT 20 FOR ADDRESS GATE
03FD E8 03CC R                           CALL    GATE_A20
0400 3C 00                               CMP     AL,0                    ; WAS THE COMMAND ACCEPTED?
0402 74 04                               JZ      BIT20_ON                ; GO IF YES
0404 B4 FF                               MOV     AH,0FFH                 ; SET THE ERROR FLAG
0406 F9                                  STC                             ; SET CARRY
0407 CF                                  IRET                            ; EARLY EXIT


0408                             BIT20_ON:
0408 06                                  PUSH    ES                      ; MOVE SEGMENT POINTER
0409 1F                                  POP     DS                      ; TO THE DATA SEGMENT

                                 ;-------------------------------------------------------------------------------
                                 ; REINITIALIZE THE 8259 INTERRUPT CONTROLLER #1 TO THE USER SPECIFIED OFFSET   :
                                 ;-------------------------------------------------------------------------------

040A B0 11                               MOV     AL,11H                  ; START INITIALIZATION SEQUENCE-ICW1
040C E6 20                               OUT     INTA00,AL               ; EDGE,INTERVAL-8,MASTER,ICW4 NEEDED
040E EB 00                               JMP     $+2
0410 8A C7                               MOV     AL,BH                   ; HARDWARE INT'S START AT INT # (BH)
0412 E6 21                               OUT     INTA01,AL               ; SEND ICW2
0414 EB 00                               JMP     $+2
0416 B0 04                               MOV     AL,04H                  ; SEND ICW3 - MASTER LEVEL 2
0418 E6 21                               OUT     INTA01,AL
041A EB 00                               JMP     $+2
041C B0 01                               MOV     AL,01H                  ; SEND ICW4 - MASTER,8086 MODE
041E E6 21                               OUT     INTA01,AL
0420 EB 00                               JMP     $+2
0422 B0 FF                               MOV     AL,0FFH                 ; MASK OFF ALL INTERRUPTS
0424 E6 21                               OUT     INTA01,AL

                                 ;-------------------------------------------------------------------------------
                                 ; REINITIALIZE THE 8259 INTERRUPT CONTROLLER #2 TO THE USER SPECIFIED OFFSET  :
                                 ;-------------------------------------------------------------------------------

0426 B0 11                               MOV     AL,11H                  ; INITIALIZE SEQUENCE-ICW1 FOR SLAVE
0428 E6 A0                               OUT     INTB00,AL               ; EDGE,INTERVAL-8,MASTER,ICW4 NEEDED
042A EB 00                               JMP     $+2
042C 8A C3                               MOV     AL,BL                   ; HARDWARE INT'S START AT INT # (BL)
042E E6 A1                               OUT     INTB01,AL               ; SEND ICW2
0430 B0 02                               MOV     AL,02H
0432 EB 00                               JMP     $+2
0434 E6 A1                               OUT     INTB01,AL               ; SEND ICW3 - SLAVE LEVEL 2
0436 EB 00                               JMP     $+2
0438 B0 01                               MOV     AL,01H
043A E6 A1                               OUT     INTB01,AL               ; SEND ICW4 - SLAVE,8086 MODE
043C EB 00                               JMP     $+2
043E B0 FF                               MOV     AL,0FFH
0440 E6 A1                               OUT     INTB01,AL               ; MASK OFF ALL INTERRUPTS

                                 ;----------------------------------------
                                 ; SETUP BIOS CODE SEGMENT DESCRIPTOR    :
                                 ;----------------------------------------
```

```

0442 C7 44 38 FFFF                 MOV    [SI].BIO_CS.SEG_LIMIT,MAX_SEG_LEN        ; SET LENGTH
0447 C6 44 3C 0F                   MOV    [SI].BIO_CS.BASE_HI_BYTE,CSEG@_HI        ; SET HIGH BYTE OF CS=0F
044B C7 44 3A 0000                 MOV    [SI].BIO_CS.BASE_LO_WORD,CSEG@_LO        ; SET LOW WORD OF CS=0
0450 C6 44 3D 9B                   MOV    [SI].BIO_CS.DATA_ACC_RIGHTS,CPL0_CODE_ACCESS
0454 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
0459 0F                    +       DB     00FH
045A                       + ??0005 LABEL BYTE
045A 8B 54 08              +       MOV    DX,WORD PTR [SI].GDTPTR
045D                       + ??0006 LABEL BYTE
045A                       +       ORG    OFFSET CS:??0005
045A 01                    +       DB     001H
045D                       +       ORG    OFFSET CS:??0006
                                   LIDT   [SI].IDTPTR               ; INTERRUPT DESCRIPTOR TABLE REGISTER
045D 0F                    +       DB     00FH
045E                       + ??0007 LABEL BYTE
045E 8B 5C 10              +       MOV    BX,WORD PTR [SI].IDTPTR
0461                       + ??0008 LABEL BYTE
045E                       +       ORG    OFFSET CS:??0007
045E 01                    +       DB     001H
0461                       +       ORG    OFFSET CS:??0008

0461 B8 0001                       MOV    AX,VIRTUAL_ENABLE         ; MACHINE STATUS WORD NEEDED TO
                                   LMSW   AX                        ;  SWITCH TO VIRTUAL MODE
0464 0F 01 F0              +       DB     00FH,001H,0F0H
0467 EA                             DB    0EAH                      ; PURGE PRE-FETCH QUEUE WITH FAR JUMP
0468 046C R                         DW    OFFSET VMODE              ;  - TO OFFSET
046A 0038                           DW    BIO_CS                    ;  - IN SEGMENT -PROTECTED MODE SELECTOR

046C                       VMODE:
                           ;----------------------------------------
                           ;       SETUP USER SEGMENT REGISTERS     :
                           ;----------------------------------------
046C B8 0018                       MOV    AX,USER_DS                ; SETUP USER'S DATA SEGMENT
046F 8E D8                         MOV    DS,AX                     ; TO PROTECTED MODE SELECTORS
0471 B8 0020                       MOV    AX,USER_ES                ; SETUP USER'S EXTRA SEGMENT
0474 8E C0                         MOV    ES,AX
0476 B8 0028                       MOV    AX,USER_SS                ; SETUP USER'S STACK SEGMENT
0479 8E D0                         MOV    SS,AX
                           ;----------------------------------------
                           ;       PUT TRANSFER ADDRESS ON STACK    :
                           ;       AND RETURN TO THE USER           :
                           ;----------------------------------------
047B 5B                            POP    BX                        ; GET RETURN IP FROM THE STACK
047C 83 C4 04                      ADD    SP,4                      ; NORMALIZE STACK POINTER
047F 6A 30                         PUSH   USER_CS                   ; SET STACK FOR A RETURN FAR
0481 53                            PUSH   BX
0482 CB                            RET                              ; RETURN TO USER IN VIRTUAL MODE

0483                       X_VIRTUAL      ENDP

                           ;--- DEVICE BUSY AND INTERRUPT COMPLETE ------------------------
                           ;                                                             :
                           ;       THIS ROUTINE IS A TEMPORARY HANDLER FOR DEVICE BUSY   :
                           ;       AND INTERRUPT COMPLETE                                :
                           ;                                                             :
                           ;       INPUT   - SEE PROLOGUE                                :
                           ;--------------------------------------------------------------

0483                       DEVICE_BUSY    PROC    NEAR
0483 F8                            CLC                              ; TURN CARRY OFF
0484 E9 0057 R                     JMP    C1_F                      ; RETURN WITH CARRY FLAG
0487                       DEVICE_BUSY    ENDP

0487                       INT_COMPLETE   PROC    NEAR
0487 CF                            IRET                             ; RETURN
0488                       INT_COMPLETE   ENDP

0488                       CODE    ENDS
                                   END
```

SECTION 5

```
                                PAGE 118,123
                                TITLE BIOS2 ---- 11/15/85  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
```

```
                                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

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 4D                            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 30                            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 20                            JNC     RTC_I_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  C6 06 00A0 R 00                  MOV     @RTC_WAIT_FLAG,0        ; SET FUNCTION ACTIVE FLAG OFF
 01C6  C5 3E 0098 R                     LDS     DI,DWORD PTR @USER_FLAG ; SET UP (DS:DI) TO POINT TO USER FLAG
 01CA  C6 05 80                         MOV     BYTE PTR [DI],80H       ; TURN ON USERS FLAG
 01CD  58                               POP     AX                      ; GET INTERRUPT SOURCE BACK
 01CE                           RTC_I_5:
 01CE  A8 20                            TEST    AL,00100000B            ; TEST FOR ALARM INTERRUPT
 01D0  74 0A                            JZ      RTC_I_7                 ; SKIP USER INTERRUPT CALL IF NOT ALARM

 01D2  B0 0D                            MOV     AL,CMOS_REG_D           ; POINT TO DEFAULT READ ONLY REGISTER
 01D4  E6 70                            OUT     CMOS_PORT,AL            ; ENABLE NMI AND CMOS ADDRESS TO DEFAULT
 01D6  FB                               STI                             ; INTERRUPTS BACK ON NOW
 01D7  52                               PUSH    DX
 01D8  CD 4A                            INT     4AH                     ; TRANSFER TO USER ROUTINE
 01DA  5A                               POP     DX
 01DB  FA                               CLI                             ; BLOCK INTERRUPT FOR RETRY
 01DC                           RTC_I_7:                                ; RESTART ROUTINE TO HANDLE DELAYED
 01DC  EB A7                            JMP     RTC_I_1                 ;  ENTRY AND SECOND EVENT BEFORE DONE


 01DE                           RTC_I_9:                                ;       EXIT - NO PENDING INTERRUPTS
 01DE  B0 0D                            MOV     AL,CMOS_REG_D           ; POINT TO DEFAULT READ ONLY REGISTER
 01E0  E6 70                            OUT     CMOS_PORT,AL            ; ENABLE NMI AND CMOS ADDRESS TO DEFAULT
 01E2  B0 20                            MOV     AL,EOI                  ; END OF INTERRUPT MASK TO 8259 - 2
 01E4  E6 A0                            OUT     INTB00,AL               ;  TO 8259 - 2
 01E6  E6 20                            OUT     INTA00,AL               ;  TO 8259 - 1
 01E8  5F                               POP     DI                      ; RESTORE REGISTERS
 01E9  58                               POP     AX
 01EA  1F                               POP     DS
 01EB  CF                               IRET                            ; END OF INTERRUPT

 01EC                           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.                     :
                                ;-------------------------------------------------------------------------------

 01EC                           PRINT_SCREEN_1  PROC    FAR
                                                                        ; DELAY INTERRUPT ENABLE TILL FLAG SET
 01EC 1E                                PUSH    DS
 01ED 50                                PUSH    AX                      ; SAVE WORK REGISTERS
 01EE 53                                PUSH    BX
 01EF 51                                PUSH    CX
 01F0 52                                PUSH    DX                      ; USE 0040:0100 FOR STATUS AREA STORAGE
 01F1 E8 0000 E                         CALL    DDS                     ; GET STATUS_BYTE DATA SEGMENT
 01F4 80 3E 0100 R 01                   CMP     @STATUS_BYTE,1          ; SEE IF PRINT ALREADY IN PROGRESS
 01F9 74 74                             JE      PR190                   ; EXIT IF PRINT ALREADY IN PROGRESS
 01FB C6 06 0100 R 01                   MOV     @STATUS_BYTE,1          ; INDICATE PRINT NOW IN PROGRESS
 0200 FB                                STI                             ; MUST RUN WITH INTERRUPTS ENABLED
 0201 B4 0F                             MOV     AH,0FH                  ; WILL REQUEST THE CURRENT SCREEN MODE
 0203 CD 10                             INT     10H                     ;       (AL)= MODE
                                                                        ;       (AH)= NUMBER COLUMNS/LINE
                                                                        ;       (BH)= VISUAL PAGE
 0205 8A CC                             MOV     CL,AH                   ; WILL MAKE USE OF (CX) REGISTER TO
 0207 8A 2E 0084 R                      MOV     CH,@ROWS                ;  CONTROL ROWS ON SCREEN & COLUMNS
 020B 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.                       :
                                        ;----------------------------------------------------------------------
 020D 33 D2                             XOR     DX,DX                   ; FIRST PRINTER
 020F B4 02                             MOV     AH,02H                  ; SET PRINTER STATUS REQUEST COMMAND
 0211 CD 17                             INT     17H                     ; REQUEST CURRENT PRINTER STATUS
 0213 80 F4 80                          XOR     AH,080H                 ; CHECK FOR PRINTER BUSY (NOT CONNECTED)
 0216 F6 C4 A0                          TEST    AH,0A0H                 ;  OR  OUT OF PAPER
 0219 75 4E                             JNZ     PR180                   ; ERROR EXIT IF PRINTER STATUS ERROR

 021B E8 0275 R                         CALL    CRLF                    ; CARRIAGE RETURN LINE FEED TO PRINTER

 021E 51                                PUSH    CX                      ; SAVE SCREEN BOUNDS
 021F B4 03                             MOV     AH,03H                  ; NOW READ THE CURRENT CURSOR POSITION
 0221 CD 10                             INT     10H                     ; AND RESTORE AT END OF ROUTINE
 0223 59                                POP     CX                      ; RECALL SCREEN BOUNDS
 0224 52                                PUSH    DX                      ; PRESERVE THE ORIGINAL POSITION
 0225 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     :
                                        ;----------------------------------------------------------------------
 0227                           PR110:
 0227 B4 02                             MOV     AH,02H                  ; INDICATE CURSOR SET REQUEST
 0229 CD 10                             INT     10H                     ; NEW CURSOR POSITION ESTABLISHED
 022B B4 08                             MOV     AH,08H                  ; INDICATE READ CHARACTER FROM DISPLAY
 022D CD 10                             INT     10H                     ; CHARACTER NOW IN (AL)
 022F 0A C0                             OR      AL,AL                   ; SEE IF VALID CHAR
 0231 75 02                             JNZ     PR120                   ; JUMP IF VALID CHAR
 0233 B0 20                             MOV     AL,' '                  ; ELSE MAKE IT A BLANK
 0235                           PR120:
 0235 52                                PUSH    DX                      ; SAVE CURSOR POSITION
 0236 33 D2                             XOR     DX,DX                   ; INDICATE FIRST PRINTER (DX= 0)
 0238 32 E4                             XOR     AH,AH                   ; INDICATE PRINT CHARACTER IN (AL)
 023A CD 17                             INT     17H                     ; PRINT THE CHARACTER
 023C 5A                                POP     DX                      ; RECALL CURSOR POSITION
 023D F6 C4 29                          TEST    AH,29H                  ; TEST FOR PRINTER ERROR
 0240 75 22                             JNZ     PR170                   ; EXIT IF ERROR DETECTED
 0242 FE C2                             INC     DL                      ; ADVANCE TO NEXT COLUMN
 0244 3A CA                             CMP     CL,DL                   ; SEE IF AT END OF LINE
 0246 75 DF                             JNZ     PR110                   ; IF NOT LOOP FOR NEXT COLUMN
 0248 32 D2                             XOR     DL,DL                   ; BACK TO COLUMN 0
 024A 8A E2                             MOV     AH,DL                   ; (AH)=0
 024C 52                                PUSH    DX                      ; SAVE NEW CURSOR POSITION
 024D E8 0275 R                         CALL    CRLF                    ; LINE FEED CARRIAGE RETURN
 0250 5A                                POP     DX                      ; RECALL CURSOR POSITION
 0251 FE C6                             INC     DH                      ; ADVANCE TO NEXT LINE
 0253 3A EE                             CMP     CH,DH                   ; FINISHED?
 0255 75 D0                             JNZ     PR110                   ; IF NOT LOOP FOR NEXT LINE

 0257 5A                                POP     DX                      ; GET CURSOR POSITION
 0258 B4 02                             MOV     AH,02H                  ; INDICATE REQUEST CURSOR SET
 025A CD 10                             INT     10H                     ; CURSOR POSITION RESTORED
 025C FA                                CLI                             ; BLOCK INTERRUPTS TILL STACK CLEARED
 025D C6 06 0100 R 00                   MOV     @STATUS_BYTE,0          ; MOVE OK RESULTS FLAG TO STATUS_BYTE
 0262 EB 0B                             JMP     SHORT PR190             ; EXIT PRINTER ROUTINE

 0264                           PR170:                                  ;       ERROR EXIT
 0264 5A                                POP     DX                      ; GET CURSOR POSITION
 0265 B4 02                             MOV     AH,02H                  ; INDICATE REQUEST CURSOR SET
 0267 CD 10                             INT     10H                     ; CURSOR POSITION RESTORED
 0269                           PR180:
 0269 FA                                CLI                             ; BLOCK INTERRUPTS TILL STACK CLEARED
 026A C6 06 0100 R FF                   MOV     @STATUS_BYTE,0FFH       ; SET ERROR FLAG
 026F                           PR190:
 026F 5A                                POP     DX                      ;       EXIT ROUTINE
 0270 59                                POP     CX                      ; RESTORE ALL THE REGISTERS USED
 0271 5B                                POP     BX
 0272 58                                POP     AX
 0273 1F                                POP     DS
 0274 CF                                IRET                            ; RETURN WITH INITIAL INTERRUPT MASK
 0275                           PRINT_SCREEN_1  ENDP
```

```

                             ;-----  CARRIAGE RETURN, LINE FEED SUBROUTINE

0275                         CRLF    PROC    NEAR
                                                                   ;       SEND CR,LF TO FIRST PRINTER
0275 33 D2                           XOR     DX,DX                 ; ASSUME FIRST PRINTER (DX= 0)
0277 B8 000D                         MOV     AX,CR                 ; GET THE PRINT CHARACTER COMMAND AND
027A CD 17                           INT     17H                   ;  THE CARRIAGE RETURN CHARACTER
027C B8 000A                         MOV     AX,LF                 ; NOW GET THE LINE FEED AND
027F CD 17                           INT     17H                   ;  SEND IT TO THE BIOS PRINTER ROUTINE
0281 C3                              RET
0282                         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.           :
                             ;----------------------------------------------------------------------------

0282                         TIMER_INT_1     PROC    FAR
0282 FB                              STI                           ; INTERRUPTS BACK ON
0283 1E                              PUSH    DS
0284 50                              PUSH    AX
0285 52                              PUSH    DX                    ; SAVE MACHINE STATE
0286 E8 0000 E                       CALL    DDS                   ; ESTABLISH ADDRESSABILITY
0289 FF 06 006C R                    INC     @TIMER_LOW            ; INCREMENT TIME
028D 75 04                           JNZ     T4                    ; GO TO TEST_DAY
028F FF 06 006E R                    INC     @TIMER_HIGH           ; INCREMENT HIGH WORD OF TIME
0293                         T4:                                   ;       TEST_DAY
0293 83 3E 006E R 18                 CMP     @TIMER_HIGH,018H      ; TEST FOR COUNT EQUALING 24 HOURS
0298 75 15                           JNZ     T5                    ; GO TO DISKETTE_CTL
029A 81 3E 006C R 00B0               CMP     @TIMER_LOW,0B0H
02A0 75 0D                           JNZ     T5                    ; GO TO DISKETTE_CTL

                             ;-----  TIMER HAS GONE 24 HOURS

02A2 2B C0                           SUB     AX,AX
02A4 A3 006E R                       MOV     @TIMER_HIGH,AX
02A7 A3 006C R                       MOV     @TIMER_LOW,AX
02AA C6 06 0070 R 01                 MOV     @TIMER_OFL,1

                             ;-----  TEST FOR DISKETTE TIME OUT

02AF                         T5:
02AF FE 0E 0040 R                    DEC     @MOTOR_COUNT          ; DECREMENT DISKETTE MOTOR CONTROL
02B3 75 0B                           JNZ     T6                    ; RETURN IF COUNT NOT OUT
02B5 80 26 003F R F0                 AND     @MOTOR_STATUS,0F0H    ; TURN OFF MOTOR RUNNING BITS
02BA B0 0C                           MOV     AL,0CH
02BC BA 03F2                         MOV     DX,03F2H              ; FDC CTL PORT
02BF EE                              OUT     DX,AL                 ; TURN OFF THE MOTOR

02C0                         T6:                                   ;       TIMER TICK INTERRUPT
02C0 CD 1C                           INT     1CH                   ; TRANSFER CONTROL TO A USER ROUTINE

02C2 5A                              POP     DX                    ; RESTORE (DX)
02C3 B0 20                           MOV     AL,EOI                ; GET END OF INTERRUPT MASK
02C5 FA                              CLI                           ; DISABLE INTERRUPTS TILL STACK CLEARED
02C6 E6 20                           OUT     INTA00,AL             ; END OF INTERRUPT TO 8259 - 1
02C8 58                              POP     AX
02C9 1F                              POP     DS                    ; RESET MACHINE STATE
02CA CF                              IRET                          ; RETURN FROM INTERRUPT

02CB                         TIMER_INT_1     ENDP

02CB                         CODE    ENDS
                                     END
```

SECTION 5

```
                                PAGE 118,123
                                TITLE ORGS ----- 11/15/85  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  K9
                                        PUBLIC  K10
                                        PUBLIC  K11
                                        PUBLIC  K12
                                ;       PUBLIC  K13
                                        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 36 32 58 30 38 32                 DB      '62X0820 COPR. IBM 1981, 1985    '
      30 20 43 4F 50 52
      2E 20 49 42 4D 20
      31 39 38 31 2C 20
      31 39 38 35 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
```

SECTION 5

```
      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

= 0273                  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

                                ;----- DRIVE TYPE 07

 0461 01CE                              DW      0462D                   ; CYLINDERS
 0463 08                                DB      08D                     ; HEADS
 0464 0000                              DW      0
```

SECTION 5

```
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 00                    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

                   ;-----  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

                    ;-----  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       *** RESERVED***

0571 0000                   DW      0000D           ; CYLINDERS
0573 00                     DB      00D             ; HEADS
0574 0000                   DW      0
0576 0000                   DW      0000D           ; WRITE PRE-COMPENSATION CYL
0578 00                     DB      0
0579 00                     DB      0               ; CONTROL BYTE
057A 00 00 00               DB      0,0,0
057D 0000                   DW      0000D           ; LANDING ZONE
057F 00                     DB      00D             ; SECTORS/TRACK
0580 00                     DB      0
```

SECTION 5

```
                              ;-----   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
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
                              ;-----   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

                ;-----     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
```

SECTION 5

```
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

                   ;-----  DRIVE TYPE 47     *** RESERVED***

06E1 0000                  DW      0000D                   ; CYLINDERS
06E3 00                    DB      00D                     ; HEADS
06E4 0000                  DW      0
06E6 0000                  DW      0000D                   ; WRITE PRE-COMPENSATION CYL
06E8 00                    DB      0
06E9 00                    DB      0                       ; CONTROL BYTE
06EA 00 00 00              DB      0,0,0
06ED 0000                  DW      0000D                   ; LANDING ZONE
06EF 00                    DB      00D                     ; SECTORS/TRACK
06F0 00                    DB      0


                   ;-----  BOOT LOADER INTERRUPT

 = 06F1            IP      =       $
                   ;;-     ORG     0E6F2H
06F2                       ORG     006F2H
 = 06F2            BOOT_STRAP      EQU     $
06F2 E9 0000 E             JMP     BOOT_STRAP_1            ; VECTOR ON TO MOVED BOOT CODE

                                                           ;       USE INT 15 H  AH= 0C0H
06F5               CONF_TBL:                               ; CONFIGURATION TABLE FOR THIS SYSTEM
06F5 0008                  DW      CONF_E-CONF_TBL-2       ; LENGTH OF FOLLOWING TABLE
06F7 FC                    DB      MODEL_BYTE              ; SYSTEM MODEL BYTE
06F8 01                    DB      SUB_MODEL_BYTE          ; SYSTEM SUB MODEL TYPE BYTE
06F9 00                    DB      BIOS_LEVEL              ; BIOS REVISION LEVEL
06FA 70                    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
06FB 00                    DB      0                       ; RESERVED
06FC 00                    DB      0                       ; RESERVED
06FD 00                    DB      0                       ; RESERVED
06FE 00                    DB      0                       ; RESERVED
 = 06FF            CONF_E  EQU     $                       ; RESERVED FOR EXPANSION

                   ;-----  BAUD RATE INITIALIZATION TABLE

 = 06FF            IP      =       $
                   ;;-     ORG     0E729H
0729                       ORG     00729H
0729 0417          A1      DW      1047                    ; 110 BAUD      ; TABLE OF VALUES
072B 0300                  DW      768                     ; 150           ; FOR INITIALIZATION
072D 0180                  DW      384                     ; 300
072F 00C0                  DW      192                     ; 600
0731 0060                  DW      96                      ; 1200
0733 0030                  DW      48                      ; 2400
0735 0018                  DW      24                      ; 4800
0737 000C                  DW      12                      ; 9600

                   ;-----  RS232

                   ;;-     ORG     0E739H
```

```
0739                                ORG     00739H
= 0739                     RS232_IO        EQU     $
0739 E9 0000 E                      JMP     RS232_IO_1              ; VECTOR ON TO MOVED RS232 CODE
                           ;-----   KEYBOARD
                           ;;-      ORG     0E82EH
082E                                ORG     0082EH
= 082E                     KEYBOARD_IO     EQU     $
082E E9 0000 E                      JMP     KEYBOARD_IO_1           ; VECTOR ON TO MOVED KEYBOARD CODE
                           ;----------------------------------------------------------------
                           ;        KEY IDENTIFICATION SCAN TABLES                          -
                           ;----------------------------------------------------------------
                           ;;-      ORG     0E87EH
087E                                ORG     0087EH
                           ;--------------  TABLE OF SHIFT KEYS AND MASK VALUES  ----------------
                           ;------ KEY_TABLE
087E                       K6       LABEL   BYTE
087E 52                             DB      INS_KEY                 ; INSERT KEY
087F 3A 45 46 38 1D                 DB      CAPS_KEY,NUM_KEY,SCROLL_KEY,ALT_KEY,CTL_KEY
0884 2A 36                          DB      LEFT_KEY,RIGHT_KEY
= 0008                     K6L      EQU     $-K6
                           ;------ MASK_TABLE
0886                       K7       LABEL   BYTE
0886 80                             DB      INS_SHIFT               ; INSERT MODE SHIFT
0887 40 20 10 08 04                 DB      CAPS_SHIFT,NUM_SHIFT,SCROLL_SHIFT,ALT_SHIFT,CTL_SHIFT
088C 02 01                          DB      LEFT_SHIFT,RIGHT_SHIFT
                           ;----------------------  TABLES FOR CTRL CASE  ----------------------
088E                       K8       LABEL   BYTE                    ;-------- CHARACTERS ---------
088E 1B FF 00 FF FF FF              DB      27,-1,00,-1,-1,-1       ; Esc, 1, 2, 3, 4, 5
0894 1E FF FF FF FF 1F              DB      30,-1,-1,-1,-1,31       ; 6, 7, 8, 9, 0, -
089A FF 7F 94 11 17 05              DB      -1,127,148,17,23,5      ; =, Bksp, Tab, Q, W, E
08A0 12 14 19 15 09 0F              DB      18,20,25,21,09,15       ; R, T, Y, U, I, O
08A6 10 1B 1D 0A FF 01              DB      16,27,29,10,-1,01       ; P, [,], Enter, Ctrl, A
08AC 13 04 06 07 08 0A              DB      19,04,06,07,08,10       ; S, D, F, G, H, J
08B2 0B 0C FF FF FF FF              DB      11,12,-1,-1,-1,-1       ; K, L, ;, ', `, LShift
08B8 1C 1A 18 03 16 02              DB      28,26,24,03,22,02       ; \, Z, X, C, V, B
08BE 0E 0D FF FF FF FF              DB      14,13,-1,-1,-1,-1       ; N, M, ,, ., /, RShift
08C4 96 FF 20 FF                    DB      150,-1,' ',-1           ; *, Alt, Space, CL
                                                                    ;--------- FUNCTIONS ---------
08C8 5E 5F 60 61 62 63              DB      94,95,96,97,98,99       ; F1 - F6
08CE 64 65 66 67 FF FF              DB      100,101,102,103,-1,-1   ; F7 - F10, NL, SL
08D4 77 8D 84 8E 73 8F              DB      119,141,132,142,115,143 ; Home, Up, PgUp, -, Left, Pad5
08DA 74 90 75 91 76 92              DB      116,144,117,145,118,146 ; Right, +, End, Down, PgDn, Ins
08E0 93 FF FF FF 89 8A              DB      147,-1,-1,-1,137,138    ; Del, SysReq, Undef, WT, F11, F12
                           ;----------------------  TABLES FOR LOWER CASE  ----------------------
08E6                       K10      LABEL   BYTE
08E6 1B 31 32 33 34 35              DB      27,'12345'
08EC 36 37 38 39 30 2D              DB      '67890-'
08F2 3D 08 09 71 77 65              DB      '=',08,09,'qwe'
08F8 72 74 79 75 69 6F              DB      'rtyuio'
08FE 70 5B 5D 0D FF 61              DB      'p[]',0DH,-1,'a'        ; LETTERS, Return, Ctrl
0904 73 64 66 67 68 6A              DB      'sdfghj'
090A 6B 6C 3B 27 60 FF              DB      "kl;'`",-1              ; LETTERS, L Shift
0910 5C 7A 78 63 76 62              DB      '\zxcvb'
0916 6E 6D 2C 2E 2F                 DB      'nm,./'
091B FF 2A FF 20 FF                 DB      -1,'*',-1,' ',-1        ; R Shift,*, Alt, Space, CL
                           ;------ LC TABLE SCAN
0920 3B 3C 3D 3E 3F                 DB      59,60,61,62,63          ; BASE STATE OF F1 - F10
0925 40 41 42 43 44                 DB      64,65,66,67,68
092A FF FF                          DB      -1,-1                   ; NL, SL
                           ;------ KEYPAD TABLE
092C                       K15      LABEL   BYTE
092C 47 48 49 FF 4B FF              DB      71,72,73,-1,75,-1       ; BASE STATE OF KEYPAD KEYS
0932 4D FF 4F 50 51 52              DB      77,-1,79,80,81,82
0938 53                             DB      83
0939 FF FF 5C 85 86                 DB      -1,-1,'\',133,134       ; SysRq, Undef, WT, F11, F12
                           ;-----   KEYBOARD INTERRUPT
                           ;;-      ORG     0E987H
0987                                ORG     00987H
= 0987                     KB_INT   EQU     $
0987 E9 0000 E                      JMP     KB_INT_1                ; VECTOR ON TO MOVED KEYBOARD HANDLER
                           ;----------------------  TABLES FOR UPPER CASE  ----------------------
098A                       K11      LABEL   BYTE
098A 1B 21 40 23 24 25              DB      27,'!@#$%'
0990 5E 26 2A 28 29 5F              DB      '^&*()_'
0996 2B 08 00 51 57 45              DB      '+',08,00,'QWE'
099C 52 54 59 55 49 4F              DB      'RTYUIO'
09A2 50 7B 7D 0D FF 41              DB      'P{}',0DH,-1,'A'        ; LETTERS, Return, Ctrl
09A8 53 44 46 47 48 4A              DB      'SDFGHJ'
09AE 4B 4C 3A 22 7E FF              DB      'KL:"~',-1              ; LETTERS, L Shift
09B4 7C 5A 58 43 56 42              DB      '|ZXCVB'
09BA 4E 4D 3C 3E 3F                 DB      'NM<>?'
09BF FF 2A FF 20 FF                 DB      -1,'*',-1,' ',-1        ; R Shift,*, Alt, Space, CL
                           ;------ UC TABLE SCAN
09C4                       K12      LABEL   BYTE
09C4 54 55 56 57 58                 DB      84,85,86,87,88          ; SHIFTED STATE OF F1 - F10
09C9 59 5A 5B 5C 5D                 DB      89,90,91,92,93
09CE FF FF                          DB      -1,-1                   ; NL, SL
                           ;------ NUM STATE TABLE
09D0                       K14      LABEL   BYTE
09D0 37 38 39 2D 34 35              DB      '789-456+1230.'         ; NUMLOCK STATE OF KEYPAD KEYS
     36 2B 31 32 33 30
     2E
09DD FF FF 7C 87 88                 DB      -1,-1,'|',135,136       ; SysRq, Undef, WT, F11, F12
```

SECTION 5

```
                         PAGE
                         ;-----  DISKETTE I/O

                         ;;-     ORG     0EC59H
0C59                             ORG     00C59H
= 0C59                   DISKETTE_IO     EQU     $
0C59 E9 0000 E                   JMP     DISKETTE_IO_1          ; VECTOR ON TO MOVED DISKETTE CODE

                         ;-----  DISKETTE INTERRUPT

                         ;;-     ORG     0EF57H
0F57                             ORG     00F57H
= 0F57                   DISK_INT        EQU     $
0F57 E9 0000 E                   JMP     DISK_INT_1             ; VECTOR ON TO MOVED DISKETTE HANDLER

                         ;-----  DISKETTE PARAMETERS

                         ;;-     ORG     0EFC7H
0FC7                             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
0FD2                             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                DB      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
1841                            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
184D                            ORG     0184DH
= 184D                   EQUIPMENT      EQU     $
184D E9 0000 E                  JMP     EQUIPMENT_1             ; VECTOR ON TO MOVED BIOS CODE

                         ;----- CASSETTE (NO BIOS SUPPORT)

                         ;;-    ORG     0F859H
1859                            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
1A6E                            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
```

SECTION 5

```
1B6E 00 00 00 00 00 00          DB      000H,000H,000H,000H,000H,000H,000H,000H ; D_20    SPACE
     00 00
1B76 30 78 78 30 30 00          DB      030H,078H,078H,030H,030H,000H,030H,000H ; D_21 !  EXCLAMATION
     30 00
1B7E 6C 6C 6C 00 00 00          DB      06CH,06CH,06CH,000H,000H,000H,000H,000H ; D_22 "  QUOTATION
     00 00
1B86 6C 6C FE 6C FE 6C          DB      06CH,06CH,0FEH,06CH,0FEH,06CH,06CH,000H ; D_23 #  LB.
     6C 00
1B8E 30 7C C0 78 0C F8          DB      030H,07CH,0C0H,078H,00CH,0F8H,030H,000H ; D_24 $  DOLLAR SIGN
     30 00
1B96 00 C6 CC 18 30 66          DB      000H,0C6H,0CCH,018H,030H,066H,0C6H,000H ; D_25 %  PERCENT
     C6 00
1B9E 38 6C 38 76 DC CC          DB      038H,06CH,038H,076H,0DCH,0CCH,076H,000H ; D_26 &  AMPERSAND
     76 00
1BA6 60 60 C0 00 00 00          DB      060H,060H,0C0H,000H,000H,000H,000H,000H ; D_27 '  APOSTROPHE
     00 00
1BAE 18 30 60 60 60 30          DB      018H,030H,060H,060H,060H,030H,018H,000H ; D_28 (  L. PARENTHESIS
     18 00
1BB6 60 30 18 18 18 30          DB      060H,030H,018H,018H,018H,030H,060H,000H ; D_29 )  R. PARENTHESIS
     60 00
1BBE 00 66 3C FF 3C 66          DB      000H,066H,03CH,0FFH,03CH,066H,000H,000H ; D_2A *  ASTERISK
     00 00
1BC6 00 30 30 FC 30 30          DB      000H,030H,030H,0FCH,030H,030H,000H,000H ; D_2B +  PLUS
     00 00
1BCE 00 00 00 00 00 30          DB      000H,000H,000H,000H,000H,030H,030H,060H ; D_2C ,  COMMA
     30 60
1BD6 00 00 00 FC 00 00          DB      000H,000H,000H,0FCH,000H,000H,000H,000H ; D_2D -  DASH
     00 00
1BDE 00 00 00 00 00 30          DB      000H,000H,000H,000H,000H,030H,030H,000H ; D_2E .  PERIOD
     30 00
1BE6 06 0C 18 30 60 C0          DB      006H,00CH,018H,030H,060H,0C0H,080H,000H ; D_2F /  SLASH
     80 00

1BEE 7C C6 CE DE F6 E6          DB      07CH,0C6H,0CEH,0DEH,0F6H,0E6H,07CH,000H ; D_30 0
     7C 00
1BF6 30 70 30 30 30 30          DB      030H,070H,030H,030H,030H,030H,0FCH,000H ; D_31 1
     FC 00
1BFE 78 CC 0C 38 60 CC          DB      078H,0CCH,00CH,038H,060H,0CCH,0FCH,000H ; D_32 2
     FC 00
1C06 78 CC 0C 38 0C CC          DB      078H,0CCH,00CH,038H,00CH,0CCH,078H,000H ; D_33 3
     78 00
1C0E 1C 3C 6C CC FE 0C          DB      01CH,03CH,06CH,0CCH,0FEH,00CH,01EH,000H ; D_34 4
     1E 00
1C16 FC C0 F8 0C 0C CC          DB      0FCH,0C0H,0F8H,00CH,00CH,0CCH,078H,000H ; D_35 5
     78 00
1C1E 38 60 C0 F8 CC CC          DB      038H,060H,0C0H,0F8H,0CCH,0CCH,078H,000H ; D_36 6
     78 00
1C26 FC CC 0C 18 30 30          DB      0FCH,0CCH,00CH,018H,030H,030H,030H,000H ; D_37 7
     30 00
1C2E 78 CC CC 78 CC CC          DB      078H,0CCH,0CCH,078H,0CCH,0CCH,078H,000H ; D_38 8
     78 00
1C36 78 CC CC 7C 0C 18          DB      078H,0CCH,0CCH,07CH,00CH,018H,070H,000H ; D_39 9
     70 00
1C3E 00 30 30 00 00 30          DB      000H,030H,030H,000H,000H,030H,030H,000H ; D_3A :  COLON
     30 00
1C46 00 30 30 00 00 30          DB      000H,030H,030H,000H,000H,030H,030H,060H ; D_3B ;  SEMICOLON
     30 60
1C4E 18 30 60 C0 60 30          DB      018H,030H,060H,0C0H,060H,030H,018H,000H ; D_3C <  LESS THAN
     18 00
1C56 00 00 FC 00 00 FC          DB      000H,000H,0FCH,000H,000H,0FCH,000H,000H ; D_3D =  EQUAL
     00 00
1C5E 60 30 18 0C 18 30          DB      060H,030H,018H,00CH,018H,030H,060H,000H ; D_3E >  GREATER THAN
     60 00
1C66 78 CC 0C 18 30 00          DB      078H,0CCH,00CH,018H,030H,000H,030H,000H ; D_3F ?  QUESTION MARK
     30 00

1C6E 7C C6 DE DE DE C0          DB      07CH,0C6H,0DEH,0DEH,0DEH,0C0H,078H,000H ; D_40 @  AT
     78 00
1C76 30 78 CC CC FC CC          DB      030H,078H,0CCH,0CCH,0FCH,0CCH,0CCH,000H ; D_41 A
     CC 00
1C7E FC 66 66 7C 66 66          DB      0FCH,066H,066H,07CH,066H,066H,0FCH,000H ; D_42 B
     FC 00
1C86 3C 66 C0 C0 C0 66          DB      03CH,066H,0C0H,0C0H,0C0H,066H,03CH,000H ; D_43 C
     3C 00
1C8E F8 6C 66 66 66 6C          DB      0F8H,06CH,066H,066H,066H,06CH,0F8H,000H ; D_44 D
     F8 00
1C96 FE 62 68 78 68 62          DB      0FEH,062H,068H,078H,068H,062H,0FEH,000H ; D_45 E
     FE 00
1C9E FE 62 68 78 68 60          DB      0FEH,062H,068H,078H,068H,060H,0F0H,000H ; D_46 F
     F0 00
1CA6 3C 66 C0 C0 CE 66          DB      03CH,066H,0C0H,0C0H,0CEH,066H,03EH,000H ; D_47 G
     3E 00
1CAE CC CC CC FC CC CC          DB      0CCH,0CCH,0CCH,0FCH,0CCH,0CCH,0CCH,000H ; D_48 H
     CC 00
1CB6 78 30 30 30 30 30          DB      078H,030H,030H,030H,030H,030H,078H,000H ; D_49 I
     78 00
1CBE 1E 0C 0C 0C CC CC          DB      01EH,00CH,00CH,00CH,0CCH,0CCH,078H,000H ; D_4A J
     78 00
1CC6 E6 66 6C 78 6C 66          DB      0E6H,066H,06CH,078H,06CH,066H,0E6H,000H ; D_4B K
     E6 00
1CCE F0 60 60 60 62 66          DB      0F0H,060H,060H,060H,062H,066H,0FEH,000H ; D_4C L
     FE 00
1CD6 C6 EE FE FE D6 C6          DB      0C6H,0EEH,0FEH,0FEH,0D6H,0C6H,0C6H,000H ; D_4D M
     C6 00
1CDE C6 E6 F6 DE CE C6          DB      0C6H,0E6H,0F6H,0DEH,0CEH,0C6H,0C6H,000H ; D_4E N
     C6 00
1CE6 38 6C C6 C6 C6 6C          DB      038H,06CH,0C6H,0C6H,0C6H,06CH,038H,000H ; D_4F O
     38 00

1CEE FC 66 66 7C 60 60          DB      0FCH,066H,066H,07CH,060H,060H,0F0H,000H ; D_50 P
     F0 00
1CF6 78 CC CC CC DC 78          DB      078H,0CCH,0CCH,0CCH,0DCH,078H,01CH,000H ; D_51 Q
     1C 00
1CFE FC 66 66 7C 6C 66          DB      0FCH,066H,066H,07CH,06CH,066H,0E6H,000H ; D_52 R
     E6 00
1D06 78 CC E0 70 1C CC          DB      078H,0CCH,0E0H,070H,01CH,0CCH,078H,000H ; D_53 S
     78 00
1D0E FC B4 30 30 30 30          DB      0FCH,0B4H,030H,030H,030H,030H,078H,000H ; D_54 T
     78 00
1D16 CC CC CC CC CC CC          DB      0CCH,0CCH,0CCH,0CCH,0CCH,0CCH,0FCH,000H ; D_55 U
     FC 00
1D1E CC CC CC CC CC 78          DB      0CCH,0CCH,0CCH,0CCH,0CCH,078H,030H,000H ; D_56 V
     30 00
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_1          ; VECTOR ON TO MOVED BIOS CODE

                             ;-----  TIMER INTERRUPT

                             ;;-     ORG     0FEA5H
     1EA5                            ORG     01EA5H
     = 1EA5                  TIMER_INT       EQU     $
     1EA5 E9 0000 E                  JMP     TIMER_INT_1            ; VECTOR ON TO MOVED BIOS CODE
```

SECTION 5

```
                           PAGE
                           ;-----  VECTOR TABLE

                           ;;-     ORG     0FEF3H
 1EF3                              ORG     01EF3H                 ;       AT LOCATION 0FEF3H
 1EF3                      VECTOR_TABLE    LABEL   WORD           ; VECTOR TABLE VALUES FOR POST TESTS
 1EF3 1EA5 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  D11            ; INT 0AH - SLAVE INTERRUPT INPUT
 1EF9 0000 E                       DW      OFFSET  D11            ; INT 0BH -                         IRQ  3
 1EFB 0000 E                       DW      OFFSET  D11            ; INT 0CH -                         IRQ  4
 1EFD 0000 E                       DW      OFFSET  D11            ; INT 0DH -                         IRQ  5
 1EFF 0F57 R                       DW      OFFSET DISK_INT        ; INT 0EH - DISKETTE                IRQ  6
 1F01 0000 E                       DW      OFFSET  D11            ; 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_IO      ; 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  D11            ; INT 72H -                         IRQ 10
 1F29 0000 E                       DW      OFFSET  D11            ; INT 73H -                         IRQ 11
 1F2B 0000 E                       DW      OFFSET  D11            ; INT 74H -                         IRQ 12
 1F2D 0000 E                       DW      OFFSET  INT_287        ; INT 75H -  -MATH COPROCESSOR      IRQ 13
 1F2F 0000 E                       DW      OFFSET  D11            ; INT 76H -  -FIXED DISK            IRQ 14
 1F31 0000 E                       DW      OFFSET  D11            ; INT 77H -                         IRQ 15

                           ;-----  DUMMY INTERRUPT HANDLER

                           ;;-     ORG     0FF53H
 1F53                              ORG     01F53H

 = 1F53                    DUMMY_RETURN    EQU     $              ; BIOS DUMMY (NULL) INTERRUPT RETURN

 1F53 CF                           IRET

                           ;-----  PRINT SCREEN

                           ;;-     ORG     0FF54H
 1F54                              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
 1FF0                              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 31 31 2F 31 35 2F            DB      '11/15/85'             ; RELEASE MARKER
      38 35
                           ;
                           ;

                           ;


                           ;

 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

Contents

Notes:

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

DEC = 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

Immmediate 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

0000 0dw	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 $\neq$ 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 $\neq$ 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

11100000	disp

JCXZ = Jump on CX Zero

11100011	disp

ENTER = Enter Procedure

11001000	data-low	data-high

LEAVE = Leave Procedure

11001001

INT = Interrupt

Type Specified

11001101	Type

Type 3

11001100

INTO = Interrupt on Overflow

11001110

IRET = Interrupt Return

11001111

BOUND = Detect Value Out of Range

01100010	mod reg r/m

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

001reg001

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)

escape 001	11111101

FPREM = Partial Remainder of ST(0) ÷ ST(1)

escape 001	11111000

FRNDINT = Round ST(0) to Integer

escape 001	11111100

FXTRACT = Extract Components of ST(0)

escape 001	11110100

FABS = Absolute Value of ST(0)

escape 001	11100001

FCHS = Change Sign of ST(0)

escape 001	11100000

Transcendental

FPTAN = Partial Tangent of ST(0)

escape 001	11110010

FPATAN = Partial Arctangent of ST(0) ÷ ST(1)

escape 001	11110011

F2XM1 = $2^{ST(0)}$ -1

escape 001	11110000

FYL2X = ST(1) x Log_2 [ST(0)]

escape 001	11110001

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

escape 101	mod 110 r/m

FRSTOR = Restore State

escape 101	mod 100 r/m

FINCSTP = Increment Stack Pointer

escape 001	11110111

FDECSTP = Decrement Stack Pointer

escape 001	111100110

FFREE = Free ST(i)

escape 101	11000ST(i)

FNOP = No Operation

escape 101	11010000

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

Contents

Notes:

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	ƒ	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

SECTION 7

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	╖	╟	╫	ϒ	≈
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. COMMUNICATIONS

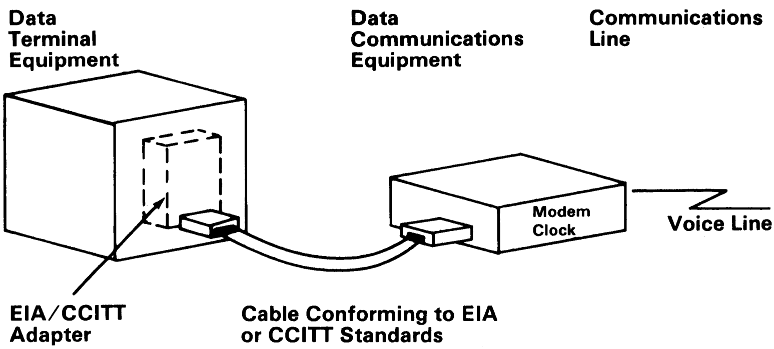
Contents

Notes:

Hardware

Information-processing equipment used for communication is called data terminal equipment (DTE.) Equipment used to connect the DTE to the communication line is called data communication equipment (DCE.)

An adapter connects the data terminal equipment to the data communication line as shown in the following figure:



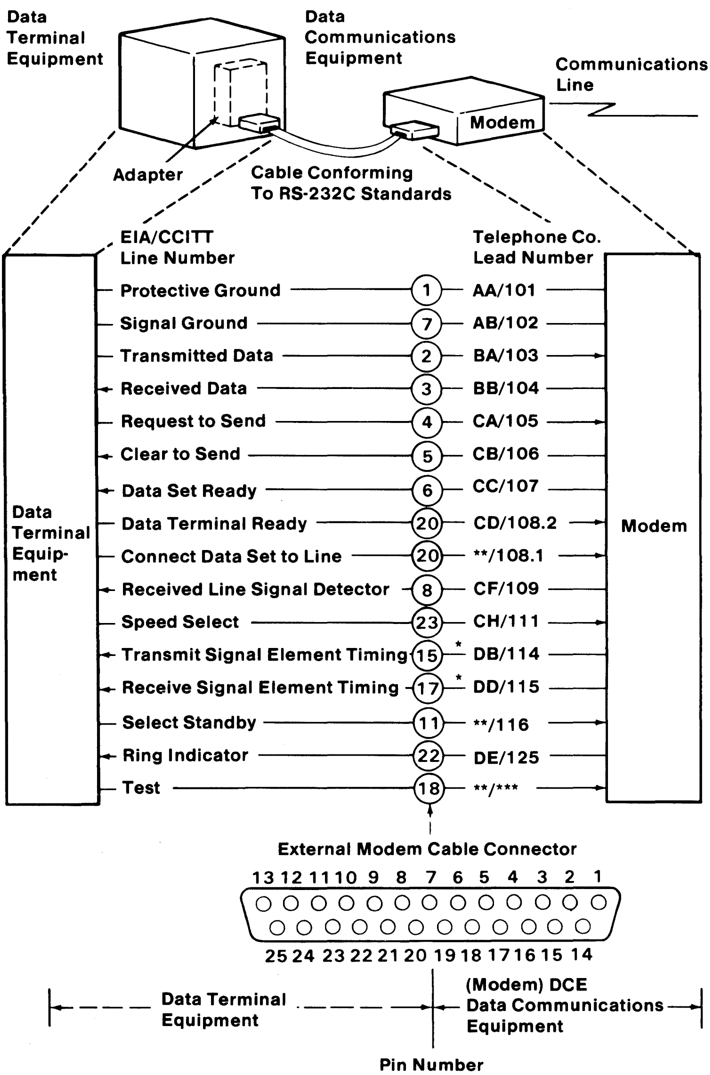
The EIA/CCITT adapter allows the data terminal equipment to be connected to the data communications equipment using EIA or CCITT standardized connections. An external modem is shown in the figure; however, other types of data communications equipment also can be connected to the data terminal equipment using EIA or CCITT standardized connections.

EIA standards are labeled RS-x (recommended standards-x), and CCITT standards are labeled V.x or X.x, where x is the number of the standard.

The EIA RS-232 interface standard defines the connector type, pin numbers, line names, and signal levels used to connect data terminal equipment to data communications equipment for the purpose of transmitting and receiving data. Since the RS-232 standard was developed, it has been revised three times. The three revised standards are RS-232A, RS-232B, and the presently used RS-232C.

The CCITT V.24 interface standard is equivalent to the RS-232C standard; therefore, the descriptions of the EIA standards also apply to the CCITT standards.

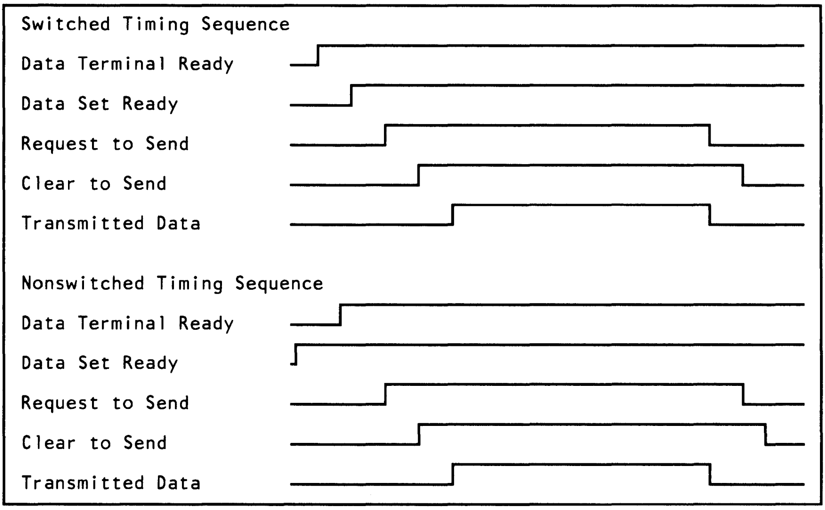
The following is an illustration of data terminal equipment connected to an external modem using connections defined by the RS-232C interface standard:



*Not used when business machine clocking is used.
**Not standardized by EIA (Electronics Industry Association).
***Not standardized by CCITT

Establishing a Communications Link

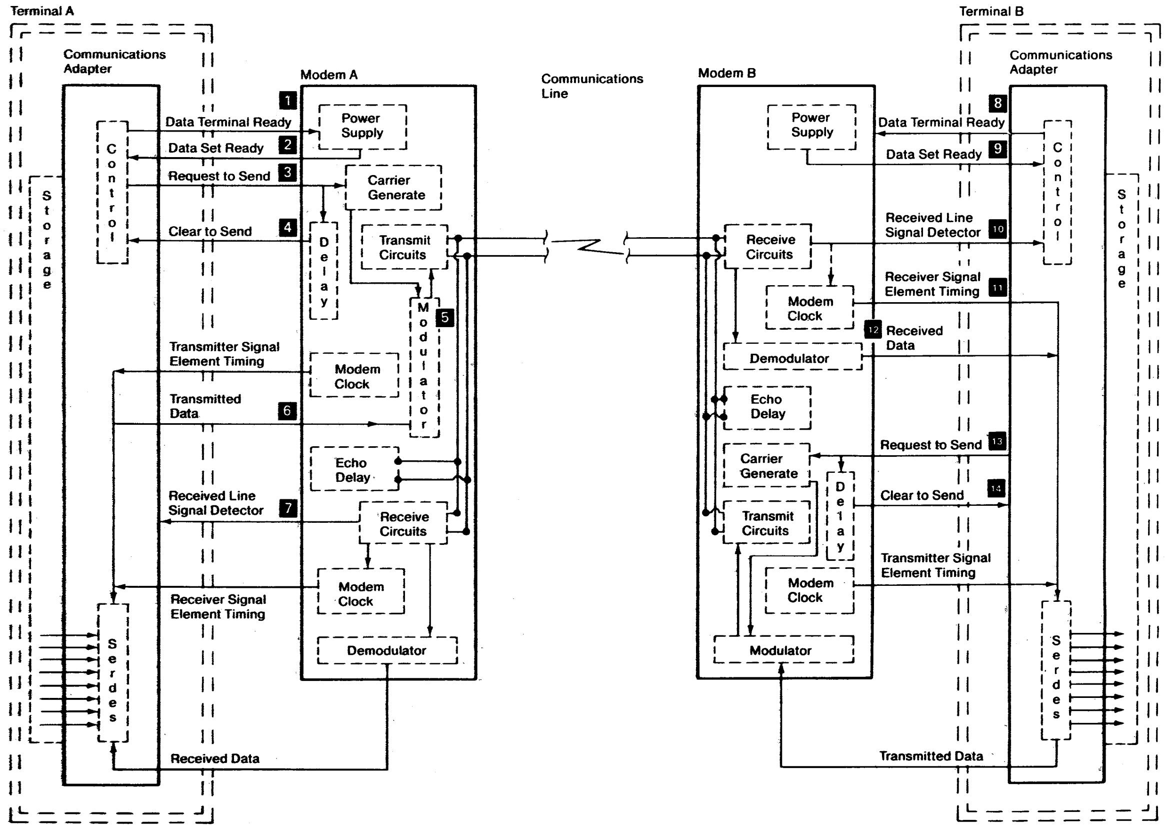
The following bar graphs represent normal timing sequences of operation during the establishment of communication for both switched (dial-up) and nonswitched (direct line) networks.



The following examples show how a link is established on a nonswitched point-to-point line, a nonswitched multipoint line, and a switched point-to-point line.

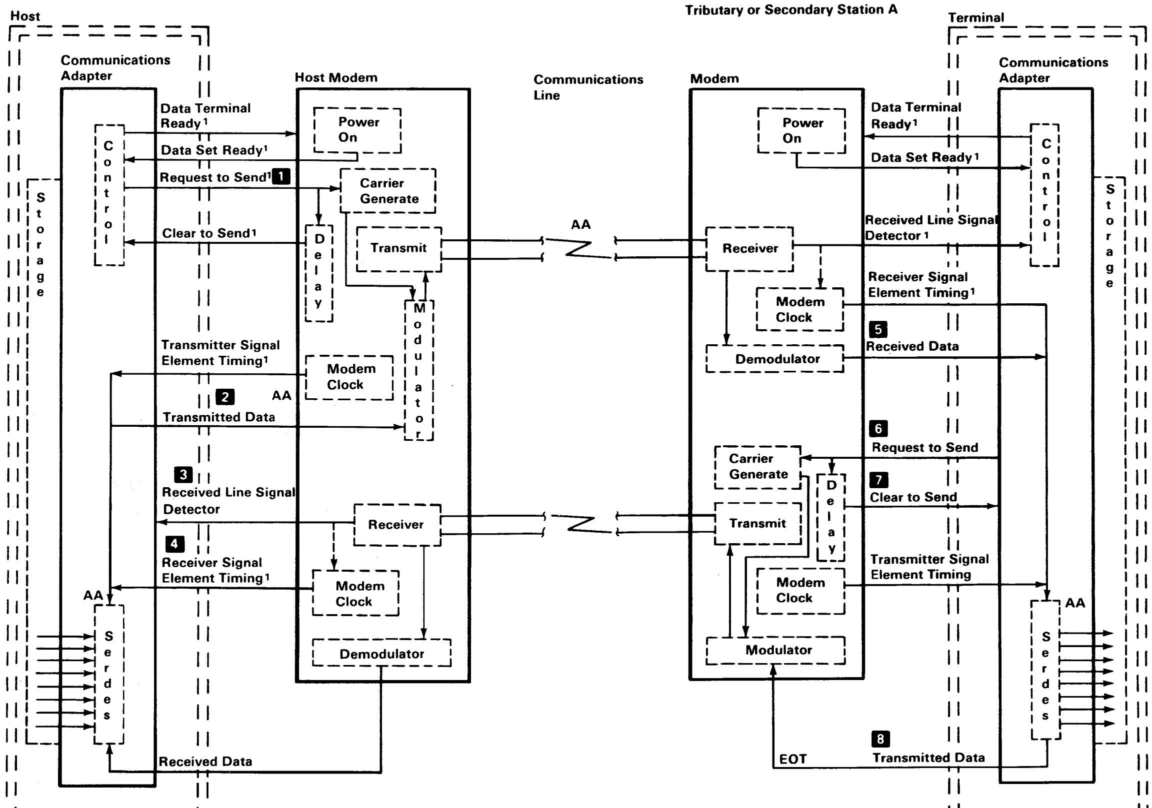
Establishing a Link on a Nonswitched Point-to-Point Line

1. The terminals at both locations activate the 'data terminal ready' lines **1** and **8**.
2. Normally the 'data set ready' lines **2** and **9** from the modems are active whenever the modems are powered on.
3. Terminal A activates the 'request to send' line **3**, which causes the modem at terminal A to generate a carrier signal.
4. Modem B detects the carrier, and activates the 'received line signal detector' line (sometimes called data carrier detect) **10**. Modem B also activates the 'receiver signal element timing' line (sometimes called receive clock) **11** to send receive clock signals to the terminal. Some modems activate the clock signals whenever the modem is powered on.
5. After a specified delay, modem A activates the 'clear to send' line **4**, which indicates to terminal A that the modem is ready to transmit data.
6. Terminal A serializes the data to be transmitted (through the serdes) and transmits the data one bit at a time (synchronized by the transmit clock) onto the 'transmitted data' line **6** to the modem.
7. The modem modulates the carrier signal with the data and transmits it to the modem B **5**.
8. Modem B demodulates the data from the carrier signal and sends it to terminal B on the 'received data' line **12**.
9. Terminal B deserializes the data (through the serdes) using the receive clock signals (on the 'receiver signal element timing' line) **11** from the modem.
10. After terminal A completes its transmission, it deactivates the 'request to send' line **3**, which causes the modem to turn off the carrier and deactivate the 'clear to send' line **4**.
11. Terminal A and modem A now become receivers and wait for a response from terminal B, indicating that all data has reached terminal B. Modem A begins an echo delay (50 to 150 milliseconds) to ensure that all echoes on the line have diminished before it begins receiving. An echo is a reflection of the transmitted signal. If the transmitting modem changed to receive too soon, it could receive a reflection (echo) of the signal it just transmitted.
12. Modem B deactivates the 'received line signal detector' line **10** and, if necessary, deactivates the receive clock signals on the 'receiver signal element timing' line **11**.
13. Terminal B now becomes the transmitter to respond to the request from terminal A. To transmit data, terminal B activates the 'request to send' line **13**, which causes modem B to transmit a carrier to modem A.
14. Modem B begins a delay that is longer than the echo delay at modem A before turning on the 'clear to send' line. The longer delay (called request-to-send to clear-to-send delay) ensures that modem A is ready to receive when terminal B begins transmitting data. After the delay, modem B activates the 'clear to send' line **14** to indicate that terminal B can begin transmitting its response.
15. After the echo delay at modem A, modem A senses the carrier from modem B (the carrier was activated in step 13 when terminal B activated the 'request to send' line) and activates the 'received line signal detector' line **7** to terminal A.
16. Modem A and terminal A are now ready to receive the response from terminal B. Remember, the response was not transmitted until after the request-to-send to clear-to-send delay at modem B (step 14).


Terminal A
Communications Adapter
Storage
Control
Serdes
Modem A
Power Supply
Carrier Generate
Transmit Circuits
Delay
Modulator
Modem Clock
Echo Delay
Receive Circuits
Modem Clock
Demodulator
1 Data Terminal Ready
2 Data Set Ready
3 Request to Send
4 Clear to Send
5
Transmitter Signal Element Timing
6 Transmitted Data
7 Received Line Signal Detector
Receiver Signal Element Timing
Received Data
Communications Line
Modem B
Power Supply
Receive Circuits
Modem Clock
Demodulator
Echo Delay
Carrier Generate
Transmit Circuits
Delay
Modem Clock
Modulator
Terminal B
Communications Adapter
Control
Storage
Serdes
8 Data Terminal Ready
9 Data Set Ready
10 Received Line Signal Detector
11 Receiver Signal Element Timing
12 Received Data
13 Request to Send
14 Clear to Send
Transmitter Signal Element Timing
Transmitted Data

Establishing a Link on a Nonswitched Multipoint Line

1. The control station serializes the address for the tributary or secondary station (AA) and sends its address to the modem on the 'transmitted data' line 2 .
2. Since the 'request to send' line and, therefore, the modem carrier, is active continuously 1 , the modem immediately modulates the carrier with the address, and, thus, the address is transmitted to all modems on the line.
3. All tributary modems, including the modem for station A, demodulate the address and send it to their terminals on the 'received data' line 5 .
4. Only station A responds to the address; the other stations ignore the address and continue monitoring their 'received data' line. To respond to the poll, station A activates its 'request to send' line 6 which causes the modem to begin transmitting a carrier signal.
5. The control station's modem receives the carrier and activates the 'received line signal detector' line 3 and the 'receiver signal element timing' line 4 (to send clock signals to the control station). Some modems activate the clock signals as soon as they are powered on.
6. After a short delay to allow the control station modem to receive the carrier, the tributary modem activates the 'clear to send' line 7 .
7. When station A detects the active 'clear to send' line, it transmits its response. (For this example, assume that station A has no data to send; therefore, it transmits an EOT 8 .)
8. After transmitting the EOT, station A deactivates the 'request to send' line 6 . This causes the modem to deactivate the carrier and the 'clear to send' line 7 .
9. When the modem at the control station (host) detects the absence of the carrier, it deactivates the 'received line signal detector' line 3 .
10. Tributary station A is now in receive mode waiting for the next poll or select transmission from the control station.



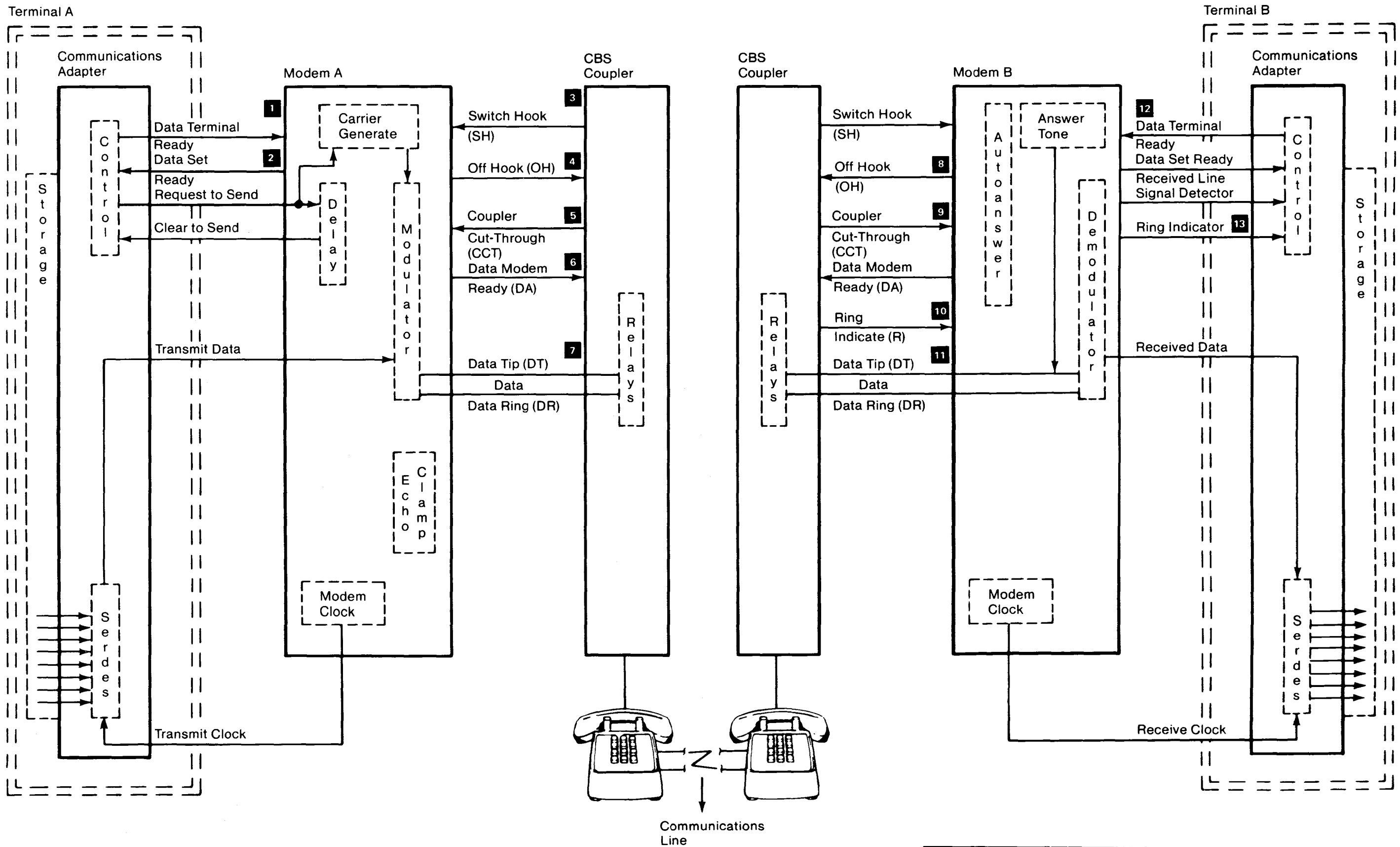
[1]These lines are active continuously.

SECTION 8

Establishing a Link on a Switched Point-to-Point Line

1. Terminal A is in communications mode; therefore, the 'data terminal ready' line [1] is active. Terminal B is in communication mode waiting for a call from terminal A.
2. When the terminal A operator lifts the telephone handset, the 'switch hook' line from the coupler is activated [3].
3. Modem A detects the 'switch hook' line and activates the 'off hook' line [4], which causes the coupler to connect the telephone set to the line and activate the 'coupler cut-through' line [5] to the modem.
4. Modem A activates the 'data modem ready' line [6] to the coupler (the 'data modem ready' line is on continuously in some modems).
5. The terminal A operator sets the exclusion key or talk/data switch to the talk position to connect the handset to the communications line. The operator then dials the terminal B number.
6. When the telephone at terminal B rings, the coupler activates the 'ring indicate' line to modem B [10]. Modem B indicates that the 'ring indicate' line was activated by activating the 'ring indicator' line [13] to terminal B.
7. Terminal B activates the 'data terminal ready' line to modem B [12], which activates the autoanswer circuits in modem B. (The 'data terminal ready' line might already be active in some terminals.)
8. The autoanswer circuits in modem B activate the 'off hook' line to the coupler [8].
9. The coupler connects modem B to the communications line through the 'data tip' and 'data ring' lines [11] and activates the 'coupler cut-through' line [9] to the modem. Modem B then transmits an answer tone to terminal A.
10. The terminal A operator hears the tone and sets the exclusion key or talk/data switch to the data position (or performs an equivalent operation) to connect modem A to the communications line through the 'data tip' and 'data ring' lines [7].
11. The coupler at terminal A deactivates the 'switch hook' line [3]. This causes modem A to activate the 'data set ready' line [2] indicating to terminal A that the modem is connected to the communications line.

The sequence of the remaining steps to establish the data link is the same as the sequence required on a nonswitched point-to-point line. When the terminals have completed their transmission, they both deactivate the 'data terminal ready' line to disconnect the modems from the line.


Terminal A
Communications Adapter
Storage
Control
Serdes
Data Terminal Ready
Data Set Ready
Request to Send
Clear to Send
Transmit Data
Transmit Clock
Modem A
Carrier Generate
Delay
Modulator
Echo Clamp
Modem Clock
Switch Hook (SH)
Off Hook (OH)
Coupler Cut-Through (CCT)
Data Modem Ready (DA)
Data Tip (DT)
Data
Data Ring (DR)
CBS Coupler
Relays
Communications Line
CBS Coupler
Relays
Switch Hook (SH)
Off Hook (OH)
Coupler Cut-Through (CCT)
Data Modem Ready (DA)
Ring Indicate (R)
Data Tip (DT)
Data
Data Ring (DR)
Modem B
Autoanswer
Answer Tone
Demodulator
Modem Clock
Terminal B
Communications Adapter
Data Terminal Ready
Data Set Ready
Received Line Signal Detector
Ring Indicator
Received Data
Receive Clock
Control
Storage
Serdes
1
2
3
4
5
6
7
8
9
10
11
12
13

Notes:

SECTION 9. IBM PERSONAL COMPUTER COMPATIBILITY

Contents

SECTION 9

Notes:

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 AT that are not supported by all of the IBM Personal Computer family.

System Board

The IBM PERSONAL COMPUTER AT system board uses an Intel 80286 (-6 or -8) 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 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."

Some adapters use a 36-pin connector in addition to the 62-pin connector. Adapters designed to use the 36-pin connectors are not compatible with all members of the IBM Personal Computer family. Refer to the "System to Adapter Compatibility Chart" in the *Technical Reference Options and Adapters* manual, Volume 1, to identify the adapters supported by each system. The IBM PERSONAL COMPUTER AT does not support an expansion unit.

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 is activated during DMA cycles.
- The -IOW signal is not active during refresh 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	---
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 AT.

Head Geometry: See "Diskette Drive Compatibility" on page 9-5

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 9-8).

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.

SECTION 9

- 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 AT. 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

SECTION 9

that hardware designed to use IRQ 2 will operate in the IBM Personal Computer AT.

- 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.

- Control of the Game Control Adapter, port hex 201

  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 9-5 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 seperated by JMP SHORT $+2.

Multitasking Provisions

The IBM Personal Computer AT 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
```

SECTION 9

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 AT 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	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 occurrs.

SECTION 9

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
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 definitions developed by the American National Standards Institute (ANSI) and the International Organization for Standardization (ISO). This material is reproduced from the *American National Dictionary for Information Processing*, copyright 1977 by the Computer and Business Equipment Manufacturers Association, copies of which may be purchased from the American National Standards Institute, 1430 Broadway, New York, New York 10018.

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.

GLOSSARY

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.

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.

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.

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

INDEX

E

F

J

K

L

S

T

U

V

W

X

Y

Z

Numerics

The Personal Computer
Hardware Reference Library

Reader's Comment Form

Technical Reference 6183355

Your comments assist us in improving the usefulness of our publication; they are an important part of the input used for revisions.

IBM may use and distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Please do not use this form for technical questions regarding the IBM Personal Computer or programs for the IBM Personal Computer, or for requests for additional publications; this only delays the response. Instead, direct your inquiries or request to your authorized IBM Personal Computer dealer.

Comments:

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM PERSONAL COMPUTER
READER COMMENT DEPARTMENT
P.O. BOX 1328-C
BOCA RATON, FLORIDA 33429-9960

Fold here

Tape Please do not staple Tape

IBM

The Personal Computer
Hardware Reference Library

Reader's Comment Form

Technical Reference 6183355

Your comments assist us in improving the usefulness of our publication; they are an important part of the input used for revisions.

IBM may use and distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Please do not use this form for technical questions regarding the IBM Personal Computer or programs for the IBM Personal Computer, or for requests for additional publications; this only delays the response. Instead, direct your inquiries or request to your authorized IBM Personal Computer dealer.

Comments:

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM PERSONAL COMPUTER
READER COMMENT DEPARTMENT
P.O. BOX 1328-C
BOCA RATON, FLORIDA 33429-9960

Fold here

Please do not staple

Tape

IBM

The Personal Computer
Hardware Reference Library

Reader's Comment Form

Technical Reference 6183355

Your comments assist us in improving the usefulness of our publication; they are an important part of the input used for revisions.

IBM may use and distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Please do not use this form for technical questions regarding the IBM Personal Computer or programs for the IBM Personal Computer, or for requests for additional publications; this only delays the response. Instead, direct your inquiries or request to your authorized IBM Personal Computer dealer.

Comments:

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM PERSONAL COMPUTER
READER COMMENT DEPARTMENT
P.O. BOX 1328-C
BOCA RATON, FLORIDA 33429-9960

Fold here

Tape Please do not staple Tape

IBM

The Personal Computer
Hardware Reference Library

Reader's Comment Form

Technical Reference 6183355

Your comments assist us in improving the usefulness of our publication; they are an important part of the input used for revisions.

IBM may use and distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Please do not use this form for technical questions regarding the IBM Personal Computer or programs for the IBM Personal Computer, or for requests for additional publications; this only delays the response. Instead, direct your inquiries or request to your authorized IBM Personal Computer dealer.

Comments:

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS **PERMIT NO. 40** **ARMONK, NEW YORK**

POSTAGE WILL BE PAID BY ADDRESSEE

IBM PERSONAL COMPUTER
READER COMMENT DEPARTMENT
P.O. BOX 1328-C
BOCA RATON, FLORIDA 33429-9960

Fold here

Please do not staple

Tape

International Business Machines Corporation

P.O. Box 1328-W
Boca Raton, Florida 33432

6183355

Printed in the United States of America