



*Personal Computer  
Hardware Reference  
Library*

---

# Technical Reference

**First Edition (August 1986)**

**The following paragraph does not apply to the United Kingdom or any country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.**

This publication could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time.

It is possible that this publication may contain reference to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such IBM products, programming, or services in your country.

Requests for copies of this publication and for technical information about IBM Personal Computer products should be made to your authorized IBM Personal Computer dealer or your IBM Marketing Representative.

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

**FEDERAL COMMUNICATIONS COMMISSION RADIO  
FREQUENCY INTERFERENCE STATEMENT**

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

**CAUTION**

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

**Notes:**

○

○

○

# Preface

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

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

This manual consists of eight sections:

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

A glossary, bibliography, and index are included.

### **Prerequisite Publications**

*Guide to Operations* for the IBM Personal Computer XT Model 286

### **Suggested Reading**

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

### **Additional Information**

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

# Contents

<b>SECTION 1. SYSTEM BOARD</b> .....	<b>1-1</b>
Memory .....	1-4
Microprocessor .....	1-4
System Performance .....	1-7
System Memory Mapping .....	1-8
Direct Memory Access .....	1-9
System Interrupts .....	1-12
Hardware Interrupt Listing .....	1-13
Interrupt Sharing .....	1-14
System Timers .....	1-22
System Clock .....	1-23
ROM Subsystem .....	1-23
RAM Subsystem .....	1-24
I/O Channel .....	1-24
Connectors .....	1-25
I/O Channel Signal Description .....	1-31
I/O Addresses .....	1-38
Other Circuits .....	1-43
Speaker .....	1-43
128K RAM Jumper (J10) .....	1-43
Display Switch .....	1-44
Keyboard Controller .....	1-44
Real-Time Clock CMOS RAM Information .....	1-59
Specifications .....	1-72
System Unit .....	1-72
Connectors .....	1-74
Logic Diagrams .....	1-77
<b>SECTION 2. COPROCESSOR</b> .....	<b>2-1</b>
Description .....	2-3
Programming Interface .....	2-3
Hardware Interface .....	2-4
<b>SECTION 3. POWER SUPPLY</b> .....	<b>3-1</b>
Inputs .....	3-3
Outputs .....	3-3
DC Output Protection .....	3-4

Output Voltage Sequencing .....	3-4
No-Load Operation .....	3-4
Power-Good Signal .....	3-4
Connectors .....	3-6
<b>SECTION 4. KEYBOARD .....</b>	<b>4-1</b>
Description .....	4-3
Power-On Routine .....	4-5
Commands from the System .....	4-6
Commands to the System .....	4-13
Keyboard Scan Codes .....	4-15
Clock and Data Signals .....	4-27
Keyboard Encoding and Usage .....	4-30
Keyboard Layouts .....	4-40
Specifications .....	4-47
Logic Diagram .....	4-48
<b>SECTION 5. SYSTEM BIOS .....</b>	<b>5-1</b>
System BIOS Usage .....	5-3
Quick Reference .....	5-14
<b>SECTION 6. INSTRUCTION SET .....</b>	<b>6-1</b>
80286 Instruction Set .....	6-3
Data Transfer .....	6-3
Arithmetic .....	6-6
Logic .....	6-9
String Manipulation .....	6-11
Control Transfer .....	6-13
Processor Control .....	6-17
Protection Control .....	6-18
80287 Coprocessor Instruction Set .....	6-22
Data Transfer .....	6-22
Comparison .....	6-23
Constants .....	6-24
Arithmetic .....	6-25
Transcendental .....	6-26
<b>SECTION 7. CHARACTERS, KEYSTROKES, AND</b>	
<b>COLORS .....</b>	<b>7-1</b>
Character Codes .....	7-3
Quick Reference .....	7-14



**SECTION 8. IBM PERSONAL COMPUTER**

<b>COMPATIBILITY</b> .....	<b>8-1</b>
Hardware Considerations .....	8-3
System Board .....	8-3
Fixed Disk Drive .....	8-5
Diskette Drive Compatibility .....	8-5
Copy Protection .....	8-5
Application Guidelines .....	8-7
High-Level Language Considerations .....	8-7
Assembler Language Programming Considerations ..	8-8
Multitasking Provisions .....	8-15
Machine-Sensitive Code .....	8-19
<b>Glossary</b> .....	<b>X-1</b>
<b>Bibliography</b> .....	<b>X-37</b>
<b>Index</b> .....	<b>X-39</b>

## Notes:

# INDEX TAB LISTING

Section 1: System Board .....

SECTION 1

Section 2: Coprocessor .....

SECTION 2

Section 3: Power Supply .....

SECTION 3

Section 4: Keyboard .....

SECTION 4

Section 5: System BIOS .....

SECTION 5

Section 6: Instruction Set .....

SECTION 6

**Notes:**

—

—

—

Section 7: Characters, Keystrokes, and Colors .....

SECTION 7

Section 8: Compatibility .....

SECTION 8

Glossary .....

GLOSSARY

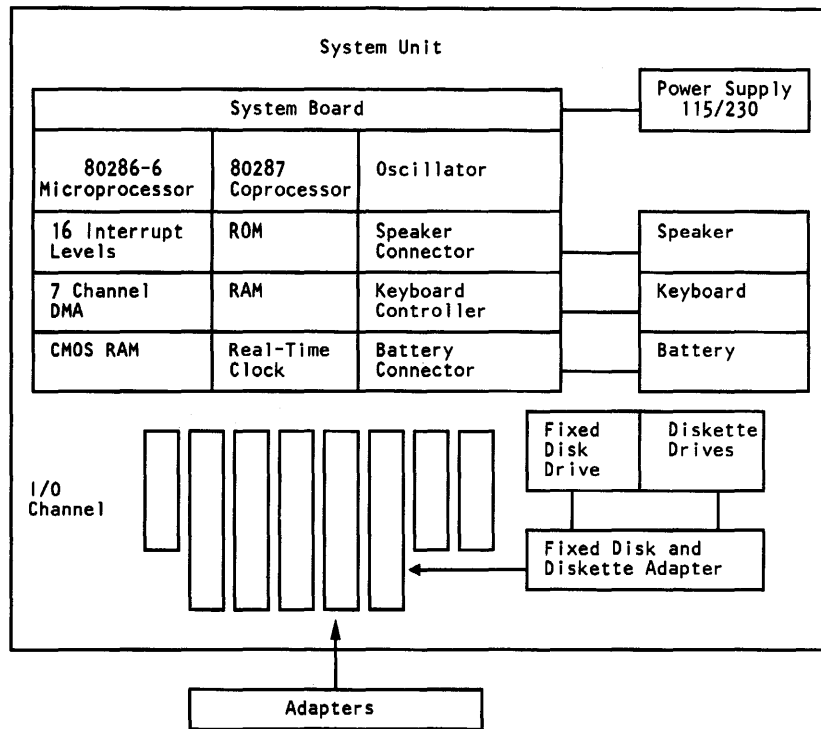
Bibliography .....

BIBLIOGRAPHY

Index .....

INDEX

# System Block Diagram



# SECTION 1. SYSTEM BOARD

Memory .....	1-4
Microprocessor .....	1-4
Real Address Mode .....	1-4
Protected (Virtual Address) Mode .....	1-5
System Performance .....	1-7
System Memory Mapping .....	1-8
Direct Memory Access .....	1-9
System Interrupts .....	1-12
Hardware Interrupt Listing .....	1-13
Interrupt Sharing .....	1-14
Design Overview .....	1-14
Program Support .....	1-15
Precautions .....	1-17
Examples .....	1-18
System Timers .....	1-22
System Clock .....	1-23
ROM Subsystem .....	1-23
RAM Subsystem .....	1-24
I/O Channel .....	1-24
Connectors .....	1-25
I/O Channel Signal Description .....	1-31
I/O Addresses .....	1-38
NMI Controls .....	1-39
I/O Port (Read/Write) .....	1-40
Diagnostic-Checkpoint Port .....	1-42
Coprocessor Controls .....	1-42
Other Circuits .....	1-43
Speaker .....	1-43
128K RAM Jumper (J10) .....	1-43
Display Switch .....	1-44
Keyboard Controller .....	1-44
Keyboard Controller Initialization .....	1-45
Receiving Data from the Keyboard .....	1-45
Scan Code Translation .....	1-46
Sending Data to the Keyboard .....	1-53
Keyboard Controller System Interface .....	1-53
Status Register .....	1-54

Status-Register Bit Definition .....	1-54
Output Buffer .....	1-56
Input Buffer .....	1-56
Commands (I/O Address Hex 64) .....	1-56
I/O Ports .....	1-58
Real-Time Clock CMOS RAM Information .....	1-59
Real-Time Clock Information .....	1-60
CMOS RAM Configuration Information .....	1-63
I/O Operations .....	1-70
Specifications .....	1-72
System Unit .....	1-72
Connectors .....	1-74
Logic Diagrams .....	1-77

## 1-2 System Board



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

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

## Memory

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

## Microprocessor

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

### Real Address Mode

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

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

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

---

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

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

## Protected (Virtual Address) Mode

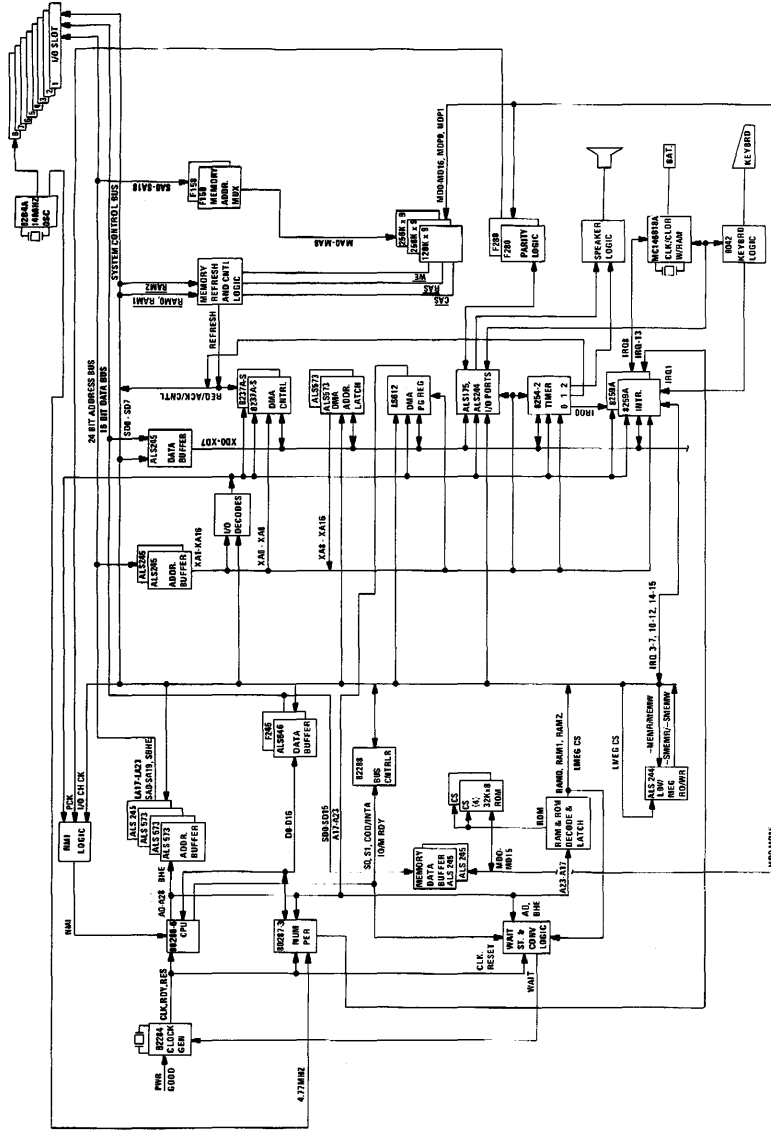
The protected mode offers extended physical and virtual memory address space, memory protection mechanisms, and new operations to support operating systems and virtual memory.

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

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

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

The following is a block diagram of the system board.



1-6 System Board

## System Performance

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

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

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

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

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

# System Memory Mapping

The following shows the mapping of the system memory.

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

## System Memory

## Direct Memory Access

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

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

### DMA Channels

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

The following figures show address generation for the DMA channels.

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

### Address Generation for DMA Channels 0 through 3

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

DMA controller 1 command code addresses follow.

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

#### DMA Controller 1 (Channels 0-3)

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

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

#### Address Generation for DMA Channels 5 through 7

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



The following figure shows the addresses for the page register.

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

**Page Register Addresses**

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

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

DMA controller 2 command code addresses follow.

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

**DMA Controller 2 (DMA Channels 4-7)**

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

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

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

## System Interrupts

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

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

## Hardware Interrupt Listing

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

Level	Function
Microprocessor NMI	Parity or I/O Channel Check
Interrupt Controllers CTRL 1    CTRL 2	
IRQ 0 IRQ 1 IRQ 2 ←	Timer Output 0 Keyboard (Output Buffer Full) Interrupt from CTRL 2
<div style="display: flex; align-items: center;"> <div style="border-left: 1px solid black; border-right: 1px solid black; border-bottom: 1px solid black; padding: 2px 5px; margin-right: 5px;">           IRQ 8 IRQ 9  IRQ 10 IRQ 11 IRQ 12 IRQ 13 IRQ 14 IRQ 15         </div> <div style="border-left: 1px solid black; border-right: 1px solid black; border-bottom: 1px solid black; padding: 2px 5px;">           IRQ 8 IRQ 9  IRQ 10 IRQ 11 IRQ 12 IRQ 13 IRQ 14 IRQ 15         </div> </div>	Realtime Clock Interrupt Software Redirected to INT 0AH PC Network * PC Network(Alt.) * Reserved Reserved Reserved Coprocessor Fixed Disk Controller Reserved
IRQ 3	Serial Port 2 BSC BSC (Alt.) PC Network * PC Network (Alt.) * SDLC
IRQ 4	Serial Port 1 BSC BSC (Alt.) SDLC
IRQ 5	Parallel Port 2
IRQ 6	Diskette Controller Fixed Disk and Diskette Drive
IRQ 7	Parallel Port 1 Data Acquisition and Control ** GPIB *** Voice Communications Adapter ****
* The PC Network is jumper selectable. ** The Data Acquisition Adapter can be set to interrupts 3 through 7. The default interrupt is 7. *** The GPIB Adapter can be set to interrupts 2 through 7. **** The Voice Communications Adapter can be set to interrupts 2, 3, 4, or 7 (Interrupt level 7 is recommended).	

Hardware Interrupt Listing

---

## **Interrupt Sharing**

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

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

### **Design Overview**

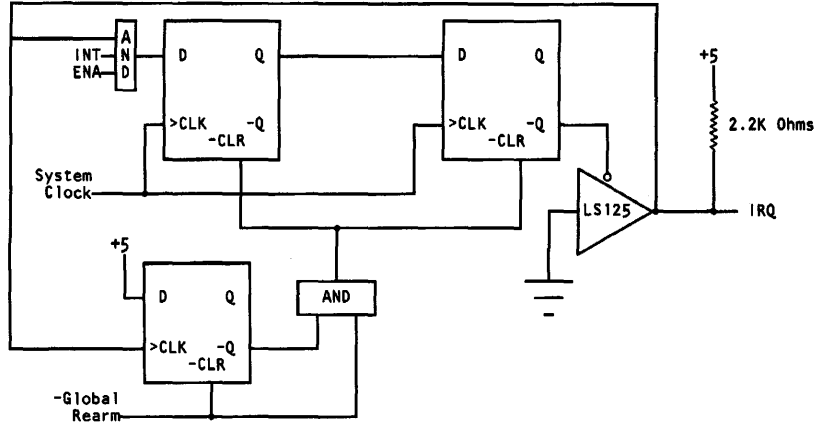
Most interrupt-supporting adapters hold the 'interrupt request' line (IRQ) at a low level and then drive the line high to cause an interrupt. In contrast, the shared-interrupt hardware design allows IRQ to float high through pull-up resistors on each adapter. Each adapter on the line may cause an interrupt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

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

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

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

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



**Shared Interrupt Logic Diagram**

## Program Support

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

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

### Linking to a Chain

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

## **Sharing the Interrupt Level**

When the new task's handler gains control as a result of an interrupt, the handler reads the contents of the adapter's interrupt status register to determine if its adapter caused the interrupt. If it did, the handler services the interrupt, disables the interrupts (CLI), issues a non-specific End of Interrupt (EOI), and then, to rearm the interrupt hardware, writes to address 02FX, where X corresponds to interrupt levels 3 through 7, and 9 (IRQ9 is 02F2). A write to address 06FX, 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 Rerm.
  - 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
PAST:   RES_BYTES DB    DUP 7 (0) ; Future expansion
        ...                               ; 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
ISB        EQU    xx            ; control/status register
REARM      EQU    2F7H          ; interrupt bit in your card's interrupt
SPC_EOI    EQU    67H           ; control status register
EOI        EQU    20H           ; Global Rearm location for interrupt
OCR        EQU    20H           ; level 7
IMR        EQU    21H           ; Specific EOI for 8259's interrupt
                                     ; level 7
                                     ; Non-specific EOI
                                     ; Location of 8259 operational control
                                     ; register
                                     ; 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            ; Issue non-specific EOI to 8259
           OUT OCR,AL            ; Rearm the cards
           MOV DX,REARM          ; Rearm the cards
           OUT DX,AL
           ...                   ; Restore registers
           POP
           IRET

RESET:     ...                   ; Disable your card
           RET                   ; Return FAR to operating system
ENTRY     ENDP
MYCSEG    ENDS
END       ENTRY

```



## Unlinking Code Example

```

        PUSH    DS
        PUSH    ES
        CLI                    ; Disable interrupts
        MOV     AX,350FH        ; DOS get interrupt vector
        INT    21H             ; ES:BX points to first of chain
        MOV     CX,ES          ; Pickup segment part of interrupt vector
; Are we the first handler in the chain?
        MOV     AX,CS          ; Get code seg into comparable register
        CMP     BX,OFFSET ENTRY ; Interrupt vector in low memory
                                ; pointing to your handler's offset?
        JNE     UNCHAIN_A      ; No, branch
        CMP     AX,CX          ; Vector pointing to your
                                ; handler's segment?
        JNE     UNCHAIN_A      ; No, branch
; Set interrupt vector in low memory to point to the handler
; pointed to by your pointer
        PUSH    DS
        MOV     DX,WORD PTR CS:FPTR
        MOV     DS,WORD PTR CS:FPTR[2]
        MOV     AX,250FH        ; DOS set interrupt vector
        INT    21H
        POP     DS
        JMP     UNCHAIN_X
UNCHAIN_A: ; BX = FPTR offset, ES = FPTR segment, CX = CS
        CMP     ES:[BX+6],4B42H ; Is handler using the appropriate
                                ; conventions (is SIGNATURE present in
                                ; the interrupt chaining structure)?
        JNE     exception      ; No, invoke error exception handler
        LDS     SI,ES:[BX+2]    ; Get FPTR's segment and offset
        CMP     SI,OFFSET ENTRY ; Is this forward pointer pointing to
                                ; your handler's offset?
        JNE     UNCHAIN_B      ; No, branch
        MOV     CX,DS          ; Move to compare
        CMP     AX,CX          ; Is this forward pointer pointing to
                                ; your handler's segment?
        JNE     UNCHAIN_B      ; No, branch
; Located your handler in the chain
        MOV     AX,WORD PTR CS:FPTR ; Get your FPTR's offset
        MOV     ES:[BX+2],AX     ; Replace offset of FPTR of handler
                                ; that points to you
        MOV     AX,WORD PTR CS:FPTR[2] ; Get your FPTR's segment
        MOV     ES:[BX+4],AX     ; Replace segment of FPTR of handler
                                ; that points to you
        MOV     AL,CS:FLAGS      ; Get your flags
        AND     AL,FIRST        ; Isolate FIRST flag
        OR     ES:[BX + 6],AL    ; Set your first flag into prior routine
        JMP     UNCHAIN_X
UNCHAIN_B: MOV     BX,SI        ; Move new offset to BX
        PUSH   DS
        PUSH   ES
        JMP    UNCHAIN_A      ; Examine next handler in chain
UNCHAIN_X: STI                    ; Enable interrupts
        POP    ES
        POP    DS

```

## System Timers

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

**Channel 0            System Timer**

GATE 0            Tied on  
CLK IN 0          1.193182 MHz OSC  
CLK OUT 0        8259A IRQ 0

**Channel 1            Refresh Request Generator**

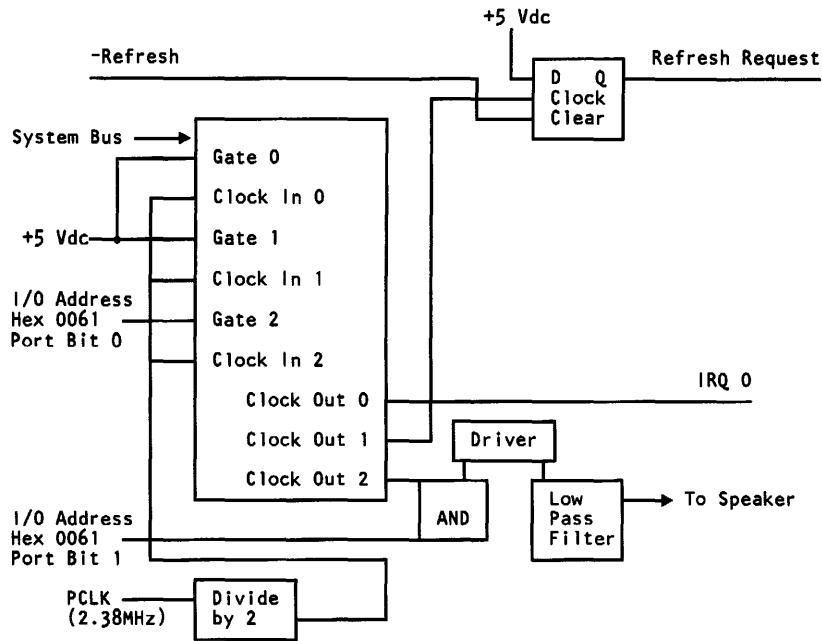
GATE 1            Tied on  
CLK IN 1          1.193182 MHz OSC  
CLK OUT 1        Request refresh cycle

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

**Channel 2            Tone Generation for Speaker**

GATE 2            Controlled by bit 0 of port hex 61, PPI bit  
CLK IN 2          1.193182 MHz OSC  
CLK OUT 2        Used to drive the speaker

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



**System-Timer Block Diagram**

## System Clock

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

## ROM Subsystem

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

## RAM Subsystem

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

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

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

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

## I/O Channel

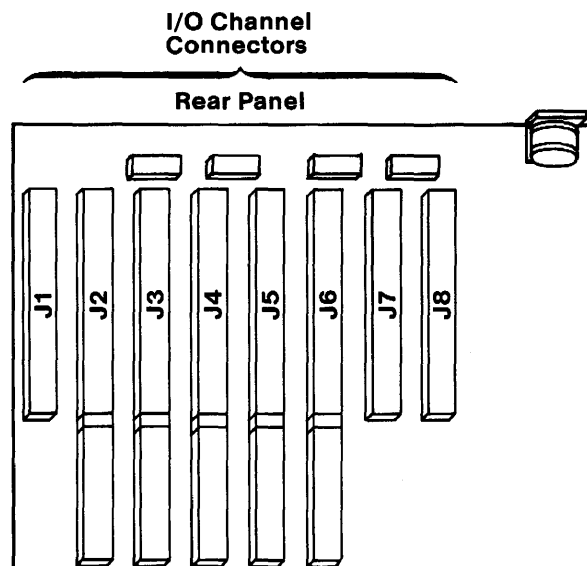
The I/O channel supports:

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

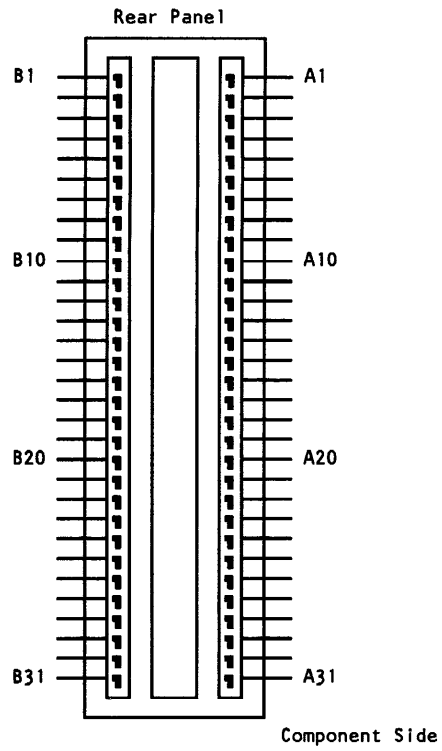
## Connectors

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

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



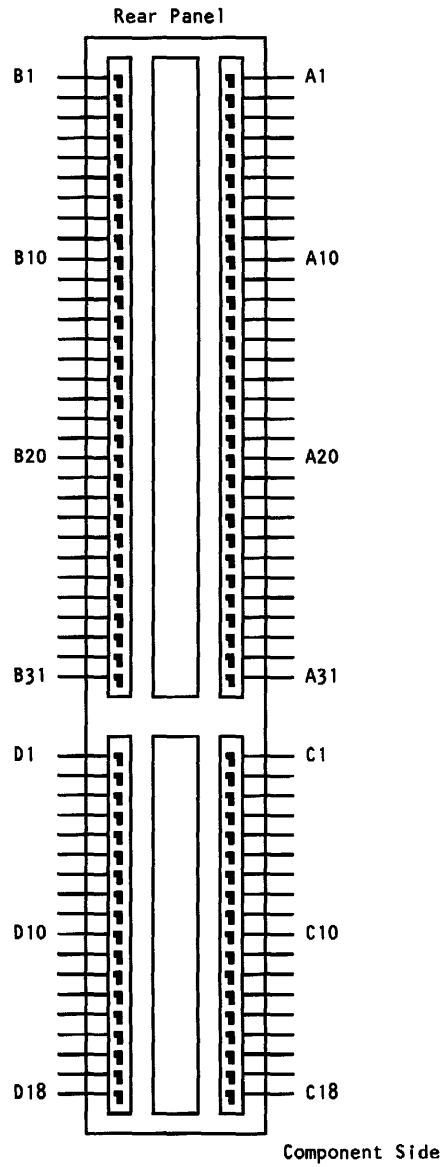
The following figure shows the pin numbering for the 62-pin I/O channel connectors J1, J7 and J8.



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



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



I/O Channel Pin Numbering (J2-J6)

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

I/O Pin	Signal Name	I/O
A1	-I/O CH CK	I
A2	SD7	I/O
A3	SD6	I/O
A4	SD5	I/O
A5	SD4	I/O
A6	SD3	I/O
A7	SD2	I/O
A8	SD1	I/O
A9	SD0	I/O
A10	+I/O CH RDY	I
A11	AEN	0
A12	SA19	I/O
A13	SA18	I/O
A14	SA17	I/O
A15	SA16	I/O
A16	SA15	I/O
A17	SA14	I/O
A18	SA13	I/O
A19	SA12	I/O
A20	SA11	I/O
A21	SA10	I/O
A22	SA9	I/O
A23	SA8	I/O
A24	SA7	I/O
A25	SA6	I/O
A26	SA5	I/O
A27	SA4	I/O
A28	SA3	I/O
A29	SA2	I/O
A30	SA1	I/O
A31	SA0	I/O

**I/O Channel (A-Side)**

I/O Pin	Signal Name	I/O
B1	GND	Ground
B2	RESET DRV	0
B3	+5 Vdc	Power
B4	IRQ 9	I
B5	-5 Vdc	Power
B6	DRQ2	I
B7	-12 Vdc	Power
B8	OVS	I
B9	+12 Vdc	Power
B10	GND	Ground
B11	-SMEMW	0
B12	-SMEMR	0
B13	-IOW	I/O
B14	-IOR	I/O
B15	-DACK3	0
B16	DRQ3	I
B17	-DACK1	0
B18	DRQ1	I
B19	-REFRESH	I/O
B20	CLK	0
B21	IRQ7	I
B22	IRQ6	I
B23	IRQ5	I
B24	IRQ4	I
B25	IRQ3	I
B26	-DACK2	0
B27	T/C	0
B28	BALE	0
B29	+5Vdc	Power
B30	14.318MHz OSC	0
B31	GND	Ground

I/O Channel (B-Side)

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

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

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

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

## I/O Channel Signal Description

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

### SA0 through SA19 (I/O)

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

### LA17 through LA23 (I/O)

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

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

### CLK (O)

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

### **RESET DRV (O)**

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

### **SD0 through SD15 (I/O)**

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

### **BALE (O) (buffered)**

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

### **-I/O CH CK (I)**

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

### **I/O CH RDY (I)**

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

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

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

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

### **-IOR (I/O)**

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

### **-IOW (I/O)**

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

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

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

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

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

### **DRQ0-DRQ3 and DRQ5-DRQ7 (I)**

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



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

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

**AEN (O)**

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

**-REFRESH (I/O)**

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

**T/C (O)**

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

**-SBHE (I/O)**

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

### **-MASTER (I)**

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

### **-MEM CS16 (I)**

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

### **-I/O CS16 (I)**

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

### **14.318MHz OSC (0)**

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

## **OWS (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, OWS is derived from an address decode gated with a Read or Write command. OWS is active low and should be driven with an open collector or tri-state driver capable of sinking 20 mA.

## I/O Addresses

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

Hex Range	Device
000-01F	DMA Controller 1, 8237A-5
020-03F	Interrupt Controller 1, 8259A, Master
040-05F	Timer, 8254-2
060	8042 (Keyboard)
061	System Board I/O port
064	8042 (Keyboard)
070-07F	Real-Time Clock, NMI (Non-maskable Interrupt) Mask
080-09F	DMA Page Register , 74LS612
0A0-0BF	Interrupt Controller 2, 8259A
0C0-0DF	DMA Controller 2, 8237A-5
0F0	Clear Math Coprocessor Busy
0F1	Reset Math Coprocessor
0F8-0FF	Math Coprocessor
1F0-1F8	Fixed Disk
20C-20D	Reserved
21F	Voice Communications Adapter
278-27F	Parallel Printer Port 2
280-2DF	Alternate Enhanced Graphics Adapter
2E1	GPiB (Adapter 0)
2E2 & 2E3	Data Acquisition (Adapter 0)
2F8-2FF	Serial Port 2
300-31F	Prototype Adapter
360-363	PC Network (low address)
364-367	Reserved
368-36B	PC Network (high address)
36C-36F	Reserved
378-37F	Parallel Printer Port 1
380-38F	SDLC, Bisynchronous 2
3A0-3AF	Bisynchronous 1
3B0-3BF	Monochrome Display and Printer Adapter
3C0-3CF	Enhanced Graphics Adapter
3D0-3DF	Color/Graphics Monitor Adapter
3F0-3F7	Diskette Controller
3F8-3FF	Serial Port 1
6E2 & 6E3	Data Acquisition (Adapter 1)
AE2 & AE3	Data Acquisition (Adapter 2)
EE2 & EE3	Data Acquisition (Adapter 3)
22E1	GPiB (Adapter 1)
42E1	GPiB (Adapter 2)
62E1	GPiB (Adapter 3)
82E1	GPiB (Adapter 4)
A2E1	GPiB (Adapter 5)
C2E1	GPiB (Adapter 6)
E2E1	GPiB (Adapter 7)

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

### I/O Address Map

## NMI Controls

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

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

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

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

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

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

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

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

## **I/O Port (Read/Write)**

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

### **Input (Read)**

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

- Bit 7** +RAM Parity Check—System board memory parity check.
- 0** No memory parity check error has occurred.
  - 1** A memory parity check error has occurred.
- A non-maskable interrupt (NMI) will occur if NMI is enabled. The error bit can be reset by toggling output port hex 061, bit 2, to a 1 and then back to a 0.
- Bit 6** +I/O Channel Check—Error on an I/O channel adapter (memory parity error or adapter errors).
- 0** No I/O channel error has occurred.
  - 1** An I/O channel error has occurred.
- A non-maskable interrupt (NMI) will occur if NMI is enabled. The error bit can be reset by toggling output port hex 061, bit 3, to a 1 and then back to a 0.
- Bit 5** Timer 2 Channel Out—Reflects the level of the Timer 2 output.
- Bit 4** Refresh Detect—Toggles every 15 microseconds, indicating normal Refresh activity
- Bits 3 to 0** Read the status of bits 3, 2, 1, and 0, respectively, written to output port hex 061.

### Output (Write)

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

**Bits 7 to 4** Not used

**Bit 3** -Enable I/O Channel Check

- 0** Enables I/O Channel Check errors.
- 1** Disables I/O Channel Check errors.

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

**Bit 2** -Enable System Board RAM Parity Check

- 0** Enables system board RAM parity check.
- 1** Disables system board RAM parity check.

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

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

**Bit 0** +Timer 2 Gate Speaker

- 0** Disables 8254 Timer 2 clock input (1.19 MHz).
- 1** Enables 8254 Timer 2 clock input (1.19 MHz).

## **Diagnostic-Checkpoint Port**

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

## **Coprocessor Controls**

The following is a description of the Math Coprocessor controls.

- 0F0** An 8-bit Out command to port F0 will clear the latched Math Coprocessor '-busy' signal. The '-busy' signal will be latched if the coprocessor asserts its '-error' signal while it is busy. The data output should be zero.
- 0F1** An 8-bit Out command to port F1 will reset the Math Coprocessor. The data output should be zero.



## Other Circuits

### Speaker

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

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

### 128K RAM Jumper (J10)

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

Pin	Assignments
1	No Connection
2	Ground
3	RAM Select

#### RAM Jumper Connector (J10)

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

**Note:** The normal mode is the enabled mode.

## Display Switch

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

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

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

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

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

## Keyboard Controller

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

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

## Keyboard Controller Initialization

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

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

## Receiving Data from the Keyboard

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

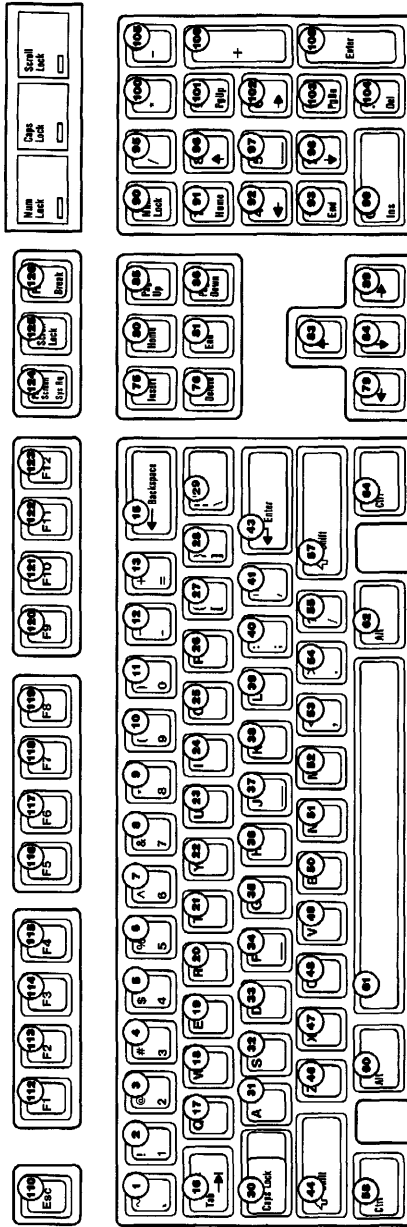
**Note:** When a receive error occurs in the default mode (bits 5, 6, and 7 of the command byte set to 0), hex 00 is placed in the output buffer instead of hex FF. See

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

### **Scan Code Translation**

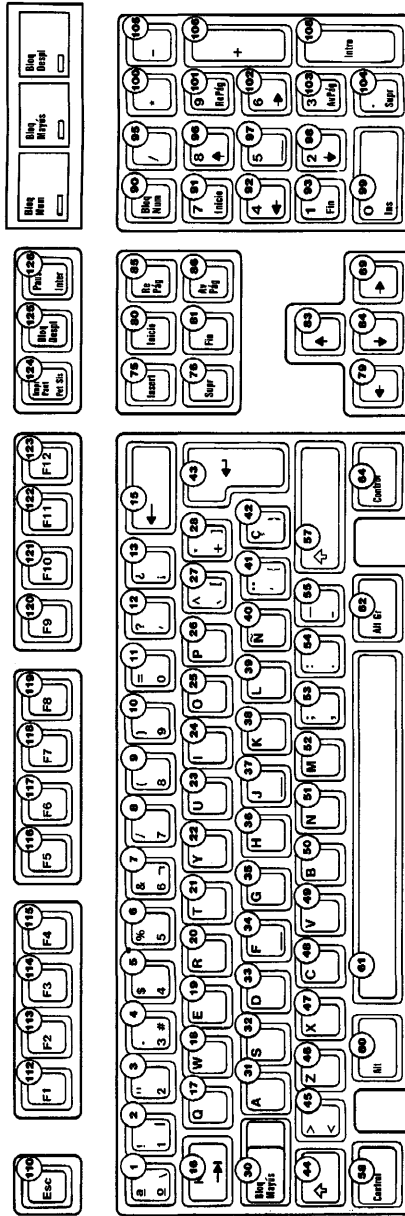
Scan codes received from the keyboard are converted by the keyboard controller before being placed into the controller’s output buffer. The following figures show the keyboard layouts. Each key position is numbered for reference.

# 101-Key Keyboard



SECTION 1

# 102-Key Keyboard



The following figure is the scan-code translation table.

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

Scan-Code Translation Table (Part 1 of 3)

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

Scan-Code Translation Table (Part 2 of 3)



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

**Scan-Code Translation Table (Part 3 of 3)**

The following scan codes are reserved.

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

**Reserved Scan-Code Translation Table**

---

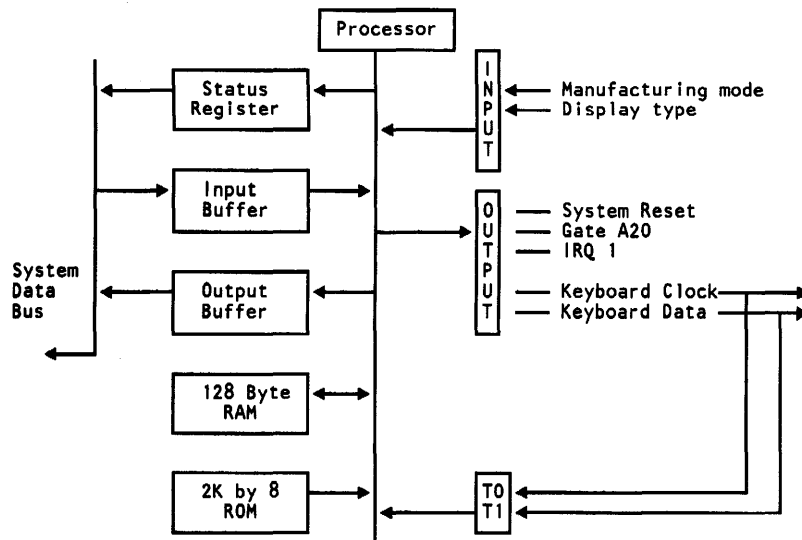
## **Sending Data to the Keyboard**

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

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

## **Keyboard Controller System Interface**

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



**Keyboard Controller Interface Block Diagram**

## Status Register

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

## Status-Register Bit Definition

- Bit 7** Parity Error—A 0 indicates the last byte of data received from the keyboard had odd parity. A 1 indicates the last byte had even parity. The keyboard should send data with odd parity.
- Bit 6** Receive Time-Out—A 1 indicates that a transmission was started by the keyboard but did not finish within the programmed receive time-out delay.
- Bit 5** Transmit Time-Out—A 1 indicates that a transmission started by the keyboard controller was not properly completed. If the transmit byte was not clocked out within the specified time limit, this will be the only error bit on.

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

**Bit 4** Always set to 1.

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

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

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

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

## **Output Buffer**

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

## **Input Buffer**

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

### **Commands (I/O Address Hex 64)**

- 20** Read Keyboard Controller's Command Byte—The controller sends its current command byte to its output buffer.
- 60** Write Keyboard Controller's Command Byte—The next byte of data written to I/O address hex 60 is placed in the controller's command byte. Bit definitions of the command byte are as follows:

**Bit 7** Reserved—Should be written as a 0.

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

- Bit 5 IBM Personal Computer Mode**—Writing a 1 to this bit programs the keyboard to support the IBM Personal Computer keyboard interface. In this mode the controller does not check parity or convert scan codes.
- Bit 4 Disable Keyboard**—Writing a 1 to this bit disables the keyboard interface by driving the 'clock' line low. Data is not sent or received.
- Bit 3** Not used.
- Bit 2 System Flag**—The value written to this bit is placed in the system flag bit of the controller's status register.
- Bit 1 Reserved**—Should be written as a 0.
- Bit 0 Enable Output-Buffer-Full Interrupt**—Writing a 1 to this bit causes the controller to generate an interrupt when it places data into its output buffer.
- AA Self-Test**—This commands the controller to perform internal diagnostic tests. A hex 55 is placed in the output buffer if no errors are detected.
- AB Interface Test**—This commands the controller to test the 'keyboard clock' and 'keyboard data' lines. The test result is placed in the output buffer as follows:
- 00** No error detected.
  - 01** The 'keyboard clock' line is stuck low.
  - 02** The 'keyboard clock' line is stuck high.
  - 03** The 'keyboard data' line is stuck low.
  - 04** The 'keyboard data' line is stuck high.
- AD Disable Keyboard Feature**—This command sets bit 4 of the controller's command byte. This disables the keyboard interface by driving the clock line low. Data will not be sent or received.
- AE Enable Keyboard Interface**—This command clears bit 4 of the command byte, which releases the keyboard interface.

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

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

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

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

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

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

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

## **I/O Ports**

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

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



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

#### Input-Port Bit Definitions

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

#### Output-Port Bit Definitions

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

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

#### Test-Input Bit Definitions

## Real-Time Clock CMOS RAM Information

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

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

### CMOS RAM Internal Address Map

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

### Real-Time Clock Information

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

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

### Real-Time Clock Internal Addresses 00 - 0D

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

### Status Register A

- Bit 7** Update in Progress (UIP)—A 1 indicates the time update cycle is in progress. A 0 indicates the current date and time are available to read.
- Bit 6–Bit 4** 22-Stage Divider (DV2 through DV0)—These three divider-selection bits identify which time-base frequency is being used. The system initializes the stage divider to 010, which selects a 32.768-kHz time base.
- Bit 3–Bit 0** Rate Selection Bits (RS3 through RS0)—These bits allow the selection of a divider output frequency. The system initializes the rate selection bits to 0110, which selects a 1.024-kHz square wave output frequency and a 976.562-microsecond periodic interrupt rate.

### Status Register B

- Bit 7** Set—A 0 updates the cycle normally by advancing the counts at one-per-second. A 1 aborts any update cycle in progress and the program can initialize the 14 time-bytes without any further updates occurring until a 0 is written to this bit.
- Bit 6** Periodic Interrupt Enable (PIE)—This bit is a read/write bit that allows an interrupt to occur at a rate specified by the rate and divider bits in register A. A 1 enables an interrupt, and a 0 disables it. The system initializes this bit to 0.
- Bit 5** Alarm Interrupt Enable (AIE)—A 1 enables the alarm interrupt, and a 0 disables it. The system initializes this bit to 0.

- Bit 4** Update-Ended Interrupt Enabled (UIE)—A 1 enables the update-ended interrupt, and a 0 disables it. The system initializes this bit to 0.
- Bit 3** Square Wave Enabled (SQWE)—A 1 enables the square-wave frequency as set by the rate selection bits in register A, and a 0 disables the square wave. The system initializes this bit to 0.
- Bit 2** Date Mode (DM)—This bit indicates whether the time and date calendar updates are to use binary or binary coded decimal (BCD) formats. A 1 indicates binary, and a 0 indicates BCD. The system initializes this bit to 0.
- Bit 1** 24/12—This bit indicates whether the hours byte is in the 24-hour or 12-hour mode. A 1 indicates the 24-hour mode and a 0 indicates the 12-hour mode. The system initializes this bit to 1.
- Bit 0** Daylight Savings Enabled (DSE)—A 1 enables daylight savings and a 0 disables daylight savings (standard time). The system initializes this bit to 0.

#### Status Register C

- Bit 7–Bit 4** IRQF, PF, AF, UF—These flag bits are read-only and are affected when the AIE, PIE, and UIE bits in register B are set to 1.
- Bit 3–Bit 0** Reserved—Should be written as a 0.

#### Status Register D

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

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

## CMOS RAM Configuration Information

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

### Diagnostic Status Byte (Hex 0E)

- Bit 7** Power status of the real-time clock chip—A 0 indicates that the chip has not lost power (battery good), and a 1 indicates that the chip lost power (battery bad).
- Bit 6** Configuration Record (Checksum Status Indicator)—A 0 indicates that checksum is good, and a 1 indicates it is bad.
- Bit 5** Incorrect Configuration Information—This is a check, at power-on time, of the equipment byte of the configuration record. A 0 indicates that the configuration information is valid, and a 1 indicates it is invalid. Power-on checks require:
- At least one diskette drive to be installed (bit 0 of the equipment byte set to 1).
  - The primary display adapter setting in configuration matches the system board's display switch setting and the actual display adapter hardware in the system.
- Bit 4** Memory Size Comparison—A 0 indicates that the power-on check determined the same memory size as in the configuration record, and a 1 indicates the memory size is different.
- Bit 3** Fixed Disk Adapter/Drive C Initialization Status—A 0 indicates that the adapter and drive are functioning properly and the system can attempt "boot up." A 1 indicates that the adapter

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

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

**Bit 1–Bit 0** Reserved

### **Shutdown Status Byte (Hex 0F)**

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

### **Diskette Drive Type Byte (Hex 10)**

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

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

**Note:** 0100 through 1111 are reserved.

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

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

**Note:** 0100 through 1111 are reserved.

**Hex address 11 contains a reserved byte.**

### **Fixed Disk Type Byte (Hex 12)**

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

**0000** No fixed disk drive is present.

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

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

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

**0000** No fixed disk drive is present.

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

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

The following table shows the BIOS fixed disk parameters.

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

**BIOS Fixed Disk Parameters**

---

**Hex address 13 contains a reserved byte.**

**Equipment Byte (Hex 14)**

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

- 00** 1 drive
- 01** 2 drives
- 10** Reserved
- 11** Reserved

**Bit 5–Bit 4** Primary display

- 00** Primary display is attached to an adapter that has its own BIOS, such as the Enhanced Graphics Adapter
- 01** Primary display is in the 40-column mode and attached to the Color/Graphics Monitor Adapter.
- 10** Primary display is in the 80-column mode and attached to the Color/Graphics Monitor Adapter.
- 11** Primary display is attached to the Monochrome Display and Printer Adapter.

**Bit 3–Bit 2** Not used.

**Bit 1** Math Coprocessor presence bit:

- 0** Math Coprocessor not installed
- 1** Math Coprocessor installed

**Bit 0** Diskette drive presence bit:

- 0** Diskette drive not installed
- 1** Diskette drive installed



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

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

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

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

Valid Sizes:

**0200H** 512K—system board RAM

**0280H** 640K—system board RAM.

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

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

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

Valid Sizes:

**0200H** 512K—Expansion Memory

**0400H** 1024K—Expansion Memory

**0600H** 1536K—Expansion Memory

**through**

**3C00H** 15360K—Expansion Memory (15M maximum).

---

### Drive C Extended Byte (Hex 19)

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

00000000 through 00001111 are reserved.

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

### Drive D Extended Byte (Hex 1A)

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

00000000 through 00001111 are reserved.

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

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

**Note:** Types 25 through 255 are reserved.

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

**BIOS Fixed Disk Parameters (Extended)**

Hex addresses 1B through 2D are reserved.

#### Checksum (Hex 2E and 2F)

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

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

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

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

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

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

Valid Sizes:

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

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

#### Date Century Byte (Hex 32)

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

---

### Information Flag (Hex 33)

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

Hex addresses 34 through 3F are reserved.

### I/O Operations

Writing to RTC CMOS RAM involves two steps:

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

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

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

Reading CMOS RAM also requires two steps:

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

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

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

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



# **Specifications**

## **System Unit**

### **Size**

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

### **Weight**

- 12.7 kilograms (28 pounds)

### **Power Cables**

- Length: 1.8 meters (6 feet)

### **Environment**

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

- System Off: 20% to 80%
- Altitude
  - Maximum altitude: 2545.1 meters (8350 feet)

### Heat Output

- 824 British Thermal Units (BTU) per hour

### Noise Level

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

### Electrical

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

## Connectors

The system board has the following additional connectors:

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

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

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

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



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

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

#### Battery Connector (P2)

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

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

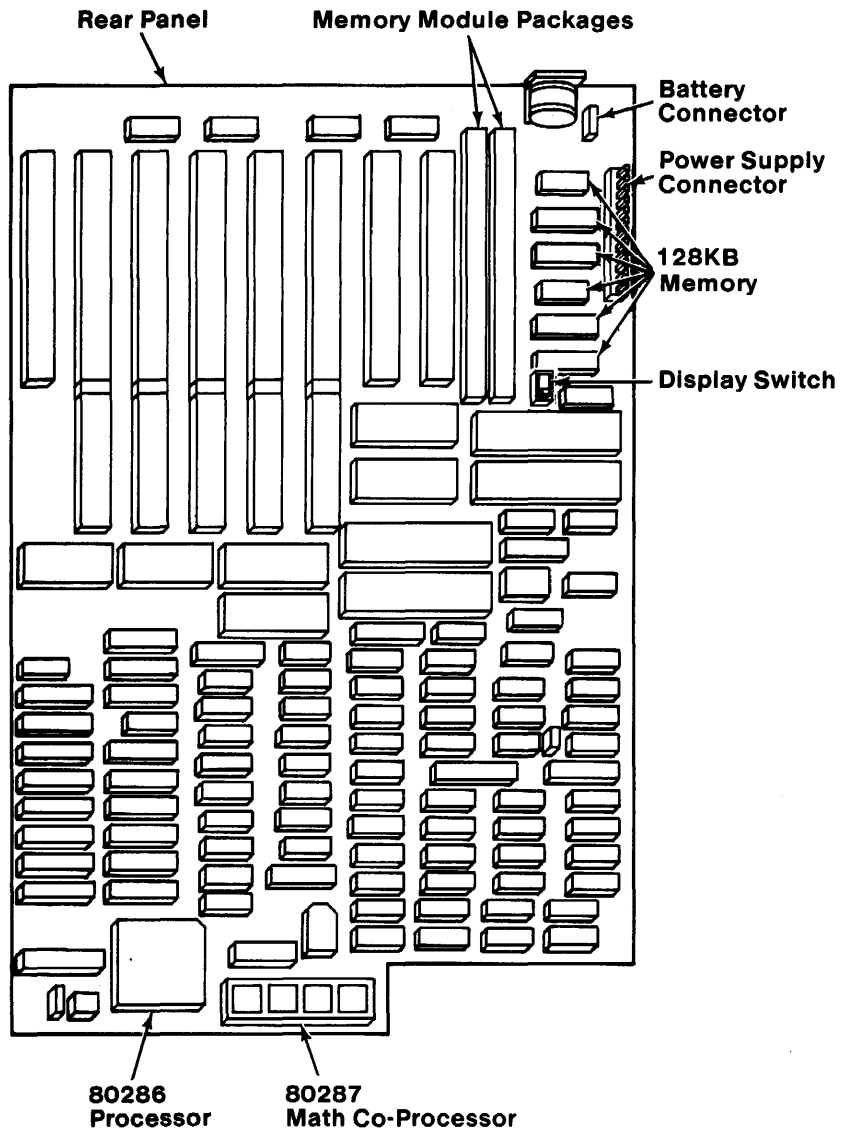
#### Speaker Connector (P3)

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

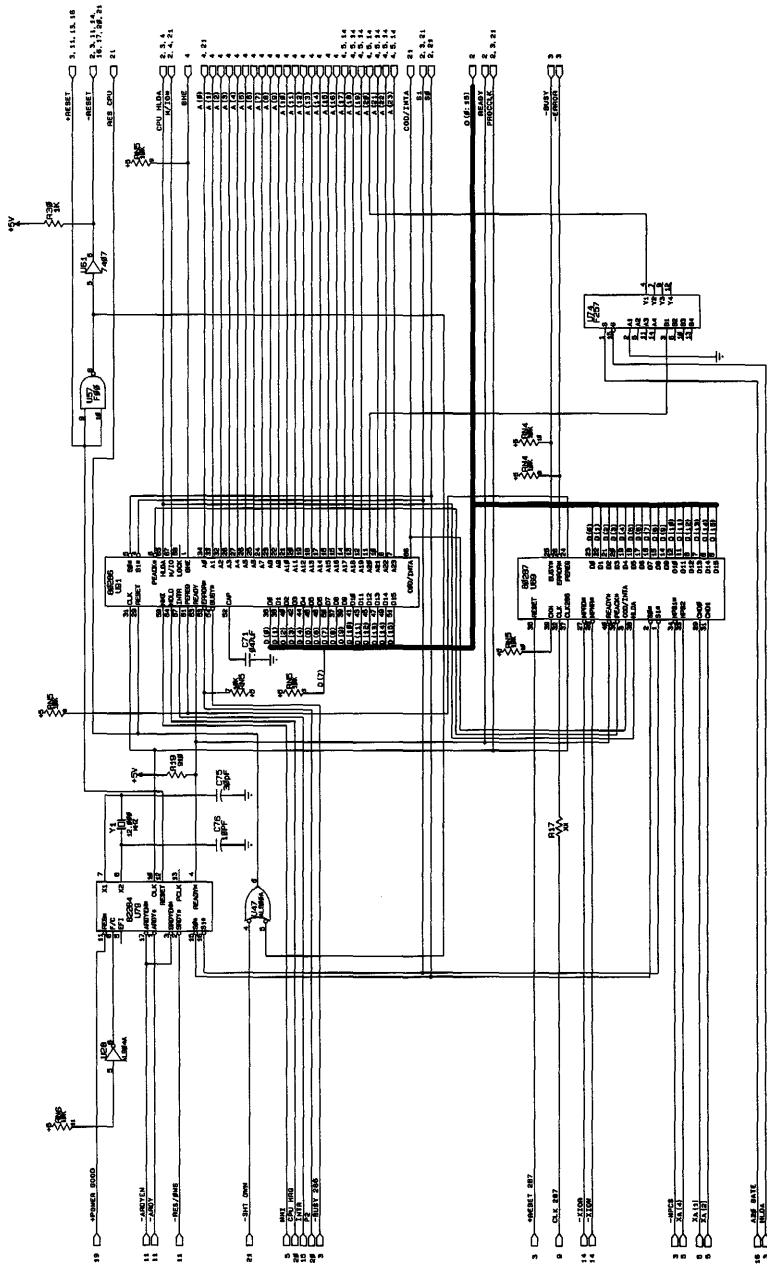
Pin	Assignments
1	Keyboard Clock
2	Keyboard Data
3	Reserved
4	Ground
5	+5 Vdc

#### Keyboard Connector (J9)

The following figure shows the layout of the system board.



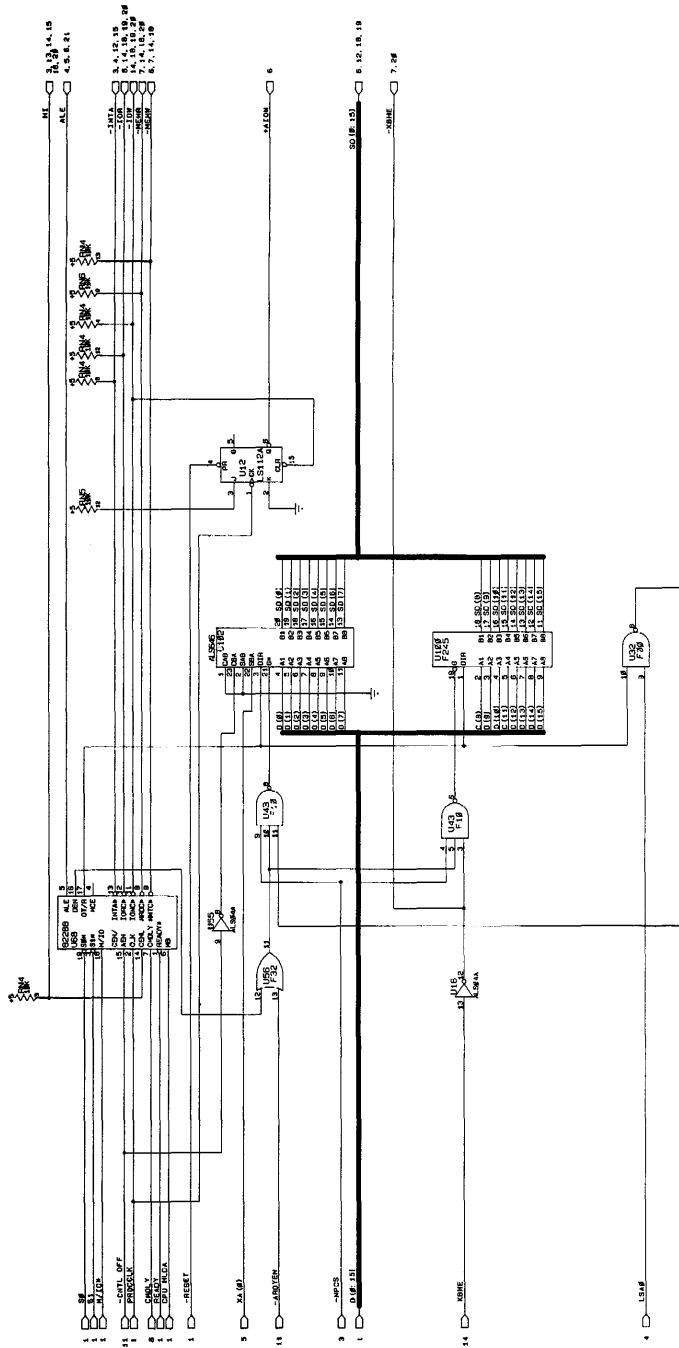
# Logic Diagrams



System Board (Sheet 1 of 22)

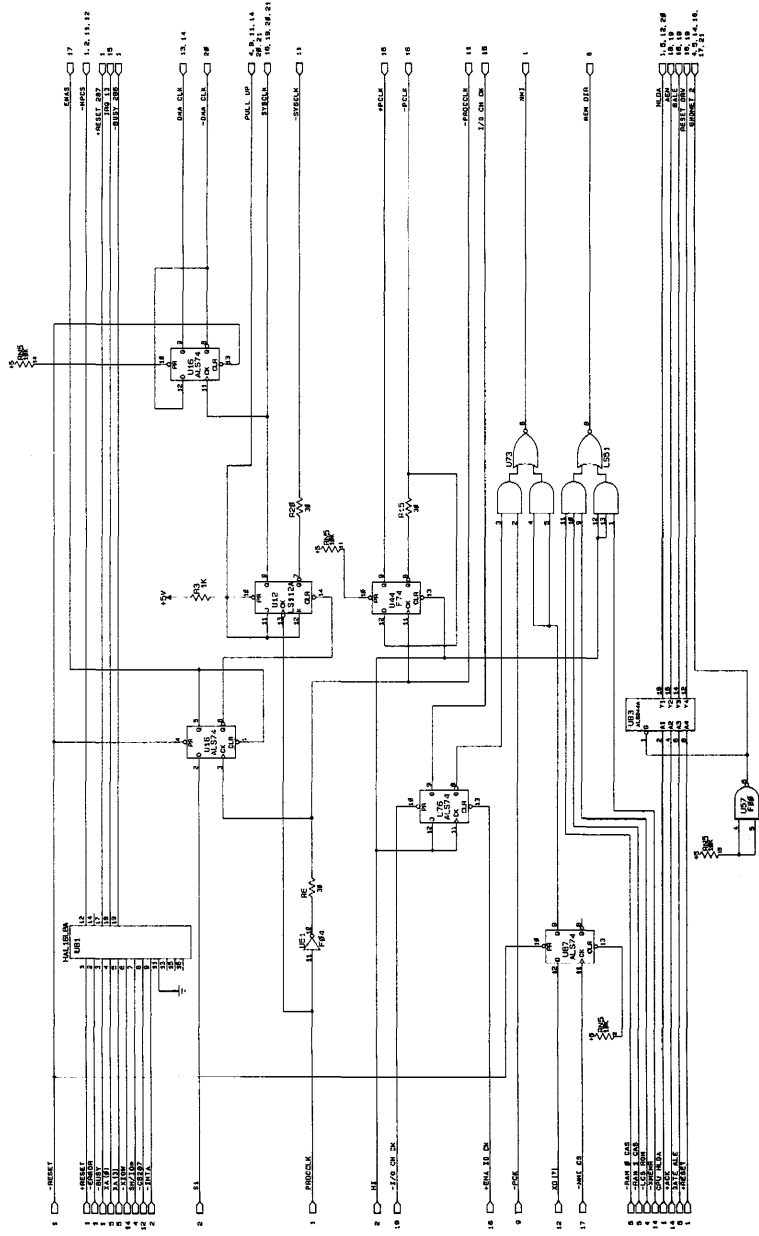
SECTION 1

1-78 System Board



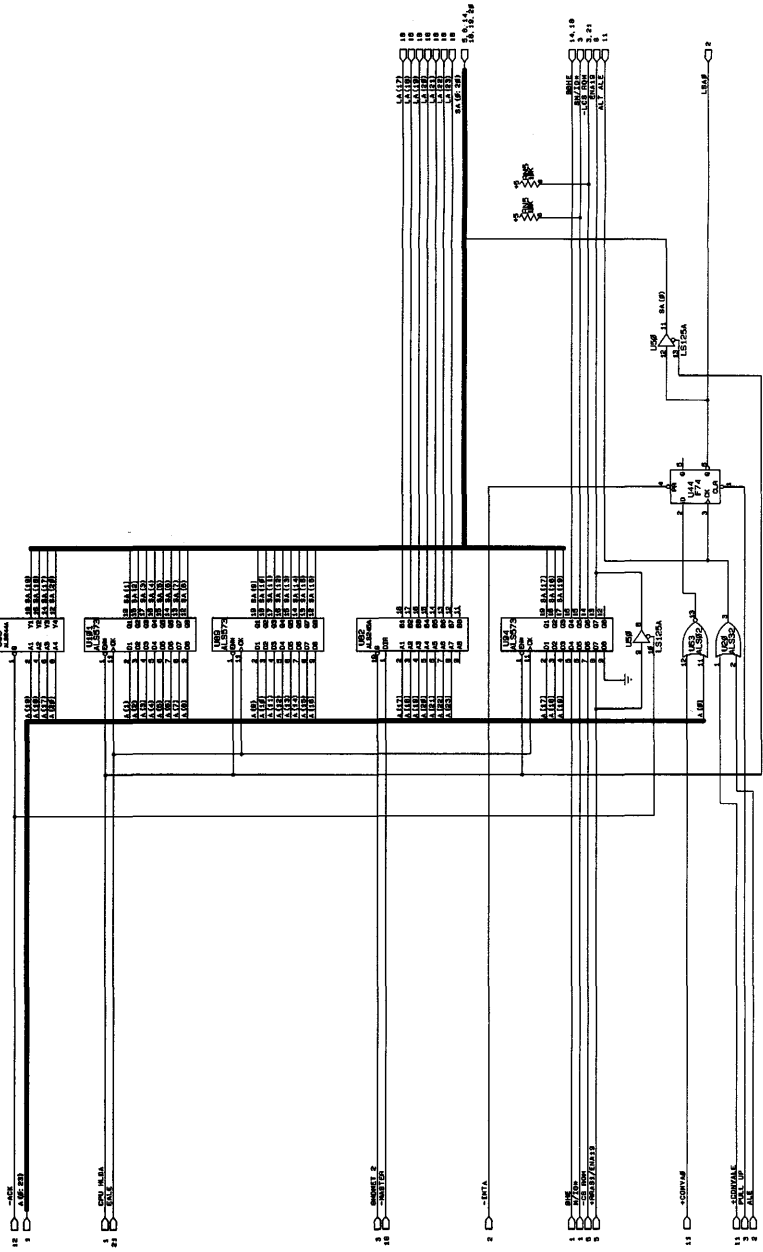
System Board (Sheet 2 of 22)

( )



System Board (Sheet 3 of 22)

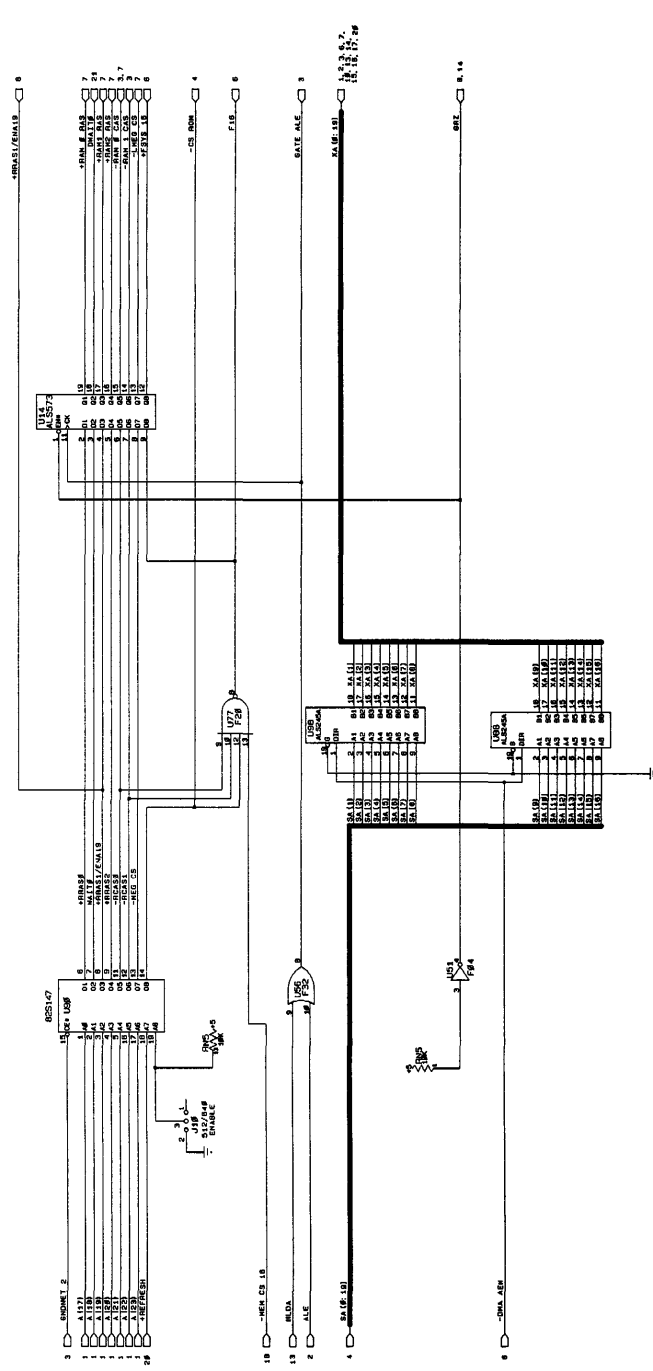
SECTION 1



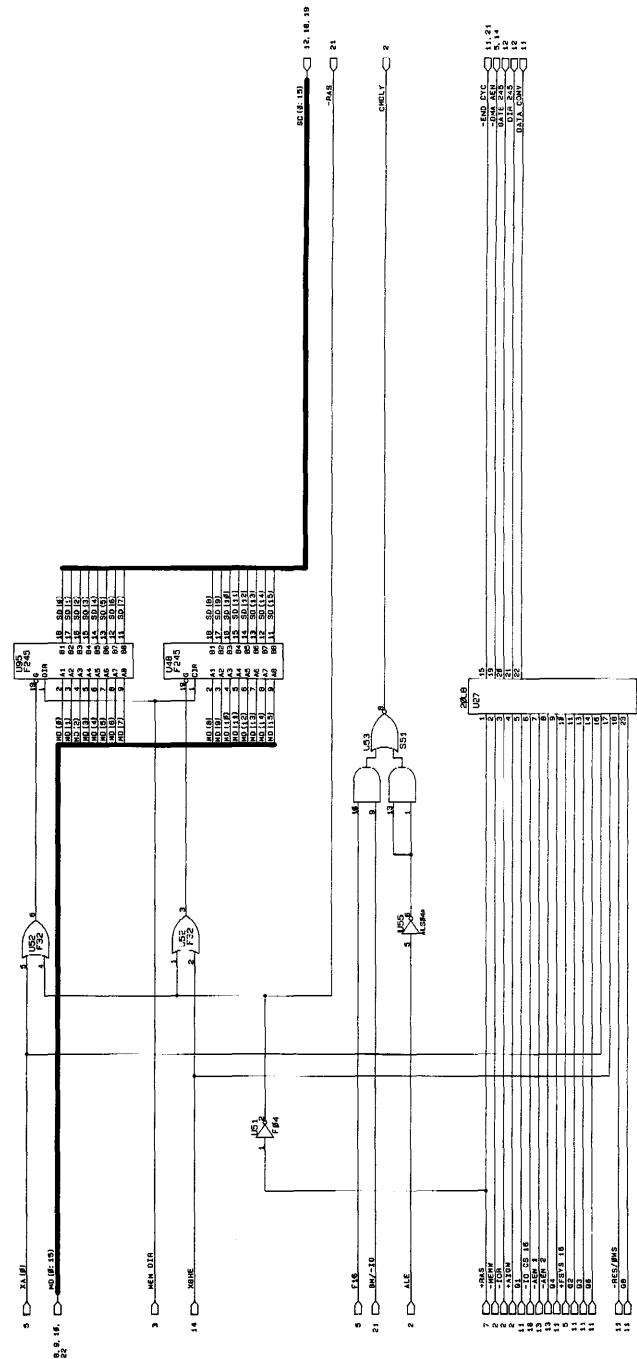
1-80 System Board

System Board (Sheet 4 of 22)

( )



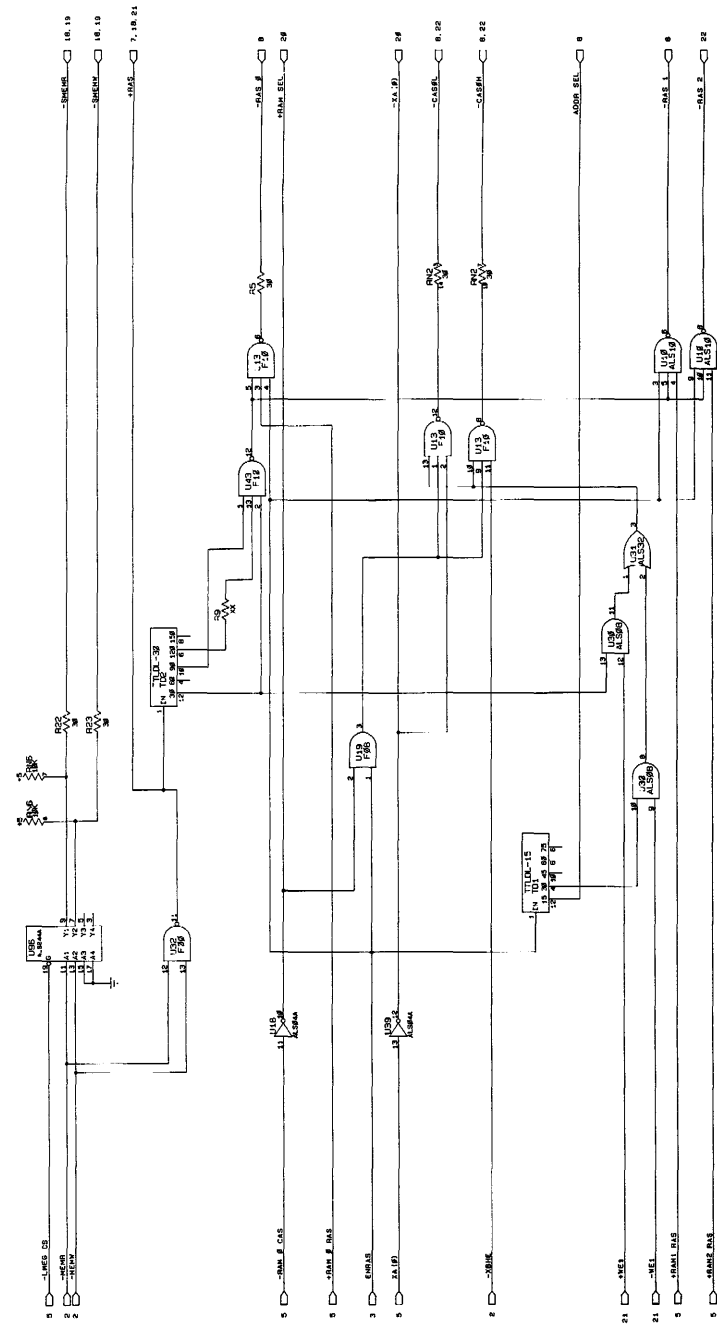
1-82 System Board



System Board (Sheet 6 of 22)



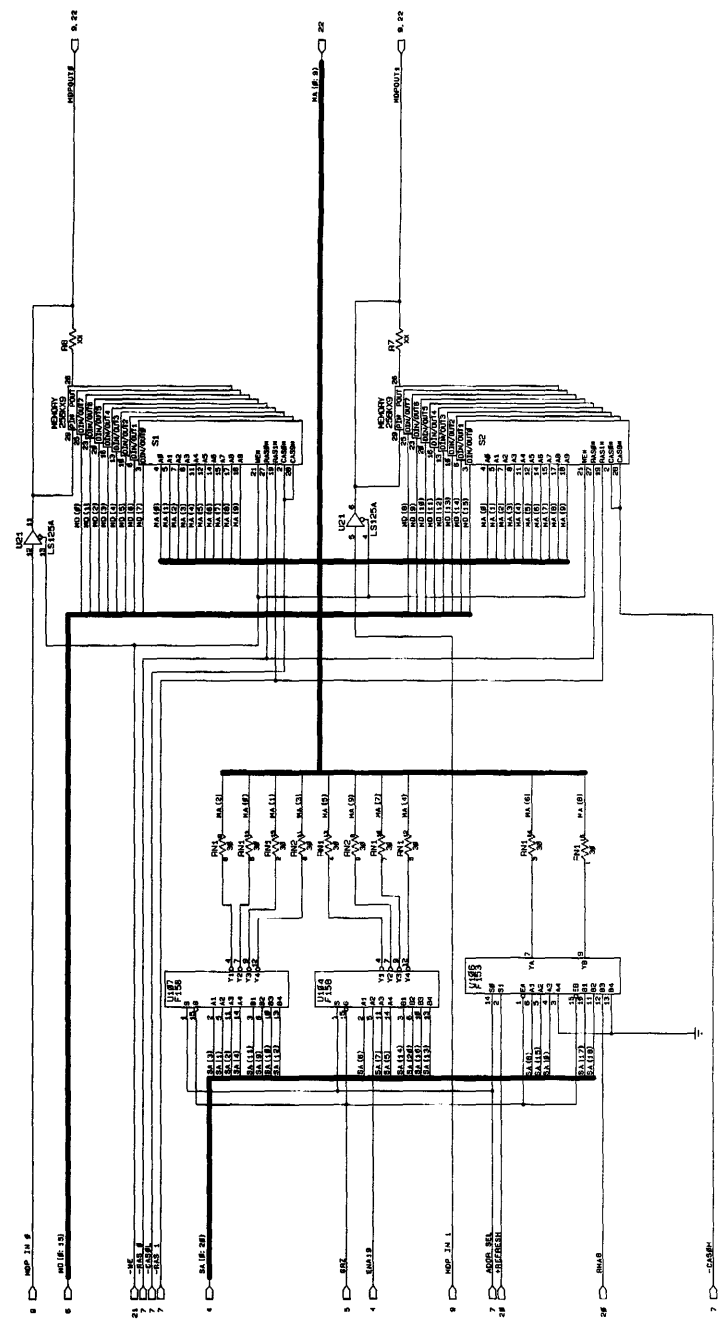
( )



System Board (Sheet 7 of 22)

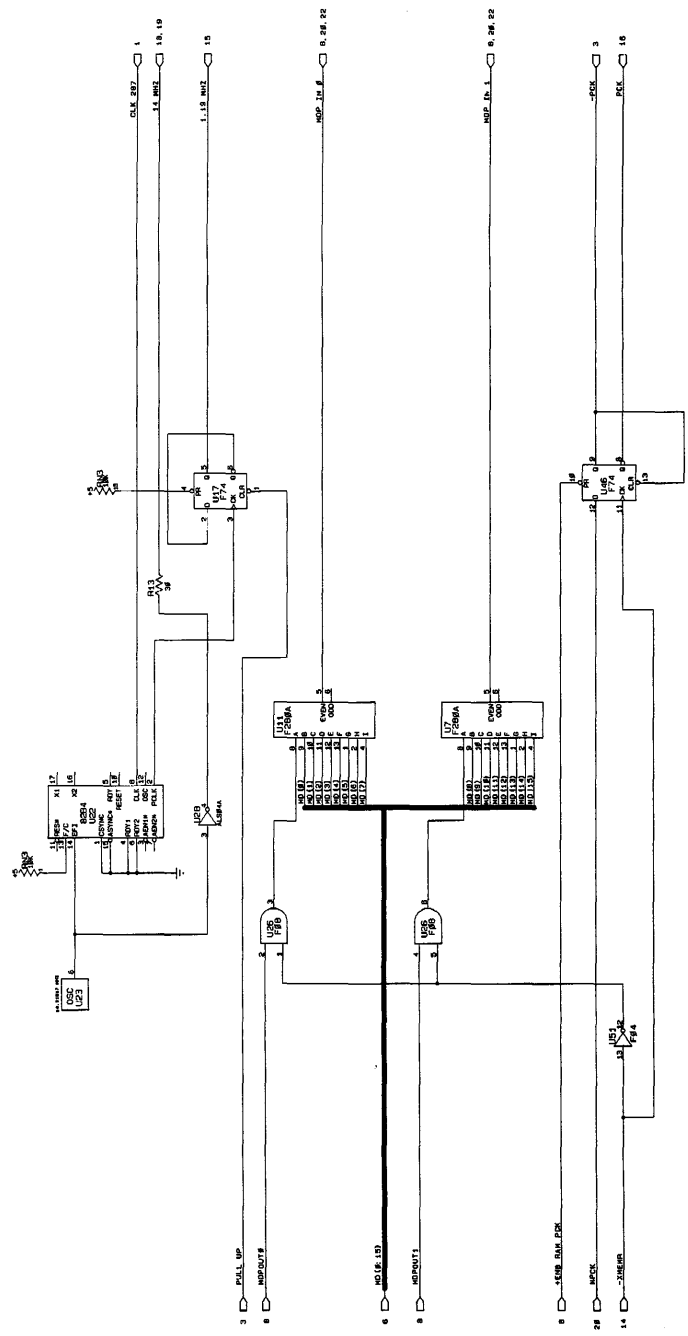
SECTION 1

1-84 System Board



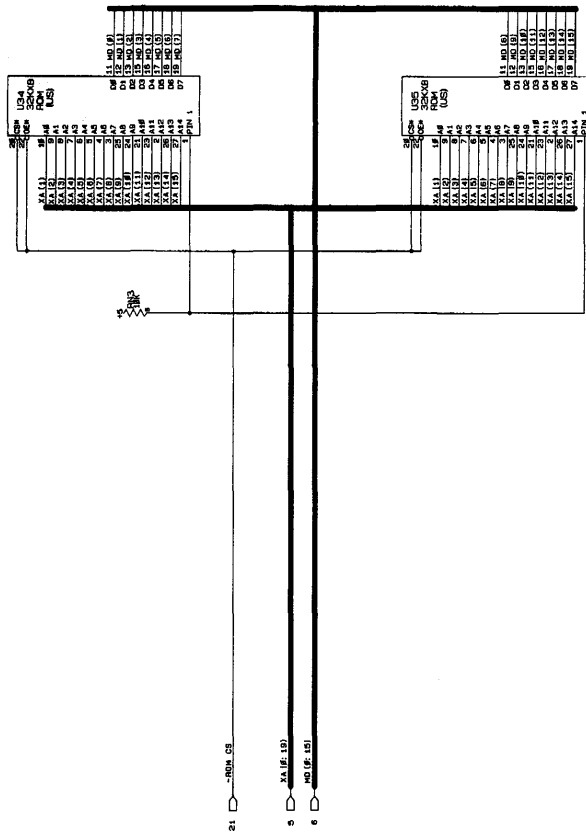
System Board (Sheet 8 of 22)

( ) ( )

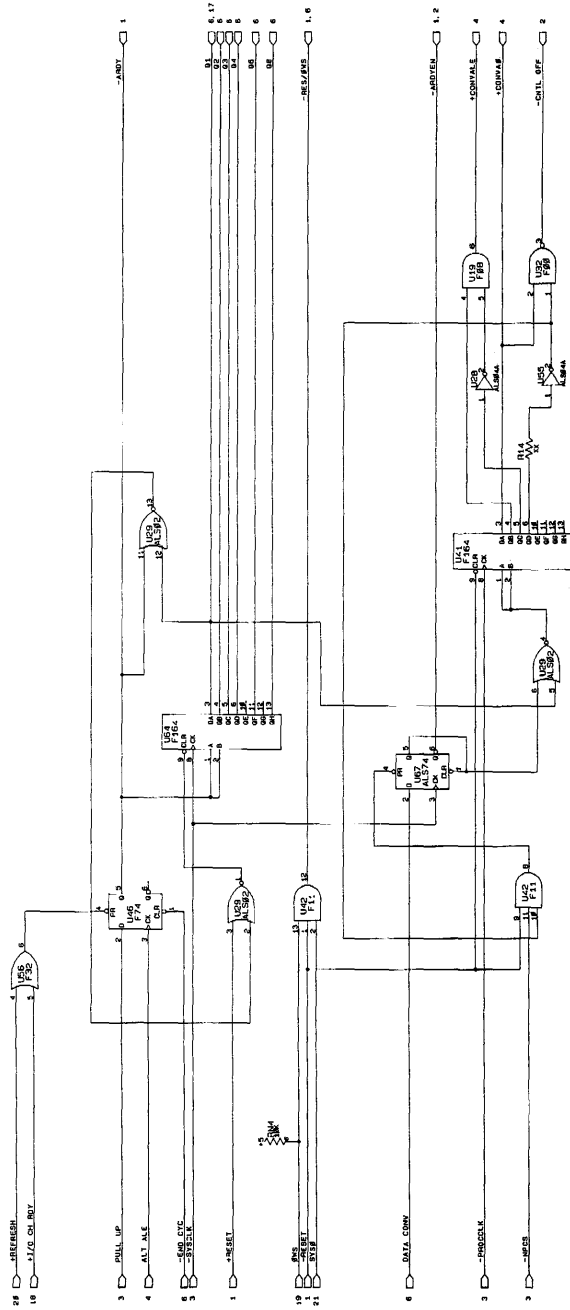


SECTION 1

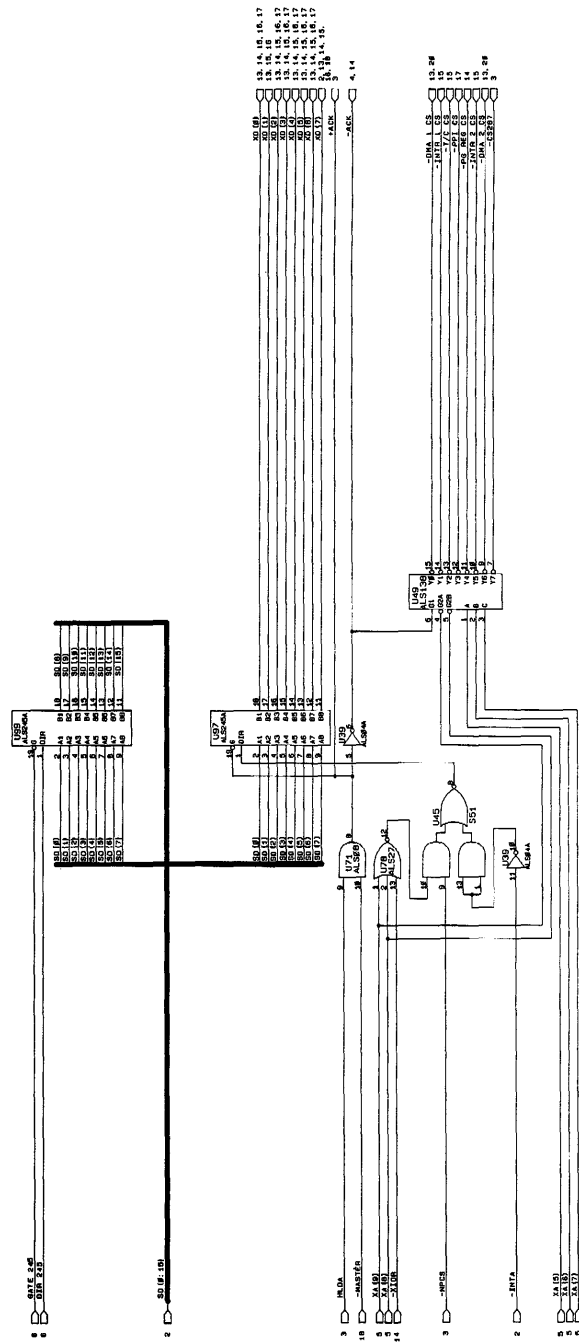
1-86 System Board



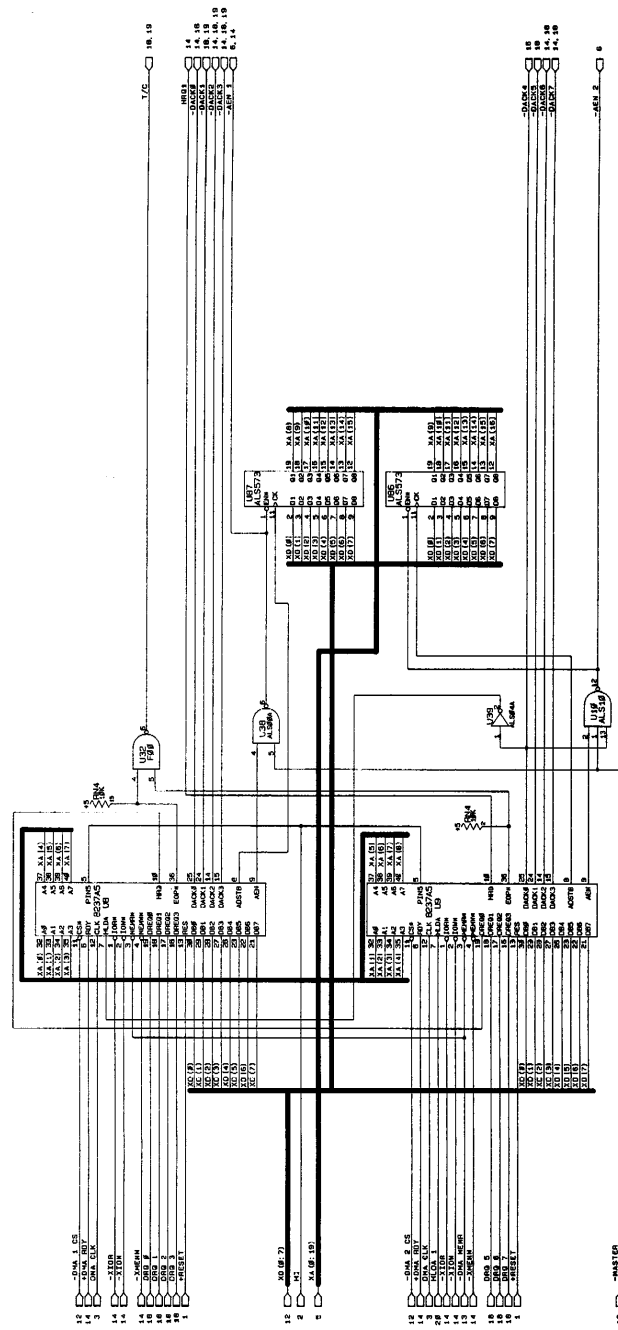
System Board (Sheet 10 of 22)



1-88 System Board



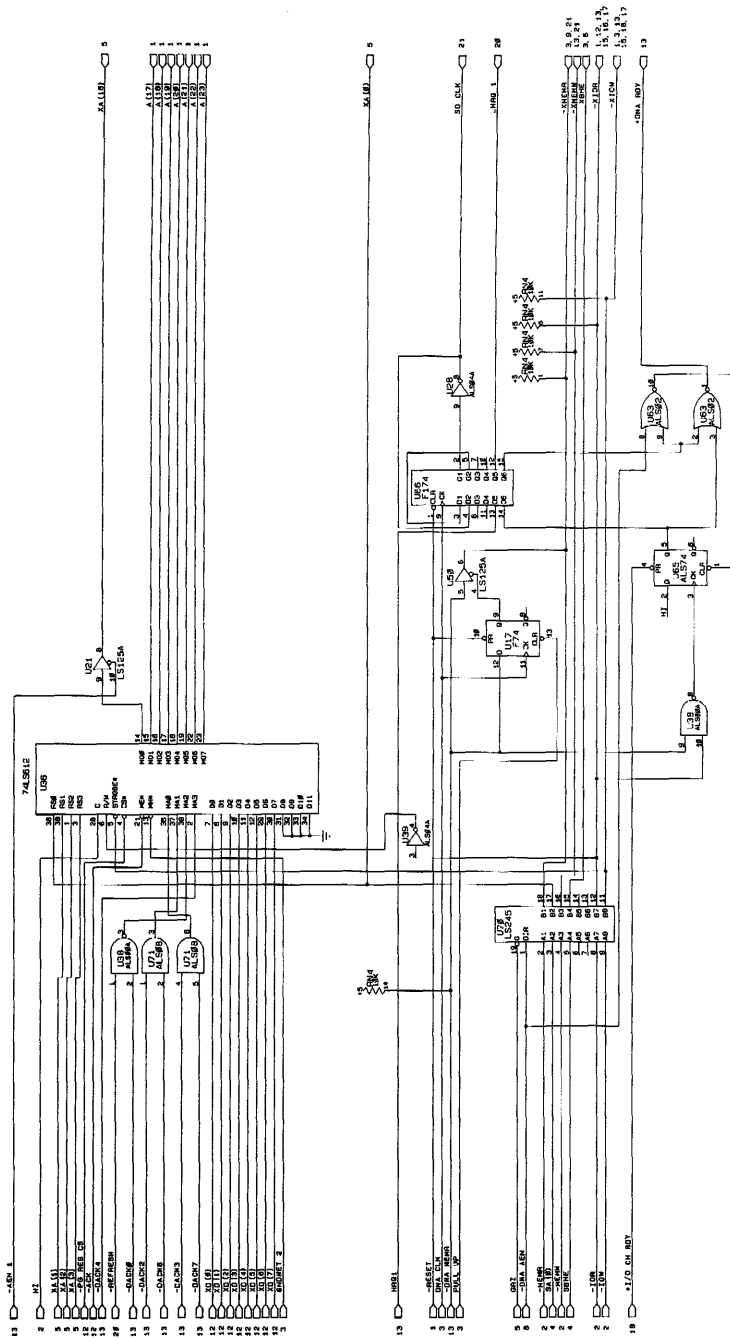
System Board (Sheet 12 of 22)



System Board (Sheet 13 of 22)

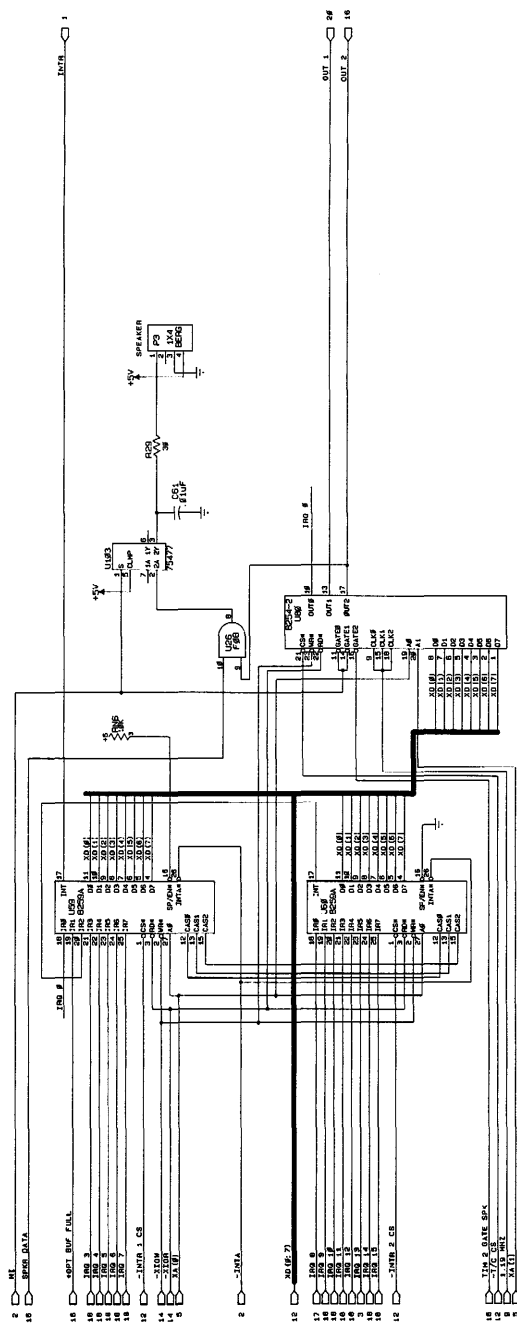
SECTION 1

# 1-90 System Board



System Board (Sheet 14 of 22)

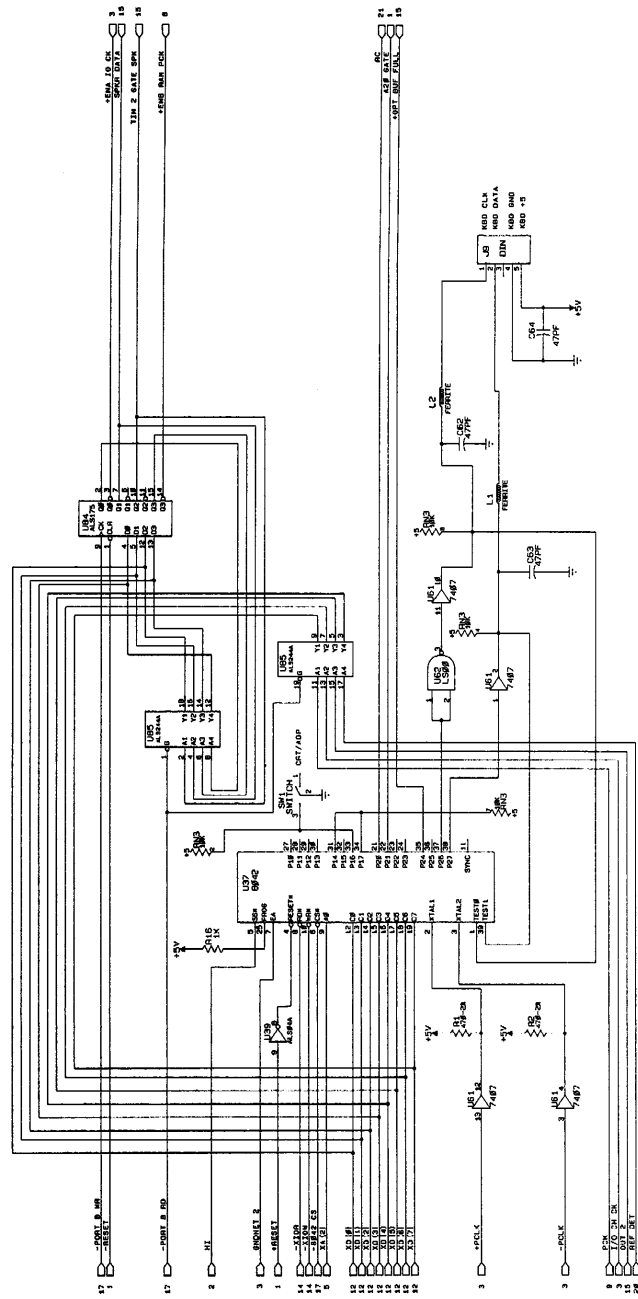




SECTION 1

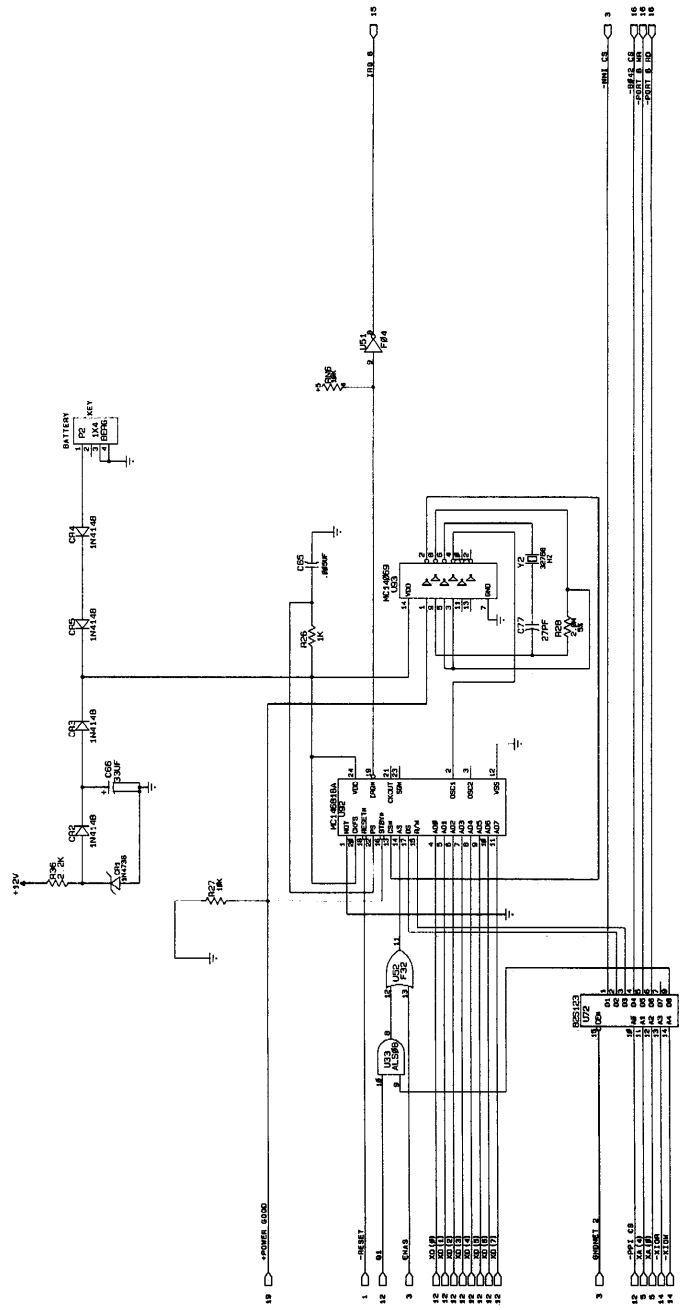
System Board (Sheet 15 of 22)

1-92 System Board



System Board (Sheet 16 of 22)

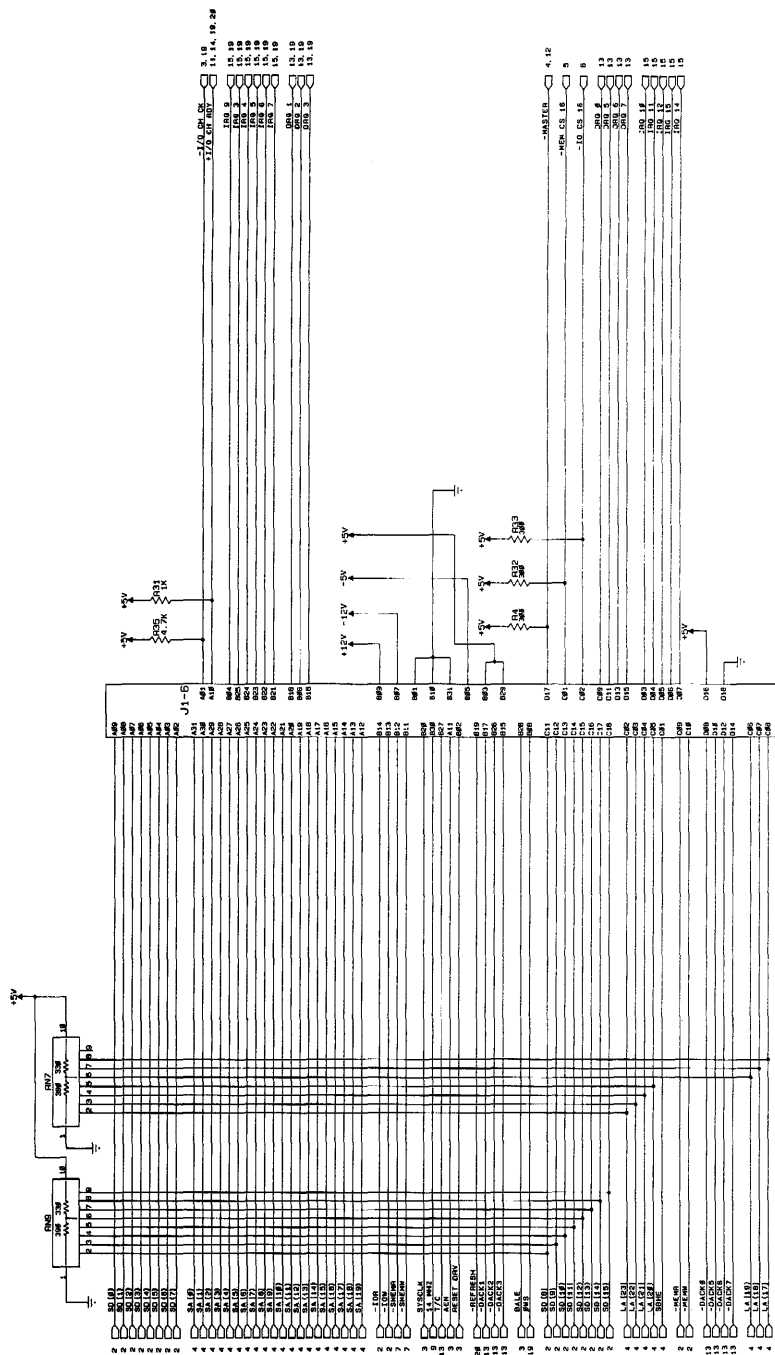
( )



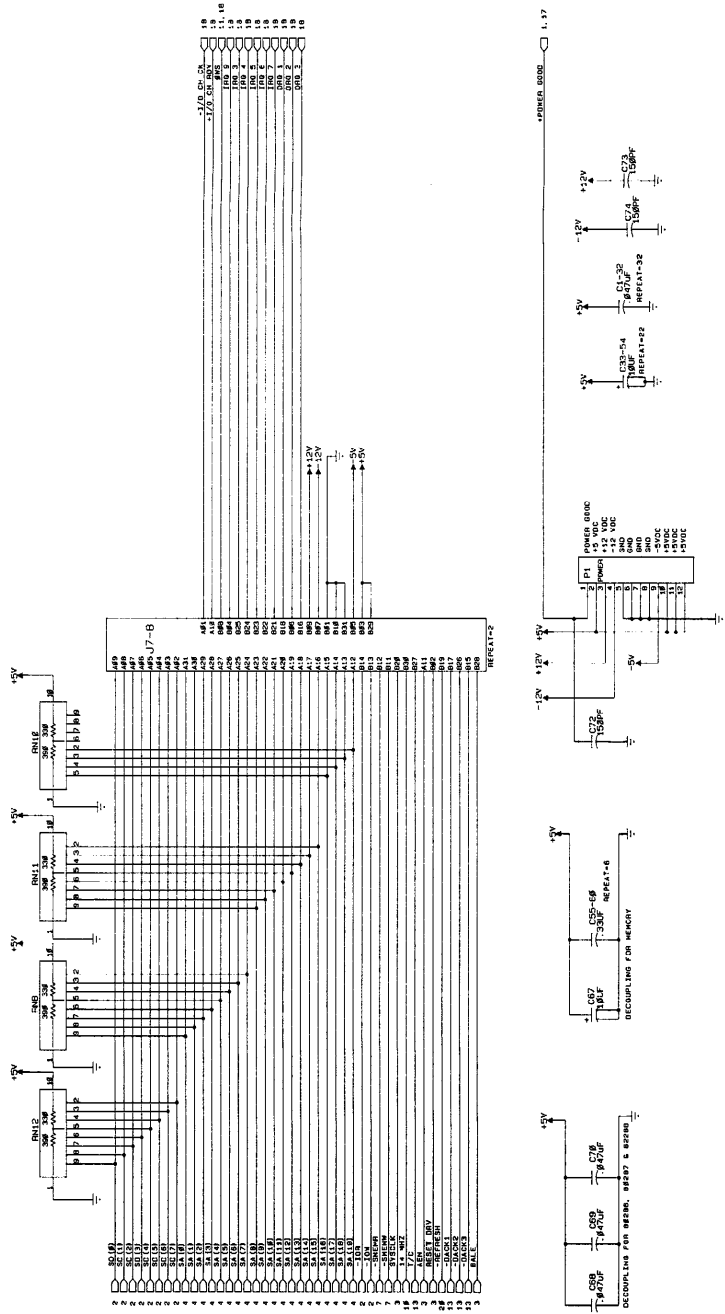
System Board (Sheet 17 of 22)

SECTION 1

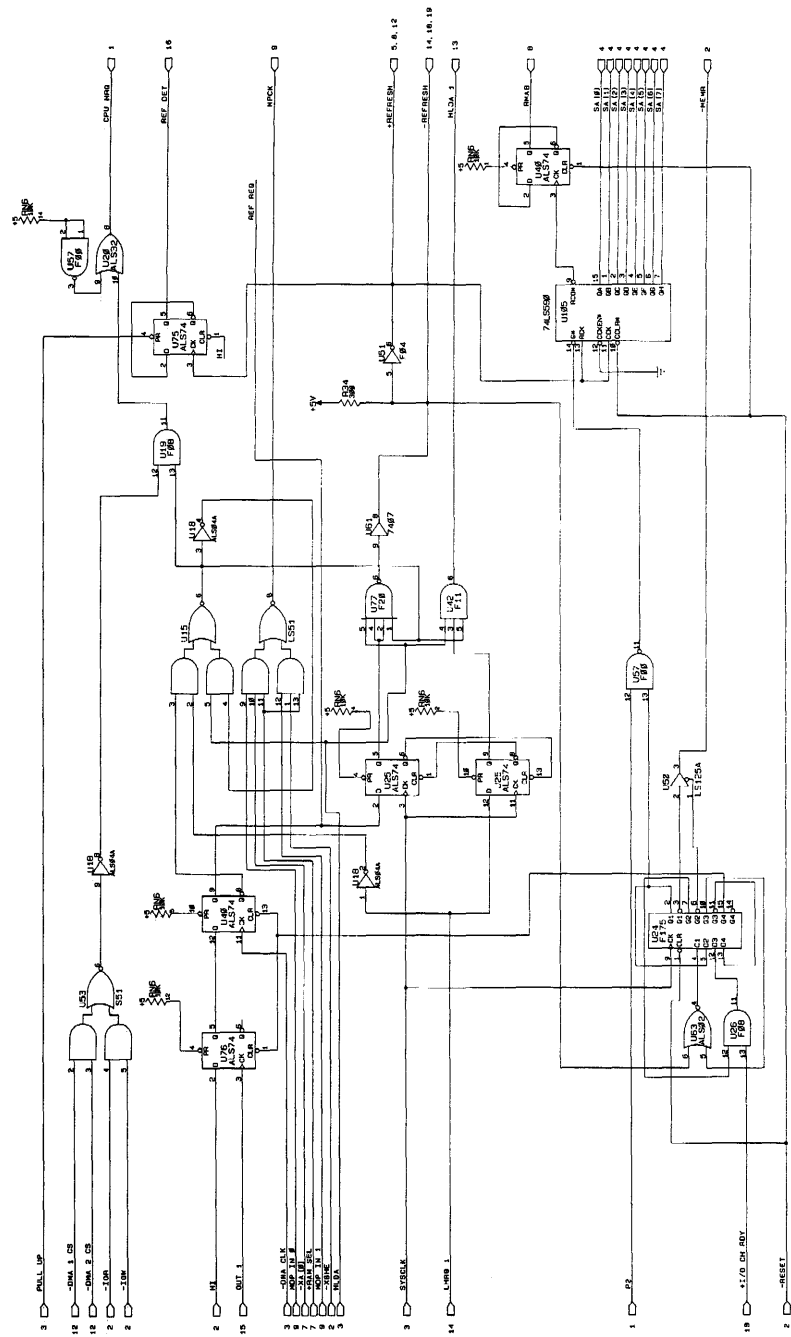
### 1-94 System Board



System Board (Sheet 18 of 22)

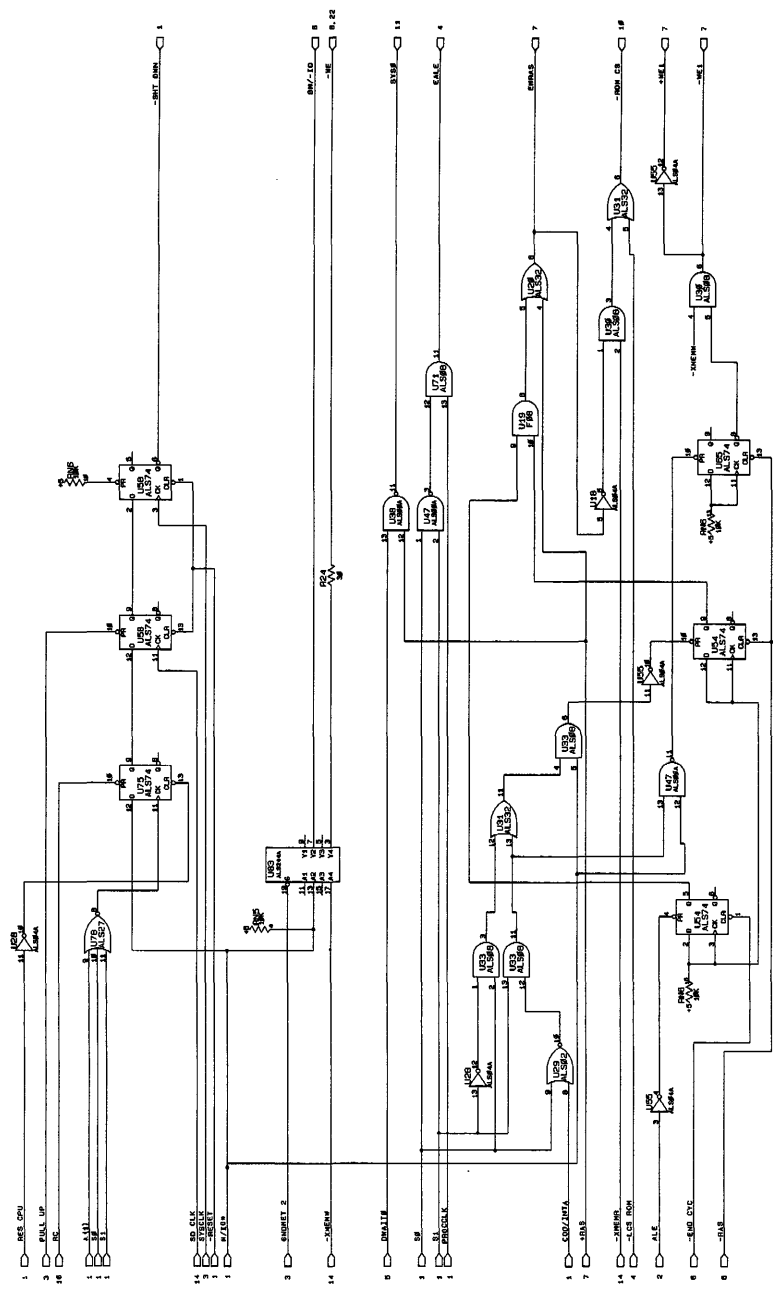


1-96 System Board

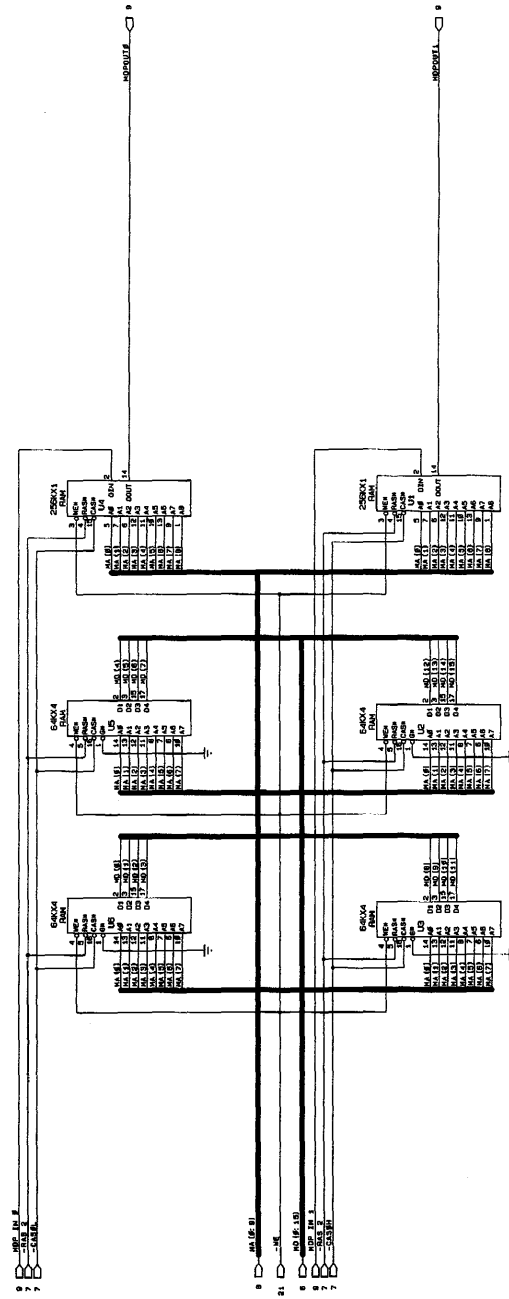


System Board (Sheet 20 of 22)

( ) ( )



1-98 System Board



System Board (Sheet 22 of 22)



## SECTION 2. COPROCESSOR

Description .....	2-3
Programming Interface .....	2-3
Hardware Interface .....	2-4

## Notes:

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

## Description

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

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

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

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

## Programming Interface

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

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

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

Data Type	Bits	Significant Digits (Decimal)	Approximate Range (Decimal)
Word Integer	16	4	$-32,768 \leq X \leq +32,767$
Short Integer	32	9	$-2 \times 10^9 \leq X \leq +2 \times 10^9$
Long Integer	64	18	$-9 \times 10^{18} \leq X \leq +9 \times 10^{18}$
Packed Decimal	80	18	$-9.99 \leq X \leq +9.99$ (18 digits)
Short Real *	32	6-7	$8.43 \times 10^{-37} \leq  X  \leq 3.37 \times 10^{38}$
Long Real *	64	15-16	$4.19 \times 10^{-307} \leq  X  \leq 1.67 \times 10^{308}$
Temporary Real	80	19	$3.4 \times 10^{-4932} \leq  X  \leq 1.2 \times 10^{4932}$

#### Data Types

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

## Hardware Interface

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

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

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

#### 2-4 Coprocessor

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

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

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

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

## Notes:

|

—

—

—

# SECTION 3. POWER SUPPLY

- Inputs ..... 3-3
- Outputs ..... 3-3
- DC Output Protection ..... 3-4
- Output Voltage Sequencing ..... 3-4
- No-Load Operation ..... 3-4
- Power-Good Signal ..... 3-4
- Connectors ..... 3-6

SECTION 3

---

**Notes:**



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

## Inputs

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

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

### Input Requirements

## Outputs

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

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

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

### DC Load Requirements

## **DC Output Protection**

An overcurrent condition will not damage the power supply.

## **Output Voltage Sequencing**

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

## **No-Load Operation**

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

## **Power-Good Signal**

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

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

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

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

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

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

**Sense Level**

## Connectors

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

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

**Power Supply Output Connectors**

## SECTION 4. KEYBOARD

Description .....	4-3
Cabling .....	4-3
Sequencing Key-Code Scanning .....	4-4
Keyboard Buffer .....	4-4
Keys .....	4-4
Power-On Routine .....	4-5
Power-On Reset .....	4-5
Basic Assurance Test .....	4-5
Commands from the System .....	4-6
Default Disable (Hex F5) .....	4-7
Echo (Hex EE) .....	4-7
Enable (Hex F4) .....	4-7
Invalid Command (Hex EF and F1) .....	4-7
Read ID (Hex F2) .....	4-7
Resend (Hex FE) .....	4-8
Reset (Hex FF) .....	4-8
Select Alternate Scan Codes (Hex F0) .....	4-8
Set All Keys (Hex F7, F8, F9, FA) .....	4-9
Set Default (Hex F6) .....	4-9
Set Key Type (Hex FB, FC, FD) .....	4-9
Set/Reset Status Indicators (Hex ED) .....	4-10
Set Typematic Rate/Delay (Hex F3) .....	4-11
Commands to the System .....	4-13
Acknowledge (Hex FA) .....	4-13
BAT Completion Code (Hex AA) .....	4-13
BAT Failure Code (Hex FC) .....	4-13
Echo (Hex EE) .....	4-13
Keyboard ID (Hex 83AB) .....	4-14
Key Detection Error (Hex 00 or FF) .....	4-14
Overrun (Hex 00 or FF) .....	4-14
Resend (Hex FE) .....	4-14
Keyboard Scan Codes .....	4-15
Scan Code Set 1 .....	4-16
Scan Code Set 2 .....	4-20
Scan Code Set 3 .....	4-24
Clock and Data Signals .....	4-27
Data Stream .....	4-27

Keyboard Data Output .....	4-28
Keyboard Data Input .....	4-29
Keyboard Encoding and Usage .....	4-30
Character Codes .....	4-30
Extended Functions .....	4-34
Shift States .....	4-36
Special Handling .....	4-38
Keyboard Layouts .....	4-40
French Keyboard .....	4-41
German Keyboard .....	4-42
Italian Keyboard .....	4-43
Spanish Keyboard .....	4-44
U.K. English Keyboard .....	4-45
U.S. English Keyboard .....	4-46
Specifications .....	4-47
Power Requirements .....	4-47
Size .....	4-47
Weight .....	4-47
Logic Diagram .....	4-48

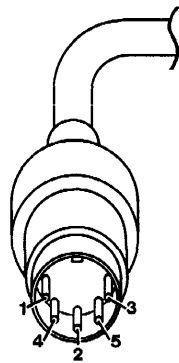
4-2 **Keyboard**

## Description

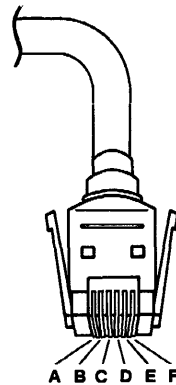
The keyboard has 101 keys (102 in countries outside the U. S.). At system power-on, the keyboard monitors the signals on the 'clock' and 'data' lines and establishes its line protocol. A bidirectional serial interface in the keyboard converts the 'clock' and 'data' signals and sends this information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

## Cabling

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



DIN Connector



SDL Connector

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

## Sequencing Key-Code Scanning

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

## Keyboard Buffer

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

A buffer-overflow condition occurs when more than 16 bytes are placed in the keyboard buffer. An overflow 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-overflow 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  $\pm 20\%$ , and begins sending a make code for that key at a rate of 10.9 characters per second  $\pm 20\%$ . The typematic rate and delay can be modified [see "Set Typematic Rate/Delay (Hex F3)" on page 4-11].

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



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

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

## **Power-On Routine**

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

### **Power-On Reset**

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

### **Basic Assurance Test**

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

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

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

## Commands from the System

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

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

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

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

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

### **Default Disable (Hex F5)**

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

### **Echo (Hex EE)**

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

### **Enable (Hex F4)**

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

### **Invalid Command (Hex EF and F1)**

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

### **Read ID (Hex F2)**

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

## **Resend (Hex FE)**

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

## **Reset (Hex FF)**

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

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

## **Select Alternate Scan Codes (Hex F0)**

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

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

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

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

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

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

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

## Set Default (Hex F6)

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

## Set Key Type (Hex FB, FC, FD)

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

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

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

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

### **Set/Reset Status Indicators (Hex ED)**

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

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

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

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

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

## Set Typematic Rate/Delay (Hex F3)

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

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

$$\text{Period} = (8 + A) \times (2^B) \times 0.00417 \text{ seconds.}$$

where:

A = binary value of bits 2, 1, and 0.

B = binary value of bits 4 and 3.

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

Bit	Typematic Rate $\pm 20\%$	Bit	Typematic Rate $\pm 20\%$
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  $\pm 20\%$ .

Delay = 500 milliseconds  $\pm 20\%$ .

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



## Commands to the System

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

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

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

### Acknowledge (Hex FA)

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

### BAT Completion Code (Hex AA)

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

### BAT Failure Code (Hex FC)

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

### Echo (Hex EE)

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

### **Keyboard ID (Hex 83AB)**

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

### **Key Detection Error (Hex 00 or FF)**

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

### **Overrun (Hex 00 or FF)**

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

### **Resend (Hex FE)**

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

## Keyboard Scan Codes

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

## Scan Code Set 1

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

### Scan Code Tables (Set 1)

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

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

\* 101-key keyboard only.

Key Number	Make Code	Break Code
34	21	A1
35	22	A2
36	23	A3
37	24	A4
38	25	A5
39	26	A6
40	27	A7
41	28	A8
42 **	2B	AB
43	1C	9C
44	2A	AA
45 **	56	D6
46	2C	AC
47	2D	AD
48	2E	AE
49	2F	AF
50	30	B0
51	31	B1
52	32	B2
53	33	B3
54	34	B4
55	35	B5
57	36	B6
58	1D	9D
60	38	B8
61	39	B9
62	E0 38	E0 B8
64	E0 1D	E0 9D
90	45	C5
91	47	C7
92	4B	CB
93	4F	CF
96	48	C8
97	4C	CC
98	50	DO
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	CO
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 4-40 to determine the character associated with each key number.

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

\* 101-key keyboard only.



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

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

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

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

SECTION 4

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

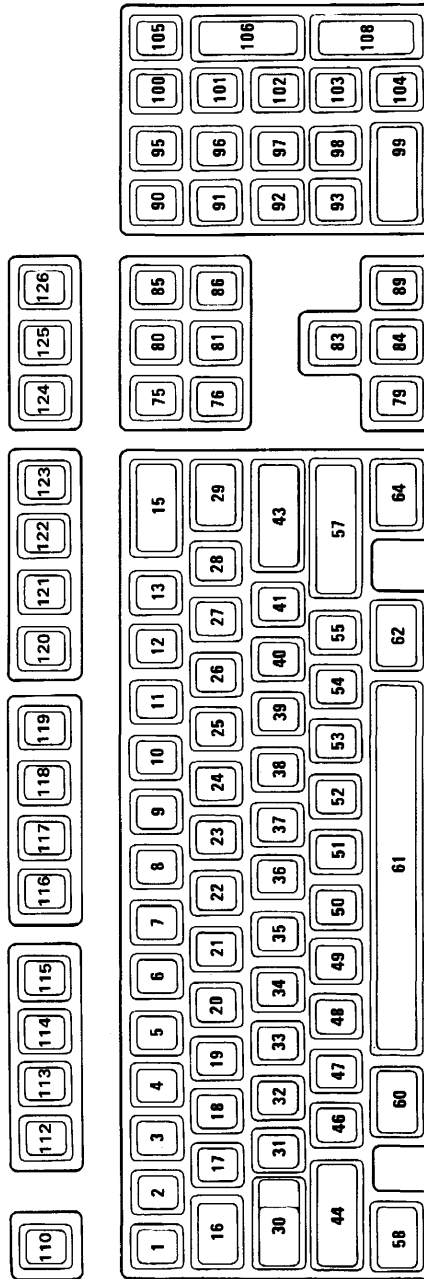
The keyboard routine, provided by IBM in the ROM BIOS, is responsible for converting the keyboard scan codes into what will be termed *Extended ASCII*. The extended ASCII codes returned by the ROM routine are mapped to the U.S. English keyboard layout. Some operating systems may make provisions for alternate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



SECTION 4

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
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
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.

SECTION 4

**Character Codes (Part 2 of 2)**



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)

SECTION 4

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.

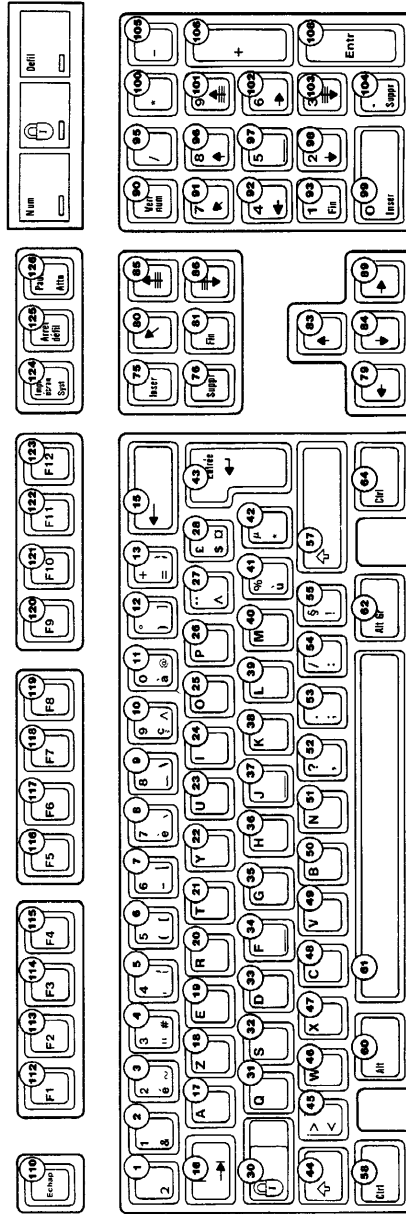
## **Keyboard Layouts**

The keyboard is available in six layouts:

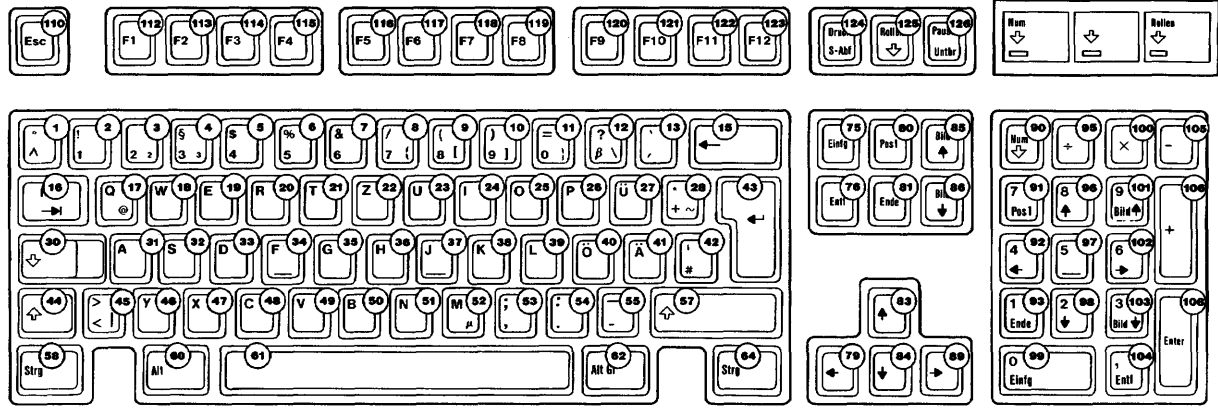
- French
- German
- Italian
- Spanish
- U.K. English
- U.S. English

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



SECTION 4



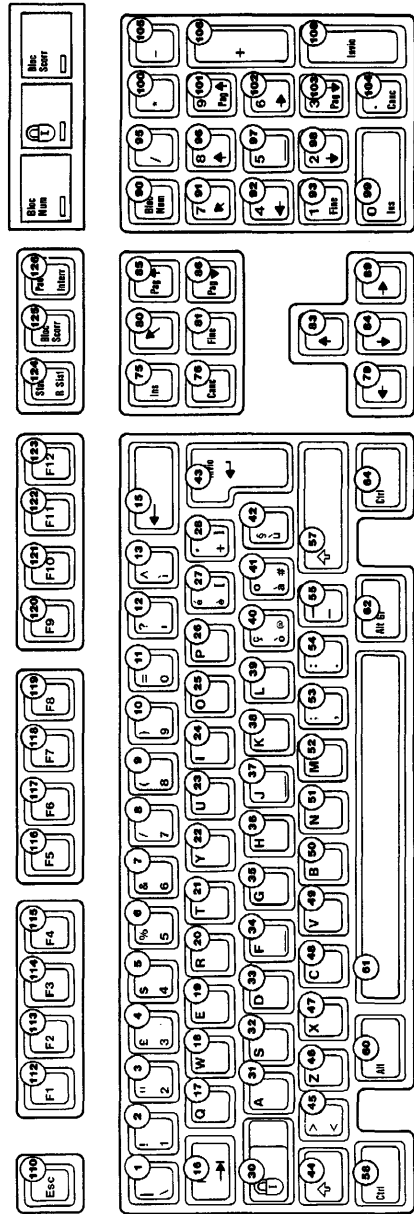
German Keyboard

(

(

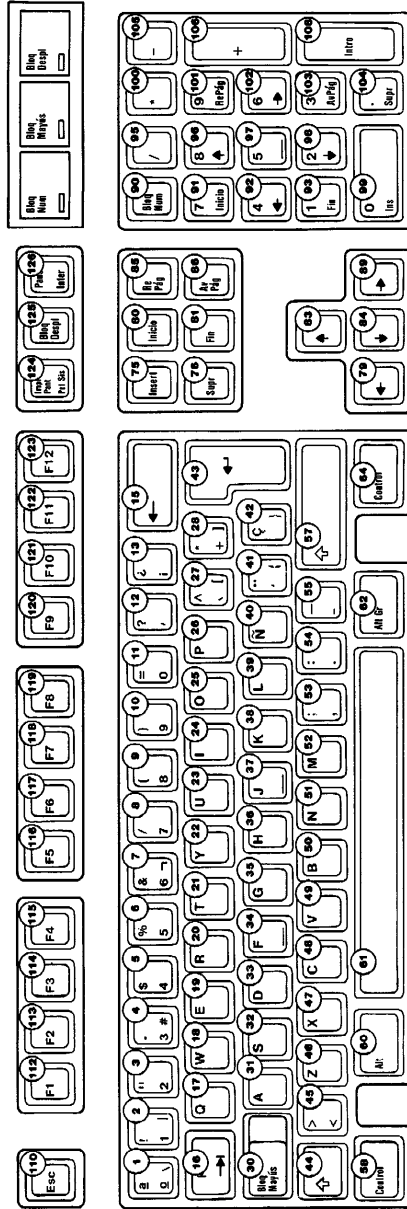
(

# Italian Keyboard



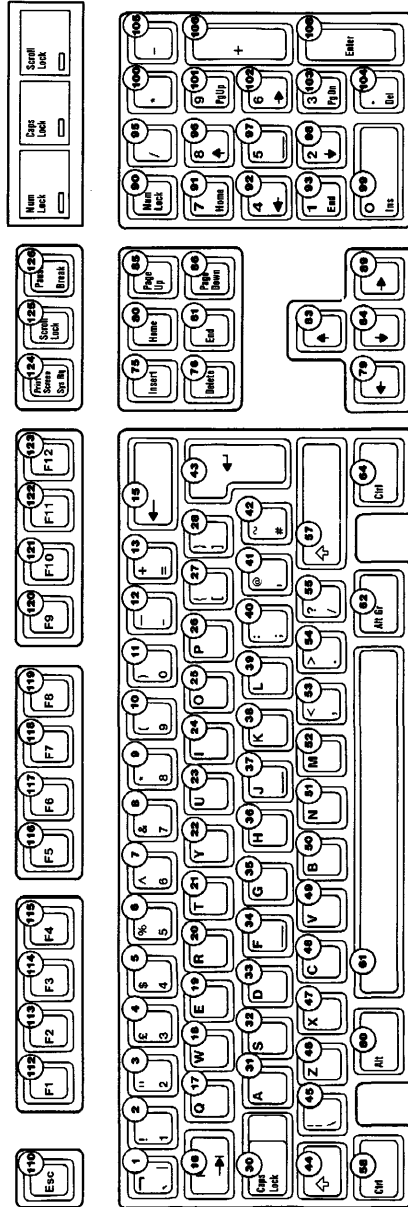
SECTION 4

# Spanish Keyboard



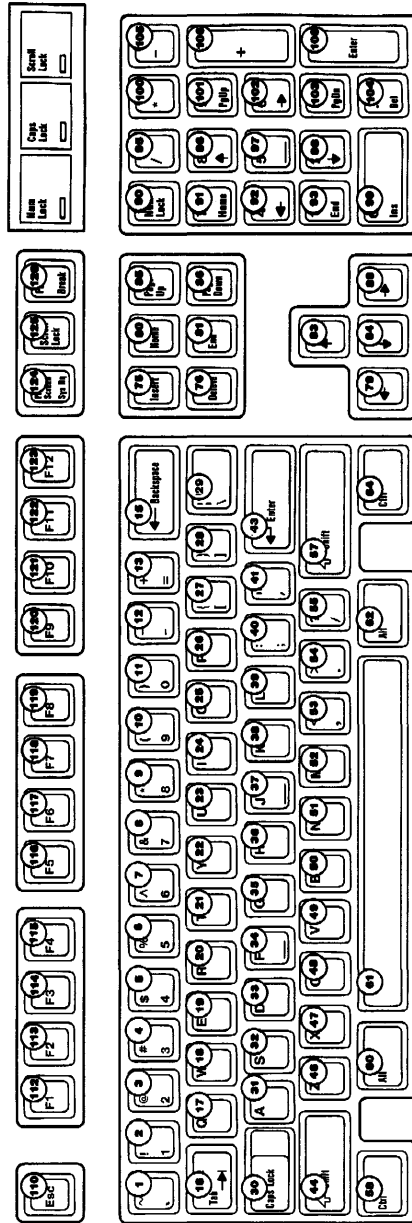


# U.K. English Keyboard



SECTION 4

# U.S. English Keyboard



## **Specifications**

The specifications for the keyboard are as follows.

### **Power Requirements**

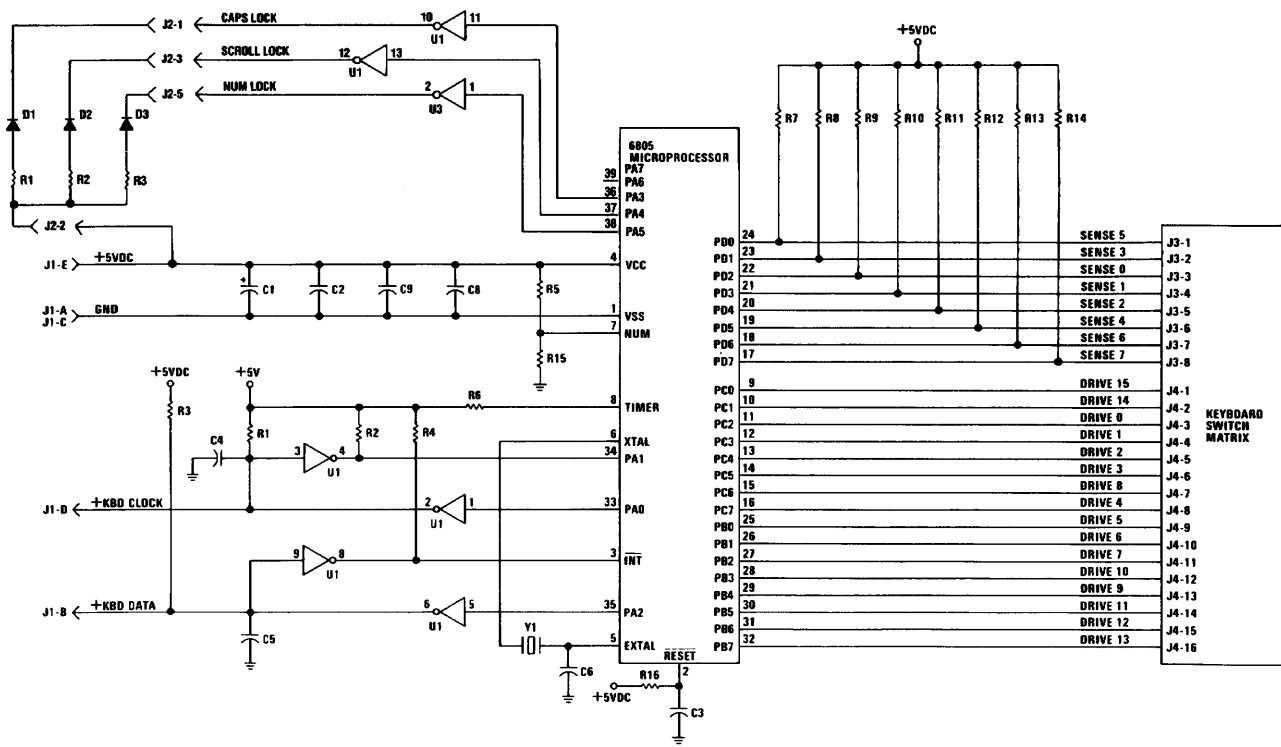
- +5 Vdc  $\pm$  10%
- Current cannot exceed 275 mA

### **Size**

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

### **Weight**

2.25 kilograms (5.0 pounds)



101/102-KEY KEYBOARD

## SECTION 5. SYSTEM BIOS

System BIOS Usage .....	5-3
Parameter Passing .....	5-4
Vectors with Special Meanings .....	5-6
Other Read/Write Memory Usage .....	5-9
BIOS Programming Hints .....	5-10
Adapters with System-Accessible ROM Modules .....	5-12
Quick Reference .....	5-14

## Notes:

## System BIOS Usage

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

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

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

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

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

## Parameter Passing

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

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

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

To read the time of day:

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

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



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

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

**80286-2 Program Interrupt Listing (Real Mode Only)**

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

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

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

#### Hardware, Basic, and DOS Interrupts

### Vectors with Special Meanings

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

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

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

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

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

This routine may retain control with the following considerations:

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

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

---

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

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

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

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

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

## Other Read/Write Memory Usage

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

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

The following figure shows the reserved memory locations.

Address	Mode	Function
400-4A1 4A2-4EF 4F0-4FF	ROM BIOS	See BIOS listing Reserved Reserved as intra-application communication area for any application
500-5FF 500	DOS	Reserved for DOS and BASIC 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**

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

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

### Basic Workspace Variables

#### Example

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

L	H
Hex 64	Hex 00

The following is a BIOS memory map.

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

### BIOS Memory Map

### BIOS Programming Hints

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

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

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

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

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

### Move Block BIOS

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

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

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

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

### Communication Interrupt Intervals

---

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

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

#### **Move Block BIOS Timing**

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

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

#### **Adapters with System-Accessible ROM Modules**

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

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

**Byte 0** Hex 55  
**Byte 1** Hex AA



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

**Byte 3** Entry by a CALL FAR

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

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

## Quick Reference

<b>BIOS MAP</b> .....	<b>5-16</b>
<b>Test1</b> .....	<b>5-18</b>
Data Area Description .....	5-20
Common POST and BIOS Equates .....	5-22
Test .01 Through Test .16 .....	5-27
POST and Manufacturing Test Routines .....	5-48
<b>Test2</b> .....	<b>5-49</b>
Test .17 Through Test .23 .....	5-49
<b>Test3. POST Exception Interrupt Tests</b> .....	<b>5-66</b>
<b>Test4. POST and BIOS Utility Routines</b> .....	<b>5-72</b>
CMOS_READ .....	5-72
CMOS_WRITE .....	5-72
E_MSG P_MSG .....	5-73
ERR_BEEP .....	5-73
BEEP .....	5-74
WAITF .....	5-74
CONFIG_BAD .....	5-74
PRT_SEG .....	5-75
KBD_RESET .....	5-75
D11 - Dummy Interrupt Handler .....	5-78
Hardware Interrupt 9 Handler (Type 71) .....	5-78
<b>Test5. Exception Interrupt Tests</b> .....	<b>5-79</b>
SYSINIT1 - Build Protected Mode Descriptors .....	5-80
GDT_BLD - Build the GDT for POST .....	5-80
SIDT_BLD - Build the IDT for POST .....	5-81
<b>Test6</b> .....	<b>5-84</b>
STGTST_CNT .....	5-84
ROM_ERR .....	5-86
XMIT_8042 .....	5-86
BOOT_STRAP .....	5-86
<b>Diskette BIOS</b> .....	<b>5-88</b>
<b>Fixed Disk BIOS</b> .....	<b>5-114</b>

<b>Keyboard BIOS</b> .....	<b>5-126</b>
<b>Printer BIOS</b> .....	<b>5-139</b>
<b>RS232 BIOS</b> .....	<b>5-141</b>
<b>Video BIOS</b> .....	<b>5-144</b>
<b>BIOS</b> .....	<b>5-162</b>
Memory Size Determine .....	5-162
Equipment Determine .....	5-162
NMI .....	5-163
<b>BIOS1.</b> .....	<b>5-164</b>
Event Wait .....	5-165
Joystick Support .....	5-166
Wait .....	5-167
Block Move .....	5-168
Extended Memory Size Determine .....	5-173
Processor to Virtual Mode .....	5-174
Configuration Parameters .....	5-174
<b>BIOS2</b> .....	<b>5-177</b>
Time of Day .....	5-177
Alarm Interrupt Handler .....	5-180
Print Screen .....	5-181
Timer 1 Interrupt Handler .....	5-182
<b>ORGS - PC Compatibility and Tables</b> .....	<b>5-183</b>
POST Error Messages .....	5-183

Address	Publics by Name	Address	Publics by Value
F000:E729	A1	F000:0000	POST1
F000:3DE6	ACT_DISP_PAGE	F000:0008	K6L
F000:16000	BASIC	F000:0010	M4
F000:1A04	SEEP	F000:0050	START_1
F000:1B2E	BLINK_INT	F000:1039D	C8042_
F000:2038	BOOT_STRAP_1	F000:103A9	OBP_42
F000:10CA3	C21	F000:10CA3	C21
F000:039D	C8042	F000:10CA3	POST2
F000:4503	CASSETTE_IO_1	F000:105F	SHUT3
F000:194F	CMOS_READ	F000:10C3	SHUT2
F000:196B	CMOS_WRITE	F000:10C6	SHUT7
F000:1A59	CONFIG_BAD	F000:10E7	SHUT6
F000:E6F5	CONF_TBL	F000:1620	SHUT4
F000:FA6E	CRT_CHAR_GEN	F000:167F	POST3
F000:1E020	D1	F000:194F	CMOS_READ
F000:1BE0	D11	F000:194F	POST4
F000:E030	D2	F000:196B	CMOS_WRITE
F000:E040	D2A	F000:1989	DDS
F000:1989	DDS	F000:1991	E_MSG
F000:12159	DISKETTE_IO_1	F000:1988	F_MSG
F000:EF7C7	DISK_BASE	F000:19C6	ERR_BEEP
F000:12C72	DISK_INT_1	F000:1A04	BEEP
F000:2E86	DISK_IO	F000:1A4A	WAITF
F000:2CDD0	DISK_SETUP	F000:1A59	CONFIG_BAD
F000:2C89	DISKETTE_SETUP	F000:1A6D	XPC_BYTE
F000:FF53	DUMMY_RETURN	F000:1A70	PRT_HEX
F000:1C2E	DUMMY_RETURN_1	F000:1A84	PRT_SEG
F000:E095E	E10	F000:1A99	PROT_PRT_HEX
F000:E077	E102	F000:1AC5	ROM_CHECKSUM
F000:E090	E103	F000:1AD1	ROM_CHECK
F000:E0A9	E104	F000:1B03	KBD_RESET
F000:E0C2	E105	F000:1B2E	BLINK_INT
F000:E0DB	E106	F000:1B3C	SET_T00
F000:E0F4	E107	F000:1BE0	D11
F000:E10D	E108	F000:1C2E	DUMMY_RETURN_1
F000:E126	E109	F000:1C2F	RE_DIRECT
F000:E13F	E161	F000:1C38	INT_287
F000:E168	E162	F000:1C47	PROC_SHUTDOWN
F000:E191	E163	F000:1C4E	POST5
F000:E1B7	E164	F000:1D40	SYSDINITI
F000:E1DB	E201	F000:1ECB	POST6
F000:1EE	E202	F000:1ECB	STATS_CNT
F000:E209	E203	F000:1FCB	ROM_ERR
F000:E224	E301	F000:1FF7	XMIT_8042
F000:E239	E302	F000:2038	BOOT_STRAP_1
F000:E2C6	E303	F000:2159	DISKETTE_IO_1
F000:E2EA	E304	F000:2B10	SEEK
F000:E30E	E401	F000:2C72	DISK_INT_1
F000:E31E	E501	F000:2C89	DISKETTE_SETUP
F000:E35E	E601	F000:2CDD0	DISK_SETUP
F000:E343	E602	F000:2E86	DISK_IO
F000:4475	EQUIPMENT_1	F000:33B7	HD_INT
F000:19C6	ERR_BEEP	F000:33DA	KEYBOARD_IO_1
F000:1991	E_MSG	F000:354F	KB_INT_1
F000:E364	FT780	F000:35AE	K15
F000:E379	F1781	F000:3A25	SND_DATA
F000:E38E	F1782	F000:3A60	PRINTER_IO_1
F000:E3AC	F1790	F000:3B4F	RS232_IO_1
F000:E3BF	F1791	F000:3C64	VIDEO_IO_1
F000:E3D0	F3A	F000:3C9B	SET_MODE
F000:E3D5	F3D	F000:3DA2	SET_CTYPE
F000:E3DF	F3D1	F000:3DA7	SET_CPOS
F000:E401	FD_TBL	F000:3DC7	READ_CURSOR
F000:4C76	FID	F000:3DE6	ACT_DISP_PAGE
F000:FF5E	FLOPPY	F000:3E08	SET_COLOR
F000:48DD	GATE_A20	F000:3E2E	VIDEO_STATE
F000:32B7	HD_INT	F000:3E4F	SCROLL_UP
F000:FF5A	HD_INT	F000:3EED0	SCROLL_DOWN
F000:1C38	INT_287	F000:3F3F	READ_AC_CURRENT
F000:E8E6	K10	F000:3F9C	WRITE_AC_CURRENT
F000:E38A	K11	F000:3FCE	WRITE_C_CURRENT
F000:E3C4	K12	F000:407F	READ_DOT
F000:E900	K14	F000:4090	WRITE_DOT
F000:E92C	K15	F000:443E	WRITE_TTY
F000:35AE	K16	F000:443C5	READ_LPEN
F000:E87E	K6	F000:446B	MEMORY_SIZE_DET_1
F000:0008	K6L	F000:4475	EQUIPMENT_1
F000:E886	K7	F000:447F	NMI_INT_1
F000:E88E	K8	F000:4503	CASSETTE_IO_1
F000:1B03	KBD_RESET	F000:4799	SHUT9
F000:354F	KB_INT_1	F000:48DD	GATE_A20
F000:33DA	KEYBOARD_IO_1	F000:4999	TIME_OF_DAY_1
F000:0010	M4	F000:481B	RTC_INT
F000:F0E4	M5	F000:4897	PRINT_SCREEN_1
F000:F0EC	M6	F000:4C2D	TIMER_INT_1
F000:F0F4	MT	F000:4C76	FILL
F000:446B	MEMORY_SIZE_DET_1	F000:1600	BASIC
F000:E2C3	NMI_INT	F000:E020	D1
F000:447F	NMI_INT_1	F000:E030	D2
F000:08A9	OBP_42	F000:E040	D2A
F000:10000	POST1	F000:E05E	E101
F000:10CA2	POST2	F000:E077	E102
F000:167F	POST3	F000:E090	E103
F000:194F	POST4	F000:E095E	E104
F000:1C4E	POST5	F000:E0C2	E105
F000:1ECB	POST6	F000:E0DB	E106
F000:3AC0	PRINTER_IO_1	F000:E0F4	E107
F000:FF54	PRINT_SCREEN	F000:E10D	E108
F000:4B97	PRINT_SCREEN_1	F000:E126	E109
F000:1C47	PROC_SHUTDOWN	F000:E13F	E161
F000:1A99	PROT_PRT_HEX	F000:E168	E162
F000:1A7D	PRT_HEX	F000:E191	E163
F000:1A84	PRT_SEG	F000:E1B7	E164
F000:1988	P_MSG	F000:E1DB	E201
F000:FF70	P_O_R	F000:1EE	E202
F000:3F3F	READ_AC_CURRENT	F000:E209	E203
F000:3DC7	READ_CURSOR	F000:E224	E301
F000:407F	READ_DOT	F000:E239	E302
F000:43C6	READ_LPEN	F000:E250	F3D
F000:1C2F	RE_DIRECT	F000:E2C3	NMI_INT
F000:1AD1	ROM_CHECK	F000:E2C6	E303
F000:1AC5	ROM_CHECKSUM	F000:E2EA	E304
F000:1FCB	ROM_ERR	F000:E30E	E401
F000:3B4F	RS232_IO_1	F000:E31E	E501
F000:481B	RTC_INT	F000:E32E	E601
F000:3EED0	SCROLL_DOWN	F000:E343	E602
F000:3E4F	SCROLL_UP	F000:E364	F1780

F000:2B1D	SEEK	F000:E379	F1781
F000:FF62	SEEKS_I	F000:E38E	F1782
F000:3E08	SET_COLOR	F000:E3AC	F1790
F000:3DA7	SET_CPOS	F000:E3BF	F1791
F000:3D82	SET_CTYPE	F000:E3D2	F3A
F000:3C98	SET_MODE	F000:E3DF	F3D1
F000:1B3C	SET_TOD	F000:E401	FD_TBL
F000:10C3	SHUT2	F000:E6F6	CONF_TBL
F000:105F	SHUT3	F000:E729	A1
F000:1620	SHUT4	F000:E87E	K6
F000:10E7	SHUT6	F000:E886	K7
F000:10C6	SHUT7	F000:E88E	K8
F000:4799	SHUT9	F000:E8E6	K10
F000:FF23	SLAVE_VECTOR_TABLE	F000:E92C	K15
F000:3A25	SND_DATA	F000:E98A	K11
F000:0050	START_I	F000:E9C4	K12
F000:1EC8	SYSTST_CNT	F000:E9D0	K14
F000:1D40	SYSINITI	F000:EFC7	DISK_BASE
F000:4C2D	TIMER_INT_I	F000:F0A4	VIDEO_PARMS
F000:4999	TIME_OF_DAY_I	F000:F0E4	M6
F000:FF64	TUTOR	F000:F0EC	M6
F000:FEF3	VECTOR_TABLE	F000:F0F4	M7
F000:3C64	VIDEO_TO_I	F000:FA6E	CRT_CHAR_GEN
F000:F0A4	VIDEO_PARMS	F000:FEF3	VECTOR_TABLE
F000:3E2E	VIDEO_STATE	F000:FF23	SLAVE_VECTOR_TABLE
F000:1A4A	WAITI	F000:FF53	DUMMY_RETURN
F000:3F9C	WRITE_AC_CURRENT	F000:FF54	PRINT_SCREEN
F000:3FC6	WRITE_C_CURRENT	F000:FF5A	HRD
F000:4090	WRITE_DOT	F000:FF5E	FLOPPY
F000:433E	WRITE_TTY	F000:FF62	SEEKS_I
F000:1FF7	XMIT_B042	F000:FF66	TUTOR
F000:1A6D	XPC_BYTE	F000:FFFF	P_D_R

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

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110

PAGE 118,121  
TITLE TEST1 ---- 04/21/86 POWER ON SELF TEST (POST)  
.286C

-----  
BIOS I/O INTERFACE

THESE LISTINGS PROVIDE INTERFACE INFORMATION FOR ACCESSING  
THE BIOS ROUTINES. THE POWER ON SELF TEST IS INCLUDED.

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

-----  
MODULE REFERENCE

TEST1.ASM --> POST AND MANUFACTURING TEST ROUTINES  
DSEG.INC --> DATA SEGMENTS LOCATIONS  
POSTEQI.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\_SCAN - ROM SCAN AND INITIALIZE  
KBD\_RESET - POST KEYBOARD RESET ROUTINE  
BLINK\_INT - MANUFACTURING TOGGLE BIT ROUTINE  
SET\_TOD - SET TIMER FROM CMOS RTC  
DUM\_INT - DUMMY INTERRUPT HANDLER ->INT 77H  
RE\_DIRECT - HARDWARE INT 9 REDIRECT (L 2)  
INT\_287 - HARDWARE INT 13 REDIRECT (287)  
PROC\_SHUTDOWN - 80286 RESET ROUTINE  
TEST5.ASM --> EXCEPTION INTERRUPT TEST HANDLERS FOR POST TESTS  
SYSINIT1 - BUILD PROTECTED MODE POINTERS  
GDT\_BLD - BUILD THE GDT FOR POST  
SIDT\_BLD - BUILD THE IDT FOR POST  
TEST6.ASM --> POST TESTS AND SYSTEM BOOT STRAP  
STGTS\_CNT - SEGMENT STORAGE TEST  
ROM\_ERR - ROM ERROR DISPLAY ROUTINE  
XMIY\_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  
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  
PROC\_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  
DEVCE\_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  
TIMER1 - TIMER1 INTERRUPT HANDLER ->INT 1CH  
ORGS.ASM --> COMPATIBILITY MODULE  
POST\_ERROR\_MESSAGES  
DISKETTE - DISK - VIDEO DATA TABLES

-----  
.LIST

```

111 PAGE
112 INCLUDE DSEG.INC
113 -----
114 | 80286 INTERRUPT LOCATIONS |
115 | REFERENCED BY POST & BIOS |
116 |-----|
117
118 0000 ABS0 SEGMENT AT 0 ; ADDRESS= 0000:0000
119
120 0000 ?? @STG_LOCO DB ? ; START OF INTERRUPT VECTOR TABLE
121
122 0008 @NMI_PTR ORG 4*002H
123 0008 ????????? DD ? ; NON-MASKABLE INTERRUPT VECTOR
124
125 0014 @INT5_PTR ORG 4*005H
126 0014 ????????? DD ? ; PRINT SCREEN INTERRUPT VECTOR
127
128 0020 @INT6_PTR ORG 4*008H
129 0020 ????????? DD ? ; HARDWARE INTERRUPT POINTER (8-F)
130
131 0040 @VIDEO_INT ORG 4*010H
132 0040 ????????? DD ? ; VIDEO I/O INTERRUPT VECTOR
133
134 004C @DRG_VECTOR ORG 4*013H
135 004C ????????? DD ? ; DISKETTE/DISK INTERRUPT VECTOR
136
137 0060 @BASIC_PTR ORG 4*018H
138 0060 ????????? DD ? ; POINTER TO CASSETTE BASIC
139
140 0074 @FARM_PTR ORG 4*01DH
141 0074 ????????? DD ? ; POINTER TO VIDEO PARAMETERS
142
143 0078 @DISK_POINTER ORG 4*01EH
144 0078 ????????? DD ? ; POINTER TO DISKETTE PARAMETER TABLE
145
146 007C @EXT_PTR ORG 4*01FH
147 007C ????????? DD ? ; POINTER TO GRAPHIC CHARACTERS 128-255
148
149 0100 @DISK_VECTOR ORG 4*040H
150 0100 ????????? DD ? ; POINTER TO DISKETTE INTERRUPT CODE
151
152 0104 @HF1_TBL_VEC ORG 4*041H
153 0104 ????????? DD ? ; POINTER TO FIRST DISK PARAMETER TABLE
154
155 0118 @HF1_TBL_VEC ORG 4*046H
156 0118 ????????? DD ? ; POINTER TO SECOND DISK PARAMETER TABLE
157
158 01C0 @SLAVE_INT_PTR ORG 4*070H
159 01C0 ????????? DD ? ; POINTER TO SLAVE INTERRUPT HANDLER
160
161 01D8 @HDISK_INT ORG 4*076H
162 01D8 ????????? DD ? ; POINTER TO FIXED DISK INTERRUPT CODE
163
164 0400 @TOS ORG 0400H
165 0400 ???? DW ? ; STACK -- USED DURING POST ONLY
166 ; USE WILL OVERLAY INTERRUPTS VECTORS
167
168 0500 @MFG_TEST_RTN ORG 0500H
169 0500 FAR ; LOAD LOCATION FOR MANUFACTURING TESTS
170
171 7C00 @BOOT_LOCN ORG 7C00H
172 7C00 FAR ; BOOT STRAP CODE LOAD LOCATION
173
174 ABS0 ENDS
    
```

SECTION 5

```

175          C PAGE
176          C |-----|
177          C |          ROM BIOS DATA AREAS          |
178          C |-----|
179          C DATA          SEGMENT AT 40H          | ADDRESS= 0040:0000
180          C
181          C
182          C #RS232_BASE    DW    ?          | BASE ADDRESSES OF RS232 ADAPTERS
183          C #RS232_BASE    DW    ?          | SECOND LOGICAL RS232 ADAPTER
184          C #RS232_BASE    DW    ?          | RESERVED
185          C #RS232_BASE    DW    ?          | RESERVED
186          C #PRINTER_BASE  DW    ?          | BASE ADDRESSES OF PRINTER ADAPTERS
187          C #PRINTER_BASE  DW    ?          | SECOND LOGICAL PRINTER ADAPTER
188          C #PRINTER_BASE  DW    ?          | THIRD LOGICAL PRINTER ADAPTER
189          C #PRINTER_BASE  DW    ?          | RESERVED
190          C #EQUIP_FLAG     DW    ?          | INSTALLED HARDWARE FLAGS
191          C #MFG_TST        DB    ?          | INITIALIZATION FLAGS
192          C #MEMORY_SIZE    DW    ?          | BASE MEMORY SIZE IN K BYTES (X 1024)
193          C #MFG_ERR_FLAG   DB    ?          | SCRATCHPAD FOR MANUFACTURING
194          C #MFG_ERR_FLAG   DB    ?          | ERROR CODES
195          C
196          C |-----|
197          C |          KEYBOARD DATA AREAS          |
198          C |-----|
199          C
200          C #KB_FLAG         DB    ?          | KEYBOARD SHIFT STATE AND STATUS FLAGS
201          C #KB_FLAG_1     DB    ?          | SECOND BYTE OF KEYBOARD STATUS
202          C #ALT_INPUT     DB    ?          | STORAGE FOR ALTERNATE KEY PAD ENTRY
203          C #BUFFER_HEAD   DW    ?          | POINTER TO HEAD OF KEYBOARD BUFFER
204          C #BUFFER_TAIL   DW    ?          | POINTER TO TAIL OF KEYBOARD BUFFER
205          C
206          C |----- HEAD = TAIL INDICATES THAT THE BUFFER IS EMPTY
207          C
208          C #KB_BUFFER      DW    16 DUP(?)    | ROOM FOR 15 SCAN CODE ENTRIES
209          C
210          C
211          C
212          C |-----|
213          C |          DISKETTE DATA AREAS          |
214          C |-----|
215          C
216          C #SEEK_STATUS     DB    ?          | DRIVE RECALIBRATION STATUS
217          C #SEEK_STATUS     DB    ?          | BIT 3-0 = DRIVE 3-0 RECALIBRATION
218          C #SEEK_STATUS     DB    ?          | BEFORE NEXT SEEK IF BIT 15 = 0
219          C #MOTOR_STATUS  DB    ?          | MOTOR STATUS
220          C #MOTOR_STATUS  DB    ?          | BIT 3-0 = DRIVE 3-0 CURRENTLY RUNNING
221          C #MOTOR_STATUS  DB    ?          | BIT 7 = CURRENT OPERATION IS A WRITE
222          C #MOTOR_COUNT   DB    ?          | TIME OUT COUNTER FOR MOTOR(S) TURN OFF
223          C #DSKETTE_STATUS DB    ?          | RETURN CODE STATUS BYTE
224          C #DSKETTE_STATUS DB    ?          | CMD_BLOCK IN STACK FOR DISK OPERATION
225          C #NEC_STATUS     DB    7 DUP(?)    | STATUS BYTES FROM DISKETTE OPERATION
226          C
227          C
228          C
229          C
230          C |-----|
231          C |          VIDEO DISPLAY DATA AREA        |
232          C |-----|
233          C
234          C #CRT_MODE         DB    ?          | CURRENT DISPLAY MODE (TYPE)
235          C #CRT_COLS       DW    ?          | NUMBER OF COLUMNS ON SCREEN
236          C #CRT_LEN        DW    ?          | LENGTH OF REGEN BUFFER IN BYTES
237          C #CRT_START      DW    ?          | STARTING ADDRESS IN REGEN BUFFER
238          C #CURSOR_POSN    DW    8 DUP(?)    | CURSOR FOR EACH OF UP TO 8 PAGES
239          C
240          C
241          C
242          C #CURSOR_MODE     DW    ?          | CURRENT CURSOR MODE SETTING
243          C #ACTIVE_PAGE    DW    ?          | CURRENT PAGE BEING DISPLAYED
244          C #ADDR_6845     DW    ?          | BASE ADDRESS FOR ACTIVE DISPLAY CARD
245          C #CRT_MODE_SET  DB    ?          | CURRENT SETTING OF THE 3x8 REGISTER
246          C #CRT_PALETTE   DB    ?          | CURRENT PALETTE SETTING - COLOR CARD
247          C
248          C
249          C |-----|
250          C |          POST AND BIOS WORK DATA AREA  |
251          C |-----|
252          C
253          C #IO_ROM_INIT     DW    ?          | STACK SAVE, etc.
254          C #IO_ROM_SEG    DW    ?          | POINTER TO ROM INITIALIZATION ROUTINE
255          C #INTR_FLAG     DB    ?          | LENGTH OF REGEN BUFFER IN BYTES
256          C #INTR_FLAG     DB    ?          | FLAG INDICATING AN INTERRUPT HAPPENED
257          C
258          C |-----|
259          C |          TIMER DATA AREA                |
260          C |-----|
261          C
262          C #TIMER_LOW        DW    ?          | LOW WORD OF TIMER COUNT
263          C #TIMER_HIGH    DW    ?          | HIGH WORD OF TIMER COUNT
264          C #TIMER_OFL     DB    ?          | TIMER HAS ROLLED OVER SINCE LAST READ
265          C
266          C |-----|
267          C |          SYSTEM DATA AREA                |
268          C |-----|
269          C
270          C #BIOS_BREAK       DB    ?          | BIT 7=1 IF BREAK KEY HAS BEEN PRESSED
271          C #RESET_FLAG    DW    ?          | WORD=1234H IF KEYBOARD RESET UNDERWAY
272          C
273          C |-----|
274          C |          FIXED DISK DATA AREAS          |
275          C |-----|
276          C
277          C #DISK_STATUS1     DB    ?          | FIXED DISK STATUS
278          C #HF_NUM        DB    ?          | COUNT OF FIXED DISK DRIVES
279          C #CONTROL_BYTE  DB    ?          | HEAD CONTROL BYTE
280          C #PORT_OFF      DB    ?          | RESERVED (PORT OFFSET)

```



```

280 PAGE
281 |-----|
282 | TIME-OUT VARIABLES |
283 |-----|
284
285 0078 ?? @PRINT_TIM_OUT DB ? ; TIME OUT COUNTERS FOR PRINTER RESPONSE
286 0079 ?? DB ? ; SECOND LOGICAL PRINTER ADAPTER
287 007A ?? DB ? ; THIRD LOGICAL PRINTER ADAPTER
288 007B ?? DB ? ; RESERVED
289 007C ?? @RS232_TIM_OUT DB ? ; TIME OUT COUNTERS FOR RS232 RESPONSE
290 007D ?? DB ? ; SECOND LOGICAL RS232 ADAPTER
291 007E ?? DB ? ; RESERVED
292 007F ?? DB ? ; RESERVED
293
294 |-----|
295 | ADDITIONAL KEYBOARD DATA AREA |
296 |-----|
297
298
299 0080 ???? @BUFFER_START DW ? ; BUFFER LOCATION WITHIN SEGMENT 40H
300 0082 ???? @BUFFER_END DW ? ; OFFSET OF KEYBOARD BUFFER START
301 ; OFFSET OF END OF BUFFER
302
303 |-----|
304 | EGA/PGA DISPLAY WORK AREA |
305 |-----|
306
307 0084 ?? @ROWS DB ? ; ROWS ON THE ACTIVE SCREEN (LESS 1)
308 0085 ???? @POINTS DB ? ; BYTES PER CHARACTER
309 0086 ?? @INFO DB ? ; MODE OPTIONS
310 0087 ?? @INFO_3 DB ? ; FEATURE BIT SWITCHES
311 0088 ?? DB ? ; RESERVED FOR DISPLAY ADAPTERS
312 0089 ?? DB ? ; RESERVED FOR DISPLAY ADAPTERS
313
314 |-----|
315 | ADDITIONAL MEDIA DATA |
316 |-----|
317
317 008B ?? @LASTRATE DB ? ; LAST DISKETTE DATA RATE SELECTED
318 008C ?? @F_STATUS DB ? ; STATUS REGISTER
319 008D ?? @F_ERROR DB ? ; ERROR REGISTER
320 008E ?? @F_INT_FLAG DB ? ; FIXED DISK INTERRUPT FLAG
321 008F ?? @F_CNTRL DB ? ; COMBO FIXED DISK/DISKETTE CARD BIT 0=1
322 0090 ?? @DSK_STATE DB ? ; DRIVE 0 MEDIA STATE
323 0091 ?? DB ? ; DRIVE 1 MEDIA STATE
324 0092 ?? DB ? ; DRIVE 0 OPERATION START STATE
325 0093 ?? DB ? ; DRIVE 1 OPERATION START STATE
326 0094 ?? @DSK_TRK DB ? ; DRIVE 0 PRESENT CYLINDER
327 0095 ?? DB ? ; DRIVE 1 PRESENT CYLINDER
328
329 |-----|
330 | ADDITIONAL KEYBOARD FLAGS |
331 |-----|
332
333 0096 ?? @KB_FLAG_3 DB ? ; KEYBOARD MODE STATE AND TYPE FLAGS
334 0097 ?? @KB_FLAG_2 DB ? ; KEYBOARD LED FLAGS
335
336 |-----|
337 | REAL TIME CLOCK DATA AREA |
338 |-----|
339
340 0098 ???? @USER_FLAG DW ? ; OFFSET ADDRESS OF USERS WAIT FLAG
341 009A ???? @USER_FLAG_SEG DW ? ; SEGMENT ADDRESS OF USER WAIT FLAG
342 009C ???? @RTC_LOW DW ? ; LOW WORD OF USER WAIT FLAG
343 009E ???? @RTC_HIGH DW ? ; HIGH WORD OF USER WAIT FLAG
344 00A0 ?? @RTC_WAIT_FLAG DB ? ; WAIT ACTIVE FLAG (01=BUSY, 00=POSTED)
345 ; (00=POST ACKNOWLEDGED)
346
347 |-----|
348 | AREA FOR NETWORK ADAPTER |
349 |-----|
350
350 00A1 07 [ ?? ] @NET DB ? DUP(?) ; RESERVED FOR NETWORK ADAPTERS
351
352 |-----|
353 | EGA/PGA PALETTE POINTER |
354 |-----|
355
356
357 00AB ?????????? @SAVE_PTR DD ? ; POINTER TO EGA PARAMETER CONTROL BLOCK
358 ; RESERVED
359
360 |-----|
361 | DATA AREA - PRINT SCREEN |
362 |-----|
363
364
365 0100 ORG 100H ; ADDRESS= 0040:0100 (REF 0050:0000)
366
367 0100 ?? @STATUS_BYTE DB ? ; PRINT SCREEN STATUS BYTE
368 ; 00=READY/DK, 01=BUSY, FF=ERROR
369
370 0101 DATA ENDS ; END OF BIOS DATA SEGMENT
371
372 .LIST
    
```

SECTION 5

```

373 PAGE
374 INCLUDE POSTEQU.INC
375
376 -----
377 EQUATES USED BY POST AND BIOS
378 -----
379
380 = 00FC MODEL_BYTE EQU 0FCH ; SYSTEM MODEL BYTE
381 = 0002 SUB_MODEL_BYTE EQU 002H ; SYSTEM SUB-MODEL TYPE
382 = 0000 BIOS_LEVEL EQU 000H ; BIOS REVISION LEVEL
383 = F780 RATE_UPPER EQU 0F780H ; UPPER LIMIT + 13%
384 = F9FD RATE_LOWER EQU 0F9FDH ; LOWER LIMIT - 20%
385
386 ----- 8042 KEYBOARD INTERFACE AND DIAGNOSTIC CONTROL REGISTERS -----
387 PORT_A EQU 060H ; 8042 KEYBOARD SCAN CODE/CONTROL PORT
388 PORT_B EQU 061H ; PORT B READ/WRITE DIAGNOSTIC REGISTER
389 RAM_PAR_ON EQU 11110011B ; AND MASK FOR PARITY CHECKING ENABLE ON
390 RAM_PAR_OFF EQU 00001100B ; OR MASK FOR PARITY CHECKING ENABLE OFF
391 PARTY_ERR EQU 11000000B ; R/W MEMORY - I/O CHANNEL PARITY ERROR
392 GATE2 EQU 00000001B ; TIMER 2 INPUT GATE CLOCK BIT
393 SPK2 EQU 00000010B ; SPEAKER OUTPUT DATA ENABLE BIT
394 REFRESH_BIT EQU 00010000B ; REFRESH TEST BIT
395 OUT2 EQU 00100000B ; SPEAKER TIMER OUT2 INPUT BIT
396 IO_CHECK EQU 01000000B ; I/O (MEMORY) CHECK OCCURRED BIT MASK
397 PARITY_CHECK EQU 10000000B ; MEMORY PARITY CHECK OCCURRED BIT MASK
398 STATUS_PORT EQU 064H ; 8042 STATUS PORT
399 OUT_BUF_FULL EQU 00000001B ; 0 = OUTPUT BUFFER FULL
400 INPT_BUF_FULL EQU 00000010B ; 1 = INPUT BUFFER FULL
401 SYS_FLAG EQU 00000100B ; 2 = SYSTEM FLAG -POST/-SELF TEST
402 CMD_DATA EQU 00001000B ; 3 = COMMAND+DATA
403 YKBD_INH EQU 00010000B ; 4 = KEYBOARD INHIBITED
404 TRANS_TMOUT EQU 00100000B ; 5 = TRANSMIT TIMEOUT
405 RCV_TMOUT EQU 01000000B ; 6 = RECEIVE TIME OUT
406 PARTTY_EVEN EQU 10000000B ; 7 = PARITY IS EVEN
407
408 ----- 8042 INPUT PORT BIT DEFINITION SAVED IN @MFG TST -----
409 BASE_MEMB EQU 00001000B ; BASE PLANAR R/W MEMORY EXTENSION 640/X
410 BASE_MEM EQU 00010000B ; BASE PLANAR R/W MEMORY SIZE 256/512
411 MFG_LOOP EQU 00100000B ; LOOP POST JUMPER BIT FOR MANUFACTURING
412 DSP_JMP EQU 01000000B ; DISPLAY TYPE SWITCH JUMPER BIT
413 KEY_BD_INH1B EQU 10000000B ; KEYBOARD INHIBIT SWITCH BIT
414
415 ----- 8042 COMMANDS -----
416 WRITE_8042_LOC EQU 060H ; WRITE 8042 COMMAND BYTE
417 SELF_TEST EQU 0AAH ; 8042 SELF TEST
418 INTR_FACE_CK EQU 0ABH ; CHECK 8042 INTERFACE COMMAND
419 DIS_KBD EQU 0ADH ; DISABLE KEYBOARD COMMAND
420 ENA_KBD EQU 0AEH ; ENABLE KEYBOARD COMMAND
421 READ_8042_INPUT EQU 0C0H ; READ 8042 INPUT PORT
422 DISABLE_BIT20 EQU 0DDH ; DISABLE ADDRESS LINE BIT 20
423 ENABLE_BIT20 EQU 0DFH ; ENABLE ADDRESS LINE BIT 20
424 KYBD_CLK_DATA EQU 0E0H ; GET KEYBOARD CLOCK AND DATA COMMAND
425 SHUT_CMD EQU 0F2H ; CAUSE A SHUTDOWN COMMAND
426 KYBD_CLK EQU 001H ; KEYBOARD CLOCK BIT 0
427
428 ----- KEYBOARD/LED COMMANDS -----
429 KB_RESET EQU 0FFH ; SELF DIAGNOSTIC COMMAND
430 KB_RESEND EQU 0FEH ; RESEND COMMAND
431 KB_MAKE_BREAK EQU 0FAH ; TYPAMATIC COMMAND
432 KB_ENABLE EQU 0F4H ; KEYBOARD ENABLE
433 KB_TYPA_RD EQU 0F3H ; TYPAMATIC RATE/DELAY COMMAND
434 KB_READ_ID EQU 0F2H ; READ KEYBOARD ID COMMAND
435 KB_ECHO EQU 0EEH ; ECHO COMMAND
436 LED_CMD EQU 0EDH ; LED WRITE COMMAND
437
438 ----- 8042 KEYBOARD RESPONSE -----
439 KB_OVER_RUN EQU 0FFH ; OVER RUN SCAN CODE
440 KB_RESEND EQU 0FEH ; RESEND REQUEST
441 KB_ACK EQU 0FAH ; ACKNOWLEDGE FROM TRANSMISSION
442 KB_BREAK EQU 0FDH ; KEYBOARD BREAK CODE
443 KB_OK EQU 0AAH ; RESPONSE FROM SELF DIAGNOSTIC
444
445 ----- KEYBOARD SCAN CODES -----
446 NUM_KEY EQU 69 ; SCAN CODE FOR NUMBER LOCK KEY
447 SCROLL_KEY EQU 70 ; SCAN CODE FOR SCROLL LOCK KEY
448 ALT_KEY EQU 56 ; SCAN CODE FOR ALTERNATE SHIFT KEY
449 CTL_KEY EQU 29 ; SCAN CODE FOR CONTROL KEY
450 CAPS_KEY EQU 58 ; SCAN CODE FOR SHIFT LOCK KEY
451 DEL_KEY EQU 83 ; SCAN CODE FOR DELETE KEY
452 INS_KEY EQU 82 ; SCAN CODE FOR INSERT KEY
453 LEFT_KEY EQU 42 ; SCAN CODE FOR LEFT SHIFT
454 RIGHT_KEY EQU 54 ; SCAN CODE FOR RIGHT SHIFT
455 SYS_KEY EQU 84 ; SCAN CODE FOR SYSTEM KEY
456
457 ----- ENHANCED KEYBOARD SCAN CODES -----
458 ID_1 EQU 0ABH ; 1ST ID CHARACTER FOR KBX
459 ID_2 EQU 041H ; 2ND ID CHARACTER FOR KBX
460 ID_2A EQU 064H ; ALTERNATE 2ND ID CHAR FOR KBX
461 FIT_M EQU 87 ; F11 KEY MAKE
462 F12_M EQU 88 ; F12 KEY MAKE
463 MC_E0 EQU 224 ; GENERAL MARKER CODE
464 MC_E1 EQU 225 ; PAUSE KEY MARKER CODE
    
```

```

464      C PAGE
465      C |----- FLAG EQUATES WITHIN *KB_FLAG -----
466 = 0001 C RIGHT_SHIFT EQU 0000001B ; RIGHT SHIFT KEY DEPRESSED
467 = 0002 C LEFT_SHIFT EQU 0000010B ; LEFT SHIFT KEY DEPRESSED
468 = 0004 C CTL_SHIFT EQU 00000100B ; CONTROL SHIFT KEY DEPRESSED
469 = 0008 C ALT_SHIFT EQU 00001000B ; ALTERNATE SHIFT KEY DEPRESSED
470 = 0010 C SCROLL_STATE EQU 00010000B ; SCROLL LOCK STATE IS ACTIVE
471 = 0020 C NUM_STATE EQU 00100000B ; NUM LOCK STATE IS ACTIVE
472 = 0040 C CAPS_STATE EQU 01000000B ; CAPS LOCK STATE IS ACTIVE
473 = 0080 C INS_STATE EQU 10000000B ; INSERT STATE IS ACTIVE
474      C |----- FLAG EQUATES WITHIN *KB_FLAG_1 -----
475      C |----- FLAG EQUATES WITHIN *KB_FLAG_1 -----
476 = 0001 C L_CTL_SHIFT EQU 00000001B ; LEFT CTL KEY DOWN
477 = 0002 C L_ALT_SHIFT EQU 00000010B ; LEFT ALT KEY DOWN
478 = 0004 C SYS_SHIFT EQU 00000100B ; SYSTEM KEY DEPRESSED AND HELD
479 = 0008 C HOLD_STATE EQU 00001000B ; SUSPEND KEY HAS BEEN TOGGLED
480 = 0010 C SCROLL_SHIFT EQU 00010000B ; SCROLL LOCK KEY IS DEPRESSED
481 = 0020 C NUM_SHIFT EQU 00100000B ; NUM LOCK KEY IS DEPRESSED
482 = 0040 C CAPS_SHIFT EQU 01000000B ; CAPS LOCK KEY IS DEPRESSED
483 = 0080 C INS_SHIFT EQU 10000000B ; INSERT KEY IS DEPRESSED
484      C |----- FLAGS EQUATES WITHIN *KB_FLAG_2 -----
485      C |----- FLAGS EQUATES WITHIN *KB_FLAG_2 -----
486 = 0007 C KB_LEDS EQU 00000111B ; KEYBOARD LED STATE BITS
487      C | 00000010B ; SCROLL LOCK INDICATOR
488      C | 00000001B ; NUM LOCK INDICATOR
489      C | 00000100B ; CAPS LOCK INDICATOR
490      C | 00001000B ; RESERVED (MUST BE ZERO)
491 = 0010 C KB_FA EQU 00010000B ; ACKNOWLEDGMENT RECEIVED
492 = 0020 C KB_FE EQU 00100000B ; RESEND RECEIVED FLAG
493 = 0040 C KB_FR_LED EQU 01000000B ; MODE INDICATOR UPDATE
494 = 0080 C KB_ERR EQU 10000000B ; KEYBOARD TRANSMIT ERROR FLAG
495      C |----- FLAGS EQUATES WITHIN *KB_FLAG_3 -----
496      C |----- FLAGS EQUATES WITHIN *KB_FLAG_3 -----
497 = 0001 C LC_E1 EQU 00000001B ; LAST CODE WAS THE E1 HIDDEN CODE
498 = 0002 C LC_E0 EQU 00000010B ; LAST CODE WAS THE E0 HIDDEN CODE
499 = 0004 C R_CTL_SHIFT EQU 00000100B ; RIGHT CTL KEY DOWN
500 = 0008 C R_ALT_SHIFT EQU 00001000B ; RIGHT ALT KEY DOWN
501 = 0008 C GRAPH_ON EQU 00001000B ; ALT GRAPHICS KEY DOWN (WT ONLY)
502 = 0010 C KBX EQU 00010000B ; ENHANCED KEYBOARD INSTALLED
503 = 0020 C SET_NUM_LK EQU 00100000B ; FORCE NUM LOCK IF READ ID AND KBX
504 = 0040 C LC_AB EQU 01000000B ; LAST CHARACTER WAS FIRST ID CHARACTER
505 = 0080 C RD_ID EQU 10000000B ; DOING A READ ID (MUST BE BIT0)
    
```

SECTION 5

```

506 C PAGE
507 C |-----|
508 C |-----|
509 C |-----|
510 C CMOS PORT EQU 070H ; I/O ADDRESS OF CMOS ADDRESS PORT
511 C CMOS_DATA EQU 071H ; I/O ADDRESS OF CMOS DATA PORT
512 C NM1 EQU 1000000B ; DISABLE NM1 INTERRUPTS MASK -
513 C ; HIGH BIT OF CMOS LOCATION ADDRESS
514 C
515 C |-----|
516 C |-----|
517 C |-----|
518 C |-----|
519 C |-----|
520 C |-----|
521 C |-----|
522 C |-----|
523 C |-----|
524 C |-----|
525 C |-----|
526 C |-----|
527 C |-----|
528 C |-----|
529 C |-----|
530 C |-----|
531 C |-----|
532 C |-----|
533 C |-----|
534 C |-----|
535 C |-----|
536 C |-----|
537 C |-----|
538 C |-----|
539 C |-----|
540 C |-----|
541 C |-----|
542 C |-----|
543 C |-----|
544 C |-----|
545 C |-----|
546 C |-----|
547 C |-----|
548 C |-----|
549 C |-----|
550 C |-----|
551 C |-----|
552 C |-----|
553 C |-----|
554 C |-----|
555 C |-----|
556 C |-----|
557 C |-----|
558 C |-----|
559 C |-----|
560 C |-----|
561 C |-----|
562 C |-----|
563 C |-----|
564 C |-----|
565 C |-----|
566 C |-----|
567 C |-----|
568 C |-----|
569 C |-----|
570 C |-----|
571 C |-----|
572 C |-----|
573 C |-----|
574 C |-----|
575 C |-----|
576 C |-----|
577 C |-----|
578 C |-----|
579 C |-----|
580 C |-----|
581 C |-----|
582 C |-----|
583 C |-----|
584 C |-----|
585 C |-----|
586 C |-----|
587 C |-----|
588 C |-----|
589 C |-----|
590 C |-----|
591 C |-----|
592 C |-----|
593 C |-----|
594 C |-----|
595 C |-----|
596 C |-----|
597 C |-----|
598 C |-----|
599 C |-----|
600 C |-----|
601 C |-----|
602 C |-----|
603 C |-----|
604 C |-----|
605 C |-----|
606 C |-----|
607 C |-----|
608 C |-----|
609 C |-----|
610 C |-----|
611 C |-----|
612 C |-----|
613 C |-----|
614 C |-----|

```

```

615          C PAGE
616          C |----- INTERRUPT EQUATES -----
617 = 0020   C EQI          EQU  020H          ; END OF INTERRUPT COMMAND TO 8259
618 = 0020   C INTA00      EQU  020H          ; 8259 PORT
619 = 0021   C INTA01      EQU  021H          ; 8259 PORT
620 = 00A0   C INTB00      EQU  0A0H          ; 2ND 8259
621 = 00A1   C INTB01      EQU  0A1H          ;
622 = 0070   C INT_TYPE    EQU  070H          ; START OF 8259 INTERRUPT TABLE LOCATION
623 = 0010   C INT_VIDED    EQU  010H          ; VIDEO VECTOR
624          C |-----
625 = 0008   C DMA08        EQU  008H          ; DMA STATUS REGISTER PORT ADDRESS
626 = 0000   C DMA         EQU  000H          ; DMA CH.0 ADDRESS REGISTER PORT ADDRESS
627 = 00C0   C DMA18        EQU  0C0H          ; 2ND DMA STATUS PORT ADDRESS
628 = 00C0   C DMA1         EQU  0C0H          ; 2ND DMA CH.0 ADDRESS REGISTER ADDRESS
629          C |-----
630 = 0040   C TIMER         EQU  040H          ; 8254 TIMER - BASE ADDRESS
631          C |-----
632          C |----- MANUFACTURING PORT -----
633 = 0080   C MFG_PORT      EQU  80H           ; MANUFACTURING AND POST CHECKPOINT PORT
634          C |-----
635          C |----- MANUFACTURING BIT DEFINITION FOR #MFG ERR FLAG+1 -----
636          C |-----
637 = 0001   C MEM_FAIL      EQU  00000001B      ; STORAGE TEST FAILED (ERROR 20X)
638 = 0002   C PRO_FAIL      EQU  00000010B      ; VIRTUAL MODE TEST FAILED (ERROR 104)
639 = 0004   C LMCS_FAIL     EQU  00000100B      ; LOW MEG CHIP SELECT FAILED (ERROR 109)
640 = 0008   C KYCLK_FAIL    EQU  00001000B      ; KEYBOARD CLOCK TEST FAILED (ERROR 304)
641 = 0010   C KY_SYS_FAIL   EQU  00010000B      ; KEYBOARD OR SYSTEM TEST FAILED (ERROR 303)
642 = 0020   C KYBD_FAIL     EQU  00100000B      ; KEYBOARD FAILED (ERROR 301)
643 = 0040   C DSK_FAIL     EQU  01000000B      ; DISKETTE TEST FAILED (ERROR 401)
644 = 0080   C KEY_FAIL      EQU  10000000B      ; KEYBOARD LOCKED (ERROR 302)
645          C |-----
646          C |-----
647 = 0081   C DMA_PAGE      EQU  081H          ; START OF DMA PAGE REGISTERS
648 = 008F   C LAST_DMA_PAGE EQU  08FH          ; LAST DMA PAGE REGISTER
649          C |-----
650          C |-----
651 = 00F0   C X287          EQU  0F0H          ; MATH COPROCESSOR CONTROL PORT
652          C |-----
653          C |-----
654 = 0000   C POST_SS       EQU  000000H      ; POST STACK SEGMENT
655 = 8000   C POST_SP       EQU  800000H      ; POST STACK POINTER
656          C |-----
657          C |-----
658 = 000D   C CR            EQU  000DH          ; CARRIAGE RETURN CHARACTER
659 = 000A   C LF            EQU  000AH          ; LINE FEED CHARACTER
660 = 0008   C RVRT         EQU  00001000B      ; VIDEO VERTICAL RETRACE BIT
661 = 0001   C RHRZ         EQU  00000001B      ; VIDEO HORIZONTAL RETRACE BIT
662 = 0100   C H            EQU  256          ; HIGH BYTE FACTOR (X 100H)
663 = 0101   C X            EQU  H+1          ; HIGH AND LOW BYTE FACTOR (X 101H)
664          C |-----
665          C .LIST
    
```

SECTION 5

```

666 PAGE
667 INCLUDE SYSDATA.INC
668 -----
669 | PROTECTED MODE EQUATES FOR POST TESTS AND BIOS ROUTINES |
670 |-----|
671 |
672 |----- LENGTH EQUATES FOR PROTECTED MODE TESTS
673 |
674 = 0300 SDA_LEN EQU 00300H ; SYSTEM DATA AREA LENGTH
675 = 0800 SYS_IDT_LEN EQU 256*8 ; 256 SYSTEM IDT ENTRIES, 8 BYTES EACH
676 = 0088 GDT_LEN EQU TYPE_GDT_DEF ; GDT STRUCTURE LENGTH
677 = 0008 DESC_LEN EQU TYPE_DATA_DESC ; LENGTH OF A DESCRIPTOR
678 = 1000 WCRT_SIZE EQU 4*1024 ; MONOCHROME CRT SIZE
679 = 4000 CCRT_SIZE EQU 16*1024 ; COMPATIBLE COLOR CRT SIZE
680 = FFFF ECRT_SIZE EQU 0FFFFH ; SIZE OF EACH PORTION OF THE ENHANCED
681 = FFFF MAX_SEG_LEN EQU 0FFFFH ; MAXIMUM SEGMENT LENGTH = 64K
682 = 0000 NULL_SEG_LEN EQU 00000H ; NULL SEGMENT LENGTH = 0
683 |
684 |----- LOCATION EQUATES FOR PROTECTED MODE TESTS
685 |
686 = 00A0 SYS_IDT_LOC EQU 00A00H ; THE SYSTEM IDT IS AT THE BOTTOM
687 = 0400 SDA_LOC EQU 00400H ; SAME AS REAL
688 = 08A0 GDT_LOC EQU (SYS_IDT_LOC + SYS_IDT_LEN)
689 = 0000 WCRT_LO EQU 00000H ; MONOCHROME CRT ADDRESS
690 = 000B WCRT_HI EQU 0BH ; (0B0000H)
691 = 8000 CCRT_LO EQU 8000H ; COMPATIBLE COLOR CRT ADDRESS
692 = 000B CCRT_HI EQU 0BH ; (0B8000H)
693 = 0000 ECRT_LO_LO EQU 0000H ;
694 = 000A ECRT_LO_HI EQU 0AH ; (0A0000H)
695 = 0000 ECRT_HI_LO EQU 0000H ;
696 = 000B ECRT_HI_HI EQU 0BH ; (0B0000H)
697 = 0000 CSEG_LO EQU 0000H ; CODE SEGMENT POST/BIOS
698 = 000F CSEG_HI EQU 0FH ; (0F0000H) FOR TESTS
699 = 8000 NSEG_LO EQU 8000H ; ABS0
700 = 0000 NSEG_HI EQU 00H ;
701 |
702 |----- DEFINITIONS FOR ACCESS RIGHTS BYTES
703 |
704 = 00F3 CPL3_DATA_ACCESS EQU 11110011B ; PRESENT
705 | ; DPL = 3
706 | ; CODE/DATA SEGMENT
707 | ; NOT EXECUTABLE
708 | ; GROW-UP (OFFSET <= LIMIT)
709 | ; WRITABLE
710 | ; ACCESSED
711 = 0093 CPL0_DATA_ACCESS EQU 10010011B ; DPL = 0
712 = 009B CPL0_CODE_ACCESS EQU 10011011B ; CPL 0 - NON-CONFORMING
713 = 00E2 LDT_CODE EQU 11100010B
714 = 00E1 FREE_TSS EQU 10000001B
715 = 0086 INT_GATE EQU 10000110B
716 = 0087 TRAP_GATE EQU 10000111B
717 |
718 = 0001 VIRTUAL_ENABLE EQU 0000000000000001B ; PROTECTED MODE ENABLE
719 |
720 |----- THE GLOBAL DESCRIPTOR TABLE DEFINITION FOR POWER ON SELF TESTS
721 |
722 |
723 GDT_DEF STRUC
724 0000 ?????????????????? DQ ? ; UNUSED ENTRY
725 0008 ?????????????????? DQ ? ; THIS ENTRY POINTS TO THIS TABLE
726 0010 ?????????????????? DQ ? ; POST INTERRUPT DESCRIPTOR TABLE
727 0018 ?????????????????? DQ ? ; THE REAL SYSTEM DATA AREA FOR POST
728 0020 ?????????????????? DQ ? ; COMPATIBLE BW CRT FOR POST
729 0028 ?????????????????? DQ ? ; COMPATIBLE COLOR CRT FOR POST
730 0030 ?????????????????? DQ ? ; ENHANCED COLOR GRAPHICS CRT (16 BYTES)
731 0038 ?????????????????? DQ ? ;
732 0040 ?????????????????? DQ ? ; CS - POST IDT, ROM RESIDENT
733 0048 ?????????????????? DQ ? ; DYNAMIC POINTER FOR ES
734 0050 ?????????????????? DQ ? ; DYNAMIC POINTER FOR CS
735 0058 ?????????????????? DQ ? ; DYNAMIC POINTER FOR SS
736 0060 ?????????????????? DQ ? ; DYNAMIC POINTER FOR DS
737 0068 ?????????????????? DQ ? ; TR VALUE FOR THIS MACHINE'S TSS
738 0070 ?????????????????? DQ ? ;
739 0078 ?????????????????? DQ ? ; LDTR VALUE FOR THIS MACHINE'S LDT
740 0080 ?????????????????? DQ ? ;
741 0088 ?????????????????? DQ ? ;
742 GDT_DEF ENDS
743 |
744 |----- SEGMENT DESCRIPTOR TABLE ENTRY STRUCTURE
745 |
746 DATA_DESC STRUC
747 0000 ???? DQ ? ; SEGMENT LIMIT (1 - 66636 BYTES)
748 0002 ???? DQ ? ; 24 BIT SEGMENT PHYSICAL
749 0004 ?? DB ? ; ADDRESS (0 - (16M-1))
750 0005 ?? DB ? ; ACCESS RIGHTS BYTE
751 0006 ???? DW ? ; RESERVED - MUST BE 0000 FOR THE 80286
752 DATA_DESC ENDS
753 |
754 |----- GATE DESCRIPTOR TABLE ENTRY STRUCTURE
755 |
756 0000 ???? GATE_DESC STRUC
757 0002 ???? DQ ? ; DESTINATION ROUTINE ENTRY POINT
758 0004 ?? DB ? ; SELECTOR FOR DESTINATION SEGMENT
759 0005 ?? DB ? ; NUMBER OF WORDS TO COPY FROM STACK
760 0006 ???? DW ? ; ACCESS RIGHTS BYTE
761 0008 ???? DW ? ; RESERVED - MUST BE 0000 FOR THE 80286
762 GATE_DESC ENDS
763 .LIST
    
```

```

764 PAGE
765 0000 CODE SEGMENT WORD PUBLIC
766
767 PUBLIC C8042
768 PUBLIC DBF 42
769 PUBLIC POST1
770 PUBLIC START_1
771
772 EXTRN CMOS_READ:NEAR
773 EXTRN CMOS_WRITE:NEAR
774 EXTRN CONFIG_BAD:NEAR
775 EXTRN D11:NEAR
776 EXTRN DS:NEAR
777 EXTRN DUMMY_RETURN:NEAR
778 EXTRN ERR_BEEP:NEAR
779 EXTRN GATE_A20:NEAR
780 EXTRN KBD_RESET:NEAR
781 EXTRN NMI_INT:NEAR
782 EXTRN POST2:NEAR
783 EXTRN PRINT_SCREEN:NEAR
784 EXTRN PROC_SHUTDOWN:NEAR
785 EXTRN ROM_CHECK:NEAR
786 EXTRN SHUT2:NEAR
787 EXTRN SHUT3:NEAR
788 EXTRN SHUT4:NEAR
789 EXTRN SHUT5:NEAR
790 EXTRN SHUT6:NEAR
791 EXTRN SHUT7:NEAR
792 EXTRN SHUT8:NEAR
793 EXTRN SLAVE_VECTOR_TABLE:NEAR
794 EXTRN STGTST_CNT:NEAR
795 EXTRN SYSINIT:NEAR
796 EXTRN VECTOR_TABLE:NEAR
797 EXTRN VIDEO_PARMS:BYTE
798
799 ASSUME CS:CODE,DS:NOTHING,ES:NOTHING,SS:NOTHING
800 0000 POST1 PROC NEAR
801
802 = 0000 BEGIN EQU $
803 0000 37 38 58 37 34 36 DB '78X7462COPR. IBM CORP. 1981,1986 ' ;COPYRIGHT NOTICE
804 32 43 4F 50 52 2E
805 20 49 42 4D 20 43
806 4F 52 50 2E 20 31
807 39 38 31 2C 31 39
808 38 36 20 20
809
810 EVEN
811 | 7 8 X 7 4 6 2 C O P R . I B M 1 9 8 6 |EVEN BOUNDARY
812 | 7 8 X 7 4 6 3 C O P R . I B M 1 9 8 6 |EVEN MODULE
813 DB '7788XX7446623 CCOOPRRR.. I18BM 11998866' ;COPYRIGHT NOTICE
814 32 33 20 20 43 43
815 4F 4F 50 50 52 52
816 2E 2E 20 20 49 49
817 42 42 4D 4D 20 20
818 31 31 39 39 38 38
819
820 004E 20 20 DB ' ' ;PAD
821
822 -----
823 | INITIAL RELIABILITY TESTS -- (POST1) |
824 |-----|
825
826 |-----|
827 | TEST_01 |
828 | 80286 PROCESSOR TEST (REAL MODE) |
829 | DESCRIPTION |
830 | VERIFY FLAGS, REGISTERS |
831 | AND CONDITIONAL JUMPS. |
832 |-----|
833
834 ASSUME DS:DATA
835
836 0050 START_1 CLJ ; DISABLE INTERRUPTS
837 0051 88 D5BD MOV AX,0D500H+CMOS_REG_D+NMI ; FLAG MASK IN (AH) AND NMI MASK IN (AL)
838 0052 E6 70 OUT CMOS_PORT,AL ; DISABLE NMI INTERRUPTS, LATCH STANDBY
839 0053 9E SAHF ; SET "SF" "ZF" "AF" "PF" "CF" FLAGS ON
840 0054 73 27 JNC ERR02 ; GO TO ERROR ROUTINE IF "CF" NOT SET
841 0055 76 25 JNZ ERR02 ; GO TO ERROR ROUTINE IF "ZF" NOT SET
842 0056 7B 23 JNP ERR02 ; GO TO ERROR ROUTINE IF "PF" NOT SET
843 0057 79 21 JNS ERR02 ; GO TO ERROR ROUTINE IF "SF" NOT SET
844 0058 9F LAHF ; LOAD FLAG IMAGE TO (AH)
845 0059 B1 05 MOV AH,CL ; LOAD CARRY BIT POSITION
846 0060 D2 EC SHR AH,CL ; SHIFT "AF" INTO CARRY BIT POSITION
847 0061 73 1A JNC ERR02 ; GO TO ERROR ROUTINE IF "AF" NOT SET
848 0062 90 40 MOV AL,40H ; SET THE "OF" FLAG ON
849 0063 D0 E0 SHL AL,1 ; SETUP FOR TESTING
850 0064 71 14 JNO ERR02 ; GO TO ERROR ROUTINE IF "OF" NOT SET
851 0065 32 E4 XOR AH,AH ; SET (AH) = 0
852 0066 9E SAHF ; CLEAR "SF", "CF", "ZF", AND "PF"
853 0067 76 0F JBE ERR02 ; GO TO ERROR ROUTINE IF "CF" ON
854 0068 73 0F JNC ERR02 ; GO TO ERROR ROUTINE IF "ZF" ON
855 0069 76 0F JNB ERR02 ; GO TO ERROR ROUTINE IF "PF" ON
856 0070 79 0F JNS ERR02 ; GO TO ERROR ROUTINE IF "SF" ON
857 0071 78 0D JF ERR02 ; LOAD FLAG IMAGE TO (AH)
858 0072 7A 0B JG ERR02 ; SHIFT "AF" INTO CARRY BIT POSITION
859 0073 7D 06 JGE ERR02 ; GO TO ERROR ROUTINE IF ON
860 0074 7C 04 JLE ERR02 ; CHECK THAT "OF" IS CLEAR
861 0075 7E 04 JG ERR02 ; GO TO ERROR ROUTINE IF ON
862 0076 7F 03 JGE ERR02 ; CONTINUE CONFIDENCE TESTS IF "ZF" SET
863 0077 74 03 JLE ERR02: ;
864 0078 7C 03 JLE ERR02: ;
865 0079 F4 HLT ; ERROR HALT
866 0080 EB FD JMP ERR02 ; ERROR LOOP TRAP
867
868 0083 CTA1 MOV AX,DATA ; SET DATA SEGMENT
869 0084 B8 ---- R MOV DS,AX ; INTO THE (DS) SEGMENT REGISTER
870 0085 8E D8
871
872 |-----|
873 | CHECK FOR PROCESSOR SHUTDOWN |
874 |-----|
875 0088 E4 64 IN AL,STATUS_PORT ; READ CURRENT KEYBOARD PROCESSOR STATUS
876 0089 A8 04 TEST AL,STATUS_FLAG ; CHECK FOR SHUTDOWN IN PROCESS FLAG
877 0090 75 03 JNZ C7B ; GO IF YES
878 0091 E9 0123 R JMP SHUT0 ; ELSE CONTINUE NORMAL POWER ON CODE
    
```

SECTION 5

```

878 PAGE
879 I----- CHECK FOR SHUTDOWN 09
880 0091 CTB:
881 0091 B0 8F MOV AL,CMOS_SHUT_DOWN+NMI ; CMOS ADDRESS FOR SHUTDOWN BYTE
882 0093 E6 70 OUT CMOS_PORT,AL
883 0095 EB 00 JMP $+2 ; I/O DELAY
884 0097 E4 71 IN AL,CMOS_DATA ; GET REQUEST NUMBER
885 0099 3C 09 CMP AL,09H ; WAS IT SHUTDOWN REQUEST ?
886 009B 86 C4 XCHG AL,AH ; SAVE THE SHUTDOWN REQUEST
887 009D 74 41 JE CTC ; BYPASS INITIALIZING INTERRUPT CHIPS
888
889 I----- CHECK FOR SHUTDOWN 0A
890
891 009F 80 FC 0A CMP AH,0AH ; WAS IT SHUTDOWN REQUEST A?
892 00A2 74 3C JE CTC ; BYPASS INITIALIZING INTERRUPT CHIPS
893
894 00A4 2A C0 SUB AL,AL ; INSURE MATH PROCESSOR RESET
895 00A6 E6 F1 OUT X287+1,AL
896
897 I----- RE-INITIALIZE THE 8259 INTERRUPT #1 CONTROLLER CHIP :
898
899
900 00A8 B0 11 MOV AL,11H ; ICW1 - EDGE, MASTER, ICW4
901 00AA E6 20 OUT INTA00,AL
902 00AC EB 00 JMP $+2 ; WAIT STATE FOR I/O
903 00AE B0 08 MOV AL,08H ; SETUP ICW2 - INTERRUPT TYPE 08 (8-F)
904 00B0 E6 21 OUT INTA01,AL
905 00B2 EB 00 JMP $+2 ; WAIT STATE FOR I/O
906 00B4 B0 04 MOV AL,04H ; SETUP ICW3 - MASTER LEVEL 2
907 00B6 E6 21 OUT INTA01,AL
908 00B8 EB 00 JMP $+2 ; I/O WAIT STATE
909 00BA B0 01 MOV AL,01H ; SETUP ICW4 - MASTER,8086 MODE
910 00BC E6 21 OUT INTA01,AL ; WAIT STATE FOR I/O
911 00BE EB 00 JMP $+2 ; MASK ALL INTERRUPTS OFF
912 00C0 B0 FF MOV AL,0FFH ; (VIDEO ROUTINE ENABLES INTERRUPTS)
913 00C2 E6 21 OUT INTA01,AL
914
915 I----- RE-INITIALIZE THE 8259 INTERRUPT #2 CONTROLLER CHIP :
916
917 00C4 B0 11 MOV AL,11H ; ICW1 - EDGE, SLAVE ICW4
918 00C6 E6 A0 OUT INTB00,AL
919 00C8 EB 00 JMP $+2 ; WAIT STATE FOR I/O
920 00CA B0 70 MOV AL,INT_TYPE ; SETUP ICW2 - INTERRUPT TYPE 70 (70-7F)
921 00CC E6 A1 OUT INTB01,AL
922 00CE B0 02 MOV AL,02H ; SETUP ICW3 - SLAVE LEVEL 2
923 00D0 EB 00 JMP $+2
924 00D2 E6 A1 OUT INTB01,AL ; I/O DELAY
925 00D4 EB 00 JMP $+2 ; SETUP ICW4 - 8086 MODE, SLAVE
926 00D6 B0 01 MOV AL,01H
927 00D8 E6 A1 OUT INTB01,AL ; WAIT STATE FOR I/O
928 00DA EB 00 JMP $+2 ; MASK ALL INTERRUPTS OFF
929 00DC B0 FF MOV AL,0FFH
930 00DE E6 A1 OUT INTB01,AL
931
932 I----- SHUTDOWN - RESTART
933 RETURN CONTROL AFTER A SHUTDOWN COMMAND IS ISSUED
934 DESCRIPTION
935 A TEST IS MADE FOR THE SYSTEM FLAG BEING SET. IF THE SYSTEM FLAG IS
936 SET, THE SHUTDOWN BYTE IN CMOS IS USED TO DETERMINE WHERE CONTROL IS
937 RETURNED.
938
939 CMOS = 0 SOFT RESET OR UNEXPECTED SHUTDOWN
940 CMOS = 1 SHUT DOWN AFTER MEMORY SIZE
941 CMOS = 2 SHUT DOWN AFTER MEMORY TEST
942 CMOS = 3 SHUT DOWN WITH MEMORY ERROR
943 CMOS = 4 SHUT DOWN WITH BOOT LOADER REQUEST
944 CMOS = 5 JMP DWORD REQUEST - (INTERRUPT CHIPS & 287 ARE INITIALIZED)
945 CMOS = 6 PROTECTED MODE TEST3 PASSED
946 CMOS = 7 PROTECTED MODE TEST3 FAILED
947 CMOS = 8 PROTECTED MODE TEST1 FAILED
948 CMOS = 9 BLOCK MOVE SHUTDOWN REQUEST
949 CMOS = A JMP DWORD REQUEST - (W/O INTERRUPT CHIPS INITIALIZED)
950
951 NOTES: RETURNS ARE MADE WITH INTERRUPTS AND NMI DISABLED.
952 USER MUST RESTORE SS:SP (POST DEFAULT SET = 0000:0400),
953 ENABLE NON-MASKABLE INTERRUPTS (NMI) WITH AN OUT TO
954 PORT 70H WITH HIGH ORDER BIT OFF, AND THEN ISSUE A
955 STI TO ENABLE INTERRUPTS. FOR SHUTDOWN (S) THE USER
956 MUST ALSO RESTORE THE INTERRUPT MASK REGISTERS.
957
958 I----- CHECK FROM WHERE
959 CTB:
960 00E0
961 00E0 B0 8F MOV AL,CMOS_SHUT_DOWN+NMI ; CLEAR CMOS BYTE
962 00E2 E6 70 OUT CMOS_PORT,AL
963 00E4 90 NOP ; I/O DELAY
964 00E5 2A C0 SUB AL,AL ; SET BYTE TO 0
965 00E7 E6 71 OUT CMOS_DATA,AL
966 00E9 86 E0 XCHG AH,AL
967 00EB 3C 0A CMP AH,0AH ; COMPARE WITH MAXIMUM TABLE ENTRIES
968 00ED 77 34 JA SHUT0 ; SKIP TO POST IF GREATER THAN MAXIMUM
969 00EF BE 0103 R MOV SI,OFFSET BRANCH ; POINT TO THE START OF THE BRANCH TABLE
970 00F2 03 F0 ADD SI,AX
971 00F4 03 F0 ADD SI,AX ; POINT TO BRANCH ADDRESS
972 00F6 2E: 8B 1C MOV BX,CS:[SI] ; MOVE BRANCH TO ADDRESS TO BX REGISTER
973
974 I----- SET TEMPORARY STACK FOR POST
975
976 00F9 BB ---- R MOV AX,ABS0 ; SET STACK SEGMENT TO ABS0 SEGMENT
977 00FC BE D0 MOV SI,AX
978 00FE BC 0400 R MOV SP,OFFSET *TOS ; SET STACK POINTER TO END OF VECTORS
979 0101 FF E3 JMP BX ; JUMP BACK TO RETURN ROUTINE
980
981 0103 0123 R BRANCH: DW SHUT0 ; NORMAL POWER UP/UNEXPECTED SHUTDOWN
982 0105 0992 R DW SHUT1 ; SHUT DOWN AFTER MEMORY SIZE
983 0107 0000 E DW SHUT2 ; SHUT DOWN AFTER MEMORY TEST
984 0109 0000 E DW SHUT3 ; SHUT DOWN WITH MEMORY ERROR
985 010B 0000 E DW SHUT4 ; SHUT DOWN WITH BOOT LOADER REQUEST
986 010D 0119 R DW SHUT5 ; JMP DWORD REQUEST WITH INTERRUPT INIT
987 010F 0000 E DW SHUT6 ; PROTECTED MODE TEST7 PASSED
988 0111 0000 E DW SHUT7 ; PROTECTED MODE TEST7 FAILED
989 0113 0795 R DW SHUT8 ; PROTECTED MODE TEST1 FAILED
990 0115 0000 E DW SHUT9 ; BLOCK MOVE SHUTDOWN REQUEST
991 0117 011F R DW SHUTA ; JMP DWORD REQUEST (W/O INTERRUPT INIT)
    
```





```

1106 ;-----
1107 ; TEST.03
1108 ; VERIFY CMOS SHUTDOWN BYTE
1109 ; DESCRIPTION
1110 ; ROLLING BIT WRITTEN AND
1111 ; VERIFIED AT SHUTDOWN ADDRESS.
1112 ;-----
1113
1114 ;----- VERIFY AND CLEAR SHUTDOWN FLAG
1115
1116 01A6
1117 01A6 B0 03
1118 01A8 E6 80
1119
1120 01AA B9 0009
1121 01AD B4 01
1122 01AF
1123 01AF B0 8F
1124 01B1 E6 70
1125 01B3 8A C4
1126 01B6 E6 71
1127 01B7 B0 8F
1128 01B9 90
1129 01BA E6 70
1130 01BC 90
1131 01BD E4 71
1132 01BF 3A C4
1133 01C1 76 8B
1134 01C3 D0 D4
1135 01C5 E2 E6
1136
1137
1138 ;-----
1139 ; TEST.04
1140 ; 8254 CHECK TIMER 1
1141 ; DESCRIPTION
1142 ; SET TIMER COUNT TO
1143 ; CHECK THAT TIMER 1 ALL BITS ON
1144 ;-----
1145
1146 01C7 B8 ---- R
1147 01CA 8E D8
1148 01CC B0 D4
1149 01CE E6 80
1150
1151 ;----- DISABLE DMA CONTROLLER
1152 ; (AL) ALREADY = 04H
1153 ; DISABLE DMA CONTROLLER 1
1154 ; DISABLE DMA CONTROLLER 2
1155
1156 01D0 E6 08
1157 01D2 E6 D0
1158
1159 ;----- VERIFY THAT TIMER 1 FUNCTIONS OK
1160
1161 01D4 8B 16 0072 R
1162 01D8 B0 84
1163 01DA E6 43
1164 01DC EB 00
1165 01DE B0 12
1166 01E0 E6 41
1167 01E2 B7 05
1168 01E4
1169 01E4 B0 40
1170 01E6 EB 00
1171 01E8 E6 43
1172 01EA 80 FB 1F
1173 01ED 74 0B
1174 01EF E4 41
1175 01F1 0A D8
1176 01F3 E2 EF
1177 01F5 FE CF
1178 01F7 75 EB
1179 01F9 F4
1180
1181 ;-----
1182 ; TEST.05
1183 ; 8254 CHECK TIMER 1 ALL BIT OFF
1184 ; DESCRIPTION
1185 ; SET TIMER COUNT
1186 ; CHECK THAT TIMER 1 ALL BITS OFF
1187 ;-----
1188
1189 01FA B0 05
1190 01FC E6 80
1191
1192 01FE 8A C3
1193 0200 2B C9
1194 0202 E6 41
1195 0204 B7 05
1196 0206
1197 0206 B0 40
1198 0208 E6 43
1199 020A EB 00
1200 020C EB 00
1201 020E E4 41
1202 0210 22 D6
1203 0212 74 07
1204 0214 E2 F0
1205 0216 FE CF
1206 0218 75 EC
1207 021A F4
1208
1209 ;-----
1210 ; TEST.06
1211 ; 8237 DMA 0 INITIALIZATION
1212 ; CHANNEL REGISTER TEST
1213 ; DESCRIPTION
1214 ; DISABLE THE 8237 DMA CONTROLLER.
1215 ; WRITE/READ THE CURRENT ADDRESS
1216 ; AND WORD COUNT REGISTERS FOR
1217 ; ALL CHANNELS.
1218 ;-----
1219
1106 ;-----
1107 ; TEST.03
1108 ; VERIFY CMOS SHUTDOWN BYTE
1109 ; DESCRIPTION
1110 ; ROLLING BIT WRITTEN AND
1111 ; VERIFIED AT SHUTDOWN ADDRESS.
1112 ;-----
1113
1114 ;----- VERIFY AND CLEAR SHUTDOWN FLAG
1115
1116 01A6
1117 01A6 B0 03
1118 01A8 E6 80
1119
1120 01AA B9 0009
1121 01AD B4 01
1122 01AF
1123 01AF B0 8F
1124 01B1 E6 70
1125 01B3 8A C4
1126 01B6 E6 71
1127 01B7 B0 8F
1128 01B9 90
1129 01BA E6 70
1130 01BC 90
1131 01BD E4 71
1132 01BF 3A C4
1133 01C1 76 8B
1134 01C3 D0 D4
1135 01C5 E2 E6
1136
1137
1138 ;-----
1139 ; TEST.04
1140 ; 8254 CHECK TIMER 1
1141 ; DESCRIPTION
1142 ; SET TIMER COUNT TO
1143 ; CHECK THAT TIMER 1 ALL BITS ON
1144 ;-----
1145
1146 01C7 B8 ---- R
1147 01CA 8E D8
1148 01CC B0 D4
1149 01CE E6 80
1150
1151 ;----- DISABLE DMA CONTROLLER
1152 ; (AL) ALREADY = 04H
1153 ; DISABLE DMA CONTROLLER 1
1154 ; DISABLE DMA CONTROLLER 2
1155
1156 01D0 E6 08
1157 01D2 E6 D0
1158
1159 ;----- VERIFY THAT TIMER 1 FUNCTIONS OK
1160
1161 01D4 8B 16 0072 R
1162 01D8 B0 84
1163 01DA E6 43
1164 01DC EB 00
1165 01DE B0 12
1166 01E0 E6 41
1167 01E2 B7 05
1168 01E4
1169 01E4 B0 40
1170 01E6 EB 00
1171 01E8 E6 43
1172 01EA 80 FB 1F
1173 01ED 74 0B
1174 01EF E4 41
1175 01F1 0A D8
1176 01F3 E2 EF
1177 01F5 FE CF
1178 01F7 75 EB
1179 01F9 F4
1180
1181 ;-----
1182 ; TEST.05
1183 ; 8254 CHECK TIMER 1 ALL BIT OFF
1184 ; DESCRIPTION
1185 ; SET TIMER COUNT
1186 ; CHECK THAT TIMER 1 ALL BITS OFF
1187 ;-----
1188
1189 01FA B0 05
1190 01FC E6 80
1191
1192 01FE 8A C3
1193 0200 2B C9
1194 0202 E6 41
1195 0204 B7 05
1196 0206
1197 0206 B0 40
1198 0208 E6 43
1199 020A EB 00
1200 020C EB 00
1201 020E E4 41
1202 0210 22 D6
1203 0212 74 07
1204 0214 E2 F0
1205 0216 FE CF
1206 0218 75 EC
1207 021A F4
1208
1209 ;-----
1210 ; TEST.06
1211 ; 8237 DMA 0 INITIALIZATION
1212 ; CHANNEL REGISTER TEST
1213 ; DESCRIPTION
1214 ; DISABLE THE 8237 DMA CONTROLLER.
1215 ; WRITE/READ THE CURRENT ADDRESS
1216 ; AND WORD COUNT REGISTERS FOR
1217 ; ALL CHANNELS.
1218 ;-----
1219

```



```

1334 02AD 2A C0          SUB     AL,AL          ; DACK SENSE LOW,DREQ SENSE HIGH
1335 02AF E6 08          OUT    DMA+8,AL       ; LATE WRITE, FIXED PRIORITY, NORMAL
1336                                     ; TIMING, CONTROLLER ENABLE, CH0 ADDRESS
1337                                     ; HOLD DISABLE, MEMORY TO MEMORY DISABLE
1338 02B1 E6 D0          OUT    DMA16,AL      ; SAME TO SECOND CONTROLLER
1339
1340                                     |----- MODE SET ALL DMA CHANNELS
1341
1342 02B3 80 40          MOV    AL,40H        ; SET MODE FOR CHANNEL 0
1343 02B5 E6 08          OUT    DMA+0BH,AL    ;
1344 02B7 80 C0          MOV    AL,0C0H      ; SET CASCADE MODE ON CHANNEL 4
1345 02B9 E6 D6          OUT    DMA18+06H,AL ;
1346 02BB EB 00          JMP    $+2           ; I/O DELAY
1347 02BD 80 41          MOV    AL,41H        ; SET MODE FOR CHANNEL 1
1348 02BF E6 08          OUT    DMA+0BH,AL    ;
1349 02C1 E6 D6          OUT    DMA18+06H,AL ; SET MODE FOR CHANNEL 5
1350 02C3 EB 00          JMP    $+2           ; I/O DELAY
1351 02C5 80 42          MOV    AL,42H        ; SET MODE FOR CHANNEL 2
1352 02C7 E6 08          OUT    DMA+0BH,AL    ;
1353 02C9 E6 D6          OUT    DMA18+06H,AL ; SET MODE FOR CHANNEL 6
1354 02CB EB 00          JMP    $+2           ; I/O DELAY
1355 02CD 80 43          MOV    AL,43H        ; SET MODE FOR CHANNEL 3
1356 02CF E6 08          OUT    DMA+0BH,AL    ;
1357 02D1 E6 D6          OUT    DMA18+06H,AL ; SET MODE FOR CHANNEL 7
1358
1359                                     |----- RESTORE RESET FLAG
1360
1361 02D3 89 1E 0072 R    MOV    @RESET_FLAG,BX
1362
1363                                     |-----
1364                                     | TEST.08
1365                                     | DMA PAGE REGISTER TEST
1366                                     | DESCRIPTION
1367                                     | WRITE/READ ALL PAGE REGISTERS
1368                                     |-----
1369
1370                                     |----- CHECKPOINT 08
1371
1372 02D7 80 08          MOV    AL,08H        ;
1373 02D9 E6 80          OUT    MFG_PORT,AL  ;
1374 02DB 2A C0          SUB    AL,AL         ;
1375 02DD BA 0081        MOV    DX,DMA_PAGE  ; DO ALL DATA PATTERNS
1376 02E0 B9 00FF        MOV    CX,OFFFH     ;
1377 02E2 EC             OUT    DX,AL        ;
1378 02E4 42             INC    DX            ;
1379 02E6 FE C0          INC    AL            ;
1380 02E7 81 FA 008F     CMP    DX,8FH        ; TEST DMA PAGES 81 THROUGH 8EH
1381 02E9 76 F6          JNZ   C22A           ;
1382 02ED 86 E0          XCHG  AH,AL         ; SAVE CURRENT DATA PATTERN
1383 02EF FE CC          DEC   AH            ; CHECK LAST WRITTEN
1384 02F1 4A             DEC   DX            ;
1385 02F2 2A C0          SUB   AL,AL         ; CHANGE DATA BEFORE READ
1386 02F4 EC             IN   AL,DX          ;
1387 02F6 3A C4          CMP   AL,AH         ; DATA AS WRITTEN?
1388 02F7 76 30          JNZ   C25           ; GO ERROR HALT IF NOT
1389 02F9 FE CC          DEC   AH            ;
1390 02FB 4A             DEC   DX            ;
1391 02FC 81 FA 0080     CMP   DX,MFG_PORT  ; CONTINUE TILL PORT 80
1392 0300 76 F0          JNZ   C22B           ;
1393 0302 FE C4          INC   AH            ; NEXT PATTERN TO RIPPLE
1394 0304 8A C4          MOV   AL,AH         ;
1395 0306 E2 D8          LOOP C22A           ;
1396
1397                                     |----- TEST LAST DMA PAGE REGISTER (USED FOR ADDRESS LINES DURING REFRESH)
1398
1399 0308 80 CC          MOV    AL,0CCH      ; WRITE AN CC TO PAGE REGISTERS
1400 030A BA 008F        MOV    DX,LAST_DMA_PAGE ; GET THE DATA FROM PAGE REGISTER
1401 030D 8A E0          MOV    AH,AL        ; SAVE THE DATA PATTERN
1402 030F EE             OUT    DX,AL        ; OUTPUT PAGE REGISTER
1403
1404                                     |----- VERIFY PAGE REGISTER 8F
1405
1406 0310 2A C0          SUB    AL,AL        ; CHANGE DATA PATTERN BEFORE READ
1407 0312 EC             IN     AL,DX         ; GET THE DATA FROM PAGE REGISTER
1408 0313 3A C4          CMP    AL,AH        ;
1409 0315 76 12          JNZ   C26           ; GO IF ERROR
1410 0317 80 FC CC       CMP    AH,0CCH      ;
1411 031A 76 04          JNZ   C25           ; GO IF ERROR
1412 031C 80 33          MOV    AL,033H     ; SET UP DATA PATTERN OF 33
1413 031E EB EA          JMP    C22          ; DO DATA 33
1414
1415 0320 80 FC 00        CMP    AH,0         ; CHECK DONE
1416 0323 74 06          JZ    C27           ; GO IF YES
1417 0325 2A C0          SUB    AL,AL        ; SET UP FOR DATA PATTERN 00
1418 0327 EB E1          JMP    C22          ; DO DATA 0
1419
1420                                     |----- ERROR HALT
1421 0329
1422 0329 F4             C26: HLT             ; HALT SYSTEM
1423
1424                                     |-----
1425                                     | TEST.09
1426                                     | STORAGE REFRESH TEST
1427                                     | DESCRIPTION
1428                                     | VERIFY REFRESH IS OCCURRING
1429                                     |-----
1430
1431                                     |----- CHECKPOINT 09 - TEST MEMORY REFRESH
1432
1433 032A
1434 032A 80 09          MOV    AL,09H        ;
1435 032C E6 80          OUT    MFG_PORT,AL  ;
1436 032E 2B C9          SUB    CX,CX         ;
1437 0330
1438 0330 E4 61          IN     AL,PORT_B    ; INSURE REFRESH BIT IS TOGGLING
1439 0332 A6 10          TEST  AL,REFRESH_BIT ; INSURE REFRESH IS OFF
1440 0334 E1 FA          LOOPZ C26           ; ERROR HALT IF TIMEOUT
1441 0336 E3 F1          JCXZ  C26           ;
1442 0338
1443 0338 E4 61          IN     AL,PORT_B    ; INSURE REFRESH IS ON
1444 033A A8 10          TEST  AL,REFRESH_BIT ;
1445 033C E0 FA          LOOPNZ C29          ; INSURE REFRESH IS ON
1446 033E E3 E9          JCXZ  C26           ; ERROR HALT IF NO REFRESH BIT
1447

```

```

1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462 0340 B0 0A
1463 0342 E6 80
1464
1465
1466
1467 0344 2B C9
1468 0346 E4 64
1469 0348 8A E0
1470 034A F6 C4 01
1471 034D 74 02
1472 034F E4 60
1473 0351 F6 C4 02
1474 0354 E0 F0
1475 0356 74 01
1476
1477 0358 F4
1478
1479
1480
1481 0359 B0 0B
1482 035B E6 80
1483
1484 035D B0 AA
1485 035F BC 03F5 R
1486 0362 EB 39
1487 0364 AB 01
1488 0366 74 02
1489 0368 E4 60
1490 036A BC 03F7 R
1491 036D EB 3A
1492 036F E4 60
1493 0371 3C 55
1494
1495 0373 B0 0C
1496 0375 E6 80
1497
1498 0377 75 DF
1499
1500
1501
1502 0379 B0 C0
1503 037B BC 03F5 R
1504 037E EB 1D
1505 0380 BC 03FD R
1506 0383 EB 24
1507 0385 E4 60
1508 0387 E6 82
1509
1510
1511
1512 0389 B0 60
1513 038B BC 03F9 R
1514 038E EB 0D
1515 0390 74 05
1516
1517 0392 B0 0D
1518 0394 E6 80
1519 0396 F4
1520 0397
1521 0397 B0 5D
1522 0399 E6 60
1523 039B EB 1D
1524
1525
1526
1527 039D FA
1528 039E E6 64
1529 03A0 2B C9
1530 03A2 E4 64
1531 03A4 AB 02
1532 03A6 E0 FA
1533 03A8 C3
1534
1535
1536
1537 03A9 2B C9
1538 03AB B3 06
1539 03AD EC 64
1540 03AF AB 01
1541 03B1 75 06
1542 03B3 E2 F6
1543 03B5 FE CB
1544 03B7 75 F4
1545 03B9 C3
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559 03BA B0 0E
1560 03BC E6 80
1561

```

```

-----
: TEST.10
:
: 8042 INTERFACE TEST
:
: READ CONFIGURATION JUMPERS
:
: DESCRIPTION
:
: ISSUE A SELF TEST TO THE 8042.
:
: INSURE A 55H IS RECEIVED.
:
: READ MANUFACTURING AND DISPLAY
:
: JUMPERS AND SAVE IN MFG_TEST.
:
-----
:----- CHECKPOINT 0A
:
: MOV AL,0AH
: OUT MFG_PORT,AL
:
:----- SOFT RESET (HANDLE ALL POSSIBLE CONDITIONS)
:
: SUB CX,CX
: TST1: IN AL,STATUS_PORT
: MOV AH,AL
: TEST AH,OUT_BUF_FULL
: JZ TST2
: IN AL,PORT_A
: TST2: TEST AH,INPT_BUF_FULL
: LOOPNZ TST1
: JZ TST4
:
: 100 MILLISECONDS FOR THIS LOOP
: CHECK FOR INPUT BUFFER FULL
:
: GO IF NOT
: FLUSH
: IS THE OUTPUT BUFFER ALSO FULL?
: TRY AGAIN
: CONTINUE IF OK
:
: HALT SYSTEM IF BUFFER FULL
:
:----- ISSUE A RESET TO THE 8042
:
: MOV AL,0BH
: OUT MFG_PORT,AL
:
: SELF TEST COMMAND
: SET RETURN ADDRESS
:
: TST4_B: TEST AL,OUT_BUF_FULL
: JZ TST4_A
: IN AL,PORT_A
: TST4_A: MOV SP,OFFSET OBF_42A
: JMP SHORT C8042
: TST4_C: IN AL,PORT_A
: CMP AL,55H
:
: IS THE OUTPUT BUFFER FULL?
: GO IF NOT
: FLUSH
: SET RETURN ADDRESS
: GO WAIT FOR BUFFER
: GET THE ENDING RESPONSE
:
: MOV AL,0CH
: OUT MFG_PORT,AL
:
: CHECKPOINT 0B
:
: GO IF NOT OK
:
:----- GET THE SWITCH SETTINGS
:
: MOV AL,READ_8042_INPUT
: MOV SP,OFFSET C8042C
: JMP SHORT C8042
: E30B: MOV SP,OFFSET OBF_42B
: JMP SHORT OBF_42
: E30C: IN AL,PORT_A
: OUT DMA_PAGE+1,AL
:
: READ INPUT COMMAND
: SET RETURN ADDRESS
: ISSUE COMMAND
: SET RETURN ADDRESS
: GO WAIT FOR RESPONSE
: GET THE SWITCH
: SAVE TEMPORARY
:
:----- WRITE BYTE 0 OF 8042 MEMORY
:
: MOV AL,WRITE_8042_LOC
: MOV SP,OFFSET C8042B
: JMP SHORT C8042
: TST4_D: JZ TST4_D1
:
: WRITE BYTE COMMAND
: SET RETURN ADDRESS
: ISSUE THE COMMAND
: CONTINUE IF COMMAND ACCEPTED
:
: MOV AL,0DH
: OUT MFG_PORT,AL
:
: CHECKPOINT 0C
:
: TST4_D1: MOV AL,5DH
: OUT PORT_A,AL
: JMP SHORT E30A
:
: ENABLE OUTPUT BUFFER FULL INTERRUPT,
: DISABLE KEYBOARD, SET SYSTEM FLAG,
: PC I COMPATIBILITY, INHIBIT OVERRIDE
:
:----- ISSUE THE COMMAND TO THE 8042
:
: C8042: CLI
: OUT STATUS_PORT,AL
: SUB CX,CX
: C42_1: IN AL,STATUS_PORT
: TEST AL,INPT_BUF_FULL
: LOOPNZ C42_1
: RET
:
: NO INTERRUPTS ALLOWED
: SEND COMMAND IN AL REGISTER
: LOOP COUNT
: WAIT FOR THE COMMAND ACCEPTED
:
:----- WAIT FOR 8042 RESPONSE
:
: OBF_42: SUB CX,CX
: MOV BL,6
: C42_2: IN AL,STATUS_PORT
: TEST AL,OUT_BUF_FULL
: JNZ C42_3
: LOOP C42_2
: DEC BL
: JNZ C42_2
: C42_3: RET
:
: 200MS/PER LOOP * 6 = 1200 MS +
: CHECK FOR RESPONSE
:
: GO IF RESPONSE
: TRY AGAIN
: DECREMENT LOOP COUNT
: 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
:
: E30A: MOV AL,0EH
: OUT MFG_PORT,AL
:
: CHECKPOINT 0E
:

```

SECTION 5



```

1676
1677
1678          |----- FILL REGEN AREA WITH BLANK
1679 045F 33 FF          Z_3: XOR    DI,DI          | SET UP POINTER FOR REGEN
1680 0461 B8 B000        MOV    AX,0B000H       | SET UP ES TO VIDEO REGEN
1681 0464 E8 C0          MOV    ES,AX
1682
1683 0466 B9 0800        MOV    CX,2048        | NUMBER OF WORDS IN MONOCHROME CARD
1684 0469 B8 0720        MOV    AX,1 *7*H      | FILL CHARACTER FOR ALPHA + ATTRIBUTE
1685 046C F3/ AB        REP    STOSW          | FILL THE REGEN BUFFER WITH BLANKS
1686
1687 046E 33 FF          XOR    DI,DI          | CLEAR COLOR VIDEO BUFFER MEMORY
1688 0470 B8 B000        MOV    BX,0B000H     | SET UP ES TO COLOR VIDEO MEMORY
1689 0473 E8 C3          MOV    ES,BX
1690 0475 B9 2000        MOV    CX,8192
1691 0478 F3/ AB        REP    STOSW          | FILL WITH BLANKS
1692
1693          |----- ENABLE VIDEO AND CORRECT PORT SETTING
1694
1695 047A BA 03B8        MOV    DX,3BBH
1696 047D B0 29          MOV    AL,29H
1697 047F EE            OUT    DX,AL          | SET VIDEO ENABLE PORT
1698
1699          |----- SET UP OVERSCAN REGISTER
1700
1701 0480 42            INC    DX              | SET OVERSCAN PORT TO A DEFAULT
1702 0481 B0 30          MOV    AL,30H         | VALUE 30H FOR ALL MODES EXCEPT 640X200
1703 0483 EE            OUT    DX,AL         | OUTPUT THE CORRECT VALUE TO 3D9 PORT
1704
1705          |----- ENABLE COLOR VIDEO AND CORRECT PORT SETTING
1706
1707 048A BA 03D8        MOV    DX,3D8H
1708 048B B0 28          MOV    AL,28H
1709 0489 EE            OUT    DX,AL         | SET VIDEO ENABLE PORT
1710
1711          |----- SET UP OVERSCAN REGISTER
1712
1713 048A 42            INC    DX              | SET OVERSCAN PORT TO A DEFAULT
1714 048B B0 30          MOV    AL,30H         | VALUE 30H FOR ALL MODES EXCEPT 640X200
1715 048D EE            OUT    DX,AL         | OUTPUT THE CORRECT VALUE TO 3D9 PORT
1716
1717          |----- DISPLAY FAILING CHECKPOINT AND
1718
1719 048E 8C C8          MOV    AX,CS          | SET STACK SEGMENT TO CODE SEGMENT
1720 0490 8E D0          MOV    SS,AX
1721
1722 0492 B8 B000        MOV    BX,0B000H
1723 0495 8E DB          MOV    DS,BX          | SET DS TO B/W DISPLAY BUFFER
1724
1725 0497 B0 30          MOV    AL,'0'         | DISPLAY BANK 000000
1726 0499 B9 0006        SUB    CX,6           | START AT 0
1727 049C 2B FF          MOV    DI,DI          | WRITE TO DISPLAY REGEN BUFFER
1728 049E B8 05          MOV    [DI],AL        | POINT TO NEXT POSITION
1729 04A0 47            INC    DI
1730 04A1 47            INC    DI
1731 04A2 E2 FA        LODP  Z
1732
1733 04A4 80 FF B8        CMP    BH,0BH         | CHECK THAT COLOR BUFFER WRITTEN
1734 04A7 74 0C          JZ     Z_1            | POINT TO START OF BUFFER
1735 04A9 2B FF          SUB    DT,DI
1736
1737 04AB B7 B0          MOV    BH,0BH
1738 04AD 8E C3          MOV    ES,BX          | ES = MONOCHROME
1739 04AF B7 B8          MOV    BH,0BH        | SET SEGMENT TO COLOR
1740 04B1 8E D8          MOV    DS,BX          | DS = COLOR
1741 04B3 EB E2          JMP    Z_0
1742
1743          |----- PRINT FAILING BIT PATTERN
1744
1745 04B5 B0 20          Z_1: MOV    AL,' '         | DISPLAY A BLANK
1746 04B7 B8 05          MOV    [DI],AL        | WRITE TO COLOR BUFFER
1747 04B9 26: B8 05      MOV    ES:[DI],AL     | WRITE TO MONOCHROME REGEN BUFFER
1748 04BC 47            INC    DI              | POINT TO NEXT POSITION
1749 04BD 47            INC    DI
1750 04BE E4 81          IN     AL,MFG_PORT+1 | GET THE HIGH BYTE OF FAILING PATTERN
1751 04C0 B1 04          MOV    CL,4           | SHIFT COUNT
1752 04C2 D2 E8          SHR    AL,CL          | NIBBLE SWAP
1753 04C4 BC 057F R      MOV    SP,OFFSET Z1_0
1754 04C7 EB 18          JMP    SHORT PR
1755
1756 04C9 E4 81          Z1:  IN     AL,MFG_PORT+1 | ISOLATE TO LOW NIBBLE
1757 04CB 24 0F          AND    AL,0FH
1758 04CD BC 05B1 R      MOV    SP,OFFSET Z2_0
1759 04D0 EB 12          JMP    SHORT PR
1760 04D2 E4 82          Z2:  IN     AL,MFG_PORT+2 | GET THE HIGH BYTE OF FAILING PATTERN
1761 04D4 B1 04          MOV    CL,4           | SHIFT COUNT
1762 04D6 D2 E8          SHR    AL,CL          | NIBBLE SWAP
1763 04D8 BC 05B3 R      MOV    SP,OFFSET Z3_0
1764 04DB EB 07          JMP    SHORT PR
1765 04DD E4 82          Z3:  IN     AL,MFG_PORT+2 | ISOLATE TO LOW NIBBLE
1766 04DF 24 0F          AND    AL,0FH
1767 04E1 BC 05B5 R      MOV    SP,OFFSET Z4_0 | RETURN TO Z4:
1768
1769          |----- CONVERT AND PRINT
1770
1771 04E4 04 90          PR:  ADD    AL,090H       | CONVERT 00-0F TO ASCII CHARACTER
1772 04E6 27            DAA                    | ADD FIRST CONVERSION FACTOR
1773 04E7 14 40          ADC    AL,040H       | ADJUST FOR NUMERIC AND ALPHA RANGE
1774 04E9 27            DAA                    | ADD CONVERSION AND ADJUST LOW NIBBLE
1775
1776 04EA B8 05          MOV    [DI],AL        | WRITE TO COLOR BUFFER
1777 04EC 26: B8 05      MOV    ES:[DI],AL     | WRITE TO MONOCHROME BUFFER
1778 04EF 47            INC    DI              | POINT TO NEXT POSITION
1779 04F0 47            INC    DI
1780 04F1 C3            RET
1781
1782          |----- DISPLAY 201 ERROR
1783
1784 04F2 B0 20          Z4:  MOV    AL,' '         | DISPLAY A BLANK
1785 04F4 B8 05          MOV    [DI],AL        | WRITE TO DISPLAY REGEN BUFFER
1786 04F6 26: B8 05      MOV    ES:[DI],AL     | WRITE TO MONOCHROME BUFFER
1787 04F9 47            INC    DI              | POINT TO NEXT POSITION
1788 04FA 47            INC    DI
1789 04FB B0 32          MOV    AL,'2'         | DISPLAY 201 ERROR
    
```

SECTION 5





```

1904
1905 05A4 B0 11          MOV     AL,11H          ;
1906 05A6 E6 80          OUT     MFG_PORT,AL    ;
1907
1908
1909
1910 05A8 32 DB          ;----- VERIFY SPEED/REFRESH CLOCK RATES ( ERROR = 1 LONG AND 1 SHORT BEEP )
1911 05AA 33 C9          XOR     BL,BL          ; CLEAR REFRESH CYCLE REPEAT COUNT
1912
1913 05AC                  XOR     CX,CX          ; INITIALIZE SPEED RATE REGISTER
1914 05AC E4 61          EVEN                    ; PLACE ON EVEN WORD BOUNDARY
1915 05AE A8 10          C34:  IN     AL,PORT_B    ; READ REFRESH BIT REGISTER
1916 05B0 E1 FA          TEST   AL,REFRESH_BIT ; MASK FOR BIT
1917 05B2                  LOOPZ  C34             ; DECREMENT LOOP COUNTER TILL ON
1918 05B2 E4 61          C35:  IN     AL,PORT_B    ; READ REFRESH BIT REGISTER
1919 05B4 A8 10          TEST   AL,REFRESH_BIT ; MASK FOR BIT
1920 05B6 E0 FA          LOOPNZ C35            ; DECREMENT LOOP COUNTER TILL OFF
1921
1922 05B8 FE CB          DEC     BL             ; DECREMENT REFRESH CYCLE REPEAT COUNT
1923 05BA 75 F0          JNZ    C34            ; REPEAT TILL CYCLE COUNT DONE
1924
1925 05BC 81 F9 F780      CMP     CX,RATE_UPPER ; CHECK FOR RATE BELOW UPPER LIMIT
1926 05C0 73 07          JAE    C36E           ; SKIP ERROR BEEP IF BELOW MAXIMUM
1927 05C2
1928 05C2 BA 0101        C36E: MOV     DX,0101H      ; GET BEEP COUNTS FOR REFRESH ERROR
1929 05C5 E8 0000 E      MOV     ERR_BEEP      ; CALL TO POST ERROR BEEP ROUTINES
1930 05C8 F4             HLT                    ; HALT SYSTEM - BAD REFRESH RATE
1931 05C9
1932 05C9 81 F9 F9FD      C36:  CMP     CX,RATE_LOWER ; CHECK FOR RATE ABOVE LOWER LIMIT
1933 05CD 77 F3          JA     C36E           ; GO TO ERROR BEEP IF BELOW MINIMUM
1934
1935
1936
1937 05CF E4 82          ;----- GET THE INPUT BUFFER (SWITCH SETTINGS)
1938 05D1 24 F8          IN     AL,DMA_PAGE+1  ; GET THE SWITCH SETTINGS
1939 05D3 A2 0012 R      AND     AL,KEY_BD_INHIB+DSP_JMP+MFG_LOOP+BASE_MEM+BASE_MEM8 ; STRIP BITS
1940 05D6 2A C0          MOV     MFG_TST,AL    ; SAVE SETTINGS
1941 05D8 E6 82          SUB     AL,AL          ; RESET DMA_PAGE
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955 05DA 0F 01 E0        ;----- VERIFY STATUS INDICATE COMPATIBILITY (REAL) MODE
1956 05DD A9 000F      +     DB     00FH,001H,0E0H ; GET THE CURRENT STATUS WORD
1957 05E0 75 34          TEST   AX,0FH         ; PE/MP/EM/TS BITS SHOULD BE ZERO
1958
1959
1960
1961 05E2 B0 12          ;----- TEST PROTECTED MODE REGISTERS
1962 05E4 E6 80          MOV     AL,12H        ;
1963
1964 05E6 1E             OUT     MFG_PORT,AL    ;
1965
1966 05E6 1E             PUSH   DS             ; SET ES TO SAME SEGMENT AS DS
1967 05E7 07             POP     ES            ;
1968 05E8 BF D0A0        MOV     DI,SYS_IDT_LOC ; USE THIS AREA TO BUILD TEST PATTERN
1969 05EB B9 0003        MOV     CX,3          ;
1970 05EE B8 AAAA        MOV     AX,0AAAAH    ; FIRST PATTERN
1971 05F1 E8 0619 R      CALL   WRT_PAT        ;
1972 05F4 B8 5555        MOV     AX,05555H    ; WRITE NEXT PATTERN
1973 05F7 E8 0619 R      CALL   WRT_PAT        ;
1974 05FA 2B C0          SUB     AX,AX         ; WRITE 0
1975 05FC E8 0619 R      CALL   WRT_PAT
1976
1977 05FF FD          ;----- TEST 286 CONTROL FLAGS
1978 0600 9C          STD     PUSHF          ; SET DIRECTION FLAG FOR DECREMENT
1979 0601 58          POP     AX             ; GET THE FLAGS
1980 0602 A9 0200        TEST   AX,0200H      ; INTERRUPT FLAG SHOULD BE OFF
1981 0605 75 0F          JNZ    ERR_PROT       ; GO IF NOT
1982 0607 A9 0400        TEST   AX,0400H      ; CHECK DIRECTION FLAG
1983 060A 74 0A          JZ     ERR_PROT       ; GO IF NOT SET
1984 060C FC          CLD                    ; CLEAR DIRECTION FLAG
1985 060D 9C          PUSHF                    ; INSURE DIRECTION FLAG IS RESET
1986 060E 58          POP     AX             ;
1987 060F A9 0400        TEST   AX,0400H      ;
1988 0612 75 02          JNZ    ERR_PROT       ; GO IF NOT
1989
1990 0614 EB 3D          JMP     SHORT C37A     ; TEST OK CONTINUE
1991 0616
ERR_PROT: HLT                    ; PROTECTED MODE REGISTER FAILURE
1992 0616 F4          JMP     SHORT ERR_PROT ; INSURE NO BREAKOUT OF HALT
1993 0617 EB FD
1994
1995
1996
1997 0619 B9 0003        ;----- WRITE TO 286 REGISTERS
1998 061C F3/ AB        WRT_PAT:MOV    CX,3      ; STORE 6 BYTES OF PATTERN
1999 061E BD D0A0        REP     STOSW          ;
2000
2001 0621 26          MOV     BP,SYS_IDT_LOC ; LOAD THE IDT
2002
2003 0622 0F          SEGOV  ES             ;
2004 0623                LIDT   [BP]           ; REGISTER FROM THIS AREA
2005 0623 01          DB     026H           ;
2006 0626                LABEL BYTE           ;
2007 0623                + ?70001 LABEL BYTE  ;
2008 0623 01          +     DB     00FH      ;
2009 0626                +     LABEL BYTE      ;
2010 0626 BD D0A0        +     MOV     BP,SYS_IDT_LOC ;
2011
2012 0629 26          +     DB     026H      ; LOAD THE GDT
2013
2014 062A 0F          +     LIDT   [BP]     ; FROM THE SAME AREA
2015 062B 01          +     DB     00FH      ;
2016 062B 01          +     LABEL BYTE      ;
2017 062E                + ?70004 LABEL BYTE  ;
2018
2019 062B 8B 56 00      +     MOV     DX,WORD PTR [BP] ;
2020
2021 062E                + ?70005 LABEL BYTE  ;

```

SECTION 5





```

2246
2247 075F BC 0000      MOV     SP,POST_SS      ; SET STACK FOR SYSINIT1
2248 0762 BE D4        MOV     SS,SP
2249 0764 BC 8000      MOV     SP,POST_SP
2250 0767 E8 0000 E    CALL    SYSINIT1       ; CALL THE DESCRIPTOR TABLE BUILDER
2251                                       ; AND REAL-TO-PROTECTED MODE SWITCHER
2252
2253 076A B0 IA        MOV     AL,IAH          ;
2254 076C E6 80        OUT     MFG_PORT,AL    ;
2255                                       ;
2256                                       ;
2257                                       ;
2258 076E 6A 08        PUSH    BYTE PTR GDT_PTR ; SET (DS:) SELECTOR TO GDT SEGMENT
2259 0770 IF          POP     DS
2260 0771 C7 06 006A 0000 MOV     DS:SS_TEMP.BASE_LO_WORD,0
2261 0777 C6 06 006C 00 MOV     BYTE PTR DS:(SS_TEMP.BASE_HI_BYTE),0
2262 077C BE 0058      MOV     SI,SS_TEMP
2263 077F BE D6        MOV     SS,SI
2264 0781 BC FFFF      MOV     SP,MAX_SEG_LEN-2
2265
2266                                       ;
2267                                       ;
2268                                       ;
2269                                       ;
2270                                       ;
2271                                       ;
2272                                       ;
2273                                       ;
2274                                       ;
2275                                       ;
2276                                       ;
2277                                       ;
2278                                       ;
2279                                       ;
2280                                       ;
2281                                       ;
2282                                       ;
2283 0784 0F 01 E0      SMSW   AX              ; GET THE MACHINE STATUS WORD
2284 0787 A9 0001      DB     00FH,001H,0E0H
2285 078A 75 0C        TEST   AX,VIRTUAL_ENABLE ; ARE WE IN PROTECTED MODE
2286                                       ;
2287                                       ;
2288 078C BB 080F      JNZ   VIR_OK          ;
2289 0792 E9 0000 E    JMP    PROC_SHUTDOWN  ; CAUSE A SHUTDOWN
2290
2291                                       ;
2292                                       ;
2293 0795 F4          SHUT8: HLT            ; ERROR HALT
2294 0796 EB FD        JMP    SHUT8
2295
2296                                       ;
2297                                       ;
2298 0798 C7 06 0048 FFFF VIR_OK: MOV     DS:ES_TEMP.SEG_LIMIT,MAX_SEG_LEN
2299
2300                                       ;
2301                                       ;
2302 079E C6 06 004D 93 MOV     BYTE PTR DS:(ES_TEMP.DATA_ACC_RIGHTS),CPL0_DATA_ACCESS
2303
2304                                       ;
2305                                       ;
2306 07A3 C6 06 004A 01 MOV     BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE),01H
2307 07A8 C7 06 004A 0000 MOV     DS:ES_TEMP.BASE_LO_WORD,0H
2308
2309 07AE B0 IB        MOV     AL,IBH          ;
2310 07B0 E6 80        OUT     MFG_PORT,AL    ;
2311                                       ;
2312 07B2 BB 0040      MOV     BX,16*4         ; SET THE FIRST 64K DONE
2313
2314                                       ;
2315                                       ;
2316 07B5              NOT_DONE:
2317 07B5 6A 48        PUSH    BYTE PTR ES_TEMP ; POINT ES TO DATA
2318 07B7 07          POP     ES              ; POINT TO SEGMENT TO TEST
2319 07B8 E8 07D4 R    CALL   HOW_BIG          ; DO THE FIRST 64K
2320 07BD 74 03        JZ     NOT_FIN          ; CHECK IF TOP OF MEMORY
2321 07BD E9 0872 R    JMP    DONE             ; CHECK IF TOP OF MEMORY
2322
2323 07C0              NOT_FIN:
2324 07C0 83 C3 40      ADD     BX,16*4         ; BUMP MEMORY COUNT BY 64K
2325
2326                                       ;
2327                                       ;
2328                                       ;
2329                                       ;
2330                                       ;
2331                                       ;
2332 07C7 80 3E 004C 0A CMP     BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE),0AH
2333 07CC 75 E7        JNZ   NOT_DONE         ; GO IF NOT
2334 07CE EB 084F R    CALL   HOW_BIG_END     ; GO SET MEMORY SIZE
2335 07D1 E9 0872 R    JMP    DONE
2336
2337                                       ;
2338                                       ;
2339 07D4              HOW_BIG:
2340 07D4 2B FF        SUB     DI,DI           ; TEST PATTERN
2341 07D6 BB AA55      MOV     AX,0AA55H      ; SAVE PATTERN
2342 07D9 BB C5        MOV     CX,AX           ; WRITE PATTERN
2343 07DB E6 89 05      MOV     ES:[DI],AX     ; WRITE PATTERN TO MEMORY
2344 07DE B0 0F        MOV     AL,0FH         ; PUT SOMETHING IN AL
2345 07E0 E6 8B 05      MOV     AX,ES:[DI]     ; GET PATTERN
2346 07E3 E6 89 05      MOV     ES:[DI],AX     ; INSURE NO PARITY I/O CHECK
2347 07E6 33 C1        XOR     AX,CX           ; COMPARE PATTERNS
2348 07E8 75 65        JNZ   HOW_BIG_END     ; GO END IF NO COMPARE
2349
2350 07EA 1E          PUSH    DS              ; POINT TO SYSTEM DATA AREA
2351 07EB 6A 18        PUSH   BYTE PTR RSDA_PTR ; GET (DS:)
2352 07ED IF          POP     DS
2353
2354                                       ;
2355                                       ;
2356 07EE 81 3E 0072 R 1234 CMP     #RESET_FLAG,1234H ; SOFT RESET
2357 07F4 IF          POP     DS              ; RESTORE DS
2358 07F5 75 36        JNZ   HOW_BIG_2        ; GO IF NOT SOFT RESET
2359
    
```



```

2474 0891 83 C3 40      DONE1: ADD    BX,16*4          ; BUMP MEMORY COUNT BY 64K
2475 0892 83 C3 40      ;----- DO NEXT 64K (XX0000) BLOCK
2476 0893 83 C3 40      ;----- DO NEXT 64K (XX0000) BLOCK
2477 0894 FE 06 004C     INC    BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE)
2478 0895 FE 06 004C     ;----- CHECK FOR TOP OF MEMORY (FE0000)
2479 0896 FE 06 004C     ;----- CHECK FOR TOP OF MEMORY (FE0000)
2480 0897 FE 06 004C     ;----- CHECK FOR TOP OF MEMORY (FE0000)
2481 0898 FE 06 004C     ;----- CHECK FOR TOP OF MEMORY (FE0000)
2482 0898 80 3E 004C FE  CMP    BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE),0FEH ; LAST OF MEMORY?
2483 0899 75 E7         JNZ    NOT_DONE1          ; GO IF NOT
2484 089F E8 0919 R     CALL   HOW_BIG_END1       ; GO SET MEMORY SIZE
2485 08A2 E9 092C R     JMP    DONE1
2486 08A3 E9 092C R     ;----- FILL/CHECK LOOP
2487 08A4 E9 092C R     ;----- FILL/CHECK LOOP
2488 08A5 E9 092C R     ;----- FILL/CHECK LOOP
2489 08A5 2B FF         HOW_BIG1: SUB    DI,DI
2490 08A5 2B FF         MOV    AX,0AA55H          ; TEST PATTERN
2491 08A7 B8 A855         MOV    CX,AX              ; SAVE PATTERN
2492 08AA 9B C8         MOV    ES:[DI],AX         ; SEND PATTERN TO MEMORY
2493 08AC 26: 89 05     MOV    AL,0FH             ; PUT SOMETHING IN AL
2494 08AF 80 0F         MOV    AX,ES:[DI]         ; GET PATTERN
2495 08B1 26: 8B 05     MOV    AX,ES:[DI]         ; GET PATTERN
2496 08B4 26: 89 05     MOV    ES:[DI],AX         ; INSURE NO PARITY I/O CHECK
2497 08B7 33 C1         XOR    AX,CX              ; COMPARE PATTERNS
2498 08B9 75 5E         JNZ    HOW_BIG_END1       ; GO END IF NO COMPARE
2499 08BA 75 5E         ;----- IS THIS A SOFT RESET
2500 08BB 75 5E         ;----- IS THIS A SOFT RESET
2501 08BC 75 5E         ;----- IS THIS A SOFT RESET
2502 08BB 1E           PUSH   DS                 ; POINT TO SYSTEM DATA AREA
2503 08BC 6A 18         PUSH   BYTE PTR RSDA_PTR
2504 08BE 1F           POP    DS
2505 08BF 81 3E 0072 R 1234  CMP    WORD PTR DS,1234H  ; SOFT RESET
2506 08C5 1F           POP    DS                 ; RESTORE DS
2507 08C6 75 2F         JNZ    HOW_BIG_2A        ; GO IF NOT SOFT RESET
2508 08C7 75 2F         ;----- CHECK PARITY WITH PARITY BITS OFF
2509 08C8 75 2F         ;----- CHECK PARITY WITH PARITY BITS OFF
2510 08C9 75 2F         ;----- CHECK PARITY WITH PARITY BITS OFF
2511 08CB 26: C7 05 0101  MOV    WORD PTR ES:[DI],0101H ; TURN OFF BOTH PARITY BITS
2512 08CD 6A FF         PUSH   BYTE PTR OFFH      ; PLACE OFFFH IN STACK (BUS BITS ON)
2513 08CF 58           POP    AX                 ; DELAY - CAUSING BUS BITS ON
2514 08D0 26: 8B 05     MOV    AX,ES:[DI]         ; CHECK PARITY
2515 08D1 26: 8B 05     ;----- CHECK PARITY WITH PARITY BITS ON
2516 08D3 E4 61         IN     AL,PORT_B          ; CHECK FOR PLANAR OR I/O PARITY CHECK
2517 08D5 24 C0         AND    AL,PARITY_ERR     ; CHECK FOR PLANAR OR I/O PARITY CHECK
2518 08D7 26: 89 05     MOV    ES:[DI],AX         ; CLEAR POSSIBLE PARITY ERROR
2519 08DA 75 3D         JNZ    HOW_BIG_END1       ; GO IF PLANAR OR I/O PARITY CHECK
2520 08DB 75 3D         ;----- CHECK ALL BITS
2521 08DC 75 3D         ;----- CHECK ALL BITS
2522 08DD 75 3D         ;----- CHECK ALL BITS
2523 08DC 26: C7 05 FFFF  MOV    WORD PTR ES:[DI],FFFFH ; TURN ON ALL BITS
2524 08E1 6A 00         PUSH   BYTE PTR 0         ; PLACE 0000H IN STACK (BUS BITS OFF)
2525 08E3 58           POP    AX                 ; DELAY - CAUSING BUS BITS OFF
2526 08E4 26: 8B 05     MOV    AX,ES:[DI]         ; CHECK FOR FFFFH
2527 08E7 80           POP    AX                 ; SAVE RESULTS
2528 08E8 E4 61         IN     AL,PORT_B          ; CHECK FOR PLANAR OR I/O PARITY CHECK
2529 08EA 24 C0         AND    AL,PARITY_ERR     ; CHECK FOR PLANAR OR I/O PARITY CHECK
2530 08EC 26: 89 05     MOV    ES:[DI],AX         ; CLEAR POSSIBLE PARITY ERROR
2531 08EF 58           POP    AX                 ; GET RESULTS
2532 08F0 75 27         JNZ    HOW_BIG_END1       ; GO IF PLANAR OR I/O PARITY CHECK
2533 08F2 3D FFFF     CMP    AX,FFFFH          ; CHECK FOR PLANAR OR I/O PARITY CHECK
2534 08F5 75 22         JNZ    HOW_BIG_END1
2535 08F6 75 22         ;----- CLEAR 64K BLOCK OF MEMORY
2536 08F7 75 22         ;----- CLEAR 64K BLOCK OF MEMORY
2537 08F8 75 22         ;----- CLEAR 64K BLOCK OF MEMORY
2538 08F7 2B C0         HOW_BIG_2A: SUB    AX,AX              ; WRITE ZEROS
2539 08F7 2B C0         MOV    CX,2000H*4        ; SET COUNT FOR 32K WORDS
2540 08F9 B9 8000     REP    STOSW              ; FILL 32K WORDS
2541 08FC F3/ AB         ;----- CHECK 64K BLOCK FOR PARITY CHECK (VALID TEST DURING SOFT RESET ONLY)
2542 08FD 75 22         ;----- CHECK 64K BLOCK FOR PARITY CHECK (VALID TEST DURING SOFT RESET ONLY)
2543 08FE 75 22         ;----- CHECK 64K BLOCK FOR PARITY CHECK (VALID TEST DURING SOFT RESET ONLY)
2544 08FE 1E           PUSH   DS                 ; GET ES TO DS
2545 08FF 06           PUSH   ES
2546 08FF 06           PUSH   ES
2547 0900 06           POP    DS
2548 0901 1F           POP    DS
2549 0902 B9 8000     MOV    CX,2000H*4        ; SET COUNT FOR 32K WORDS
2550 0905 2B F6         SUB    SI,SI
2551 0907 F3/ AD     REP    LODSW
2552 0909 2B FF     SUB    DI,DI              ; SET TO BEGINNING OF BLOCK
2553 090B E4 61         IN     AL,PORT_B          ; CHECK FOR PLANAR OR I/O PARITY CHECK
2554 090D 24 C0         AND    AL,PARITY_ERR     ; CHECK FOR PLANAR OR I/O PARITY CHECK
2555 090F 26: C7 05 0000  MOV    WORD PTR ES:[DI],0  ; CLEAR POSSIBLE PARITY ERROR
2556 0914 07           POP    ES                 ; RESTORE SEGMENT
2557 0915 1F           POP    DS
2558 0916 75 01         JNZ    HOW_BIG_END1       ; GO IF PLANAR OR I/O PARITY CHECK
2559 0917 75 01         ;----- TEST ADDRESS LINES 19 - 23
2560 0918 C3         RET
2561 0919 C3         ;----- TEST ADDRESS LINES 19 - 23
2562 0919 C3         ;----- TEST ADDRESS LINES 19 - 23
2563 0919 B0 1E         HOW_BIG_END1: MOV    AL,1EH             ; 0000000000000000
2564 091B E6 50         OUT    MFG_PORT,AL       ; 00 CHECKPOINT 1E 00
2565 091C E6 50         ;----- SET EXPANSION MEMORY SIZE DETERMINED IN CMOS
2566 091D E6 50         ;----- SET EXPANSION MEMORY SIZE DETERMINED IN CMOS
2567 091E E6 50         ;----- SET EXPANSION MEMORY SIZE DETERMINED IN CMOS
2568 091D B0 80         MOV    AL,CMOS_U_M_S_LO+NM1 ; ADDRESS LOW BYTE
2569 091F 8A E3         MOV    AH,BL             ; GET LOW MEMORY SIZE
2570 0921 E8 0000 E     CALL   CMOS_WRITE        ; SET LOW BYTE
2571 0924 B0 B1         MOV    AL,CMOS_U_M_S_HI+NM1 ; ADDRESS HI BYTE
2572 0926 8A E7         MOV    AH,BH             ; GET THE HIGH MEMORY SIZE
2573 0928 E8 0000 E     CALL   CMOS_WRITE        ; PLACE IN CMOS
2574 092B C3         RET
2575 092C C3         ;----- TEST ADDRESS LINES 19 - 23
2576 092D C3         ;----- TEST ADDRESS LINES 19 - 23
2577 092E C3         ;----- TEST ADDRESS LINES 19 - 23
2578 092C B0 1F         DONE1: MOV    AL,1FH             ; 0000000000000000
2579 092E E6 50         OUT    MFG_PORT,AL       ; 00 CHECKPOINT 1F 00
2580 0930 C6 06 004C 00  MOV    BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE),00H
2581 0932 2B FF         SUB    DI,DI              ; SET LOCATION POINTER TO ZERO
2582 0937 BA FFFF     MOV    DX,0FFFFH         ; WRITE FFFF AT ADDRESS 0
2583 093A E8 0969 R     CALL   SDO                ; SDO
2584 093D 2B D2         SUB    DX,DX              ; WRITE 0
2585 093E 2B D2         ;----- TEST ADDRESS LINES 19 - 23
2586 093F C6 06 004C 08  MOV    BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE),08H
2587 0944 E8 0969 R     CALL   SDO

```



```

2702
2703 0A04 81 C1 0080      ADD    CX,080H          ; POINT TO NEXT 2K BLOCK
2704 0A08 81 F9 C800      CMP    CX,0C800H       ; TOP OF VIDEO ROM AREA YET?
2705 0A0C 7C E3           JL     CHK_VIDEO1      ; TRY AGAIN
2706 0A0E 23 C9           AND    CX,CX           ; SET NON ZERO FLAG
2707 0A10                   CHK_VIDEO2:
2708 0A10 C3           RET                     ; RETURN TO CALLER
2709
2710 ;----- CMOS VIDEO BITS NON ZERO (CHECK FOR PRIMARY DISPLAY AND NO VIDEO ROM)
2711
2712 0A11                   MOS_OK_1:
2713 0A11 E8 09EE R        CALL   CHK_VIDEO       ; IS THE VIDEO ROM INSTALLED?
2714 0A14 74 26           JZ     BAD_MOS         ; WRONG CONFIGURATION IN CONFIG BYTE
2715
2716 0A16 8A C4           MOV    AL,AH           ; RESTORE CONFIGURATION
2717 0A18 F6 06 0012 R 40  TEST   #MFG_TST,DSP_JMP ; CHECK FOR DISPLAY JUMPER
2718 0A1D 74 0A           JZ     MOS_OK_2        ; GO IF COLOR CARD IS PRIMARY DISPLAY
2719
2720 ;----- MONOCHROME CARD IS PRIMARY DISPLAY (NO JUMPER INSTALLED)
2721
2722 0A1F 24 30           AND    AL,30H         ; INSURE MONOCHROME IS PRIMARY
2723 0A21 3C 30           CMP    AL,30H         ; CONFIGURATION OK?
2724 0A23 75 17           JNZ   BAD_MOS         ; GO IF NOT
2725 0A25 8A C4           MOV    AL,AH           ; RESTORE CONFIGURATION
2726 0A27 E8 08           JMP    SHORT MOS_OK    ; USE THE CONFIGURATION BYTE FOR DISPLAY
2727
2728 ;----- COLOR CARD
2729
2730 0A29                   MOS_OK_2:
2731 0A29 24 30           AND    AL,30H         ; STRIP UNWANTED BITS
2732 0A2B 3C 30           CMP    AL,30H         ; MUST NOT BE MONO WITH JUMPER INSTALLED
2733 0A2D 8A C4           MOV    AL,AH           ; RESTORE CONFIGURATION
2734 0A2F 74 0B           JZ     BAD_MOS         ; GO IF YES
2735
2736 ;----- CONFIGURATION MUST HAVE AT LEAST ONE DISKETTE
2737
2738 0A31 A8 01           MOS_OK: TEST   AL,01H       ; MUST HAVE AT LEAST ONE DISKETTE
2739 0A33 75 26           JNZ   NORMAL_CONFIG   ; GO SET CONFIGURATION IF OK
2740 0A35 F6 06 0012 R 40  TEST   #MFG_TST,MFG_LOOP ; EXCEPT IF MFG JUMPER IS INSTALLED
2741 0A3A 74 1F           JZ     NORMAL_CONFIG   ; GO IF INSTALLED
2742
2743 ;----- MINIMUM CONFIGURATION WITH BAD CMOS OR NON VALID VIDEO
2744
2745 0A3C                   BAD_MOS:
2746 0A3C 8B 008E         MOV    AX,CMOS_DIAG+MMI ; GET THE DIAGNOSTIC STATUS
2747 0A3F E8 0900 E        CALL   CMOS_READ       ; CMOS READ
2748 0A42 A8 C0           TEST   AL,BAD_BAT+BAD_CKSUM ; WAS BATTERY DEFECTIVE OR BAD CHECKSUM
2749 0A44 75 03           JNZ   BAD_MOS1        ; GO IF YES
2750
2751 0A46 E8 0000 E        CALL   CONFIG_BAD      ; SET THE MINIMUM CONFIGURATION FLAG
2752 0A49
2753 0A49 E8 09EE R        CALL   CHK_VIDEO       ; CHECK FOR VIDEO ROM
2754 0A4C 80 01           MOV    AL,01H         ; DISKETTE ONLY
2755 0A4E 74 0B           JZ     NORMAL_CONFIG   ; GO IF VIDEO ROM PRESENT
2756
2757 0A50 F6 06 0012 R 40  TEST   #MFG_TST,DSP_JMP ; CHECK FOR DISPLAY JUMPER
2758 0A55 80 11           MOV    AL,11H         ; DEFAULT TO 48X25 COLOR
2759 0A57 74 02           JZ     NORMAL_CONFIG   ; GO IF JUMPER IS INSTALLED
2760
2761 0A59 80 31           MOV    AL,31H         ; DISKETTE / B/W DISPLAY 80X25
2762
2763 ;-----
2764 ;----- CONFIGURATION AND MFG MODE
2765 ;-----
2766
2767 0A5B                   NORMAL_CONFIG:
2768 0A5B F6 06 0012 R 20  TEST   #MFG_TST,MFG_LOOP ; IS THE MANUFACTURING JUMPER INSTALLED
2769 0A60 75 02           JNZ   NORM1           ; GO IF NOT
2770 0A62 24 3E           AND    AL,03EH        ; STRIP DISKETTE FOR MFG TEST
2771
2772 0A64 2A E4           NORM1: SUB    AH,AH           ; SAVE SWITCH INFORMATION
2773 0A66 A3 0010 R        MOV    #REQ1IP_FLAG,AX ; BYPASS IF SOFT RESET
2774 0A69 81 3E 0072 R 1234  CMP    #RESET_FLAG,1234H
2775 0A6F 74 2C           JZ     E6
2776
2777 ;----- GET THE FIRST SELF TEST RESULTS FROM KEYBOARD
2778
2779 0A71 80 60           MOV    AL,WRITE_8042_LOC ; ENABLE KEYBOARD
2780 0A73 E8 039D R        CALL   C8042           ; ISSUE WRITE BYTE COMMAND
2781 0A76 80 4D           MOV    AL,4DH         ; ENABLE OUTPUT BUFFER FULL INTERRUPT,
2782 ; SET SYSTEM FLAG, PC 1 COMPATIBILITY,
2783 ; INHIBIT OVERRIDE, ENABLE KEYBOARD
2784
2785 0A7A 2B C9           SUB    CX,CX           ; WAIT FOR COMMAND ACCEPTED
2786 0A7C E8 03A2 R        CALL   C42_1
2787
2788 0A7F B9 7FFF         MOV    CX,07FFFH      ; SET LOOP COUNT FOR APPROXIMATELY 100MS
2789 ; TO RESPOND
2790 0A82 E4 64           TST6: IN    AL,STATUS_PORT ; WAIT FOR OUTPUT BUFFER FULL
2791 0A84 A8 01           TEST   AL,OUT_BUF_FULL
2792 0A86 E1 FA           LOOPZ  TST6           ; TRY AGAIN IF NOT
2793
2794 0A88 9C           PUSHF
2795 0A89 80 AD           MOV    AL,DIS_KBD     ; SAVE FLAGS
2796 0A8B E8 039D R        CALL   C8042           ; DISABLE KEYBOARD
2797 0A8E 9D           POPF
2798 0A8F 74 0C           JZ     E6             ; ISSUE THE COMMAND
2799 ; RESTORE FLAGS
2800 ; CONTINUE WITHOUT RESULTS
2800 0A91 E4 60           IN    AL,PORT_A       ; GET INPUT FROM KEYBOARD
2801 0A93 A2 0072 R        MOV    BYTE PTR #RESET_FLAG,AL ; TEMPORARY SAVE FOR AA RECEIVED
2802
2803 ;----- CHECK FOR MFG REQUEST
2804
2805 0A96 3C 65           CMP    AL,065H        ; LOAD MANUFACTURING TEST REQUEST?
2806 0A98 75 03           JNE   E6             ; CONTINUE IF NOT
2807 0A9A E9 0C34 R        JMP    MFG_BOOT       ; ELSE GO TO MANUFACTURING BOOTSTRAP

```







```

3036 0C00 81 26 0010 R FFCF      AND    @EQUIP_FLAG,OFFCFH    ; TURN OFF VIDEO BITS
3037 0C06 81 0E 0010 R 0010      OR     @EQUIP_FLAG,10H      ; SET COLOR 40X24
3038 0C0C B0 01                MOV    AL,01H
3039 0C0E 2A E4                SUB    AH,AH
3040 0C10 CD 10                INT    INT_VIDEO
3041 0C12                        E17_1:
3042 0C12 88                POP    AX                    ; SET NEW VIDEO TYPE ON STACK
3043 0C13 A1 0010 R          MOV    AX,@EQUIP_FLAG
3044 0C14 24 30                AND    AL,30H
3045 0C18 3C 30                CMP    AL,30H                ; IS IT THE B/W?
3046 0C1A 2A C0                SUB    AL,AL
3047 0C1C 74 02                JZ     E17_2                ; GO IF YES
3048 0C1E FE C0                JNC   AL                    ; INITIALIZE FOR 40X25
3049 0C20                        E17_2:
3050 0C20 50                PUSH   AX
3051 0C21                        E17_4:
3052 0C21 E9 0B4C R          JMP    E18
3053                                ;---- BOTH VIDEO CARDS FAILED SET DUMMY RETURN IF RETRACE FAILURE
3054
3055                                E17_3:
3056 0C24                        E17_3:
3057 0C24 1E                PUSH   DS                    ; SET DS SEGMENT TO 0
3058 0C26 2B C0                SUB    AX,AX
3059 0C27 8E D8                MOV    DS,AX
3060 0C29 BF 0040 R          MOV    DI,OFFSET @VIDEO_INT ; SET INTERRUPT 10H TO DUMMY
3061 0C2C C7 05 0000 E      MOV    WORD PTR [DI],OFFSET DUMMY_RETURN ; RETURN IF NO VIDEO CARD
3062 0C30 1F                POP    DS
3063 0C31 E9 0B51 R          JMP    E18_1                ; BYPASS REST OF VIDEO TEST
    
```

SECTION 5

```

3064 PAGE
3065 -----
3066 ; MANUFACTURING BOOT TEST CODE ROUTINE
3067 ; LOAD A BLOCK OF TEST CODE THROUGH THE KEYBOARD PORT FOR MANUFACTURING
3068 ; TESTS.
3069 ; THIS ROUTINE WILL LOAD A TEST (MAX LENGTH=FAFFH) THROUGH THE KEYBOARD
3070 ; PORT. CODE WILL BE LOADED AT LOCATION 000010500. AFTER LOADING,
3071 ; CONTROL WILL BE TRANSFERRED TO LOCATION 000010500. THE STACK WILL
3072 ; BE LOCATED AT 000010400. THIS ROUTINE ASSUMES THAT THE FIRST 2 BYTES
3073 ; TRANSFERRED CONTAIN THE COUNT OF BYTES TO BE LOADED
3074 ; (BYTE 1=COUNT LOW, BYTE 2=COUNT HI.)
3075 ;-----
3076 ;
3077 ;----- DEGATE ADDRESS LINE 20
3078
3079 MFG_BOOT:
3080 MOV AH,DISABLE_BIT20 ; DEGATE COMMAND FOR ADDRESS LINE 20
3081 CALL GATE_A20 ; ISSUE TO KEYBOARD ADAPTER AND CLI
3082
3083 ;----- SETUP HARDWARE INTERRUPT VECTOR TABLE LEVEL 0-7 AND SOFTWARE INTERRUPTS
3084
3085 PUSH ABS0 ; SET ES SEGMENT REGISTER TO ABS0
3086 POP ES
3087 MOV CX,24 ; GET VECTOR COUNT
3088 MOV AX,C5 ; GET THE CURRENT CODE SEGMENT VALUE
3089 MOV DS,AX ; SETUP DS SEGMENT REGISTER TO
3090 MOV SI,OFFSET VECTOR_TABLE ; POINT TO THE ROUTINE ADDRESS TABLE
3091 MOV DI,OFFSET @INT_PTR ; SET DESTINATION TO FIRST USED VECTOR
3092 MFG_B1:
3093 MOVS# ; MOVE ONE ROUTINE OFFSET ADDRESS
3094 STOS# ; INSERT CODE SEGMENT VALUE
3095 LOOP MFG_B1 ; MOVE THE NUMBER OF ENTRIES REQUIRED
3096
3097 ;----- SETUP HARDWARE INTERRUPT VECTORS LEVEL 8-15 (VECTORS START AT INT 70 H)
3098
3099 MOV CX,08 ; GET VECTOR COUNT
3100 MOV SI,OFFSET SLAVE_VECTOR_TABLE ; SETUP DS SEGMENT REGISTER TO
3101 MOV DI,OFFSET @SLAVE_INT_PTR ; SET DESTINATION TO FIRST USED VECTOR
3102 MFG_B2:
3103 MOVS# ; MOVE ONE ROUTINE OFFSET ADDRESS
3104 STOS# ; INSERT CODE SEGMENT VALUE
3105 LOOP MFG_B2
3106
3107 ;----- SET UP OTHER INTERRUPTS AS NECESSARY
3108
3109 ASSUME DS:ABS0,ES:ABS0
3110 PUSH ES ; ES= ABS0
3111 POP DS ; SET DS TO ABS0
3112 MOV WORD PTR @NMI_PTR,OFFSET NMI_INT ; NMI INTERRUPT
3113 MOV WORD PTR @INT5_PTR,OFFSET PRINT_SCREEN ; PRINT SCREEN
3114 MOV WORD PTR @BASIC_PTR+2,OF600H ; CASSETTE BASIC SEGMENT
3115
3116 ;----- ENABLE KEYBOARD PORT
3117
3118 MOV AL,60H ; WRITE 8042 MEMORY LOCATION 0
3119 CALL CB042 ; ISSUE THE COMMAND
3120 MOV AL,00001001B ; SET INHIBIT OVERRIDE/ENABLE OBF
3121 OUT PORT_A,AL ; INTERRUPT AND NOT PC COMPATIBLE
3122
3123 CALL MFG_B4 ; GET COUNT LOW
3124 MOV BH,AL ; SAVE IT
3125 CALL MFG_B4 ; GET COUNT HI
3126 MOV CH,AL
3127 MOV CL,BH ; CX NOW HAS COUNT
3128 CLD ; SET DIRECTION FLAG TO INCREMENT
3129 MOV DI,OFFSET @MFG_TEST_RTN ; SET TARGET OFFSET (DS=0000)
3130 MFG_B3:
3131 IN AL,STATUS_PORT ; GET 8042 STATUS PORT
3132 TEST AL,OUT_BUF_FULL ; KEYBOARD REQUEST PENDING?
3133 JZ MFG_B3 ; LOOP TILL DATA PRESENT
3134 IN AL,FORT_A ; GET DATA
3135 STOSB ; STORE IT
3136 OUT MFG_PORT,AL ; DISPLAY CHARACTER AT MFG PORT
3137 LOOP MFG_B3 ; LOOP TILL ALL BYTES READ
3138
3139 JMP @MFG_TEST_RTN ; FAR JUMP TO CODE THAT WAS JUST LOADED
3140
3141 MFG_B4:
3142 IN AL,STATUS_PORT ; CHECK FOR OUTPUT BUFFER FULL
3143 TEST AL,OUT_BUF_FULL ; HANG HERE IF NO DATA AVAILABLE
3144 LOOPZ MFG_B4
3145
3146 IN AL,PORT_A ; GET THE COUNT
3147 RET
3148
3149 POST1 ENDP
3150 CODE ENDS
3151

```



```
115  
116 0021 B0 FF          MOV    AL,OFFH           ; DISABLE DEVICE INTERRUPTS  
117 0023 E6 21          OUT    INTA01,AL        ; WRITE TO INTERRUPT MASK REGISTER  
118 0025 E6 A1          OUT    INTB01,AL       ; WRITE TO 2ND INTERRUPT MASK REGISTER  
119 0027 EB 00          JMP    $+2              ; I/O DELAY  
120 0029 E4 21          IN     AL,INTA01       ; READ INTERRUPT MASK REGISTER  
121 002B 8A E0          MOV    AH,AL           ; SAVE RESULTS  
122 002D E4 A1          IN     AL,INTB01      ; READ 2ND INTERRUPT MASK REGISTER  
123  
124 002F 05 0001       ADD    AX,1             ; ALL IMR BITS ON?  
125 0032 75 15         JNZ    D6               ; NO - GO TO ERR ROUTINE  
126  
127  
128 ;----- CHECK FOR HOT INTERRUPTS  
129 ;----- INTERRUPTS ARE MASKED OFF.  CHECK THAT NO INTERRUPTS OCCUR.  
130  
131 0034 A2 006B R      MOV    @INTR_FLAG,AL   ; CLEAR INTERRUPT FLAG  
132  
133 0037 B0 26          MOV    AL,26H          ;  
134 0039 E6 80          OUT    MFG_PORT,AL    ; @@@@@@@@@@@@@@@@@@@@  
135 ; @@@ CHECKPOINT 26 @@@  
136 003B FB            STI    CX,6628         ; ENABLE EXTERNAL INTERRUPTS  
137 003C B9 19E4       MOV    CX,6628         ; WAIT 100 MILLISECONDS FOR ANY  
138 003F E6 0000 E     CALL   WAITF           ; INTERRUPTS THAT OCCUR  
139 0042 80 3E 006B R 00  CMP    @INTR_FLAG,00H ; DID ANY INTERRUPTS OCCUR?  
140 0047 74 0D         JZ     DT              ; NO - GO TO NEXT TEST  
141  
142 0049 C6 06 0015 R 05 D6: MOV    @MFG_ERR_FLAG,05H ;  
143 ; @@@ CHECKPOINT 5 @@@  
144 004E BE 0000 E     MOV    SI,OFFSET E101 ; DISPLAY 101 ERROR  
145 0051 E6 0000 E     D6A: CALL   E_MSG         ;  
146 0054 FA            CL1  
147 0056 F4            HLT                     ; HALT THE SYSTEM  
148  
149 ;----- CHECK THE CONVERTING LOGIC  
150  
151 0056 B0 27          D7: MOV    AL,27H          ;  
152 0058 E6 80          OUT    MFG_PORT,AL    ; @@@@@@@@@@@@@@@@@@@@  
153 ; @@@ CHECKPOINT 27 @@@  
154 005A B8 AA55       MOV    AX,0AA55H      ; WRITE A WORD  
155 005D E7 82          OUT    MFG_PORT+2,AX  ;  
156 005F EA 82          IN     AL,MFG_PORT+2  ; GET THE FIRST BYTE  
157 0061 86 C4          XCHG  AL,AH           ; SAVE IT  
158 0063 E4 83          IN     AL,MFG_PORT+3  ; GET THE SECOND BYTE  
159 0065 3D 55AA       CMP    AX,55AAH       ; IS IT OK?  
160 0068 74 05         JZ     DT_A           ; GO IF YES  
161  
162 006A BE 0000 E     MOV    SI,OFFSET E106 ; DISPLAY 106 ERROR  
163 006D EB E2         JMP    D6A  
164  
165 ;----- CHECK FOR HOT NMI INTERRUPTS WITHOUT I/O-MEMORY PARITY ENABLED  
166  
167 006F B0 0F          DT_A: MOV    AL,CMOS_SHUT_DOWN ; TURN ON NMI  
168 0071 E6 70          OUT    CMOS_PORT,AL   ; ADDRESS DEFAULT READ ONLY REGISTER  
169 0073 B9 0007       MOV    CX,7            ; DELAY COUNT FOR 100 MICROSECONDS  
170 0076 E6 0000 E     CALL   WAITF           ; WAIT FOR HOT NMI TO PROCESS  
171 0079 B0 8F          MOV    AL,CMOS_SHUT_DOWN+NMI ; TURN NMI ENABLE BACK OFF  
172 007B E6 70          OUT    CMOS_PORT,AL   ;  
173 007D 80 3E 006B R 00  CMP    @INTR_FLAG,00H ; DID ANY INTERRUPTS OCCUR?  
174 0082 74 09         JZ     DT_C           ; CONTINUE IF NOT  
175  
176  
177 0084 B0 28          MOV    AL,28H          ;  
178 0086 E6 80          OUT    MFG_PORT,AL    ; @@@@@@@@@@@@@@@@@@@@  
179 ; @@@ CHECKPOINT 28 @@@  
180 0088 BE 0000 E     MOV    SI,OFFSET E107 ; DISPLAY 107 ERROR  
181 008B EB C4         JMP    D6A  
182  
183 ;----- TEST THE DATA BUS TO TIMER 2  
184  
185 008D B0 29          DT_C: MOV    AL,29H          ;  
186 008F E6 80          OUT    MFG_PORT,AL    ; @@@@@@@@@@@@@@@@@@@@  
187 0091 E4 61          IN     AL,PORT_B      ; GET CURRENT SETTING OF PORT  
188 0093 8A E0          MOV    AH,AL           ; SAVE THAT SETTING  
189 0095 24 FC          AND   AL,0FCH         ; INSURE SPEAKER OFF  
190 0097 E6 61          OUT    PORT_B,AL      ;  
191  
192 0099 B0 B0          MOV    AL,10110000B   ; SELECT TIM 2, LSB, MSB, BINARY, MODE 0  
193 009B E4 43          OUT    TIMER+3,AL     ; WRITE THE TIMER MODE REGISTER  
194 009D EB 00          JMP    $+2            ; I/O DELAY  
195 009F B8 AA55       MOV    AX,0AA55H      ; WRITE AN AA55  
196 00A2 E6 42          OUT    TIMER+2,AL     ; WRITE TIMER 2 COUNT - LSB  
197 00A4 EB 00          JMP    $+2            ; I/O DELAY  
198 00A6 8A C4          MOV    AL,AH           ; WRITE TIMER 2 COUNT - MSB  
199 00A8 E6 42          OUT    TIMER+2,AL     ;  
200 00AA EB 00          JMP    $+2            ; I/O DELAY  
201 00AC E4 42          IN     AL,TIMER+2     ; GET THE LSB  
202 00AE 86 60          XCHG  AH,AL           ; SAVE IT  
203 00B0 EB 00          JMP    $+2            ; I/O DELAY  
204 00B2 E4 42          IN     AL,TIMER+2     ; GET THE MSB  
205 00B4 3D 55AA       CMP    AX,55AAH       ; BUS OK?  
206 00B7 74 05         JZ     DT_D           ; GO IF OK  
207  
208 00B9 BE 0000 E     MOV    SI,OFFSET E108 ; DISPLAY 108 ERROR  
209 00BC EB 93         JMP    D6A  
210  
211 ;-----  
212 ; TEST_18  
213 ; 8254 TIMER CHECKOUT  
214 ; DESCRIPTION  
215 ; VERIFY THAT THE SYSTEM TIMER (0) DOESN'T COUNT  
216 ; TOO FAST OR TOO SLOW.  
217 ;-----  
218  
219 00BE B0 2A          DT_D: MOV    AL,2AH          ;  
220 00C0 E6 80          OUT    MFG_PORT,AL    ; @@@@@@@@@@@@@@@@@@@@  
221 00C2 FA            CL1  
222 00C3 B0 FE          MOV    AL,0FEH        ; MASK ALL INTERRUPTS EXCEPT LEVEL 0  
223 00C5 E6 21          OUT    INTA01,AL     ; WRITE THE 8259 IMR  
224 00C7 B0 10          MOV    AL,00010000B   ; SELECT TIM 0, LSB, MODE 0, BINARY  
225 00C9 E6 43          OUT    TIMER+3,AL     ; WRITE TIMER CONTROL, MODE REGISTER  
226 00CB B9 002C       MOV    CX,2CH         ; SET PROGRAM LOOP COUNT  
227  
228 00CE EB 00          JMP    $+2            ; I/O DELAY
```



```

343 0171 261 C7 06 006A 0000      MOV     ES,SS_TEMP_BASE_LO_WORD,0
344 0176 261 C6 06 006C 00      MOV     BYTE PTR ES:[SS_TEMP_BASE_HI_BYTE],0
345 017E BE 0058                  MOV     SI,SS_TEMP
346 0181 8E D6                    MOV     SS,SI
347 0183 BC FFFD                  MOV     SP,MAX_SEG_LEN-2
348
349
350
351 0184 6A 18                    PUSH   BYTE PTR RSDA_PTR      ; POINT TO DATA AREA
352 0188 1F                        POP
353
354 0189 B0 80                    MOV     AL,PARITY_CHECK      ; SET CHECK PARITY
355 018B E6 87                    OUT     DMA_PAGE+6,AL        ; SAVE WHICH CHECK TO USE
356
357
358
359 018D B8 0040                  MOV     AX,64                ; STARTING AMOUNT OF MEMORY OK
360 0190 EA 09A0 R                CALL   PRT_OK                ; POST 64K OK MESSAGE
361
362
363
364
365 0193 B8 80B1                  MOV     AX,(CMOS_U_M_S_LO+NM1)*H+CMOS_U_M_S_HI+NM1
366 0196 EA 0000 E                CALL   CMOS_READ            ; HIGH BYTE
367 0198 EA 0000 E                XCHG  AH,AL                 ; SAVE HIGH BYTE
368 019E BB 1E 0013 R            CALL   CMOS_READ            ; LOW BYTE
369 01A2 BB D3                    MOV     BX,MEMORY_SIZE      ; LOAD THE BASE MEMORY SIZE
370 01A4 03 D8                    MOV     DX,BX               ; SAVE BASE MEMORY SIZE
371                                ADD     BX,AX               ; SET TOTAL MEMORY SIZE
372
373
374 01A6 B0 8E                    MOV     AL,CMOS_DIAG+NM1    ; DETERMINE THE CONDITION OF CMOS
375 01A8 EA 0000 E                CALL   CMOS_READ            ; GET THE CMOS STATUS
376
377 01AB A8 C0                    TEST   AL,BAD_BAT+BAD_CKSUM ; CMOS OK?
378 01AD 74 02                    JZ     E20B0                ; GO IF YES
379 01AF EB 5B                    JMP    SHORT E20C           ; DEFAULT IF NOT
380
381
382
383
384 01B1
385 01B1 B8 9596                  MOV     AX,(CMOS_B_M_S_LO+NM1)*H+CMOS_B_M_S_HI+NM1
386 01B4 EA 0000 E                CALL   CMOS_READ            ; HIGH BYTE
387 01B7 24 3F                    AND    AL,03FH              ; MASK OFF THE MANUFACTURING TEST BITS
388 01B9 86 E0                    XCHG  AH,AL                 ; SAVE HIGH BYTE
389 01BB EA 0000 E                CALL   CMOS_READ            ; LOW BYTE
390 01C0 74 13                    JZ     E20B1                ; IS MEMORY SIZE GREATER THAN CONFIG?
391                                ; GO IF EQUAL
392
393
394
395 01C2 50                        PUSH   AX                   ; SAVE AX
396 01C3 B8 8E8E                  MOV     AX,X*(CMOS_DIAG+NM1) ; ADDRESS THE STATUS BYTE
397 01C6 EA 0000 E                CALL   CMOS_READ            ; GET THE STATUS
398 01C9 0C 10                    OR     AL,W_MEM_SIZE        ; SET CMOS FLAG
399 01CB 86 C4                    XCHG  AL,AH                 ; SAVE AL AND GET ADDRESS
400 01CD EA 0000 E                CALL   CMOS_WRITE           ; WRITE UPDATED STATUS
401 01D0 58                        POP    AX                   ; RESTORE AX
402 01D1 3B D0                    CMP    DX,AX                ; IS MEMORY SIZE GREATER THAN CONFIG?
403 01D3 77 37                    JA     E20B1                ; DEFAULT TO MEMORY SIZE DETERMINED?
404 01D5 8B D8                    MOV     BX,AX               ; SET BASE MEMORY SIZE IN TOTAL REGISTER
405 01D7 8B D0                    MOV     DX,AX               ; SAVE IN BASE SIZE REGISTER
406
407
408
409 01D9 B8 9798                  MOV     AX,(CMOS_E_M_S_LO+NM1)*H+(CMOS_E_M_S_HI+NM1)
410 01DC EA 0000 E                CALL   CMOS_READ            ; HIGH BYTE
411 01DF 86 E0                    XCHG  AH,AL                 ; SAVE HIGH BYTE
412 01E1 EA 0000 E                CALL   CMOS_READ            ; LOW BYTE
413 01E4 8B C8                    MOV     CX,AX               ; SAVE THE ABOVE 640K MEMORY SIZE
414
415
416
417 01E6 B8 80B1                  MOV     AX,(CMOS_U_M_S_LO+NM1)*H+(CMOS_U_M_S_HI+NM1)
418 01E9 EA 0000 E                CALL   CMOS_READ            ; HIGH BYTE
419 01EE EA 0000 E                XCHG  AH,AL                 ; SAVE HIGH BYTE
420 01F0 EA 0000 E                CALL   CMOS_READ            ; LOW BYTE
421
422
423 01F1 3B C8                    CMP    CX,AX                ; WHICH IS GREATER - AX = MEMORY SIZE DETERMINE
424 01F3 74 0F                    JZ     SET_MEM1             ; CX = CONFIGURATION (ABOVE 640) BX = SIZE (BELOW 640)
425
426
427
428
429 01F5 50                        PUSH   AX                   ; SAVE AX
430 01F6 B8 8E8E                  MOV     AX,X*(CMOS_DIAG+NM1) ; ADDRESS THE STATUS BYTE
431 01F9 EA 0000 E                CALL   CMOS_READ            ; GET THE STATUS
432 01FC 0C 10                    OR     AL,W_MEM_SIZE        ; SET CMOS FLAG
433 01FE 86 C4                    XCHG  AL,AH                 ; SAVE AL
434 0200 EA 0000 E                CALL   CMOS_WRITE           ; UPDATE STATUS BYTE
435 0203 58                        POP    AX                   ; RESTORE AX
436
437
438
439 0204 3B C8                    SET_MEM1: CMP    CX,AX                ; IS CONFIG GREATER THAN DETERMINED?
440 0206 77 02                    JA     SET_MEM              ; GO IF YES
441 0208 8B C8                    MOV     CX,AX               ; USE MEMORY SIZE DETERMINE IF NOT
442 020A
443 020A 03 D9                    ADD     BX,CX               ; SET TOTAL MEMORY SIZE
444 020C
445 020C 81 FA 0201                E20C: CMP    DX,513             ; CHECK IF BASE MEMORY LESS 512K
446 0210 72 0D                    JB     NO_640K              ; GO IF YES
447
448
449 0212 B8 83B3                  MOV     AX,X*(CMOS_INFO128+NM1) ; SET 640K BASE MEMORY BIT
450 0215 EA 0000 E                CALL   CMOS_READ            ; GET THE CURRENT STATUS
451 0218 0C 80                    CALL   AL,M640K             ; TURN ON 640K BIT IF NOT ALREADY ON
452 021A 86 C4                    XCHG  AL,AH                 ; SAVE THE CURRENT DIAGNOSTIC STATUS
453 021C EA 0000 E                CALL   CMOS_WRITE           ; RESTORE THE STATUS
454
455
456 021F 89 1E 0017 R            NO_640K: MOV     WORD PTR *KB_FLAG,BX ; SET TOTAL SIZE FOR LATER TESTING
457 0223 C1 EB 06                SHR    BX,6                 ; DIVIDE BY 64
458 0226 4B                        DEC    BX                   ; IS 64K ALREADY DONE
459 0227 C1 EA 06                SHR    DX,6                 ; DIVIDE BY 64 FOR BASE
460

```









```

753 PAGE
754 -----
755 MEMORY ERROR REPORTING (R/W/ MEMORY OR PARITY ERRORS)
756
757 DESCRIPTION FOR ERRORS 201 (CMP ERROR OR PARITY)
758 OR 202 (ADDRESS LINE 0-15 ERROR)
759
760 "AABBCC DDEE 201" (OR 202)
761 AA=HIGH BYTE OF 24 BIT ADDRESS
762 BB=MIDDLE BYTE OF 24 BIT ADDRESS
763 CC=LOW BYTE OF 24 BIT ADDRESS
764 DD=HIGH BYTE OF XOR FAILING BIT PATTERN
765 EE=LOW BYTE OF XOR FAILING BIT PATTERN
766
767 DESCRIPTION FOR ERROR 202 (ADDRESS LINE 00-15)
768 A WORD OF FFFF IS WRITTEN AT THE FIRST WORD AND LAST WORD
769 OF EACH 64K BLOCK WITH ZEROS AT ALL OTHER LOCATIONS OF THE
770 BLOCK. A SCAN OF THE BLOCK IS MADE TO INSURE ADDRESS LINE
771 0-15 ARE FUNCTIONING.
772
773 DESCRIPTION FOR ERROR 203 (ADDRESS LINE 16-23)
774 AT THE LAST PASS OF THE STORAGE TEST, FOR EACH BLOCK OF
775 64K, THE CURRENT STORAGE SIZE (ID) IS WRITTEN AT THE FIRST
776 WORD OF EACH BLOCK. IT IS USED TO FIND ADDRESSING FAILURES.
777
778 "AABBCC DDEE 203"
779 SAME AS ABOVE EXCEPT FOR DDEE
780
781 GENERAL DESCRIPTION FOR BLOCK ID (DDEE WILL NOW CONTAINED THE ID)
782 DD=HIGH BYTE OF BLOCK ID
783 EE=LOW BYTE OF BLOCK ID
784
785 BLOCK ID ADDRESS RANGE
786 0000 000000 --> 00FFFF
787 0040 010000 --> 01FFFF
788 //
789 0200 090000 --> 09FFFF (512->576K) IF 640K BASE
790 100000 --> 10FFFF (1024->1088K) IF 512K BASE
791
792 EXAMPLE (640K BASE MEMORY + 512K I/O MEMORY = 1152K TOTAL)
793 NOTE: THE CORRECT BLOCK ID FOR THIS FAILURE IS 0280 HEX.
794 DUE TO AN ADDRESS FAILURE THE BLOCK ID+128K OVERLAYED
795 THE CORRECT BLOCK ID.
796
797 00640K OK <-- LAST OK MEMORY
798 10000 0300 202 <-- ERROR DUE TO ADDRESS FAILURE
799
800 IF A PARITY LATCH WAS SET THE CORRESPONDING MESSAGE WILL DISPLAY.
801
802 *PARITY CHECK 1* (OR 2)
803
804 DMA PAGE REGISTERS ARE USED AS TEMPORARY SAVE AREAS FOR SEGMENT
805 DESCRIPTOR VALUES.
806 -----
807
808 03BC 03BC E8 0000 E SHUT3: CALL DOS ; ENTRY FROM PROCESSOR SHUTDOWN 3
809 03BC E8 0000 E ; SET REAL MODE DATA SEGMENT
810
811
812 03BF C6 06 0016 R 01 MOV MFG_ERR_FLAG+1, MEM_FAIL ; MEMORY FAILED <<>
813 03C4 B0 0D MOV AL, CR ; CLEAR AND SET MANUFACTURING ERROR FLAG
814 03C6 E8 0000 E CALL PRT_HEX ; CARRIAGE RETURN
815 03C9 B0 0A MOV AL, LF ; LINE FEED
816 03CB E8 0000 E CALL PRT_HEX
817 03CE E4 84 IN AL, DMA_PAGE+3 ; GET THE HIGH BYTE OF 24 BIT ADDRESS
818 03D0 E8 0000 E CALL XPC_BYTE ; CONVERT AND PRINT CODE
819 03D3 E4 86 IN AL, DMA_PAGE+4 ; GET THE MIDDLE BYTE OF 24 BIT ADDRESS
820 03D6 E8 0000 E CALL XPC_BYTE
821 03D8 E4 88 IN AL, DMA_PAGE+5 ; GET THE LOW BYTE OF 24 BIT ADDRESS
822 03DA E8 0000 E CALL XPC_BYTE
823 03DD B0 20 MOV AL, ' ' ; SPACE TO MESSAGE
824 03DF E8 0000 E CALL PRT_HEX
825 03E2 E4 83 IN AL, DMA_PAGE+2 ; GET HIGH BYTE FAILING BIT PATTERN
826 03E4 E8 0000 E CALL XPC_BYTE ; CONVERT AND PRINT CODE
827 03E7 E4 82 IN AL, DMA_PAGE+1 ; GET LOW BYTE FAILING BIT PATTERN
828 03E9 E8 0000 E CALL XPC_BYTE ; CONVERT AND PRINT CODE
829
830
831
832 03EC E4 80 IN AL, MFG_PORT ; GET THE CHECKPOINT
833 03EE 3C 33 CMP AL, 33H ; IS IT AN ADDRESS FAILURE?
834 03F0 0E 0000 E MOV SI, OFFSET E203 ; LOAD ADDRESS ERROR 16->23
835 03F3 74 0A JZ ERR2 ; GO IF YES
836
837 03F5 BE 0000 E MOV SI, OFFSET E202 ; LOAD ADDRESS ERROR 00->15
838 03F8 3C 32 CMP AL, 32H ; GO IF YES
839 03FA 74 03 JZ ERR2
840
841 03FC BE 0000 E MOV SI, OFFSET E201 ; SETUP ADDRESS OF ERROR MESSAGE
842 03FF E8 0000 E CALL E_MSG ; PRINT ERROR MESSAGE
843 0402 E4 88 IN AL, DMA_PAGE+7 ; GET THE PORT_B VALUE
844
845
846
847
848 0404 A8 80 TEST AL, PARITY_CHECK ; CHECK FOR PLANAR ERROR
849 0406 74 0B JZ NMI_M1 ; SKIP IF NOT
850
851 0408 80 PUSH AX ; SAVE STATUS
852 0409 EB 0990 R CALL PADING ; INSERT BLANKS
853 040C BE 0000 E MOV SI, OFFSET D1 ; PLANAR ERROR ADDRESS *PARITY CHECK 1*
854 040F E8 0000 E CALL P_MSG ; DISPLAY *PARITY CHECK 1* MESSAGE
855 0412 58 POP AX ; AND RECOVER STATUS
856 0413
857 0413 A8 40 TEST AL, I/O_CHECK ; I/O PARITY CHECK ?
858 0415 74 09 JZ NMI_M2 ; SKIP IF CORRECT ERROR DISPLAYED
859
860 0417 EB 0990 R CALL PADING ; INSERT BLANKS
861 041A BE 0000 E MOV SI, OFFSET D2 ; ADDRESS OF *PARITY CHECK 2* MESSAGE
862 041D E8 0000 E CALL P_MSG ; DISPLAY *PARITY CHECK 2* ERROR
863 0420
864 ; CONTINUE TESTING SYSTEM ....

```







```

1203          PAGE
1204          I----- SETUP KEYBOARD PARAMETERS
1205
1206          MOV     @INTR_FLAG,00H          ; SET STRAY INTERRUPT FLAG = 00
1207          MOV     SI,OFFSET @KB_BUFFER    ; SETUP KEYBOARD PARAMETERS
1208          MOV     @BUFFER_HEAD,SI
1209          MOV     @BUFFER_TAIL,SI
1210          MOV     @BUFFER_START,SI
1211          ADD     SI,32                    ; DEFAULT BUFFER OF 32 BYTES
1212          MOV     @BUFFER_END,SI
1213
1214          I----- SET PRINTER TIMEOUT DEFAULT
1215
1216          MOV     DI,OFFSET @PRINT_TIM_OUT; SET DEFAULT PRINTER TIMEOUT
1217          PUSH   DS
1218          POP     ES
1219          MOV     AX,1414H                 ; DEFAULT=20
1220          STOSW
1221          STOSW
1222
1223          I----- SET RS232 DEFAULT
1224
1225          MOV     AX,0101H                 ; RS232 DEFAULT=01
1226          STOSW
1227          STOSW
1228
1229          I----- ENABLE TIMER INTERRUPTS
1230
1231          IN     AL,INTA01
1232          AND   AL,0FEH                    ; ENABLE TIMER INTERRUPTS
1233          JMP   $+2                         ; I/O DELAY
1234          OUT   INTA01,AL
1235
1236          I----- CHECK CMOS BATTERY AND CHECKSUM
1237
1238          TEST   @MFG_TST,MFG_LOOP        ; MFG JUMPER?
1239          JNZ   B1_OK                      ; GO IF NOT
1240          JMP   F15C                       ; BYPASS IF YES
1241          B1_OK:
1242          MOV   AL,CMOS_DIAG+NMI          ; ADDRESS DIAGNOSTIC STATUS BYTE
1243          CALL  CMOS_READ                  ; READ IT FROM CMOS
1244
1245          MOV   SI,OFFSET E161             ; LOAD BAD BATTERY MESSAGE 161
1246          TEST  AL,BAD_BAT                ; BATTERY BAD?
1247          JNZ   B1_ER                      ; DISPLAY ERROR IF BAD
1248
1249          MOV   SI,OFFSET E162             ; LOAD CHECKSUM BAD MESSAGE 162
1250          TEST  AL,BAD_CHKSUM+BAD_CONFIG  ; CHECK FOR CHECKSUM OR NO DISKETTE
1251          JZ    B1_ER                      ; SKIP AND CONTINUE TESTING CMOS CLOCK
1252          B1_ER:
1253          CALL  E_MSG                       ; ELSE DISPLAY ERROR MESSAGE
1254          OR    BP,@8000H                  ; FLAG "SET SYSTEM OPTIONS" DISPLAYED
1255          JMP   SHORT H_OK1A               ; SKIP CLOCK TESTING IF ERROR
1256
1257          I----- TEST CLOCK UPDATING
1258
1259          C_OK: MOV   BL,04H                 ; OUTER LOOP COUNT
1260          D_OK: SUB   CX,CX                 ; INNER LOOP COUNT
1261          E_OK: MOV   AL,CMOS_REG_A+NMI     ; GET THE CLOCK UPDATE BYTE
1262          CALL  CMOS_READ
1263          TEST  AL,80H                     ; CHECK FOR UPDATE IN PROGRESS
1264          JNZ   G_OK                       ; GO IF YES
1265          LOOP  E_OK                       ; TRY AGAIN
1266          DEC   BL                         ; DEC OUTER LOOP
1267          JNZ   D_OK                       ; TRY AGAIN
1268          MOV   SI,OFFSET E163             ; PRINT MESSAGE
1269          CALL  E_MSG
1270
1271          I----- SET CMOS DIAGNOSTIC STATUS TO 04 (CLOCK ERROR)
1272
1273          MOV   AX,X*CMOS_DIAG+NMI        ; SET CLOCK ERROR
1274          CALL  CMOS_READ                  ; GET THE CURRENT STATUS
1275          OR    AL,CMOS_CLK_FAIL          ; SET NEW STATUS
1276          XCHG  AL,AH                      ; GET STATUS ADDRESS AND SAVE NEW STATUS
1277          CALL  CMOS_WRITE                 ; MOVE NEW DIAGNOSTIC STATUS TO CMOS
1278          JMP   SHORT H_OK
1279
1280          I----- CHECK CLOCK UPDATE
1281
1282          G_OK: MOV   CX,800                ; LOOP COUNT
1283          I_OK: MOV   AL,CMOS_REG_A+NMI     ; CHECK FOR OPPOSITE STATE
1284          CALL  CMOS_READ
1285          TEST  AL,80H
1286          LOOPNZ I_OK                      ; TRY AGAIN
1287          JCXZ  F_OK                       ; PRINT ERROR IF TIMEOUT
1288
1289          I----- CHECK MEMORY SIZE DETERMINED = CONFIGURATION
1290
1291          H_OK:
1292          MOV   AL,CMOS_DIAG+NMI          ; GET THE STATUS BYTE
1293          CALL  CMOS_READ
1294          TEST  AL,W_MEM_SIZE
1295          JZ    H_OK1A                     ; WAS THE CONFIG= MEM_SIZE_DETERMINED?
1296
1297          I----- MEMORY SIZE ERROR
1298
1299          MOV   SI,OFFSET E164             ; PRINT SIZE ERROR
1300          CALL  E_MSG                       ; DISPLAY ERROR
1301
1302          I----- CHECK FOR CRT ADAPTER ERROR
1303
1304          H_OK1A: CMP   @MFG_ERR_FLAG,0CH  ; CHECK FOR MONOCHROME CRT ERROR
1305          MOV   SI,OFFSET E401             ; LOAD MONOCHROME CRT ERROR
1306          JZ    H_OK1B                     ; GO IF YES
1307
1308          H_OK1B: CMP   @MFG_ERR_FLAG,0DH  ; CHECK FOR COLOR CRT ADAPTER ERROR
1309          JNZ   J_OK                       ; CONTINUE IF NOT
1310          MOV   SI,OFFSET E501             ; CRT ADAPTER ERROR MESSAGE
1311          H_OK1B: CALL  E_MSG
1312
1313          MOV   AL,CMOS_DIAG+NMI          ; GET THE STATUS BYTE
1314          CALL  CMOS_READ
1315          TEST  AL,W_MEM_SIZE
1316          JZ    H_OK1A
1317
1318          MOV   SI,OFFSET E164
1319          CALL  E_MSG
1320
1321          H_OK1A: CMP   @MFG_ERR_FLAG,0CH
1322          MOV   SI,OFFSET E401
1323          JZ    H_OK1B
1324
1325          H_OK1B: CMP   @MFG_ERR_FLAG,0DH
1326          JNZ   J_OK
1327          MOV   SI,OFFSET E501
1328          H_OK1B: CALL  E_MSG
    
```







```

1541 0816 9B DD 3D          FSTSW WORD PTR [DI]          ; STORE THE STATUS WORD (WITH WAIT)
1542 0819 60              PUSHA                          ; TIME FOR 80287 TO RESPOND
1543 081A 61              POPA                           ;
1544 081B F7 05 88BF      TEST WORD PTR [DI],0B8BFH     ; ALL BITS SHOULD BE OFF (OR ERROR)
1545 081F 75 08              JNZ NO_287                    ; GO IF NOT INSTALLED
1546
1547 0821 E4 A1            IN AL,INTB01                  ; GET THE SLAVE INTERRUPT MASK
1548 0823 24 DF          AND AL,0DFH                   ; ENABLE 80287 INTERRUPTS
1549 0825 B4 02          MOV AH,02H                    ; SET WORK REGISTER FOR 80287 FOUND
1550 0827 E6 A1            OUT INTB01,AL                  ;
1551 0829
NO_287:
1552 0829 A0 0010 R      MOV AL,BYTE PTR @EQUIP_FLAG   ; GET LOW EQUIPMENT FLAG
1553 082C 24 02          AND AL,02H                    ; STRIP OFF OTHER BITS
1554 082E 3A C4          CMP AL,AH                      ; DOES CMOS MATCH HARDWARE ?
1555 0830 74 08          JE OK_287                      ; SKIP IF EQUIPMENT FLAG CORRECT
1556
1557 0832 80 36 0010 R 02   XOR BYTE PTR @EQUIP_FLAG,2H   ; ELSE SET 80287 BIT TO CORRECT VALUE
1558 0837 E8 0000 E      CALL CONFIG_BAD                ; AND SET THE CONFIGURATION ERROR FLAG
1559 083A
OK_287:
1560
1561
1562 083A C7 06 0017 R 0000 MOV WORD PTR @KB_FLAG,0       ; RESET ALL KEYBOARD STATUS FLAGS
1563
1564
1565
1566
1567
1568 0840 E4 21            IN AL,INTA01                  ; ENABLE TIMER AND KEYBOARD INTERRUPTS
1569 0842 24 FC          AND AL,0FCH                    ; I/O DELAY
1570 0844 E8 00          JMP $+2                         ;
1571 0846 E6 21            OUT INTA01,AL                  ;
1572 0848 C6 06 0015 R 00 MOV @MFG_ERR_FLAG,0          ; CLEAR MFG ERROR FLAG
1573
1574
1575
1576 084D C6 06 0096 R A0 MOV @KB_FLAG_3,RD_ID+SET_NUM_LK ; SET READ ID COMMAND FOR KBX
1577 0852 B0 F2          MOV AL,KB_READ_ID             ; GET THIS SYSTEMS KEYBOARD ID REQUEST
1578 0854 E8 0000 E      CALL SND_DXA                   ; USE KEYBOARD TRANSMISSION ROUTINE
1579 0857 B9 067A        MOV CX,1658                    ; SET DELAY COUNT TO 26 MILLI SECONDS
1580 0859 C7 13          CALL WAITF                       ; WAIT FOR READ ID RESPONSE (20 MS)
1581 085D 80 26 0096 R IF AND @KB_FLAG_3,NOT RD_ID+LC_AB+SET_NUM_LK ; RESET READ ID COMMAND
1582
1583
1584
1585
1586 0862 80 3E 0075 R 02   CMP @MFG_NUM,2                ; CHECK FOR TWO DRIVES DEFINED BY CMOS
1587 0867 74 13          JE F15G                        ; SKIP TEST IF TWO DRIVES DEFINED
1588
1589 0869 B4 10            MOV AH,010H                    ; GET TEST DRIVE READY COMMAND
1590 086B B2 81          MOV DL,081H                    ; POINT TO SECOND FIXED DISK
1591 086D FE 06 0075 R R INC @MFG_NUM                    ; TELL BIOS IT HAS TWO DRIVES
1592 0871 CD 13          INT 13H                         ; CHECK READY THROUGH BIOS
1593 0873 FE 0E 0075 R R DEC @MFG_NUM                    ; RESTORE CORRECT COUNT (RETAIN CY)
1594 0877 72 03          JC F15G                        ; SKIP IF SECOND DRIVE NOT READY
1595
1596 0879 E8 0000 E      CALL CONFIG_BAD                ; SET CONFIGURATION BAD
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608 087C 0B ED          OR BP,BP                        ; CHECK (BP)=NON-ZERO (ERROR HAPPENED)
1609 087E 74 55          JE F15A_0                       ; SKIP PAUSE IF NO ERROR
1610
1611
1612 0880 80 3E 0072 R 64   CMP BYTE PTR @RESET_FLAG,64H ; MFG RUN IN MODE?
1613 0885 BA 0002        MOV DX,2                        ; 2 SHORT BEEP COUNT FOR ERROR(S)
1614 0888 75 0E          JNZ ERR_WAIT                    ; GO IF NOT
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629 088A C6 06 0015 R AA MOV @MFG_ERR_FLAG,0AAH        ; INDICATE ERROR
1630 088F E4 64          IN AL,STATUS_PORT              ; CHECK KEY LOCK STATUS
1631 0891 24 10          AND AL,KYBD_INH                ; IS THE KEYBOARD LOCKED
1632 0893 75 40          JNZ F15A_0                       ; CONTINUE MFG MODE IF NOT LOCKED
1633
1634
1635
1636
1637
1638
1639 0895 B8 0005        MOV DX,5                        ; 5 SHORT BEEPS FOR MFG SETUP ERROR
ERR_WAIT:
1640 0898          CALL ERR_BEEP                   ; BEEPS FOR ERROR(S)
1641 089B          MOV AL,CMOS_DIAG                ; ADDRESS CMOS
1642 089D          CALL CMOS_READ                  ; GET THE DIAGNOSTIC STATUS BYTE
1643 08A0          TEST AL,BAD_CONFIG              ; CHECK FOR BAD HARDWARE CONFIGURATION
1644 08A2          JZ ERR_WKEY                     ; SKIP IF NOT SET
1645
1646
1647 08A4 F7 C5 8000      TEST BP,08000H                 ; ELSE CHECK FOR E161/E162 POSTED
1648 08A8 75 06          JNZ ERR_WKEY                     ; SKIP IF DISPLAYED BEFORE NOW
1649
1650
1651 08AA BE 0000 E      MOV SI,OFFSET E162             ; ELSE DISPLAY "OPTIONS NOT SET"
1652 08AD E8 0000 E      CALL P_MSG                      ; WITH NON HALTING ROUTINE
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000

```

SECTION 5







```

115 0056 B0 F2      TT_2:  MOV  AL,0F2H          |
116 0086 E6 80      OUT  MFG_PORT,AL      |
117 005A B0 9D      MOV  AL,90H           |
118 005C E6 8B      OUT  DMA_PAGE+0AH,AL  |
119                                     |
120                                     |
121                                     |
122                                     |
123 005E C7 06 0048 0000  |
124                                     |
125                                     |
126                                     |
127 0044 C6 06 004D 93  |
128 0069 C6 06 004C 01  |
129 006E C7 06 004A 0000  |
130                                     |
131 0074 6A 48      PUSH BYTE PTR ES_TEMP  |
132 0076 07        POP  ES               |
133                                     |
134                                     |
135                                     |
136 0077 2B FF      SUB  DI,DI            |
137 0079 26 8B 05  |
138 007C 2B C9      SUB  CX,CX            |
139 007E E4 8B      IN  AL,DMA_PAGE+0AH  |
140 0080 22 C0      AND  AL,AL            |
141 0082 E0 FA      LOOPNZ LOOP2          |
142 0084 74 03      JZ   TT_3              |
143 0086 E9 02CD R  JMP  ERROR_EXIT       |
144                                     |
145                                     |
146                                     |
147                                     |
148                                     |
149                                     |
150                                     |
151                                     |
152                                     |
153                                     |
154                                     |
155 0089          |
156 0089 B0 F3      TT_3:  MOV  AL,0F3H          |
157 008B E6 80      OUT  MFG_PORT,AL      |
158 008D BF 0078  |
159                                     |
160 0090 0F        +  DB  00FH             |
161 0091          +  LABEL BYTE           |
162 0091 8B D7      +  MOV  DX,DI           |
163 0093          +  LABEL BYTE           |
164 0091          +  ORG  OFFSET CS:??0000 |
165 0091 00        +  DB  000H             |
166 0093          +  ORG  OFFSET CS:??0001 |
167                                     |
168                                     |
169                                     |
170 0093 2B C0      SUB  AX,AX            |
171                                     |
172 0095 0F        +  SLDT AX              |
173 0096          +  DB  00FH             |
174 0096 03 C0      +  ??0002 LABEL BYTE   |
175 0098          +  ADD  AX,AX           |
176 0096          +  ??0003 LABEL BYTE   |
177 0096 00        +  ORG  OFFSET CS:??0002 |
178 0098          +  ORG  OFFSET CS:??0003 |
179 0098 25 00F8  |
180 009B 3D 0078  |
181 009E 75 1B      AND  AX,0F8H         |
182                                     |
183                                     |
184                                     |
185 00A0 BF 0068  |
186                                     |
187 00A3 0F        +  MOV  DI,POST_TR     |
188 00A4          +  LTR  DI              |
189 00A4 8B DF      +  DB  00FH             |
190 00A6          +  LABEL BYTE           |
191 00A4          +  MOV  BX,DI           |
192 00A4 00        +  ??0005 LABEL BYTE   |
193 00A6          +  ORG  OFFSET CS:??0004 |
194                                     |
195                                     |
196                                     |
197 00A6 2B C0      SUB  AX,AX            |
198                                     |
199 00A8 0F        +  STR  AX              |
200 00A9          +  DB  00FH             |
201 00A9 8B C8      +  ??0006 LABEL BYTE   |
202 00AB          +  MOV  CX,AX           |
203 00A9          +  LABEL BYTE           |
204 00A9 00        +  ORG  OFFSET CS:??0006 |
205 00AB          +  ORG  OFFSET CS:??0007 |
206 00AB 25 00F8  |
207 00AE 3D 0068  |
208 00B1 75 08      AND  AX,0F8H         |
209                                     |
210                                     |
211                                     |
212 00B3 FD      STD  |
213 00B4 9C      PUSHF |
214 00B5 58      POP  AX              |
215 00B6 A9 0200  |
216 00B9 74 03      TEST AX,0200H        |
217 00BB E9 02CD R  JZ   TT_4              |
218 00BE          |
219 00BE A9 0400  |
220 00C1 75 03      JMP  ERROR_EXIT       |
221 00C3 E9 02CD R  |
222 00C6          |
223 00C6 FC      CLD  |
224 00C7 9C      PUSHF |
225 00C8 58      POP  AX              |
226 00C9 A9 0400  |
227 00CC 74 03      TEST AX,0400H        |
228 00CE E9 02CD R  JZ   TT_5              |
229                                     |
230                                     |
231                                     |
232                                     |
233                                     |
234                                     |
235                                     |
236                                     |
237                                     |
238                                     |
239                                     |
240                                     |
241                                     |
242                                     |
243                                     |
244                                     |
245                                     |
246                                     |
247                                     |
248                                     |
249                                     |
250                                     |
251                                     |
252                                     |
253                                     |
254                                     |
255                                     |
256                                     |
257                                     |
258                                     |
259                                     |
260                                     |
261                                     |
262                                     |
263                                     |
264                                     |
265                                     |
266                                     |
267                                     |
268                                     |
269                                     |
270                                     |
271                                     |
272                                     |
273                                     |
274                                     |
275                                     |
276                                     |
277                                     |
278                                     |
279                                     |
280                                     |
281                                     |
282                                     |
283                                     |
284                                     |
285                                     |
286                                     |
287                                     |
288                                     |
289                                     |
290                                     |
291                                     |
292                                     |
293                                     |
294                                     |
295                                     |
296                                     |
297                                     |
298                                     |
299                                     |
300                                     |
301                                     |
302                                     |
303                                     |
304                                     |
305                                     |
306                                     |
307                                     |
308                                     |
309                                     |
310                                     |
311                                     |
312                                     |
313                                     |
314                                     |
315                                     |
316                                     |
317                                     |
318                                     |
319                                     |
320                                     |
321                                     |
322                                     |
323                                     |
324                                     |
325                                     |
326                                     |
327                                     |
328                                     |
329                                     |
330                                     |
331                                     |
332                                     |
333                                     |
334                                     |
335                                     |
336                                     |
337                                     |
338                                     |
339                                     |
340                                     |
341                                     |
342                                     |
343                                     |
344                                     |
345                                     |
346                                     |
347                                     |
348                                     |
349                                     |
350                                     |
351                                     |
352                                     |
353                                     |
354                                     |
355                                     |
356                                     |
357                                     |
358                                     |
359                                     |
360                                     |
361                                     |
362                                     |
363                                     |
364                                     |
365                                     |
366                                     |
367                                     |
368                                     |
369                                     |
370                                     |
371                                     |
372                                     |
373                                     |
374                                     |
375                                     |
376                                     |
377                                     |
378                                     |
379                                     |
380                                     |
381                                     |
382                                     |
383                                     |
384                                     |
385                                     |
386                                     |
387                                     |
388                                     |
389                                     |
390                                     |
391                                     |
392                                     |
393                                     |
394                                     |
395                                     |
396                                     |
397                                     |
398                                     |
399                                     |
400                                     |
401                                     |
402                                     |
403                                     |
404                                     |
405                                     |
406                                     |
407                                     |
408                                     |
409                                     |
410                                     |
411                                     |
412                                     |
413                                     |
414                                     |
415                                     |
416                                     |
417                                     |
418                                     |
419                                     |
420                                     |
421                                     |
422                                     |
423                                     |
424                                     |
425                                     |
426                                     |
427                                     |
428                                     |
429                                     |
430                                     |
431                                     |
432                                     |
433                                     |
434                                     |
435                                     |
436                                     |
437                                     |
438                                     |
439                                     |
440                                     |
441                                     |
442                                     |
443                                     |
444                                     |
445                                     |
446                                     |
447                                     |
448                                     |
449                                     |
450                                     |
451                                     |
452                                     |
453                                     |
454                                     |
455                                     |
456                                     |
457                                     |
458                                     |
459                                     |
460                                     |
461                                     |
462                                     |
463                                     |
464                                     |
465                                     |
466                                     |
467                                     |
468                                     |
469                                     |
470                                     |
471                                     |
472                                     |
473                                     |
474                                     |
475                                     |
476                                     |
477                                     |
478                                     |
479                                     |
480                                     |
481                                     |
482                                     |
483                                     |
484                                     |
485                                     |
486                                     |
487                                     |
488                                     |
489                                     |
490                                     |
491                                     |
492                                     |
493                                     |
494                                     |
495                                     |
496                                     |
497                                     |
498                                     |
499                                     |
500                                     |
501                                     |
502                                     |
503                                     |
504                                     |
505                                     |
506                                     |
507                                     |
508                                     |
509                                     |
510                                     |
511                                     |
512                                     |
513                                     |
514                                     |
515                                     |
516                                     |
517                                     |
518                                     |
519                                     |
520                                     |
521                                     |
522                                     |
523                                     |
524                                     |
525                                     |
526                                     |
527                                     |
528                                     |
529                                     |
530                                     |
531                                     |
532                                     |
533                                     |
534                                     |
535                                     |
536                                     |
537                                     |
538                                     |
539                                     |
540                                     |
541                                     |
542                                     |
543                                     |
544                                     |
545                                     |
546                                     |
547                                     |
548                                     |
549                                     |
550                                     |
551                                     |
552                                     |
553                                     |
554                                     |
555                                     |
556                                     |
557                                     |
558                                     |
559                                     |
560                                     |
561                                     |
562                                     |
563                                     |
564                                     |
565                                     |
566                                     |
567                                     |
568                                     |
569                                     |
570                                     |
571                                     |
572                                     |
573                                     |
574                                     |
575                                     |
576                                     |
577                                     |
578                                     |
579                                     |
580                                     |
581                                     |
582                                     |
583                                     |
584                                     |
585                                     |
586                                     |
587                                     |
588                                     |
589                                     |
590                                     |
591                                     |
592                                     |
593                                     |
594                                     |
595                                     |
596                                     |
597                                     |
598                                     |
599                                     |
600                                     |
601                                     |
602                                     |
603                                     |
604                                     |
605                                     |
606                                     |
607                                     |
608                                     |
609                                     |
610                                     |
611                                     |
612                                     |
613                                     |
614                                     |
615                                     |
616                                     |
617                                     |
618                                     |
619                                     |
620                                     |
621                                     |
622                                     |
623                                     |
624                                     |
625                                     |
626                                     |
627                                     |
628                                     |
629                                     |
630                                     |
631                                     |
632                                     |
633                                     |
634                                     |
635                                     |
636                                     |
637                                     |
638                                     |
639                                     |
640                                     |
641                                     |
642                                     |
643                                     |
644                                     |
645                                     |
646                                     |
647                                     |
648                                     |
649                                     |
650                                     |
651                                     |
652                                     |
653                                     |
654                                     |
655                                     |
656                                     |
657                                     |
658                                     |
659                                     |
660                                     |
661                                     |
662                                     |
663                                     |
664                                     |
665                                     |
666                                     |
667                                     |
668                                     |
669                                     |
670                                     |
671                                     |
672                                     |
673                                     |
674                                     |
675                                     |
676                                     |
677                                     |
678                                     |
679                                     |
680                                     |
681                                     |
682                                     |
683                                     |
684                                     |
685                                     |
686                                     |
687                                     |
688                                     |
689                                     |
690                                     |
691                                     |
692                                     |
693                                     |
694                                     |
695                                     |
696                                     |
697                                     |
698                                     |
699                                     |
700                                     |
701                                     |
702                                     |
703                                     |
704                                     |
705                                     |
706                                     |
707                                     |
708                                     |
709                                     |
710                                     |
711                                     |
712                                     |
713                                     |
714                                     |
715                                     |
716                                     |
717                                     |
718                                     |
719                                     |
720                                     |
721                                     |
722                                     |
723                                     |
724                                     |
725                                     |
726                                     |
727                                     |
728                                     |
729                                     |
730                                     |
731                                     |
732                                     |
733                                     |
734                                     |
735                                     |
736                                     |
737                                     |
738                                     |
739                                     |
740                                     |
741                                     |
742                                     |
743                                     |
744                                     |
745                                     |
746                                     |
747                                     |
748                                     |
749                                     |
750                                     |
751                                     |
752                                     |
753                                     |
754                                     |
755                                     |
756                                     |
757                                     |
758                                     |
759                                     |
760                                     |
761                                     |
762                                     |
763                                     |
764                                     |
765                                     |
766                                     |
767                                     |
768                                     |
769                                     |
770                                     |
771                                     |
772                                     |
773                                     |
774                                     |
775                                     |
776                                     |
777                                     |
778                                     |
779                                     |
780                                     |
781                                     |
782                                     |
783                                     |
784                                     |
785                                     |
786                                     |
787                                     |
788                                     |
789                                     |
790                                     |
791                                     |
792                                     |
793                                     |
794                                     |
795                                     |
796                                     |
797                                     |
798                                     |
799                                     |
800                                     |
801                                     |
802                                     |
803                                     |
804                                     |
805                                     |
806                                     |
807                                     |
808                                     |
809                                     |
810                                     |
811                                     |
812                                     |
813                                     |
814                                     |
815                                     |
816                                     |
817                                     |
818                                     |
819                                     |
820                                     |
821                                     |
822                                     |
823                                     |
824                                     |
825                                     |
826                                     |
827                                     |
828                                     |
829                                     |
830                                     |
831                                     |
832                                     |
833                                     |
834                                     |
835                                     |
836                                     |
837                                     |
838                                     |
839                                     |
840                                     |
841                                     |
842                                     |
843                                     |
844                                     |
845                                     |
846                                     |
847                                     |
848                                     |
849                                     |
850                                     |
851                                     |
852                                     |
853                                     |
854                                     |
855                                     |
856                                     |
857                                     |
858                                     |
859                                     |
860                                     |
861                                     |
862                                     |
863                                     |
864                                     |
865                                     |
866                                     |
867                                     |
868                                     |
869                                     |
870                                     |
871                                     |
872                                     |
873                                     |
874                                     |
875                                     |
876                                     |
877                                     |
878                                     |
879                                     |
880                                     |
881                                     |
882                                     |
883                                     |
884                                     |
885                                     |
886                                     |
887                                     |
888                                     |
889                                     |
890                                     |
891                                     |
892                                     |
893                                     |
894                                     |
895                                     |
896                                     |
897                                     |
898                                     |
899                                     |
900                                     |
901                                     |
902                                     |
903                                     |
904                                     |
905                                     |
906                                     |
907                                     |
908                                     |
909                                     |
910                                     |
911                                     |
912                                     |
913                                     |
914                                     |
915                                     |
916                                     |
917                                     |
918                                     |
919                                     |
920                                     |
921                                     |
922                                     |
923                                     |
924                                     |
925                                     |
926                                     |
927                                     |
928                                     |
929                                     |
930                                     |
931                                     |
932                                     |
933                                     |
934                                     |
935                                     |
936                                     |
937                                     |
938                                     |
939                                     |
940                                     |
941                                     |
942                                     |
943                                     |
944                                     |
945                                     |
946                                     |
947                                     |
948                                     |
949                                     |
950                                     |
951                                     |
952                                     |
953                                     |
954                                     |
955                                     |
956                                     |
957                                     |
958                                     |
959                                     |
960                                     |
961                                     |
962                                     |
963                                     |
964                                     |
965                                     |
966                                     |
967                                     |
968                                     |
969                                     |
970                                     |
971                                     |
972                                     |
973                                     |
974                                     |
975                                     |
976                                     |
977                                     |
978                                     |
979                                     |
980                                     |
981                                     |
982                                     |
983                                     |
984                                     |
985                                     |
986                                     |
987                                     |
988                                     |
989                                     |
990                                     |
991                                     |
992                                     |
993                                     |
994                                     |
995                                     |
996                                     |
997                                     |
998                                     |
999                                     |
1000                                    |
    
```

SECTION 5









```

571 ;----- DO FOR SEGMENT IA0000
572
573 028A C6 06 004C 1A      MOV    BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE),1AH
574 028F C7 06 004A 0000    MOV    DS:ES_TEMP.BASE_LO_WORD,0
575 0295 6A 48              PUSH   BYTE PTR ES_TEMP      ; LOAD ES REGISTER
576 0297 07                POP    ES
577 0298 26: C7 05 AA55      MOV    WORD PTR ES:[DI],0AA55H ; WRITE A TEST PATTERN
578
579 ;----- B/W VIDEO CARD
580
581 029D 6A 20              PUSH   BYTE PTR C_BWCRT_PTR
582 029F 1F                POP    DS                    ; SET DS TO B/W DISPLAY REGEN BUFFER
583 02A0 8B 05              MOV    AX,DS:[DI]           ; GET THE WORD FROM B/W VIDEO
584
585 ;----- COMPATIBLE COLOR
586
587 02A2 6A 28              PUSH   BYTE PTR C_CCRT_PTR  ; SET DS TO COMPATIBLE COLOR MEMORY
588 02A4 1F                POP    DS                    ; GET THE WORD FROM COLOR MEMORY
589 02A5 8B 1D              MOV    BX,DS:[DI]
590
591 ;----- EGA COLOR
592
593 02A7 6A 30              PUSH   BYTE PTR E_CCRT_PTR  ; EGA COLOR CRT POINTER LOW 64K
594 02A9 1F                POP    DS
595 02AA 8B 0D              MOV    CX,DS:[DI]
596
597 ;----- TEST FOR ERROR
598
599 02AC 50                PUSH   AX                    ; SAVE RESULTS
600 02AD B0 35              MOV    AL,35H                ;
601 02AF E6 80              OUT    MFG_PORT,AL          ;
602 02B1 58                POP    AX                    ;
603 02B2 3D AA55           CMP    AX,0AA55H             ;
604 02B5 74 16              JZ    ERROR_EXIT            ;
605 02B7 81 F8 AA55           CMP    BX,0AA55H             ;
606 02BB 74 10              JZ    ERROR_EXIT            ;
607 02BD 81 F9 AA55           CMP    CX,0AA55H             ;
608 02C1 74 0A              JZ    ERROR_EXIT            ;
609 02C3 B0 34              MOV    AL,34H                ; RESTORE CHECKPOINT
610 02C5 E6 80              OUT    MFG_PORT,AL          ;
611
612 ;----- SHUTDOWN
613
614 02C7                    NORMAL_EXIT:
615 02C7 B8 068F             MOV    AX,6*H+CMDS_SHUT_DOWN+NM1 ; ADDRESS FOR SHUTDOWN BYTE
616 02CA E8 0000 E          CALL   CMDS_WRITE_           ; SET GOOD ENDING
617 02CD                    ERROR_EXIT:
618 02CD E9 0000 E          JMP    PROC_SHUTDOWN
619
620 02D0                    POST3   ENDP
621 02D0                    CODE   ENDS
622
623 END
    
```

SECTION 5

```

1 PAGE 118,121
2 TITLE TEST4 ---- 06/10/85 POST AND BIOS UTILITY ROUTINES
3 .286C
4 .LIST
5 0000 CODE SEGMENT BYTE PUBLIC
6
7 PUBLIC BEEP
8 PUBLIC BLINK_INT
9 PUBLIC CMOS_READ
10 PUBLIC CMOS_WRITE
11 PUBLIC CONFIG_BAD
12 PUBLIC D1
13 PUBLIC DDS
14 PUBLIC DUMMY_RETURN_I
15 PUBLIC ERR_BEEP
16 PUBLIC E_MSG
17 PUBLIC INT_287
18 PUBLIC KBD_RESET
19 PUBLIC POST4
20 PUBLIC PROT_PRT_HEX
21 PUBLIC PROC_SHUTDOWN
22 PUBLIC PRT_HEX
23 PUBLIC PRT_MSG
24 PUBLIC P_MSG
25 PUBLIC RE_DIRECT
26 PUBLIC ROM_CHECK
27 PUBLIC ROM_CHECKSUM
28 PUBLIC SET_TOD
29 PUBLIC WAITF
30 PUBLIC XPC_BYTE
31
32 EXTRN E163:NEAR
33 EXTRN 0B7:42:NEAR
34 EXTRN ROM_ERR:NEAR
35 EXTRN XMIT_0042:NEAR
36
37 ASSUME CS:CODE,DS:DATA
38
39 0000 POST4:
40 ;----- CMOS_READ -----
41 ; READ BYTE FROM CMOS SYSTEM CLOCK CONFIGURATION TABLE ;
42 ; INPUT: (AL) = CMOS TABLE ADDRESS TO BE READ ;
43 ; BIT 7 = 0 FOR NMI ENABLED AND 1 FOR NMI DISABLED ON EXIT ;
44 ; BITS 6-0 = ADDRESS OF TABLE LOCATION TO READ ;
45 ; OUTPUT: (AL) VALUE AT LOCATION (AL) MOVED INTO (AL). IF BIT 7 OF (AL) WAS ;
46 ; ON THEN NMI LEFT DISABLED. DURING THE CMOS READ BOTH NMI AND ;
47 ; NORMAL INTERRUPTS ARE DISABLED TO PROTECT CMOS DATA INTEGRITY. ;
48 ; THE CMOS ADDRESS REGISTER IS POINTED TO A DEFAULT VALUE AND ;
49 ; THE INTERRUPT FLAG RESTORED TO THE ENTRY STATE ON RETURN. ;
50 ; ONLY THE (AL) REGISTER AND THE NMI STATE IS CHANGED. ;
51 ;-----
52
53
54 0000 CMOS_READ PROC NEAR ; READ LOCATION (AL) INTO (AL)
55 0000 9C PUSHF ; SAVE INTERRUPT ENABLE STATUS AND FLAGS
56 0001 D0 C0 ROL AL,1 ; MOVE NMI BIT TO LOW POSITION
57 0003 F9 STC ; FORCE NMI BIT ON IN CARRY FLAG
58 0004 D0 D8 RCR AL,1 ; HIGH BIT ON TO DISABLE NMI - OLD IN CY
59 0006 FA CLI ; DISABLE INTERRUPTS
60 0007 E6 70 OUT CMOS_PORT,AL ; ADDRESS LOCATION AND DISABLE NMI
61 0009 90 NOP ; I/O DELAY
62 000A E4 71 IN AL,CMOS_DATA ; READ THE REQUESTED CMOS LOCATION
63 000C 50 PUSH AX ; SAVE (AH) REGISTER VALUE AND CMOS BYTE
64 000D 80 IE MOV AL,CMOS_SHUT_DOWN*2 ; GET ADDRESS OF DEFAULT LOCATION
65 000F D0 D8 RCR AL,1 ; PUT ORIGINAL NMI MASK BIT INTO ADDRESS
66 0011 E6 70 OUT CMOS_PORT,AL ; SET DEFAULT TO READ ONLY REGISTER
67 0013 90 NOP ; I/O DELAY
68 0014 E4 71 IN AL,CMOS_DATA ; OPEN STANDBY LATCH
69 0016 58 POP AX ; RESTORE (AH) AND (AL) = CMOS BYTE
70 0017 0E PUSH CS ; *PLACE CODE SEGMENT IN STACK AND
71 0018 E5 0039 R CALL CMOS_POPF ; *HANDLE POPF FOR B- LEVEL 80286
72 001B C3 RET ; RETURN WITH FLAGS RESTORED
73
74 001C CMOS_READ ENDP
75
76 ;----- CMOS_WRITE -----
77 ; WRITE BYTE TO CMOS SYSTEM CLOCK CONFIGURATION TABLE ;
78 ; INPUT: (AL) = CMOS TABLE ADDRESS TO BE WRITTEN TO ;
79 ; BIT 7 = 0 FOR NMI ENABLED AND 1 FOR NMI DISABLED ON EXIT ;
80 ; BITS 6-0 = ADDRESS OF TABLE LOCATION TO WRITE ;
81 ; (AH) = NEW VALUE TO BE PLACED IN THE ADDRESSED TABLE LOCATION ;
82 ; OUTPUT: VALUE IN (AH) PLACED IN LOCATION (AL) WITH NMI LEFT DISABLED ;
83 ; IF BIT 7 OF (AL) IS ON. DURING THE CMOS UPDATE BOTH NMI AND ;
84 ; NORMAL INTERRUPTS ARE DISABLED TO PROTECT CMOS DATA INTEGRITY. ;
85 ; THE CMOS ADDRESS REGISTER IS POINTED TO A DEFAULT VALUE AND ;
86 ; THE INTERRUPT FLAG RESTORED TO THE ENTRY STATE ON RETURN. ;
87 ; ONLY THE CMOS LOCATION AND THE NMI STATE IS CHANGED. ;
88 ;-----
89
90
91 001C CMOS_WRITE PROC NEAR ; WRITE (AH) TO LOCATION (AL)
92 001C 9C PUSHF ; SAVE INTERRUPT ENABLE STATUS AND FLAGS
93 001D 50 PUSH AX ; SAVE WORK REGISTER VALUES
94 001E D0 C0 ROL AL,1 ; MOVE NMI BIT TO LOW POSITION
95 0020 F9 STC ; FORCE NMI BIT ON IN CARRY FLAG
96 0021 D0 D8 RCR AL,1 ; HIGH BIT ON TO DISABLE NMI - OLD IN CY
97 0023 FA CLI ; DISABLE INTERRUPTS
98 0024 E6 70 OUT CMOS_PORT,AL ; ADDRESS LOCATION AND DISABLE NMI
99 0026 8A C4 MOV AL,AH ; GET THE DATA BYTE TO WRITE
100 0028 E6 71 OUT CMOS_DATA,AL ; PLACE IN REQUESTED CMOS LOCATION
101 002A 80 IE MOV AL,CMOS_SHUT_DOWN*2 ; GET ADDRESS OF DEFAULT LOCATION
102 002C D0 D8 RCR AL,1 ; PUT ORIGINAL NMI MASK BIT INTO ADDRESS
103 002E E6 70 OUT CMOS_PORT,AL ; SET DEFAULT TO READ ONLY REGISTER
104 0030 90 NOP ; I/O DELAY
105 0031 E4 71 IN AL,CMOS_DATA ; OPEN STANDBY LATCH
106 0033 58 POP AX ; RESTORE WORK REGISTERS
107 0034 0E PUSH CS ; *PLACE CODE SEGMENT IN STACK AND
108 0036 E5 0039 R CALL CMOS_POPF ; *HANDLE POPF FOR B- LEVEL 80286
109 0038 C3 RET ; RETURN WITH FLAGS RESTORED
110
111 CMOS_WRITE ENDP
112 0039

```

```

113
114 0039          CMOS_POPF      PROC    NEAR          ; POPF FOR LEVEL B- PARTS
115 0039 CF      IRET              ; RETURN FAR AND RESTORE FLAGS
116
117 003A          CMOS_POPF      ENDP
118
119
120 003A          DDS          PROC    NEAR          ; LOAD (DS) TO DATA AREA
121 003A 2E: BE 1E 0040 R      MOV     DS,CS:DDSDATA ; PUT SEGMENT VALUE OF DATA AREA INTO DS
122 003F C3      RET              ; RETURN TO USER WITH (DS)= DATA
123
124 0040 ---- R      DDSDATA DW    DATA          ; SEGMENT SELECTOR VALUE FOR DATA AREA
125
126 0042          DDS          ENDP
127
128
129 ;----- E_MSG -- P_MSG -----
130 ; THIS SUBROUTINE WILL PRINT A MESSAGE ON THE DISPLAY
131 ;
132 ; ENTRY REQUIREMENTS:
133 ; SI = OFFSET (ADDRESS) OF MESSAGE BUFFER
134 ; CX = MESSAGE BYTE COUNT
135 ; MAXIMUM MESSAGE LENGTH IS 36 CHARACTERS
136 ; BP = BIT 0=E161/E162, BIT 1=CONFIG_BAD, 2-15= FIRST MSG OFFSET
137 ;-----
138 0042          E_MSG      PROC    NEAR
139 0042 F7 C5 3FFF      TEST   BP,03FFFFH ; CHECK FOR NOT FIRST ERROR MESSAGE
140 0046 75 08          JNZ   E_MSG1      ; SKIP IF NOT FIRST ERROR MESSAGE
141
142 0048 56          PUSH  SI          ; SAVE MESSAGE POINTER
143 0049 81 E6 3FFF      AND   SI,03FFFFH ; USE LOW 14 BITS OF MESSAGE OFFSET
144 004D 0B EE          OR    BP,SI       ; AS FIRST ERROR MESSAGE FLAG
145 004F 5E          POP   SI          ; (BIT 0 = E161/E162, BIT 1 = BAD_CONFIG)
146 0050          E_MSG1     PROC    NEAR
147 0050 E8 0069 R      CALL  P_MSG      ; PRINT MESSAGE
148 0053 1E          PUSH  DS          ; SAVE CALLERS (DS)
149 0054 E8 003A R      CALL  DDS        ; POINT TO POST/BIOS DATA SEGMENT
150 0057 F6 06 0010 R 01  TEST   BYTE PTR @EQUIP_FLAG,01H ; LOOP/HALT ON ERROR SWITCH ON ?
151 005C 74 02          JZ    MFG_HALT   ; YES - THEN GO TO MANUFACTURING HALT
152
153 005E 1F          POP   DS          ; RESTORE CALLERS (DS)
154 005F C3      RET
155
156 0060          MFG_HALT;   PROC    NEAR
157 0060 FA          CLI              ; DISABLE INTERRUPTS
158 0061 A0 0015 R      MOV     AL,@MFG_ERR_FLAG ; RECOVER ERROR INDICATOR
159 0064 E6 80          OUT   MFG_PORT,AL ; SET INTO MANUFACTURING PORT
160 0066 F4          HLT              ; HALT SYSTEM
161 0067 EB FT          JMP   MFG_HALT   ; HOT NMI TRAP
162
163 0069          E_MSG      ENDP
164
165
166 0069          P_MSG      PROC    NEAR          ; DISPLAY STRING FROM (CS:)
167 0069 2E: 8A 04      MOV     AL,CS:[SI] ; PUT CHARACTER IN (AL)
168 006C 46          INC   SI          ; POINT TO NEXT CHARACTER
169 006D 50          PUSH  AX          ; SAVE PRINT CHARACTER
170 006E E8 012E R      CALL  PRT_HEX    ; CALL VIDEO_IO
171 0071 58          POP   AX          ; RECOVER PRINT CHARACTER
172 0072 3C 0A          CMP   AL,LF      ; WAS IT LINE FEED?
173 0074 75 F3          JNE   P_MSG      ; NO, KEEP PRINTING STRING
174 0076 C3      RET
175
176 0077          P_MSG      ENDP
177
178
179 ;----- ERR_BEEP -----
180 ; THIS PROCEDURE WILL ISSUE LONG TONES (1-3/4 SECONDS) AND ONE OR
181 ; MORE SHORT TONES (9/32 SECOND) TO INDICATE A FAILURE ON THE
182 ; PLANAR BOARD, A BAD MEMORY MODULE, OR A PROBLEM WITH THE CRT.
183 ;
184 ; ENTRY PARAMETERS:
185 ; DH = NUMBER OF LONG TONES TO BEEP.
186 ; DL = NUMBER OF SHORT TONES TO BEEP.
187 ;-----
188 0077          ERR_BEEP  PROC    NEAR
189 0077 9C          PUSHF             ; SAVE FLAGS
190 0078 FA          CLI              ; DISABLE SYSTEM INTERRUPTS
191 0079 0A F6          OR    DH,DH      ; ANY LONG TONES TO BEEP
192 007B 74 1E          JZ    G1         ; NO, DO THE SHORT ONES
193 007D B3 70          MOV     BL,112   ; LONG BEEPS
194 007F B9 0500      MOV     CX,1280  ; COUNTER FOR LONG BEEPS (1-3/4 SECONDS)
195 0082 E8 0085 R      CALL  BEEP      ; DIVISOR FOR 932 HZ
196 0085 B9 C233      MOV     CX,49715 ; DO THE BEEP
197 0088 E8 00FB R      CALL  WAITF     ; 2/3 SECOND DELAY AFTER LONG BEEP
198 008B FE CE          DEC   DH         ; DELAY BETWEEN BEEPS
199 008D 75 EE          JNZ   G1         ; ANY MORE LONG BEEPS TO DO
200
201 008F 1E          PUSH  DS         ; LOOP TILL DONE
202 0090 E8 003A R      CALL  DDS        ; SAVE DS REGISTER CONTENTS
203 0093 80 3E 0012 R 01  CMP   @MFG_TST,01H ; MANUFACTURING TEST MODE?
204 0098 1F          POP   DS         ; RESTORE ORIGINAL CONTENTS OF (DS)
205 0099 74 C5          JE    MFG_HALT   ; YES - STOP BLINKING LED
206
207 009B          G1:         PROC    NEAR
208 009B B3 12          MOV     BL,18    ; SHORT BEEPS
209 009D B9 0488      MOV     CX,1208  ; COUNTER FOR A SHORT BEEP (9/32)
210 00A0 E8 0085 R      CALL  BEEP      ; DIVISOR FOR 987 HZ
211 00A3 B9 8178      MOV     CX,32144 ; DO THE SOUND
212 00A6 E8 00FB R      CALL  WAITF     ; 1/2 SECOND DELAY AFTER SHORT BEEP
213 00A9 FE CA          DEC   DL         ; DELAY BETWEEN BEEPS
214 00AB 75 EE          JNZ   G3         ; DONE WITH SHORT BEEPS COUNT
215
216 00AD B9 8178      MOV     CX,33144 ; LOOP TILL DONE
217 00B0 E8 00FB R      CALL  WAITF     ; 1/2 SECOND DELAY AFTER LAST BEEP
218 00B3 9D          POP   FPU        ; MAKE IT ONE SECOND DELAY BEFORE RETURN
219 00B4 C3      RET              ; RESTORE FLAGS TO ORIGINAL SETTINGS
220
221 00B5          ERR_BEEP  ENDP

```

SECTION 5

```

222 PAGE
223 ----- BEEP -----
224 | ROUTINE TO SOUND THE BEEPER USING TIMER 2 FOR TONE |
225 | ENTRY: |
226 | (BL) = DURATION COUNTER ( 1 FOR 1/64 SECOND ) |
227 | (CX) = FREQUENCY DIVISOR (1193180/FREQUENCY) (1331 FOR 886 HZ) |
228 | EXIT: |
229 | (AX), (BL), (CX) MODIFIED. |
230 -----
231
232 BEEP PROC NEAR ; SETUP TIMER 2
233 00B5 9C PUSHF ; SAVE INTERRUPT STATUS
234 00B6 FA CLI ; BLOCK INTERRUPTS DURING UPDATE
235 00B7 B0 B6 MOV AL,10110110B ; SELECT TIMER 2,LSB,MSB,BINARY
236 00B8 E4 43 OUT TIMER+3,AL ; WRITE THE TIMER MODE REGISTER
237 00B9 EB 00 JMP $+2 ; I/O DELAY
238 00BA 8A C1 MOV AL,CL ; DIVISOR FOR HZ (LOW)
239 00BB E6 42 OUT TIMER+2,AL ; WRITE TIMER 2 COUNT - LSB
240 00BC ED 00 JMP $+2 ; I/O DELAY
241 00CD 8A C5 MOV AL,CH ; DIVISOR FOR HZ (HIGH)
242 00CE E6 42 OUT TIMER+2,AL ; WRITE TIMER 2 COUNT - MSB
243 00CF E4 61 IN AL,PORT_B ; GET CURRENT SETTING OF PORT
244 00D0 8A E0 MOV AH,AL ; SAVE THAT SETTING
245 00D1 0C 03 OR AL,GATE2+SPK2 ; GATE TIMER 2 AND TURN SPEAKER ON
246 00D2 0C 61 OR PORT_B,AL ; AND RESTORE INTERRUPT STATUS
247 00D3 9D POPF
248 00D4 00 ;
249 00D5 B9 040B GT: MOV CX,1035 ; 1/64 SECOND PER COUNT (BL)
250 00D6 E8 00FB R CALL WAITF ; DELAY COUNT FOR 1/64 OF A SECOND
251 00D7 FE C8 DEC BL ; GO TO BEEP DELAY 1/64 COUNT
252 00D8 75 F6 JNZ GT ; (BL) LENGTH COUNT EXPIRED?
253 ; NO - CONTINUE BEEPING SPEAKER
254 00DA 9C PUSHF ; SAVE INTERRUPT STATUS
255 00DB FA CLI ; BLOCK INTERRUPTS DURING UPDATE
256 00DC E4 61 IN AL,PORT_B ; GET CURRENT PORT VALUE
257 00DD 0C FC OR AL,NOT (GATE2+SPK2) ; ISOLATE CURRENT SPEAKER BITS IN CASE
258 00DE 22 E0 AND AH,AL ; SOMEONE TURNED THEM OFF DURING BEEP
259 00DF 8A C4 MOV AL,AH ; RECOVER VALUE OF PORT
260 00E0 24 FC AND AL,NOT (GATE2+SPK2) ; FORCE SPEAKER DATA OFF
261 00E1 E6 61 OUT PORT_B,AL ; AND STOP SPEAKER TIMER
262 00E2 9D POPF ; RESTORE INTERRUPT FLAG STATE
263 00E3 B9 040B MOV CX,1035 ; FORCE 1/64 SECOND DELAY (SHORT)
264 00E4 E8 00FB R CALL WAITF ; MINIMUM DELAY BETWEEN ALL BEEPS
265 00E5 9C PUSHF ; SAVE INTERRUPT STATUS
266 00E6 FA CLI ; BLOCK INTERRUPTS DURING UPDATE
267 00E7 E4 61 IN AL,PORT_B ; GET CURRENT PORT VALUE IN CASE
268 00E8 24 03 OR AL,GATE2+SPK2 ; SOMEONE TURNED THEM ON
269 00E9 DA C4 AND AH,AL ; RECOVER VALUE OF PORT_B
270 00EA E4 61 OUT PORT_B,AL ; RESTORE SPEAKER STATUS
271 00EB 9D POPF ; RESTORE INTERRUPT FLAG STATE
272 00EC FA RET
273
274 BEEP ENDP
275
276 |--- WAITF |
277 | FIXED TIME WAIT ROUTINE (HARDWARE CONTROLLED - NOT PROCESSOR) |
278 | |
279 | ENTRY: |
280 | (CX) = COUNT OF 15.085737 MICROSECOND INTERVALS TO WAIT |
281 | MEMORY REFRESH TIMER 1 OUTPUT USED AS REFERENCE |
282 | |
283 | EXIT: |
284 | AFTER (CX) TIME COUNT (PLUS OR MINUS 16 MICROSECONDS) |
285 | (CX) = 0 |
286 |---
287 00FB PROC NEAR ; DELAY FOR (CX)*15.085737 US
288 00FC 90 PUSH AX ; SAVE WORK REGISTER (AH)
289 ;
290 00FD WAITF: IN AL,PORT_B ; USE TIMER 1 OUTPUT BITS
291 00FE E4 61 AND AL,REFRESH_BIT ; READ CURRENT COUNTER OUTPUT STATUS
292 00FF 3A C4 CMP AL,AH ; MASK FOR REFRESH DETERMINE BIT
293 0100 74 F8 JE WAITF ; DID IT JUST CHANGE
294 ; WAIT FOR A CHANGE IN OUTPUT LINE
295 ;
296 0101 8A E0 MOV AH,AL ; SAVE NEW FLAG STATE
297 0102 E2 F4 LOOP WAITF ; DECREMENT HALF CYCLES TILL COUNT END
298 ;
299 0103 58 POP AX ; RESTORE (AH)
300 0104 C3 RET ; RETURN (CX) = 0
301
302 WAITF ENDP
303
304 |--- CONFIG_BAD |
305 | SET CMOS_DIAG WITH CONFIG ERROR BIT (WITH NMI DISABLED) |
306 | (BP) BIT 14 SET ON TO INDICATE CONFIGURATION ERROR |
307 |---
308
309 010A CONFIG_BAD PROC NEAR
310 010B 50 PUSH AX
311 010C B8 BEBE MOV AX,X*(CMOS_DIAG+NMI) ; ADDRESS CMOS DIAGNOSTIC STATUS BYTE
312 010D E8 0000 R CALL CMOS_READ ; GET CURRENT VALUE
313 010E 0C 20 OR AL,BAD_CONFIG ; SET BAD CONFIGURATION BIT
314 010F 86 E0 XCHG AH,AL ; SETUP FOR WRITE
315 0110 E8 001C R CALL CMOS_WRITE ; UPDATE CMOS WITH BAD CONFIGURATION
316 0111 58 POP AX
317 0112 81 CD 4000 OR BP,04000H ; SET CONFIGURATION BAD FLAG IN (BP)
318 0113 C3 RET
319
320 011E CONFIG_BAD ENDP

```

```

321 PAGE
322 |---- XPC_BYTE -- XLATE_PR -- PRT_HEX -----|
323 |
324 | CONVERT AND PRINT ASCII CODE CHARACTERS |
325 |
326 | AL CONTAINS NUMBER TO BE CONVERTED. |
327 | AX AND BX DESTROYED. |
328 |-----|
329
330 011E XPC_BYTE PROC NEAR ; DISPLAY TWO HEX DIGITS
331 011E 50 PUSH AX ; SAVE FOR LOW NIBBLE DISPLAY
332 011F CD E8 04 SHR AL,4 ; NIBBLE SWAP
333 0122 E8 0128 R CALL XLATE_PR ; DO THE HIGH NIBBLE DISPLAY
334 0125 58 POP AX ; RECOVER THE NIBBLE
335 0126 24 0F AND AL,0FH ; ISOLATE TO LOW NIBBLE
336 ; FALL INTO LOW NIBBLE CONVERSION
337
338 0128 XLATE_PR PROC NEAR ; CONVERT 00-0F TO ASCII CHARACTER
339 0128 04 90 ADD AL,090H ; ADD FIRST CONVERSION FACTOR
340 012A 27 DAA ; ADJUST FOR NUMERIC AND ALPHA RANGE
341 012B 14 40 ADC AL,040H ; ADD CONVERSION AND ADJUST LOW NIBBLE
342 012D 27 DAA ; ADJUST HIGH NIBBLE TO ASCII RANGE
343
344 012E PRT_HEX PROC NEAR
345 012E B4 0E MOV AH,0EH ; DISPLAY CHARACTER IN (AL) COMMAND
346 0130 B7 00 MOV BH,0
347 0132 CD 10 INT 10H ; CALL VIDEO_IO
348 0134 C3 RET
349
350 0135 PRT_HEX ENDP
351 0135 XLATE_PR ENDP
352 0135 XPC_BYTE ENDP
353
354 |---- PRT_SEG -----|
355 | PRINT A SEGMENT VALUE TO LOOK LIKE A 21 BIT ADDRESS |
356 | DX MUST CONTAIN SEGMENT VALUE TO BE PRINTED |
357 |-----|
358
359 0135 PRT_SEG PROC NEAR
360 0135 BA C6 MOV AL,DH ; GET MSB
361 0137 E8 011E R CALL XPC_BYTE ; DISPLAY SEGMENT HIGH BYTE
362 013A BA C2 MOV AL,DL ; LSB
363 013C E8 011E R CALL XPC_BYTE ; DISPLAY SEGMENT LOW BYTE
364 013F B0 30 MOV AL,'0' ; PRINT A '0'
365 0141 E8 012E R CALL PRT_HEX ; TO MAKE LOOK LIKE ADDRESS
366 0144 B0 20 MOV AL,' ' ; ADD ENDING SPACE
367 0146 E8 012E R CALL PRT_HEX
368 0149 C3 RET
369
370 014A PRT_SEG ENDP
371
372 |---- PROT_PRT_HEX -----|
373 |
374 | PUT A CHARACTER TO THE DISPLAY BUFFERS WHEN IN PROTECTED MODE |
375 |
376 | (AL)= ASCII CHARACTER |
377 | (DI)= DISPLAY REGEN BUFFER POSITION |
378 |-----|
379
380 014A PROT_PRT_HEX PROC NEAR ; SAVE CURRENT SEGMENT REGISTERS
381 014A 06 PUSH ES
382 014B 57 PUSH DI
383 014C D1 ET SAL DI,1 ; MULTIPLY OFFSET BY TWO
384
385 |----- MONOCHROME VIDEO CARD
386
387 014E 6A 20 PUSH BYTE PTR C_BWCRT_PTR ; GET MONOCHROME BUFFER SEGMENT SELECTOR
388 0150 07 POP ES ; SET (ES) TO B/W DISPLAY BUFFER
389 0151 AA STOSB ; PLACE CHARACTER IN BUFFER
390 0152 4F DEC DI ; ADJUST POINTER BACK
391
392 |----- ENHANCED GRAPHICS ADAPTER
393
394 0153 6A 30 PUSH BYTE PTR E_CCRT_PTR ; ENHANCED COLOR DISPLAY POINTER LOW 64K
395 0155 07 POP ES ; LOAD SEGMENT SELECTOR
396 0156 AA STOSB ; PLACE CHARACTER IN BUFFER
397 0157 4F DEC DI ; ADJUST POINTER BACK
398 0158 6A 38 PUSH BYTE PTR E_CCRT_PTR ; ENHANCED COLOR DISPLAY POINTER HI 64K
399 015A 07 POP ES ; LOAD SEGMENT SELECTOR
400 015B AA STOSB ; PLACE CHARACTER IN BUFFER
401 015C 4F DEC DI ; ADJUST POINTER BACK
402
403 |----- COMPATIBLE COLOR
404
405 015D 6A 28 PUSH BYTE PTR C_CCRT_PTR ; SET (DS) TO COMPATIBLE COLOR MEMORY
406 015F 07 POP ES ; SAVE WORK REGISTERS
407 0160 58 PUSH BX
408 0161 52 PUSH DX
409 0162 51 PUSH CX
410 0163 93 C9 XOR CX,CX ; TIMEOUT LOOP FOR "BAD" HARDWARE
411 0165 BA 03DA MOV DX,03DAH ; STATUS ADDRESS OF COLOR CARD
412 0168 93 XCHG AX,BX ; SAVE IN (BX) REGISTER
413 0169
414 0169 EC IN AL,DX ; GET COLOR CARD STATUS
415 016A A8 09 TEST AL,RVRT+RHRZ ; CHECK FOR VERTICAL RETRACE (OR HORZ)
416 016C E1 FB LOOPZ PROT_S ; TIMEOUT LOOP TILL FOUND
417 016E 93 XCHG AX,BX ; RECOVER CHARACTERS
418 016F AA STOSB ; PLACE CHARACTER IN BUFFER
419
420 0170 59 POP CX ; RESTORE REGISTERS
421 0171 5A POP DX
422 0172 5B POP BX
423 0173 5F POP DI
424 0174 07 POP ES
425 0175 C3 RET
426
427 0176 PROT_PRT_HEX ENDP
    
```

SECTION 5

```

428 PAGE
429 ;
430 ; ROM CHECKSUM SUBROUTINE
431 ;
432
433 0176 ROM_CHECKSUM PROC NEAR
434 0176 2B C9 SUB CX,CX ; NUMBER OF BYTES TO ADD IS 64K
435
436 0178 ROM_CHECKSUM_CNT: ; ENTRY FOR OPTIONAL ROM TEST
437 0178 32 C0 XOR AL,AL
438 017A ROM_L:
439 017A 02 07 ADD AL,[BX] ; GET (DS:BX)
440 017C 43 INC BX ; POINT TO NEXT BYTE
441 017D E2 FB LOOP ROM_L ; ADD ALL BYTES IN ROM MODULE
442
443 017F 0A C0 OR AL,AL ; SUM = 0?
444 0181 C3 RET
445
446 0182 ROM_CHECKSUM ENDP
447
448 ;
449 ; THIS ROUTINE CHECKSUMS OPTIONAL ROM MODULES AND
450 ; IF CHECKSUM IS OK, CALLS INITIALIZATION/TEST CODE IN MODULE
451 ;
452
453 0182 ROM_CHECK PROC NEAR
454 0182 88 ---- R MOV AX,DATA ; POINT ES TO DATA AREA
455 0185 8E C0 MOV ES,AX
456 0187 2A E4 SUB AH,AH ; ZERO OUT AH
457 0189 8A 47 02 MOV AL,[BX+2] ; GET LENGTH INDICATOR
458 018C C1 E0 09 SHL AX,9 ; MULTIPLY BY 512
459 018F 8B C8 MOV CX,AX ; SET COUNT
460 0191 C1 E8 04 SHR AX,4 ; SET POINTER TO NEXT MODULE
461 0194 03 D0 ADD DX,AX ; DO CHECKSUM
462 0196 E8 0178 R CALL ROM_CHECKSUM_CNT
463 0199 74 05 JZ ROM_CHECK_1
464
465 019B E8 0000 E CALL ROM_ERR ; POST CHECKSUM ERROR
466 019E EB 13 JMP SHORT ROM_CHECK_END ; AND EXIT
467
468 ROM_CHECK_1:
469 01A0 52 PUSH DX ; SAVE POINTER
470 01A1 26: C7 06 0067 R 0003 MOV ES:#10 ROM_INIT,0003H ; LOAD OFFSET
471 01A8 26: 8C 1E 0069 R MOV ES:#10 ROM_SEG_DS ; LOAD SEGMENT
472 01AD 26: FF 1E 0067 R CALL DWORD PTR ES:#10 ROM_INIT; CALL INITIALIZE/TEST ROUTINE
473 01B2 5A POP DX
474
475 01B3 ROM_CHECK_END:
476 01B3 C3 RET ; RETURN TO CALLER
477
478 01B4 ROM_CHECK ENDP
479
480 ;--- KBD RESET
481 ; THIS PROCEDURE WILL SEND A SOFTWARE RESET TO THE KEYBOARD.
482 ; SCAN CODE 0AAH SHOULD BE RETURNED TO THE PROCESSOR.
483 ; SCAN CODE 065H IS DEFINED FOR MANUFACTURING TEST
484 ;
485
486 01B4 KBD_RESET PROC NEAR
487 01B4 80 FF MOV AL,OFFH ; SET KEYBOARD RESET COMMAND
488 01B6 E8 0000 E CALL XMIT_8042 ; GO ISSUE THE COMMAND
489 01B9 E3 23 JCXZ G13 ; EXIT IF ERROR
490
491 01BB 3C FA CMP AL,KB_ACK
492 01BD 75 1F JNZ G13
493
494 01BF 80 FD MOV AL,OFDM ; ENABLE KEYBOARD INTERRUPTS
495 01C1 E6 21 OUT INTA01,AL ; WRITE 8259 INTERRUPT MASK REGISTER
496 01C3 C6 06 006B R 00 MOV #INTR_FLAG,0 ; RESET INTERRUPT INDICATOR
497 01C8 FB STI ; ENABLE INTERRUPTS
498 01C9 B3 0A MOV BL,10 ; TRY FOR 400 MILLISECONDS
499 01CB 2B C9 SUB CX,CX ; SETUP INTERRUPT TIMEOUT COUNT
500 01CD TEST #INTR_FLAG,02H ; DID A KEYBOARD INTERRUPT OCCUR ?
501 01D2 75 06 JNZ G12 ; YES - READ SCAN CODE RETURNED
502 01D4 E2 F7 LOOP G11 ; NO - LOOP TILL TIMEOUT
503
504 01D6 FE CB DEC BL ; TRY AGAIN
505 01D8 75 F3 JNZ G11
506 01DA IN AL,PORT_A ; READ KEYBOARD SCAN CODE
507 01DC 8A D8 MOV BL,AL ; SAVE SCAN CODE JUST READ
508 01DE C3 RET ; RETURN TO CALLER
509
510 01DF KBD_RESET ENDP
511
512 ;
513 ; BLINK LED PROCEDURE FOR MFG RUN-IN TESTS
514 ; IF LED IS ON, TURN IT OFF. IF OFF, TURN ON.
515 ;
516
517 01DF BLINK_INT PROC NEAR
518 01E0 50 STI AX ; SAVE AX REGISTER CONTENTS
519 01E1 E4 80 IN AL,MFG_PORT ; READ CURRENT VALUE OF MFG_PORT
520 01E3 34 40 XOR AL,01000000B ; FLIP CONTROL BIT
521 01E5 E6 80 OUT MFG_PORT,AL
522 01E7 80 20 MOV AL,E01
523 01E9 E6 20 OUT INTA00,AL
524 01EB 58 POP AX ; RESTORE AX REGISTER
525 01EC CF IRET
526
527 01ED BLINK_INT ENDP

```



```

532                                     PAGE
533 -----
534 THIS ROUTINE INITIALIZES THE TIMER DATA AREA IN THE ROM BIOS
535 DATA AREA. IT IS CALLED BY THE POWER ON ROUTINES. IT CONVERTS
536 HR:MIN:SEC FROM CMOS TO TIMER TIC'S. IF CMOS IS INVALID, TIMER
537 IS SET TO ZERO.
538
539 INPUT NONE PASSED TO ROUTINE BY CALLER
540 CMOS LOCATIONS USED FOR TIME
541
542 OUTPUT *TIMER_LOW
543 *TIMER_HIGH
544 *TIMER_OFI
545 ALL REGISTERS UNCHANGED
546 -----
547 = 0012 COUNTS_SEC EQU 18 ; TIMER DATA CONVERSION EQUATES
548 = 0444 COUNTS_MIN EQU 1092 ;
549 = 0007 COUNTS_HOUR EQU 7 ; 65543 - 65536
550 = 0080 UPDATE_TIMER EQU 10000000B ; RTC UPDATE IN PROCESS BIT MASK
551
552 SET_TOD PROC NEAR
553 01ED 60 PUSHA
554 01EE 1E PUSH DS
555 01EF E8 003A R CALL DDS ; ESTABLISH SEGMENT
556 01F2 2B C0 SUB AX,AX
557 01F4 A2 0070 R MOV *TIMER_OFI,AL ; RESET TIMER ROLL OVER INDICATOR
558 01F7 A3 006C R MOV *TIMER_LOW,AX ; AND TIMER COUNT
559 01FA A3 006E R MOV *TIMER_HIGH,AX
560 01FD 80 8E MOV AL,CMOS_DIAG+NM1 ; CHECK CMOS VALIDITY
561 01FF E8 0000 R CALL CMOS_READ ; READ DIAGNOSTIC LOCATION IN CMOS
562 0202 24 C4 AND AL,BAD_BAT+BAD_CKSUM+CMOS_CLK_FAIL ; BAD BATTERY, CKSUM ERROR, CLOCK ERROR
563
564 0204 75 48 JNZ POD_DONE ; CMOS NOT VALID -- TIMER SET TO ZERO
565 0206 2B C9 SUB CX,CX
566 0208
567 0208 80 8A MOV AL,CMOS_REG_A+NM1 ; ACCESS REGISTER A
568 020A E8 0000 R CALL CMOS_READ ; READ CMOS CLOCK REGISTER A
569 020D A8 80 TEST AL,UPDATE_TIMER ; ARE THE SECONDS WITHIN LIMITS?
570 020F E1 F7 LOOPZ UIP ; WAIT TILL UPDATE BIT IS ON
571
572 0211 E3 58 JCXZ POD_DONE ; CMOS CLOCK STUCK IF TIMEOUT
573 0213 2B C9 SUB CX,CX
574 0215
575 0215 80 8A MOV AL,CMOS_REG_A+NM1 ; ACCESS REGISTER A
576 0217 E8 0000 R CALL CMOS_READ ; READ CMOS CLOCK REGISTER A
577 021A A8 80 TEST AL,UPDATE_TIMER ; ARE THE SECONDS WITHIN LIMITS?
578 021C E0 F7 LOOPNZ UIPOFF ; NEXT WAIT TILL END OF UPDATE
579
580 021E E3 4E JCXZ POD_DONE ; CMOS CLOCK STUCK IF TIMEOUT
581
582 0220 80 80 MOV AL,CMOS_SECONDS+NM1 ; TIME JUST UPDATED
583 0222 E8 0000 R CALL CMOS_READ ; ACCESS SECONDS VALUE IN CMOS
584 0225 3C 59 CMP AL,59H ; ARE THE SECONDS WITHIN LIMITS?
585 0227 77 48 JA TOD_ERROR ; GO IF NOT
586
587 0229 E8 0287 R CALL CVT_BINARY ; CONVERT IT TO BINARY
588 022C 8B C8 MOV CX,AX ; MOVE COUNT TO ACCUMULATION REGISTER
589 022E C1 E9 02 SHR CX,2 ; ADJUST FOR SYSTEMATIC SECONDS ERROR
590 0231 B3 12 MOV BL,COUNTS_SEC ; COUNT FOR SECONDS
591 0233 F4 E3 MUL BL ;
592 0235 03 C8 ADD CX,AX
593 0237 80 82 MOV AL,CMOS_MINUTES+NM1 ; ACCESS MINUTES VALUE IN CMOS
594 0239 E8 0000 R CALL CMOS_READ ; ARE THE MINUTES WITHIN LIMITS?
595 023C 3C 59 CMP AL,59H ; GO IF NOT
596 023E 77 31 JA TOD_ERROR ; CONVERT IT TO BINARY
597 0240 E8 0287 R CALL CVT_BINARY ; CONVERT IT TO BINARY
598 0243 50 AX PUSH AX ; SAVE MINUTES COUNT
599 0244 D1 E8 SHR AX,1 ; ADJUST FOR SYSTEMATIC MINUTES ERROR
600 0246 03 C8 ADD CX,AX ; ADD ADJUSTMENT TO COUNT
601 0248 58 AX POP AX ; RECOVER BCD MINUTES VALUE
602 0249 8B 0444 MOV BX,COUNTS_MIN ; COUNT FOR MINUTES
603 024C F7 E3 MUL BX ; ADD TO ACCUMULATED VALUE
604 024E 03 C8 ADD CX,AX ; ACCESS HOURS VALUE IN CMOS
605 0250 80 84 MOV AL,CMOS_HOURS+NM1 ; ARE THE HOURS WITHIN LIMITS?
606 0252 E8 0000 R CALL CMOS_READ ; GO IF NOT
607 0255 3C 23 CMP AL,23H ; CONVERT IT TO BINARY
608 0257 77 18 JA TOD_ERROR ; MOVE COUNT TO ACCUMULATION REGISTER
609
610 0259 E8 0287 R CALL CVT_BINARY ; CONVERT IT TO BINARY
611 025C 8B D0 MOV DX,AX ;
612 025E B3 07 MOV BL,COUNTS_HOUR ; COUNT FOR HOURS
613 0260 F4 E3 MUL BL ;
614 0262 03 C1 ADD AX,CX ;
615 0264 83 D2 00 ADC DX,0000H ;
616 0267 89 16 006E R MOV *TIMER_HIGH,DX ;
617 0268 A3 006C R MOV *TIMER_LOW,AX ;
618 026E
619 026E 1F POP DS ;
620 026F 61 POPA ;
621 0270 C3 RET
622
623 0271
624 0271 1F POP DS ; RESTORE SEGMENT
625 0272 61 POPA ; RESTORE REGISTERS
626 0273 BE 0000 E MOV SI,OFFSET E163 ; DISPLAY CLOCK ERROR
627 0276 E8 0042 R CALL E_MSG ;
628 0279 88 8E8E MOV AX,X*(CMOS_DIAG+NM1) ; SET CLOCK ERROR IN STATUS
629 027C E8 0000 R CALL CMOS_READ ; READ DIAGNOSTIC CMOS LOCATION
630 027F 0C 04 OR AL,CMOS_CLK_FAIL ; SET NEW STATUS WITH CMOS CLOCK ERROR
631 0281 86 C4 XCHG AL,AH ; MOVE NEW STATUS TO WORK REGISTER
632 0283 E8 001C R CALL CMOS_WRITE ; UPDATE STATUS LOCATION
633 0286 C3 RET
634
635 0287
636
637 0287
638 0287 8A E0 MOV AH,AL ; UNPACK 2 BCD DIGITS IN AL
639 0289 C0 EC 04 SHR AH,4 ;
640 028C 24 0F AND AL,0FH ; RESULT IS IN AX
641 028E D5 0A AAD ; CONVERT UNPACKED BCD TO BINARY
642 0290 C3 RET
643
644 0291 CVT_BINARY ENDP
    
```

SECTION 5

```

646 PAGE
647 |--- D11 -- INT ?? H -- ( IRQ LEVEL ?? ) -----
648 | TEMPORARY INTERRUPT SERVICE ROUTINE FOR POST
649 |
650 | THIS ROUTINE IS ALSO LEFT IN PLACE AFTER THE POWER ON DIAGNOSTICS
651 | TO SERVICE UNUSED INTERRUPT VECTORS. LOCATION *@INTR_FLAG* WILL
652 | CONTAIN EITHER:
653 | 1) LEVEL OF HARDWARE INTERRUPT THAT CAUSED CODE TO BE EXECUTED, OR
654 | 2) "FF" FOR A NON-HARDWARE INTERRUPT THAT WAS EXECUTED ACCIDENTALLY.
655 |-----
656
657 0291 PROC NEAR
658 0291 50 PUSH AX ; SAVE REGISTER AX CONTENTS
659 0292 53 PUSH BX
660 0293 80 0B MOV AL,0BH ; READ IN-SERVICE REGISTER
661 0294 E6 20 OUT INTA00,AL ; (FIND OUT WHAT LEVEL BEING
662 0295 E4 20 JMP $+2 ; SERVICED)
663 0296 8A E0 IN AL,INTA00 ; GET LEVEL
664 0297 0A C4 OR AL,AH ; SAVE IT
665 0298 75 04 JNZ HW_INT ; 00? (NO HARDWARE ISR ACTIVE)
666
667 02A1 B4 FF MOV AH,OFFH
668 02A2 EB 2F JMP SHORT SET_INTR_FLAG ; SET FLAG TO "FF" IF NON-HARDWARE
669 02A5
670 02A5 B0 0B HW_INT: MOV AL,0BH ; READ IN-SERVICE REGISTER FROM
671 02A7 E6 A0 OUT INTB00,AL ; INTERRUPT CHIP #2
672 02A8 EB 00 JMP $+2 ; I/O DELAY
673 02AB E4 A0 IN AL,INTB00 ; CHECK THE SECOND INTERRUPT CHIP
674 02AD 8A F8 MOV BH,AL ; SAVE IT
675 02AF 0A FF OR BH,BH ; CONTINUE IF NOT
676 02B1 74 10 JZ NOT_SEC
677
678 02B3 E4 A1 IN AL,INTB01 ; GET SECOND INTERRUPT MASK
679 02B5 0A CT OR AL,BH ; MASK OFF LEVEL BEING SERVICED
680 02B7 EB 00 JMP $+2 ; I/O DELAY
681 02B9 E6 A1 OUT INTB01,AL ; SEND EOI TO SECOND CHIP
682 02BB 80 20 MOV AL,E01 ; I/O DELAY
683 02BD EB 00 JMP $+2
684 02BF E6 A0 OUT INTB00,AL ; I/O DELAY
685 02C1 EB 0D JMP SHORT IS_SEC
686 02C3
687 02C3 E4 21 NOT_SEC: IN AL,INTA01 ; GET CURRENT MASK VALUE
688 02C5 EB 00 JMP $+2 ; I/O DELAY
689 02C7 80 E4 FB AND AH,0FBH ; DO NOT DISABLE SECOND CONTROLLER
690 02CA 0A C4 OR AL,AH ; MASK OFF LEVEL BEING SERVICED
691 02CC E6 21 OUT INTA01,AL ; SET NEW INTERRUPT MASK
692 02CE EB 00 JMP $+2 ; I/O DELAY
693 02D0
694 02D0 B0 20 IS_SEC: MOV AL,E01
695 02D2 E6 20 OUT INTA00,AL
696 02D4
697 02D4 5B SET_INTR_FLAG: POP BX ; RESTORE (BX) FROM STACK
698 02D5 1E PUSH DS ; SAVE ACTIVE (DS)
699 02D6 E8 003A R CALL DDS ; SET DATA SEGMENT
700 02D9 88 26 006B R MOV @INTR_FLAG,AH ; SET FLAG
701 02DD 1F POP DS
702 02DE 58 POP AX ; RESTORE REGISTER AX CONTENTS
703 02DF DUMMY_RETURN_I: IRET ; NEED IRET FOR VECTOR TABLE
704 02DF CF
705
706 02E0 D11 ENDP
707
708 |--- HARDWARE INT 71 H -- ( IRQ LEVEL 9 ) -- TO INT 0A H -----
709 | REDIRECT SLAVE INTERRUPT 9 TO INTERRUPT LEVEL 2
710 | THIS ROUTINE FIELDS LEVEL 9 INTERRUPTS AND
711 | CONTROL IS PASSED TO MASTER INTERRUPT LEVEL 2
712 |-----
713
714 02E0 RE_DIRECT PROC NEAR
715 02E0 50 PUSH AX ; SAVE (AX)
716 02E1 B0 20 MOV AL,E01
717 02E3 E6 A0 OUT INTB00,AL ; EOI TO SLAVE INTERRUPT CONTROLLER
718 02E5 58 POP AX ; RESTORE (AX)
719 02E6 CD 0A INT 0AH ; GIVE CONTROL TO HARDWARE LEVEL 2
720
721 02E8 CF IRET ; RETURN
722
723 02E9 RE_DIRECT ENDP
724
725 |--- HARDWARE INT 75 H -- ( IRQ LEVEL 13 ) -----
726 | SERVICE X287 INTERRUPTS
727 | THIS ROUTINE FIELDS X287 INTERRUPTS AND CONTROL
728 | IS PASSED TO THE NMI INTERRUPT HANDLER FOR
729 | COMPATIBILITY.
730 |-----
731
732 02E9 INT_287 PROC NEAR
733 02E9 50 PUSH AX ; SAVE (AX)
734 02EA 32 C0 XOR AL,AL
735 02EC E6 F0 OUT X287,AL ; REMOVE THE INTERRUPT REQUEST
736
737 02EE B0 20 MOV AL,E01 ; ENABLE THE INTERRUPT
738 02F0 E6 A0 OUT INTB00,AL ; THE SLAVE
739 02F2 E6 20 OUT INTA00,AL ; THE MASTER
740 02F4 58 POP AX ; RESTORE (AX)
741 02F5 CD 02 INT 02H ; GIVE CONTROL TO NMI
742
743 02F7 CF IRET ; RETURN
744
745 02F8 INT_287 ENDP
746
747 02F8 PROC_SHUTDOWN PROC ; COMMON 80286 SHUTDOWN WAIT
748
749 02F8 B0 FE MOV AL,SHUT_CMD ; SHUTDOWN COMMAND
750 02FA E6 64 OUT STATUS_PORT,AL ; SEND TO KEYBOARD CONTROL PORT
751 02FC
752 02FC F4 PROC_S: HLT ; WAIT FOR 80286 RESET
753 02FD EB FD JMP PROC_S ; INSURE HALT
754
755 02FF PROC_SHUTDOWN ENDP
756 02FF CODE ENDS
757 02FF END

```

```

1 PAGE 118,121
2 TITLE TEST5 ---- 06/10/85 EXCEPTION INTERRUPT TEST HANDLERS
3 .286C
4 .LIST
5 0000 CODE SEGMENT BYTE PUBLIC
6
7 PUBLIC POST5
8 PUBLIC SYSINIT1
9
10
11
12 -----
13 EXCEPTION INTERRUPT ROUTINE
14 -----
15
16 POST5: ASSUME CS:CODE,DS:ABS0
17 EXC_00: MOV AL,90H ; GO TEST IF EXCEPTION WAS EXPECTED
18 0002 E9 00B2 R JMP TEST_EXC
19
20 EXC_01: MOV AL,91H ; GO TEST IF EXCEPTION WAS EXPECTED
21 0007 E9 00B2 R JMP TEST_EXC
22
23 EXC_02: MOV AL,92H ; GO TEST IF EXCEPTION WAS EXPECTED
24 000C E9 00B2 R JMP TEST_EXC
25
26 EXC_03: MOV AL,93H ; GO TEST IF EXCEPTION WAS EXPECTED
27 0011 E9 00B2 R JMP TEST_EXC
28
29 EXC_04: MOV AL,94H ; GO TEST IF EXCEPTION WAS EXPECTED
30 0016 E9 00B2 R JMP TEST_EXC
31
32 EXC_05: PUSH ES ; LOAD ES REGISTER WITH SELECTOR
33 001A 6A 48 PUSH BYTE PTR ES_TEMP
34 001C 07 POP ES
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114

```

SECTION 5

```

115 008C          EXC_29:    MOV     AL,0ADH          ;
116 008C B0 AD    JMP     SHORT TEST_EXC  ; <<<< SET CHECKPOINT <<<<
117 008E EB 22          ; GO TEST IF EXCEPTION WAS EXPECTED
118 0090          EXC_30:    MOV     AL,0AEH          ;
119 0090 B0 AE    JMP     SHORT TEST_EXC  ; <<<< SET CHECKPOINT <<<<
120 0092 EB 1E          ; GO TEST IF EXCEPTION WAS EXPECTED
121 0094          EXC_31:    MOV     AL,0AFH          ;
122 0094 B0 AF    JMP     SHORT TEST_EXC  ; <<<< SET CHECKPOINT <<<<
123 0096 EB 1A          ; GO TEST IF EXCEPTION WAS EXPECTED
124 0098          SYS_32:    MOV     AL,0B0H          ;
125 0098 B0 B0    JMP     SHORT TEST_EXC  ; <<<< SET CHECKPOINT <<<<
126 009A EB 16          ; GO TEST IF INTERRUPT WAS EXPECTED
127 009C          SYS_33:    MOV     AL,0B1H          ;
128 009C B0 B1    JMP     SHORT TEST_EXC  ; <<<< SET CHECKPOINT <<<<
129 009E EB 12          ; GO TEST IF INTERRUPT WAS EXPECTED
130 00A0          SYS_34:    MOV     AL,0B2H          ;
131 00A0 B0 B2    JMP     SHORT TEST_EXC  ; <<<< SET CHECKPOINT <<<<
132 00A2 EB 0E          ; GO TEST IF INTERRUPT WAS EXPECTED
133 00A4          SYS_35:    MOV     AL,0B3H          ;
134 00A4 B0 B3    JMP     SHORT TEST_EXC  ; <<<< SET CHECKPOINT <<<<
135 00A6 EB 0A          ; GO TEST IF INTERRUPT WAS EXPECTED
136 00A8          SYS_36:    MOV     AL,0B4H          ;
137 00A8 B0 B4    JMP     SHORT TEST_EXC  ; <<<< SET CHECKPOINT <<<<
138 00AA EB 06          ; GO TEST IF INTERRUPT WAS EXPECTED
139 00AC          SYS_37:    MOV     AL,0B5H          ;
140 00AC B0 B5    JMP     SHORT TEST_EXC  ; <<<< SET CHECKPOINT <<<<
141 00AE EB 02          ; GO TEST IF INTERRUPT WAS EXPECTED
142 00B0          SYS_38:    MOV     AL,0B6H          ;
143 00B0 B0 B6    JMP     SHORT TEST_EXC  ; <<<< SET CHECKPOINT <<<<
144 00B2          ; GO TEST IF INTERRUPT WAS EXPECTED
145
146 00B2          TEST_EXC1:  OUT     MFG_PORT,AL      ; OUTPUT THE CHECKPOINT
147 00B2 E6 80      CMP     AL,0AFH          ; CHECK FOR EXCEPTION
148 00B4 3C AF      JA     TEST_EXC0         ; GO IF A SYSTEM INTERRUPT
149 00B6 77 1C          ;
150
151 00B8 1E          PUSH    DS                ; SAVE THE CURRENT DATA SEGMENT
152 00B9 6A 08      PUSH   BYTE PTR GDT_PTR
153 00BB 1F          POP     DS
154 00BC C7 06 0048 FFFF MOV    DS,ES TEMP_SEG_LIMIT,MAX_SEG_LEN
155 00BE C6 06 004D 93 MOV    BYTE PTR DS:(ES_TEMP_DATA_ACC_RIGHTS),CPL0_DATA_ACCESS
156 00C7 6A 48      PUSH   BYTE PTR ES_TEMP
157 00C9 07          POP     ES
158 00CA 1F          POP     DS                ; RESTORE REGISTERS
159 00CB 5A          POP     DX                ; CHECK IF CODE SEGMENT SECOND ON STACK
160 00CC 59          POP     CX
161 00CD 51          PUSH   CX
162 00CE 83 F9 40    CMP    CX,SYS_ROM_CS     ;
163 00D1 75 01      JNZ    TEST_EXC0         ; CONTINUE IF ERROR CODE
164
165 00D3 52          PUSH   DX                ; PUT SEGMENT BACK ON STACK
166 00D4          TEST_EXC0:  XCHG   AH,AL            ; SAVE THE CHECKPOINT
167 00D4 86 E0      IN     AL,DMA_PAGE+0AH   ;
168 00D6 E4 8B      CMP    AL,AH            ; WAS THE EXCEPTION EXPECTED?
169 00D8 3A C4      JZ     TEST_EXC3         ; GO IF YES
170 00DA 74 0E      JZ     TEST_EXC1         ;
171 00DC          TEST_EXC1:  IN     AL,MFG_PORT       ; CHECK THE CURRENT CHECKPOINT
172 00DC E4 80      CMP    AL,0B8H          ; HALT IF CHECKPOINT BELOW 3BH
173 00DE 3C 3B      JB     TEST_EXC2         ;
174 00E0 72 01      JBE   TEST_EXC2         ;
175 00E2 CF          IRET
176
177 00E3          TEST_EXC2:  XCHG   AH,AL            ; OUTPUT THE CURRENT CHECKPOINT
178 00E3 86 E0      OUT    MFG_PORT,AL      ; <<<< CHECKPOINT 90 THRU B5 <<<<
179 00E5 E6 80      HLT
180 00E7 F4          JMP    TEST_EXC2         ; INSURE SYSTEM HALT
181 00E8 EB F9
182
183 00EA          TEST_EXC3:  SUB    AL,AL            ; CLEAR DMA PAGE
184 00EA 2A C0      DMA_PAGE+0AH,AL        ;
185 00EC E6 8B      MOV    AX,0100H         ; FOR BOUND INSTRUCTION EXPECTED (INT 5)
186 00EE B8 0100    MOV    AX,0100H         ; RETURN
187 00F1 CF          IRET
188
189
190
191
192
193
194
195
196 00F2          SYSINIT1:  PROC    NEAR             ;
197 00F2 FA          CLI                                     ; NO INTERRUPTS ALLOWED
198 00F4 B0 81      PUSH   BP                ; SAVE BP
199 00F6 E6 80      MOV    AL,81H           ;
200 00F8 E8 0149 R  OUT    MFG_PORT,AL      ; <<<< CHECKPOINT 81 <<<<
201 00FB 8B EF      CALL  SIDT_BLD           ;
202
203
204
205 00FD B8 0800    MOV    AX,SYS_IDT_LEN   ; SAVE THE POINTER TO JUST PAST THE IDT
206 0100 AB          STOSW                          ; AS WE HAVE NO SDA, USE THE SIX BYTES
207 0101 B8 D0A0    MOV    AX,SYS_IDT_LOC   ; HERE TO LOAD THE IDTR. WE WILL SIDT
208 0104 AB          STOSW                          ; WHEN WE GET TO SDA INITIALIZATION.
209 0105 B8 0000    MOV    AX,0              ; SEGMENT LIMIT = LENGTH OF IDT
210 0108 AB          STOSW                          ; STORE THAT AS IDT LIMIT
211
212 0109 26          SEGOV ES                  ; LOAD THE IDT
213          DB 026H
214 010A 0F          LIDT [BP]                    ; REGISTER FROM THIS AREA
215 010B          DB 00FH
216 010B 8B 5E 00  + ?70001 LABEL BYTE
217 010E          MOV    BX,WORD PTR [BP]
218 010B          LABEL BYTE
219 010E 01          + ?70002 LABEL BYTE
220 010E          ORG   OFFSET CS:??0001
221 010E 01          + DB 001H
222 010E          + ORG   OFFSET CS:??0002
223          MOV    DI,BP            ; ES:DI NOW --> END OF IDT AGAIN
224
225          I----- BUILD THE GDT.
226 0110 BF D8A0    MOV    DI,GDT_LOC
227 0113 E8 0140 R  CALL  GDT_BLD
228 0116 8B EF      MOV    BP,DI
229 0118 B8 0888    MOV    AX,GDT_LEN      ; SAVE THE ES:DI POINTER
230
231
232
233

```



```

325 PAGE
326 ; THE FOLLOWING DATA DEFINES THE PRE-INITIALIZED GDT FOR POST TESTS.
327 ; THESE MUST BE INITIALIZED IN THE ORDER IN WHICH THEY APPEAR IN THE
328 ; GDT_DEF STRUCTURE DEFINITION AS IT IS IN 'SYSDATA.INC'.
329
330 = 01AF GDT_DATA_START EQU $
331
332 ;----- FIRST ENTRY UNUSABLE - (UNUSED_ENTRY)
333
334 DW 0 ; SEGMENT LIMIT
335 DW 0 ; SEGMENT BASE ADDRESS - LOW WORD
336 DB 0 ; SEGMENT BASE ADDRESS - HIGH BYTE
337 DB 0 ; ACCESS RIGHTS BYTE
338 DW 0 ; RESERVED - MUST BE ZERO
339
340 ;----- THE GDT ITSELF - (GDT_PTR)
341
342 DW GDT_LEN ; SEGMENT LIMIT
343 DW GDT_LO ; SEGMENT BASE ADDRESS - LOW WORD
344 DB 0 ; SEGMENT BASE ADDRESS - HIGH BYTE
345 DB CPLD_DATA_ACCESS ; ACCESS RIGHTS BYTE
346 DW 0 ; RESERVED - MUST BE ZERO
347
348 ;----- THE SYSTEM IDT DESCRIPTOR - (SYS_IDT_PTR)
349
350 DW SYS_IDT_LEN ; SEGMENT LIMIT
351 DW SYS_IDT_LO ; SEGMENT BASE ADDRESS - LOW WORD
352 DB 0 ; SEGMENT BASE ADDRESS - HIGH BYTE
353 DB CPLD_DATA_ACCESS ; ACCESS RIGHTS BYTE
354 DW 0 ; RESERVED - MUST BE ZERO
355
356 ;----- THE SYSTEM DATA AREA DESCRIPTOR - (RSDA_PTR)
357
358 DW SDA_LEN ; SEGMENT LIMIT
359 DW SDA_LO ; SEGMENT BASE ADDRESS - LOW WORD
360 DB 0 ; SEGMENT BASE ADDRESS - HIGH BYTE
361 DB CPLD_DATA_ACCESS ; ACCESS RIGHTS BYTE
362 DW 0 ; RESERVED - MUST BE ZERO
363
364 ;----- COMPATIBLE MONOCHROME DISPLAY REGEN BUFFER - (C_BWCRT_PTR)
365
366 DW MCRT_SIZE ; SEGMENT LIMIT
367 DW MCRT_LO ; SEGMENT BASE ADDRESS - LOW WORD
368 DB MCRT_HI ; SEGMENT BASE ADDRESS - HIGH BYTE
369 DB CPLD_DATA_ACCESS ; ACCESS RIGHTS BYTE
370 DW 0 ; RESERVED - MUST BE ZERO
371
372 ;----- COMPATIBLE COLOR DISPLAY REGEN BUFFER - (C_CCRT_PTR)
373
374 DW CCRT_SIZE ; SEGMENT LIMIT
375 DW CCRT_LO ; SEGMENT BASE ADDRESS - LOW WORD
376 DB CCRT_HI ; SEGMENT BASE ADDRESS - HIGH BYTE
377 DB CPLD_DATA_ACCESS ; ACCESS RIGHTS BYTE
378 DW 0 ; RESERVED - MUST BE ZERO
379
380 ;----- ENHANCED GRAPHIC ADAPTER REGEN BUFFER - (E_CCRT_PTR)
381
382 DW ECRT_SIZE ; SEGMENT LIMIT
383 DW ECRTS_LO_LO ; SEGMENT BASE ADDRESS - LOW WORD
384 DB ECRTS_LO_HI ; SEGMENT BASE ADDRESS - HIGH BYTE
385 DB CPLD_DATA_ACCESS ; ACCESS RIGHTS BYTE
386 DW 0 ; RESERVED - MUST BE ZERO
387
388 ;----- SECOND PART OF EGA - (E_CCRT_PTR2)
389
390 DW ECRT_SIZE ; SEGMENT LIMIT
391 DW ECRTS_HI_LO ; SEGMENT BASE ADDRESS - LOW WORD
392 DB ECRTS_HI_HI ; SEGMENT BASE ADDRESS - HIGH BYTE
393 DB CPLD_DATA_ACCESS ; ACCESS RIGHTS BYTE
394 DW 0 ; RESERVED - MUST BE ZERO
395
396 ;----- CODE SEGMENT FOR POST CODE, SYSTEM IDT - (SYS_ROM_CS)
397
398 DW MAX_SEG_LEN ; SEGMENT LIMIT
399 DW CSEG_LO ; SEGMENT BASE ADDRESS - LOW WORD
400 DB CSEG_HI ; SEGMENT BASE ADDRESS - HIGH BYTE
401 DB CPLD_CODE_ACCESS ; ACCESS RIGHTS BYTE
402 DW 0 ; RESERVED - MUST BE ZERO
403
404 ;----- TEMPORARY DESCRIPTOR FOR ES - (ES_TEMP)
405
406 DW MAX_SEG_LEN ; SEGMENT LIMIT
407 DW NSEG_LO ; SEGMENT BASE ADDRESS - LOW WORD
408 DB NSEG_HI ; SEGMENT BASE ADDRESS - HIGH BYTE
409 DB CPLD_DATA_ACCESS ; ACCESS RIGHTS BYTE
410 DW 0 ; RESERVED - MUST BE ZERO
411
412 ;----- TEMPORARY DESCRIPTOR FOR CS AS A DATA SEGMENT - (CS_TEMP)
413
414 DW MAX_SEG_LEN ; SEGMENT LIMIT
415 DW NSEG_LO ; SEGMENT BASE ADDRESS - LOW WORD
416 DB NSEG_HI ; SEGMENT BASE ADDRESS - HIGH BYTE
417 DB CPLD_DATA_ACCESS ; ACCESS RIGHTS BYTE
418 DW 0 ; RESERVED - MUST BE ZERO
419
420 ;----- TEMPORARY DESCRIPTOR FOR SS - (SS_TEMP)
421
422 DW MAX_SEG_LEN ; SEGMENT LIMIT
423 DW NSEG_LO ; SEGMENT BASE ADDRESS - LOW WORD
424 DB NSEG_HI ; SEGMENT BASE ADDRESS - HIGH BYTE
425 DB CPLD_DATA_ACCESS ; ACCESS RIGHTS BYTE
426 DW 0 ; RESERVED - MUST BE ZERO
427
428 ;----- TEMPORARY DESCRIPTOR FOR DS - (DS_TEMP)
429
430 DW MAX_SEG_LEN ; SEGMENT LIMIT
431 DW NSEG_LO ; SEGMENT BASE ADDRESS - LOW WORD
432 DB NSEG_HI ; SEGMENT BASE ADDRESS - HIGH BYTE
433 DB CPLD_DATA_ACCESS ; ACCESS RIGHTS BYTE
434 DW 0 ; RESERVED - MUST BE ZERO

```

```

435 PAGE
436 |----- (POST_TR)
437 0217 TR_LOC:
438 0217 0800 DW 00800H ; SEGMENT LIMIT
439 0219 C000 DW 0C000H ; SEGMENT BASE ADDRESS - LOW WORD
440 021B 00 DB 0 ; SEGMENT BASE ADDRESS - HIGH BYTE
441 021C 81 DB FREE_TSS ; ACCESS RIGHTS BYTE
442 021D 0000 DW 0 ; RESERVED - MUST BE ZERO
443
444 |----- (POST_TSS_PTR)
445
446 021F 0800 DW 00800H ; SEGMENT LIMIT
447 0221 0217 R DW TR_LOC ; SEGMENT BASE ADDRESS - LOW WORD
448 0223 00 DB 0 ; SEGMENT BASE ADDRESS - HIGH BYTE
449 0224 93 DB CPLD_DATA_ACCESS ; ACCESS RIGHTS BYTE
450 0225 0000 DW 0 ; RESERVED - MUST BE ZERO
451
452 |----- (POST_LDTR)
453 0227 LDT_LOC:
454 0227 0888 DW GDT_LEN ; SEGMENT LIMIT
455 0229 0000 DW 0D000H ; SEGMENT BASE ADDRESS - LOW WORD
456 022B 00 DB 0 ; SEGMENT BASE ADDRESS - HIGH BYTE
457 022C E2 DB LDT_DESC ; ACCESS RIGHTS BYTE
458 022D 0000 DW 0 ; RESERVED - MUST BE ZERO
459
460 |----- (POST_LDT_PTR)
461
462 022F 0888 DW GDT_LEN ; SEGMENT LIMIT
463 0231 0217 R DW LDT_LOC ; SEGMENT BASE ADDRESS - LOW WORD
464 0233 00 DB 0 ; SEGMENT BASE ADDRESS - HIGH BYTE
465 0234 93 DB CPLD_DATA_ACCESS ; ACCESS RIGHTS BYTE
466 0235 0000 DW 0 ; RESERVED - MUST BE ZERO
467
468 = 0237 GDT_DATA_END EQU $
469
470 |----- END OF PRE-ALLOCATED GDT
471
472 |----- ENTRY POINTS FOR THE FIRST 32 SYSTEM INTERRUPTS
473
474
475 0237 SYS_IDT_OFFSETS LABEL WORD
476
477 0237 0000 R DW OFFSET EXC_00 ; INTERRUPTS AS DEFINED
478 0239 0005 R DW OFFSET EXC_01 ; EXCPT 00 - DIVIDE ERROR
479 023B 000A R DW OFFSET EXC_02 ; EXCPT 01 - SINGLE STEP
480 023D 000F R DW OFFSET EXC_03 ; EXCPT 02 - NMI, SYSTEM REQUEST FOR 01
481 023F 0014 R DW OFFSET EXC_04 ; EXCPT 03 - BREAKPOINT
482 0241 0019 R DW OFFSET EXC_05 ; EXCPT 04 - INTO DETECT
483 0243 0030 R DW OFFSET EXC_06 ; EXCPT 05 - BOUND
484 0245 0034 R DW OFFSET EXC_07 ; EXCPT 06 - INVALID OPCODE
485 0247 0038 R DW OFFSET EXC_08 ; EXCPT 07 - PROCESSOR EXT NOT AVAIL
486 0249 003C R DW OFFSET EXC_09 ; EXCPT 08 - DOUBLE EXCEPTION
487 024B 0040 R DW OFFSET EXC_10 ; EXCPT 09 - PROCESSOR EXT SEGMENT ERR
488 024D 0044 R DW OFFSET EXC_11 ; EXCPT 10 - TSS BAD IN GATE TRANSFER
489 024F 0048 R DW OFFSET EXC_12 ; EXCPT 11 - SEGMENT NOT PRESENT
490 0251 004C R DW OFFSET EXC_13 ; EXCPT 12 - STACK SEGMENT NOT PRESENT
491 0253 0050 R DW OFFSET EXC_14 ; EXCPT 13 - GENERAL PROTECTION
492 0255 0054 R DW OFFSET EXC_15
493 0257 0058 R DW OFFSET EXC_16 ; EXCPT 16 - PROCESSOR EXTENSION ERROR
494 0259 005C R DW OFFSET EXC_17
495 025B 0060 R DW OFFSET EXC_18
496 025D 0064 R DW OFFSET EXC_19
497 025F 0068 R DW OFFSET EXC_20
498 0261 006C R DW OFFSET EXC_21
499 0263 0070 R DW OFFSET EXC_22
500 0265 0074 R DW OFFSET EXC_23
501 0267 0078 R DW OFFSET EXC_24
502 0269 007C R DW OFFSET EXC_25
503 026B 0080 R DW OFFSET EXC_26
504 026D 0084 R DW OFFSET EXC_27
505 026F 0088 R DW OFFSET EXC_28
506 0271 008C R DW OFFSET EXC_29
507 0273 0090 R DW OFFSET EXC_30
508 0275 0094 R DW OFFSET EXC_31
509
510 |----- FORMAT INTERRUPT DESCRIPTORS (GATES) 32 - 255
511
512 0277 01AE R FREE_INTS DW OFFSET IRET_ADDR ; DESTINATION OFFSET
513 0279 0040 R DW SYS_ROM_CS ; DESTINATION SEGMENT
514 027B 00 86 DB 0,INT_GATE ; UNUSED AND ACCESS RIGHTS BYTE
515 027D
516
517 027D CODE ENDS
518 END

```

SECTION 5

```

1      PAGE 118,121
2      TITLE TEST6 ---- 06/10/85 POST TESTS AND SYSTEM BOOT STRAP
3      .286C
4      .LIST
5      0000      CODE      SEGMENT BYTE PUBLIC
6
7      PUBLIC   BOOT_STRAP_1
8      PUBLIC   POST6
9      PUBLIC   STGTST_CNT
10     PUBLIC   ROM_ERR
11     PUBLIC   XMIT_8042
12
13     EXTRN   CMOS_READ:NEAR
14     EXTRN   DDS:NEAR
15     EXTRN   DISK_BASE:NEAR
16     EXTRN   EGD2:NEAR
17     EXTRN   ERR_BEEP:NEAR
18     EXTRN   E_MSG:NEAR
19     EXTRN   FSA:NEAR
20     EXTRN   PRT_SEG:NEAR
21
22     ASSUME   CS:CODE,DS:DATA
23
24     0000      PROC      NEAR
25
26     ;-----
27     ; THIS SUBROUTINE PERFORMS A READ/WRITE STORAGE TEST ON A BLOCK ;
28     ; OF STORAGE. ;
29     ; ENTRY REQUIREMENTS: ;
30     ; ES = ADDRESS OF STORAGE SEGMENT BEING TESTED ;
31     ; DS = ADDRESS OF STORAGE SEGMENT BEING TESTED ;
32     ; CX = WORD COUNT OF STORAGE BLOCK TO BE TESTED ;
33     ; EXIT PARAMETERS: ;
34     ; ZERO FLAG = 0 IF STORAGE ERROR (DATA COMPARE OR PARITY ;
35     ; CHECK), AL=0 DENOTES A PARITY CHECK, ELSE AL=XOR'ED ;
36     ; BIT PATTERN OF THE EXPECTED DATA PATTERN VS THE ACTUAL ;
37     ; DATA READ. ;
38     ; AX,BX,CX,DX,DI, AND SI ARE ALL DESTROYED. ;
39     ;-----
40     0000      MOV      PROC      NEAR
41     0000 8B D9      MOV      BX,CX ; SAVE WORD COUNT OF BLOCK TO TEST
42     0002 E4 61      IN       AL,PORT_B
43     0004 0C 0C      OR      AL,RAM_PAR_OFF ; TOGGLE PARITY CHECK LATCHES
44     0006 E6 61      OUT      PORT_B,AL ; TO RESET ANY PENDING ERROR
45     0008 24 F3      AND     AL,RAM_PAR_ON
46     000A E6 61      OUT      PORT_B,AL
47
48     ;-----
49     ; ROLL A BIT THROUGH THE FIRST WORD
50     000C 33 D2      XOR     DX,DX ; CLEAR THE INITIAL DATA PATTERN
51     000E 89 0010    MOV     CX,16 ; ROLL 16 BIT POSITIONS
52     0011 2B FF      SUB     DI,DI ; START AT BEGINNING OF BLOCK
53     0013 2B F6      SUB     SI,SI ; INITIALIZE DESTINATION POINTER
54     0015 F9        STC ; SET CARRY FLAG ON FOR FIRST BIT
55     0016 D1 D2      RCL     DX,1 ; MOVE BIT OVER LEFT TO NEXT POSITION
56     0018 89 15      MOV     [DI],DX ; STORE DATA PATTERN
57     001A 8B 05      MOV     AX,[DI] ; GET THE DATA WRITTEN
58     001C 33 C2      XOR     AX,DX ; INSURE DATA AS EXPECTED (CLEAR CARRY)
59     001E E1 F6      LOOPZ  C1 ; LOOP TILL DONE OR ERROR
60
61     0020 75 66      JNZ    C13 ; EXIT IF ERROR
62
63     ;-----
64     ; CHECK CAS LINES FOR HIGH BYTE LOW BYTE
65     0022 BA FF00    MOV     DX,0FF00H ; TEST DATA - AX= 0000H
66     0025 89 05      MOV     [DI],AX ; STORE DATA PATTERN = 0000H
67     0027 8B 75 01    MOV     [DI-1],DH ; WRITE A BYTE OF FFH AT ODD LOCATION
68     002A 8B 05      MOV     AX,[DI] ; GET THE DATA - SHOULD BE 0FF00H
69     002C 33 C2      XOR     AX,DX ; CHECK THE FIRST WRITTEN
70     002E 75 58      JNZ    C13 ; ERROR EXIT IF NOT ZERO
71
72     0030 89 05      MOV     [DI],AX ; STORE DATA PATTERN OF 0000H
73     0032 8B 35      MOV     [DI],DH ; WRITE A BYTE OF FFH AT EVEN LOCATION
74     0034 8B F2      XCHG   DH,DL ; SET DX= 000FFH AND BUS SETTLE
75     0036 8B 05      MOV     AX,[DI] ; GET THE DATA
76     0038 33 C2      XOR     AX,DX ; CHECK THE FIRST WRITTEN
77     003A 75 4C      JNZ    C13 ; EXIT IF NOT
78
79     ;-----
80     ; CHECK FOR I/O OR BASE MEMORY ERROR
81     003C E4 61      IN     AL,PORT_B ; CHECK FOR I/O - PARITY CHECK
82     003E 86 C4      XCHG   AL,AH ; SAVE ERROR
83     0040 E4 87      IN     AL,DMA_PAGE*6 ; CHECK FOR R/W OR I/O ERROR
84     0042 22 E0      AND    AH,AL ; MASK FOR ERROR EXPECTED
85
86     ;-----
87     ; PARITY ERROR EXIT
88     0044 8B 0000    MOV     AX,0 ; RESTORE AX TO 0000
89     0047 75 3F      JNZ    C13 ; EXIT IF PARITY ERROR
90
91     0049 BA AA55    MOV     DX,0AA55H ; WRITE THE INITIAL DATA PATTERN
92     004C          C3:
93     004C 2B FF      SUB     DI,DI ; START AT BEGINNING OF BLOCK
94     004E 2B F6      SUB     SI,SI ; INITIALIZE DESTINATION POINTER
95     0050 8B CB      MOV     CX,BX ; SETUP BYTE COUNT FOR LOOP
96     0052 8B C2      MOV     AX,DX ; GET THE PATTERN
97     0054 F3 AB      REP     STOSW ; STORE 64K BYTES (32K WORDS)
98     0056 8B CB      MOV     CX,BX ; SET COUNT
99     0058 2B F6      SUB     SI,SI ; START AT BEGINNING
100    005A          C6:
101    005A AD      LODSW  ; GET THE FIRST WRITTEN
102    005B 33 C2      XOR     AX,DX ; INSURE DATA AS EXPECTED
103    005D E1 FB      LOOPZ  C6 ; LOOP TILL DONE OR ERROR
104
105    005F 75 27      JNZ    C13 ; EXIT IF NOT EXPECTED (ERROR BITS ON)
106
107     ;-----
108     ; CHECK FOR I/O OR BASE MEMORY ERROR
109    0061 E4 61      IN     AL,PORT_B ; CHECK FOR I/O -PARITY CHECK
110    0063 86 C4      XCHG   AL,AH ; SAVE ERROR
111    0065 E4 87      IN     AL,DMA_PAGE*6 ; CHECK FOR R/W OR I/O ERROR
112    0067 22 E0      AND    AH,AL
113
114

```









```

1 PAGE 118,121
2 TITLE DISKETTE -- 04/21/86 DISKETTE BIOS
3 .286C
4 .LIST
5 SUBTTL (DSK1.ASM)
6 .LIST
7
8 INT 13H
9 DISKETTE I/O
10 THIS INTERFACE PROVIDES DISK ACCESS TO THE 5.25 INCH 360 KB,
11 1.2 MB, 720 KB, AND 1.44 MB DISKETTE DRIVES.
12
13 INPUT
14 (AH)=00H RESET DISKETTE SYSTEM
15 HARD RESET TO NEC, PREPARE COMMAND, RECALIBRATE REQUIRED
16 ON ALL DRIVES
17
18 (AH)=01H READ THE STATUS OF THE SYSTEM INTO (AH)
19 *DISKETTE STATUS FROM LAST OPERATION IS USED
20
21 REGISTERS FOR READ/WRITE/VERIFY/FORMAT
22 (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)
23 (DH) - HEAD NUMBER (0-1 ALLOWED, NOT VALUE CHECKED)
24 (CH) - TRACK NUMBER (NOT VALUE CHECKED)
25
26 MEDIA DRIVE TRACK NUMBER
27 320/360 320/360 0-39
28 320/360 1.2M 0-39
29 1.2M 1.2M 0-79
30 720K 720K 0-79
31 1.44M 1.44M 0-79
32
33 (CL) - SECTOR NUMBER (NOT VALUE CHECKED, NOT USED FOR FORMAT)
34 MEDIA DRIVE SECTOR NUMBER
35 320/360 320/360 1-8/9
36 320/360 1.2M 1-8/9
37 1.2M 1.2M 1-15
38 720K 720K 1-9
39 1.44M 1.44M 1-15
40
41 (AL) - NUMBER OF SECTORS (NOT VALUE CHECKED, NOT USED FOR FORMAT)
42 MEDIA DRIVE MAX NUMBER OF SECTORS
43 320/360 320/360 8/9
44 320/360 1.2M 8/9
45 1.2M 1.2M 15
46 720K 720K 9
47 1.44M 1.44M 18
48
49 (ES:BX) - ADDRESS OF BUFFER (NOT REQUIRED FOR VERIFY)
50
51 (AH)=02H READ THE DESIRED SECTORS INTO MEMORY
52
53 (AH)=03H WRITE THE DESIRED SECTORS FROM MEMORY
54
55 (AH)=04H VERIFY THE DESIRED SECTORS
56
57 (AH)=05H FORMAT THE DESIRED TRACK
58 (ES:BX) MUST POINT TO THE COLLECTION OF DESIRED ADDRESS FIELDS
59 FOR THE TRACK. EACH FIELD IS COMPOSED OF 4 BYTES, (C,H,R,N),
60 WHERE C = TRACK NUMBER, H=HEAD NUMBER, R = SECTOR NUMBER,
61 N= NUMBER OF BYTES PER SECTOR (00=128,01=256,02=512,03=1024).
62 THERE MUST BE ONE ENTRY FOR EVERY SECTOR ON THE TRACK.
63 THIS INFORMATION IS USED TO FIND THE REQUESTED SECTOR DURING
64 READ/WRITE ACCESS.
65
66 PRIOR TO FORMATTING A DISKETTE, IF THERE EXISTS MORE THAN
67 ONE SUPPORTED MEDIA FORMAT TYPE WITHIN THE DRIVE IN QUESTION,
68 THEN "SET DASH TYPE" (INT 13H, AH = 17H) OR "SET MEDIA TYPE"
69 (INT 13H, AH = 18H) MUST BE CALLED TO SET THE DISKETTE TYPE
70 THAT IS TO BE FORMATTED. IF "SET DASH TYPE" OR "SET MEDIA TYPE"
71 IS NOT CALLED, THE FORMAT ROUTINE WILL ASSUME THE MEDIA FORMAT
72 TO BE THE MAXIMUM CAPACITY OF THE DRIVE.
73
74 THESE PARAMETERS OF DISK_BASE MUST BE CHANGED IN ORDER TO
75 FORMAT THE FOLLOWING MEDIA:
76
77 MEDIA DRIVE PARM 1 PARM 2
78 320K 320K/360K/1.2M 50H 8
79 360K 320K/360K/1.2M 50H 9
80 1.2M 1.2M 54H 15
81 720K 720K/1.44M 50H 9
82 1.44M 1.44M 6CH 18
83
84 NOTES: - PARM 1 = GAP LENGTH FOR FORMAT
85 - PARM 2 = EOT (LAST SECTOR ON TRACK)
86 - DISK BASE IS POINTED TO BY DISK POINTER LOCATED
87 AT ABSOLUTE ADDRESS 0178.
88 - WHEN FORMAT OPERATIONS ARE COMPLETE, THE PARAMETERS
89 SHOULD BE RESTORED TO THEIR RESPECTIVE INITIAL VALUES
90
91 (AH)=06H READ DRIVE PARAMETERS
92 REGISTERS
93 INPUT
94 (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)
95 (ES:DI) POINTS TO DRIVE PARAMETERS TABLE
96 (CH) - LOW ORDER 8 OF 10 BITS MAXIMUM NUMBER OF TRACKS
97 (CL) - BITS 7 & 6 - HIGH ORDER TWO BITS OF MAXIMUM TRACKS
98 - BITS 5 THRU 0 - MAXIMUM SECTORS PER TRACK
99 (DH) - MAXIMUM HEAD NUMBER
100 (DL) - NUMBER OF DISKETTE DRIVES INSTALLED
101 (BH) - 0
102 (BL) - BITS 7 THRU 4 - 0
103 - BITS 3 THRU 0 - VALID DRIVE TYPE VALUE IN CMOS
104 (AX) - 0
105 UNDER THE FOLLOWING CIRCUMSTANCES:
106 (1) THE DRIVE NUMBER IS INVALID.
107 (2) THE DRIVE TYPE IS UNKNOWN AND CMOS IS NOT PRESENT,
108 (3) THE DRIVE TYPE IS UNKNOWN AND CMOS IS BAD,
109 (4) OR THE DRIVE TYPE IS UNKNOWN AND THE CMOS DRIVE TYPE IS INVALID
110 THEN ES:AX,BX,CX,DX,DI=0 ; DL=NUMBER OF DRIVES.
111 IF NO DRIVES ARE PRESENT THEN: ES,AX,BX,CX,DX,DI=0.
112 *DISKETTE STATUS = 0 AND CY IS RESET.
113
114 (AH)=18H READ DASH TYPE
115 OUTPUT REGISTERS
    
```

```

115 ; (AH) - ON RETURN IF CARRY FLAG NOT SET, OTHERWISE ERROR
116 ; 00 - DRIVE NOT PRESENT
117 ; 01 - DISKETTE, NO CHANGE LINE AVAILABLE
118 ; 02 - DISKETTE, CHANGE LINE AVAILABLE
119 ; 03 - RESERVED
120 ; (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)
121 ;
122 ;-----
123 ; (AH)=16H DISK CHANGE LINE STATUS
124 ; OUTPUT REGISTERS
125 ; (AH) - 00 - DISK CHANGE LINE NOT ACTIVE
126 ; 06 - DISK CHANGE LINE ACTIVE & CARRY BIT ON
127 ; (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)
128 ;
129 ;-----
130 ; (AH)=17H SET DASH TYPE FOR FORMAT
131 ; INPUT REGISTERS
132 ; (AL) - 00 - NOT USED
133 ; 01 - DISKETTE 320/360K IN 360K DRIVE
134 ; 02 - DISKETTE 360K IN 1.2M DRIVE
135 ; 03 - DISKETTE 1.2M IN 1.2M DRIVE
136 ; 04 - DISKETTE 720K IN 720K DRIVE
137 ; (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)
138 ; DO NOT USE WHEN DISKETTE ATTACH CARD USED)
139 ;
140 ;-----
141 ; (AH)=18H SET MEDIA TYPE FOR FORMAT
142 ; INPUT REGISTERS
143 ; (CH) - LOW ORDER 8 OF 10 BITS MAXIMUM NUMBER OF TRACKS
144 ; (CL) - BITS 7 2 - HIGH ORDER TWO BITS OF MAXIMUM TRACKS
145 ; - BITS 5 THRU 0 - MAXIMUM SECTORS PER TRACK
146 ; (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)
147 ; OUTPUT REGISTERS
148 ; (ES:DI) - POINTER TO DRIVE PARAMETERS TABLE FOR THIS MEDIA TYPE,
149 ; UNCHANGED IF (AH) IS NON-ZERO
150 ; (AH) - 00H, CY = 0, TRACK AND SECTORS/TRACK COMBINATION IS SUPPORTED
151 ; 01H, CY = 1, FUNCTION IS NOT AVAILABLE
152 ; 02H, CY = 1, TRACK AND SECTORS/TRACK COMBINATION IS NOT SUPPORTED
153 ; OR DRIVE TYPE UNKNOWN
154 ; 80H, CY = 1, TIME OUT (DISKETTE NOT PRESENT)
155 ;
156 ; DISK CHANGE STATUS IS ONLY CHECKED WHEN A MEDIA SPECIFIED IS OTHER
157 ; THAN 360 KB DRIVE, IF THE DISK CHANGE LINE IS FOUND TO BE
158 ; ACTIVE THE FOLLOWING ACTIONS TAKE PLACE:
159 ; ATTEMPT TO RESET DISK CHANGE LINE TO INACTIVE STATE.
160 ; IF ATTEMPT SUCCEEDS SET DASH TYPE FOR FORMAT AND RETURN DISK
161 ; CHANGE ERROR CODE
162 ; IF ATTEMPT FAILS RETURN TIMEOUT ERROR CODE AND SET DASH TYPE
163 ; TO A PREDETERMINED STATE INDICATING MEDIA TYPE UNKNOWN.
164 ; IF THE DISK CHANGE LINE IN INACTIVE PERFORM SET DASH TYPE FOR FORMAT.
165 ;
166 ; DATA VARIABLE -- #DISK POINTER
167 ; DOUBLE WORD POINTER TO THE CURRENT SET OF DISKETTE PARAMETERS
168 ;-----
169 ; OUTPUT FOR ALL FUNCTIONS
170 ; AH = STATUS OF OPERATION
171 ; STATUS BITS ARE DEFINED IN THE EQUATES FOR #DISKETTE_STATUS
172 ; VARIABLE IN THE DATA SEGMENT OF THIS MODULE
173 ; CY = 0 SUCCESSFUL OPERATION (AH=0 ON RETURN, EXCEPT FOR READ DASH
174 ; TYPE AH=[16]).
175 ; CY = 1 FAILED OPERATION (AH HAS ERROR REASON)
176 ; FOR READ/WRITE/VERIFY
177 ; AL = COUNT OF SECTORS TRANSFERRED
178 ; DS,BX,DX,CX PRESERVED
179 ; NOTE: IF AN ERROR IS REPORTED BY THE DISKETTE CODE, THE APPROPRIATE
180 ; ACTION IS TO RESET THE DISKETTE, THEN RETRY THE OPERATION.
181 ; ON READ ACCESSES, NO MOTOR START DELAY IS TAKEN, SO THAT
182 ; THREE RETRIES ARE REQUIRED ON READS TO ENSURE THAT THE
183 ; PROBLEM IS NOT DUE TO MOTOR START-UP.
184 ;-----
185 ;.LIST
186 ; DISKETTE STATE MACHINE - ABSOLUTE ADDRESS 40:90 (DRIVE A) & 91 (DRIVE B)
187 ;.LIST
188 ;
189 ; 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
190 ;-----
191 ;
192 ;
193 ;
194 ;
195 ; RESERVED
196 ;
197 ; PRESENT STATE
198 ; 000: 360K IN 360K DRIVE UNESTABLISHED
199 ; 001: 360K IN 1.2M DRIVE UNESTABLISHED
200 ; 010: 1.2M IN 1.2M DRIVE UNESTABLISHED
201 ; 011: 360K IN 360K DRIVE ESTABLISHED
202 ; 100: 360K IN 1.2M DRIVE ESTABLISHED
203 ; 101: 1.2M IN 1.2M DRIVE ESTABLISHED
204 ; 110: RESERVED
205 ; 111: NONE OF THE ABOVE
206 ;-----
207 ; MEDIA/DRIVE ESTABLISHED
208 ;-----
209 ; DOUBLE STEPPING REQUIRED (360K IN 1.2M
210 ; DRIVE)
211 ;-----
212 ; DATA TRANSFER RATE FOR THIS DRIVE:
213 ;
214 ; 00: 800 KBS
215 ; 01: 300 KBS
216 ; 10: 280 KBS
217 ; 11: RESERVED
218 ;
219 ;.LIST
220 ;-----
221 ; STATE OPERATION STARTED - ABSOLUTE ADDRESS 40:92 (DRIVE A) & 93 (DRIVE B)
222 ;
223 ; PRESENT CYLINDER NUMBER - ABSOLUTE ADDRESS 40:94 (DRIVE A) & 95 (DRIVE B)
224 ;
225 ; SUBTTL (DSK2.ASM)
    
```

SECTION 5

```

224 PAGE
225
226 MD_STRUC          STRUC
227 MD_SPEC1        DB  ?      ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
228 MD_SPEC2        DB  ?      ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
229 MD_OFF_TIM      DB  ?      ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
230 MD_BYT_SEC      DB  ?      ; 512 BYTES/SECTOR
231 MD_SEC_TRK      DB  ?      ; EOT ( LAST SECTOR ON TRACK)
232 MD_GAP          DB  ?      ; GAP LENGTH
233 MD_DTL          DB  ?      ; DTL
234 MD_GAP3         DB  ?      ; GAP LENGTH FOR FORMAT
235 MD_FIL_BYT     DB  ?      ; FILL BYTE FOR FORMAT
236 MD_HD_TIM       DB  ?      ; HEAD SETTLE TIME (MILLISECONDS)
237 MD_STR_TIM      DB  ?      ; MOTOR START TIME (1/8 SECONDS)
238 MD_MAX_TRK      DB  ?      ; MAX. TRACK NUMBER
239 MD_RATE         DB  ?      ; DATA TRANSFER RATE
240 MD_STRUC        ENDS
241
242 = 007F          BITOFF      EQU  7FH
243 = 0080          BITON       EQU  80H
244
245 PUBLIC DISK_INT_1
246 PUBLIC SEEK
247 PUBLIC DISKETTE_SETUP
248 PUBLIC DISKETTE_IO_1
249
250 EXTRN CMOS_READ:NEAR
251 EXTRN DOS:NEAR
252 EXTRN DISK_BASE:NEAR
253 EXTRN WAITF:NEAR
254
255
256 CODE SEGMENT BYTE PUBLIC
257
258 ASSUME CS:CODE,DS:DATA,ES:DATA
259
260 ;-----
261 ; DRIVE TYPE TABLE
262 ;-----
263 DR_TYPE          DB  01      ; DRIVE TYPE, MEDIA TABLE
264 MD_001_0012     DW  OFFSET MD_TBL1
265 MD_003_82       DW  02+BITTON
266 MD_004_001F     DW  OFFSET MD_TBL2
267 MD_006_02       DB  02
268 MD_007_002C     DW  OFFSET MD_TBL3
269 MD_009_03       DB  03
270 MD_00A_0039     DW  OFFSET MD_TBL4
271 MD_00C_84       DB  04+BITTON
272 MD_00D_0046     DW  OFFSET MD_TBL5
273 MD_00F_04       DB  04
274 MD_010_0053     DW  OFFSET MD_TBL6
275 MD_012         = 8        ; END OF TABLE
276 MD_006         EQU  (DR_TYPE_E-DR_TYPE)/3
277
278 ;-----
279 ; MEDIA/DRIVE PARAMETER TABLES
280 ;-----
281
282 ;-----
283 ; 360 KB MEDIA IN 360 KB DRIVE
284 ;-----
285
286 MD_TBL1          LABEL BYTE
287 MD_TBL1         DB  1101111B ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
288 MD_TBL1         DB  2        ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
289 MD_TBL1         DB  MOTOR_WAIT ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
290 MD_TBL1         DB  2        ; 512 BYTES/SECTOR
291 MD_TBL1         DB  9        ; EOT ( LAST SECTOR ON TRACK)
292 MD_TBL1         DB  02AH     ; GAP LENGTH
293 MD_TBL1         DB  0F6H     ; DTL
294 MD_TBL1         DB  050H     ; GAP LENGTH FOR FORMAT
295 MD_TBL1         DB  0F6H     ; FILL BYTE FOR FORMAT
296 MD_TBL1         DB  15       ; HEAD SETTLE TIME (MILLISECONDS)
297 MD_TBL1         DB  8        ; MOTOR START TIME (1/8 SECONDS)
298 MD_TBL1         DB  39       ; MAX. TRACK NUMBER
299 MD_TBL1         DB  RATE_250 ; DATA TRANSFER RATE
300
301 ;-----
302 ; 360 KB MEDIA IN 1.2 MB DRIVE
303 ;-----
304
305 MD_TBL2          LABEL BYTE
306 MD_TBL2         DB  1101111B ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
307 MD_TBL2         DB  2        ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
308 MD_TBL2         DB  MOTOR_WAIT ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
309 MD_TBL2         DB  2        ; 512 BYTES/SECTOR
310 MD_TBL2         DB  9        ; EOT ( LAST SECTOR ON TRACK)
311 MD_TBL2         DB  02AH     ; GAP LENGTH
312 MD_TBL2         DB  0F6H     ; DTL
313 MD_TBL2         DB  050H     ; GAP LENGTH FOR FORMAT
314 MD_TBL2         DB  0F6H     ; FILL BYTE FOR FORMAT
315 MD_TBL2         DB  15       ; HEAD SETTLE TIME (MILLISECONDS)
316 MD_TBL2         DB  8        ; MOTOR START TIME (1/8 SECONDS)
317 MD_TBL2         DB  39       ; MAX. TRACK NUMBER
318 MD_TBL2         DB  RATE_300 ; DATA TRANSFER RATE
319
320 ;-----
321 ; 1.2 MB MEDIA IN 1.2 MB DRIVE
322 ;-----
323
324 MD_TBL3          LABEL BYTE
325 MD_TBL3         DB  1101111B ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
326 MD_TBL3         DB  2        ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
327 MD_TBL3         DB  MOTOR_WAIT ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
328 MD_TBL3         DB  2        ; 512 BYTES/SECTOR
329 MD_TBL3         DB  15       ; EOT ( LAST SECTOR ON TRACK)
330 MD_TBL3         DB  018H     ; GAP LENGTH
331 MD_TBL3         DB  0F6H     ; DTL
332 MD_TBL3         DB  054H     ; GAP LENGTH FOR FORMAT
333 MD_TBL3         DB  0F6H     ; FILL BYTE FOR FORMAT
334 MD_TBL3         DB  15       ; HEAD SETTLE TIME (MILLISECONDS)
335 MD_TBL3         DB  8        ; MOTOR START TIME (1/8 SECONDS)
336 MD_TBL3         DB  79       ; MAX. TRACK NUMBER
337 MD_TBL3         DB  RATE_500 ; DATA TRANSFER RATE

```

```

338
339
340
341
342
343 0039 DF
344 003A 02
345 003B 26
346 003C 02
347 003D 09
348 003E 2A
349 003F FF
350 0040 80
351 0041 F6
352 0042 0F
353 0043 08
354 0044 4F
355 0045 80
356
357
358
359
360
361
362 0046 DF
363 0047 02
364 0048 26
365 0049 02
366 004A 09
367 004B 2A
368 004C FF
369 004D 80
370 004E F6
371 004F 0F
372 0050 08
373 0051 4F
374 0052 80
375
376
377
378
379
380
381 0053
382 0053 AF
383 0054 02
384 0055 26
385 0056 02
386 0057 12
387 0058 1B
388 0059 FF
389 005A 6C
390 005B F6
391 005C 0F
392 005D 08
393 005E 4F
394 005F 00
395
396
397 0060
398
399 0060 FB
400 0061 53
401 0062 57
402 0063 52
403 0064 53
404 0065 51
405 0066 8B EC
406
407
408
409
410
411
412
413
414
415
416
417
418
419 0068 1E
420 0069 56
421 006A E8 0000 E
422 006D 80 FC 19
423 0070 72 02
424 0072 B4 14
425
426 0074
427 0074 80 FC 01
428 0077 76 0C
429 0079 80 FC 08
430 007C 74 07
431 007E 80 FA 01
432 0081 76 02
433 0083 B4 14
434
435 0085
436 0085 8A CC
437 0087 32 ED
438 0089 D0 E1
439 008B BB 00B5 R
440 008E 03 D9
441 0090 8A E6
442 0092 32 F6
443 0094 8B F0
444 0096 8B FA
445 0098 8A 26 0041 R
446 009C C6 06 0041 R 00
447
448
449
450
451

```

```

-----
; 720 KB MEDIA IN 720 KB DRIVE
-----
MD_TBL4 LABEL BYTE
DB 11011111B ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
DB 2 ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
DB MOTOR_WAIT ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
DB 2 ; 512 BYTES/SECTOR
DB 09 ; EOT ( LAST SECTOR ON TRACK)
DB 02AH ; GAP LENGTH
DB 0FFH ; DTL
DB 050H ; GAP LENGTH FOR FORMAT
DB 0F6H ; FILL BYTE FOR FORMAT
DB 15 ; HEAD SETTLE TIME (MILLISECONDS)
DB 8 ; MOTOR START TIME (1/8 SECONDS)
DB 79 ; MAX. TRACK NUMBER
DB RATE_250 ; DATA TRANSFER RATE
-----
; 720 KB MEDIA IN 1.44 MB DRIVE
-----
MD_TBL5 LABEL BYTE
DB 11011111B ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
DB 2 ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
DB MOTOR_WAIT ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
DB 2 ; 512 BYTES/SECTOR
DB 09 ; EOT ( LAST SECTOR ON TRACK)
DB 02AH ; GAP LENGTH
DB 0FFH ; DTL
DB 050H ; GAP LENGTH FOR FORMAT
DB 0F6H ; FILL BYTE FOR FORMAT
DB 15 ; HEAD SETTLE TIME (MILLISECONDS)
DB 8 ; MOTOR START TIME (1/8 SECONDS)
DB 79 ; MAX. TRACK NUMBER
DB RATE_250 ; DATA TRANSFER RATE
-----
; 1.44 MB MEDIA IN 1.44 MB DRIVE
-----
MD_TBL6 LABEL BYTE
DB 10101111B ; SRT=A, HD UNLOAD=0F - 1ST SPECIFY BYTE
DB 2 ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
DB MOTOR_WAIT ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
DB 2 ; 512 BYTES/SECTOR
DB 18 ; EOT ( LAST SECTOR ON TRACK)
DB 01BH ; GAP LENGTH
DB 0FFH ; DTL
DB 050H ; GAP LENGTH FOR FORMAT
DB 0F6H ; FILL BYTE FOR FORMAT
DB 15 ; HEAD SETTLE TIME (MILLISECONDS)
DB 8 ; MOTOR START TIME (1/8 SECONDS)
DB 79 ; MAX. TRACK NUMBER
DB RATE_500 ; DATA TRANSFER RATE
-----
DISKETTE_IQ_1 PROC FAR ;>>> ENTRY POINT FOR ORG 0EC69H
STI ; INTERRUPTS BACK ON
PUSH BP ; USER REGISTER
PUSH DI ; USER REGISTER
PUSH DX ; HEAD #, DRIVE # OR USER REGISTER
PUSH BX ; BUFFER OFFSET PARAMETER OR REGISTER
PUSH CX ; TRACK #-SECTOR # OR USER REGISTER
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
; ; 512 BYTES/SECTOR
; ; 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
; ; BUFFER SEGMENT PARM OR USER REGISTER
; ; USER REGISTERS
; ; SEGMENT OF BIOS DATA AREA TO DS
; ; CHECK FOR > LARGEST FUNCTION
; ; FUNCTION OK
; ; REPLACE WITH KNOWN INVALID FUNCTION
OK_FUNC:
CMP AH,1 ; RESET OR STATUS ?
JBE OK_DRV ; IF RESET OR STATUS DRIVE ALWAYS OK
CMP AH,8 ; READ DRIVE PARAMS ?
JZ OK_DRV ; IF SO DRIVE CHECKED LATER
CMP DL,1 ; DRIVES 0 AND 1 OK
JBE OK_DRV ; IF 0 OR 1 THEN JUMP
MOV AH,14H ; REPLACE WITH KNOWN INVALID FUNCTION
OK_DRV:
MOV CL,AH ; CL = FUNCTION
XOR CH,CH ; CX = FUNCTION
SHL CL,1 ; FUNCTION TIMES 2
MOV BX,OFFSET FNC_TAB ; LOAD START OF FUNCTION TABLE
ADD BX,CX ; ADD OFFSET INTO TABLE => ROUTINE
MOV AH,DH ; AX = HEAD #, # OF SECTORS OR DASD TYPE
XOR DH,DH ; DX = DRIVE #
MOV SI,AX ; SI = HEAD #, # OF SECTORS OR DASD TYPE
MOV DI,DX ; DI = DRIVE #
MOV AH,@DSKETTE_STATUS ; LOAD STATUS TO AH FOR STATUS FUNCTION
MOV @DSKETTE_STATUS,0 ; INITIALIZE FOR ALL OTHERS
;
; THROUGHOUT THE DISKETTE BIOS, THE FOLLOWING INFORMATION IS CONTAINED IN
; THE FOLLOWING MEMORY LOCATIONS AND REGISTERS. NOT ALL DISKETTE BIOS
; FUNCTIONS REQUIRE ALL OF THESE PARAMETERS.
;

```

SECTION 5

```

482 ;          DI          : DRIVE #
483 ;          SI-HI      : HEAD #
484 ;          SI-LOW     : # OF SECTORS OR DASD TYPE FOR FORMAT
485 ;          ES         : BUFFER SEGMENT
486 ;          [BP]       : SECTOR #
487 ;          [BP+1]    : TRACK #
488 ;          [BP+2]    : BUFFER OFFSET
489 ;
490 ; ACROSS CALLS TO SUBROUTINES THE CARRY FLAG (CY=1), WHERE INDICATED IN
491 ; SUBROUTINE PROLOGUES, REPRESENTS AN EXCEPTION RETURN (NORMALLY AN ERROR
492 ; CONDITION). IN MOST CASES, WHEN CY = 1, @DSKETTE_STATUS CONTAINS THE
493 ; SPECIFIC ERROR CODE.
494 ;
495 ;          (AH) = @DSKETTE_STATUS
496 ;          CALL THE REQUESTED FUNCTION
497 00A1 2E: FF 17      CALL WORD PTR CS:[BX]
498 ;
499 ;          SI          : RESTORE ALL REGISTERS
500 00A4 5E           POP SI
501 00A5 1F           POP DS
502 00A6 59           POP CX
503 00A7 5B           POP BX
504 00A8 5A           POP DX
505 00A9 5F           POP DI
506 00AA 8B EC       MOV BP,SP
507 00AC 50           PUSH AX
508 00AD 9C           PUSHF
509 00AE 58           POP AX
510 00AF 89 46 06     MOV [BP+6],AX
511 00B2 58           POP AX
512 00B3 5D           POP BP
513 00B4 CF           IRET
514 ;
515 ;-----
516 FNC_TAB DW DISK_RESET          ; AH = 00H: RESET
517          DW DISK_STATUS        ; AH = 01H: STATUS
518          DW DISK_READ          ; AH = 02H: READ
519          DW DISK_WRITE         ; AH = 03H: WRITE
520          DW DISK_VERIFY        ; AH = 04H: VERIFY
521          DW DISK_FORMAT        ; AH = 05H: FORMAT
522          DW FNC_ERR             ; AH = 06H: INVALID
523          DW FNC_ERR             ; AH = 07H: INVALID
524          DW DISK_PARAMS        ; AH = 08H: READ DRIVE PARAMETERS
525          DW FNC_ERR             ; AH = 09H: INVALID
526          DW FNC_ERR             ; AH = 0AH: INVALID
527          DW FNC_ERR             ; AH = 0BH: INVALID
528          DW FNC_ERR             ; AH = 0CH: INVALID
529          DW FNC_ERR             ; AH = 0DH: INVALID
530          DW FNC_ERR             ; AH = 0EH: INVALID
531          DW FNC_ERR             ; AH = 0FH: INVALID
532          DW FNC_ERR             ; AH = 10H: INVALID
533          DW FNC_ERR             ; AH = 11H: INVALID
534          DW FNC_ERR             ; AH = 12H: INVALID
535          DW FNC_ERR             ; AH = 13H: INVALID
536          DW FNC_ERR             ; AH = 14H: INVALID
537          DW DISK_TYPE           ; AH = 15H: READ DASD TYPE
538          DW DISK_CHANGE        ; AH = 16H: CHANGE STATUS
539          DW FORMAT_SET         ; AH = 17H: SET DASD TYPE
540          DW SET_MEDIA          ; AH = 18H: SET MEDIA TYPE
541          $                      ; END
542 ;-----
543 FNC_TAB EQU DISKETTE_10:1 ENDP
544 ;-----
545 ; DISK_RESET (AH = 00H)
546 ; RESET THE DISKETTE SYSTEM.
547 ;
548 ; ON EXIT: @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
549 ;-----
550 DISK_RESET PROC NEAR
551             DX,03F2H          ; ADAPTER CONTROL PORT
552             CLT              ; NO INTERRUPTS
553             MOV AL,@MOTOR_STATUS ; GET DIGITAL OUTPUT REGISTER REFLECTION
554             AND AL,00111111B   ; KEEP SELECTED AND MOTOR ON BITS
555             ROL AL,4          ; MOTOR VALUE TO HIGH NIBBLE
556             OR AL,00001000B    ; DRIVE SELECT TO LOW NIBBLE
557             OUT DX,AL         ; TURN ON INTERRUPT ENABLE
558             MOV @SEEK_STATUS,0 ; RESET THE ADAPTER
559             JMP $+2           ; SET RECALIBRATE REQUIRED ON ALL DRIVES
560             JMP $+2           ; WAIT FOR I/O (TO INSURE MINIMUM
561             ; PULSE WIDTH)
562             OR AL,00000100B    ; TURN OFF RESET BIT
563             OUT DX,AL         ; RESET THE ADAPTER
564             STI              ; ENABLE THE INTERRUPTS
565             CALL WAIT_INT     ; WAIT FOR THE INTERRUPT
566             JC DR_ERR        ; IF ERROR, RETURN IT
567             MOV CX,11000000B  ; CL = EXPECTED @NEC_STATUS
568 ;-----
569 NXT_DRV:   PUSH CX          ; SAVE FOR CALL
570             MOV AX,OFFSET DR_POP_ERR ; LOAD NEC_OUTPUT ERROR ADDRESS
571             PUSH AX          ;
572             MOV AH,08H       ; SENSE INTERRUPT STATUS COMMAND
573             CALL NEC_OUTPUT  ;
574             POP AX           ;
575             CALL RESULTS     ; THROW AWAY ERROR RETURN
576             POP CX          ; READ IN THE RESULTS
577             JC DR_ERR        ; RESTORE AFTER CALL
578             CMP CL,@NEC_STATUS ; TEST FOR DRIVE READY TRANSITION
579             JNZ DR_ERR       ; EVERYTHING OK
580             INC CL           ; NEXT EXPECTED @NEC STATUS
581             CMP CL,11000011B ; ALL POSSIBLE DRIVES CLEARED
582             JBE NXT_DRV     ; FALL THRU IF 11000100B OR >
583             CALL SEND_SPEC   ; SEND SPECIFY COMMAND TO NEC
584 ;-----
585 RESBAC:   CALL SETUP_END     ; VARIOUS CLEANUPS
586             MOV BX,SI        ; GET SAVED AL TO BL
587             MOV AL,BL        ; PUT BACK FOR RETURN
588             RET
589 ;-----
590 DR_POP_ERR: POP CX          ; CLEAR STACK
591 ;-----
592 DR_ERR:   OR @DSKETTE_STATUS,BAD_NEC ; SET ERROR CODE
593             JMP SHORT RESBAC ; RETURN FROM RESET
594 ;-----
595 DISK_RESET ENDP
    
```



```

566
567
568
569
570
571
572
573
574 013C
575 013C 88 26 0041 R
576 0140 E8 0854 R
577 0143 8B DE
578 0145 8A C3
579 0147 C3
580 0148
581
582
583
584
585
586
587
588
589
590
591
592
593
594 0148
595 0148 80 26 003F R 7F
596 014D B8 E646
597 0150 E8 04AE R
598 0153 C3
599 0154
600
601
602
603
604
605
606
607
608
609
610
611
612
613 0154
614 0154 B8 C54A
615 0157 80 0E 003F R 80
616 015C E8 04AE R
617 015F C3
618 0160
619
620
621
622
623
624
625
626
627
628
629
630
631
632 0160
633 0160 80 26 003F R 7F
634 0165 B8 E642
635 0168 E8 04AE R
636 016B C3
637 016C
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653 016C
654 018C E8 040F R
655 018F E8 089B R
656 0172 80 0E 003F R 80
657 0177 E8 08E9 R
658 017A 72 3F
659 017C E8 03DB R
660 017F E8 063D R
661 0182 74 03
662 0184 E8 0624 R
663 0187
664 0187 E8 06AD R
665 018A 72 2F
666 018C B4 4D
667 018E E8 0700 R
668 0191 72 28
669 0193 B8 01BB R
670 0196 50
671 0197 B2 03
672 0199 E8 0905 R
673 019C E8 09F8 R
674 019F B2 04
675 01A1 E8 0905 R
676 01A4 E8 09F8 R
677 01A7 B2 07
678 01A9 E8 0905 R
679 01AC E8 09F8 R

```

```

-----
; DISK_STATUS (AH = 01H)
; DISKETTE STATUS.
; ON ENTRY: AH = STATUS OF PREVIOUS OPERATION
; ON EXIT: #DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
-----
DISK_STATUS PROC NEAR
MOV #DSKETTE_STATUS,AH ; PUT BACK FOR SETUP_END
CALL SETUP_END ; VARIOUS CLEANUPS
MOV BX,SI ; GET SAVED AL TO BL
MOV AL,BL ; PUT BACK FOR RETURN
RET
DISK_STATUS ENDP
-----
; DISK_READ (AH = 02H)
; DISKETTE READ.
; ON ENTRY: DI = DRIVE #
; SI-HI = HEAD #
; SI-LOW = # OF SECTORS
; ES = BUFFER SEGMENT
; [BP] = SECTOR #
; [BP+1] = TRACK #
; [BP+2] = BUFFER OFFSET
; ON EXIT: #DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
-----
DISK_READ PROC NEAR
AND #MOTOR_STATUS,01111111B ; INDICATE A READ OPERATION
MOV AX,0E646H ; AX = NEC COMMAND, DMA COMMAND
CALL RD_WR_VF ; COMMON READ/WRITE/VERIFY
RET
DISK_READ ENDP
-----
; DISK_WRITE (AH = 03H)
; DISKETTE WRITE.
; ON ENTRY: DI = DRIVE #
; SI-HI = HEAD #
; SI-LOW = # OF SECTORS
; ES = BUFFER SEGMENT
; [BP] = SECTOR #
; [BP+1] = TRACK #
; [BP+2] = BUFFER OFFSET
; ON EXIT: #DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
-----
DISK_WRITE PROC NEAR
MOV AX,0C54AH ; AX = NEC COMMAND, DMA COMMAND
OR #MOTOR_STATUS,10000000B ; INDICATE WRITE OPERATION
CALL RD_WR_VF ; COMMON READ/WRITE/VERIFY
RET
DISK_WRITE ENDP
-----
; DISK_VERIFY (AH = 04H)
; DISKETTE VERIFY.
; ON ENTRY: DI = DRIVE #
; SI-HI = HEAD #
; SI-LOW = # OF SECTORS
; ES = BUFFER SEGMENT
; [BP] = SECTOR #
; [BP+1] = TRACK #
; [BP+2] = BUFFER OFFSET
; ON EXIT: #DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
-----
DISK_VERIFY PROC NEAR
AND #MOTOR_STATUS,01111111B ; INDICATE A READ OPERATION
MOV AX,0E642H ; AX = NEC COMMAND, DMA COMMAND
CALL RD_WR_VF ; COMMON READ/WRITE/VERIFY
RET
DISK_VERIFY ENDP
-----
; DISK_FORMAT (AH = 05H)
; DISKETTE FORMAT.
; ON ENTRY: DI = DRIVE #
; SI-HI = HEAD #
; SI-LOW = # OF SECTORS
; ES = BUFFER SEGMENT
; [BP] = SECTOR #
; [BP+1] = TRACK #
; [BP+2] = BUFFER OFFSET
; #DISK_POINTER POINTS TO THE PARAMETER TABLE OF
; THIS DRIVE
; ON EXIT: #DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
-----
DISK_FORMAT PROC NEAR
CALL XLAT_NEW ; TRANSLATE STATE TO PRESENT ARCH.
CALL FMT_INIT ; ESTABLISH STATE IF UNESTABLISHED
OR #MOTOR_STATUS,10000000B ; INDICATE WRITE OPERATION
CALL MED_CHANGE ; CHECK MEDIA CHANGE AND RESET IF SO
CALL FM_DON ; MEDIA CHANGED, SKIP
CALL SEND_SPEC ; SEND SPECIFY COMMAND TO NEC
CALL CHK_LASTRATE ; ZF=1 ATTEMPT RATE IS SAME AS LAST RATE
CALL FM_WR ; YES, SKIP SPECIFY COMMAND
CALL SEND_RATE ; SEND DATA RATE TO CONTROLLER
FM_WR:
CALL FMTDMA_SET ; SET UP THE DMA FOR FORMAT
JC FM_DON ; RETURN WITH ERROR
MOV AH,04DH ; ESTABLISH THE FORMAT COMMAND
CALL NEC_INIT ; INITIALIZE THE NEC
JC FM_DON ; ERROR - EXIT
MOV AX,OFFSET FM_DON ; LOAD ERROR ADDRESS
PUSH AX ; PUSH NEC_OUT ERROR RETURN
MOV DL,3 ; BYTES/SECTOR VALUE TO NEC
CALL GET_PARM
CALL NEC_OUTPUT
MOV DL,4 ; SECTORS/TRACK VALUE TO NEC
CALL GET_PARM
CALL NEC_OUTPUT
MOV DL,7 ; GAP LENGTH VALUE TO NEC
CALL GET_PARM
CALL NEC_OUTPUT

```

SECTION 5

```

680 01AF B2 08      MOV     DL,8           ; FILLER BYTE TO NEC
681 01B1 E8 0905 R  CALL    GET_PARM
682 01B4 E8 09F8 R  CALL    NEC_OUTPUT
683 01B7 58        POP     AX
684 01B8 E8 075B R  CALL    NEC_TERM      ; THROW AWAY ERROR
                        ; TERMINATE, RECEIVE STATUS, ETC.
685 01BB          FM_DONE:
686 01BB E8 0435 R  CALL    XLAT_OLD      ; TRANSLATE STATE TO COMPATIBLE MODE
687 01BE E8 0854 R  CALL    SETUP_END    ; VARIOUS CLEANUPS
688 01C1 8B DE      MOV     BX,SI         ; GET SAVED AL TO BL
689 01C3 8A C3     MOV     AL,BL         ; PUT BACK FOR RETURN
690 01C5 C3       RET
691 01C6          DISK_FORMAT  ENDP
692
693          FNC_ERR
694          ; INVALID FUNCTION REQUESTED OR INVALID DRIVE;
695          ; SET BAD COMMAND IN STATUS.
696
697          ; ON EXIT:  @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION ;
698          ;-----
699 01C6          FNC_ERR PROC  NEAR      ; INVALID FUNCTION REQUEST
700 01C6 8B C6     MOV     AX,SI         ; RESTORE AL
701 01C8 B4 01     MOV     AH,BAD_CMD   ; SET BAD COMMAND ERROR
702 01CA 8B 26 0041 R  MOV     AH,@DSKETTE_STATUS,AH ; STORE IN DATA AREA
703 01CE F9       STC
704 01CF C3       RET
705 01D0          FNC_ERR ENDP
706
707          DISK_PARAMS (AH = 0BH)
708          ; READ DRIVE PARAMETERS.
709          ; ON ENTRY:
710          ; D1 = DRIVE #
711          ; ON EXIT:
712          ; CL/[BP] = BITS 7 & 6 HIGH 2 BITS OF MAX CYLINDER
713          ;             BITS 0-5 MAX SECTORS/TRACK
714          ; CH/[BP+1] = LOW 8 BITS OF MAX CYLINDER
715          ; BL/[BP+2] = BITS 7-4 = 0
716          ;             BITS 3-0 = VALID CMOS DRIVE TYPE
717          ; BH/[BP+3] = 0
718          ; DL/[BP+4] = # DRIVES INSTALLED
719          ; DH/[BP+5] = MAX HEAD #
720          ; DI/[BP+6] = OFFSET OF MEDIA/DRIVE PARAMETER TABLE
721          ; ES = SEGMENT OF MEDIA/DRIVE PARAMETER TABLE
722          ; AX = 0
723
724          ; NOTE: THE ABOVE INFORMATION IS STORED IN THE USERS STACK AT
725          ; THE LOCATIONS WHERE THE MAIN ROUTINE WILL POP THEM
726          ; INTO THE APPROPRIATE REGISTERS BEFORE RETURNING TO THE
727          ; CALLER.
728          ;-----
729 01D0          DISK_PARAMS PROC  NEAR
730 01D0 E8 040F R  CALL    XLAT_NEW      ; TRANSLATE STATE TO PRESENT ARCH.
731 01D3 C7 46 02 0000 MOV     WORD PTR [BP+2],0 ; DRIVE TYPE = 0
732 01D8 A1 0010 R  MOV     AX,@EQUIP_FLAG ; LOAD EQUIPMENT FLAG FOR # DISKETTES
733 01DB 24 C1     AND     AL,11000001B ; KEEP DISKETTE DRIVE BITS
734 01DD B2 02     MOV     DL,2         ; DISKETTE DRIVES = 2
735 01DF 3C 41     CMP     AL,01000001B ; 2 DRIVES INSTALLED ?
736 01E1 74 06     JZ     STO_DL        ; IF YES JUMP
737
738          DEC     DL           ; DISKETTE DRIVES = 1
739          CMP     AL,00000001B ; 1 DRIVE INSTALLED ?
740          JNZ    NON_DRY     ; IF NO JUMP
741
742          STO_DL: MOV     [BP+4],DL      ; STORE NUMBER OF DRIVES
743          CMP     DI,1       ; CHECK FOR VALID DRIVE
744          JNA     NON_DRY   ; DRIVE INVALID
745          MOV     BYTE PTR [BP+5],1 ; MAXIMUM HEAD NUMBER = 1
746          CALL    CMOS_TYPE ; RETURN DRIVE TYPE IN AL
747          JC     CHK_EST   ; ON CMOS BAD CHECK ESTABLISHED
748          OR     AL,AL      ; TEST FOR NO DRIVE TYPE
749          JZ     CHK_EST   ; JUMP IF SO
750          CALL    DR_TYPE_CHECK ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
751          JC     CHK_EST   ; TYPE NOT IN TABLE (POSSIBLE BAD CMOS)
752          MOV     [BP+2],AL ; STORE VALID CMOS DRIVE TYPE
753          MOV     CL,CS:[BX].MD_SEC_TRK ; GET SECTOR/TRACK
754          MOV     CH,CS:[BX].MD_MAX_TRK ; GET MAX. TRACK NUMBER
755          JMP     SHORT STO_CX ; CMOS GOOD, USE CMOS
756
757          CHK_EST: MOV     AH,@DSK_STATE[D1] ; LOAD STATE FOR THIS DRIVE
758          TEST    AH,MED_DET ; CHECK FOR ESTABLISHED STATE
759          JZ     NON_DRY   ; CMOS BAD/INVALID AND UNESTABLISHED
760
761          USE_EST: AND     AH,RATE_MSK ; ISOLATE STATE
762          CMP     AH,RATE_250 ; RATE 250 ?
763          JNE     USE_EST2 ; NO, GO CHECK OTHER RATE
764
765          ;--- DATA RATE IS 250 KBS, TRY 360 KB TABLE FIRST
766
767          MOV     AL,01     ; DRIVE TYPE 1 (360KB)
768          CALL    DR_TYPE_CHECK ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
769          MOV     CL,CS:[BX].MD_SEC_TRK ; GET SECTOR/TRACK
770          MOV     CH,CS:[BX].MD_MAX_TRK ; GET MAX. TRACK NUMBER
771          TEST    @DSK_STATE[D1],TRK_CAPA ; 80 TRACK ?
772          JZ     STO_CX   ; MUST BE 360KB DRIVE
773
774          ;--- IT IS 1.44 MB DRIVE
775
776          PARM144: MOV     AL,04     ; DRIVE TYPE 4 (1.44MB)
777          CALL    DR_TYPE_CHECK ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
778          MOV     CL,CS:[BX].MD_SEC_TRK ; GET SECTOR/TRACK
779          MOV     CH,CS:[BX].MD_MAX_TRK ; GET MAX. TRACK NUMBER
780
781          STO_CX: MOV     [BP],CX ; SAVE IN STACK FOR RETURN
782          MOV     [BP+1],BX ; ADDRESS OF MEDIA/DRIVE PARAM TABLE
783          MOV     AX,CS ; SEGMENT OF MEDIA/DRIVE PARAMETER TABLE
784          MOV     ES,AX ; ES IS SEGMENT OF TABLE
785
786          DP_OUT: CALL    XLAT_OLD ; TRANSLATE STATE TO COMPATIBLE MODE
787          XOR     AX,AX ; CLEAR
788          CLC
789          RET
790
791 024C E8 0435 R  CALL    XLAT_OLD
792 024F 33 C0     XOR     AX,AX
793 0251 F8       CLC
794 0252 C3       RET
    
```

```

794
795
796
797 0253
798 0253 C6 46 04 00
799
800 0257
801 0257 81 FF 0080
802 025B 72 09
803
804
805
806 025D E8 0435 R
807 0260 8B C6
808 0262 84 01
809 0264 F9
810 0265 C3
811
812 0266
813 0266 33 C0
814 0268 89 46 00
815 026B 88 66 05
816 026E 89 46 06
817 0271 8E C0
818 0273 EB DT
819
820
821
822 0275
823 0275 80 02
824 0277 E8 03B5 R
825 027A 2E 8A 4F 04
826 027E 2E 8A 6F 0B
827 0282 50 FC 40
828 0285 74 B5
829 0287 EB AC
830
831 0289
832
833
834
835
836
837
838
839
840 0289
841 0289 EB 040F R
842 029C 8A 85 0090 R
843 0290 0A C0
844 0292 74 13
845 0294 84 01
846 0296 A8 01
847 0298 74 02
848 029A B4 02
849
850 029C
851 029C 50
852 029D E8 0435 R
853 02A0 58 01
854 02A1 F8
855 02A2 8B DE
856 02A4 8A C3
857 02A6 C3
858 02A7
859 02A7 32 E4
860 02A9 EB F1
861 02AB
862
863
864
865
866
867
868
869
870
871
872 02AB
873 02AB E8 040F R
874 02AC 8A 85 0090 R
875 02B5 0A C0
876 02B4 74 19
877 02B6 A8 01
878 02B8 74 05
879
880 02BA E8 0B28 R
881 02B0 74 05
882
883 02BF C6 06 0041 R 06
884
885 02C4 88 0435 R
886 02C7 E8 0854 R
887 02CA 8B DE
888 02CC 8A C3
889 02CE C3
890
891 02CF
892 02CF 80 0E 0041 R 80
893 02D4 EB EE
894 02D6
895
896
897
898
899
900
901
902
903
904
905
906
907

;----- NO DRIVE PRESENT HANDLER
NON_DRY:
MOV     BYTE PTR [BP+4],0 ; CLEAR NUMBER OF DRIVES
NON_DRY1:
CMP     DI,80H ; CHECK FOR FIXED MEDIA TYPE REQUEST
JB      NON_DRY2 ; CONTINUE IF NOT REQUEST FALL THROUGH

;----- FIXED DISK REQUEST FALL THROUGH ERROR
CALL    XLAT_OLD ; ELSE TRANSLATE TO COMPATIBLE MODE
MOV     AX,ST ; RESTORE AL
MOV     AH,BAD_CMD ; SET BAD COMMAND ERROR
STC ; SET ERROR RETURN CODE
RET

NON_DRY2:
XOR     AX,AX ; CLEAR PARMS IF NO DRIVES OR CMOS BAD
MOV     [BP],AX ; TRACKS, SECTORS/TRACK = 0
MOV     [BP+5],AH ; HEAD = 0
MOV     [BP+6],AX ; RATE 300 ?
MOV     ES,AX ; ES IS SEGMENT OF TABLE
JMP     SHORT DP_OUT

;--- DATA RATE IS EITHER 300 KBS OR 500 KBS, TRY 1.2 MB TABLE FIRST
USE_EST2:
MOV     AL,02 ; DRIVE TYPE 2 (1.2MB)
CALL    DR_TYPE_CHECK ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
MOV     CL,Cs:[BX].MD_SEC_TRK ; GET SECTOR/TRACK
MOV     CH,Cs:[BX].MD_MAX_TRK ; GET MAX. TRACK NUMBER
CMP     AH,RATE_300 ; RATE 300 ?
JE      STO_CX ; MUST BE 1.2MB DRIVE
JMP     SHORT PARM144 ; ELSE, IT IS 1.44MB DRIVE

DISK_PARMS ENDP

;-----
; DISK_TYPE (AH = 18H)
; THIS ROUTINE RETURNS THE TYPE OF MEDIA INSTALLED.
; ON ENTRY: DI = DRIVE #
; ON EXIT: AH = DRIVE TYPE, CY=0
;-----
DISK_TYPE PROC NEAR
CALL    XLAT_NEW ; TRANSLATE STATE TO PRESENT ARCH.
MOV     AL,MDSK_STATE[DI] ; GET MEDIA STATE INFORMATION
OR      AL,AL ; CHECK FOR NO DRIVE
JZ      NO_DRV ;
MOV     AH,NOCHGLN ; NO CHANGE LINE FOR 40 TRACK DRIVE
TEST    AL,TRK_CAPA ; IS THIS DRIVE AN 80 TRACK DRIVE?
JZ      DT_BACK ; IF NO JUMP
MOV     AH,CHGLN ; CHANGE LINE FOR 80 TRACK DRIVE
DT_BACK:
PUSH    AX ; SAVE RETURN VALUE
CALL    XLAT_OLD ; TRANSLATE STATE TO COMPATIBLE MODE
POP     AX ; RESTORE RETURN VALUE
CLC ; NO ERROR
MOV     BX,SI ; GET SAVED AL TO BL
MOV     AL,BL ; PUT BACK FOR RETURN
RET

NO_DRV:
XOR     AH,AH ; NO DRIVE PRESENT OR UNKNOWN
JMP     SHORT DT_BACK
DISK_TYPE ENDP

;-----
; DISK_CHANGE (AH = 18H)
; 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
;-----
DISK_CHANGE PROC NEAR
CALL    XLAT_NEW ; TRANSLATE STATE TO PRESENT ARCH.
MOV     AL,MDSK_STATE[DI] ; GET MEDIA STATE INFORMATION
OR      AL,AL ; DRIVE PRESENT ?
JZ      DC_NON ; JUMP IF NO DRIVE
TEST    AL,TRK_CAPA ; 80 TRACK DRIVE ?
JZ      SETIT ; IF 50, CHECK CHANGE LINE
DC0: CALL    READ_DSKCHNG ; GO CHECK STATE OF DISK CHANGE LINE
JZ      FINIS ; CHANGE LINE NOT ACTIVE
SETIT: MOV     DSKETTE_STATUS,MEDIA_CHANGE ; INDICATE MEDIA REMOVED
FINIS: CALL    XLAT_OLD ; TRANSLATE STATE TO COMPATIBLE MODE
CALL    SETUP_END ; VARIOUS CLEANUPS
MOV     BX,SI ; GET SAVED AL TO BL
MOV     AL,BL ; PUT BACK FOR RETURN
RET

DC_NON: OR      DSKETTE_STATUS,TIME_OUT ; SET TIMEOUT, NO DRIVE
JMP     SHORT FINIS
DISK_CHANGE ENDP

;-----
; FORMAT SET (AH = 17H)
; THIS ROUTINE IS USED TO ESTABLISH THE TYPE OF
; MEDIA TO BE USED FOR THE FOLLOWING FORMAT OPERATION.
; ON ENTRY: SI LOW = DASD TYPE FOR FORMAT
;          DI = DRIVE #
; ON EXIT: DSKETTE_STATUS REFLECTS STATUS
;          AH = DSKETTE STATUS
;          CY = 1 IF ERROR
;-----

```

SECTION 5

```

908 02D6          PROC NEAR
909 02D6 E8 040F R    CALL XLAT_NEW          ; TRANSLATE STATE TO PRESENT ARCH.
910 02D9 56          PUSH SI                ; SAVE DASD TYPE
911 02DA 8B C6       MOV AX,SI              ; AH = ? , AL = DASD TYPE
912 02DC 32 E4       XOR AH,AH              ; AH = 0 , AL = DASD TYPE
913 02DE 8B F0       MOV SI,AX              ; SI = DASD TYPE
914 02E0 80 A5 0090 R DF AND @DSK_STATE[D1],NOT MED_DET+DBL_STEP+RATE_MSK ; CLEAR STATE
915 02E5 4E          DEC SI                 ; CHECK FOR 320/360K MEDIA & DRIVE
916 02E6 75 07       JNZ NOT_320           ; BYPASS IF NOT
917 02E8 80 8D 0090 R 90 OR @DSK_STATE[D1],MED_DET+RATE_250 ; SET TO 320/360
918 02ED EB 37       JMP SHORT S0
919
920 02EF          NOT_320:
921 02EF E8 05E9 R    CALL MED_CHANGE        ; CHECK FOR TIME_OUT
922 02F2 80 3E 0041 R 80 CMP @DSKETTE_STATUS,TIME_OUT ; @DSKETTE_STATUS,TIME_OUT
923 02F7 74 2D       JZ S0                 ; IF TIME OUT TELL CALLER
924
925 02F9 4E          S31: DEC SI              ; CHECK FOR 320/360K IN 1.2M DRIVE
926 02FA 75 07       JNZ NOT_320_12        ; BYPASS IF NOT
927 02FC 80 8D 0090 R 70 OR @DSK_STATE[D1],MED_DET+DBL_STEP+RATE_300 ; SET STATE
928 0301 EB 23       JMP SHORT S0
929
930 0303          NOT_320_12:
931 0303 4E          DEC SI                 ; CHECK FOR 1.2M MEDIA IN 1.2M DRIVE
932 0304 75 07       JNZ NOT_12            ; BYPASS IF NOT
933 0306 80 8D 0090 R 10 OR @DSK_STATE[D1],MED_DET+RATE_500 ; SET STATE VARIABLE
934 030B EB 19       JMP SHORT S0          ; RETURN TO CALLER
935
936 030D          NOT_12:
937 030D 4E          DEC SI                 ; CHECK FOR SET DASD TYPE 04
938 030E 75 20       JNZ FS_ERR            ; BAD COMMAND EXIT IF NOT VALID TYPE
939
940 0310 F6 85 0090 R 04 TEST @DSK_STATE[D1],DRV_DET ; DRIVE DETERMINED ?
941 0315 74 09       JZ ASSUME             ; IF STILL NOT DETERMINED ASSUME
942 0317 80 50       MOV AL,MED_DET+RATE_300 ;
943 0319 F6 85 0090 R 02 TEST @DSK_STATE[D1],FMT_CAPA ; MULTIPLE FORMAT CAPABILITY ?
944 031E 75 02       JNZ OR_IT_IN         ; IF 1.2 M THEN DATA RATE 300
945
946 0320          ASSUME:
947 0320 80 90       MOV AL,MED_DET+RATE_250 ; SET UP
948
949 0322          OR_IT_IN:
950 0322 08 55 0090 R OR @DSK_STATE[D1],AL ; OR IN THE CORRECT STATE
951
952 0326 E8 0435 R    S0: CALL XLAT_OLD          ; TRANSLATE STATE TO COMPATIBLE MODE
953 0329 E8 0854 R    CALL SETUP_END        ; VARIOUS CLEANUPS
954 032C 58          POP SI                ; GET SAVED AL TO BL
955 032D 8A C3       MOV BL,AL             ; PUT BACK FOR RETURN
956 032F C3          RET
957
958 0330          FS_ERR:
959 0330 C6 06 0041 R 01 MOV @DSKETTE_STATUS,BAD_CMD ; UNKNOWN STATE,BAD COMMAND
960 0335 EB EF       JMP SHORT S0
961
962 0337          FORMAT_SET ENDP
963
964 -----
965 | SET_MEDIA (AH = 18H)
966 | THIS ROUTINE SETS THE TYPE OF MEDIA AND DATA RATE
967 | TO BE USED FOR THE FOLLOWING FORMAT OPERATION.
968 |
969 | ON ENTRY:
970 | [BP] = SECTOR PER TRACK
971 | [BP+1] = TRACK #
972 | DI = DRIVE #
973 |
974 | ON EXIT:
975 | @DSKETTE_STATUS REFLECTS STATUS
976 | IF NO ERROR:
977 | AH = 0
978 | CY = 0
979 |
980 | ES = SEGMENT OF MEDIA/DRIVE PARAMETER TABLE
981 | DI/[BP+6] = OFFSET OF MEDIA/DRIVE PARAMETER TABLE
982 | IF ERROR:
983 | AH = @DSKETTE_STATUS
984 | CY = 1
985 |-----
986
987 0337          SET_MEDIA PROC NEAR
988 0337 E8 040F R    CALL XLAT_NEW          ; TRANSLATE STATE TO PRESENT ARCH.
989 033A F6 85 0090 R 01 TEST @DSK_STATE[D1],TRK_CAPA ; CHECK FOR CHANGE LINE AVAILABLE
990 033F 74 0F       JZ SM_CMOS            ; JUMP IF 40 TRACK DRIVE
991 0341 E8 05E9 R    CALL MED_CHANGE        ; RESET CHANGE LINE
992 0344 80 3E 0041 R 80 CMP @DSKETTE_STATUS,TIME_OUT ; IF TIME OUT TELL CALLER
993 0349 74 66       JE SM_RTN             ; CLEAR STATUS
994 034B C6 06 0041 R 00 MOV @DSKETTE_STATUS,0
995
996 0350          SM_CMOS:
997 0350 E8 08EC R    CALL CMOS_TYPE         ; RETURN DRIVE TYPE IN (AL)
998 0353 72 38       JC MD_NOT_FND         ; ERROR IN CMOS
999 0355 0A C0       OR AL,TAL             ; TEST FOR NO DRIVE
1000 0357 74 58       JZ SM_RTN             ; RETURN IF SO
1001 0359 E8 03B8 R    CALL DR_TYPE_CHECK     ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
1002 035C 72 2F       JC MD_NOT_FND         ; TYPE NOT IN TABLE (BAD CMOS)
1003 035E 87          PUSH DI               ; SAVE REG.
1004 035F 33 DB       XOR BX,BX             ; BX = INDEX TO DR_TYPE TABLE
1005 0361 89 0006     MOV CX,DR_CNT         ; CX = LOOP COUNT
1006
1007 0364          DR_SEARCH:
1008 0364 87          MOV AH,CS:DR_TYPE[BX] ; GET DRIVE TYPE
1009 0366 80 E4 7F     AND AH,BITTOFF        ; MASK OUT MSB
1010 0368 8A C4       CMP AL,AH             ; DRIVE TYPE MATCH ?
1011 036A 75 17       JNE NXT_MD           ; NO, CHECK NEXT DRIVE TYPE
1012
1013 0370 87          MOV DI,CS:WORD_PTR DR_TYPE[BX+1] ; DI = MEDIA/DRIVE PARAMETER TABLE
1014
1015 0375 87          MOV AH,CS:[DI].MD_SEC_TRK ; GET SECTOR/TRACK
1016 0377 86 66 00     CMP [BP],AH           ; MATCH ?
1017 0379 75 09       JNE NXT_MD           ; NO, CHECK NEXT MEDIA
1018 037E 2E: 8A 65 0B MOV AH,CS:[DI].MD_MAX_TRK ; GET MAX. TRACK #
1019 0382 86 66 01     CMP [BP+1],AH         ; MATCH ?
1020 0384 75 0D       JNE NXT_MD           ; YES, GO GET RATE
1021
1022 0387          NXT_MD:
1023 0387 83 C3 03         ADD BX,3              ; CHECK NEXT DRIVE TYPE
1024 0389 82 D8         LOOP DR_SEARCH        ; RESTORE REG.
1025 038B 8F          POP DI
1026
1027 038D          MD_NOT_FND:
1028 038D C6 06 0041 R 0C MOV @DSKETTE_STATUS,MED_NOT_FND ; ERROR, MEDIA TYPE NOT FOUND
1029 0392 EB 1D       JMP SHORT SM_RTN

```

```

1022
1023 0394 MD_FND:
1024 0394 2E: 8A 45 0C MOV AL,CS:[DI].MD_RATE ; GET RATE
1025 0396 3C 40 CMP AL,RATE_300 ; DOUBLE STEP REQUIRED FOR RATE 300
1026 039A 76 02 JNE MD_SET
1027 039C 0C 20 OR AL,DBL_STEP
1028 039E MD_SET:
1029 039E 89 7E 06 MOV [BP+6],DI ; SAVE TABLE POINTER IN STACK
1030 03A1 0C 10 OR AL,MED_DET ; SET MEDIA ESTABLISHED
1031 03A3 5F POP DI ; RESTORE REG.
1032 03A4 80 A5 0090 R OF AND #DSK_STATE[DI],NOT MED_DET+DBL_STEP+RATE_MSK ; CLEAR STATE
1033 03A9 08 85 0090 R OR #DSK_STATE[DI],AL ; SET STATE
1034 03AD 8C C8 MOV AX,CS ; SEGMENT MEDIA/DRIVE PARAMETER TABLE
1035 03AF 8E C0 MOV ES,AX ; ES IS SEGMENT OF TABLE
1036 03B1 SM_RTN:
1037 03B1 E8 0435 R CALL XLAT_OLD ; TRANSLATE STATE TO COMPATIBLE MODE
1038 03B4 E8 0854 R CALL SETUP_END ; VARIOUS CLEANUPS
1039 03B7 C3 RET
1040 03B8 SET_MEDIA ENDP
1041
1042 -----
1043 ; DR_TYPE_CHECK
1044 ; CHECK IF THE GIVEN DRIVE TYPE IN REGISTER (AL)
1045 ; IS SUPPORTED IN BIOS DRIVE TYPE TABLE
1046 ; ON ENTRY:
1047 ; AL = DRIVE TYPE
1048 ; ON EXIT:
1049 ; CS = SEGMENT OF MEDIA/DRIVE PARAMETER TABLE (CODE)
1050 ; CY = 0 DRIVE TYPE SUPPORTED
1051 ; BX = OFFSET TO MEDIA/DRIVE PARAMETER TABLE
1052 ; CY = 1 DRIVE TYPE NOT SUPPORTED
1053 ; REGISTERS ALTERED: BX
1054 -----
1055 03B8 DR_TYPE_CHECK PROC NEAR
1056 03B8 50 PUSH AX
1057 03B9 51 PUSH CX
1058 03BA 33 DB XOR BX,BX ; BX = INDEX TO DR_TYPE TABLE
1059 03BC B9 0006 MOV CX,DR_CNT ; CX = LOOP COUNT
1060 TYPE_CHK:
1061 03BF 2E: 8A AT 0000 R MOV AH,CS:DR_TYPE[BX] ; GET DRIVE TYPE
1062 03C4 3A C4 CMP AL,AH ; DRIVE TYPE MATCH ?
1063 03C6 74 08 JE DR_TYPE_VALID ; YES, RETURN WITH CARRY RESET
1064 03C8 83 C3 03 ADD BX,3 ; CHECK NEXT DRIVE TYPE
1065 03CB E2 F2 LOOP TYPE_CHK ; DRIVE TYPE NOT FOUND IN TABLE
1066 03CD F9 STC
1067 03CE EB 05 JMP SHORT TYPE_RTN
1068 03D0 DR_TYPE_VALID:
1069 03D0 2E: 8B 9F 0001 R MOV BX,CS:WORD PTR DR_TYPE[BX+1] ; BX = MEDIA TABLE
1070 03D5 TYPE_RTN:
1071 03D5 59 POP CX
1072 03D6 58 POP AX
1073 03D7 C3 RET
1074 03D8 DR_TYPE_CHECK ENDP
1075
1076 -----
1077 ; SEND_SPEC
1078 ; SEND THE SPECIFY COMMAND TO CONTROLLER USING DATA FROM
1079 ; THE DRIVE PARAMETER TABLE POINTED BY #DISK_POINTER
1080 ; ON ENTRY: #DISK_POINTER = DRIVE PARAMETER TABLE
1081 ; ON EXIT: NONE
1082 ; REGISTERS ALTERED: CX, DX
1083 -----
1084 03D8 SEND_SPEC PROC NEAR
1085 03D8 50 PUSH AX ; SAVE AX
1086 03D9 B8 03F3 R MOV AX,OFFSET_SPECBAC ; LOAD ERROR ADDRESS
1087 03DC 50 PUSH AX ; PUSH NEC_OUT ERROR RETURN
1088 03DD B4 03 MOV AH,03H ; SPECIFY COMMAND
1089 03DF E8 09F8 R CALL NEC_OUTPUT ; OUTPUT THE COMMAND
1090 03E2 2A 02 SUB DL,D ; FIRST SPECIFY BYTE
1091 03E4 E8 0905 R CALL GET_PARM ; GET PARAMETER TO AH
1092 03E7 E8 09F8 R CALL NEC_OUTPUT ; OUTPUT THE COMMAND
1093 03EA 92 01 MOV DL,T ; SECOND SPECIFY BYTE
1094 03EC E8 0905 R CALL GET_PARM ; GET PARAMETER TO AH
1095 03EF E8 09F8 R CALL NEC_OUTPUT ; OUTPUT THE COMMAND
1096 03F2 58 POP AX ; POP ERROR RETURN
1097 SPECBAC:
1098 03F3 58 POP AX ; RESTORE ORIGINAL AX VALUE
1099 03F4 C3 RET
1100 03F5 SEND_SPEC ENDP
1101
1102 -----
1103 ; SEND_SPEC_MD
1104 ; SEND THE SPECIFY COMMAND TO CONTROLLER USING DATA FROM
1105 ; THE MEDIA/DRIVE PARAMETER TABLE POINTED BY (CS:BX)
1106 ; ON ENTRY: CS:BX = MEDIA/DRIVE PARAMETER TABLE
1107 ; ON EXIT: NONE
1108 ; REGISTERS ALTERED: AX
1109 -----
1110 03F5 SEND_SPEC_MD PROC NEAR
1111 03F5 50 PUSH AX ; SAVE RATE DATA
1112 03F6 B8 040D R MOV AX,OFFSET_SPEC_ESBAC ; LOAD ERROR ADDRESS
1113 03F9 50 PUSH AX ; PUSH NEC_OUT ERROR RETURN
1114 03FA B4 03 MOV AH,03H ; SPECIFY COMMAND
1115 03FC E8 09F8 R CALL NEC_OUTPUT ; OUTPUT THE COMMAND
1116 03FF 2E: 8A 2T MOV AH,CS:[BX].MD_SPEC1 ; GET 1ST SPECIFY BYTE
1117 0402 E8 09F8 R CALL NEC_OUTPUT ; OUTPUT THE COMMAND
1118 0405 2E: 8A 67 01 MOV AH,CS:[BX].MD_SPEC2 ; GET SECOND SPECIFY BYTE
1119 0409 E8 09F8 R CALL NEC_OUTPUT ; OUTPUT THE COMMAND
1120 040C 58 POP AX ; POP ERROR RETURN
1121 040D SPEC_ESBAC:
1122 040D 58 POP AX ; RESTORE RATE
1123 040E C3 RET
1124 040F SEND_SPEC_MD ENDP

```

SECTION 5

```

1125 PAGE
1126 -----
1127 ; XLAT_NEW
1128 ; TRANSLATES DISKETTE STATE LOCATIONS FROM COMPATIBLE
1129 ; MODE TO NEW ARCHITECTURE.
1130 ;
1131 ; ON ENTRY: DI : DRIVE
1132 -----
1133 XLAT_NEW PROC NEAR
1134 040F 83 FF 01 CMP DI,1 ; VALID DRIVE ?
1135 0412 77 1C JA XN_OUT ; IF INVALID BACK
1136 0414 80 8D 0090 R 00 CMP #DSK_STATE[DI],0 ; NO DRIVE ?
1137 0419 74 16 JZ DO_DET ; IF NO DRIVE ATTEMPT DETERMINE
1138 041B 8B CF MOV CX,DI ; CX = DRIVE NUMBER
1139 041D C0 E1 02 SHL CL,2 ; CL = SHIFT COUNT, A=0, B=4
1140 0420 A0 008F R MOV AL,#HF_CNTRL ; DRIVE INFORMATION
1141 0423 D2 CB ROR AL,CL ; TO LOW NIBBLE
1142 0425 24 07 AND AL,DRV_DET+FMT_CAPA+TRK_CAPA ; KEEP DRIVE BITS
1143 0427 80 A5 0090 R F8 AND #DSK_STATE[DI],NOT DRV_DET+FMT_CAPA+TRK_CAPA
1144 042C 08 85 0090 R OR #DSK_STATE[DI],AL ; UPDATE DRIVE STATE
1145 0430
1146 0430 C3 XN_OUT: RET
1147
1148 0431 DO_DET: CALL DRIVE_DET ; TRY TO DETERMINE
1149 0431 E8 0B32 R CALL RET
1150 0434 C3 RET
1151
1152 0435 XLAT_NEW ENDP
1153 -----
1154 ; XLAT_OLD
1155 ; TRANSLATES DISKETTE STATE LOCATIONS FROM NEW
1156 ; ARCHITECTURE TO COMPATIBLE MODE.
1157 ;
1158 ; ON ENTRY: DI : DRIVE
1159 -----
1160 XLAT_OLD PROC NEAR
1161 0435 83 FF 01 CMP DI,1 ; VALID DRIVE ?
1162 0438 77 73 JA XO_OUT ; IF INVALID BACK
1163 043A 80 8D 0090 R 00 CMP #DSK_STATE[DI],0 ; NO DRIVE ?
1164 043F 74 6C JZ XO_OUT ; IF NO DRIVE TRANSLATE DONE
1165
1166 ;----- TEST FOR SAVED DRIVE INFORMATION ALREADY SET
1167
1168 0441 8B CF MOV CX,DI ; CX = DRIVE NUMBER
1169 0443 C0 E1 02 SHL CL,2 ; CL = SHIFT COUNT, A=0, B=4
1170 0446 B4 02 MOV AH,FMT_CAPA ; LOAD MULTI DATA RATE BIT MASK
1171 0448 D2 CC ROR AH,CL ; ROTATE BY MASK
1172 044A 84 26 008F R TEST #HF_CNTRL,AH ; MULTI-DATA RATE DETERMINED ?
1173 044E 75 16 JNZ SAVE_SET ; IF SO, NO NEED TO RE-SAVE
1174
1175 ;----- ERASE DRIVE BITS IN #HF_CNTRL FOR THIS DRIVE
1176
1177 0450 B4 07 MOV AH,DRV_DET+FMT_CAPA+TRK_CAPA ; MASK TO KEEP
1178 0452 D2 CC ROR AH,CL ; IF MASK TO KEEP
1179 0454 F4 D4 NOT AH ; TRANSLATE MASK
1180 0456 20 26 008F R AND #HF_CNTRL,AH ; KEEP BITS FROM OTHER DRIVE INTACT
1181
1182 ;----- ACCESS CURRENT DRIVE BITS AND STORE IN #HF_CNTRL
1183
1184 045A 8A 85 0090 R MOV AL,#DSK_STATE[DI] ; ACCESS STATE
1185 045E 24 07 AND AL,DRV_DET+FMT_CAPA+TRK_CAPA ; KEEP DRIVE BITS
1186 0460 D2 CC ROR AL,CL ; FIX FOR THIS DRIVE
1187 0462 08 06 008F R OR #HF_CNTRL,AL ; UPDATE SAVED DRIVE STATE
1188
1189 ;----- TRANSLATE TO COMPATIBILITY MODE
1190
1191 0466 SAVE_SET: MOV AH,#DSK_STATE[DI] ; ACCESS STATE
1192 0468 8A A5 0090 R MOV BH,AH ; TO BH FOR LATER
1193 046A 8A FC MOV AH,RATE_MSK ; KEEP ONLY RATE
1194 046C 80 E4 C0 CMP AH,RATE_500 ; RATE 500 ?
1195 046F 80 FC 00 CMP AH,RATE_300 ; RATE 300 ?
1196 0472 74 10 JZ CHK_144 ; YES, 1.2/1.2 OR 1.44/1.44
1197 0474 80 01 MOV AL,#D3D1U ; AL = 360 IN 1.2 UNESTABLISHED
1198 0476 80 FC 40 CMP AH,RATE_300 ; RATE 300 ?
1199 0479 75 16 JNZ CHK_250 ; NO, 360/360 ,720/720 OR 720/1.44
1200 047B F6 C7 20 TEST BH,DSL_STEP ; YES, DOUBLE STEP ?
1201 047E 75 1D JNZ TST_DET ; YES, MUST BE 360 IN 1.2
1202
1203 0480 UNKN01: MOV AL,MED_UNK ; 'NONE OF THE ABOVE'
1204 0480 80 07 JMP SHORT AL_SET ; PROCESS COMPLETE
1205 0482 EB 20
1206
1207 0484 CHK_144: CALL CHOS_TYPE ; RETURN DRIVE TYPE IN (AL)
1208 0484 E8 08EC R JC UNKN01 ; ERROR, SET 'NONE OF THE ABOVE'
1209 0487 72 F7 JNC UNKN01 ; 1.2MB DRIVE ?
1210 0489 3C 02 CMP AL,02 ; NO, GO SET 'NONE OF THE ABOVE'
1211 048B 75 F3 JNE UNKN01 ; AL = 1.2 IN 1.2 UNESTABLISHED
1212 048D 80 02 MOV AL,#D3D1U ; AL = 1.2 IN 1.2 UNESTABLISHED
1213 048F EB 0C JMP SHORT TST_DET
1214
1215 0491 CHK_250: MOV AL,#D3D3U ; AL = 360 IN 360 UNESTABLISHED
1216 0491 80 00 CMP AH,RATE_250 ; RATE 250 ?
1217 0493 80 FC 80 JNZ UNKN01 ; IF SO FALL THRU
1218 0496 75 E8 JNC UNKN01 ; IF SO FALL THRU
1219 0498 F6 C7 01 TEST BH,TRK_CAPA ; 80 TRACK CAPABILITY ?
1220 049B 75 E3 JNZ UNKN01 ; IF SO JUMP, FALL THRU TEST DET
1221
1222 049D TST_DET: TEST BH,MED_DET ; DETERMINED ?
1223 049D F6 C7 10 JZ AL_SET ; IF NOT THEN SET
1224 04A0 74 02 JZ AL_3 ; IF NOT THEN SET
1225 04A2 04 03 ADD AL,3 ; MAKE DETERMINED/ESTABLISHED
1226
1227 04A4 AL_SET: AND #DSK_STATE[DI],NOT DRV_DET+FMT_CAPA+TRK_CAPA ; CLEAR DRIVE
1228 04A4 80 A5 0090 R F8 OR #DSK_STATE[DI],AL ; REPLACE WITH COMPATIBLE MODE
1229 04A9 08 85 0090 R
1230 04AD C3 XN_OUT: RET
1231 04AD C3 XLAT_OLD ENDP
1232 04AE
    
```

```

1233 PAGE
1234 -----
1235 ; RD_WR_VF
1236 ; COMMON READ, WRITE AND VERIFY;
1237 ; MAIN LOOP FOR STATE RETRIES.
1238 ;
1239 ; ON ENTRY: AH: READ/WRITE/VERIFY NEC PARAMETER
1240 ; AL: READ/WRITE/VERIFY DMA PARAMETER
1241 ;
1242 ; ON EXIT: 0DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
1243 ;-----
1244 RD_WR_VF PROC NEAR
1245 04AE 50 PUSH AX ; SAVE DMA, NEC PARAMETERS
1246 04AF EB 040F R CALL XLAT_NEW ; TRANSLATE STATE TO PRESENT ARCH.
1247 04B2 EB 056A R CALL SETUP_STATE ; INITIALIZE START AND END RATE
1248 04B5 5B POP AX ; RESTORE READ/WRITE/VERIFY
1249
1250 04B6 DO_AGAIN:
1251 04B6 50 PUSH AX ; SAVE READ/WRITE/VERIFY PARAMETER
1252 04B7 EB 05E9 R CALL MED_CHANGE ; MEDIA CHANGE AND RESET IF CHANGED
1253 04BA 5B POP AX ; RESTORE READ/WRITE/VERIFY
1254 ; JNC RWV_END ; MEDIA CHANGE ERROR OR TIME-OUT
1255 04BB 73 03 JNC RWV_END
1256 04BD E9 055B R JMP RWV_END
1257 04C0 RWV:
1258 04C0 50 PUSH AX ; SAVE READ/WRITE/VERIFY PARAMETER
1259
1260 04C1 8A B5 0090 R MOV DH,0DSK_STATE[D1] ; GET RATE STATE OF THIS DRIVE
1261 04C5 80 E6 C0 AND DH,RATE_MSK ; KEEP ONLY RATE
1262 04C8 EB 08EC R CALL CMOS_TYPE ; RETURN DRIVE TYPE IN (AL)
1263 04CB 72 45 JC RWV_ASSUME ; ERROR IN CMOS
1264 04CD 3C 01 CMP AL,1 ; 40 TRACK DRIVE?
1265 04CF 75 0B JNE RWV_1 ; NO, BYPASS CMOS VALIDITY CHECK
1266 04D1 F6 85 0090 R 01 TEST 0DSK_STATE[D1],TRK_CAPA ; CHECK FOR 40 TRACK DRIVE
1267 04D6 74 0F JZ RWV_2 ; YES, CMOS IS CORRECT
1268 04D8 B0 02 MOV AL,2 ; CHANGE TO 1.2 M
1269 04DA EB 0B JMP SHORT RWV_2 ; CONTINUE
1270 04DC RWV_1:
1271 04DC 72 09 JB RWV_2 ; NO DRIVE SPECIFIED, CONTINUE
1272 04DE F6 85 0090 R 01 TEST 0DSK_STATE[D1],TRK_CAPA ; IS IT REALLY 40 TRACK?
1273 04E3 75 02 JNZ RWV_2 ; NO, 80 TRACK
1274 04E5 B0 01 MOV AL,1 ; IT'S 40 TRACK, FIX CMOS VALUE
1275 04E7 RWV_2:
1276
1277 04E7 0A C0 OR AL,AL ; TEST FOR NO DRIVE
1278 04E9 74 28 JZ RWV_ASSUME ; ASSUME TYPE, USE MAX TRACK
1279 04EB EB 03B8 R CALL DR_TYPE_CHECK ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
1280 04EE 72 23 JC RWV_ASSUME ; TYPE NOT IN TABLE (BAD CMOS)
1281
1282 ;--- SEARCH FOR MEDIA/DRIVE PARAMETER TABLE
1283
1284 04F0 57 PUSH DI ; SAVE DRIVE #
1285 04F1 33 DB XOR BX,BX ; BX = INDEX TO DR_TYPE TABLE
1286 04F3 B9 0006 MOV CX,DR_CNT ; CX = LOOP COUNT
1287 04F6 RWV_DR_SEARCH:
1288 04F6 2E: 8A A7 0000 R MOV AH,CS:DR_TYPE[BX] ; GET DRIVE TYPE
1289 04FB 80 E4 7F AND AH,BITTOFF ; MASK OUT MSB
1290 04FE 3A C4 CMP AL,AH ; DRIVE TYPE MATCH?
1291 0500 75 0B JNE RWV_NXT_MD ; NO, CHECK NEXT DRIVE TYPE
1292 0502 RWV_DR_FND:
1293 0502 2E: 8B BF 0001 R MOV DI,WORD PTR CS:DR_TYPE[BX+1] ; DI = MEDIA/DRIVE PARAMETER TABLE
1294 0507 RWV_MD_SEARCH:
1295 0507 2E: 3A 75 0C CMP DH,CS:[DI],MD_RATE ; MATCH?
1296 050B 74 16 JE RWV_MD_FND ; YES, GO GET 1ST SPECIFY BYTE
1297 050D RWV_NXT_MD:
1298 050D 83 C3 03 ADD BX,3 ; CHECK NEXT DRIVE TYPE
1299 0510 F2 E4 LOOP RWV_DR_SEARCH ; RESTORE DRIVE #
1300 0512 5F POP DI
1301
1302 ;--- ASSUME PRIMARY DRIVE IS INSTALLED AS SHIPPED
1303
1304 0513 RWV_ASSUME:
1305 0513 BB 0012 R MOV BX,OFFSET MD_TBL1 ; POINT TO 40 TK 250 KBS
1306 0516 75 85 0090 R 01 TEST 0DSK_STATE[D1],TRK_CAPA ; TEST FOR 80 TRACK
1307 051B 74 09 JZ RWV_MD_FND1 ; MUST BE 40 TRACK
1308 051D BB 002C R MOV BX,OFFSET MD_TBL3 ; POINT TO 80 TK 500 KBS
1309 0520 EB 04 90 JMP RWV_MD_FND1 ; GO SET SPECIFY PARAMETERS
1310
1311 ;--- CS:BX POINTS TO MEDIA/DRIVE PARAMETER TABLE
1312
1313 0523 RWV_MD_FND:
1314
1315 0523 BB DF MOV BX,D1 ; BX = MEDIA/DRIVE PARAMETER TABLE
1316 0525 5F POP DI ; RESTORE DRIVE #
1317 0526 RWV_MD_FND1:
1318
1319 ;--- SEND THE SPECIFY COMMAND TO THE CONTROLLER
1320
1321 0526 EB 03F5 R CALL SEND_SPEC_MD ; SEND SPEC MD
1322 0529 EB 063D R CALL CHK_LASRATE ; ZF=1 ATTEMPT RATE IS SAME AS LAST RATE
1323 052C 74 03 JZ RWV_DBL ; YES, SKIP SEND RATE COMMAND
1324 052E EB 0624 R CALL SEND_RATE ; SEND DATA RATE TO NEC
1325
1326 RWV_DBL:
1327 0531 53 PUSH BX ; SAVE MEDIA/DRIVE PARAM ADDRESS
1328 0532 EB 006E R CALL SETUP_DBL ; CHECK FOR DOUBLE STEP
1329 0535 5B POP BX ; RESTORE ADDRESS
1330 0536 72 1A JC CHK_RET ; ERROR FROM READ ID, POSSIBLE RETRY
1331 0538 5B POP AX ; RESTORE NEC, DMA COMMAND
1332 0539 50 PUSH AX ; SAVE NEC COMMAND
1333 053A 53 PUSH BX ; SAVE MEDIA/DRIVE PARAM ADDRESS
1334 053B EB 064D R CALL DMA_SETUP ; SET UP THE DMA
1335 053E 5B POP BX ; RESTORE ADDRESS
1336 053F 58 POP AX ; RESTORE NEC COMMAND
1337 0540 72 1F JC RWV_BAC ; CHECK FOR DMA BOUNDARY ERROR
1338 0542 50 PUSH AX ; SAVE NEC COMMAND
1339 0543 53 PUSH BX ; SAVE MEDIA/DRIVE PARAM ADDRESS
1340 0544 EB 0700 R CALL NEC_INIT ; INITIALIZE NEC
1341 0547 5B POP BX ; RESTORE ADDRESS
1342 0548 72 08 JC CHK_RET ; ERROR - EXIT
1343 054A EB 0725 R CALL RWV_COM ; DP CODE COMMON TO READ/WRITE/VERIFY
1344 054D 72 03 JC CHK_RET ; ERROR - EXIT
1345 054F EB 075B R CALL NEC_TERM ; TERMINATE, GET STATUS, ETC.

```

SECTION 5

```

1347
1348 0552          CHK_RET:
1349 0552 EB 07E5 R    CALL    RETRY          ; CHECK FOR SETUP RETRY
1350 0555 58        POP     AX              ; RESTORE READ/WRITE/VERIFY PARAMETER
1351 0556 73 03     JNC    RWV_END        ; CY = 0 NO RETRY
1352 0558 E9 04B6 R    JMP     DO_AGAIN       ; CY = 1 MEANS RETRY
1353
1354 055B          RWV_END:
1355 055B EB 07AD R    CALL    DSTATE        ; ESTABLISH STATE IF SUCCESSFUL
1356 055E EB 0827 R    CALL    NUM_TRANS     ; AL = NUMBER TRANSFERRED
1357
1358 0561          RWV_BAC:
1359 0561 80        PUSH   AX              ; BAD DMA ERROR ENTRY
1360 0562 EB 0435 R    CALL    XLAT_OLD     ; SAVE NUMBER TRANSFERRED
1361 0565 58        POP     AX              ; TRANSLATE STATE TO COMPATIBLE MODE
1362 0566 EB 0854 R    CALL    SETUP_END    ; RESTORE NUMBER TRANSFERRED
1363 0569 C3        RET     ; VARIOUS CLEANUPS
1364 056A
1365
1366
1367
-----
; SETUP_STATE: INITIALIZES START AND END RATES.
;
1368 056A          SETUP_STATE: PROC NEAR
1369 056A F6 85 0090 R 10 TEST  #DSK_STATE[D1],MED_DET ; MEDIA DETERMINED ?
1370 056F 75 29     JNZ    JIC            ; NO STATES IF DETERMINED
1371 0571 86 0040     MOV     AX,AX         ; AH = START RATE, AL = END RATE
1372 0574 F6 85 0090 R 04 TEST  #DSK_STATE[D1],DRV_DET ; DRIVE ?
1373 0579 74 0A     JZ     AX_SET        ; DO NOT KNOW DRIVE
1374 057B F6 85 0090 R 02 TEST  #DSK_STATE[D1],FMT_CAPA ; MULTI-RATE ?
1375 0580 75 03     JNZ    AX_SET        ; JUMP IF YES
1376 0582 B6 8080     MOV     AX,AX        ; START & END RATE = 250 FOR 360 DRIVE
1377
1378 0585          AX_SET:
1379 0585 80 A5 0090 R 1F AND   #DSK_STATE[D1],NOT RATE_MSK+DBL_STEP ; TURN OFF THE RATE
1380 058A 08 A5 0090 R 0R OR    #DSK_STATE[D1],AH ; RATE FIRST TO TRY
1381 058E 80 26 008B R F3 AND   #LAstrate,NOT STRT_MSK ; ERASE LAST TO TRY RATE BITS
1382 0593 C8 04 04     OR     AL,4          ; TO OPERATION LAST RATE LOCATION
1383 0594 08 06 008B R 0R OR    #LAstrate,AL ; LAST RATE
1384 059A
1385 059A C3        RET     SETUP_STATE ; ENDP
1386 059B
1387
1388
-----
; FMT_INIT: ESTABLISH STATE IF UNESTABLISHED AT FORMAT TIME.
;
1389 059B          FMT_INIT: PROC NEAR
1390 059B F6 85 0090 R 10 TEST  #DSK_STATE[D1],MED_DET ; IS MEDIA ESTABLISHED
1391 059B F6 85 0090 R 10 TEST  #DSK_STATE[D1],MED_DET ; IS MEDIA ESTABLISHED
1392 05A0 75 42     JNZ    FI_ODT        ; IF SO RETURN
1393 05A2 EB 08EC R    CALL    CMDS_TYPE    ; RETURN DRIVE TYPE IN AL
1394 05A5 72 3E     JC     CL_DRV        ; ERROR IN CMDS ASSUME NO DRIVE
1395 05A7 FE C8     DEC    AL            ; MAKE ZERO ORIGIN
1396 05A9 78 3A     JS     CL_DRV        ; NO DRIVE IF AL 0
1397 05AB EA A5 0090 R 08 MOV     AH,#DSK_STATE[D1] ; AH = CURRENT STATE
1398 05AF 80 E4 0F     AND   AH,NOT MED_DET+DBL_STEP+RATE_MSK ; CLEAR
1399 05B2 0A C0     OR    AL,AL         ; CHECK FOR 360
1400 05B4 75 05     JNZ    N_360        ; IF 360 WILL BE 0
1401 05B5 80 C8 90     OR    AH,MED_DET+RATE_250 ; ESTABLISH MEDIA
1402 05B9 EB 25     JMP     SHORT_SKP_STATE ; SKIP OTHER STATE PROCESSING
1403
1404 05BB          N_360:
1405 05BB DEC    AL            ; 1.2 M DRIVE
1406 05BD 75 05     JNZ    FI_RATE1     ; JUMP IF NOT
1407 05BF 80 CC 10     OR    AH,MED_DET+RATE_500 ; SET FORMAT RATE
1408 05C2 EB 1C     JMP     SHORT_SKP_STATE ; SKIP OTHER STATE PROCESSING
1409
1410 05C4          N_12:
1411 05C4 DEC    AL            ; CHECK FOR TYPE 3
1412 05C4 75 0F     JNZ    N_720        ; JUMP IF NOT
1413 05C6 F6 C4 04     TEST  AH,DRV_DET    ; IS DRIVE DETERMINED
1414 05CB 74 10     JZ     ISNT_12      ; TREAT AS NON 1.2 DRIVE
1415 05CD F6 C8 02     TEST  AH,FRT_CAPA   ; IS 1-2?
1416 05D0 74 0B     JZ     ISNT_12      ; JUMP IF NOT
1417 05D2 80 CC 50     OR    AH,MED_DET+RATE_300 ; RATE 300
1418 05D5 EB 09     JMP     SHORT_SKP_STATE ; CONTINUE
1419
1420 05D7          N_720:
1421 05D7 DEC    AL            ; CHECK FOR TYPE 4
1422 05D9 75 0A     JNZ    CL_DRY       ; NO DRIVE, CMOS BAD
1423 05DB EB E2     JMP     SHORT_FI_RATE
1424
1425 05DD          ISNT_12:
1426 05DD OR    AH,MED_DET+RATE_250 ; MUST BE RATE 250
1427
1428 05E0          SKP_STATE:
1429 05E0 MOV     #DSK_STATE[D1],AH ; STORE AWAY
1430
1431 05E4          FI_OUT:
1432 05E4 RET
1433
1434 05E5          CL_DRV:
1435 05E5 XOR    AH,AH         ; CLEAR STATE
1436 05E7 EB F7     JMP     SHORT_SKP_STATE ; SAVE IT
1437 05E9
1438
-----
; MED_CHANGE
; CHECKS FOR MEDIA CHANGE, RESETS MEDIA CHANGE,
; CHECKS MEDIA CHANGE AGAIN.
;
1439
1440
1441
1442
1443
1444
1445
-----
; ON EXIT: CY = 1 MEANS MEDIA CHANGE OR TIMEOUT
; #DSKETTE_STATUS = ERROR CODE
;
1446 05E9          MED_CHANGE: PROC NEAR
1447 05E9 E8 0B28 R    CALL    READ_DSKCHNG ; READ DISK CHANGE LINE STATE
1448 05EC 74 3A     JZ     MC_ODT        ; BYPASS HANDLING DISK CHANGE LINE
1449 05EE 80 A5 0090 R EF AND   #DSK_STATE[D1],NOT MED_DET ; CLEAR STATE FOR THIS DRIVE
1450
1451
1452
-----
; 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).
;
1453
1454
1455 05F3 8B CF     MOV     CX,D1        ; CL = DRIVE #
1456 05F5 80 01     MOV     AL,1         ; MOTOR ON BIT MASK
1457 05F7 D2 E0     SHL    AL,CL         ; TO APPROPRIATE POSITION
1458 05F9 F6 D0     NOT    AL            ; KEEP ALL BUT MOTOR ON
1459 05FB 7A        CLD                 ; NO INTERRUPTS
1460 05FC 20 06 003F R AND   #MOTOR_STATUS,AL ; TURN MOTOR OFF INDICATOR

```



```

1461 0600 FB          STI          ; INTERRUPTS ENABLED
1462 0601 E8 091A R   CALL      MOTOR_ON      ; TURN MOTOR ON
1463
1464                ;----- THIS SEQUENCE OF SEEKS IS USED TO RESET DISKETTE CHANGE SIGNAL
1465
1466 0604 E8 00E7 R   CALL      DISK RESET    ; RESET NEC
1467 0605 B5 01      MOV       CH,07H    ; MOVE TO CYLINDER 1
1468 0609 E8 0A24 R   CALL      SEEK          ; ISSUE SEEK
1469 060C 32 ED      XOR       CH,CH    ; MOVE TO CYLINDER 0
1470 060E E8 0A24 R   CALL      SEEK          ; ISSUE SEEK
1471 0611 C6 06 0041 R 06 MOV       @DSKETTE_STATUS,MEDIA_CHANGE ; STORE IN STATUS
1472
1473 0616 E8 0B28 R   OK1: CALL    READ_DSKCHNG ; CHECK MEDIA CHANGED AGAIN
1474 0619 74 05      OK2: JZ          ; IF ACTIVE, NO DISKETTE, TIMEOUT
1475
1476 061B C6 06 0041 R 80 OK4: MOV       @DSKETTE_STATUS,TIME_OUT; TIMEOUT IF DRIVE EMPTY
1477
1478 0620 F9          OK2: STC          ; MEDIA CHANGED, SET CY
1479 0621 C3          RET
1480 0622             MC_OUT: CLC
1481 0622 F6          RET          ; NO MEDIA CHANGED, CLEAR CY
1482 0623 C3
1483 0624             MED_CHANGE ENDP
1484
1485             ;-----
1486             ; SEND_RATE
1487             ; SENDS DATA RATE COMMAND TO NEC
1488             ; ON ENTRY:  DI = DRIVE #
1489             ; ON EXIT:   NONE
1490             ; REGISTERS ALTERED: DX
1491             ;-----
1491 0624             SEND_RATE PROC NEAR
1492
1493 0624 80          PUSH     AX          ; SAVE REG.
1494 0625 80 26 008B R 3F AND     @LAstrate,NOT SEND_MSK ; ELSE CLEAR LAST RATE ATTEMPTED
1495 062A 8A 85 0090 R MOV     AL,@DSK_STATE[DI]    ; GET RATE STATE OF THIS DRIVE
1496 062E 24 C0      AND     AL,SEND_MSK        ; KEEP ONLY RATE BITS
1497 0630 06 06 008B R OR      @LAstrate,AL        ; SAVE NEW RATE FOR NEXT CHECK
1498 0634 C0 C0 02   ROL     AL,2              ; MOVE TO BIT OUTPUT POSITIONS
1499 0637 BA 03F7   MOV     DX,03F7H          ; OUTPUT NEW DATA RATE
1500 063A EE          OUT     DX,AL
1501
1502 063B 5B          POP     AX          ; RESTORE REG.
1503 063C C3          RET
1504 063D             SEND_RATE ENDP
1505
1506             ;-----
1507             ; CHK_LAstrate
1508             ; CHECK PREVIOUS DATA RATE SENT TO THE CONTROLLER.
1509             ; ON ENTRY:  DI = DRIVE #
1510             ; ON EXIT:   ZF = 1 DATA RATE IS THE SAME AS LAST RATE SENT TO NEC ;
1511             ;             ZF = 0 DATA RATE IS DIFFERENT FROM LAST RATE ;
1512             ; REGISTERS ALTERED: NONE
1513             ;-----
1514             ;
1515             ;
1516 063D             CHK_LAstrate PROC NEAR
1517 063D 50          PUSH     AX          ; SAVE REG
1518 063E 8A 26 008B R MOV     AH,@LAstrate        ; GET LAST DATA RATE SELECTED
1519 0642 8A 85 0090 R MOV     AL,@DSK_STATE[DI]    ; GET RATE STATE OF THIS DRIVE
1520 0646 26 C0C0    AND     AX,SEND_MSK*8       ; KEEP ONLY RATE BITS OF BOTH
1521 0649 3A C4      CMP     AL,AH              ; COMPARE TO PREVIOUSLY TRIED
1522
1523 064B 5B          POP     AX          ; RESTORE REG.
1524 064C C3          RET
1525 064D             CHK_LAstrate ENDP
1526
1527 SUBTTL (DSK3.ASM)
    
```

SECTION 5

```

1528                                     PAGE
1529                                     -----
1530 | DMA_SETUP                           |
1531 | THIS ROUTINE SETS UP THE DMA FOR READ/WRITE/VERIFY |
1532 | OPERATIONS.                         |
1533 |                                     |
1534 | ON ENTRY:   AL = DMA COMMAND        |
1535 |                                     |
1536 | ON EXIT:    #DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION |
1537 |-----|
1538 064D DMA_SETUP PROC NEAR
1539 064E FA CL I DMA+12,AL ; DISABLE INTERRUPTS DURING DMA SET-UP
1540 064E E6 OC OUT DMA+12,AL ; SET THE FIRST/LAST F/F
1541 0650 EB 00 JMP $+2 ; WAIT FOR I/O
1542 0652 E6 0B OUT DMA+11,AL ; OUTPUT THE MODE BYTE
1543 0654 3C 42 CMP AL,42H ; DMA VERIFY COMMAND
1544 0656 75 04 JNE NOT_VERF ; NO
1545 0658 33 C0 XOR AX,AX ; START ADDRESS
1546 065A EB 10 JMP SHORT J33
1547 065C
1548 065C 8C C0 NOT_VERF: MOV AX,ES ; GET THE ES VALUE
1549 065E C1 C0 04 ROL AX,4 ; ROTATE LEFT
1550 0661 8A E8 MOV CH,AL ; GET HIGHEST NIBBLE OF ES TO CH
1551 0663 24 F0 AND AL,11110000B ; ZERO THE LOW NIBBLE FROM SEGMENT
1552 0665 03 46 02 ADD AX,[BP+2] ; TEST FOR CARRY FROM ADDITION
1553 0668 73 02 JNC J33
1554 066A FE C5 J33: INC CH ; CARRY MEANS HIGH 4 BITS MUST BE INC
1555 066C
1556 066C 50 PUSH AX ; SAVE START ADDRESS
1557 066D E6 04 OUT DMA+4,AL ; OUTPUT LOW ADDRESS
1558 066F EB 00 JMP $+2 ; WAIT FOR I/O
1559 0671 8A C4 MOV AL,AH
1560 0673 E6 04 OUT DMA+4,AL ; OUTPUT HIGH ADDRESS
1561 0675 8A C5 MOV AL,CH ; GET HIGH 4 BITS
1562 0677 EB 00 JMP $+2 ; I/O WAIT STATE
1563 0679 24 0F AND AL,00001111B
1564 067B E6 01 OUT 081H,AL ; OUTPUT HIGH 4 BITS TO PAGE REGISTER
1565
1566 |----- DETERMINE COUNT
1567
1568 067D 8B C6 MOV AX,S1 ; AL = # OF SECTORS
1569 067F 86 C4 XCHG AL,AH ; AH = # OF SECTORS
1570 0681 2A C0 SUB AL,AL ; AL = 0, AX = # OF SECTORS * 256
1571 0683 D1 E8 SHR AX,1 ; AX = # SECTORS * 128
1572 0685 50 PUSH AX ; SAVE # OF SECTORS * 128
1573 0686 B2 03 MOV DL,3 ; GET BYTES/SECTOR PARAMETER
1574 0688 EB 0905 R CALL GET_PARM ;
1575 068B 8A CC MOV CL,AH ; SHIFT COUNT (0=128, 1=256 ETC)
1576 068D 58 POP AX ; AX = # OF SECTORS * 128
1577 068E D3 E0 SHL AX,CL ; SHIFT BY PARAMETER VALUE
1578 0690 48 DEC AX ; -1 FOR DMA VALUE
1579 0691 50 PUSH AX ; SAVE COUNT VALUE
1580 0692 E6 05 OUT DMA+5,AL ; LOW BYTE OF COUNT
1581 0694 EB 00 JMP $+2 ; WAIT FOR I/O
1582 0696 8A C4 MOV AL,AH ; HIGH BYTE OF COUNT
1583 0698 E6 05 OUT DMA+5,AL ; RE-ENABLE INTERRUPTS
1584 069A FB STI ; RECOVER COUNT VALUE
1585 069B 59 POP CX ; RECOVER ADDRESS VALUE
1586 069C 58 POP AX ;
1587 069D 03 C1 ADD AX,CX ; ADD, TEST FOR 64K OVERFLOW
1588 069F B0 02 MOV AL,2 ; MODE FOR B22
1589 06A1 EB 00 JMP $+2 ; WAIT FOR I/O
1590 06A3 E6 0A OUT DMA+10,AL ; INITIALIZE THE DISKETTE CHANNEL
1591
1592 06A5 73 05 JNC NO_BAD ; CHECK FOR ERROR
1593 06A7 C6 06 0041 R 09 MOV #DSKETTE_STATUS,DMA_BOUNDARY ; SET ERROR
1594
1595 06AC NO_BAD:
1596 06AC C3 RET ; CY SET BY ABOVE IF ERROR
1597 06AD DMA_SETUP ENDP
    
```

```

1598 PAGE
1599 -----
1600 ; FMTDMA_SET
1601 ; THIS ROUTINE SETS UP THE DMA CONTROLLER FOR A FORMAT
1602 ; OPERATION.
1603 ;
1604 ; ON ENTRY: NOTHING REQUIRED
1605 ;
1606 ; ON EXIT:  @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
1607 ;
1608 FMTDMA_SET PROC NEAR
1609 MOV AL,04AH ; WILL WRITE TO THE DISKETTE
1610 CL1 ; DISABLE INTERRUPTS DURING DMA SET-UP
1611 OUT DMA+12,AL ; SET THE FIRST/LAST F/F
1612 JMP $+2 ; WAIT FOR I/O
1613 OUT DMA+11,AL ; OUTPUT THE MODE BYTE
1614 ;
1615 MOV AX,ES ; GET THE ES VALUE
1616 ROL AX,4 ; ROTATE LEFT
1617 MOV CH,AL ; GET HIGHEST NIBBLE OF ES TO CH
1618 AND AL,11110000B ; ZERO THE LOW NIBBLE FROM SEGMENT
1619 ADD AX,[BP+2] ; TEST FOR CARRY FROM ADDITION
1620 JNC J33A ; J33A
1621 INC CH ; CARRY MEANS HIGH 4 BITS MUST BE INC
1622 J33A:
1623 PUSH AX ; SAVE START ADDRESS
1624 OUT DMA+4,AL ; OUTPUT LOW ADDRESS
1625 JMP $+2 ; WAIT FOR I/O
1626 MOV AL,AH ;
1627 OUT DMA+4,AL ; OUTPUT HIGH ADDRESS
1628 MOV AL,CH ; GET HIGH 4 BITS
1629 JMP $+2 ; I/O WAIT STATE
1630 AND AL,00001111B ;
1631 OUT 0B1H,AL ; OUTPUT HIGH 4 BITS TO PAGE REGISTER
1632 ;
1633 ;----- DETERMINE COUNT
1634 ;
1635 MOV DL,4 ; SECTORS/TRACK VALUE IN PARM TABLE
1636 CALL GET_PARM ;
1637 XCHG AL,AH ; AL = SECTORS/TRACK VALUE
1638 SUB AH,AH ; AX = SECTORS/TRACK VALUE
1639 SHL AX,2 ; AX = SEC/TRK * 4 (OFFSET FOR C.H.R.N)
1640 DEC AX ; -1 FOR DMA VALUE
1641 PUSH AX ; SAVE # OF BYTES TO BE TRANSFERRED
1642 OUT DMA+5,AL ; LOW BYTE OF COUNT
1643 JMP $+2 ; WAIT FOR I/O
1644 MOV AL,AH ; HIGH BYTE OF COUNT
1645 OUT DMA+5,AL ;
1646 STI ; RE-ENABLE INTERRUPTS
1647 POP CX ; RECOVER COUNT VALUE
1648 POP AX ; RECOVER ADDRESS VALUE
1649 ADD AX,CX ; ADD. TEST FOR 84K OVERFLOW
1650 MOV AL,2 ; MODE FOR 8237
1651 JMP $+2 ; WAIT FOR I/O
1652 OUT DMA+10,AL ; INITIALIZE THE DISKETTE CHANNEL
1653 ;
1654 JNC FMTDMA_OK ; CHECK FOR ERROR
1655 MOV @DSKETTE_STATUS,DMA_BOUNDARY ; SET ERROR
1656 ;
1657 FMTDMA_OK:
1658 RET ; CY SET BY ABOVE IF ERROR
1659 FMTDMA_SET ENDP
1660 ;-----
1661 ; NEC_INIT
1662 ; THIS ROUTINE SEEKS TO THE REQUESTED TRACK AND
1663 ; INITIALIZES THE NEC FOR THE READ/WRITE/VERIFY/FORMAT
1664 ; OPERATION.
1665 ;
1666 ; ON ENTRY: AH : NEC COMMAND TO BE PERFORMED
1667 ;
1668 ; ON EXIT:  @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
1669 ;
1670 NEC_INIT PROC NEAR
1671 PUSH AX ; SAVE NEC COMMAND
1672 CALL MOTOR_ON ; TURN MOTOR ON FOR SPECIFIC DRIVE
1673 ;
1674 ;----- DO THE SEEK OPERATION
1675 ;
1676 MOV CH,[BP+1] ; CH = TRACK #
1677 CALL SEEK ; MOVE TO CORRECT TRACK
1678 POP AX ; RECOVER COMMAND
1679 JC ER_1 ; ERROR ON SEEK
1680 MOV BX,OFFSET ER_1 ; LOAD ERROR ADDRESS
1681 PUSH BX ; PUSH NEC_OUT ERROR RETURN
1682 ;
1683 ;----- SEND OUT THE PARAMETERS TO THE CONTROLLER
1684 ;
1685 CALL NEC_OUTPUT ; OUTPUT THE OPERATION COMMAND
1686 MOV AX,S1 ; AH = HEAD #
1687 MOV BX,D1 ; BL = DRIVE #
1688 SAL AH,2 ; MOVE IT TO BIT 2
1689 AND AH,0000100B ; ISOLATE THAT BIT
1690 OR AH,BL ; OR IN THE DRIVE NUMBER
1691 CALL NEC_OUTPUT ; FALL THRU CY SET IF ERROR
1692 POP BX ; THROW AWAY ERROR RETURN
1693 ER_1:
1694 RET
1695 NEC_INIT ENDP
1696 ;-----
1697 ; RWV_COM
1698 ; THIS ROUTINE SENDS PARAMETERS TO THE NEC SPECIFIC
1699 ; TO THE READ/WRITE/VERIFY OPERATIONS.
1700 ;
1701 ; ON ENTRY:  CS:BX = ADDRESS OF MEDIA/DRIVE PARAMETER TABLE
1702 ; ON EXIT:  @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
1703 ;
1704 RWV_COM PROC NEAR
1705 MOV AX,OFFSET ER_2 ; LOAD ERROR ADDRESS
1706 PUSH AX ; PUSH NEC_OUT ERROR RETURN
1707 MOV AH,[BP+1] ; OUTPUT TRACK #
1708 CALL NEC_OUTPUT ;
1709 MOV AX,S1 ; OUTPUT HEAD #
1710 CALL NEC_OUTPUT ;
1711 MOV AH,[BP] ; OUTPUT SECTOR #
    
```

SECTION 5

```

1712 0737 E8 09F8 R      CALL    NEC_OUTPUT          ; BYTES/SECTOR PARAMETER FROM BLOCK
1713 073A B2 03          MOV     DL,3                ; TO THE NEC
1714 073C E8 0905 R      CALL    GET_PARM           ; OUTPUT TO CONTROLLER
1715 073F E8 09F8 R      CALL    NEC_OUTPUT          ; EOT PARAMETER FROM BLOCK
1716 0742 B2 04          MOV     DL,4                ; TO THE NEC
1717 0744 E8 0905 R      CALL    GET_PARM           ; OUTPUT TO CONTROLLER
1718 0747 E8 09F8 R      CALL    NEC_OUTPUT
1719
1720 074A 2E 8A 67 05     MOV     AH,CS:[BX],MD_GAP   ; GET GAP LENGTH
1721 074E E8 09F8 R      CALL    NEC_OUTPUT          ; DTL PARAMETER FROM BLOCK
1722 0751 B2 04          MOV     DL,4                ; TO THE NEC
1723 0753 E8 0905 R      CALL    GET_PARM           ; OUTPUT TO CONTROLLER
1724 0756 E8 09F8 R      CALL    NEC_OUTPUT          ; THROW AWAY ERROR EXIT
1725 0759 58             POP     AX
1726 075A             ER_2:
1727 075A C3             RET
1728 075B             RWV_COM ENDP
1729
1730             ; NEC_TERM
1731             ; THIS ROUTINE WAITS FOR THE OPERATION THEN ACCEPTS
1732             ; THE STATUS FROM THE NEC FOR THE READ/WRITE/VERIFY/
1733             ; FORMAT OPERATION.
1734             ;
1735             ; ON EXIT:  #DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION:
1736             ;-----
1737 075B             NEC_TERM PROC NEAR
1738
1739             ;----- LET THE OPERATION HAPPEN
1740
1741 075B 56             PUSH   SI                   ; SAVE HEAD #, # OF SECTORS
1742 075C E8 0AC1 R      CALL    WAIT_INT           ; WAIT FOR THE INTERRUPT
1743 075F 9C             PUSHF                      ;
1744 0760 E8 0AE9 R      CALL    RESULTS            ; GET THE NEC STATUS
1745 0763 72 45         JC     SET_END_POP         ;
1746 0765 9D             POPF                       ;
1747 0766 72 3A         JC     SET_END             ; LOOK FOR ERROR
1748
1749             ;----- CHECK THE RESULTS RETURNED BY THE CONTROLLER
1750
1751 0768 FC             CLD                        ; SET THE CORRECT DIRECTION
1752 0769 BE 0042 R      MOV     SI,OFFSET #NEC_STATUS ; POINT TO STATUS FIELD
1753 076C AC             LODS                      ; GET ST0
1754 076D 24 C0         AND     AL,11000000B       ; TEST FOR NORMAL TERMINATION
1755 076F 74 31         JZ     SET_END             ;
1756 0771 3C 40         CMP     AL,01000000B       ; TEST FOR ABNORMAL TERMINATION
1757 0773 75 27         JNB    J18                 ; NOT ABNORMAL, BAD NEC
1758
1759             ;----- ABNORMAL TERMINATION, FIND OUT WHY
1760
1761 0778 AC             LODS                      ; GET ST1
1762 0779 D0 E0         SAL    AL,1                ; TEST FOR EOT FOUND
1763 077B B4 04         MOV     AH,RECORD_NOT_FND
1764 077A 72 22         JC     J19                 ;
1765 077C C0 E0 02      SAL    AL,2                ;
1766 077F B4 10         MOV     AH,BAD_CRC
1767 0781 72 1B         JC     J19                 ;
1768 0783 D0 E0         SAL    AL,1                ; TEST FOR DMA OVERRUN
1769 0785 B4 08         MOV     AH,BAD_DMA
1770 0787 72 18         JC     J19                 ;
1771 0789 C0 E0 02      SAL    AL,2                ; TEST FOR RECORD NOT FOUND
1772 078C B4 04         MOV     AH,RECORD_NOT_FND
1773 078E 72 0E         JC     J19                 ;
1774 0790 D0 E0         SAL    AL,1                ;
1775 0792 B4 03         MOV     AH,WRITE_PROTECT   ; TEST FOR WRITE_PROTECT
1776 0794 72 08         JC     J19                 ;
1777 0796 D0 E0         SAL    AL,1                ; TEST MISSING ADDRESS MARK
1778 0798 B4 02         MOV     AH,BAD_ADDR_MARK
1779 079A 72 02         JC     J19
1780
1781             ;----- NEC MUST HAVE FAILED
1782 079C             J18:
1783 079C B4 20         MOV     AH,BAD_NEC
1784 079E             OR     #DSKETTE_STATUS,AH
1785 079E 08 26 0041 R  SET_END:
1786 07A2             CMP    #DSKETTE_STATUS,1   ; SET ERROR CONDITION
1787 07A2 80 3E 0041 R 01  CMC
1788 07A7 F5             POP     SI                  ; RESTORE HEAD #, # OF SECTORS
1789 07A8 5E             RET
1790 07A9 C3
1791
1792 07AA             SET_END_POP:
1793 07AA 9D             POPF
1794 07AB EB F5             JMP    SHORT SET_END
1795 07AD             NEC_TERM ENDP
1796
1797             ;----- ESTABLISH STATE UPON SUCCESSFUL OPERATION.
1798             ;-----
1799 07AD             DSTATE PROC NEAR
1800 07AD 80 3E 0041 R 00  CMP    #DSKETTE_STATUS,0   ; CHECK FOR ERROR
1801 07B2 75 30         JNZ    SETBAC              ; IF ERROR JUMP
1802 07B4 80 80 0090 R 10  OR     #DSK_STATE[D1],MED_DET ; NO ERROR, MARK MEDIA AS DETERMINED
1803 07B9 F4 E8 0090 R 04  TEST   #DSK_STATE[D1],DRV_DET ; DRIVE DETERMINED ?
1804 07BE 75 24         JNZ    SETBAC              ; IF DETERMINED NO TRY TO DETERMINE
1805 07C0 8A 85 0090 R      MOV     AL,#DSK_STATE[D1]   ; LOAD STATE
1806 07C4 24 C0         AND     AL,RATE_MSK         ; KEEP ONLY RATE
1807 07C6 3C 80         CMP     AL,RATE_250        ; RATE 250 ?
1808 07C8 75 16         JNE    M_12                ; NO, MUST BE 1.2M OR 1.44M DRV
1809
1810             ;--- CHECK IF IT IS 1.44M
1811
1812 07CA E8 08EC R      CALL    CMOS_TYPE          ; RETURN DRIVE TYPE IN (AL)
1813 07CD 72 10         JC     M_12                ; CMOS BAD
1814 07CF 3C 04         CMP     AL,04              ; 1.44MB DRIVE ?
1815 07D1 74 0C         JE     M_12                ; YES
1816 07D3             M_720:
1817 07D3 80 A5 0090 R FD  AND    #DSK_STATE[D1],NOT_FMT_CAPA ; TURN OFF FORMAT_CAPA
1818 07D8 80 8D 0090 R 04  OR     #DSK_STATE[D1],DRV_DET  ; MARK DRIVE DETERMINED
1819 07DD EB 05             JMP    SHORT SETBAC        ; BACK
1820
1821 07DF 80 8D 0090 R 06  M_12: OR     #DSK_STATE[D1],DRV_DET+_FMT_CAPA ; TURN ON DETERMINED & FMT_CAPA
1822
1823 07E4             SETBAC: RET
1824 07E4 C3
1825 07E5             DSTATE ENDP
    
```

```

1826 ;-----
1827 ; RETRY
1828 ; DETERMINES WHETHER A RETRY IS NECESSARY. IF RETRY IS
1829 ; REQUIRED THEN STATE INFORMATION IS UPDATED FOR RETRY.
1830 ;
1831 ; ON EXIT: CY = 1 FOR RETRY, CY = 0 FOR NO RETRY
1832 ;-----
1833 07E5      PROC    NEAR
1834 07E5 80 3E 0041 R 00  CMP    #DSKETTE_STATUS,0 ; GET STATUS OF OPERATION
1835 07EA 74 39          JZ     NO_RETRY           ; SUCCESSFUL OPERATION
1836 07EC 80 3E 0041 R 80  CMP    #DSKETTE_STATUS,TIME_OUT ; IF TIME OUT NO RETRY
1837 07F1 74 32          JZ     NO_RETRY           ;
1838 07F3 8A A5 0090 R   MOV    AH,#DSK_STATE[D1] ; GET MEDIA STATE OF DRIVE
1839 07F7 F6 C4 10     TEST   AH,MED_DET        ; ESTABLISHED/DETERMINED ?
1840 07FA 75 29          JNZ   NO_RETRY           ; IF ESTABLISHED STATE THEN TRUE ERROR
1841 07FC 80 E4 C0      AND    AH,RATE_MSK       ; ISOLATE RATE
1842 07FF 8A 2E 00BB R  MOV    CH,#LASTRATE     ; GET START OPERATION STATE
1843 0803 C0 C5 04      ROL    CH,4              ; TO CORRESPONDING BITS
1844 0806 80 E5 C0      AND    CH,RATE_MSK       ; ISOLATE RATE BITS
1845 0809 3A EC         CMP    CH,AH             ; ALL RATES TRIED
1846 080B 74 18          JE     NO_RETRY           ; IF YES, THEN TRUE ERROR
1847 ;
1848 ; SETUP STATE INDICATOR FOR RETRY ATTEMPT TO NEXT RATE
1849 ; 00000000B (500) -> 10000000B (250)
1850 ; 10000000B (250) -> 01000000B (300)
1851 ; 01000000B (300) -> 00000000B (500)
1852 ;
1853 080D 80 FC 01      CMP    AH,RATE_500+1    ; SET CY FOR RATE 500
1854 0810 DD DC         RCR    AH,1             ; TO NEXT STATE
1855 0812 80 E4 C0      AND    AH,RATE_MSK       ; KEEP ONLY RATE BITS
1856 0815 80 A5 0090 R 1F  AND    #DSK_STATE[D1],NOT RATE_MSK+DBL_STEP ; RATE, DBL STEP OFF
1857 081A 08 A5 0090 R   OR     #DSK_STATE[D1],AH ; TURN ON NEW RATE
1858 081E C6 06 0041 R 00  MOV    #DSKETTE_STATUS,0 ; RESET STATUS FOR RETRY
1859 0823 F9           STC                    ; SET CARRY FOR RETRY
1860 0824 C3           RET                    ; RETRY RETURN
1861 ;
1862 0825      NO_RETRY:
1863 0825 F8           CLC                    ; CLEAR CARRY NO RETRY
1864 0826 C3           RET                    ; NO RETRY RETURN
1865 0827 ;
1866 ;-----
1867 ; NUM_TRANS
1868 ; THIS ROUTINE CALCULATES THE NUMBER OF SECTORS THAT
1869 ; WERE ACTUALLY TRANSFERRED TO/FROM THE DISKETTE.
1870 ;
1871 ; ON ENTRY: [BP+1] = TRACK
1872 ;           [SI+1] = HEAD
1873 ;           [BP] = START SECTOR
1874 ;
1875 ; ON EXIT: AL = NUMBER ACTUALLY TRANSFERRED
1876 ;-----
1877 0827      PROC    NEAR
1878 0827 32 C0      XOR    AL,AL            ; CLEAR FOR ERROR
1879 0829 80 3E 0041 R 00  CMP    #DSKETTE_STATUS,0 ; CHECK FOR ERROR
1880 082E 75 23          JNZ   NT_OUT           ; IF ERROR 0 TRANSFERRED
1881 0830 B2 04          MOV    DL,4            ; SECTORS/TRACK OFFSET TO DL
1882 0832 E8 0905 R   CALL   GET_PARM        ; AH = SECTORS/TRACK
1883 0835 8A 1E 0047 R   MOV    BL,#NEC_STATUS+5 ; GET ENDING SECTOR
1884 0839 B8 CE       MOV    CX,S1           ; CH = HEAD # STARTED
1885 083B 3A 2E 0046 R  CMP    CH,#NEC_STATUS+4 ; GET HEAD ENDED UP ON
1886 083F 75 0B          JNZ   DIF_HD          ; IF ON SAME HEAD, THEN NO ADJUST
1887 ;
1888 0841 8A 2E 0045 R   MOV    CH,#NEC_STATUS+3 ; GET TRACK ENDED UP ON
1889 0845 3A 6E 01      CMP    CH,[BP+1]        ; IS IT ASKED FOR TRACK
1890 0848 74 04          JZ     SAME_TRK        ; IF SAME TRACK NO INCREASE
1891 ;
1892 084A 02 DC         ADD    BL,AH           ; ADD SECTORS/TRACK
1893 084C 02 DC         ADD    BL,AH           ; ADD SECTORS/TRACK
1894 084E           SAME_TRK:
1895 084E           SUB    BL,[BP]         ; SUBTRACT START FROM END
1896 084E 2A 5E 00     MOV    AL,BL           ; TO AL
1897 0851 8A C3         MOV    AL,BL
1898 ;
1899 0853      NT_OUT:
1900 0853 C3           RET
1901 0854 ;
1902 ;-----
1903 ; SETUP_END
1904 ; RESTORES #MOTOR COUNT TO PARAMETER PROVIDED IN TABLE
1905 ; AND LOADS #DSKETTE_STATUS TO AH, AND SETS CY.
1906 ;
1907 ; ON EXIT: AH, #DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
1908 ;-----
1909 0854      PROC    NEAR
1910 0854 B2 02      MOV    DL,2            ; GET THE MOTOR WAIT PARAMETER
1911 0856 50          PUSH   AX              ; SAVE NUMBER TRANSFERRED
1912 0857 E8 0905 R   CALL   GET_PARM        ;
1913 085A 88 26 0040 R  MOV    #MOTOR_COUNT,AH ; STORE UPON RETURN
1914 085E 58          POP     AX             ; RESTORE NUMBER TRANSFERRED
1915 085F 8A 26 0041 R  MOV    AH,#DSKETTE_STATUS ; GET STATUS OF OPERATION
1916 0863 0A E4       OR     AH,AH           ; CHECK FOR ERROR
1917 0865 74 02       JZ     NUN_ERR         ; NO ERROR
1918 0867 32 C0      XOR    AL,AL           ; CLEAR NUMBER RETURNED
1919 ;
1920 0869      NUN_ERR:
1921 0869 80 FC 01      CMP    AH,1            ; SET THE CARRY FLAG TO INDICATE
1922 086C F8          CMC                    ; SUCCESS OR FAILURE
1923 086D C3           RET
1924 086E ;
    
```

SECTION 5

```

1925 PAGE
1926 -----
1927 ; SETUP_DBL
1928 ; CHECK DOUBLE STEP.
1929 ; ON ENTRY:
1930 ; DI = DRIVE
1931 ; ON EXIT: CY = 1 MEANS ERROR
1932 -----
1933 086E PROC NEAR
1934 086E 8A A5 0090 R MOV AH,#DSK_STATE[DI] ; ACCESS STATE
1935 0872 F8 C4 10 TEST AH,#MED_DET ; ESTABLISHED STATE ?
1936 0875 75 5E JNZ NO_DBL ; IF ESTABLISHED THEN DOUBLE DONE
1937
1938 ;----- CHECK FOR TRACK 0 TO SPEED UP ACKNOWLEDGE OF UNFORMATTED DISKETTE
1939
1940 0877 C6 06 003E R 00 MOV #SEEK_STATUS,0 ; SET RECALIBRATE REQUIRED ON ALL DRIVES
1941 087C E8 091A R CALL MOTOR_ON ; ENSURE MOTOR STAY ON
1942 087F B5 00 MOV CH,0 ; LOAD TRACK 0
1943 0881 E8 0A24 R CALL SEEK ; SEEK TO TRACK 0
1944 0884 E8 08D7 R CALL READ_ID ; READ ID FUNCTION
1945 0887 72 37 JC SD_ERR ; IF ERROR NO TRACK 0
1946
1947 ;----- INITIALIZE START AND MAX TRACKS (TIMES 2 FOR BOTH HEADS)
1948
1949 0889 B9 0480 MOV CX,0480H ; START, MAX TRACKS
1950 088C F8 85 0090 R 01 TEST #DSK_STATE[DI],TRK_CAPA ; TEST FOR 80 TRACK CAPABILITY
1951 0891 74 02 CNT_OK JZ ; IF NOT COUNT IS SETUP
1952 0893 B1 A0 MOV CL,0A0H ; MAXIMUM TRACK 1.2 MB
1953
1954 ; ATTEMPT READ ID OF ALL TRACKS, ALL HEADS UNTIL SUCCESS; UPON SUCCESS,
1955 ; MUST SEE IF ASKED FOR TRACK IN SINGLE STEP MODE = TRACK ID READ; IF NOT
1956 ; THEN SET DOUBLE STEP ON.
1957
1958 0895 C6 06 0040 R FF CNT_OK: MOV #MOTOR_COUNT,OFFH ; ENSURE MOTOR STAYS ON FOR OPERATION
1959 089A 51 PUSH CX ; SAVE TRACK, COUNT
1960 089B C6 06 0041 R 00 MOV #DSKETTE_STATUS,0 ; CLEAR STATUS, EXPECT ERRORS
1961 08A0 33 C0 XOR AX,AX ; CLEAR AX
1962 08A2 D0 ED SHR CH,1 ; HALVE TRACK, CY = HEAD
1963 08A4 C0 D0 03 RCL AL,3 ; AX = HEAD IN CORRECT BIT
1964 08A7 50 PUSH AX ; SAVE HEAD
1965 08A8 E8 0A24 R CALL SEEK ; SEEK TO TRACK
1966 08AB 58 POP AX ; RESTORE HEAD
1967 08AC 0B F8 OR DI,AX ; DI = HEAD OR'ED DRIVE
1968 08AE E8 08D7 R CALL READ_ID ; READ ID HEAD 0
1969 08B1 9C PUSHF ; SAVE RETURN FROM READ_ID
1970 08B2 81 E7 00FB AND DI,11111011B ; TURN OFF HEAD 1 BIT
1971 08B6 9D POPF ; RESTORE ERROR RETURN
1972 08B7 59 POP CX ; RESTORE COUNT
1973 08B8 73 08 JNC DO_CHK ; IF OK, ASKED = RETURNED TRACK ?
1974 08BA FE C5 INC CH ; INC FOR NEXT TRACK
1975 08BC 3A E9 CMP CH,CL ; REACHED MAXIMUM YET
1976 08BE 75 D5 JNZ CNT_OK ; CONTINUE TILL ALL TRIED
1977
1978 ;----- FALL THRU, READ ID FAILED FOR ALL TRACKS
1979
1980 08C0 F9 SD_ERR: STC ; SET CARRY FOR ERROR
1981 08C1 C3 RET ; SETUP_DBL ERROR EXIT
1982
1983 08C2 8A 0E 0045 R DO_CHK: MOV CL,#NEC_STATUS+3 ; LOAD RETURNED TRACK
1984 08C6 88 8D 0094 R MOV #DSK_TRK[DI],CL ; STORE TRACK NUMBER
1985 08CA D0 ED SHR CH,1 ; HALVE TRACK
1986 08CC 3A E9 CMP CH,CL ; IS IT THE SAME AS ASKED FOR TRACK
1987 08CE 74 08 JZ NO_DBL ; IF SAME THEN NO DOUBLE STEP
1988 08D0 80 8D 0090 R 20 OR #DSK_STATE[DI],DBL_STEP ; TURN ON DOUBLE STEP REQUIRED
1989
1990 08D8 F8 NO_DBL: CLC ; CLEAR ERROR FLAG
1991 08D9 C3 RET
1992 08D7 ENDP
1993
1994 ;----- READ ID FUNCTION.
1995 ; ON ENTRY: DI = BIT 2 = HEAD; BITS 1,0 = DRIVE
1996 ; ON EXIT: DI = BIT 2 IS RESET, BITS 1,0 = DRIVE
1997 ; #DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
1998 ;-----
2000 READ_ID PROC NEAR
2001 08D7 MOV AX,OFFSET ER_3 ; MOVE NEC OUTPUT ERROR ADDRESS
2002 08D8 50 PUSH AX
2003 08DA 50 MOV AH,4AH ; READ ID COMMAND
2004 08DB B4 4A CALL NEC_OUTPUT ; TO CONTROLLER
2005 08DD E8 09FB R MOV AX,DI ; DRIVE # TO AH, HEAD 0
2006 08E0 8B C7 MOV AH,AL
2007 08E2 BA E0 CALL NEC_OUTPUT ; TO CONTROLLER
2008 08E4 E8 09FB R CALL NEC_TERM ; WAIT FOR OPERATION, GET STATUS
2009 08E7 E8 075D R POP AX ; THROW AWAY ERROR ADDRESS
2010 08EA 58
2011 08EB ER_3: RET
2012 08EC C3 READ_ID ENDP
2013
2014 ;----- CMOS_TYPE
2015 ; RETURNS DISKETTE TYPE FROM CMOS
2016 ; ON ENTRY: DI = DRIVE #
2017 ; ON EXIT: AL = TYPE; CY REFLECTS STATUS
2018 ;-----
2020 CMOS_TYPE PROC NEAR
2021 08EC MOV AL,CMOS_DIAG ; CMOS DIAGNOSTIC STATUS BYTE ADDRESS
2022 08ED CALL CMOS_READ ; GET CMOS STATUS
2023 08EE B0 0E TEST AL,BAD_BAT+BAD_CKSUM ; BATTERY GOOD AND CHECKSUM VALID ?
2024 08EF A8 C0 STC ; SET CY = 1 INDICATING ERROR FOR RETURN
2025 08F1 A8 C0 JNZ BAD_CM ; ERROR IF EITHER BIT ON
2026 08F3 F9
2027 08F4 75 0E
2028
2029 08F6 B0 10 MOV AL,CMOS_DISKETTE ; ADDRESS OF DISKETTE BYTE IN CMOS
2030 08F8 E8 0000 R CALL CMOS_READ ; GET DISKETTE BYTE
2031 08FB 0B FF OR DI,DI ; SEE WHICH DRIVE IN QUESTION
2032 08FD 75 03 JNZ TB ; IF DRIVE 1, DATA IN LOW NIBBLE
2033 08FF C0 C8 04 ROR AL,4 ; EXCHANGE NIBBLES IF SECOND DRIVE
2034 0902
2035 0902 24 0F TB: AND AL,00FH ; KEEP ONLY DRIVE DATA, RESET CY = 0
2036 0904 BAD_CM:
2037 0904 C3 RET
2038 0905 CMOS_TYPE ENDP
    
```

```

2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052 0905
2053 0905 IE
2054 0906 56
2055 0907 2B C0
2056 0909 8E D8
2057 090B 81 D3
2058 090D 2A FF
2059
2060 090F C5 36 0078 R
2061 0913 8A 20
2062 0915 81 D3
2063 0917 5E
2064 0918 1F
2065 0919 C3
2066
2067 091A
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088 091A
2089 091A 53
2090 091B E8 0965 R
2091 091E 72 43
2092 0920 E8 0435 R
2093 0923 B8 90FD
2094 0926 CD 15
2095 0928 9C
2096 0929 E8 040F R
2097 092C 9D
2098 092D 73 05
2099 092F E8 0965 R
2100 0932 72 2F
2101
2102 0934
2103 0934 B2 0A
2104 0936 E8 0905 R
2105 0939 8A C4
2106 093B 32 E4
2107 093D 3C 08
2108 093F 73 02
2109 0941 B0 08
2110
2111
2112
2113 0943 50
2114 0944 BA F424
2115 0947 F7 E2
2116 0949 8B CA
2117 094B 8B D0
2118 094D F8
2119 094E D1 D2
2120 0950 D1 D1
2121 0952 B4 86
2122 0954 CD 15
2123 0956 58
2124 0957 73 0A
2125
2126
2127
2128 0959
2129 0959 B9 205E
2130 096C E8 0000 E
2131 096F FE C8
2132 0961 75 F6
2133
2134 0963
2135 0963 5B
2136 0964 C3
2137 0965
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148 0965
2149 0965 8B DF
2150 0967 8A CB
2151 0969 C0 C3 04
2152 096C FA

```

```

-----
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,DX DESTROYED
-----
GET_PARM PROC NEAR
PUSH DS
SUB AX,AX ; DS = 0, BIOS DATA AREA
MOV DS,AX ; BL = INDEX
XCHG DX,BX ; BX = INDEX
SUB BH,BH
ASSUME DS:ABS0
LDS SI,DISK_POINTER ; POINT TO BLOCK
MOV AH,[SI+BX] ; GET THE WORD
XCHG DX,BX ; RESTORE BX
POP SI
POP DS
RET
ASSUME DS:DATA
GET_PARM ENDP
-----
MOTOR_ON
TURN MOTOR ON AND WAIT FOR MOTOR START UP TIME. THE MOTOR COUNT:
IS REPLACED WITH A SUFFICIENTLY HIGH NUMBER (0FFH) TO ENSURE
THAT THE MOTOR DOES NOT GO OFF DURING THE OPERATION. IF THE
MOTOR NEEDED TO BE TURNED ON, THE MULTITASKING HOOK FUNCTION
(AH=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 WAIT LOOP WILL THEN DO THE WAIT.
ON ENTRY: DI = DRIVE #
ON EXIT: AX,CX,DX DESTROYED
-----
MOTOR_ON PROC NEAR
PUSH BX ; SAVE REG.
CALL TURN_ON ; TURN ON MOTOR
JNC MOT_IS_ON ; IF CY=1 NO WAIT
CALL XLAT_OLD ; TRANSLATE STATE TO COMPATIBLE MODE
MOV AX,90FDH ; LOAD WAIT CODE & TYPE
INT 15H ; TELL OPERATING SYSTEM ABOUT TO DO WAIT
PUSHF ; SAVE CY FOR TEST
CALL XLAT_NEW ; TRANSLATE STATE TO PRESENT ARCH.
POPF ; RESTORE CY FOR TEST
JNC M_WAIT ; BYPASS LOOP IF OP SYSTEM HANDLED WAIT
CALL TURN_ON ; CHECK AGAIN IF MOTOR ON
JNC MOT_IS_ON ; IF NO WAIT MEANS IT IS ON
M_WAIT:
MOV DL,10 ; GET THE MOTOR WAIT PARAMETER
CALL GET_PARM
MOV AL,AH ; AL = MOTOR WAIT PARAMETER
XOR AH,AH ; AX = MOTOR WAIT PARAMETER
CMP AL,8 ; SEE IF AT LEAST A SECOND IS SPECIFIED
JAE GP2 ; IF YES, CONTINUE
MOV AL,8 ; ONE SECOND WAIT FOR MOTOR START UP
;----- AX CONTAINS NUMBER OF 1/8 SECONDS (125000 MICROSECONDS) TO WAIT
GP2: PUSH AX ; SAVE WAIT PARAMETER
MOV DX,62500 ; LOAD LARGEST POSSIBLE MULTIPLIER
MUL DX ; MULTIPLY BY HALF OF WHAT'S NECESSARY
CX = HIGH WORD
MOV DX,AX ; CX,DX = 1/2 * (# OF MICROSECONDS)
CLC ; CLEAR CARRY FOR ROTATE
RCL DX,1 ; DOUBLE LOW WORD, CY CONTAINS OVERFLOW
RCL CX,1 ; DOUBLE HI, INCLUDING LOW WORD OVERFLOW
MOV AH,86H ; LOAD WAIT CODE
INT 15H ; PERFORM WAIT
POP AX ; RESTORE WAIT PARAMETER
JNC MOT_IS_ON ; CY MEANS WAIT COULD NOT BE DONE
;----- FOLLOWING LOOPS REQUIRED WHEN RTC WAIT FUNCTION IS ALREADY IN USE
J13: MOV CX,8286 ; WAIT FOR 1/8 SECOND PER (AL)
CALL WAITF ; COUNT FOR 1/8 SECOND AT 15.085737 US
DEC ; GO TO FIXED WAIT ROUTINE
JNZ J13 ; DECREMENT TIME VALUE
; ARE WE DONE YET
MOT_IS_ON:
POP BX ; RESTORE REG.
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
-----
TURN_ON PROC NEAR
MOV BX,DI ; BX = DRIVE #
MOV CL,BL ; CL = DRIVE #
ROL BL,4 ; BL = DRIVE SELECT
CLI ; NO INTERRUPTS WHILE DETERMINING STATUS

```

SECTION 5

```

2153 096D C6 06 0040 R FF      MOV     #MOTOR_COUNT,OFFH      ; ENSURE MOTOR STAYS ON FOR OPERATION
2154 0972 A0 003F R            MOV     AL,#MOTOR_STATUS      ; GET DIGITAL OUTPUT REGISTER REFLECTION
2155 0975 24 30              AND     AL,00110000B         ; KEEP ONLY DRIVE SELECT BITS
2156 0977 84 01              MOV     AH,1                 ; MASK FOR DETERMINING MOTOR BIT
2157 0979 D2 E4              SHL     AH,CL                ; AH = MOTOR ON, A=00000001, B=00000010
2158
2159      ; AL = DRIVE SELECT FROM #MOTOR_STATUS
2160      ; BL = DRIVE SELECT DESIRED
2161      ; AH = MOTOR ON MASK DESIRED
2162
2163 097B 3A C3              CMP     AL,BL                ; REQUESTED DRIVE ALREADY SELECTED ?
2164 097D 75 06              JNZ     TURN_IT_ON          ; IF NOT SELECTED JUMP
2165 097F 84 26 003F R       TEST    AH,#MOTOR_STATUS     ; TEST MOTOR ON BIT
2166 0983 75 2C              JNZ     NO_MOT_WAIT         ; JUMP IF MOTOR ON AND SELECTED
2167
2168 0985      TURN_IT_ON:
2169 0985 0A E3              OR     AH,BL                 ; AH = DRIVE SELECT AND MOTOR ON
2170 0987 8A 3E 003F R       MOV     BH,#MOTOR_STATUS     ; SAVE COPY OF #MOTOR_STATUS BEFORE
2171 098B 80 E7 0F           AND     BH,00001111B         ; KEEP ONLY MOTOR BITS
2172 098E 80 26 003F R CF    AND     #MOTOR_STATUS,11001111B ; CLEAR OUT DRIVE SELECT
2173 0993 08 26 003F R       OR     #MOTOR_STATUS,AH      ; OR IN DRIVE SELECTED AND MOTOR ON
2174 0997 A0 003F R         MOV     AL,#MOTOR_STATUS     ; GET DIGITAL OUTPUT REGISTER REFLECTION
2175 099A 8A D8              MOV     BL,AL                ; BL=#MOTOR_STATUS AFTER, BH=BEFORE
2176 099C 80 E3 0F           AND     BL,00001111B         ; KEEP ONLY MOTOR BITS
2177 099F FB 03              STI     ; ENABLE INTERRUPTS AGAIN
2178 09A0 24 3F           AND     AL,00111111B         ; STRIP AWAY UNWANTED BITS
2179 09A2 C0 C0 C0 04        ROL     AL,4                 ; PUT BITS IN DESIRED POSITIONS
2180 09A5 0C 0C           OR     AL,00001100B         ; NO RESET, ENABLE DMA/INTERRUPT
2181 09A7 8A 0F 09F2 R      MOV     DX,09F2H            ; SELECT DRIVE AND TURN ON MOTOR
2182 09AA EE 02              OUT    DX,AL                 ;
2183 09AB 3A DF              CMP     BL,BH                ; NEW MOTOR TURNED ON ?
2184 09AD 74 02              JZ     NO_MOT_WAIT          ; NO WAIT REQUIRED IF JUST SELECT
2185 09AF FB 03              CLC     ; SET CARRY MEANING WAIT
2186 09B0 C3              RET
2187
2188 09B1      NO_MOT_WAIT:
2189 09B1 F9              STC     ; SET NO WAIT REQUIRED
2190 09B2 FB              STI     ; INTERRUPTS BACK ON
2191 09B3 C3              RET
2192 09B4
2193
2194      ;-----
2194      ; HD_WAIT
2195      ; WAIT FOR HEAD SETTLE TIME.
2196
2197      ; ON ENTRY:  DI : DRIVE #
2198
2199      ; ON EXIT:  AX,BX,CX,DX DESTROYED
2200      ;-----
2201 09B4      HD_WAIT      PROC    NEAR
2202 09B4 B2 09              MOV     DL,9                 ; GET HEAD SETTLE PARAMETER
2203 09B6 E8 0905 R         CALL    GET_FARM             ; GET FARM
2204 09B9 F6 06 003F R 80    TEST    #MOTOR_STATUS,10000000B ; SEE IF A WRITE OPERATION
2205 09BE 74 14              JZ     ISNT_WRITE           ; IF NOT, DO NOT ENFORCE ANY VALUES
2206 09C0 0A E4              OR     AH,AH                 ; CHECK FOR ANY WAIT?
2207 09C2 75 14              JNZ    DO_WAIT              ; IF THERE DO NOT ENFORCE
2208 09C4 B4 0F              MOV     AH,HD320_SETTLE     ; LOAD 1.2M HEAD SETTLE MINIMUM
2209 09C6 8A 85 0090 R      MOV     AL,#DSK_STATE[D1]   ; LOAD STATE
2210 09CA C4 C0              AND     AL,RATE_MSK         ; KEEP ONLY RATE
2211 09CC 3C 80              CMP     AL,RATE_250         ; 1.2 M DRIVE ?
2212 09CE 75 08              JNZ    DO_WAIT              ; DEFAULT HEAD SETTLE LOADED
2213
2214 09D0 B4 14              GP3:   MOV     AH,HD320_SETTLE  ; USE 320/360 HEAD SETTLE
2215 09D2 EB 04              JMP     SHORT_DO_WAIT       ;
2216
2217 09D4      ISNT_WRITE:
2218 09D4 0A E4              OR     AH,AH                 ; CHECK FOR NO WAIT
2219 09D6 74 1F              JZ     HW_DONE              ; IF NOT WRITE AND 0 ITS OK
2220
2221      ;----- AH CONTAINS NUMBER OF MILLISECONDS TO WAIT
2222
2223 09D8 8A C4              DO_WAIT: MOV    AL,AH              ; AL = # MILLISECONDS
2224 09DA 82 E4              XOR    AH,AH                 ; AX = # MILLISECONDS
2225 09DC 80              PUSH   AX                    ; SAVE HEAD SETTLE PARAMETER
2226 09DD BA 03E8           MOV    DX,1000              ; SET UP FOR MULTIPLY TO MICROSECONDS
2227 09E0 F7 E2              MUL    DX                    ; DX,AX = # MICROSECONDS
2228 09E2 8B CA              MOV    CX,DX                 ; CX,AX = # MICROSECONDS
2229 09E4 8B D0              MOV    DX,AX                 ; CX,DX = # MICROSECONDS
2230 09E6 B4 86              MOV    AH,86H               ; LOAD WAIT CODE
2231 09E8 CD 15              INT    15H                  ; PERFORM WAIT
2232 09EA 5A 03              POP    AX                    ; RESTORE HEAD SETTLE PARAMETER
2233 09EB 73 0A              JNC    HW_DONE              ; CHECK FOR EVENT WAIT ACTIVE
2234
2235 09ED      J29:
2236 09ED B9 0042           MOV    CX,66                 ; 1 MILLISECOND LOOP
2237 09F0 E8 0000 E         CALL    WAITF                ; COUNT AT 15.085737 US PER COUNT
2238 09F3 FE C8              DEC    AL                    ; DELAY FOR 1 MILLISECOND
2239 09F5 75 F6              JNZ    J29                   ; DECREMENT THE COUNT
2240 09F7
2241 09F7 C3              HW_DONE: RET
2242 09F8
2243      ;-----
2244      ; NEC_OUTPUT
2245      ; THIS ROUTINE SENDS A BYTE TO THE NEC CONTROLLER AFTER
2246      ; TESTING FOR CORRECT DIRECTION AND CONTROLLER READY THIS
2247      ; ROUTINE WILL TIME OUT IF THE BYTE IS NOT ACCEPTED WITHIN
2248      ; A REASONABLE AMOUNT OF TIME, SETTING THE DISKETTE STATUS
2249      ; ON COMPLETION.
2250
2251      ; ON ENTRY:
2252      ; AH = BYTE TO BE OUTPUT
2253
2254      ; ON EXIT:
2255      ; CY = 0 SUCCESS
2256      ; CY = 1 FAILURE -- DISKETTE STATUS UPDATED
2257      ; IF A FAILURE HAS OCCURRED, THE RETURN IS MADE
2258      ; ONE LEVEL HIGHER THAN THE CALLER OF NEC_OUTPUT.
2259      ; THIS REMOVES THE REQUIREMENT OF TESTING AFTER
2260      ; EVERY CALL OF NEC_OUTPUT.
2261      ; AX,CX,DX DESTROYED
2262
2263 09F8      NEC_OUTPUT  PROC    NEAR
2264 09F8 53              PUSH   BX                    ; SAVE REG.
2265 09F9 BA 03F4           MOV    DX,03F4H             ; STATUS PORT
2266 09FC 83 02              MOV    BL,2                 ; HIGH ORDER COUNTER
2267 09FE 33 C9              XOR    CX,CX                 ; COUNT FOR TIME OUT
    
```



```

2267
2268 0A00 EC J23: IN AL,DX ; GET STATUS
2269 0A01 24 C0 AND AL,11000000B ; KEEP STATUS AND DIRECTION
2270 0A03 3C 80 CMP AL,10000000B ; STATUS 1 AND DIRECTION 0 ?
2271 0A05 74 0F JZ J27 ; STATUS AND DIRECTION OK
2272 0A07 E2 F7 LOOP J23 ; CONTINUE TILL CX EXHAUSTED
2273
2274 0A09 FE CB DEC BL ; DECREMENT COUNTER
2275 0A0B 75 F3 JNZ J23 ; REPEAT TILL DELAY FINISHED, CX = 0
2276
2277
2278
2279 0A0D 80 0E 0041 R 80 I----- FALL THRU TO ERROR RETURN
2280 0A12 5B OR #DSKETTE_STATUS,TIME_OUT ;
2281 0A13 58 POP BX ; RESTORE REG.
2282 0A14 F9 POP AX ; DISCARD THE RETURN ADDRESS
2283 0A15 C3 STC ; INDICATE ERROR TO CALLER
2284 RET
2285
2286
2287 0A16 J27: I----- DIRECTION AND STATUS OK; OUTPUT BYTE
2288 0A16 8A C4 MOV AL,AH ; GET BYTE TO OUTPUT
2289 0A18 42 INC DL ; DATA PORT = STATUS PORT + 1
2290 0A19 EE OUT DX,AL ; OUTPUT THE BYTE
2291
2292 0A1A 9C PUSHF ; SAVE FLAGS
2293 0A1B B9 0003 MOV CX,3 ; 30 TO 45 MICROSECOND WAIT FOR
2294 0A1E E8 0000 E CALL WAITF ; NEC FLAGS UPDATE CYCLE
2295 0A21 9D POPF ; RESTORE FLAGS FOR EXIT
2296 0A22 5B POP BX ; RESTORE REG.
2297 0A23 C3 RET ; CY = 0 FROM TEST INSTRUCTION
2298 0A24
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312 0A24 SEEK PROC NEAR
2313 0A24 8B DF MOV BX,DI ; BX = DRIVE #
2314 0A26 80 01 MOV AL,1 ; ESTABLISH MASK FOR RECALIBRATE TEST
2315 0A28 86 CB XCHG CL,BL ; GET DRIVE VALUE INTO CL
2316 0A2A D2 C0 ROL AL,CL ; SHIFT MASK BY THE DRIVE VALUE
2317 0A2C 86 0B XCHG CL,BL ; RECOVER TRACK VALUE
2318 0A2E 84 06 003E R TEST AL,#SEEK_STATUS ; TEST FOR RECALIBRATE REQUIRED
2319 0A32 75 1C JNZ J28A ; JUMP IF RECALIBRATE NOT REQUIRED
2320
2321 0A34 08 06 003E R OR #SEEK_STATUS,AL ; TURN ON THE NO RECALIBRATE BIT IN FLAG
2322 0A38 E8 0A83 R CALL RECAL ; RECALIBRATE DRIVE
2323 0A3B 73 0A JNC AFT_RECAL ; RECALIBRATE DONE
2324
2325
2326
2327
2328
2329
2330
2331 0A47 I----- ISSUE RECALIBRATE FOR 80 TRACK DISKETTES
2332 0A47 C6 06 0041 R 00 MOV #DSKETTE_STATUS,0 ; CLEAR OUT INVALID STATUS
2333 0A48 E8 0A83 R CALL RECAL ; RECALIBRATE DRIVE
2334 0A45 72 3B JC RB ; IF RECALIBRATE FAILS TWICE THEN ERROR
2335
2336
2337
2338 0A47 AFT_RECAL:
2339 0A47 C6 85 0094 R 00 MOV #DSK_TRK[DI],0 ; SAVE NEW CYLINDER AS PRESENT POSITION
2340 0A4C 04 ED OR CH,CR ; CHECK FOR SEEK TO TRACK 0
2341 0A4E 74 2D JZ DO_WAIT ; HEAD SETTLE, CY = 0 IF JUMP
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359 0A7D I----- DRIVE IS IN SYNCHRONIZATION WITH CONTROLLER, SEEK TO TRACK
2360 0A7D 9C J28A: TEST #DSK_STATE[DI],OBL_STEP ; CHECK FOR DOUBLE STEP REQUIRED
2361 0A7E E8 09B4 R JZ RT ; SINGLE STEP REQUIRED BYPASS DOUBLE
2362 0A81 9D SHL CH,1 ; DOUBLE NUMBER OF STEP TO TAKE
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375 0A89 RT: MOV CH,#DSK_TRK[DI] ; SEE IF ALREADY AT THE DESIRED TRACK
2376 0A8B 3A AD 0094 R JE RB ; IF YES, DO NOT NEED TO SEEK
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2999
3000

```

SECTION 5

```

(DSK3.ASM)
2381 0A8D 8B DF      MOV     BX,DI          ; BX = DRIVE #
2382 0A8F 8A E9      MOV     AH,BL
2383 0A91 E8 09F8 R  CALL    NEC_OUTPUT    ; OUTPUT THE DRIVE NUMBER
2384 0A94 E8 0A9A R  CALL    CHK_STAT_2    ; GET THE INTERRUPT AND SENSE INT STATUS
2385 0A97 58        POP     AX             ; THROW AWAY ERROR
2386 0A98          RC_BACK:
2387 0A98 59        POP     CX
2388 0A99 C3        RET
2389 0A9A          RECAL ENDP
2390
2391          ;-----
2392          ; CHK_STAT_2
2393          ; THIS ROUTINE HANDLES THE INTERRUPT RECEIVED AFTER
2394          ; RECALIBRATE OR SEEK TO THE ADAPTER. THE
2395          ; INTERRUPT IS WAITED FOR, THE INTERRUPT STATUS SENSED,
2396          ; AND THE RESULT RETURNED TO THE CALLER.
2397          ;
2398          ; ON EXIT:  *DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION.
2399          ;-----
2399 0A9A          CHK_STAT_2 PROC NEAR
2400 0A9A 88 0A8B R  MOV     AX,OFFSET CS_BACK ; LOAD NEC_OUTPUT ERROR ADDRESS
2401 0A9D 50        PUSH    AX
2402 0A9E E8 0AC1 R  CALL    WAIT_INT      ; WAIT FOR THE INTERRUPT
2403 0AA1 72 14      JC      J34           ; IF ERROR, RETURN IT
2404 0AA3 84 08      MOV     AH,08H        ; SENSE INTERRUPT STATUS COMMAND
2405 0AA5 E8 09F8 R  CALL    NEC_OUTPUT    ; CALL NEC_OUTPUT
2406 0AA8 E8 0A99 R  CALL    RESULTS       ; READ IN THE RESULTS
2407 0AAB 72 0A      JC      J34           ;
2408 0AAD A0 0042 R  MOV     AL,*NEC_STATUS ; GET THE FIRST STATUS BYTE
2409 0AB0 24 60      AND    AL,01100000B   ; ISOLATE THE BITS
2410 0AB2 3C 60      CMP    AL,01100000B   ; TEST FOR CORRECT VALUE
2411 0AB4 74 03      JZ     J36           ; IF ERROR, GO MARK IT
2412 0AB6 F8        CLC
2413 0AB7          J34:
2414 0AB7 58        POP     AX             ; THROW AWAY ERROR RETURN
2415 0AB8          CS_BACK:
2416 0AB8 C3        RET
2417
2418 0AB9          J35:
2419 0AB9 80 0E 0041 R 40 OR     *DSKETTE_STATUS,BAD_SEEK ; ERROR RETURN CODE
2420 0ABE F9        STC
2421 0ABF EB F6      JMP    SHORT J34
2422 0AC1          CHK_STAT_2 ENDP
2423
2424          ;-----
2425          ; WAIT_INT
2426          ; THIS ROUTINE WAITS FOR AN INTERRUPT TO OCCUR A TIME OUT
2427          ; ROUTINE TAKES PLACE DURING THE WAIT, SO THAT AN ERROR
2428          ; MAY BE RETURNED IF THE DRIVE IS NOT READY.
2429          ;
2430          ; ON EXIT:  *DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION.
2431          ;-----
2430 0AC1          WAIT_INT PROC NEAR
2431 0AC1 80          STI
2432 0AC1 FB        CLC                   ; TURN ON INTERRUPTS, JUST IN CASE
2433 0AC2 F8        MOV     AX,09001H     ; CLEAR TIMEOUT INDICATOR
2434 0AC3 88 9001  MOV     ISH,09001H    ; LOAD WAIT CODE AND TYPE
2435 0AC6 CD 15      INT    15H           ; PERFORM OTHER FUNCTION
2436 0AC8 72 11      JC      J36A          ; BYPASS TIMING LOOP IF TIMEOUT DONE
2437 0ACA 8B 0A      MOV     BL,10         ; CLEAR THE COUNTERS
2438 0ACC 33 C9      XOR    CX,CX         ; FOR 2 SECOND WAIT
2439 0ACE          J36:
2440 0ACE F6 06 003E R 80 TEST   *SEEK_STATUS,INT_FLAG ; TEST FOR INTERRUPT OCCURRING
2441 0AD3 75 0C      JNZ    J37           ; COUNT DOWN WHILE WAITING
2442 0AD5 E2 F7      LOOP   J36          ; SECOND LEVEL COUNTER
2443 0AD7 FE CB      DEC    BL
2444 0AD9 75 F3      JNZ    J36          ;
2445
2446 0ADB 80 0E 0041 R 80 J36A:  OR     *DSKETTE_STATUS,TIME_OUT ; NOTHING HAPPENED
2447 0AE0 F9        STC                   ; ERROR RETURN
2448 0AE1          J37:
2449 0AE1 9C        PUSHF                ; SAVE CURRENT CARRY
2450 0AE2 80 26 003E R 7F AND    *SEEK_STATUS,NOT_INT_FLAG ; TURN OFF INTERRUPT FLAG
2451 0AET 9D        POPF                 ; RECOVER CARRY
2452 0AE8 C3        RET                 ; GOOD RETURN CODE
2453 0AE9          WAIT_INT ENDP
2454
2455          ;-----
2456          ; RESULTS
2457          ; THIS ROUTINE WILL READ ANYTHING THAT THE NEC CONTROLLER
2458          ; RETURNS FOLLOWING AN INTERRUPT.
2459          ;
2460          ; ON EXIT:  *DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION.
2461          ; AX,BX,CX,DX DESTROYED
2462          ;-----
2462 0AE9          RESULTS PROC NEAR
2463 0AE9 57        PUSH    DI
2464 0AEA 8F 0042 R  MOV     DI,OFFSET *NEC_STATUS ; POINTER TO DATA AREA
2465 0AED B3 07      MOV     BL,7         ; MAX STATUS BYTES
2466 0AEF BA 03F4   MOV     DX,03F4H     ; STATUS PORT
2467
2468          ;----- WAIT FOR REQUEST FOR MASTER
2469
2470 0AF2 87 02      R10:  MOV     BH,2         ; HIGH ORDER COUNTER
2471 0AF4 33 C9      XOR    CX,CX         ; COUNTER
2472 0AF6          J39:
2473 0AF6 EC        IN     AL,DX         ; WAIT FOR MASTER
2474 0AF7 24 C0      AND    AL,11000000B ; GET STATUS
2475 0AF9 3C C0      CMP    AL,11000000B ; KEEP ONLY STATUS AND DIRECTION
2476 0AFB 74 0E      JZ     J42           ; STATUS AND DIRECTION OK
2477 0AFD E2 F7      LOOP   J39          ; STATUS AND DIRECTION OK
2478
2479 0AFF FE CF      DEC    BH           ; LOOP TILL TIMEOUT
2480 0B01 75 F3      JNZ    J39          ; DECREMENT HIGH ORDER COUNTER
2481
2482 0B03 80 0E 0041 R 80 OR     *DSKETTE_STATUS,TIME_OUT ; REPEAT TILL DELAY DONE
2483 0B08 F9        STC                   ; SET ERROR RETURN
2484 0B09 EB 18      JMP    SHORT POPRES  ; POP REGISTERS AND RETURN
2485
2486          ;----- READ IN THE STATUS
2487
2488 0B0B          J42:
2489          ;
2490 0B0B 42        JMP     J+2           ; I/O DELAY
2491 0B0C EC        INC    DX           ; POINT AT DATA PORT
2492 0B0D 88 05      IN     AL,DX         ; GET THE DATA
2493 0B0F 47        MOV    [DI],AL       ; STORE THE BYTE
2494          INC    DI           ; INCREMENT THE POINTER
    
```

```

2495 0B10 B9 0003      MOV     CX,3           ; MINIMUM 24 MICROSECONDS FOR NEC
2496 0B13 EB 0000 E    CALL   WAITF          ; WAIT 30 TO 48 MICROSECONDS
2497 0B16 4A           DEC     DX             ; POINT AT STATUS PORT
2498 0B17 EC           IN      AL,DX          ; GET STATUS
2499 0B18 A8 10       TEST   AL,00010000B   ; TEST FOR NEC STILL BUSY
2500 0B1A 74 0A       JZ     POPRES         ; RESULTS DONE ?
2501
2502 0B1C FE CB        DEC     BL             ; DECREMENT THE STATUS COUNTER
2503 0B1E 75 D2       JNZ   R10             ; GO BACK FOR MORE
2504 0B20 80 0E 0041 R 20 OR     R10,0SKETTE_STATUS,BAD_NEC ; TOO MANY STATUS BYTES
2505 0B25 F9           STC                    ; SET ERROR FLAG
2506
2507 ;----- RESULT OPERATION IS DONE
2508
2509 0B26             POPRES:
2510 0B26 5F           POP     DI             ; RETURN WITH CARRY SET
2511 0B27 C3           RET
2512 0B28             RESULTS ENDP
2513 ;-----
2514 ; READ_DSKCHNG
2515 ; READS THE STATE OF THE DISK CHANGE LINE.
2516 ;
2517 ; ON ENTRY:  DI = DRIVE #
2518 ;
2519 ; ON EXIT:   DI = DRIVE #
2520 ;           ZF = 0 ; DISK CHANGE LINE INACTIVE
2521 ;           ZF = 1 ; DISK CHANGE LINE ACTIVE
2522 ;           AX,CX,DX DESTROYED
2523 ;-----
2524 0B28             READ_DSKCHNG PROC NEAR
2525 0B28 EB 091A R    CALL   MOTOR_ON      ; TURN ON THE MOTOR IF OFF
2526 0B2B BA 03F7     MOV     DX,03F7H      ; ADDRESS DIGITAL INPUT REGISTER
2527 0B2E EC           IN      AL,DX          ; INPUT DIGITAL INPUT REGISTER
2528 0B2F AB 80       TEST   AL,DSK_CHG    ; CHECK FOR DISK CHANGE LINE ACTIVE
2529 0B31 C3           RET                    ; RETURN TO CALLER WITH ZERO FLAG SET
2530 0B32             READ_DSKCHNG ENDP
2531 ;-----
2532 ; DRIVE_DET
2533 ; DETERMINES WHETHER DRIVE IS 80 OR 40 TRACKS AND
2534 ; UPDATES STATE INFORMATION ACCORDINGLY.
2535 ;
2536 ; ON ENTRY:  DI = DRIVE #
2537 ;-----
2538 0B32             DRIVE_DET PROC NEAR
2539 0B32 EB 091A R    CALL   MOTOR_ON      ; TURN ON MOTOR IF NOT ALREADY ON
2540 0B35 EB 0A83 R    CALL   RECAL         ; RECALIBRATE DRIVE
2541 0B38 72 3C       JC     DD_BAC         ; ASSUME NO DRIVE PRESENT
2542 0B3A B5 30       MOV     CH,TRK_SLAP   ; SEEK TO TRACK 48
2543 0B3C EB 0A24 R    CALL   SEEK          ; SEEK
2544 0B3F 72 38       JC     DD_BAC         ; ERROR NO DRIVE
2545 0B41 B5 08       MOV     CH,QUIET_SEEK+1 ; SEEK TO TRACK 10
2546 0B43             SK_GIN:
2547 0B43 FE CD        DEC     CH             ; DECREMENT TO NEXT TRACK
2548 0B45 51         PUSH    CX             ; SAVE TRACK
2549 0B46 EB 0A24 R    CALL   SEEK          ; SEEK
2550 0B49 72 2C       JC     POP_BAC        ; POP AND RETURN
2551 0B4B B8 0B77 R    MOV     AX,OFFSET POP_BAC ; LOAD NEC OUTPUT ERROR ADDRESS
2552 0B4E 50         PUSH    AX             ;
2553 0B4F B4 04       MOV     AH,SENSE_DRV_ST ; SENSE DRIVE STATUS COMMAND BYTE
2554 0B51 EB 09F8 R    CALL   NEC_OUTPUT    ; OUTPUT TO NEC
2555 0B54 B8 C7       MOV     AL,DI          ; AL = DRIVE
2556 0B56 B8 E0       MOV     AH,AL          ; AH = DRIVE
2557 0B58 EB 09F8 R    CALL   NEC_OUTPUT    ; OUTPUT TO NEC
2558 0B5B EB 0AE9 R    CALL   RESULTS       ; GO GET STATUS
2559 0B5E 58         POP     AX             ; THROW AWAY ERROR ADDRESS
2560 0B5F 59         POP     CX             ; RESTORE TRACK
2561 0B60 F6 06 0042 R 10 TEST   #NEC_STATUS,HOME ; TRACK 0 ?
2562 0B65 74 DC       JZ     SK_GIN         ; GO TILL TRACK 0
2563 0B67 0A ED       OR     CH,CH          ; IS HOME AT TRACK 0 ?
2564 0B69 74 06       JZ     IS_80          ; MUST BE 80 TRACK DRIVE
2565
2566 ; DRIVE IS A 360; SET DRIVE TO DETERMINED;
2567 ; SET MEDIA TO DETERMINED AT RATE 250.
2568
2569 0B6B 80 8D 0090 R 94 OR     #DSK_STATE[DI],DRV_DET+MED_DET+RATE 250
2570 0B70 C3           RET                    ; ALL INFORMATION SET
2571
2572 0B71             IS_80:
2573 0B71 80 8D 0090 R 01 OR     #DSK_STATE[DI],TRK_CAPA ; SETUP 80 TRACK CAPABILITY
2574 0B76             DD_BAC:
2575 0B76 C3           RET
2576
2577 0B77             POP_BAC:
2578 0B77 59         POP     CX             ; THROW AWAY
2579 0B78 C3           RET
2580
2581 0B79             DRIVE_DET ENDP
2582 ;-----
2583 ; DISK_INT
2584 ; THIS ROUTINE HANDLES THE DISKETTE INTERRUPT.
2585 ;
2586 ; ON EXIT:   THE INTERRUPT FLAG IS SET IN #SEEK_STATUS.
2587 ;-----
2588 0B79             DISK_INT PROC FAR
2589 0B79 60         PUSH    AX             ; ENTRY POINT FOR ORG 0EF57H
2590 0B7A 1E         PUSH    DS             ; SAVE WORK REGISTER
2591 0B7B EB 0000 E    CALL   DDS            ; SAVE REGISTERS
2592 0B7E 80 0E 003E R 80 OR     #SEEK_STATUS,INT_FLAG ; SETUP DATA ADDRESSING
2593 0B83 1F         POP     DS             ; TURN ON INTERRUPT OCCURRED
2594 0B84 B0 20       MOV     AL,DSI        ; RESTORE USER (DS)
2595 0B86 E6 20       OUT    INTA00,AL      ; END OF INTERRUPT MARKER
2596 0B88 FB         STI                    ; INTERRUPT CONTROL PORT
2597 0B89 B8 9101     MOV     AX,09101H     ; RE-ENABLE INTERRUPTS
2598 0B8C CD 15       INT    15H            ; INTERRUPT POST CODE AND TYPE
2599 0B8E 58         POP     AX             ; GO PERFORM OTHER TASK
2600 0B8F CF         IRET                   ; RECOVER REGISTER
2601 0B90             DISK_INT_1 ENDP
    
```

SECTION 5

```

2602 PAGE
2603 ;-----
2604 ; DSKETTE SETUP
2605 ; THIS ROUTINE DOES A PRELIMINARY CHECK TO SEE WHAT TYPE
2606 ; OF DISKETTE DRIVES ARE ATTACH TO THE SYSTEM.
2607 ;-----
2608 DSKETTE_SETUP PROC NEAR
2609     PUSH AX ; SAVE REGISTERS
2610     PUSH BX
2611     PUSH CX
2612     PUSH DX
2613     PUSH DI
2614     PUSH DS
2615     CALL DDS ; POINT DATA SEGMENT TO BIOS DATA AREA
2616     OR #RTC_WAIT_FLAG,01 ; NO RTC WAIT, FORCE USE OF LOOP
2617     XOR DI,DI ; INITIALIZE DRIVE POINTER
2618     MOV WORD PTR #DSK_STATE,0 ; INITIALIZE STATES
2619     AND #LAstrate,NOT#STRT_MSK+SEND_MSK ; CLEAR START & SEND
2620     OR #LAstrate,SEND_MSK ; INITIALIZE SENT TO IMPOSSIBLE
2621     MOV #SEEK_STATUS,0 ; INDICATE RECALIBRATE NEEDED
2622     MOV #MOTOR_COUNT,0 ; INITIALIZE MOTOR COUNT
2623     MOV #MOTOR_STATUS,0 ; INITIALIZE DRIVES TO OFF STATE
2624     MOV #DSKETTE_STATUS,0 ; NO ERRORS
2625
2626 SUP0:
2627     CALL DRIVE_DET ; DETERMINE DRIVE
2628     CALL XLAT_OLD ; TRANSLATE STATE TO COMPATIBLE MODE
2629     INC DI ; POINT TO NEXT DRIVE
2630     CMP DI,#MAX_DRV ; SEE IF DONE
2631     JNZ SUP0 ; REPEAT FOR EACH DRIVE
2632     MOV #SEEK_STATUS,0 ; FORCE RECALIBRATE
2633     AND #RTC_WAIT_FLAG,OFEH ; ALLOW FOR RTC WAIT
2634     CALL SETUP_END ; VARIOUS CLEANUPS
2635     POP DS ; RESTORE CALLERS REGISTERS
2636     POP DI
2637     POP DX
2638     POP CX
2639     POP BX
2640     POP AX
2641     RET
2642 DSKETTE_SETUP ENDP
2643 CODE _ENDS
2644 END
    
```

```

1 PAGE 118,121
2 TITLE DISK ----- 09/25/85 FIXED DISK BIOS
3 .286C
4 .LIST
5 0000 CODE SEGMENT BYTE PUBLIC
6
7 PUBLIC DISK_IO
8 PUBLIC DISK_SETUP
9 PUBLIC HD_INT
10
11 EXTRN CMOS_READ:NEAR
12 EXTRN CMOS_WRITE:NEAR
13 EXTRN DOS:NEAR
14 EXTRN E_MSG:NEAR
15 EXTRN F1780:NEAR
16 EXTRN F1781:NEAR
17 EXTRN F1782:NEAR
18 EXTRN F1790:NEAR
19 EXTRN F1791:NEAR
20 EXTRN FD_TBL:NEAR
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
    
```

----- INT 13H -----

FIXED DISK I/O INTERFACE

THIS INTERFACE PROVIDES ACCESS TO 5 1/4" FIXED DISKS THROUGH THE IBM FIXED DISK CONTROLLER.

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

INPUT (AH)= HEX COMMAND VALUE

- (AH) = 00H RESET DISK (DL = 80H,81H) / DISKETTE
- (AH) = 01H READ THE STATUS OF THE LAST DISK OPERATION INTO (AL)  
NOTE: DL < 80H - DISKETTE  
DL > 80H - DISK
- (AH) = 02H READ THE DESIRED SECTORS INTO MEMORY
- (AH) = 03H WRITE THE DESIRED SECTORS FROM MEMORY
- (AH) = 04H VERIFY THE DESIRED SECTORS
- (AH) = 05H FORMAT THE DESIRED TRACK
- (AH) = 06H UNUSED
- (AH) = 07H UNUSED
- (AH) = 08H RETURN THE CURRENT DRIVE PARAMETERS
- (AH) = 09H INITIALIZE DRIVE CHARACTERISTICS  
INTERRUPT 41 POINTS TO DATA BLOCK FOR DRIVE 0  
INTERRUPT 46 POINTS TO DATA BLOCK FOR DRIVE 1
- (AH) = 0AH READ LONG
- (AH) = 0BH WRITE LONG (READ & WRITE LONG ENCOMPASS 512 + 4 BYTES ECC)
- (AH) = 0CH SEEK
- (AH) = 0DH ALTERNATE DISK RESET (SEE DL)
- (AH) = 0EH UNUSED
- (AH) = 0FH UNUSED
- (AH) = 10H TEST DRIVE READY
- (AH) = 11H RECALIBRATE
- (AH) = 12H UNUSED
- (AH) = 13H UNUSED
- (AH) = 14H CONTROLLER INTERNAL DIAGNOSTIC
- (AH) = 15H READ DASD TYPE

REGISTERS USED FOR FIXED DISK OPERATIONS

- (DL) - DRIVE NUMBER (80H-81H FOR DISK, VALUE CHECKED)
  - (DH) - HEAD NUMBER (0-15 ALLOWED, NOT VALUE CHECKED)
  - (CH) - CYLINDER NUMBER (0-1023, NOT VALUE CHECKED) (SEE CL)
  - (CL) - SECTOR NUMBER (1-17, NOT VALUE CHECKED)
- NOTE: HIGH 2 BITS OF CYLINDER NUMBER ARE PLACED IN THE HIGH 2 BITS OF THE CL REGISTER (10 BITS TOTAL)
- (AL) - NUMBER OF SECTORS (MAXIMUM POSSIBLE RANGE 1-80H, FOR READ/WRITE LONG 1-79H)
  - (ES:BX) - ADDRESS OF BUFFER FOR READS AND WRITES, (NOT REQUIRED FOR VERIFY)
- FORMAT (AH=8) 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         |

SECTION 5

99  
 100  
 101  
 102  
 103  
 104  
 105  
 106  
 107  
 108  
 109  
 110  
 111  
 112  
 113  
 114  
 115  
 116  
 117  
 118  
 119  
 120  
 121  
 122  
 123  
 124  
 125  
 126  
 127  
 128  
 129  
 130  
 131  
 132  
 133  
 134  
 135  
 136  
 137  
 138  
 139  
 140  
 141  
 142 = 00FF  
 143 = 00E0  
 144 = 00CC  
 145 = 00BB  
 146 = 00AA  
 147 = 0080  
 148 = 0040  
 149 = 0020  
 150 = 0011  
 151 = 0010  
 152 = 000B  
 153 = 000A  
 154 = 0009  
 155 = 0007  
 156 = 0005  
 157 = 0004  
 158 = 0002  
 159 = 0001  
 160  
 161  
 162  
 163  
 164  
 165  
 166  
 167  
 168  
 169  
 170  
 171  
 172  
 173  
 174  
 175  
 176  
 177  
 178  
 179  
 180  
 181  
 182  
 183  
 184  
 185  
 186  
 187

```

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 DASH 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.
    
```

```

SENSE_FAIL EQU 0FFH ; NOT IMPLEMENTED
NO_ERR EQU 0E0H ; STATUS ERROR/ERROR REGISTER=0
WRITE_FAULT EQU 0CCH ; WRITE FAULT ON SELECTED DRIVE
UNDEF_ERR EQU 0BBH ; UNDEFINED ERROR OCCURRED
NOT_RDY EQU 0AAH ; DRIVE NOT READY
TIME_OUT EQU 80H ; ATTACHMENT FAILED TO RESPOND
BAD_SEEK EQU 40H ; SEEK OPERATION FAILED
BAD_CNTRL EQU 20H ; CONTROLLER HAS FAILED
DATA_CORRECTED EQU 11H ; ECC CORRECTED DATA ERROR
BAD_ECC EQU 10H ; BAD ECC ON DISK READ
BAD_TRACK EQU 0BH ; NOT IMPLEMENTED
BAD_SECTOR EQU 0AH ; BAD SECTOR FLAG DETECTED
DMA_BOUNDARY EQU 09H ; DATA EXTENDS TOO FAR
INIT_FAIL EQU 07H ; DRIVE PARAMETER ACTIVITY FAILED
BAD_RESET EQU 05H ; RESET FAILED
RECORD_NOT_FND EQU 04H ; REQUESTED SECTOR NOT FOUND
BAD_ADDR_MARK EQU 02H ; ADDRESS MARK NOT FOUND
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 5 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.
    
```

```

188 PAGE
189 |-----|
190 | HARDWARE SPECIFIC VALUES |
191 | |
192 | - CONTROLLER I/O PORT |
193 | |
194 | > WHEN READ FROM: |
195 | HF_PORT+0 - READ DATA (FROM CONTROLLER TO CPU) |
196 | HF_PORT+1 - GET ERROR REGISTER |
197 | HF_PORT+2 - GET SECTOR COUNT |
198 | HF_PORT+3 - GET SECTOR NUMBER |
199 | HF_PORT+4 - GET CYLINDER LOW |
200 | HF_PORT+5 - GET CYLINDER HIGH (2 BITS) |
201 | HF_PORT+6 - GET SIZE/DRIVE/HEAD |
202 | HF_PORT+7 - GET STATUS REGISTER |
203 | |
204 | > WHEN WRITTEN TO: |
205 | HF_PORT+0 - WRITE DATA (FROM CPU TO CONTROLLER) |
206 | HF_PORT+1 - SET PRECOMPENSATION CYLINDER |
207 | HF_PORT+2 - SET SECTOR COUNT |
208 | HF_PORT+3 - SET SECTOR NUMBER |
209 | HF_PORT+4 - SET CYLINDER LOW |
210 | HF_PORT+5 - SET CYLINDER HIGH (2 BITS) |
211 | HF_PORT+6 - SET SIZE/DRIVE/HEAD |
212 | HF_PORT+7 - SET COMMAND REGISTER |
213 |-----|
214
215
216
217 = 01F0 HF_PORT EQU 01F0H ; DISK PORT
218 = 03F6 HF_REG_PORT EQU 03F6H
219
220 |-----|
221 | STATUS REGISTER |
222 | ST_ERROR EQU 00000001B ;
223 | ST_INDEX EQU 00000100B ;
224 | ST_CORRCTD EQU 00000100B ; ECC CORRECTION SUCCESSFUL
225 | ST_DRG EQU 00001000B ;
226 | ST_SEEK_COMPL EQU 00010000B ; SEEK COMPLETE
227 | ST_WRT_FLT EQU 00100000B ; WRITE FAULT
228 | ST_READY EQU 01000000B ;
229 | ST_BUSY EQU 10000000B ;
230
231 |-----|
232 | ERROR REGISTER |
233 | ERR_DAM EQU 00000001B ; DATA ADDRESS MARK NOT FOUND
234 | ERR_TRK_0 EQU 00000100B ; TRACK 0 NOT FOUND ON RECAL
235 | ERR_ABORT EQU 00000100B ; ABORTED COMMAND
236 | ERR_ID EQU 00001000B ; NOT USED
237 | ERR_ID EQU 00010000B ; ID NOT FOUND
238 | ERR_ID EQU 00100000B ; NOT USED
239 | ERR_DATA_ECC EQU 01000000B ;
240 | ERR_BAD_BLOCK EQU 10000000B ;
241
242
243 | RECAL_CMD EQU 00010000B ; DRIVE RECAL (10H)
244 | READ_CMD EQU 00100000B ; READ (20H)
245 | WRITE_CMD EQU 00110000B ; WRITE (30H)
246 | VERIFY_CMD EQU 01000000B ; VERIFY (40H)
247 | FMTRK_CMD EQU 01010000B ; FORMAT TRACK (50H)
248 | INIT_CMD EQU 01100000B ; INITIALIZE (60H)
249 | SEEK_CMD EQU 01110000B ; SEEK (70H)
250 | DIAG_CMD EQU 10010000B ; DIAGNOSTIC (90H)
251 | SET_PARM_CMD EQU 10010001B ; DRIVE PARAMS (91H)
252 | NO_RETRIES EQU 00000001B ; CMD MODIFIER (01H)
253 | ECC_MODE EQU 00000010B ; CMD MODIFIER (02H)
254 | BUFFER_MODE EQU 00001000B ; CMD MODIFIER (08H)
255
256 = 0002 MAX_FILE EQU 2
257 = 0002 S_MAX_FILE EQU 2
258
259 = 0025 DELAY_1 EQU 25H ; DELAY FOR OPERATION COMPLETE
260 = 0600 DELAY_2 EQU 0600H ; DELAY FOR READY
261 = 0100 DELAY_3 EQU 0100H ; DELAY FOR DATA REQUEST
262
263 = 0008 HF_FAIL EQU 08H ; CMOS FLAG IN BYTE 0EH
264
265 |-----|
266 | COMMAND BLOCK REFERENCE |
267 = *CMD_BLOCK EQU BYTE PTR [BP]-8 ; *CMD_BLOCK REFERENCES BLOCK HEAD IN SS
268 ; (BP) POINTS TO COMMAND BLOCK TAIL
269 ; AS DEFINED BY THE *ENTER* PARAMS
    
```

SECTION 5

```

270 PAGE
271 -----
272 | FIXED DISK I/O SETUP |
273 | |
274 | - ESTABLISH TRANSFER VECTORS FOR THE FIXED DISK |
275 | - PERFORM POWER ON DIAGNOSTICS |
276 | SHOULD AN ERROR OCCUR A "I701" MESSAGE IS DISPLAYED |
277 | |
278 -----
279 ASSUME CS:CODE,DS:ABS0 ; WORK OFF DS REGISTER
280
281 0000 DISK_SETUP PROC NEAR
282 0000 FA CL1
283 0001 BB ---- R MOV AX,ABS0 ; GET ABSOLUTE SEGMENT
284 0004 8E DB MOV DS,AX ; SET SEGMENT REGISTER
285 0006 A1 004C R MOV AX,WORD PTR *ORG_VECTOR ; GET DISKETTE VECTOR
286 0009 A3 0100 R MOV WORD PTR *DISK_VECTOR,AX ; INTO INT 40H
287 000C A1 004E R MOV AX,WORD PTR *ORG_VECTOR+2
288 000F A3 0102 R MOV WORD PTR *DISK_VECTOR+2,AX
289 0012 C7 06 004C R 01A9 R MOV WORD PTR *ORG_VECTOR,OFFSET DISK_IO ; FIXED DISK HANDLER
290 0018 8C 0E 004E R MOV WORD PTR *ORG_VECTOR+2,CS
291 001C C7 06 010B R 06DA R MOV WORD PTR *DISK_INT,OFFSET HD_INT ; FIXED DISK INTERRUPT
292 0022 8C 0E 01DA R MOV WORD PTR *DISK_INT+2,CS
293 0026 C7 06 0104 R 0000 E MOV WORD PTR *HF_TBL_VEC,OFFSET FD_TBL ; PARM TABLE DRIVE 80
294 002C 8C 0E 0106 R MOV WORD PTR *HF_TBL_VEC+2,CS
295 0030 C7 06 0118 R 0000 E MOV WORD PTR *HF1_TBL_VEC,OFFSET FD_TBL ; PARM TABLE DRIVE 81
296 0036 8C 0E 011A R MOV WORD PTR *HF1_TBL_VEC+2,CS
297 003A E4 A1 IN AL,INTB01 ; TURN ON SECOND INTERRUPT CHIP
298 003C 24 BF AND AL,0BFH
299 003E EB 00 JMP $+2
300 0040 E6 A1 OUT INTB01,AL
301 0042 E4 21 IN AL,INTA01 ; LET INTERRUPTS PASS THRU TO
302 0044 24 BF AND AL,0BFH ; SECOND CHIP
303 0046 EB 00 JMP $+2
304 0048 E6 21 OUT INTA01,AL
305
306 004A FB STI
307 ASSUME DS:DATA,ES:ABS0
308 004B IE PUSH DS ; MOVE ABS0 POINTER TO
309 004C 07 POP ES ; EXTRA SEGMENT POINTER
310 004E 8B 0000 E CALL DOS ; ESTABLISH DATA SEGMENT
311 0050 C6 06 0074 R 00 MOV *DISK_STATUS1,0 ; RESET THE STATUS INDICATOR
312 0055 C6 06 0075 R 00 MOV *HF_NUM,0 ; ZERO NUMBER OF FIXED DISKS
313 005A C6 06 0076 R 00 MOV *CONTROL_BYTE,0
314 005F B0 8E MOV AL,CMOS_DIAG+NM1
315 0061 E8 0000 E CALL CMOS_READ ; CHECK CMOS VALIDITY
316 0064 8A E0 MOV AH,AL ; SAVE CMOS FLAG
317 0066 24 C0 AND AL,BAD_BAT+BAD_CKSUM ; CHECK FOR VALID CMOS
318 0068 74 03 JZ L1
319 006A E9 00F8 R JMP POD_DONE ; CMOS NOT VALID -- NO FIXED DISKS
320 006D
321 006D 80 E4 F7 L1: AND AH,NOT HF_FAIL ; ALLOW FIXED DISK IPL
322 0070 B0 8E MOV AL,CMOS_OTAG+NM1 ; WRITE IT BACK
323 0072 EB 0000 E CALL CMOS_WRITE
324 0075 B0 2F MOV AL,CMOS_DISK+NM1
325 0077 E8 0000 E CALL CMOS_READ
326 007A C6 06 0077 R 00 MOV *PORT_OFF,0 ; ZERO CARD OFFSET
327 007F 8A D8 MOV BL,AL ; SAVE FIXED DISK BYTE
328 0081 25 00F0 AND AX,00F0FH ; GET FIRST DRIVE TYPE AS OFFSET
329 0084 74 72 JZ POD_DONE ; NO FIXED DISKS
330
331 0086 3C F0 CMP AL,0F0H ; CHECK FOR EXTENDED DRIVE TYPE BYTE USE
332 0088 75 10 JNE L2 ; USE DRIVE TYPE 1 --> 14 IF NOT IN USE
333
334 008A B0 99 MOV AL,CMOS_DISK_1+NM1 ; GET EXTENDED TYPE FOR DRIVE C:
335 008C E8 00D0 E CALL CMOS_READ ; FROM CMOS
336 008F 3C 00 CMP AL,0 ; IS TYPE SET TO ZERO
337 0091 74 65 JE POD_DONE ; EXIT IF NOT VALID AND NO FIXED DISKS
338 0093 3C 2F CMP AL,47 ; IS TYPE WITHIN VALID RANGE
339 0095 77 41 JA POD_DONE ; EXIT WITH NO FIXED DISKS IF NOT VALID
340 0097 C1 E0 04 SHL AX,4 ; ADJUST TYPE TO HIGH NIBBLE
341 009A
342 009A 05 FFF0 E L2: ADD AX,OFFSET FD_TBL-16D ; COMPUTE OFFSET OF FIRST DRIVE TABLE
343 009D 2E1 A3 0104 R MOV WORD PTR *HF_TBL_VEC,AX ; SAVE IN VECTOR POINTER
344 00A1 C6 06 0075 R 01 MOV *HF_NUM,1 ; AT LEAST ONE DRIVE
345 00A6 8A C3 MOV AL,BL
346 00A8 C0 E0 04 SHL AL,4 ; GET SECOND DRIVE TYPE
347 00AB 74 2A JZ SHORT L4 ; ONLY ONE DRIVE
348 00AD B4 00 MOV AH,0
349
350 00AF 3C F0 CMP AL,0F0H ; CHECK FOR EXTENDED DRIVE TYPE BYTE USE
351 00B1 75 10 JNE L3 ; USE DRIVE TYPE 1 --> 14 IF NOT IN USE
352
353 00B3 B0 9A MOV AL,CMOS_DISK_2+NM1 ; GET EXTENDED TYPE FOR DRIVE D:
354 00B5 E8 00D0 E CALL CMOS_READ ; FROM CMOS
355 00B8 3C 00 CMP AL,0 ; IS TYPE SET TO ZERO
356 00BA 74 1B JE L4 ; SKIP IF SECOND FIXED DISK NOT VALID
357 00BC 3C 2F CMP AL,47 ; IS TYPE WITHIN VALID RANGE
358 00BE 77 17 JA L4 ; SKIP IF NOT VALID
359 00C0 C1 E0 04 SHL AX,4 ; ADJUST TYPE TO HIGH NIBBLE
360 00C3
361 00C3 05 FFF0 E L3: ADD AX,OFFSET FD_TBL-16D ; COMPUTE OFFSET FOR SECOND FIXED DISK
362 00C6 8B D0 MOV BX,AX
363 00C8 2E1 B8 3F 00 CMP WORD PTR CS:[BX],0 ; CHECK FOR ZERO CYLINDERS IN TABLE
364 00CC 74 09 JE L4 ; SKIP DRIVE IF NOT A VALID TABLE ENTRY
365 00CE 2E1 A3 0118 R MOV WORD PTR *HF1_TBL_VEC,AX
366 00D2 C6 06 0075 R 02 MOV *HF_NUM,2 ; TWO DRIVES
367 00D7
368 00D7 B2 50 MOV DL,80H ; CHECK THE CONTROLLER
369 00D9 B4 14 MOV AH,14H ; USE CONTROLLER DIAGNOSTIC COMMAND
370 00DB CD 13 INT 13H ; CALL BIOS WITH DIAGNOSTIC COMMAND
371 00DD 72 1A JC CTL_ERR ; DISPLAY ERROR MESSAGE IF BAD RETURN
372 00DF A1 006C R MOV AX,*TIMER_LOW ; GET START TIMER COUNTS
373 00E2 8B D8 MOV BX,AX
374 00E4 05 0444 ADD AX,6*182 ; 60 SECONDS* 18.2
375 00E7 8B C6 MOV CX,AX
376 00E9 E8 0104 R CALL HD_RESET_1 ; SET UP DRIVE 0
377 00EC 80 3E 0075 R 01 CMP *HF_NUM,1 ; WERE THERE TWO DRIVES?
378 00F1 76 05 JBE POD_DONE ; NO-ALL DONE
379 00F3 B2 51 MOV DL,51H ; SET UP DRIVE 1
380 00F5 E8 0104 R CALL HD_RESET_1
381 00F8
382 00F8 C3 POD_DONE: RET
383
    
```



```

384          |----- POD ERROR
385
386 00F9          CTL_ERRX:
387 00F9 BE 0000 E      MOV     SI,OFFSET F1782      ; CONTROLLER ERROR
388 00FC E8 017C R      CALL   SET_FAIL            ; DO NOT IPL FROM DISK
389 00FF E8 0000 E      CALL   E_MSG               ; DISPLAY ERROR AND SET (BP) ERROR FLAG
390 0102 EB F4          JMP     POD_DONE
391
392
393 0104          HD_RESET:      PROC   NEAR
394 0104 53          PUSH  BX                    ; SAVE TIMER LIMITS
395 0105 51          PUSH  CX
396 0106 B4 09      RES_1:  MOV   AH,09H          ; SET DRIVE PARAMETERS
397 0108 CD 13      INT   13H
398 010A 72 06      JC    RES_2
399 010C B4 11      MOV   AH,T1H              ; RECALIBRATE DRIVE
400 010E CD 13      INT   13H
401 0110 73 19      JNC   RES_OK              ; DRIVE OK
402 0112 E8 018A R  RES_2:  CALL  POD_TCHK            ; CHECK TIME OUT
403 0115 73 EF      JNC   RES_1
404 0117 BE 0000 E  RES_FL:  MOV   SI,OFFSET F1781  ; INDICATE DISK 1 FAILURE
405 011A F6 C2 01   AND   DL,1
406 011D 74 5F      JNZ   RES_E1
407 011F BE 0000 E   MOV   SI,OFFSET F1780  ; INDICATE DISK 0 FAILURE
408 0122 E8 017C R   CALL  SET_FAIL            ; DO NOT TRY TO IPL DISK 0
409 0125 EB 4F      JMP   SHORT RES_E1
410 0127 B4 05      RES_RS:  MOV   AH,00H              ; RESET THE DRIVE
411 0129 CD 13      INT   13H
412 012B B4 08      RES_CK:  MOV   AH,08H            ; GET MAX CYLINDER,HEAD,SECTOR
413 012D BA 0F      MOV   DL,DL
414 012F CD 13      INT   13H
415 0131 72 38      JC    RES_ER              ; SAVE MAX CYLINDER, SECTOR
416 0133 89 0E 0042 R  MOV   DL,BL
417 0137 8A D3      MOV   DL,BL
418 0139 B8 0401    RES_3:  MOV   AX,0401H          ; VERIFY THE LAST SECTOR
419 013C CD 13      INT   13H
420 013E 73 39      JNC   RES_OK              ; VERIFY OK
421 0140 80 FC 0A    CMP   AH,BAD_SECTOR      ; OK ALSO IF JUST ID READ
422 0143 74 34      JE    RES_OK
423 0145 80 FC 11    CMP   AH,DATA_CORRECTED
424 0148 74 2F      JE    RES_OK
425 014A 80 FC 10    CMP   AH,BAD_ECC
426 014D 74 2A      JE    RES_OK
427 014F E8 018A R   CALL  POD_TCHK            ; CHECK FOR TIME OUT
428 0152 72 17      JC    RES_ER              ; FAILED
429 0154 8B 0E 0042 R  MOV   CX,WORD PTR #NEC_STATUS ; GET SECTOR ADDRESS, AND CYLINDER
430 0158 8A C1      MOV   AL,CL
431 015A 24 3F      AND   AL,3FH
432 015C FE C6      AL    DEC
433 015E 74 C7      JZ    RES_RS              ; TRY PREVIOUS ONE
434 0160 80 E1 C0    AND   CL,0C0H            ; WE'VE TRIED ALL SECTORS ON TRACK
435 0163 0A C8      OR    CL,AL
436 0165 89 0E 0042 R  MOV   WORD PTR #NEC_STATUS,CX ; KEEP CYLINDER BITS
437 0169 EB CE      JMP   RES_3              ; MERGE SECTOR WITH CYLINDER BITS
438 016B BE 0000 E   RES_ER:  MOV   SI,OFFSET F1791  ; SAVE CYLINDER, NEW SECTOR NUMBER
439 016E F6 C2 01   AND   DL,1
440 0171 75 03      JNZ   RES_E1              ; TRY AGAIN
441 0173 BE 0000 E   MOV   SI,OFFSET F1790  ; INDICATE DISK 0 ERROR
442 0176 74 0A      JC    RES_E1
443 0178 E8 0000 E   CALL  E_MSG               ; DISPLAY ERROR AND SET (BP) ERROR FLAG
444 0179
445 0179 59          POP   CX
446 017A 58          POP   BX
447 017B C3          RET
448 017C
449
450 017C          HD_RESET_I:      ENDP
451 017C B8 BE8E
452 017F E8 0000 E   SET_FAIL:  MOV   AX,X*(CMOS_DIAG+NM1) ; GET CMOS ERROR BYTE
453 0182 0C 08      OR    AL,#F FAIL
454 0184 86 E0      XCHG  AH,AL
455 0186 E8 0000 E   CALL  CMOS_WRITE          ; SET DO NOT IPL FROM DISK FLAG
456 0189 C3          RET
457 018A
458
459 018A          POD_TCHK:      PROC   NEAR
460 018A 58          POP   AX                    ; CHECK FOR 30 SECOND TIME OUT
461 018B 59          POP   CX                    ; SAVE RETURN
462 018C 5B          POP   BX                    ; GET TIME OUT LIMITS
463 018D 53          PUSH  BX                    ; AND SAVE THEM AGAIN
464 018E 51          PUSH  CX
465 018F 50          PUSH  AX
466 0190 A1 006C R  MOV   AX,*TIMER_LOW
467
468
469 0193 3B D9      CMP   BX,CX
470 0195 72 06      JB    TCHK1
471 0197 3B D8      CMP   BX,AX
472 0199 72 0C      JB    TCHK2
473 019B EB 04      JMP   SHORT TCHK2
474 019D 3B C3      TCHK1:  CMP   AX,BX
475 019F 72 04      JB    TCHKNG
476 01A1 3B C1      TCHK2:  CMP   AX,CX
477 01A3 72 02      JB    TCHKNG
478
479 01A5 F9          TCHKNG:  STC
480 01A6 C3          RET
481 01A7 F8          TCHKG:  CLC
482 01A8 C3          RET
483 01A9
484
485 01A9          POD_TCHK:      ENDP
486
487 01A9          DISK_SETUP:    ENDP
    
```

SECTION 5

```

486 PAGE
487 -----
488 |-----|
489 |-----|
490 |-----|
491 01A9 DISK_IO PROC FAR
492 ASSUME DS:DATA,ES:NOTHING
493 01A9 80 FA 80 CMP DL,80H ; TEST FOR FIXED DISK DRIVE
494 01AC 73 05 JAE A1 ; YES, HANDLE HERE
495 01AE CD 40 INT 40H ; DISKETTE HANDLER
496 01B0 RET_2: RET 2 ; BACK TO CALLER
497 01B0 CA 0002
498
499 01B3 A1: ; ENABLE INTERRUPTS
500 01B3 FB STI
501 01B4 0A E4 OR AH,AH
502 01B6 73 09 JNZ A2
503 01B8 CD 40 INT 40H ; RESET NEC WHEN AH=0
504 01BA 2A E4 SUB AH,AH
505 01BC 80 FA 81 CMP DL,(80H + S_MAX_FILE - 1)
506 01BF 77 EF JA RET_2 ; SUCCESS OR FAILURE
507 01C1
508 01C1 80 FC 08 A2: CMP AH,08H ; GET PARAMETERS IS A SPECIAL CASE
509 01C4 73 03 JNZ A3
510 01C6 E9 0393 R JMP GET_PARAM_N
511 01C9 80 FC 15 A3: CMP AH,7FH ; READ DASD TYPE IS ALSO
512 01CC 73 03 JNZ A4
513 01CE E9 0353 R JMP READ_DASD_TYPE
514
515 01D1 A4: ; SAVE REGISTERS DURING OPERATION
516 01D1 C8 0008 00 ENTER 8,0 ; SAVE (BP) AND MAKE ROOM FOR %CMD_BLOCK
517 01D8 63 PUSH BX ; IN THE STACK. THE COMMAND_BLOCK=151
518 01D6 81 PUSH CX ; %CMD_BLOCK += BYTE PTR [BP]-8
519 01D7 82 PUSH DX
520 01D8 1E PUSH DS
521 01D9 06 PUSH ES
522 01DA 56 PUSH SI
523 01DB 87 PUSH DI
524 01DC 0A E4 OR AH,AH
525 01DE 73 02 JNZ AB ; CHECK FOR RESET
526 01E0 B2 60 MOV DL,80H ; FORCE DRIVE 80 FOR RESET
527 01E2 EA 0226 R CALL DISK_IO_CONT ; PERFORM THE OPERATION
528 01E5 E9 0000 E CALL DCS ; ESTABLISH SEGMENT
529 01E8 8A 26 0074 R MOV AH,%DISK_STATUS ; GET STATUS FROM OPERATION
530 01EC 80 FC 01 CMP AH,1 ; SET THE CARRY FLAG TO INDICATE
531 01EF F5 CMC ; SUCCESS OR FAILURE
532 01F0 8F POP DI ; RESTORE REGISTERS
533 01F1 5E POP SI
534 01F2 07 POP ES
535 01F3 1F POP DS
536 01F4 5A POP DX
537 01F5 59 POP CX
538 01F6 58 POP BX
539 01F7 C9 LEAVE
540 01FA CA 0002 DISK_IO RET 2 ; ADJUST (SP) AND RESTORE (BP)
541 01FB ENDP ; THROW AWAY SAVED FLAGS
542
543 01FB MI LABEL WORD ; FUNCTION TRANSFER TABLE
544 01FB 02C1 R DW DISK_RESET ; 000H
545 01FD 0315 R DW RETURN_STATUS ; 001H
546 01FF 031E R DW DISK_READ ; 002H
547 0201 0325 R DW DISK_WRITE ; 003H
548 0203 032C R DW DISK_VERIFY ; 004H
549 0205 033E R DW FAT_TRK ; 006H
550 0207 02B9 R DW BAD_COMMAND_006H ; 006H FORMAT BAD SECTORS
551 0209 02B9 R DW BAD_COMMAND_007H ; 007H FORMAT DRIVE
552 020B 02B9 R DW BAD_COMMAND_008H ; 008H RETURN PARAMETERS
553 020D 03F1 R DW INIT_DRV ; 009H
554 020F 0423 R DW RD_LONG ; 00AH
555 0211 042A R DW WR_LONG ; 00BH
556 0213 0431 R DW DISK_SEEK ; 00CH
557 0215 02C1 R DW DISK_RESET ; 00DH
558 0217 02B9 R DW BAD_COMMAND_00EH ; 00EH READ BUFFER
559 0219 02B9 R DW BAD_COMMAND_00FH ; 00FH WRITE BUFFER
560 021B 044F R DW TST_RDY ; 010H
561 021D 0468 R DW HDISK_RECAL ; 011H
562 021F 02B9 R DW BAD_COMMAND_012H ; 012H MEMORY DIAGNOSTIC
563 0221 02B9 R DW BAD_COMMAND_013H ; 013H DRIVE DIAGNOSTIC
564 0223 048E R DW CTRL_DIAGNOSTIC ; 014H CONTROLLER DIAGNOSTIC
565 = 002A
566
567 0225 DISK_IO_CONT PROC NEAR
568 0225 E8 0000 E CALL DCS ; ESTABLISH SEGMENT
569 0228 80 FC 01 CMP AH,01H ; RETURN STATUS
570 022B 73 03 JNZ SU0
571 022D E9 0315 R JMP RETURN_STATUS
572 0230
573 0230 C6 06 0074 R 00 SU0: MOV %DISK_STATUS,0 ; RESET THE STATUS INDICATOR
574 0235 53 PUSH BX ; SAVE DATA ADDRESS
575 0236 8A 1E 0075 R MOV BL,%HF_NUM ; GET NUMBER OF DRIVES
576 023A 50 PUSH AX
577 023B 80 E2 7F AND DL,7FH ; GET DRIVE AS 0 OR 1
578 023E 3A DA CMP BL,DL
579 0240 76 75 JBE BAD_COMMAND_POP ; INVALID DRIVE
580 0242 06 FUSH ES
581 0243 E8 06C4 R CALL GET_VEC ; GET DISK PARAMETERS
582 0246 26 8B 47 08 MOV AX,WORD PTR ES:[BX][6] ; GET WRITE PRE-COMPENSATION CYLINDER
583 024A C1 E8 02 SHR AX,2
584 024D 8B 46 F8 MOV %CMD_BLOCK,AL
585 0250 26 8A 47 08 MOV AL,BYTE PTR ES:[BX][8] ; GET CONTROL BYTE MODIFIER
586 0254 52 PUSH DX
587 0255 BA 03F6 MOV DX,%F_REG_PORT
588 0258 EE OUT DX,AL ; SET EXTRA HEAD OPTION
589 0259 5A POP DX
590 025A 07 POP ES
591 025B 8A 26 0076 R MOV AH,%CONTROL_BYTE ; SET EXTRA HEAD OPTION IN
592 025F 80 E4 C0 AND AH,0C0H ; CONTROL BYTE
593 0262 0A E0 OR AH,AL
594 0264 8B 26 0076 R MOV %CONTROL_BYTE,AH
595 0268 58 POP AX
596 0269 8B 46 F9 MOV %CMD_BLOCK+1,AL ; SECTOR COUNT
597 026C 50 PUSH AX
598 026D 8A C1 MOV AL,CL ; GET SECTOR NUMBER
599 026F 24 3F AND AL,3FH
    
```

```

600 0271 88 46 FA      MOV     @CMD_BLOCK+2,AL
601 0274 88 6E FB      MOV     @CMD_BLOCK+3,CH      ; GET CYLINDER NUMBER
602 0277 8A C1        MOV     AL,CL
603 0279 C0 E8 06      SHR     AL,6
604 027C 88 46 FC      MOV     @CMD_BLOCK+4,AL      ; CYLINDER HIGH ORDER 2 BITS
605 027F 8A C2        MOV     AL,DL      ; DRIVE NUMBER
606 0281 C0 E0 04      SHL     AL,4
607 0284 80 E6 0F      AND     DH,0FH      ; HEAD NUMBER
608 0287 0A C6        OR      AL,DH
609 0289 DC A0        OR      AL,20H OR 20H      ; ECC AND 512 BYTE SECTORS
610 028B 88 46 FD      MOV     @CMD_BLOCK+5,AL      ; ECC/SIZE/DRIVE/HEAD
611 028E 58            POP     AX
612 028F 80            PUSH    AX
613 0290 8A C4        MOV     AL,AH      ; GET INTO LOW BYTE
614 0292 32 E4        XOR     AH,AH      ; ZERO HIGH BYTE
615 0294 D1 E0        SAL     AX,1      ; *2 FOR TABLE LOOKUP
616 0296 BB F0        MOV     SI,AX      ; PUT INTO SI FOR BRANCH
617 0298 3D 002A      CMP     AX,M1L      ; TEST WITHIN RANGE
618 029B 73 1A        JNB     BAD_COMMAND_POP
619 029D 58            POP     AX      ; RESTORE AX
620 029E 58            POP     BX      ; AND DATA ADDRESS
621 029F 51            PUSH    CX
622 02A0 80            PUSH    AX      ; ADJUST ES:BX
623 02A1 8B C9        MOV     CX,BX      ; GET 3 HIGH ORDER NIBBLES OF BX
624 02A3 C1 E9 04      SHR     CX,4
625 02A6 8C C0        MOV     AX,ES
626 02A8 03 C1        ADD     AX,CX
627 02AA 8E C0        MOV     ES,AX
628 02AC 81 E3 000F    AND     BX,000FH      ; ES:BX CHANGED TO ES:000X
629 02B0 58            POP     AX
630 02B1 59            POP     CX
631 02B2 2E 1F A4 01FB R  JMP     WORD PTR CS:[SI + OFFSET MI]
BAD_COMMAND_POP:
632 02B7 58            POP     AX
633 02B8 5B            POP     BX
BAD_COMMAND:
634 02B9 58            POP     AX
635 02BA 58            POP     BX
636 02BB C6 06 0074 R 01  MOV     @DISK_STATUS1,BAD_CMD ; COMMAND ERROR
637 02BE 80 00        MOV     AL,0
638 02C0 C3            RET
DISK_IO_CNT      ENDP
640
641 ;-----
642 ; RESET THE DISK SYSTEM (AH=00H) ;
643 ;-----
644
DISK_RESET      PROC NEAR
645 02C1          CLI
646 02C1 FA          IN      AL,INTB01      ; GET THE MASK REGISTER
647 02C2 E4 A1        AND     $+2
648 02C4 EB 00        AND     AL,0BFH      ; ENABLE FIXED DISK INTERRUPT
649 02C6 24 BF        OUT     INTB01,AL      ; START INTERRUPTS
650 02C8 E6 A1        STI
651 02CA FB          MOV     AL,04H
652 02CB 80 04        MOV     DX,HF_REG_PORT
653 02CD 8A 03F6      MOV     DX,AL      ; RESET
654 02D0 EE          OUT     DX,AL      ; DELAY COUNT
655 02D1 89 000A      MOV     CX,10
656 02D4 49          DEC     CX
657 02D5 75 FD        JNZ     DRD           ; WAIT 4.8 MICRO-SEC
658 02D7 A0 0076 R    MOV     AL,@CONTROL_BYTE
659 02DA 24 0F        AND     AL,0FH      ; SET HEAD OPTION
660 02DC EE          OUT     DX,AL      ; TURN RESET OFF
661 02DD EB 05F3 R    CALL    NOT_BUSY
662 02E0 75 2D        JNZ     DRERR        ; TIME OUT ON RESET
663 02E2 8A 01F1      MOV     DX,HF_PORT+1
664 02E5 EC          IN      AL,DX
665 02E6 3C 01        CMP     AL,1
666 02E8 75 25        JNZ     DRERR        ; BAD RESET STATUS
667 02EA 80 66 FD EF  AND     @CMD_BLOCK+5,0EFH ; SET TO DRIVE 0
668 02EE 2A D2        SUB     DL,DL
669 02F0 E8 03F1 R    CALL    INIT_DRV     ; SET MAX HEADS
670 02F3 EB 0466 R    CALL    HDISK_RECAL  ; RECAL TO RESET SEEK SPEED
671 02F6 80 3E 0075 R 01  CMP     @HF_NDM,1    ; CHECK FOR DRIVE 1
672 02FB 76 0C        JBE     DRE
673 02FD 80 4E FD 10  OR      @CMD_BLOCK+5,010H ; SET TO DRIVE 1
674 0301 82 01        MOV     DL,1
675 0303 E8 03F1 R    CALL    INIT_DRV     ; SET MAX HEADS
676 0306 EB 0466 R    CALL    HDISK_RECAL  ; RECAL TO RESET SEEK SPEED
677 0309 C6 06 0074 R 00  MOV     @DISK_STATUS1,0 ; IGNORE ANY SET UP ERRORS
678 030E C3            RET
DRD:             MOV     @DISK_STATUS1,BAD_RESET ; CARD FAILED
DRERR:          RET
DISK_RESET      ENDP
682
683 ;-----
684 ; DISK STATUS ROUTINE (AH = 01H) ;
685 ;-----
686
RETURN_STATUS   PROC NEAR
687 0315          MOV     AL,@DISK_STATUS1 ; OBTAIN PREVIOUS STATUS
688 0315 A0 0074 R    MOV     @DISK_STATUS1,0 ; RESET STATUS
689 0318 C6 05 0074 R 00  RET
RETURN_STATUS   ENDP
691 031E
    
```

SECTION 5

```

692 PAGE
693 -----
694 ; DISK READ ROUTINE (AH = 02H) ;
695 -----
696 DISK_READ PROC NEAR
697 031E MOV @CMD_BLOCK+6,READ_CMD
698 0322 E9 04C6 R JMP COMMAND1
700 0325 ENDP
701 DISK_READ
702 -----
703 ; DISK WRITE ROUTINE (AH = 03H) ;
704 -----
705 DISK_WRITE PROC NEAR
706 0325 MOV @CMD_BLOCK+6,WRITE_CMD
707 0326 C6 46 FE 30 JMP COMMAND0
708 0329 E9 0505 R ENDP
709 032C DISK_WRITE
710 -----
711 ; DISK VERIFY (AH = 04H) ;
712 -----
713 DISK_VERF PROC NEAR
714 032C MOV @CMD_BLOCK+6,VERIFY_CMD
715 0330 CALL COMMAND
716 0333 75 08 JNZ VERF_EXIT ; CONTROLLER STILL BUSY
717 0335 E8 05C2 R CALL WAIT
718 0338 75 03 JNZ VERF_EXIT ; TIME OUT
719 033A E8 0630 R CALL CHECK_STATUS
720 033D VERF_EXIT: RET
721 033E DISK_VERF ENDP
722 -----
723 ; FORMATTING (AH = 05H) ;
724 -----
725 FMT_TRK PROC NEAR
726 033E MOV @CMD_BLOCK+6,FMTTRK_CMD ; FORMAT TRACK (AH = 005H)
727 0342 06 PUSH ES
728 0343 53 PUSH BX
729 0344 E8 04C4 R CALL GET_VEC ; GET DISK PARAMETERS ADDRESS
730 0347 26 8A 47 0E MOV AL,ES:[BX][14] ; GET SECTORS/TRACK
731 0348 88 46 F9 MOV @CMD_BLOCK+1,AL ; SET SECTOR COUNT IN COMMAND
732 034E 5B POP BX
733 034F 07 POP ES
734 0350 E9 050A R JMP CMD_OF ; GO EXECUTE THE COMMAND
735 0353 ENDP
736 FMT_TRK
737 -----
738 ; READ DASD TYPE (AH = 15H) ;
739 -----
740 READ_DASD_TYPE LABEL NEAR
741 0353 PROC FAR ; GET DRIVE PARAMETERS
742 0353 PUSH DS ; SAVE REGISTERS
743 0354 06 PUSH ES
744 0355 53 PUSH BX
745 0356 ASSUME DS:DATA
746 0357 CALL DDS ; ESTABLISH ADDRESSING
747 0359 C6 06 0074 R 00 MOV @DISK_STATUS1,0 ; DISK STATUS1,0
748 035E 8A 1E 0075 R MOV BL,@HF_NUM ; GET NUMBER OF DRIVES
749 0362 80 E2 7F AND DL,7FH ; GET DRIVE NUMBER
750 0365 3A DA CMP BL,DL
751 0367 76 22 JBE RDT_NOT_PRESENT ; RETURN DRIVE NOT PRESENT
752 0369 E8 06C4 R CALL GET_VEC ; GET DISK PARAMETER ADDRESS
753 036C 26 8A 47 02 MOV AL,ES:[BX][2] ; HEADS
754 0370 26 8A 4F 0E MOV CL,ES:[BX][14] ; * NUMBER OF SECTORS
755 0374 F6 E9 IMUL CL ; MAX NUMBER OF CYLINDERS
756 0376 26 8B 0F MOV CX,ES:[BX] ; LEAVE ONE FOR DIAGNOSTICS
757 0379 49 DEC CX ; NUMBER OF SECTORS
758 037A F7 E9 IMUL CX ; HIGH ORDER HALF
759 037C 8B CA MOV DX,AX ; LOW ORDER HALF
760 037E 8B D0 MOV AX,AX
761 0380 2B C0 SUB AX,AX
762 0382 B4 03 MOV AH,03H ; INDICATE FIXED DISK
763 0384 5B POP BX ; RESTORE REGISTERS
764 0385 07 POP ES
765 0386 1F POP DS
766 0387 F8 CLC ; CLEAR CARRY
767 0388 CA 0002 RET 2
768 038B RDT_NOT_PRESENT: SUB AX,AX ; DRIVE NOT PRESENT RETURN
769 038D 8B C8 MOV CX,AX ; ZERO BLOCK COUNT
770 038F 8B D0 MOV DX,AX
771 0391 EB F1 JMP RDT2
772 0393 READ_D_T ENDP

```

```

781 PAGE
782 |-----|
783 | GET PARAMETERS (AH = 08H) |
784 |-----|
785
786 0393 GET_PARM_N LABEL NEAR
787 0393 GET_PARM PROC FAR ; GET DRIVE PARAMETERS
788 0393 IE PUSH DS ; SAVE REGISTERS
789 0394 06 PUSH ES
790 0395 53 PUSH BX
791 ASSUME DS:ABS0
792 0396 88 ---- R MOV AX,ABS0 ; ESTABLISH ADDRESSING
793 0399 9E D8 MOV DS,AX
794 039B 76 C2 01 TEST DS,1 ; CHECK FOR DRIVE 1
795 039E 74 06 JZ G0
796 03A0 C4 1E 0118 R LES BX,#HF1_TBL_VEC
797 03A4 ED 04 JMP SHORT GT
798 03A6 C4 1E 0104 R LES BX,#HF_TBL_VEC
799 ASSUME DS:DATA
800 03AA CALL DDS ; ESTABLISH SEGMENT
801 03AD 80 EA 80 SUB DL,80H
802 03B0 80 FA 02 CMP DL,MAX_FILE ; TEST WITHIN RANGE
803 03B3 73 2C JAE G4
804 03B5 C6 06 0074 R 00 MOV #DISK_STATUS1,0
805 03BA 26 18 07 MOV AX,EST[BX] ; MAX NUMBER OF CYLINDERS
806 03BD 2D 0002 SUB AX,2 ; ADJUST FOR 0-N
807 03C0 8A E8 MOV CH,AL
808 03C2 2B 0800 AND AX,0800H ; HIGH TWO BITS OF CYLINDER
809 03C5 D1 E8 SHR AX,1
810 03C7 D1 E8 SHR AX,1
811 03C9 26 18 47 0E OR AL,ES:[BX][14] ; SECTORS
812 03CD 8A C8 CL,AL
813 03CF 26 18 77 02 MOV DH,ES:[BX][2] ; HEADS
814 03D3 FE CE DEC DH ; 0-N RANGE
815 03D5 8A 16 0075 R MOV DL,#HF_NUM ; DRIVE COUNT
816 03D9 2B C0 SUB AX,AX
817 03DB POP BX ; RESTORE REGISTERS
818 03DC 07 POP ES
819 03DD 1F POP DS
820 03DE CA 0002 RET 2
821 03E1 MOV #DISK_STATUS1,INIT_FAIL ; OPERATION FAILED
822 03E6 B4 07 MOV AH,INIT_FAIL
823 03E8 2A C0 SUB AL,AL
824 03EA 2B D2 SUB DX,DX
825 03EC 2B C9 SUB CX,CX
826 03EE F9 STC ; SET ERROR FLAG
827 03EF EB EA JMP GS
828 03F1 GET_PARM ENDP
829
830 |-----|
831 | INITIALIZE DRIVE (AH = 09H) |
832 |-----|
833
834 INIT_DRV PROC NEAR
835 03F1 MOV #CMD_BLOCK+6,SET_PARM_CMD
836 03F3 EB 06C4 R CALL GET_VEC ; ES:BX -> PARAMETER BLOCK
837 03F8 26 18 47 02 MOV AL,ES:[BX][2] ; GET NUMBER OF HEADS
838 03FC FE C8 DEC AL ; CONVERT TO 0-INDEX
839 03FE 8A 66 FD MOV AH,#CMD_BLOCK+5 ; GET SDH REGISTER
840 0401 80 E4 F0 AND AH,0F0H ; CHANGE HEAD NUMBER
841 0404 0A E0 OR AH,AL ; TO MAX HEAD
842 0406 8B 66 FD MOV #CMD_BLOCK+5,AH
843 0409 26 18 47 0E MOV AL,ES:[BX][14] ; MAX SECTOR NUMBER
844 040D 8B 46 F9 MOV #CMD_BLOCK+1,AL
845 0410 2B C0 SUB AX,AX
846 0412 8B 46 FB MOV #CMD_BLOCK+3,AL ; ZERO FLAGS
847 0415 E8 055C R CALL COMMAND ; TELL CONTROLLER
848 0418 75 08 JNZ INIT_EXIT ; CONTROLLER BUSY ERROR
849 041A E8 05F3 R CALL NOT_BUSY ; WAIT FOR IT TO BE DONE
850 041D 75 03 JNZ INIT_EXIT ; TIME OUT
851 041F E8 0630 R CALL CHECK_STATUS
852 0422 C3 INIT_EXIT: RET
853 0423 INIT_DRV ENDP
854
855 |-----|
856 | READ LONG (AH = 0AH) |
857 |-----|
858
859 RD_LONG PROC NEAR
860 0423 MOV #CMD_BLOCK+6,READ_CMD OR ECC_MODE
861 0427 E9 04C6 R JMP COMMAND
862 042A RD_LONG ENDP
863
864 |-----|
865 | WRITE LONG (AH = 0BH) |
866 |-----|
867
868 WR_LONG PROC NEAR
869 042A MOV #CMD_BLOCK+6,WRITE_CMD OR ECC_MODE
870 042E E9 0505 R JMP COMMAND
871 0431 WR_LONG ENDP
872
873 |-----|
874 | SEEK (AH = 0CH) |
875 |-----|
876
877 DISK_SEEK PROC NEAR
878 0431 MOV #CMD_BLOCK+6,SEEK_CMD
879 0435 E8 055C R CALL COMMAND
880 0438 75 14 JNZ DS_EXIT ; CONTROLLER BUSY ERROR
881 043A E8 08C2 R CALL WAIT ; TIME OUT ON SEEK
882 043D 75 0F JNZ DS_EXIT
883 043F E8 0630 R CALL CHECK_STATUS
884 0442 30 0E 0074 R 40 CMP #DISK_STATUS1,BAD_SEEK
885 0447 75 05 JNE DS_EXIT
886 0449 C6 06 0074 R 00 MOV #DISK_STATUS1,0
887 044E C3 DS_EXIT: RET
888 044F DISK_SEEK ENDP
889
890
891
892
893

```

SECTION 5

```

894 PAGE
895 ;-----
896 ; TEST DISK READY (AH = 10H) ;
897 ;-----
898
899 TST_RDY PROC NEAR
900 044F EB 05F3 R CALL NOT_BUSY ; WAIT FOR CONTROLLER
901 0452 75 11 JNZ TR_EX
902 0454 BA 46 FD MOV AL,PCMD_BLOCK+5 ; SELECT DRIVE
903 0457 BA 01F6 MOV DX,HF_PORT+6
904 045A EE OUT DX,AL
905 045B EB 0642 R CALL CHECK_ST ; CHECK STATUS ONLY
906 045E 75 05 JNZ TR_EX
907 0460 C6 06 0074 R 00 MOV @DISK_STATUS1,0 ; WIPE OUT DATA CORRECTED ERROR
908 0465 C3 RET
909 0466 TST_RDY ENDP
910
911 ;-----
912 ; RECALIBRATE (AH = 11H) ;
913 ;-----
914
915 HDISK_RECAL PROC NEAR
916 0466 C6 46 FE 10 MOV @CMD_BLOCK+8,RECAL_CMD
917 046A EB 085C R CALL COMMAND ; START THE OPERATION
918 046D 75 19 JNZ RECAL_EXIT ; ERROR
919 046F EB 08C2 R CALL WAIT ; WAIT FOR COMPLETION
920 0472 74 05 JZ RECAL_X ; TIME OUT ONE OK ?
921 0474 EB 08C2 R CALL WAIT ; WAIT FOR COMPLETION LONGER
922 0477 75 0F JNZ RECAL_EXIT ; TIME OUT TWO TIMES IS ERROR
923 0479
924 0479 EB 0690 R CALL CHECK_STATUS
925 047C 80 3E 0074 R 40 CMP @DISK_STATUS1,BAD_SEEK ; SEEK NOT COMPLETE
926 0481 75 05 JNE RECAL_EXIT ; IS OK
927 0483 C6 06 0074 R 00 MOV @DISK_STATUS1,0
928 0488
929 0488 80 3E 0074 R 00 CMP @DISK_STATUS1,0
930 048D C3 RET
931 048E HDISK_RECAL ENDP
932
933 ;-----
934 ; CONTROLLER DIAGNOSTIC (AH = 14H) ;
935 ;-----
936
937 CTLR_DIAGNOSTIC PROC NEAR
938 048E FA CLI ; DISABLE INTERRUPTS WHILE CHANGING MASK
939 048F EA A1 IN AL,INTB01 ; TURN ON SECOND INTERRUPT CHIP
940 0491 24 BF AND AL,0BFH
941 0493 EB 00 JMP $+2
942 0495 EA A1 OUT INTB01,AL
943 0497 E4 21 IN AL,INTA01 ; LET INTERRUPTS PASS THRU TO
944 0499 24 FB AND AL,0FBH ; SECOND CHIP
945 049B EB 00 JMP $+2
946 049D E4 21 OUT INTA01,AL
947 049F FB STI
948 04A0 EB 05F3 R CALL NOT_BUSY ; WAIT FOR CARD
949 04A3 75 1A JNZ CD_ERR ; BAD CARD
950 04A5 BA 01F7 MOV DX,HF_PORT+7
951 04A8 80 90 MOV AL,D1AG_CMD ; START DIAGNOSE
952 04AA EE OUT DX,AL
953 04AB EB 05F3 R CALL NOT_BUSY ; WAIT FOR IT TO COMPLETE
954 04AE B4 80 MOV AH,TIME_OUT
955 04B0 75 0F JNZ CD_EXIT ; TIME OUT ON DIAGNOSTIC
956 04B2 BA 01F1 MOV DX,HF_PORT+1 ; GET ERROR REGISTER
957 04B5 EC IN AL,DX
958 04B6 A2 008D R MOV @HF_ERROR,AL ; SAVE IT
959 04B9 B4 00 MOV AH,0
960 04BB DC 01 CMP AL,1 ; CHECK FOR ALL OK
961 04BD 74 02 JE SHORT CD_EXIT
962 04BF B4 20 MOV AH,BAD_CNTRLR
963 04C1 CD_ERR: MOV AH,BAD_CNTRLR
964 04C1 88 26 0074 R MOV @DISK_STATUS1,AH
965 04C5 C3 RET
966 04C6 CTLR_DIAGNOSTIC ENDP
967
968 ;-----
969 ; COMMAND ;
970 ; REPEATEDLY INPUTS DATA TILL ;
971 ; NSECTOR RETURNS ZERO ;
972 ;-----
973
974 CMD_1: CALL CHECK_DMA ; CHECK 64K BOUNDARY ERROR
975 04C6 EB 06A1 R JC CMD_ABORT
976 04CB 8B FB MOV DI,BX
977 04CD EB 085C R CALL COMMAND ; OUTPUT COMMAND
978 04D0 75 32 JNZ CMD_ABORT
979 04D2
980 04D2 EB 08C2 R CALL WAIT ; WAIT FOR DATA REQUEST INTERRUPT
981 04D5 75 2D JNZ TM_OUT ; TIME OUT
982 04D7 B9 0100 MOV CX,256D ; SECTOR SIZE IN WORDS
983 04DA BA 01F0 MOV DX,HF_PORT
984 04DD FA CLI
985 04DE FC CLD
986 04DF F3 / 6D REP ; GET THE SECTOR
987 04E1 FB STI
988 04E2 F6 46 FE 02 TEST @CMD_BLOCK+6,ECC_MODE ; CHECK FOR NORMAL INPUT
989 04E6 74 12 JZ CMD_TS
990 04E8 EB 061A R CALL WAIT_DRQ ; WAIT FOR DATA REQUEST
991 04EB 72 17 JC TM_OUT
992 04ED BA 01F0 MOV DX,HF_PORT ; GET ECC BYTES
993 04F0 D9 0004 CQ 4
994 04F3 EC IN AL,DX
995 04F4 26: 88 05 MOV ES:BYTE PTR [DI],AL ; GO SLOW FOR BOARD
996 04F7 47 INC DI
997 04F8 E2 F9 LOOP CMD_12
998 04FA EB 0630 R CALL CHECK_STATUS
999 04FD 75 05 JNZ CMD_ABORT ; ERROR RETURNED
1000 04FF FE 4E F9 DEC @CMD_BLOCK+1 ; CHECK FOR MORE
1001 0502 75 CE JNZ SHORT CMD_11
1002 0504 CMD_ABORT:
1003 0504 TM_OUT:
1004 0504 C3 RET

```

```

1005 PAGE
1006 -----
1007 | COMMAND0 |
1008 | REPEATEDLY OUTPUTS DATA TILL |
1009 | NSECTOR RETURNS ZERO |
1010 -----
1011 0505 COMMAND0:
1012 0505 EB 06A1 R CALL CHECK_DMA ; CHECK 64K BOUNDARY ERROR
1013 0506 T2 FA JC CMD_ABORT
1014 050A 08 F3 MOV S1,BX
1015 050C EB 055C R CALL COMMAND ; OUTPUT COMMAND
1016 050F T5 F3 JNZ CMD_ABORT
1017 0511 EB 061A R CALL WAIT_DRQ ; WAIT FOR DATA REQUEST
1018 0514 T2 EE JC TM_OUT ; TOO LONG
1019 0516 IE CMD_01: PUSH DS
1020 0517 06 PUSH ES ; MOVE ES TO DS
1021 0518 IF POP DS
1022 0519 B9 0100 MOV CX,2560
1023 051C BA 01F0 MOV DX,HF_PORT ; PUT THE DATA OUT TO THE CARD
1024 051F FA CLI
1025 0520 FC CLD
1026 0521 F3 J 6F REP OUTSW
1027 0523 FB STI
1028 0524 IF POP DS ; RESTORE DS
1029 0525 F4 46 FE 02 TEST #CMD_BLOCK+6,ECC_MODE ; CHECK FOR NORMAL OUTPUT
1030 0529 T4 12 JC CMD_03
1031 052B EB 061A R CALL WAIT_DRQ ; WAIT FOR DATA REQUEST
1032 052E T2 D4 JC DX,HF_PORT
1033 0530 BA 01F0 MOV DX,HF_PORT
1034 0533 B9 0004 MOV CX,4 ; OUTPUT THE ECC BYTES
1035 0536 26 8A 04 CMD_02: MOV AL,ES+BYTE PTR [S1]
1036 0539 EE OUT DX,AL
1037 053A 46 INC SI
1038 053B E2 F9 LOOP CMD_02
1039 053D CMD_03:
1040 053D EB 05C2 R CALL WAIT ; WAIT FOR SECTOR COMPLETE INTERRUPT
1041 0540 T5 C2 JNZ TM_OUT ; ERROR RETURNED
1042 0542 EB 0630 R CALL CHECK_STATUS
1043 0545 T5 BD 04 JNZ CMD_ABORT
1044 0547 F6 06 008C R 08 TEST #HF_STATUS,ST_DRQ ; CHECK FOR MORE
1045 054C T5 C8 JNZ SHORT_CMD_01
1046 054E BA 01F2 MOV DX,HF_PORT+2 ; CHECK RESIDUAL SECTOR COUNT
1047 0551 EC IN AL,DX
1048 0552 A8 FF TEST AL,OFFH
1049 0554 T4 05 JZ CMD_04 ; COUNT = 0 OK
1050 0556 C6 06 0074 R BB MOV #DISK_STATUS1,UNDEF_ERR ; OPERATION ABORTED - PARTIAL TRANSFER
1051 0558 CMD_04:
1052 055B C3 RET
1053
1054 | COMMAND |
1055 | THIS ROUTINE OUTPUTS THE COMMAND BLOCK |
1056 | OUTPUT |
1057 | BL = STATUS |
1058 | BH = ERROR REGISTER |
1059 |
1060 -----
1061
1062 055C COMMAND PROC NEAR
1063 055C 53 BX ; WAIT FOR SEEK COMPLETE AND READY
1064 055D B9 0600 MOV CX,DELAY_2 ; SET INITIAL DELAY BEFORE TEST
1065 0560 COMMAND1:
1066 0560 51 PUSH CX ; SAVE LOOP COUNT
1067 0561 EB 044F R CALL TST_RDY ; CHECK DRIVE READY
1068 0564 59 POP CX
1069 0565 T4 0B JZ COMMAND2 ; DRIVE IS READY
1070 0567 80 3E 0074 R 80 CMP #DISK_STATUS1,TIME_OUT ; TST_RDY TIMED OUT--GIVE UP
1071 056C T4 48 JZ CMD_TMOUT
1072 056E E2 F0 LOOP COMMAND1 ; KEEP TRYING FOR A WHILE
1073 0570 EB 49 JMP SHORT_COMMAND4 ; ITS NOT GOING TO GET READY
1074 0572 COMMAND2:
1075 0572 5B POP BX
1076 0573 57 PUSH DI
1077 0574 C6 06 008E R 00 MOV #HF_INT_FLAG,0 ; RESET INTERRUPT FLAG
1078 0579 FA CLI ; INHIBIT INTERRUPTS WHILE CHANGING MASK
1079 057A E4 A1 IN AL,INTB0 ; TURN ON SECOND INTERRUPT CHIP
1080 057C 24 BF AND AL,0BFH
1081 057E EB 00 JMP $+2
1082 0580 E6 A1 OUT INTB0,AL ; LET INTERRUPTS PASS THRU TO
1083 0582 E4 21 IN AL,INTA0 ; SECOND CHIP
1084 0584 24 BF AND AL,0BFH
1085 0586 EB 00 JMP $+2
1086 0588 E6 21 OUT INTA0,AL
1087 058A FB STI
1088 058B 33 FF XOR D1,D1 ; INDEX THE COMMAND TABLE
1089 058D BA 01F1 MOV DX,HF_PORT+1 ; DISK ADDRESS
1090 0590 F6 06 0076 R C0 TEST #CONTROL_BYTE,0C0H ; CHECK FOR RETRY SUPPRESSION
1091 0595 T4 11 JZ COMMAND3
1092 0597 8A 46 FE MOV #CMD_BLOCK+6 ; YES=GET OPERATION CODE
1093 059A 24 F0 AND AL,0F0H ; GET RID OF MODIFIERS
1094 059C 3C 20 CMP AL,20H ; 20H-40H IS READ, WRITE, VERIFY
1095 059E 12 08 JB COMMAND3
1096 05A0 3C 40 CMP AL,40H
1097 05A2 77 04 JA COMMAND3
1098 05A4 80 4E FE 01 OR #CMD_BLOCK+6,NO_RETRIES ; VALID OPERATION FOR RETRY SUPPRESS
1099 05A8 COMMAND3:
1100 05A8 8A 43 F8 MOV AL,[#CMD_BLOCK+D1] ; GET THE COMMAND STRING BYTE
1101 05AB EE OUT DX,AL ; GIVE IT TO CONTROLLER
1102 05AC 47 INC DI ; NEXT BYTE IN COMMAND BLOCK
1103 05AD 42 INC DX ; NEXT DISK ADAPTER REGISTER
1104 05AE 81 FA 01F8 CMP DX,HF_PORT+8 ; ALL DONE?
1105 05B2 T5 F4 JNZ COMMAND3 ; NO--GO DO NEXT ONE
1106 05B4 BF POP DI
1107 05B6 C3 RET ; ZERO FLAG IS SET
1108 05B8 CMD_TMOUT:
1109 05B8 C6 06 0074 R 20 MOV #DISK_STATUS1,BAD_CNTRL
1110 05BB COMMAND4:
1111 05BB 5B POP BX
1112 05BC 80 3E 0074 R 00 CMP #DISK_STATUS1,0 ; SET CONDITION CODE FOR CALLER
1113 05C1 C3 RET
1114 05C2 COMMAND ENDP
    
```

SECTION 5

```

1115 PAGE
1116 -----
1117 ; WAIT FOR INTERRUPT
1118 ;-----
1119 05C2 PROC NEAR
1120 05C2 FB STI ; MAKE SURE INTERRUPTS ARE ON
1121 05C3 2B C9 SUB CX,CX ; SET INITIAL DELAY BEFORE TEST
1122 05C5 F8 CLC
1123 05C6 B8 9000 MOV AX,9000H ; DEVICE WAIT INTERRUPT
1124 05C9 CD 15 INT 15H
1125 05CB 72 0F JC WT2 ; DEVICE TIMED OUT
1126 MOV BL,DELAY_1 ; SET DELAY COUNT
1127 05CD B3 25
1128 ;-----
1129 ; WAIT LOOP
1130 ;-----
1131 05CF F6 04 008E R 80 WT1: TEST 04H,INT_FLAG,80H ; TEST FOR INTERRUPT
1132 05D4 E1 F9 LOOPZ WT1 ; INTERRUPT--LETS GO
1133 05D6 75 0B JNZ WT3 ; INTERRUPT--LETS GO
1134 05D8 FE CB DEC BL ; KEEP TRYING FOR A WHILE
1135 05DA 75 F3 JZ WT1
1136 ;-----
1137 05DC C6 06 0074 R 80 WT2: MOV 0DISK_STATUS1,TIME_OUT ; REPORT TIME OUT ERROR
1138 05E1 EB 0A JMP SHORT WT4
1139 05E3 C6 06 0074 R 00 WT3: MOV 0DISK_STATUS1,0
1140 05E8 C6 06 008E R 00 WT4: MOV 04H,INT_FLAG,0
1141 05ED 80 3E 0074 R 00 WT4: CMP 0DISK_STATUS1,0 ; SET CONDITION CODE FOR CALLER
1142 05F2 C3 RET
1143 05F5 ENDP
1144 ;-----
1145 ; WAIT FOR CONTROLLER NOT BUSY
1146 ;-----
1147 NOT_BUSY PROC NEAR
1148 05F3 STI ; MAKE SURE INTERRUPTS ARE ON
1149 05F3 FB PUSH BX
1150 05F4 53 SUB CX,CX ; SET INITIAL DELAY BEFORE TEST
1151 05F5 2B C9 MOV DX,HP_PORT+7
1152 05F7 BA 01F7 MOV BL,DELAY_1
1153 05FA B3 25 IN AL,DX ; CHECK STATUS
1154 05FC EC TEST AL,ST_BUSY
1155 05FD A8 80 LOOPNZ NB1 ; NOT BUSY--LETS GO
1156 05FF E0 FB JZ NB2
1157 0601 74 0B DEC BL
1158 0603 FE CB JNZ NB1 ; KEEP TRYING FOR A WHILE
1159 0605 75 F5 MOV 0DISK_STATUS1,TIME_OUT ; REPORT TIME OUT ERROR
1160 0607 C6 06 0074 R 80 JMP SHORT NB3
1161 060C EB 05 NB2: MOV 0DISK_STATUS1,0
1162 060E C6 06 0074 R 00 NB3: POP BX
1163 0613 5B 5B CMP 0DISK_STATUS1,0 ; SET CONDITION CODE FOR CALLER
1164 0614 80 3E 0074 R 00
1165 0619 C3 RET
1166 061A ENDP
1167 ;-----
1168 ; WAIT FOR DATA REQUEST
1169 ;-----
1170 WAIT_DRQ PROC NEAR
1171 061A MOV CX,DELAY_3
1172 061A B9 0100 MOV DX,HP_PORT+7
1173 061D BA 01F7 IN AL,DX ; GET STATUS
1174 0620 EC 08 TEST AL,ST_DRQ ; WAIT FOR DRQ
1175 0621 A8 08 JNZ WQ_OK
1176 0623 75 09 LOOP WQ_1 ; KEEP TRYING FOR A SHORT WHILE
1177 0625 E2 F9 MOV 0DISK_STATUS1,TIME_OUT ; ERROR
1178 0627 C6 06 0074 R 80 STC
1179 062C F9 RET
1180 062D C3 RET
1181 062E F8 WQ_OK: CLC
1182 062F C3 RET
1183 0630 ENDP
1184 ;-----
1185 ; CHECK FIXED DISK STATUS
1186 ;-----
1187 0630 CHECK_STATUS PROC NEAR
1188 0630 E8 0642 R CALL CHECK_ST ; CHECK THE STATUS BYTE
1189 0633 75 07 JNZ CHECK_S1 ; AN ERROR WAS FOUND
1190 0635 A8 01 TEST AL,ST_ERROR ; WERE THERE ANY OTHER ERRORS
1191 0637 74 02 JZ CHECK_S1 ; NO ERROR REPORTED
1192 0639 E8 0676 R CALL CHECK_ER ; ERROR REPORTED
1193 063C CHECK_S1: CMP 0DISK_STATUS1,0 ; SET STATUS FOR CALLER
1194 063C 80 3E 0074 R 00
1195 0641 C3 RET
1196 0642 ENDP
1197 ;-----
1198 ; CHECK FIXED DISK STATUS BYTE
1199 ;-----
1200 0642 CHECK_ST PROC NEAR
1201 0642 BA 01F7 MOV DX,HP_PORT+7 ; GET THE STATUS
1202 0645 EC IN AL,DX
1203 0646 A2 008C R MOV 04H,STATUS.AL
1204 0649 B4 00 MOV AH,0
1205 064B A8 80 TEST AL,ST_BUSY ; IF STILL BUSY
1206 064D 75 1A JNZ CKST_EXIT ; REPORT OK
1207 064F B4 CC MOV AH,WRITE_FAULT
1208 0651 A8 20 TEST AL,ST_WRT_FLT ; CHECK FOR WRITE FAULT
1209 0653 75 14 JNZ CKST_EXIT
1210 0655 B4 AA MOV AH,NOT_RDY
1211 0657 A8 40 TEST AL,ST_READY ; CHECK FOR NOT READY
1212 0659 74 0E JZ CKST_EXIT
1213 065B B4 40 MOV AH,BAD_SEEK
1214 065D A8 10 TEST AL,ST_SEEK_COMPL ; CHECK FOR SEEK NOT COMPLETE
1215 065F 74 08 JZ CKST_EXIT
1216 0661 B4 11 MOV AH,DATA_CORRECTED
1217 0663 A8 04 TEST AL,ST_CORRECTD ; CHECK FOR CORRECTED ECC
1218 0665 75 02 JNZ CKST_EXIT
1219 0667 B4 00 MOV AH,0
1220 0669 CKST_EXIT: MOV 0DISK_STATUS1,AH ; SET ERROR FLAG
1221 0669 88 26 0074 R CMP AH,DATA_CORRECTED ; KEEP GOING WITH DATA CORRECTED
1222 066D 80 FC 11 JZ CKST_EXIT
1223 0670 74 03 CMP AH,0
1224 0672 80 FC 00
1225 0675 CKST_EXIT: RET
1226 0675 C3
1227 0676 CHECK_ST ENDP
    
```



```

1228 PAGE
1229 |-----|
1230 | CHECK FIXED DISK ERROR REGISTER |
1231 |-----|
1232 0676 PROC NEAR
1233 0676 BA 01F1 MOV DX,HPF_PORT+1 ; GET THE ERROR REGISTER
1234 0679 EC IN AL,DX
1235 067A A2 008D R MOV %HF_ERROR,AL
1236 067D 53 PUSH BX
1237 067E B9 0008 MOV CX,8 ; TEST ALL 8 BITS
1238 0681 D0 E0 SHL AL,1 ; MOVE NEXT ERROR BIT TO CARRY
1239 0683 72 02 JC CK2 ; FOUND THE ERROR
1240 0685 E2 FA LOOP CK1 ; KEEP TRYING
1241 0687 B8 0698 R CK2: MOV BX,OFFSET ERR_TBL ; COMPUTE ADDRESS OF
1242 068A 03 09 ADD BX,CX ; ERROR CODE
1243 068C 2E: 8A 27 MOV AH,BYTE PTR CS:[BX] ; GET ERROR CODE
1244 068F 85 26 0074 R CKEX: MOV %DISK_STATUS1,AH ; SAVE ERROR CODE
1245 0693 5B POP BX
1246 0694 80 FC 00 CMP AH,0
1247 0697 C3 RET
1248 0698 E0 ERR_TBL DB NO_ERR
1249 0699 02 40 01 BB DB BAD_ADOR_MARK,BAD_SEEK,BAD_CMD,UNDEF_ERR
1250 069D 04 B8 10 0A DB RECORD_NOT_FND,UNDEF_ERR,BAD_ECC,BAD_SECTOR
1251 06A1 CHECK_ER ENDP
1252 |-----|
1253 | CHECK DMA |
1254 | -CHECK ES:BX AND # SECTORS TO MAKE SURE THAT IT WILL |
1255 | FIT WITHOUT SEGMENT OVERFLOW. |
1256 | -ES:BX HAS BEEN REVISED TO THE FORMAT SSSS:000X |
1257 | -OK IF # SECTORS < 80H (7FH IF LONG READ OR WRITE) |
1258 | -OK IF # SECTORS = 80H (7FH) AND BX <= 00H (04H) |
1259 | -ERROR OTHERWISE |
1260 |-----|
1261 06A1 PROC NEAR
1262 06A1 50 CHECK_DMA PUSH AX ; SAVE REGISTERS
1263 06A2 B8 8000 MOV AX,8000H ; AH = MAX # SECTORS AL = MAX OFFSET
1264 06A5 F6 46 FE 02 TEST %CMD_BLOCK+6,ECC_MODE
1265 06A9 74 03 JZ CKD1
1266 06AB B8 7F04 MOV AX,7F04H ; ECC IS 4 MORE BYTES
1267 06AE 3A 66 F9 CMP AH,%CMD_BLOCK+1 ; NUMBER OF SECTORS
1268 06B1 77 06 JA CKDK ; IT WILL FIT
1269 06B3 72 07 JB CKDERR ; TOO MANY
1270 06B5 3A C3 CMP AL,BL ; CHECK OFFSET ON MAX SECTORS
1271 06B7 72 03 JB CKDERR ; ERROR
1272 06B9 F8 CKDK: CLC ; CLEAR CARRY
1273 06BA 58 POP AX
1274 06BB C3 RET ; NORMAL RETURN
1275 06BD F9 CKDERR: STC ; INDICATE ERROR
1276 06BE C6 06 0074 R 09 MOV %DISK_STATUS1,DMA_BOUNDARY
1277 06C0 58 POP AX
1278 06C2 C3 RET
1279 06C4 03 CHECK_DMA ENDP
1280 06C4 ENDP
1281 |-----|
1282 | SET UP ES:BX-> DISK PARMS |
1283 |-----|
1284 06C4 PROC NEAR
1285 06C4 2B C0 GET_VEC SUB AX,AX ; GET DISK PARAMETER ADDRESS
1286 06C6 8E C0 MOV ES,AX
1287 06C8 0E C0 ASSUME ES:ABS0
1288 06CB F6 C2 01 TEST DL,1
1289 06CD 74 07 JZ GV_0
1290 06CF 26: C4 1E 0118 R LES BX,%HF1_TBL_VEC ; ES:BX -> DRIVE PARAMETERS
1291 06D2 EB 05 JMP SHORT GV_EXIT
1292 06D4 GV_0: LES BX,%HF1_TBL_VEC ; ES:BX -> DRIVE PARAMETERS
1293 06D6 26: C4 1E 0104 R JMP SHORT GV_EXIT
1294 06D9 GV_EXIT: RET
1295 06DB C3 GET_VEC ENDP
1296 06DA ENDP
1297 06DA ENDP
1298 |-----|
1299 |--- HARDWARE INT 76H -- ( IRQ LEVEL 14 ) -----|
1300 | |
1301 | |
1302 | |
1303 | |
1304 |-----|
1305 06DA PROC NEAR
1306 06DA 50 HD_INT PUSH AX
1307 06DB 1E PUSH DS
1308 06DD E8 0000 E CALL DDS
1309 06DF C6 06 008E R FF MOV %HF_INT_FLAG,OFFH ; ALL DONE
1310 06E4 B0 20 MOV AL,001 ; NON-SPECIFIC END OF INTERRUPT
1311 06E6 E6 A0 OUT INT$00,AL ; FOR CONTROLLER #2
1312 06E8 E8 00 JMP $+2 ; WAIT
1313 06EA E6 20 OUT INT$00,AL ; FOR CONTROLLER #1
1314 06EC 1F POP DS
1315 06ED FB STI ; RE-ENABLE INTERRUPTS
1316 06EE B8 9100 MOV AX,9100H ; DEVICE POST
1317 06F1 CD 15 INT 15H ; INTERRUPT
1318 06F3 58 POP AX
1319 06F4 CF IRET ; RETURN FROM INTERRUPT
1320 06F8 HD_INT ENDP
1321 1321
1322 06F5 31 31 2F 31 36 2F DB '11/15/85' ; RELEASE MARKER
1323 38 35
1324 06FD CODE ENDS
1325 END ENDP
    
```

SECTION 5

```

1 PAGE 118,121
2 TITLE KYBD ----- 03/06/86 KEYBOARD BIOS
3 .LIST
4 0000 SEGMENT BYTE PUBLIC
5
6 PUBLIC K16
7 PUBLIC KEYBOARD_IO_1
8 PUBLIC KB_INT_1
9 PUBLIC SND_DATA
10
11 EXTRN BEEP:NEAR
12 EXTRN DDS:NEAR
13 EXTRN START_I:NEAR
14 EXTRN K6:BYTE
15 EXTRN K6L:ABS
16 EXTRN K7:BYTE
17 EXTRN K8:BYTE
18 EXTRN K10:BYTE
19 EXTRN K11:BYTE
20 EXTRN K12:BYTE
21 EXTRN K14:BYTE
22 EXTRN K15:BYTE
23
24 --- INT 16 H
25 KEYBOARD I/O
26 THESE ROUTINES PROVIDE READ KEYBOARD SUPPORT
27 INPUT
28 (AH) = 00H READ THE NEXT ASCII CHARACTER ENTERED FROM THE KEYBOARD,
29 RETURN THE RESULT IN (AL), SCAN CODE IN (AH),
30 THIS IS THE COMPATIBLE READ INTERFACE, EQUIVALENT TO THE
31 STANDARD PC OR PCAT KEYBOARD
32
33 (AH) = 01H SET THE Z FLAG TO INDICATE IF AN ASCII CHARACTER IS
34 AVAILABLE TO BE READ.
35 (ZF) = 1 -- NO CODE AVAILABLE
36 (ZF) = 0 -- CODE IS AVAILABLE (AX) = CHARACTER
37 IF (ZF) = 0, THE NEXT CHARACTER IN THE BUFFER TO BE READ IS
38 IN (AX), AND THE ENTRY REMAINS IN THE BUFFER.
39 THIS WILL RETURN ONLY PC/PCAT KEYBOARD COMPATIBLE CODES
40
41 (AH) = 02H RETURN THE CURRENT SHIFT STATUS IN AL REGISTER
42 THE BIT SETTINGS FOR THIS CODE ARE INDICATED IN THE
43 EQUATES FOR #KB_FLAG
44
45 (AH) = 03H SET TYPAMATIC RATE AND DELAY
46 (AL) = 05H
47 (BL) = TYPAMATIC RATE (BITS 5 - 7 MUST BE RESET TO 0)
48
49 REGISTER RATE REGISTER RATE
50 VALUE SELECTED VALUE SELECTED
51
52 00H 30.0 10H 7.5
53 01H 26.7 11H 6.7
54 02H 24.0 12H 6.0
55 03H 21.8 13H 5.5
56 04H 20.0 14H 5.0
57 05H 18.5 15H 4.6
58 06H 17.1 16H 4.3
59 07H 16.0 17H 4.0
60 08H 15.0 18H 3.7
61 09H 13.3 19H 3.3
62 0AH 12.0 1AH 3.0
63 0BH 10.9 1BH 2.7
64 0CH 10.0 1CH 2.6
65 0DH 9.2 1DH 2.3
66 0EH 8.6 1EH 2.1
67 0FH 8.0 1FH 2.0
68
69 (BH) = TYPAMATIC DELAY (BITS 2 - 7 MUST BE RESET TO 0)
70
71 REGISTER DELAY
72 VALUE VALUE
73
74 00H 250 ms
75 01H 500 ms
76 02H 750 ms
77 03H 1000 ms
78
79 (AH) = 05H PLACE ASCII CHARACTER/SCAN CODE COMBINATION IN KEYBOARD
80 BUFFER AS IF STRUCK FROM KEYBOARD
81 ENTRY: (CL) = ASCII CHARACTER
82 (CH) = SCAN CODE
83 EXIT: (AL) = 00H = SUCCESSFUL OPERATION
84 (AL) = 01H = UNSUCCESSFUL - BUFFER FULL
85 FLAGS: CARRY IF ERROR
86
87 (AH) = 10H EXTENDED READ INTERFACE FOR THE ENHANCED KEYBOARD,
88 OTHERWISE SAME AS FUNCTION AH=0
89
90 (AH) = 11H EXTENDED ASCII STATUS FOR THE ENHANCED KEYBOARD.
91 OTHERWISE SAME AS FUNCTION AH=1
92
93 (AH) = 12H RETURN THE EXTENDED SHIFT STATUS IN AX REGISTER
94 AL = BITS FROM KB_FLAG, AH = BITS FOR LEFT AND RIGHT
95 CTL AND ALT KEYS FROM KB_FLAG_1 AND KB_FLAG_3
96
97 OUTPUT
98 AS NOTED ABOVE, ONLY (AX) AND FLAGS CHANGED
99 ALL REGISTERS RETAINED
100
101 ASSUME CS:CODE,DS:DATA
102
103 0000 KEYBOARD_IO_1 PROC FAR ; >>> ENTRY POINT FOR ORG 0E82EH
104 0000 FB STI ; INTERRUPTS BACK ON
105 0001 IE PUSH DS ; SAVE CURRENT DS
106 0002 S3 PUSH BX ; SAVE BX TEMPORARILY
107 0003 S1 PUSH CX ; SAVE CX TEMPORARILY
108 0004 E8 0000 E CALL DDS ; ESTABLISH POINTER TO DATA REGION
109 0007 0A E4 OR AH,AH ; CHECK FOR (AH) = 00H
110 0009 74 2D JZ K1 ; ASCII READ
111 000B FE CC DEC AH ; CHECK FOR (AH) = 01H
112 000D 74 3E JZ K2 ; ASCII STATUS
113 000F FE CC DEC AH ; CHECK FOR (AH) = 02H
114 0011 74 68 JZ K3 ; SHIFT STATUS

```

```

115 0013 FE CC          DEC AH          ; CHECK FOR (AH)= 03H
116 0015 74 CC          JZ K300         ; SET TYPAMATIC RATE/DELAY
117 0017 80 EC 02      SUB AH,2        ; CHECK FOR (AH)= 05H
118 001A 75 03         JNZ K101        ;
119 001C E9 00A4 R     JMP K500        ; KEYBOARD WRITE
120 001F 80 EC 0B     SUB AH,11       ; AH = 10
121 0022 74 0C         JZ K1E          ; EXTENDED ASCII READ
122 0024 FE CC          DEC AH          ; CHECK FOR (AH)= 11H
123 0026 74 1A         JZ K2E          ; EXTENDED_ASCII_STATUS
124 0028 FE CC          DEC AH          ; CHECK FOR (AH)= 15H
125 002A 74 39         JZ K3E          ; EXTENDED_SHIFT_STATUS
126 002C                K10_EXIT:
127 002C 59           POP CX          ; RECOVER REGISTER
128 002D 5B           POP BX          ; RECOVER REGISTER
129 002E 1F           POP DS          ; RECOVER SEGMENT
130 002F CF           IRET          ; INVALID COMMAND
131
132
133
134 0030 E8 00C7 R     K1E: CALL K15          ; GET A CHARACTER FROM THE BUFFER (EXTENDED)
135 0032 E8 0125 R     CALL K10_E_XLAT ; ROUTINE TO XLATE FOR EXTENDED CALLS
136 0036 EB F4         JMP K10_EXIT    ; GIVE IT TO THE CALLER
137
138 0038 E8 00C7 R     K1:  CALL K15          ; GET A CHARACTER FROM THE BUFFER
139 003B E8 0130 R     CALL K10_S_XLAT ; ROUTINE TO XLATE FOR STANDARD CALLS
140 003E 72 F8         JC K1           ; CARRY SET MEANS THROW CODE AWAY
141 0040 EB EA         JMP K1A        ; RETURN TO CALLER
142
143
144
145 0042 E8 0103 R     K2E: CALL K25          ; TEST FOR CHARACTER IN BUFFER (EXTENDED)
146 0045 74 18         JZ K2B         ; RETURN IF BUFFER EMPTY
147 0047 9C           PUSHF          ; SAVE ZF FROM TEST
148 0048 E8 0125 R     CALL K10_E_XLAT ; ROUTINE TO XLATE FOR EXTENDED CALLS
149 004B EB 11         JMP SHORT K2A   ; GIVE IT TO THE CALLER
150
151 004D E8 0103 R     K2:  CALL K25          ; TEST FOR CHARACTER IN BUFFER
152 0050 74 0D         JZ K2B         ; RETURN IF BUFFER EMPTY
153 0052 9C           PUSHF          ; SAVE ZF FROM TEST
154 0053 E8 0130 R     CALL K10_S_XLAT ; ROUTINE TO XLATE FOR STANDARD CALLS
155 0056 73 06         JNC K2A        ; CARRY CLEAR MEANS PASS VALID CODE
156 0058 9D           POPF          ; INVALID CODE FOR THIS TYPE OF CALL
157 0059 E8 00C7 R     CALL K15        ; THROW THE CHARACTER AWAY
158 005C EB EF         JMP K2         ; GO LOOK FOR NEXT CHAR, IF ANY
159
160 005E 9D           POPF          ; RESTORE ZF FROM TEST
161 005F 59           POP CX          ; RECOVER REGISTER
162 0060 5B           POP BX          ; RECOVER REGISTER
163 0061 1F           POP DS          ; RECOVER SEGMENT
164 0062 CA 0002      RET 2          ; THROW AWAY FLAGS
165
166
167
168 0065
169 0065 8A 26 0018 R   K3E: MOV AH,0KB_FLAG_1 ; GET THE EXTENDED SHIFT STATUS FLAGS
170 0069 80 E4 04      AND AH,SYS_SHIFT ; GET SYSTEM SHIFT KEY STATUS
171 006C B1 05         MOV CL,S        ; MASK ALL BUT SYS KEY BIT
172 006E D2 E4         SHL AH,CL       ; SHIFT THE SYSTEM KEY BIT OVER TO
173 0070 A0 0018 R     MOV AL,0KB_FLAG_1 ; BIT 7 POSITION
174 0073 24 73         AND AL,01110011B ; GET SHIFT STATES BACK
175 0075 0A E0         OR AH,AL        ; ELIMINATE SYS_SHIFT, HOLD_STATE, AND INS_SHIFT
176 0077 A0 0096 R     MOV AL,0KB_FLAG_3 ; MERGE THE REMAINING BITS INTO AH
177 007A 24 0C         AND AL,00001100B ; GET RIGHT CTL AND ALT
178 007C 0A E0         OR AH,AL        ; ELIMINATE LC EO AND LC EI
179 007E A0 0017 R     MOV AL,0KB_FLAG ; OR THE SHIFT FLAGS TOGETHER
180 0081 EB A9         JMP K3         ; GET THE SHIFT STATUS FLAGS
181
182
183
184 0083 3C 05         CMP AL,5        ; SET TYPAMATIC RATE AND DELAY
185 0085 75 A5         JNE K10_EXIT    ; CORRECT FUNCTION CALL?
186 0087 F6 C5 E0      TEST BL,0E0H    ; NO, RETURN
187 008A 75 A0         JNZ K10_EXIT    ; TEST FOR OUT-OF-RANGE RATE
188 008C F6 C7 FC      TEST BH,0FCH    ; RETURN IF SO
189 008F 75 98         JNZ K10_EXIT    ; TEST FOR OUT-OF-RANGE DELAY
190 0091 B0 F3         MOV AL,RB_TYPA_RD ; COMMAND FOR TYPAMATIC RATE/DELAY
191 0093 E8 064B R     CALL SND_DATA   ; SEND TO KEYBOARD
192 0096 B9 0005      MOV CX,5        ; SHIFT COUNT
193 0099 D2 E7         SHL BH,CL       ; SHIFT DELAY OVER
194 009B 8A C3         MOV AL,BL       ; PUT IN RATE
195 009D 0A C7         OR AL,BH        ; AND DELAY
196 009F E8 064B R     CALL SND_DATA   ; SEND TO KEYBOARD
197 00A2 EB 88         JMP K10_EXIT    ; RETURN TO CALLER
198
199
200
201 00A4 86           K500: PUSH SI        ; WRITE TO KEYBOARD BUFFER
202 00A5 FA           CLI            ; SAVE SI
203 00A6 8B 1E 001C R   MOV BX,0BUFFER_TAIL ; GET THE "IN TO" POINTER TO THE BUFFER
204 00AA 8B F3         MOV SI,BX       ; SAVE A COPY IN CASE BUFFER NOT FULL
205 00AC E8 0168 R     CALL K4         ; BUMP THE POINTER TO SEE IF BUFFER IS FULL
206 00AF 3B 1E 001A R   CMP BX,0BUFFER_HEAD ; WILL THE BUFFER OVERRUN IF WE STORE THIS?
207 00B3 74 0B         JE K502         ; YES - INFORM CALLER OF ERROR
208 00B5 89 0C         MOV [SI],CX     ; NO - PUT THE ASCII/SCAN CODE INTO BUFFER
209 00B7 89 1E 001C R   MOV 0BUFFER_TAIL,BX ; ADJUST "IN TO" POINTER TO REFLECT CHANGE
210 00BB 2A C0         SUB AL,AL       ; TELL CALLER THAT OPERATION WAS SUCCESSFUL
211 00BD EB 03 90      JMP K504        ; SUB INSTRUCTION ALSO RESETS CARRY FLAG
212 00C0
213 00C0 B0 01         K502: MOV AL,01H      ; BUFFER FULL INDICATION
214 00C2
215 00C2 F9           K504: STI
216 00C3 8E           POP SI          ; RECOVER SI
217 00C4 EB 002C R     JMP K10_EXIT    ; RETURN TO CALLER WITH STATUS IN AL
218
219 00C7          KEYBOARD_IO_1 ENDP

```

SECTION 5

```

220 PAGE
221 |----- READ THE KEY TO FIGURE OUT WHAT TO DO -----
222
223 00C7 K15 PROC NEAR
224 00C7 8B 1E 001A R MOV BX,#BUFFER_HEAD ; GET POINTER TO HEAD OF BUFFER
225 00CB 3B 1E 001C R CMP BX,#BUFFER_TAIL ; TEST END OF BUFFER
226 00CF 75 07 JNE K1U ; IF ANYTHING IN BUFFER DONT DO INTERRUPT
227
228 00D1 8B 9002 MOV AX,09002H ; MOVE IN WAIT CODE & TYPE
229 00D4 CD 15 INT 15H ; PERFORM OTHER FUNCTION
230 00D6 K1T: ; ASCII READ
231 00D6 FB STI ; INTERRUPTS BACK ON DURING LOOP
232 00D7 90 NOP ; ALLOW AN INTERRUPT TO OCCUR
233 00D8 FA CLI ; INTERRUPTS BACK OFF
234 00D9 8B 1E 001A R MOV BX,#BUFFER_HEAD ; GET POINTER TO HEAD OF BUFFER
235 00DD 3B 1E 001C R CMP BX,#BUFFER_TAIL ; TEST END OF BUFFER
236 00E1 53 PUSH ; SAVE ADDRESS
237 00E2 9C PUSHF ; SAVE FLAG
238 00E3 E8 06D8 R CALL MAKE_LED ; GO GET MODE INDICATOR DATA BYTE
239 00E6 8A 1E 0097 R MOV BL,#KB_FLAG_2 ; GET PREVIOUS BITS
240 00EA 32 D8 XOR BL,AL ; SEE IF ANY DIFFERENT
241 00EC 80 E3 07 AND BL,07H ; ISOLATE INDICATOR BITS
242 00EF 74 04 JZ K1V ; IF NO CHANGE BYPASS UPDATE
243
244 00F1 E6 069A R CALL SND_LED1 ; GO TURN ON MODE INDICATORS
245 00F4 FA CLI ; DISABLE INTERRUPTS
246 00F5 9D POPF ; RESTORE FLAGS
247 00F6 66 POPS ; RESTORE ADDRESS
248 00F7 74 DD JZ K1T ; LOOP UNTIL SOMETHING IN BUFFER
249
250 00F9 8B 07 MOV AX,[BX] ; GET SCAN CODE AND ASCII CODE
251 00FB E8 0168 R CALL K4 ; MOVE POINTER TO NEXT POSITION
252 00FE 89 1E 001A R MOV #BUFFER_HEAD,BX ; STORE VALUE IN VARIABLE
253 0102 C3 RET ; RETURN
254 0103
255
256 |----- READ THE KEY TO SEE IF ONE IS PRESENT -----
257
258
259 0103 K2S PROC NEAR
260 0103 FA CLI ; INTERRUPTS OFF
261 0104 8B 1E 001A R MOV BX,#BUFFER_HEAD ; GET HEAD POINTER
262 0108 3B 1E 001C R CMP BX,#BUFFER_TAIL ; IF EQUAL (Z=1) THEN NOTHING THERE
263 010C 8B 07 MOV AX,[BX] ; GET SCAN CODE AND ASCII CODE
264 010E 9C PUSHF ; SAVE FLAGS
265
266 010F 80 PUSH AX ; SAVE CODE
267 0110 E8 06D8 R CALL MAKE_LED ; GO GET MODE INDICATOR DATA BYTE
268 0113 8A 1E 0097 R MOV BL,#KB_FLAG_2 ; GET PREVIOUS BITS
269 0117 32 D8 XOR BL,AL ; SEE IF ANY DIFFERENT
270 0119 80 E3 07 AND BL,07H ; ISOLATE INDICATOR BITS
271 011C 74 03 JZ K2T ; IF NO CHANGE BYPASS UPDATE
272
273 011E E6 0687 R CALL SND_LED ; GO TURN ON MODE INDICATORS
274 0121 58 POP AX ; RESTORE CODE
275 0122 9D POPF ; RESTORE FLAGS
276 0123 FB STI ; INTERRUPTS BACK ON
277 0124 C3 RET ; RETURN
278 0125
279
280
281 |----- ROUTINE TO TRANSLATE SCAN CODE PAIRS FOR EXTENDED CALLS
282
283 0125 K10_E_XLAT:
284 0128 3C F0 CMP AL,0F0h ; IS IT ONE OF THE FILL-INs?
285 0127 75 06 JNE K10_E_RET ; NO, PASS IT ON
286 0129 0A E4 OR AH,AH ; AH = 0 IS SPECIAL CASE
287 012B 74 02 JZ K10_E_RET ; PASS THIS ON UNCHANGED
288 012D 32 C0 XOR AL,AL ; OTHERWISE SET AL = 0
289 012F K10_E_RET:
290 012F C3 RET ; GO BACK
291
292
293 |----- ROUTINE TO TRANSLATE SCAN CODE PAIRS FOR STANDARD CALLS
294
295 0130 K10_S_XLAT:
296 0130 80 FC ED CMP AH,0E0h ; IS IT KEYPAD ENTER OR / ?
297 0133 75 12 JNE K10_S2 ; NO, CONTINUE
298 0136 3C 0D CMP AL,0Dh ; REPAD ENTER CODE?
299 0137 74 09 JE K10_S1 ; YES, MESSAGE A BIT
300 0139 3C 0A CMP AL,0Ah ; CTRL KEYPAD ENTER CODE?
301 013B 74 05 JE K10_S1 ; YES, MESSAGE THE SAME
302 013D B4 35 MOV AH,35h ; NO, MUST BE KEYPAD /
303 013F EB 28 90 JMP K10_USE ; GIVE TO CALLER
304 0142 B4 1C MOV AH,1Ch ; CONVERT TO COMPATIBLE OUTPUT
305 0144 EB 1E 90 JMP K10_USE ; GIVE TO CALLER
306
307 0147 80 FC B4 K10_S2: CMP AH,84h ; IS IT ONE OF THE EXTENDED ONES?
308 014A 77 1A JA K10_DIS ; YES, THROW AWAY AND GET ANOTHER CHAR
309
310 014C 3C F0 CMP AL,0F0h ; IS IT ONE OF THE FILL-INs?
311 014E 75 07 JNE K10_S3 ; NO, TRY LAST TEST
312 0150 0A E4 OR AH,AH ; AH = 0 IS SPECIAL CASE
313 0152 74 10 JZ K10_USE ; PASS THIS ON UNCHANGED
314 0154 EB 10 90 JMP K10_DIS ; THROW AWAY THE REST
315
316 0157 3C E0 CMP AL,0E0h ; IS IT AN EXTENSION OF A PREVIOUS ONE?
317 0159 75 09 JNE K10_USE ; NO, MUST BE A STANDARD CODE
318 015B 0A E4 OR AH,AH ; AH = 0 IS SPECIAL CASE
319 015D 74 05 JZ K10_USE ; JUMP IF AH = 0
320 015F 32 C0 XOR AL,AL ; CONVERT TO COMPATIBLE OUTPUT
321 0161 EB 01 90 JMP K10_USE ; PASS IT ON TO CALLER
322
323 0164 K10_USE:
324 0164 F8 CLC ; CLEAR CARRY TO INDICATE GOOD CODE
325 0165 C3 RET ; RETURN
326 0166
327 0166 F9 K10_DIS:
328 0167 C3 STC ; SET CARRY TO INDICATE DISCARD CODE
329 0167 C3 RET ; RETURN

```

```

329                                     PAGE
330                                     ;-----
331                                     ; INCREMENT BUFFER POINTER ROUTINE
332                                     ;-----
333
334 0168 K4 PROC NEAR
335 0168 43 INC BX ; MOVE TO NEXT WORD IN LIST
336 0169 43 INC BX
337
338 016A 3B 1E 0082 R CMP BX,#BUFFER_END ; AT END OF BUFFER?
339 016E 75 04 JNE KB ; NO, CONTINUE
340 0170 8B 1E 0080 R MOV BX,#BUFFER_START ; YES, RESET TO BUFFER BEGINNING
341 0174 C3 RET
342 0175 K4 ENDP
343
344 ;--- HARDWARE INT 09 H -- ( IRQ LEVEL 1 ) ---
345 ;
346 ; KEYBOARD INTERRUPT ROUTINE
347 ;
348 ;-----
349
350 0175 KB_INT_1 PROC FAR
351 0175 FB STI ; ENABLE INTERRUPTS
352 0176 55 PUSH BP
353 0177 50 PUSH AX
354 0178 53 PUSH BX
355 0179 51 PUSH CX
356 017A 52 PUSH DX
357 017B 56 PUSH SI
358 017C 57 PUSH DI
359 017D 1E PUSH DS
360 017E 06 PUSH ES
361 017F FC CLD ; FORWARD DIRECTION
362 0180 E8 0000 E CALL DDS ; SET UP ADDRESSING
363
364 ;----- WAIT FOR KEYBOARD DISABLE COMMAND TO BE ACCEPTED
365
366 0183 80 AD MOV AL,DIS_KBD ; DISABLE THE KEYBOARD COMMAND
367 0185 E8 063C R CALL SHIP_IT ; EXECUTE DISABLE
368 0188 FA CLI ; DISABLE INTERRUPTS
369 0189 2B C9 SUB CX,CX ; SET MAXIMUM TIMEOUT
370 018B KB_INT_01:
371 018B E4 64 IN AL,STATUS_PORT ; READ ADAPTER STATUS
372 018D A8 02 TEST AL,INPT_BUF_FULL ; CHECK INPUT BUFFER FULL STATUS BIT
373 018F E0 FA LOOPNZ KB_INT_01 ; WAIT FOR COMMAND TO BE ACCEPTED
374
375 ;----- READ CHARACTER FROM KEYBOARD INTERFACE
376
377 0191 E4 60 IN AL,PORT_A ; READ IN THE CHARACTER
378
379 ;----- SYSTEM HOOK INT 15H - FUNCTION 4FH (ON HARDWARE INTERRUPT LEVEL 9H)
380
381 0193 B4 4F MOV AH,04FH ; SYSTEM INTERCEPT - KEY CODE FUNCTION
382 0195 F9 STC ; SET CY= 1 (IN CASE OF IRET)
383 0196 CD 15 INT 15H ; CASSETTE CALL (AL)= KEY SCAN CODE
384 ; RETURNS CY= 1 FOR INVALID FUNCTION
385 0198 72 03 JC KB_INT_02 ; CONTINUE IF CARRY FLAG SET ((AL)=CODE)
386 019A E9 03A0 R JMP K26 ; EXIT IF SYSTEM HANDLED SCAN CODE
387 ; EXIT HANDLES HARDWARE E01 AND ENABLE
388
389 ;----- CHECK FOR A RESEND COMMAND TO KEYBOARD
390
391 019D KB_INT_02:
392 019D FB STI ; (AL)= SCAN CODE
393 019E 3C FE CMP AL,KB_RESEND ; ENABLE INTERRUPTS AGAIN
394 01A0 74 0D JE KB_INT_4 ; IS THE INPUT A RESEND
395 ; GO IF RESEND
396
397 ;----- CHECK FOR RESPONSE TO A COMMAND TO KEYBOARD
398
399 01A2 3C FA CMP AL,KB_ACK ; IS THE INPUT AN ACKNOWLEDGE
400 01A4 75 12 JNZ KB_INT_2 ; GO IF NOT
401
402 ;----- A COMMAND TO THE KEYBOARD WAS ISSUED
403
404 01A6 FA CLI ; DISABLE INTERRUPTS
405 01A7 80 0E 0097 R 10 OR #KB_FLAG_2,KB_FA ; INDICATE ACK RECEIVED
406 01AC E9 03A0 R JMP K26 ; RETURN IF NOT (ACK RETURNED FOR DATA)
407
408 ;----- RESEND THE LAST BYTE
409
410 01AF KB_INT_4:
411 01AF FA CLI ; DISABLE INTERRUPTS
412 01B0 80 0E 0097 R 20 OR #KB_FLAG_2,KB_FE ; INDICATE RESEND RECEIVED
413 01B5 E9 03A0 R JMP K26 ; RETURN IF NOT (ACK RETURNED FOR DATA)
414
415 ;----- UPDATE MODE INDICATORS IF CHANGE IN STATE
416
417 01B8 KB_INT_2:
418 01B8 50 PUSH AX ; SAVE DATA IN
419 01B9 E8 06D8 R CALL MAKE_LED ; GO GET MODE INDICATOR DATA BYTE
420 01BC 5A 1E 0097 R MOV BL,#RB_FLAG_2 ; GET PREVIOUS BITS
421 01C0 32 D8 XOR BL,AL ; SEE IF ANY DIFFERENT
422 01C2 80 E3 07 AND BL,KB_LEDS ; ISOLATE INDICATOR BITS
423 01C5 74 03 JZ UP0 ; IF NO CHANGE BYPASS UPDATE
424 01C7 E8 06B7 R CALL SND_LED ; GO TURN ON MODE INDICATORS
425 01CA 58 POP AX ; RESTORE DATA IN

```

SECTION 5

```

426 PAGE
427 |-----
428 | START OF KEY PROCESSING
429 |-----
430
431 01CB 8A E0 MOY AH,AL ; SAVE SCAN CODE IN AH ALSO
432
433 |----- TEST FOR OVERRUN SCAN CODE FROM KEYBOARD
434
435 01CD 3C FF CMP AL,KB_OVER_RUN ; IS THIS AN OVERRUN CHAR?
436 01CF 75 03 JNZ K16 ; NO, TEST FOR SHIFT KEY
437 01D1 E9 062D R JMP K62 ; BUFFER_FULL_BEEP
438
439 01D4 0E K16: PUSH CS
440 01D5 07 ES ; ESTABLISH ADDRESS OF TABLES
441 01D6 8A 3E 0096 R MOY BH,*KB_FLAG_3 ; LOAD FLAGS FOR TESTING
442
443 |----- TEST TO SEE IF A READ_ID IS IN PROGRESS
444
445 01DA F6 C7 C0 TEST BH,RD_ID+LC_AB ; ARE WE DOING A READ ID?
446 01DD 74 34 JZ NOT_ID ; CONTINUE IF NOT
447 01DF 79 10 JNS TST_ID_2 ; IS THE RD_ID FLAG ON?
448 01E1 3C AB CMP AL,ID_1 ; IS THIS THE 1ST ID CHARACTER?
449 01E3 74 0E JNE RST_RD ; NO
450 01E5 80 0E 0096 R 40 OR *KB_FLAG_3,LC_AB ; INDICATE 1ST ID WAS OK
451 01EA AND RST_RD_ID: ;
452 01EA 80 26 0096 R 7F AND *KB_FLAG_3,NOT_RD_ID ; RESET THE READ ID FLAG
453 01EF EB 1F JMP SHORT_ID_EX ; AND EXIT
454
455 01F1 TST_ID_2: ;
456 01F1 80 26 0096 R BF AND *KB_FLAG_3,NOT_LC_AB ; RESET FLAG
457 01F6 3C 54 CMP AL,ID_2A ; IS THIS THE 2ND ID CHARACTER?
458 01F8 74 11 JE KX_BIT ; JUMP IF SO
459 01FA 3C 41 CMP AL,ID_2 ; IS THIS THE 2ND ID CHARACTER?
460 01FC 75 12 JNE ID_EX ; LEAVE IF NOT
461
462 |----- A READ ID SAID THAT IT WAS ENHANCED KEYBOARD
463
464 01FE F6 C7 20 TEST BH,SET_NUM_LK ; SHOULD WE SET NUM LOCK?
465 0201 74 08 JZ KX_BIT ; EXIT IF NOT
466 0203 80 0E 0017 R 20 OR *KB_FLAG_NUM_STATE ; FORCE NUM LOCK ON
467 0205 EB 0687 R CALL SND_LED ; GO SET THE NUM LOCK INDICATOR
468 0208 80 0E 0096 R 10 OR *KB_FLAG_3,KBX ; INDICATE ENHANCED KEYBOARD WAS FOUND
469 0210 E9 03A0 R JMP K26 ; EXIT
470
471 0213 NOT_ID: ;
472 0213 3C E0 CMP AL,MC_E0 ; IS THIS THE GENERAL MARKER CODE?
473 0215 75 07 JNE TEST_E1 ; NO
474 0217 80 0E 0096 R 12 OR *KB_FLAG_3,LC_E0+KBX ; SET FLAG BIT, SET KBX, AND
475 021C EB 09 JMP SHORT_EXIT ; THROW AWAY THIS CODE
476
477 021E TEST_E1: ;
478 021E 3C E1 CMP AL,MC_E1 ; IS THIS THE PAUSE KEY?
479 0220 75 08 JNE NOT_HC ; NO
480 0222 80 0E 0096 R 11 OR *KB_FLAG_3,LC_E1+KBX ; SET FLAG, PAUSE KEY MARKER CODE
481 0227 E9 03A6 R JMP K26A ; THROW AWAY THIS CODE
482
483 022A NOT_HC: ;
484 022A 24 7F AND *KB_FLAG_3,NOT_HC ; TURN OFF THE BREAK BIT
485 022C F6 C7 02 TEST BH,LC_E0 ; LAST CODE THE E0 MARKER CODE?
486 022F 74 0C JZ NOT_LC_E0 ; JUMP IF NOT
487
488 0231 MOV CX,2 ; LENGTH OF SEARCH
489 0234 BF 0006 E MOV DI,OFFSET K6+6 ; IS THIS A SHIFT KEY?
490 0237 F2/ AE REPNE SCASB ; CHECK IT
491 0239 75 65 JNE K16A ; NO, CONTINUE KEY PROCESSING
492 023B EB 49 JMP SHORT_K16B ; YES, THROW AWAY & RESET FLAG
493
494 023D NOT_LC_E0: ;
495 023D F6 C7 01 TEST BH,LC_E1 ; LAST CODE THE E1 MARKER CODE?
496 0240 74 1D JZ T_SYS_KEY ; JUMP IF NOT
497
498 0242 MOV CX,4 ; LENGTH OF SEARCH
499 0245 BF 0004 E MOV DI,OFFSET K6+4 ; IS THIS AN ALT, CTL, OR SHIFT?
500 0248 F2/ AE REPNE SCASB ; CHECK IT
501 024A 74 DB JE EXIT ; THROW AWAY IF SO
502
503 024C CMP AL,NUM_KEY ; IS IT THE PAUSE KEY?
504 024E 75 36 JNE K16B ; NO, THROW AWAY & RESET FLAG
505 0250 F6 C4 80 TEST AH,80H ; YES, IS IT THE BREAK OF THE KEY?
506 0253 75 31 JNZ K16B ; YES, THROW THIS AWAY, TOO
507 0255 F6 06 0018 R 08 TEST *KB_FLAG_1,HOLD_STATE ; NO, ARE WE PAUSED ALREADY?
508 025A 75 2A JNZ K16B ; YES, THROW AWAY
509 025C E9 04DB R JMP K39P ; NO, THIS IS THE REAL PAUSE STATE
510
511 |----- TEST FOR SYSTEM KEY
512
513 T_SYS_KEY: ;
514 025F CMP AL,SYS_KEY ; IS IT THE SYSTEM KEY?
515 0261 75 3D JNE K16A ; CONTINUE IF NOT
516
517 0263 F6 C4 80 TEST AH,080H ; CHECK IF THIS A BREAK CODE
518 0266 75 21 JNZ K16C ; DON'T TOUCH SYSTEM INDICATOR IF TRUE
519
520 0268 TEST *KB_FLAG_1,SYS_SHIFT ; SEE IF IN SYSTEM KEY HELD DOWN
521 026D 75 17 JNZ K16B ; IF YES, DON'T PROCESS SYSTEM INDICATOR
522
523 026F OR *KB_FLAG_1,SYS_SHIFT ; INDICATE SYSTEM KEY DEPRESSED
524 0274 80 20 MOV AL,E01 ; END OF INTERRUPT COMMAND
525 0276 E6 20 MOV OUT,020H,AL ; SEND COMMAND TO INTERRUPT CONTROL PORT
526
527 0278 MOV AL,ENA_KBD ; INSURE KEYBOARD IS ENABLED
528 027A EB 063C R CALL SHIP_IT ; EXECUTE ENABLE
529 027D 88 8500 MOV AX,08500H ; FUNCTION VALUE FOR MAKE OF SYSTEM KEY
530 0280 STI ; MAKE SURE INTERRUPTS ENABLED
531 0281 CD 15 INT 15H ; USER INTERRUPT
532 0283 E9 03AF R JMP K27A ; END PROCESSING
533
534 0286 K16B: JMP K26 ; IGNORE SYSTEM KEY
535
536 0289 K16C: AND *KB_FLAG_1,NOT_SYS_SHIFT ; TURN OFF SHIFT KEY HELD DOWN
537 028E 80 20 MOV AL,E01 ; END OF INTERRUPT COMMAND
538 0290 E6 20 MOV OUT,020H,AL ; SEND COMMAND TO INTERRUPT CONTROL PORT

```

```

540
541 0292 B0 AE          MOV     AL,ENA_KBD          ; INTERRUPT-RETURN-NO-EOI
542 0294 E8 063C R     CALL  SHIP_IT             ; INSURE KEYBOARD IS ENABLED
543 0297 B8 8501       MOV     AX,08501H         ; EXECUTE ENABLE
544 029A F8            STI                     ; FUNCTION VALUE FOR BREAK OF SYSTEM KEY
545 029B CD 15        INT     15H              ; MAKE SURE INTERRUPTS ENABLED
546 029D E9 03AF R     JMP     K27A             ; USER INTERRUPT
547
548
549
550 02A0 8A 1E 0017 R   K16A: MOV     BL,*KB_FLAG      ; PUT STATE FLAGS IN BL
551 02A4 BF 0000 E     MOV     DI,OFFSET K6     ; SHIFT KEY TABLE
552 02A7 B9 0000 E     MOV     CX,OFFSET K6L    ; LENGTH
553 02AA F2/ AE       REPNE  SCASB            ; LOOK THROUGH THE TABLE FOR A MATCH
554 02AC 8A C4        MOV     AL,AH            ; RECOVER SCAN CODE
555 02AE 74 03        JBE    K17              ; JUMP IF MATCH FOUND
556 02B0 E9 038C R     JMP     K25              ; IF NO MATCH, THEN SHIFT NOT FOUND
557
558
559
560 02B3 81 EF 0001 E   K17:  SUB     DI,OFFSET K6+1  ; ADJUST PTR TO SCAN CODE MTCX
561 02B7 2E: 8A A5 0000 E  MOV     AH,CS:[DI]       ; GET MASK INTO AH
562 02BC B1 02        MOV     CL,2             ; SET UP COUNT FOR FLAG SHIFTS
563 02BE A8 80        TEST    AL,80H          ; TEST FOR BREAK KEY
564 02C0 74 03        JZ     K17C             ;
565 02C2 EB 78 90     JMP     K23              ; JUMP IF BREAK
566
567
568
569 02C5 80 FC 10     K17C: CMP     AH,SCROLL_SHIFT ;
570 02C8 73 21        JAE    K18              ; IF SCROLL SHIFT OR ABOVE, TOGGLE KEY
571
572
573
574 02CA 08 26 0017 R   OR     *KB_FLAG,AH      ; TURN ON SHIFT BIT
575 02CC F6 C4 0C     TEST    AH,CTL_SHIFT+ALT_SHIFT ; IS IT ALT OR CTRL?
576 02D1 75 03        JNZ    K17D             ; YES, MORE FLAGS TO SET
577 02D3 E9 03A0 R     JMP     K26              ; NO, INTERRUPT_RETURN
578 02D6 F6 C7 02     TEST    BH,LC_E0        ; IS THIS ONE OF THE NEW KEYS?
579 02D9 74 07        JZ     K17E             ; NO, JUMP
580 02DB 08 26 0096 R  OR     *KB_FLAG_3,AH    ; SET BITS FOR RIGHT CTRL, ALT
581 02DF E9 03A0 R     JMP     K26              ; INTERRUPT_RETURN
582 02E2 D2 EC        SHR    AH,CL            ; MOVE FLAG BITS TWO POSITIONS
583 02E4 08 26 0018 R  OR     *KB_FLAG_1,AH    ; SET BITS FOR LEFT CTRL, ALT
584 02E8 E9 03A0 R     JMP     K26              ; INTERRUPT_RETURN
585
586
587
588 02EB F6 C3 04     K18:  TEST    BL,CTL_SHIFT    ; SHIFT-TOGGLE
589 02EE 74 03        JZ     K18A             ; CHECK CTL SHIFT STATE
590 02F0 E9 038C R     JMP     K25              ; JUMP IF NOT CTL STATE
591 02F3 3C 52        CMP     AL,INS_KEY      ; CHECK FOR INSERT KEY
592 02F5 75 21        JNE    K18B             ; JUMP IF NOT INSERT KEY
593 02F7 F6 C3 08     TEST    BL,ALT_SHIFT    ; CHECK FOR ALTERNATE SHIFT
594 02FA 74 03        JZ     K18B             ; JUMP IF NOT ALTERNATE SHIFT
595 02FC E9 038C R     JMP     K25              ; JUMP IF ALTERNATE SHIFT
596 02FF F6 C7 02     TEST    BH,LC_E0        ; IS THIS THE NEW INSERT KEY?
597 0302 75 14        JNZ    K22              ; YES, THIS ONE'S NEVER A "0"
598 0304 F6 C3 20     TEST    BL,NUM_STATE    ; CHECK FOR BASE STATE
599 0307 75 0A        JNZ    K19              ; JUMP IF NUM LOCK IS ON
600 0309 F6 C3 03     TEST    BL,LEFT_SHIFT+RIGHT_SHIFT ; TEST FOR SHIFT STATE
601 030C 74 0A        JZ     K22              ; JUMP IF BASE STATE
602 030E 8A E0        MOV     AH,AL           ; PUT SCAN CODE BACK IN AH
603 0310 EB 7A 90     JMP     K25              ; NUMERAL "0", STDNRD. PROCESSING
604
605
606 0313 F6 C3 03     K21:  TEST    BL,LEFT_SHIFT+RIGHT_SHIFT ; MIGHT BE NUMERIC
607 0316 74 F6        JZ     K20              ; IS NUMERIC, STD. PROC.
608
609 0318
610 031B 84 26 0018 R   TEST    AH,*KB_FLAG_1  ; SHIFT TOGGLE KEY HIT; PROCESS IT
611 031C 74 03        JZ     K22A             ; IS KEY ALREADY DEPRESSED?
612 031E E9 03A0 R     JMP     K26              ; JUMP IF KEY ALREADY DEPRESSED
613 0321 08 26 0018 R  OR     *KB_FLAG_1,AH    ; INDICATE THAT THE KEY IS DEPRESSED
614 0325 30 26 0017 R  XOR    *KB_FLAG,AH     ; TOGGLE THE SHIFT STATE
615
616
617
618 0329 F6 C4 70     K22:  TEST    AH,CAPS_SHIFT+NUM_SHIFT+SCROLL_SHIFT ; SHIFT TOGGLE?
619 032C 74 05        JZ     K22B            ; GO IF NOT
620 032E 50          PUSH   AX               ; SAVE SCAN CODE AND SHIFT MASK
621 032F E8 0687 R   CALL  SND_LED          ; GO TURN MODE INDICATORS ON
622 0332 58          POP    AX               ; RESTORE SCAN CODE
623
624 0333 3C 52        K22B: CMP     AL,INS_KEY      ; TEST FOR 1ST MAKE OF INSERT KEY
625 0335 75 09        JNE    K21              ; JUMP IF NOT INSERT KEY
626 0337 8A E0        MOV     AH,AL           ; SCAN CODE IN BOTH HALVES OF AX
627 0339 EB 7F 90     JMP     K28              ; FLAGS UPDATED, PROC. FOR BUFFER
628
629
630
631
632 033C
633 033C 80 FC 10     K23:  CMP     AH,SCROLL_SHIFT ; BREAK-SHIFT-FOUND
634 0341 73 43        NOT    AH               ; IS THIS A TOGGLE KEY?
635 0343 20 26 0017 R  AND    *KB_FLAG,AH     ; INVERT MASK
636 0347 80 FC FB     JAE    K24              ; YES, HANDLE BREAK TOGGLE
637 034A 77 26        AND    *KB_FLAG,AH     ; TURN OFF SHIFT BIT
638
639 034C F6 C7 02     CMP     AH,NOT_CTL_SHIFT ; IS THIS ALT OR CTL?
640 034F 74 06        JZ     K23A             ; NO, ALL DONE
641 0351 20 26 0096 R  AND    *KB_FLAG_3,AH    ; 2ND ALT OR CTL?
642 0355 EB 06        JMP     SHORT K23B      ; NO, HANDLE NORMALLY
643 0357 D2 FC        AND    *KB_FLAG_3,AH    ; RESET BIT FOR RIGHT ALT OR CTL
644 0359 20 26 0018 R  AND    *KB_FLAG_1,AH    ; CONTINUE
645 035D 8A E0        SAR    AH,CL            ; MOVE THE MASK BIT TWO POSITIONS
646 035F A0 0096 R   AND    *KB_FLAG_1,AH    ; RESET BIT FOR LEFT ALT OR CTL
647 0362 D2 EB        MOV     AH,AL           ; SAVE SCAN CODE
648 0364 0A 06 0018 R  OR     AL,*KB_FLAG_3    ; GET RIGHT ALT & CTRL FLAGS
649 0368 D2 E0        SHR    AL,CL            ; MOVE TO BITS 1 & 0
650 036A 24 0C        OR     AL,*KB_FLAG_1    ; PUT IN LEFT ALT & CTL FLAGS
651 036C 08 06 0017 R  AND    AL,CTL_SHIFT+CTL_SHIFT ; MOVE BACK TO BITS 3 & 2
652 0370 8A C4        OR     *KB_FLAG,AL     ; FILTER OUT OTHER GARBAGE
653
654
655
656
657
658
659
660
661
662
663

```

SECTION 5

```

654 0372 3C B8          K23D:  CMP    AL,ALT_KEY+80H      ; IS THIS ALTERNATE SHIFT RELEASE
655 0374 76 2A          JNE    K26                ; INTERRUPT_RETURN
656
657                    ;----- ALTERNATE SHIFT KEY RELEASED, GET THE VALUE INTO BUFFER
658
659 0376 A0 0019 R      MOV    AL,@ALT_INPUT      ; SCAN CODE OF 0
660 0379 B4 00          MOV    AH,0              ; ZERO OUT THE FIELD
661 037B 88 26 0019 R  MOV    @ALT_INPUT,AH     ; WAS THE INPUT = 0?
662 037F 3C 00          CMP    AL,0              ; INTERRUPT_RETURN
663 0381 74 1D          JE     K26                ; IT WASN'T, SO PUT IN BUFFER
664 0383 E9 0601 R      JMP    K61
665
666 0386 20 26 0018 R   K24:   AND    @KB_FLAG_1,AH     ; BREAK-TOGGLE
667 038A EB 14          JMP    SHORT K26         ; INDICATE NO LONGER DEPRESSED
668
669                    ;----- TEST FOR HOLD STATE
670
671                    ; AL, AH = SCAN CODE
672 038C                    K25:   ; NO-SHIFT-FOUND
673 038C 3C 80          JAE    K26                ; TEST FOR BREAK KEY
674 038E 73 10          JAE    K26                ; NOTHING FOR BREAK CHARS FROM HERE ON
675 0390 F6 06 0018 R 08 TEST   @KB_FLAG_1,HOLD_STATE ; ARE WE IN HOLD STATE
676 0395 74 23          JZ     K26                ; BRANCH AROUND TEST IF NOT
677 0397 3C 45          CMP    AL,NUM_KEY        ; CAN'T END HOLD ON NUM LOCK
678 0399 74 05          JE     K26                ; TURN OFF THE HOLD STATE BIT
679 039B 80 26 0018 R F7 AND    @KB_FLAG_1,NOT_HOLD_STATE
680
681 03A0                    K26:   AND    @KB_FLAG_3,NOT LC_E0+LC_E1 ; RESET LAST CHAR H.C. FLAG
682 03A0 80 26 0096 R FC
683
684 03A5                    K26A:  CLI
685 03A5 FA          MOV    AL,E01            ; INTERRUPT_RETURN
686 03A6 B0 20          OUT   020H,AL           ; TURN OFF INTERRUPTS
687 03A8 E6 20          OUT   020H,AL           ; END OF INTERRUPT COMMAND
688
689                    ; SEND COMMAND TO INTERRUPT CONTROL PORT
690 03AA B0 AE          MOV    AL,ENA_KBD        ; INTERRUPT_RETURN-NO-E01
691 03AC E8 063C R      CALL  SHIP_IT           ; INSURE KEYBOARD IS ENABLED
692
693                    ; EXECUTE ENABLE
694 03AF FA          CLI
695 03B0 07          POP   ES                ; DISABLE INTERRUPTS
696 03B1 1F          POP   DS                ; RESTORE REGISTERS
697 03B2 5F          POP   DI
698 03B3 6E          POP   SI
699 03B4 5A          POP   DX
700 03B5 59          POP   CX
701 03B6 5B          POP   BX
702 03B7 58          POP   AX
703 03B8 5D          POP   BP
704 03B9 CF          IRET                    ; RETURN

```



```

704 PAGE
705 |----- NOT IN HOLD STATE
706
707 03BA K28: ; AL, AH = SCAN CODE (ALL MAKES)
708 03BA 3C 58 ; NO-HOLD-STATE
709 03BC 77 E2 ; TEST FOR OUT-OF-RANGE SCAN CODES
710 ; IGNORE IF OUT-OF-RANGE
711 03BE F4 C3 08 ; ARE WE IN ALTERNATE SHIFT?
712 03C1 74 0C ; JUMP IF NOT ALTERNATE
713
714 03C3 F6 C7 10 ; IS THIS THE ENHANCED KEYBOARD?
715 03C6 74 0A ; NO, ALT STATE IS REAL
716
717 03C8 F6 06 0018 R 04 ; YES, IS SYSREQ KEY DOWN?
718 03CD 74 03 ; NO, ALT STATE IS REAL
719 03CF E9 04A3 R ; YES, THIS IS PHONY ALT STATE
720 ; DUE TO PRESSING SYSREQ
721
722 |----- TEST FOR RESET KEY SEQUENCE (CTL ALT DEL)
723
724 03D2 K29: ; TEST-RESET
725 03D2 F6 C3 04 ; ARE WE IN CONTROL SHIFT ALSO?
726 03D3 74 31 ; NO, RESET
727 03D7 3C 53 ; SHIFT STATE IS THERE, TEST KEY
728 03D9 75 2D ; NO, RESET, IGNORE
729
730 |----- CTL-ALT-DEL HAS BEEN FOUND, DO I/O CLEANUP
731
732 03DB CT 06 0072 R 1234 ; SET FLAG FOR RESET FUNCTION
733 03E1 E9 0000 E ; JUMP TO POWER ON DIAGNOSTICS
734
735 |----- TABLES FOR ALT CASE -----
736
737 03E4 K30: ; ALT-INPUT-TABLE
738 03E4 62 4F 50 51 4B ; LABEL BYTE
739 03E9 4C 4D 47 48 49 ; DB 82,79,80,81,75
740 ; 76,77,71,72,73 ; 10 NUMBERS ON KEYPAD
741 03EE 10 11 12 13 14 15 ; SUPER-SHIFT-TABLE
742 03FA 16 17 18 19 1E 1F ; DB 16,17,18,19,20,21
743 03FA 20 21 22 23 24 25 ; DB 22,23,24,25,30,31
744 0400 26 2C 2D 2E 2F 30 ; DB 32,33,34,35,36,37
745 0406 31 32 ; DB 38,44,45,46,47,48
746 ; 49,50
747
748 |----- IN ALTERNATE SHIFT, RESET NOT FOUND
749
750 0408 K31: ; NO-RESET
751 0408 3C 39 ; TEST FOR SPACE KEY
752 040A 75 05 ; JNE K311
753 040C B0 20 ; MOV AL, ' '
754 040E E9 05F6 R ; JMP K57
755 ; SET SPACE CHAR
756 ; BUFFER_FILL
757 0411 3C 0F ; TEST FOR TAB KEY
758 0413 75 06 ; JNE K312
759 0415 B8 A500 ; MOV AX,0A500h
760 041B 3C 4A ; TEST FOR KEYPAD -
761 041D 74 79 ; JE K37B
762 041F 3C 4E ; TEST FOR KEYPAD +
763 0421 74 75 ; JE K37B
764 ; GO PROCESS
765
766 |----- LOOK FOR KEY PAD ENTRY
767
768 0423 K32: ; ALT-KEY-PAD
769 0423 BF 03E4 R ; MOV D1,OFFSET K30
770 0425 F2 AE ; MOV CX,10
771 042B 75 18 ; REPNE SCASB
772 042D F6 C7 02 ; JNE K33
773 0430 75 68 ; TEST BH,LC_E0
774 0432 81 EF 03E5 R ; JNB K37C
775 0436 A0 0019 R ; SUB D1,OFFSET K30+1
776 0439 B4 0A ; MOV AL,%ALT_INPUT
777 043B F6 E4 ; MOV AH,10
778 043D 03 C7 ; MUL AH
779 043F A2 0019 R ; ADD AX,D1
780 0442 E9 03A0 R ; MOV %ALT_INPUT,AL
781 ; STORE IT AWAY
782 ; THROW AWAY THAT KEYSTROKE
783
784 |----- LOOK FOR SUPERSHIFT ENTRY
785
786 0445 K33: ; NO-ALT-KEYPAD
787 0445 C6 06 0019 R 00 ; MOV %ALT_INPUT,0
788 044A B9 001A ; MOV CX,25
789 044D F2 AE ; REPNE SCASB
790 ; LOOK FOR MATCH IN ALPHABET
791 ; MATCH FOUND, GO FILL THE BUFFER
792
793 |----- LOOK FOR TOP ROW OF ALTERNATE SHIFT
794
795 0451 K34: ; ALT-TOP-ROW
796 0451 3C 02 ; CMP AL,2
797 0453 72 43 ; JB K37B
798 0455 3C 0D ; CMP AL,13
799 0457 77 05 ; JA K35
800 0459 80 C4 76 ; ADD AH,118
801 045C EB 35 ; JMP SHORT K37A
802 ; GO FILL THE BUFFER
803
804 |----- TRANSLATE ALTERNATE SHIFT PSEUDO SCAN CODES
805
806 045E K35: ; ALT-FUNCTION
807 045E 3C 57 ; IS IT F11?
808 0460 72 09 ; JB K35A
809 0462 3C 58 ; IS IT F12?
810 0464 77 05 ; JA K35A
811 0466 80 C4 34 ; ADD AH,52
812 0469 EB 28 ; JMP SHORT K37A
813 ; CONVERT TO PSEUDO SCAN CODE
814 ; GO FILL THE BUFFER
815
816 046B K35A: ; DO WE HAVE ONE OF THE NEW KEYS?
817 046B F6 C7 02 ; JZ K37
818 046E 74 15 ; NO, JUMP
819 0470 3C 1C ; CMP AL,28
820 0472 75 06 ; JNE K35B
821 0474 B8 A600 ; MOV AX,0A600h
822 0477 E9 05F5 R ; JMP K57
823 ; BUFFER_FILL
824 047A 3C 53 ; CMP AL,83
825 047C 74 1F ; JE K37C
826 ; TEST FOR DELETE KEY
827 ; HANDLE WITH OTHER EDIT KEYS

```

SECTION 5

```

818 047E 3C 95          CMP     AL,59          ; TEST FOR KEYPAD /
819 0480 78 CD          JNE    K32A           ; NOT THERE, NO OTHER E0 SPECIALS
820 0482 B8 A400        MOV    AX,0A400h      ; SPECIAL CODE
821 0485 E9 05F5 R      JMP    K57             ; BUFFER FILL
822
823 0488 3C 9B          K37:  CMP     AL,59          ; TEST FOR FUNCTION KEYS (F1)
824 048A 72 0C          JBE    K37B           ; NO FN, HANDLE W/OTHER EXTENDED
825 048C 3C 44          CMP    AL,66          ; IN KEYPAD REGION?
826
827 048E 77 B2          JA     K32A           ; OR NUMLOCK, SCROLLLOCK?
828 0490 80 C4 2D        ADD    AH,45          ; IF SO, IGNORE
829
830 0493 B0 00          K37A: MOV    AL,0          ; CONVERT TO PSEUDO SCAN CODE
831 0495 E9 05F5 R      JMP    K57             ; ASCII CODE OF ZERO
832
833 0498 B0 F0          K37B: MOV    AL,0F0h      ; USE SPECIAL ASCII CODE
834 049A E9 05F5 R      JMP    K57             ; PUT IT IN THE BUFFER
835
836 049D 04 50          K37C: ADD    AL,50          ; CONVERT SCAN CODE (EDIT KEYS)
837 049F 8A E0          JNZ    K38A           ; (SCAN CODE NOT IN AH FOR INSERT)
838 04A1 EB F0          JMP    K37A           ; PUT IT IN THE BUFFER
839
840
841
842 04A3                K38:
843
844 04A5 F6 C3 04        TEST   BL,CTL_SHIFT   ; NOT-ALT-SHIFT
845 04A8 75 03          JNZ    K38A           ; BL STILL HAS SHIFT FLAGS
846 04AB E9 0535 R      JMP    K44             ; ARE WE IN CONTROL SHIFT?
847
848
849
850
851
852 04AD 3C 46          K38A: CMP    AL,SCROLL_KEY ; TEST FOR BREAK
853 04AF 75 23          JNE    K39            ; JUMP, NO-BREAK
854 04B1 F6 C7 10        TEST   BH,KBX         ; IS THIS THE ENHANCED KEYBOARD?
855 04B4 74 05          JZ     K38B           ; NO, BREAK IS VALID
856 04B6 F6 C7 02        TEST   BH,LC_E0       ; YES, WAS LAST CODE AN E0?
857 04B9 74 19          JZ     K39            ; NO-BREAK, TEST FOR PAUSE
858
859 04BB 8B 1E 001A R     K38B: MOV    BX,BUFFER_HEAD ; RESET BUFFER TO EMPTY
860 04BD 89 1E 001C R     MOV    #BUFFER_TAIL,BX ;
861 04C1 C6 06 0071 R 80  MOV    #BIOS_BREAK,80H ; TURN ON BIOS_BREAK BIT
862
863
864
865 04C6 B0 AE          K38B: MOV    AL,ENA_KBD    ; ENABLE KEYBOARD
866 04C8 EB 063C R      CALL   SHIP_IT        ; EXECUTE ENABLE
867 04CB CD 18          INT    18H            ; BREAK INTERRUPT VECTOR
868 04CD 2B CD          SUB    AX,AX          ; PUT OUT DUMMY CHARACTER
869 04CF E9 05F5 R      JMP    K57             ; BUFFER_FILL
870
871
872
873 04D2                K39:
874 04D2 F6 C7 10        TEST   BH,KBX         ; NO-BREAK
875 04D5 75 2A          JNE    K41            ; IS THIS THE ENHANCED KEYBOARD?
876 04D7 3C 45          CMP    AL,NUM_KEY     ; YES, THEN THIS CAN'T BE PAUSE
877 04D9 75 26          JNE    K41            ; LOOK FOR PAUSE KEY
878 04DB 80 0E 0018 R 80  OR     #KB_FLAG_1,HOLD_STATE ; NO-PAUSE
879
880
881
882 04E0 B0 AE          K39P: OR     #KB_FLAG_1,HOLD_STATE ; TURN ON THE HOLD FLAG
883 04E2 EB 063C R      CALL   SHIP_IT        ; EXECUTE ENABLE
884 04E5 B0 20          MOV    AL,E0I         ; ENABLE KEYBOARD
885 04E7 E6 20          OUT    020h,AL        ; EXECUTE ENABLE
886
887
888
889 04E9 80 3E 0049 R 07  CMP    #CRT_MODE,7    ; IS THIS BLACK AND WHITE CARD
890 04EE 74 07          JBE    K40            ; YES, NOTHING TO DO
891 04F0 BA 030B        MOV    DX,030Bh       ; PORT FOR COLOR CARD
892 04F3 AD 0045 R      MOV    AL,#CRT_MODE_SET ; GET THE VALUE OF THE CURRENT MODE
893 04F6 EE          OUT    DX,AL          ; SET THE CRT MODE, SO THAT CRT IS ON
894 04F7
895 04F7 F6 06 0018 R 08  K40:  TEST   #KB_FLAG_1,HOLD_STATE ; PAUSE-LOOP
896 04FC 75 16          JNE    K40            ; LOOP UNTIL FLAG TURNED OFF
897 04FE E9 03AA R      JMP    K27            ; INTERRUPT_RETURN_NO_E0I
898
899
900
901 0501                K41:
902 0501 3C 37          CMP    AL,55          ; NO-PAUSE
903 0503 75 10          JNE    K42            ; TEST FOR */PRTS KEY
904 0505 F6 C7 10        TEST   BH,KBX         ; NOT-KEY-55
905 0508 74 05          JZ     K41A           ; IS THIS THE ENHANCED KEYBOARD?
906 050A F6 C7 02        TEST   BH,LC_E0       ; NO, CTL-PRTS IS VALID
907 050D 74 20          JZ     K42B           ; YES, WAS LAST CODE AN E0?
908 050F B8 7200        MOV    AX,114*266     ; NO, TRANSLATE TO A FUNCTION
909 0512 E9 05F5 R      JMP    K57             ; START/STOP PRINTING SWITCH
910
911
912
913 0515                K41A: MOV    AX,114*266     ; BUFFER_FILL
914 0517 3C 0F          CMP    AL,15          ; NOT-KEY-55
915 0519 74 16          JBE    K42B           ; IS IT THE TAB KEY?
916 051B 3C 35          CMP    AL,53          ; YES, XLATE TO FUNCTION CODE
917 051D 76 0B          JNE    K42A           ; IS IT THE / KEY?
918 051F F6 C7 02        TEST   BH,LC_E0       ; NO, NO MORE SPECIAL CASES
919 0520 74 06          JZ     K42A           ; YES, IS IT FROM THE KEYPAD?
920 0522 B8 9500        MOV    AX,9500h       ; NO, JUST TRANSLATE
921 0525 E9 05F5 R      JMP    K57             ; YES, SPECIAL CODE FOR THIS ONE
922
923 0528 BB 0000 E       K42A: MOV    BX,OFFSET K8 ; SET UP TO TRANSLATE CTL
924 052B 3C 3B          CMP    AL,59          ; IS IT IN CHARACTER TABLE?
925 052D 72 5E          JBE    K45F           ; YES, GO TRANSLATE CHAR
926 052F B8 0000 E       K42B: MOV    BX,OFFSET K8 ; SET UP TO TRANSLATE CTL
927 0532 E9 05E4 R      JMP    K64            ; NO, GO TRANSLATE_SCAN
928
929
930
931 0535                K44:  CMP    AL,55          ; NOT IN CONTROL SHIFT
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999

```

```

932 0627 75 26           JNE    K45           ; NOT-PRINT-SCREEN
933 0539 F6 C7 10       TEST   BH,KBX        ; IS THIS ENHANCED KEYBOARD?
934 063C 74 07          JZ     K44A          ; NO, TEST FOR SHIFT STATE
935 053E F6 C7 02       TEST   BH,LC_E0      ; YES, LAST CODE A MARKER?
936 0541 75 07          JNZ   K44B          ; YES, IS PRINT SCREEN
937 0543 EB 3B          JMP    SHORT K45C    ; NO, XLATE TO "*" CHARACTER
938 0545 F6 C3 03       K44A: TEST   BL,LEFT_SHIFT+RIGHT_SHIFT ; NOT 101 KBD, SHIFT KEY DOWN?
939 0548 74 36          JZ     K45C          ; NO, XLATE TO "*" CHARACTER
940
941
942 054A 80 AE           I----- ISSUE INTERRUPT TO PERFORM PRINT SCREEN FUNCTION
K44B: MOV    AL,ENA_KBD   ; INSURE KEYBOARD IS ENABLED
      CALL  SHIP_IT    ; EXECUTE ENABLE
      MOV    AL,E01    ; END OF CURRENT INTERRUPT
      OUT   020H,AL    ; SO FURTHER THINGS CAN HAPPEN
      PUSH  BP        ; SAVE POINTER
      INT  3H         ; ISSUE PRINT SCREEN INTERRUPT
      POP   BP        ; RESTORE POINTER
947 0554 CD 05          AND   #KB_FLAG_3,NOT LC_E0+LC_E1 ; ZERO OUT THESE FLAGS
948 0556 5D            JMP   K27           ; GO BACK WITHOUT E01 OCCURRING
949 0557 80 26 0096 R FC
950 055C E9 03AA R
951
952
953
954 055F 3C 3A           I----- HANDLE THE IN-CORE KEYS
K45:  CMP    AL,68       ; NOT-PRINT-SCREEN
      JA     K46        ; TEST FOR IN-CORE AREA
      JZ     K44        ; JUMP IF NOT
957
958 0563 3C 35           ; IS THIS THE '/' KEY?
      JNE   K45        ; NO, JUMP
959 0565 75 05           TEST   BH,LC_E0      ; WAS LAST CODE THE MARKER?
960 0567 F6 CT 02       JNZ   K45C          ; YES, TRANSLATE TO CHARACTER
961 056A 75 14
962
963 056C B9 001A        K45A: MOV    CX,26     ; LENGTH OF SEARCH
      MOV    DI,OFFSET K30+10 ; POINT TO TABLE OF A-Z CHARS
      REPNE SCASB      ; IS THIS A LETTER KEY?
      JNE   K45B      ; NO, SYMBOL KEY
964 056F BF 03EE R
965 0572 F2 / AE       TEST   BL,CAPS_STATE ; ARE WE IN CAPS_LOCK?
966 0574 75 05           JNZ   K45D          ; TEST FOR SURE
967
968 0576 F6 C3 40       K45B: TEST   BL,LEFT_SHIFT+RIGHT_SHIFT ; SHIFTED TEMP OUT OF CAPS STATE?
969 0579 75 0A           JNZ   K45E          ; YES, UPPERCASE
970 057B F6 C3 03       JZ     K45C          ; NO, LOWERCASE
971 057E 75 0A           JZ     K45E          ; TRANSLATE TO LOWERCASE LETTERS
972
973 0580 BB 0000 E      K45C: MOV    BX,OFFSET K10 ; ALMOST-CAPS-STATE
974 0583 EB 50           JMP    SHORT K56     ; CL ON. IS SHIFT ON, TOO?
975 0585
976 0585 F6 C3 03       K45D: TEST   BL,LEFT_SHIFT+RIGHT_SHIFT ; SHIFTED TEMP OUT OF CAPS STATE
977 0588 75 16           JNZ   K45E          ; TRANSLATE TO UPPERCASE LETTERS
978 058A BB 0000 E
979 058D EB 46           K45E: MOV    BX,OFFSET K11 ; TRANSLATE TO UPPERCASE LETTERS
980 058F JMP    SHORT K56
981
982
983 058F
984 058F 3C 44           I----- TEST FOR KEYS F1 - F10
K46:  CMP    AL,68       ; NOT IN-CORE AREA
      JA     K47        ; TEST FOR F1 - F10
      JZ     K47        ; JUMP IF NOT
987 0591 77 02           JMP   SHORT K53     ; YES, GO DO FN KEY PROCESS
988 0593 EB 36
989
990
991 0595 3C 53           I----- HANDLE THE NUMERIC PAD KEYS
K47:  CMP    AL,B3       ; NOT F1 - F10
      JA     K52        ; TEST FOR NUMPAD KEYS
      JZ     K49        ; JUMP IF NOT
993
994
995 0599 3C 4A           I----- KEYPAD KEYS, MUST TEST NUM LOCK FOR DETERMINATION
K48:  CMP    AL,74       ; SPECIAL CASE FOR MINUS
      JE     K45E      ; GO TRANSLATE
      JNE   AL,78     ; SPECIAL CASE FOR PLUS
      JE     K45E      ; GO TRANSLATE
998 059F 74 E9           TEST   BH,LC_E0      ; IS THIS ONE OF THE NEW KEYS?
999 05A1 F6 CT 02       JNZ   K49           ; YES, TRANSLATE TO BASE STATE
1000 05A4 75 0A
1001
1002 05A6 F6 C3 20       TEST   BL,NUM_STATE  ; ARE WE IN NUM LOCK?
1003 05A9 75 13         JNZ   K50           ; TEST FOR SURE
1004 05AB F6 C3 03       TEST   BL,LEFT_SHIFT+RIGHT_SHIFT ; ARE WE IN SHIFT STATE?
1005 05AE 75 13         JNZ   K51           ; IF SHIFTED, REALLY NUM STATE
1006
1007
1008 05B0 3C 4C           I----- BASE CASE FOR KEYPAD
K49:  CMP    AL,76       ; SPECIAL CASE FOR BASE STATE 5
      JNE   K49A      ; CONTINUE IF NOT KEYPAD 5
      MOV    AL,0F0h  ; SPECIAL ASCII CODE
      JMP   K57        ; BUFFER FILL
1010 05B4 8D F0         K49A: MOV    BX,OFFSET K10 ; BASE CASE TABLE
1011 05B6 EB 3D 90       JMP    SHORT K64     ; CONVERT TO PSEUDO SCAN
1012 05B9 BB 0000 E
1013 05BC EB 26
1014
1015
1016 05BE F6 C3 03       I----- MIGHT BE NUM LOCK, TEST SHIFT STATUS
K50:  TEST   BL,LEFT_SHIFT+RIGHT_SHIFT ; ALMOST-NUM-STATE
1017 05C1 75 03         JNZ   K51           ; SHIFTED TEMP OUT OF NUM STATE
1018 05C3 EB C5         JMP    SHORT K45E    ; REALLY_NUM_STATE
1019
1020
1021
1022
1023 06C5
1024 05C5 3C 56           I----- TEST FOR THE NEW KEY ON WT KEYBOARDS
K52:  CMP    AL,86       ; NOT A NUMPAD KEY
      JNE   K53        ; IS IT THE NEW WT KEY?
      JZ     K53        ; JUMP IF NOT
1026 05C9 EB B0         JMP    SHORT K45B    ; HANDLE WITH REST OF LETTER KEYS
1027
1028
1029
1030
1031 05CB F6 C3 03       I----- MUST BE F11 OR F12
K53:  TEST   BL,LEFT_SHIFT+RIGHT_SHIFT ; F1 - F10 COME HERE, TOO
1032 05CE 74 E0         JZ     K49          ; TEST SHIFT STATE
1033
1034 05D0 BB 0000 E      MOV    BX,OFFSET K11 ; UPPER CASE PSEUDO SCAN CODES
1035 05D3 EB 0F         JMP    SHORT K64     ; TRANSLATE_SCAN
1036
1037
1038
1039 08D5
1040 08D5 FE C8           I----- TRANSLATE THE CHARACTER
K56:  DEC    AL           ; TRANSLATE-CHAR
      XLAT CS:K11     ; CONVERT ORIGIN
1042 08D9 F6 06 0096 R 02 ; CONVERT THE SCAN CODE TO ASCII
1043 08DE 74 15         TEST   #KB_FLAG_3,LC_E0 ; IS THIS A NEW KEY?
1044 08E0 B4 E0         JZ     K57          ; NO, GO FILL BUFFER
1045 08E2 EB 11         MOV    AH,MC_E0    ; YES, PUT SPECIAL MARKER IN AH
      JMP   SHORT K57 ; PUT IT INTO THE BUFFER

```

SECTION 5

```

1046
1047
1048
1049 05E4
1050 05E4 FE C8
1051 05E6 2E1 D7
1052 05E8 8A E0
1053 05EA B0 00
1054 05EC F6 06 0096 R 02
1055 05F1 74 02
1056 05F3 B0 E0
1057
1058
1059
1060 05F5
1061 05F5 3C FF
1062 05F7 74 05
1063 05F9 80 FC FF
1064 05FC 75 03
1065
1066 05FE
1067 05FE E9 03A0 R
1068
1069 0601
1070 0601 8B 1E 001C R
1071 0605 8B F3
1072 0607 E8 0168 R
1073 060A 3B 1E 001A R
1074 060E 74 1D
1075 0610 89 04
1076 0612 89 1E 001C R
1077 0616 FA
1078 0617 B0 20
1079 0619 E6 20
1080 061B B0 AE
1081 061D E8 063C R
1082 0620 B8 9102
1083 0623 CD 15
1084 0625 80 26 0096 R FC
1085 062A E9 03AF R
1086
1087
1088
1089 062D
1090 062D B0 20
1091 062F E6 20
1092 0631 B9 02A6
1093 0634 B3 04
1094 0636 E8 0000 E
1095 0639 E9 03AA R
1096
1097 063C

```

```

;----- TRANSLATE SCAN FOR PSEUDO SCAN CODES
K64:
DEC AL ; TRANSLATE-SCAN-ORGD
XLAT CS:K8 ; CONVERT ORIGIN
MOV AH,AL ; CTL TABLE SCAN
MOV AL,0 ; PUT VALUE INTO AH
TEST 0KB_FLAG_3,LC_E0 ; ZERO ASCII CODE
JZ K67 ; IS THIS A NEW KEY?
MOV AL,MC_E0 ; NO, GO FILL BUFFER
; YES, PUT SPECIAL MARKER IN AL
;----- PUT CHARACTER INTO BUFFER
K57:
CMP AL,-1 ; BUFFER-FILL
JE K59 ; IS THIS AN IGNORE CHAR
CMP AH,-1 ; YES, DO NOTHING WITH IT
JNE K61 ; LOOK FOR -1 PSEUDO SCAN
; NEAR_INTERRUPT_RETURN
K59:
JMP K26 ; NEAR-INTERRUPT-RETURN
; INTERRUPT_RETURN
K61:
MOV BX,0BUFFER_TAIL ; GET THE END POINTER TO THE BUFFER
MOV SI,BX ; SAVE THE VALUE
CALL K4 ; ADVANCE THE TAIL
CMP BX,0BUFFER_HEAD ; HAS THE BUFFER WRAPPED AROUND
JE K62 ; BUFFER FULL_BEEP
MOV [SI],AX ; STORE THE VALUE
MOV 0BUFFER_TAIL,BX ; MOVE THE POINTER UP
CLI ; TURN OFF INTERRUPTS
AL,ED1 ; END OF INTERRUPT COMMAND
OUT INTA00,AL ; SEND COMMAND TO INTERRUPT CONTROL PORT
MOV AL,ENA_KBD ; INSURE KEYBOARD IS ENABLED
CALL SHIP_IT ; EXECUTE ENABLE
MOV AX,09102H ; MOVE IN POST CODE & TYPE
INT 15H ; PERFORM OTHER FUNCTION
AND 0KB_FLAG_3,NOT LC_E0+LC_E1 ; RESET LAST CHAR H.C. FLAG
JMP K27A ; INTERRUPT_RETURN
;----- BUFFER IS FULL SOUND THE BEEPER
K62:
MOV AL,ED1 ; ENABLE INTERRUPT CONTROLLER CHIP
OUT INTA00,AL ;
MOV CX,678 ; DIVISOR FOR 1760 HZ
MOV BL,4 ; SHORT BEEP COUNT (1/16 + 1/64 DELAY)
CALL BEEP ; GO TO COMMON BEEP HANDLER
JMP K27 ; EXIT
KB_INT_1 ENDP

```

```

1098 PAGE
1099 |-----|
1100 | |
1101 | SHIP_IT |
1102 | THIS ROUTINE HANDLES TRANSMISSION OF COMMAND AND DATA BYTES |
1103 | TO THE KEYBOARD CONTROLLER. |
1104 |-----|
1105 |
1106 SHIP_IT PROC NEAR
1107 063C PUSH AX ; SAVE DATA TO SEND
1108 063D 50
1109 |-----|
1110 |----- WAIT FOR COMMAND TO BE ACCEPTED |
1111 063D FA CL1 ; DISABLE INTERRUPTS TILL DATA SENT
1112 063E 2B C9 SUB CX,CX ; CLEAR TIMEOUT COUNTER
1113 0640 S10: IN AL,STATUS_PORT ; READ KEYBOARD CONTROLLER STATUS
1114 0640 E4 64 AL,INPT_BUF_FULL ; CHECK FOR ITS INPUT BUFFER BUSY
1115 0642 A8 02 TEST AL,INPT_BUF_FULL ; WAIT FOR COMMAND TO BE ACCEPTED
1116 0644 E0 FA LOOPNZ S10
1117 |
1118 0646 58 POP AX ; GET DATA TO SEND
1119 0647 E6 64 OUT STATUS_PORT,AL ; SEND TO KEYBOARD CONTROLLER
1120 0649 FB STI ; ENABLE INTERRUPTS AGAIN
1121 064A C3 RET ; RETURN TO CALLER
1122 064B SHIP_IT ENDP
1123 |-----|
1124 |
1125 | SND_DATA |
1126 |-----|
1127 |
1128 | THIS ROUTINE HANDLES TRANSMISSION OF COMMAND AND DATA BYTES |
1129 | TO THE KEYBOARD AND RECEIPT OF ACKNOWLEDGEMENTS. IT ALSO |
1130 | HANDLES ANY RETRIES IF REQUIRED |
1131 |-----|
1132 |
1133 SND_DATA PROC NEAR
1134 064B PUSH AX ; SAVE REGISTERS
1135 064B 50 PUSH BX ; *
1136 064C 53 PUSH CX ;
1137 064D 51 MOV BH,AL ; SAVE TRANSMITTED BYTE FOR RETRIES
1138 064E 8A F8 MOV BL,3 ; LOAD RETRY COUNT
1139 0650 B3 03 SD0: CLI ; DISABLE INTERRUPTS
1140 0652 FA AND 0KB_FLAG_2,NOT (KB_FE+KB_FA) ; CLEAR ACK AND RESEND FLAGS
1141 0653 80 26 0097 R CF
1142 |-----|
1143 |----- WAIT FOR COMMAND TO BE ACCEPTED |
1144 |
1145 0658 2B C9 SUB CX,CX
1146 065A E4 64 S05: IN AL,STATUS_PORT
1147 065C A8 02 TEST AL,INPT_BUF_FULL ; WAIT FOR COMMAND TO BE ACCEPTED
1148 065E E0 FA LOOPNZ S05
1149 |
1150 0660 8A CT MOV AL,BH ; REESTABLISH BYTE TO TRANSMIT
1151 0662 E6 60 OUT PORT_A,AL ; SEND BYTE
1152 0664 FB STI ; ENABLE INTERRUPTS
1153 0665 B9 1A00 MOV CX,01A00H ; LOAD COUNT FOR 10ms+
1154 0668 F6 06 0097 R 30 SD1: TEST 0KB_FLAG_2,KB_FE+KB_FA ; SEE IF EITHER BIT SET
1155 066D 75 0D JNZ SD3 ; IF SET, SOMETHING RECEIVED GO PROCESS
1156 066F E2 F7 LOOP SD1 ; OTHERWISE WAIT
1157 |
1158 0671 FE CB SD2: DEC BL ; DECREMENT RETRY COUNT
1159 0673 75 0D JNZ SD0 ; RETRY TRANSMISSION
1160 0675 80 0E 0097 R 80 OR 0KB_FLAG_2,KB_ERR ; TURN ON TRANSMIT ERROR FLAG
1161 067A EB 07 JMP SHORT SD4 ; RETRIES EXHAUSTED FORGET TRANSMISSION
1162 |
1163 067C F6 06 0097 R 10 SD3: TEST 0KB_FLAG_2,KB_FA ; SEE IF THIS IS AN ACKNOWLEDGE
1164 0681 74 EE JZ SD2 ; IF NOT, GO RESEND
1165 |
1166 0683 59 POP CX ; RESTORE REGISTERS
1167 0684 5B POP BX ; *
1168 0685 58 POP AX ; *
1169 0686 C3 RET ; RETURN, GOOD TRANSMISSION
1170 0687 SND_DATA ENDP

```

SECTION 5

```

1171                                     PAGE
1172                                     -----
1173                                     |
1174                                     |           SND_LED
1175                                     |
1176                                     |           THIS ROUTINE TURNS ON THE MODE INDICATORS.
1177                                     |
1178                                     -----
1179 0687          SND_LED PROC          NEAR
1180 0687 FA          CLI
1181 0688 F6 06 0097 R 40          TEST    #KB_FLAG_2,KB_PR_LED  ; TURN OFF INTERRUPTS
1182 068D 75 47          JNZ          SL1          ; CHECK FOR MODE INDICATOR UPDATE
1183                                     |           ; DONT UPDATE AGAIN IF UPDATE UNDERWAY
1184 068F 80 0E 0097 R 40          OR      #KB_FLAG_2,KB_PR_LED  ; TURN ON UPDATE IN PROCESS
1185 0694 B0 20          MOV      AL,E01          ; END OF INTERRUPT COMMAND
1186 0696 E6 20          OUT      020H,AL      ; SEND COMMAND TO INTERRUPT CONTROL PORT
1187 0698 EB 0D          JMP      SHORT SL0          ; GO SEND MODE INDICATOR COMMAND
1188
1189 069A          SND_LED1:
1190 069A FA          CLI
1191 069B F6 06 0097 R 40          TEST    #KB_FLAG_2,KB_PR_LED  ; TURN OFF INTERRUPTS
1192 06A0 75 34          JNZ          SL1          ; CHECK FOR MODE INDICATOR UPDATE
1193                                     |           ; DONT UPDATE AGAIN IF UPDATE UNDERWAY
1194 06A2 80 0E 0097 R 40          OR      #KB_FLAG_2,KB_PR_LED  ; TURN ON UPDATE IN PROCESS
1195 06A7 B0 ED          MOV      AL,LED_CMD          ; LED CMD BYTE
1196 06A9 E8 064B R          CALL   SND_DATA          ; SEND DATA TO KEYBOARD
1197 06AC FA          CLI
1198 06AD E8 06D8 R          CALL   MAKE_LED          ; GO FORM INDICATOR DATA BYTE
1199 06B0 80 26 0097 R F8          AND     #KB_FLAG_2,0F8H      ; CLEAR MODE INDICATOR BITS
1200 06B5 08 06 0097 R          OR      #KB_FLAG_2,AL      ; SAVE PRESENT INDICATORS FOR NEXT TIME
1201 06B9 F6 06 0097 R 80          TEST    #KB_FLAG_2,KB_ERR    ; TRANSMIT ERROR DETECTED
1202 06BE 75 0B          JNZ          SL2          ; YES, BYPASS SECOND BYTE TRANSMISSION
1203
1204 06C0 E8 064B R          CALL   SND_DATA          ; SEND DATA TO KEYBOARD
1205 06C3 FA          CLI
1206 06C4 F6 06 0097 R 80          TEST    #KB_FLAG_2,KB_ERR    ; TRANSMIT ERROR DETECTED
1207 06C9 74 06          JZ      SL3          ; IF NOT, DONT SEND AN ENABLE COMMAND
1208
1209 06CB B0 F4          MOV     AL,KB_ENABLE        ; GET KEYBOARD CSA ENABLE COMMAND
1210 06CD E8 064B R          CALL   SND_DATA          ; SEND DATA TO KEYBOARD
1211 06D0 FA          CLI
1212 06D1 80 26 0097 R 3F          AND     #KB_FLAG_2,NOT(KB_PR_LED+KB_ERR) ; TURN OFF MODE INDICATOR
1213                                     |           ; UPDATE AND TRANSMIT ERROR FLAG
1214 06D6 FB          STI          ; ENABLE INTERRUPTS
1215 06D7 C3          RET          ; RETURN TO CALLER
1216 06D8          SND_LED ENDP
1217
1218                                     -----
1219                                     |
1220                                     |           MAKE_LED
1221                                     |
1222                                     |           THIS ROUTINE FORMS THE DATA BYTE NECESSARY TO TURN ON/OFF
1223                                     |           THE MODE INDICATORS
1224                                     |
1225                                     -----
1226 06D8          MAKE_LED PROC          NEAR
1227 06D8 51          PUSH   CX          ; SAVE CX
1228 06D9 A0 0017 R          MOV     AL,#KB_FLAG          ; GET CAPS & NUM LOCK INDICATORS
1229 06DC 24 70          AND     AL,CAPS_STATE+NUM_STATE+SCROLL_STATE ; ISOLATE INDICATORS
1230 06DE B1 04          MOV     CL,4          ; SHIFT COUNT
1231 06E0 D2 C0          ROL     AL,CL          ; SHIFT BITS OVER TO TURN ON INDICATORS
1232 06E2 24 07          AND     AL,07H          ; MAKE SURE ONLY MODE BITS ON
1233 06E4 59          POP     CX
1234 06E5 C3          RET          ; RETURN TO CALLER
1235 06E6          MAKE_LED ENDP
1236
1237 06E6          CODE          ENDS
1238                                     END

```

```

1 PAGE 118,121
2 TITLE PRT ----- 06/10/85 PRINTER ADAPTER BIOS
3 .286C
4 .LIST
5 0000 CODE SEGMENT BYTE PUBLIC
6
7 PUBLIC PRINTER_IO_1
8 EXTRN DDS:NEAR
9
10 INT 17 H
11 PRINTER_IO
12 THIS ROUTINE PROVIDES COMMUNICATION WITH THE PRINTER
13 INPUT
14 (AH)= 00H PRINT THE CHARACTER IN (AL)
15 ON RETURN, (AH)= 1 IF CHARACTER NOT BE PRINTED (TIME OUT)
16 OTHER BITS SET AS ON NORMAL STATUS CALL
17 (AH)= 01H INITIALIZE THE PRINTER PORT
18 RETURNS WITH (AH) SET WITH PRINTER STATUS
19 (AH)= 02H READ THE PRINTER STATUS INTO (AH)
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114

```

-----

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)

-----

(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

```

48 0000 PROC FAR ; ENTRY POINT FOR ORG 0EFD2H
49 0001 STI ; INTERRUPTS BACK ON
50 0002 PUSH DS ; SAVE SEGMENT
51 0003 PUSH SI
52 0004 PUSH DX
53 0005 PUSH CX
54 0006 PUSH BX
55 0007 CALL DDS ; ADDRESS DATA SEGMENT
56 0008 MOV SI,DX ; GET PRINTER PARAMETER
57 0009 SHR DX,2 ; LEFT PARAMETER
58 000A JNZ B10 ; RETURN IF NOT IN RANGE
59 000B MOV BL,#PRINT_TIM_OUT[SI] ; LOAD TIMEOUT VALUE
60 000C SHL SI,1 ; WORD OFFSET INTO TABLE INTO (SI)
61 000D MOV DX,#PRINTER_BASE[SI] ; GET BASE ADDRESS FOR PRINTER CARD
62 000E OR DX,DX ; TEST DX = ZERO, INDICATING NO PRINTER
63 000F JZ B10 ; EXIT, NO PRINTER ADAPTER AT OFFSET
64 0010 OR AH,AH ; TEST FOR (AH)= 00H
65 0011 JZ B20 ; PRINT CHARACTER IN (AL)
66 0012 DEC AH ; TEST FOR (AH)= 01H
67 0013 JZ B80 ; INITIALIZE PRINTER
68 0014 DEC AH ; TEST FOR (AH)= 02H
69 0015 JZ B50 ; GET PRINTER STATUS
70 0016 B10: POP BX ; RETURN
71 0017 POP CX
72 0018 POP DX
73 0019 POP SI
74 001A POP DS ; RECOVER REGISTERS
75 001B IRET ; RETURN TO CALLING PROGRAM
76
77
78
79
80 0020 B20: PUSH AX ; SAVE VALUE TO PRINT
81 0021 OUT DX,AL ; OUTPUT CHARACTER TO DATA PORT
82 0022 INC DX ; POINT TO STATUS PORT
83
84
85
86
87 0023 B25: PUSH BX ; SAVE TIMEOUT BASE COUNT
88 0024 IN AL,DX ; GET STATUS PORT VALUE
89 0025 TEST AL,80H ; IS THE PRINTER CURRENTLY BUSY
90 0026 JNZ B25 ; SKIP SYSTEM DEVICE BUSY CALL IF NOT
91
92
93
94 0027 INT 15 H ; DEVICE BUSY
95
96
97
98 0028 B25: MOV AX,90FEH ; FUNCTION 90 PRINTER ID
99 0029 INT 15H ; SYSTEM CALL
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114

```

SECTION 5

```

118 005A          B60:      POP     DX          ; SEND STROBE PULSE
116 005A 5B      MOV     AL,0DH      ; RESTORE (BX) WITH TIMEOUT COUNT
117 005B B0 0D   INC     DX          ; SET THE STROBE LOW (BIT ON)
118 005D 42      CLT          ; OUTPUT STROBE TO CONTROL PORT
119 005E FA      OUT     DX,AL      ; PREVENT INTERRUPT PULSE STRETCHING
120 005F EE      OUT     DX,AL      ; OUTPUT STROBE BIT > 1us < 5us
121 0060 EB 00   JMP     $+2        ; I/O DELAY TO ALLOW FOR LINE LOADING
122 0062 EB 00   JMP     $+2        ; AND FOR CORRECT PULSE WIDTH
123 0064 B0 0C   MOV     AL,0CH     ; SET THE -STROBE HIGH
124 0066 EE      OUT     DX,AL
125 0067 FB      STI          ; INTERRUPTS BACK ON
126 0068 58      POP     AX          ; RECOVER THE OUTPUT CHAR
127
128             ;----- PRINTER STATUS
129
130 0069          B60:      PUSH    AX          ; SAVE (AL) REGISTER
131 0069 50
132 006A          B60:      MOV     DX,0PRINTER_BASE[SI] ; GET PRINTER ATTACHMENT BASE ADDRESS
133 006A 8B 94 0008 R INC     DX          ; POINT TO CONTROL PORT
134 006E 42      IN     AL,DX      ; PRE-CHARGE +BUSY LINE IF FLOATING
135 006F EC      IN     AL,DX      ; GET PRINTER STATUS HARDWARE BITS
136 0070 EC      MOV     AH,AL     ; SAVE
137 0071 8A E0   AND     AH,0F8H   ; TURN OFF UNUSED BITS
138 0073 80 E4 F8
139 0076          B70:      POP     DX          ; RECOVER (AL) REGISTER
140 0076 5A      MOV     AL,DL     ; MOVE CHARACTER INTO (AL)
141 0077 8A C2   XOR     AH,48H    ; FLIP A COUPLE OF BITS
142 0079 80 F4 48
143 007C EB AC      JMP     B10       ; RETURN FROM ROUTINE WITH STATUS IN AH
144
145             ;----- INITIALIZE THE PRINTER PORT
146
147 007E          B80:      PUSH    AX          ; SAVE (AL)
148 007E 50      INC     DX          ; POINT TO OUTPUT PORT
149 007F 42      INC     DX          ; SET INIT LINE LOW
150 0080 42      MOV     AL,8      ; ADJUST FOR INITIALIZATION DELAY LOOP
151 0081 B0 08   OUT     DX,AL     ; INIT LOOP
152 0083 EE      OUT     DX,AL     ; LOOP FOR RESET TO TAKE
153 0084 B8 0FA0 MOV     AX,1000*4 ; INIT LOOP
154 0087          B90:      DEC     AX          ; NO INTERRUPTS, NON AUTO LF, INIT HIGH
155 0087 48      JNZ    B90       ; EXIT THROUGH STATUS ROUTINE
156 0088 75 FD   JNZ    B90
157 008A 80 0C   MOV     AL,0CH
158 008C EE      OUT     DX,AL
159 008D EB DB   JMP     B60
160
161 008F          PRINTER_IO_1 ENDP
162
163 008F          CODE     ENDS
164
164          END

```



```

1 PAGE 118,121
2 TITLE RS232 ---- 06/10/85 COMMUNICATIONS BIOS (RS232)
3 .286C
4 .LIST
5 0000 CODE SEGMENT BYTE PUBLIC
6 PUBLIC RS232_IO_1
7 EXTRN A1:NEAR
8 EXTRN DDS:NEAR
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114

```

```

----- INT 14 H -----
RS232_IO_1
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 - 180 01 - ODD 1 - 2 11 - 8 BITS
010 - 300 11 - EVEN
011 - 600
100 - 1200
101 - 2400
110 - 4800
111 - 9600
ON RETURN, CONDITIONS SET AS IN CALL TO COMMO STATUS (AH=03H)
(AH)= 01H SEND THE CHARACTER IN (AL) OVER THE COMMO LINE
(AL) REGISTER IS PRESERVED
ON EXIT, BIT 7 OF AH IS SET IF THE ROUTINE WAS UNABLE TO
TO TRANSMIT THE BYTE OF DATA OVER THE LINE.
IF BIT 7 OF AH IS NOT SET, THE
REMAINDER OF (AH) IS SET AS IN A STATUS REQUEST,
REFLECTING THE CURRENT STATUS OF THE LINE.
(AH)= 02H RECEIVE A CHARACTER IN (AL) FROM COMMO LINE BEFORE
RETURNING TO CALLER
ON EXIT, (AH) HAS THE CURRENT LINE STATUS, AS SET BY THE
THE STATUS ROUTINE, EXCEPT THAT THE ONLY BITS
LEFT ON ARE THE ERROR BITS (7,4,3,2,1)
IF (AH) HAS BIT 7 ON (TIME OUT) THE REMAINING
BITS ARE NOT PREDICTABLE.
THUS, (AH) IS NON ZERO ONLY WHEN AN ERROR OCCURRED.
(AH)= 03H RETURN THE COMMO PORT STATUS IN (AX)
(AH) CONTAINS THE LINE CONTROL STATUS
BIT 7 = TIME OUT
BIT 6 = TRANSMIT SHIFT REGISTER EMPTY
BIT 5 = TRANSMIT HOLDING REGISTER EMPTY
BIT 4 = BREAK DETECT
BIT 3 = FRAMING ERROR
BIT 2 = PARITY ERROR
BIT 1 = OVERRUN ERROR
BIT 0 = DATA READY
(AL) CONTAINS THE MODEM STATUS
BIT 7 = RECEIVE LINE SIGNAL DETECT
BIT 6 = RING INDICATOR
BIT 5 = DATA SET READY
BIT 4 = CLEAR TO SEND
BIT 3 = DELTA RECEIVE LINE SIGNAL DETECT
BIT 2 = TRAILING EDGE RING DETECTOR
BIT 1 = DELTA DATA SET READY
BIT 0 = DELTA CLEAR TO SEND
(DX) = PARAMETER INDICATING WHICH RS232 CARD (0 - 3 ALLOWED)
DATA AREA #RS232_BASE CONTAINS THE BASE ADDRESS OF THE 8250 ON THE CARD
LOCATION 400H CONTAINS UP TO 4 RS232 ADDRESSES POSSIBLE
DATA AREA LABEL #RS232_TIM_OUT (BYTE) CONTAINS OUTER LOOP COUNT
VALUE FOR TIMEOUT (DEFAULT=1)
OUTPUT AX MODIFIED ACCORDING TO PARAMETERS OF CALL
ALL OTHERS UNCHANGED
-----
ASSUME CS:CODE,DS:DATA
RS232_IO_1 PROC FAR
----- VECTOR TO APPROPRIATE ROUTINE -----
81 0000 FB STI ; INTERRUPTS BACK ON
82 0001 12 PUSH DS ; SAVE SEGMENT
83 0002 52 PUSH DX
84 0003 56 PUSH SI
85 0004 57 PUSH DI
86 0005 51 PUSH CX
87 0006 53 PUSH BX
88 0007 8B F2 MOV SI,DX ; RS232 VALUE TO (SI)
89 0009 8B FA MOV DI,DX ; AND TO (DI) (FOR TIMEOUTS)
90 000B C1 EA 02 SHR DX,2 ; TEST PARAMETER
91 000E 75 20 JNZ A3 ; RETURN IF NOT IN RANGE
92 0010 D1 E6 SHL SI,1 ; WORD OFFSET
93 0012 E8 0000 E CALL DDS
94 0015 8B 94 0000 R MOV DX,#RS232_BASE[SI] ; GET BASE ADDRESS
95 0019 0B D2 OR DX,DX ; TEST FOR 0 BASE ADDRESS
96 001B 74 13 JZ ; RETURN
97 001D 0A E4 OR AH,AH ; TEST FOR (AH)= 00H
98 001F 74 16 JZ A4 ; COMMO INITIALIZATION
99 0021 FE CC DEC AH ; TEST FOR (AH)= 01H
100 0023 74 4B JZ A5 ; SEND (AL)
101 0025 FE CC DEC AH ; TEST FOR (AH)= 02H
102 0027 74 70 JZ A12 ; RECEIVE INTO (AL)
103 0029 FE CC A2: DEC AH ; TEST FOR (AH)= 03H
104 002B 75 03 JNZ A3
105 002D E9 00BB R JMP A18 ; COMMUNICATION STATUS
106 0030 5B ; RETURN FROM RS232
107 0031 59 POP CX
108 0032 5F POP DI
109 0033 5E POP SI
110 0034 5A POP DX
111 0035 5F POP DS
112 0036 5A POP DX
113 0038 1F POP DS
114 0036 CF IRET ; RETURN TO CALLER, NO ACTION

```

SECTION 5

```

115                                     PAGE
116                                     |----- INITIALIZE THE COMMUNICATIONS PORT
117
118 0037                                     A4:
119 0037 8A E0                               MOV     AH,AL           ; SAVE INITIALIZATION PARAMETERS IN (AH)
120 0039 83 C2 03                          ADD     DX,3           ; POINT TO 8250 CONTROL REGISTER
121 003C B0 80                               MOV     AL,80H        ; SET DLAB=1
122 003E EE                               OUT     DX,AL
123
124                                     |----- DETERMINE BAUD RATE DIVISOR
125
126 003F 8A D4                               MOV     DL,AH         ; GET PARAMETERS TO (DL)
127 0041 B1 04                               MOV     CL,4
128 0043 D2 C2 03                          ROL     DL,CL
129 0045 81 E2 000E                        AND     DX,0EH        ; ISOLATE THEM
130 0049 BF 0000 E                          MOV     D1,OFFSET A1 ; BASE OF TABLE
131 004C 03 FA                               ADD     D1,DX         ; PUT INTO INDEX REGISTER
132 004E 8B 94 0000 R                       MOV     DX,0RS232_BASE[S1] ; POINT TO HIGH ORDER OF DIVISOR
133 0052 42                                INC     DX
134 0053 2E 8A 45 01                       MOV     AL,CS:[D1]+1 ; GET HIGH ORDER OF DIVISOR
135 0057 EE                               OUT     DX,AL         ; SET hi OF DIVISOR TO 0
136 0058 4A                                DEC     DX
137 0059 EB 00                               JMP     $-2           ; I/O DELAY
138 005B 2E 8A 05                           MOV     AL,CS:[D1]   ; GET LOW ORDER OF DIVISOR
139 005E EE                               OUT     DX,AL         ; SET LOW OF DIVISOR
140 005F 83 C2 03                          ADD     DX,3
141 0062 8A C4                               MOV     AL,AH        ; GET PARAMETERS BACK
142 0064 24 1F                               AND     AL,01FH      ; STRIP OFF THE BAUD BITS
143 0066 EE                               OUT     DX,AL         ; LINE CONTROL TO 8 BITS
144 0067 4A                                DEC     DX
145 0068 4A                                DEC     DX
146 0069 EB 00                               JMP     $-2           ; I/O DELAY
147 006B B0 00                               MOV     AL,0
148 006D EE                               OUT     DX,AL        ; INTERRUPT ENABLES ALL OFF
149 006E EB 4B                               JMP     SHORT A18    ; COM_STATUS
150
151                                     |----- SEND CHARACTER IN (AL) OVER COMMO LINE
152
153 0070                                     A5:
154 0070 50                               PUSH    AX            ; SAVE CHAR TO SEND
155 0071 83 C2 04                          ADD     DX,4         ; MODEM CONTROL REGISTER
156 0074 B0 03                               MOV     AL,3         ; DTR AND RTS
157 0076 EE                               OUT     DX,AL        ; DATA TERMINAL READY, REQUEST TO SEND
158 0077 42                                INC     DX           ; MODEM STATUS REGISTER
159 0078 42                                INC     DX
160 0079 B7 30                               MOV     BH,30H      ; DATA SET READY & CLEAR TO SEND
161 007B E8 00CA R                          CALL    WAIT_FOR_STATUS ; ARE BOTH TRUE
162 007E 74 08                               JE      A9            ; YES, READY TO TRANSMIT CHAR
163
164 0080 59                                     A7:
165 0081 8A C1                               POP     CX           ; RELOAD DATA BYTE
166 0083 EE                               MOV     AL,CL
167 0085 80 CC 80                          OR      AH,80H      ; INDICATE TIME OUT
168 0086 EB A8                               JMP     A3           ; RETURN
169
170 0088                                     A9:
171 0088 4A                                DEC     DX           ; CLEAR TO SEND
172 0089 EE                               MOV     AL,CL        ; LINE STATUS REGISTER
173 0089 B7 20                               MOV     BH,20H      ; WAIT SEND
174 008B E8 00CA R                          CALL    WAIT_FOR_STATUS ; IS TRANSMITTER READY
175 008E 75 F0                               JNZ    A7           ; TEST FOR TRANSMITTER READY
176 0090                               JNZ    A7           ; RETURN WITH TIME OUT SET
177 0090 83 EA 05                          SUB     DX,5         ; OUT_CHAR
178 0092 89                               POP     CX           ; DATA PORT
179 0094 8A C1                               MOV     AL,CL        ; RECOVER IN CX TEMPORARILY
180 0096 EE                               OUT     DX,AL        ; MOVE CHAR TO AL FOR OUT, STATUS IN AH
181 0097 EB 97                               JMP     A3           ; OUTPUT CHARACTER
182
183                                     |----- RECEIVE CHARACTER FROM COMMO LINE
184
185 0099                                     A12:
186 0099 83 C2 04                          ADD     DX,4         ; MODEM CONTROL REGISTER
187 009C B0 01                               MOV     AL,1         ; DATA TERMINAL READY
188 009E EE                               OUT     DX,AL        ; TEST FOR DSR
189 009F 42                                INC     DX           ; MODEM STATUS REGISTER
190 00A0 42                                INC     DX
191 00A1                               INC     DX
192 00A1 B7 20                               MOV     BH,20H      ; WAIT DSR
193 00A3 E8 00CA R                          CALL    WAIT_FOR_STATUS ; DATA SET READY
194 00A6 75 DB                               JNZ    A5           ; TEST FOR DSR
195 00A8 EE                               MOV     AL,A8        ; RETURN WITH ERROR
196 00A8 4A                                DEC     DX           ; WAIT DSR END
197 00A9 EE                               MOV     AL,A8        ; LINE STATUS REGISTER
198 00A9 B7 01                               MOV     BH,1         ; WAIT_RECVD
199 00AB E8 00CA R                          CALL    WAIT_FOR_STATUS ; RECEIVE BUFFER FULL
200 00AE 76 D3                               JNZ    A5           ; TEST FOR RECEIVE BUFFER FULL
201 00B0                               JNZ    A5           ; SET TIME OUT ERROR
202 00B0 80 E4 1E                          AND     AH,0001110B ; GET_CHAR
203 00B1                               JNZ    A5           ; TEST FOR ERROR CONDITIONS ON RECEIVE
204 00B3 8B 94 0000 R                       MOV     DX,0RS232_BASE[S1] ; DATA PORT
205 00B7 EC                               IN      AL,DX        ; GET CHARACTER FROM LINE
206 00B8 E9 0030 R                          JMP     A3           ; RETURN
207
208                                     |----- COMMO PORT STATUS ROUTINE
209
210 00BB                                     A18:
211 00BB 8B 94 0000 R                       MOV     DX,0RS232_BASE[S1] ; CONTROL PORT
212 00BF 83 C2 05                          ADD     DX,5
213 00C2 EC                               IN      AL,DX        ; GET LINE CONTROL STATUS
214 00C3 8A E0                               MOV     AH,AL        ; PUT IN (AH) FOR RETURN
215 00C5 42                                INC     DX           ; POINT TO MODEM STATUS REGISTER
216 00C6 EC                               IN      AL,DX        ; GET MODEM CONTROL STATUS
217 00C7 E9 0030 R                          JMP     A3           ; RETURN

```

```

218                                     PAGE
219                                     ;-----
220                                     ; WAIT FOR STATUS ROUTINE ;
221 ;ENTRY: (BH) = STATUS BIT(S) TO LOOK FOR ;
222 ; (DX) = ADDRESS OF STATUS REG ;
223 ;EXIT: ZERO FLAG ON = STATUS FOUND ;
224 ; ZERO FLAG OFF = TIMEOUT. ;
225 ; (AH) = LAST STATUS READ ;
226                                     ;-----
227
228 00CA                                     WAIT_FOR_STATUS PROC NEAR
229 00CA 8A 9D 007C R                       MOV     BL,#RS232_TIM_OUT[D1] ; LOAD OUTER LOOP COUNT
230
231                                     ;----- ADJUST OUTER LOOP COUNT
232
233 00CE 55                                     PUSH    BP ; SAVE (BP)
234 00CF 53                                     PUSH    BX ; SAVE (BX)
235 00D0 5D                                     POP     BP ; USE BP FOR OUTER LOOP COUNT
236 00D1 81 E5 00FF                           AND     BP,00FFH ; STRIP HIGH BITS
237 00D5 D1 D5                                 RCL     BP,1 ; MULTIPLY OUTER COUNT BY 4
238 00D7 D1 D5                                 RCL     BP,1
239 00D9                                         WFS0:
240 00D9 2B C9                                 SUB     CX,CX
241 00DB                                         WFS1:
242 00DB EC                                     IN     AL,DX ; GET STATUS
243 00DC 8A E0                                 MOV     AH,AL ; MOVE TO (AH)
244 00DE 22 CT                                 AND     AL,BH ; ISOLATE BITS TO TEST
245 00E0 3A CT                                 CMP     AL,BH ; EXACTLY = TO MASK
246 00E2 74 07                                 JE      WFS_END ; RETURN WITH ZERO FLAG ON
247
248 00E4 E2 F5                                 LOOP   WFS1 ; TRY AGAIN
249
250 00E6 4D                                     DEC     BP
251 00E7 75 F0                                 JNZ    WFS0
252
253 00E9 0A FF                                 OR      BH,BH ; SET ZERO FLAG OFF
254 00EB                                         WFS_END:
255 00EB 5D                                     POP     BP ; RESTORE (BP)
256 00EC C3                                     RET
257
258 00ED                                     WAIT_FOR_STATUS ENDP
259
260 00ED                                     RS232_IO_1 ENDP
261
262 00ED                                     CODE     ENDS
263                                     END
  
```

SECTION 5

```

1 PAGE 118,121
2 TITLE VIDEO1 --- 03/24/86 VIDEO DISPLAY BIOS
3 .LIST
4 0000 CODE SEGMENT BYTE PUBLIC
5
6 PUBLIC ACT_DISP_PAGE
7 PUBLIC READ_AC_CURRENT
8 PUBLIC READ_CURSOR
9 PUBLIC READ_DOT
10 PUBLIC READ_LPEN
11 PUBLIC SCROLL_DOWN
12 PUBLIC SCROLL_UP
13 PUBLIC SET_COLOR
14 PUBLIC SET_CPOS
15 PUBLIC SET_CTYPE
16 PUBLIC SET_MODE
17 PUBLIC WRITE_AC_CURRENT
18 PUBLIC WRITE_C_CURRENT
19 PUBLIC WRITE_DOT
20 PUBLIC WRITE_TTY
21 PUBLIC VIDEO_ID
22 PUBLIC VIDEO_STATE
23
24 EXTRN BEEP_NEAR ; SPEAKER BEEP ROUTINE
25 EXTRN CRT_CHAR_GEN_NEAR ; CHARACTER GENERATOR GRAPHICS TABLE
26 EXTRN DOSTNEAR ; LOAD (DS) WITH DATA SEGMENT SELECTOR
27 EXTRN M5:WORD ; REGEN BUFFER LENGTH TABLE
28 EXTRN M6:BYTE ; COLUMNS PER MODE TABLE
29 EXTRN M7:BYTE ; MODE SET VALUE PER MODE TABLE
30
31 --- INT 10 H ---
32
33 VIDEO_ID
34 THESE ROUTINES PROVIDE THE CRT DISPLAY INTERFACE
35 THE FOLLOWING FUNCTIONS ARE PROVIDED:
36
37 (AH) = 00H SET MODE (AL) CONTAINS MODE VALUE
38 (AL) = 00H 40X25 BW MODE (POWER ON DEFAULT)
39 (AL) = 01H 40X25 COLOR
40 (AL) = 02H 80X25 BW
41 (AL) = 03H 80X25 COLOR
42 GRAPHICS MODES
43 (AL) = 04H 320X200 COLOR
44 (AL) = 05H 320X200 BW MODE
45 (AL) = 06H 640X200 BW MODE
46 (AL) = 07H 80X25 MONOCHROME (USED INTERNAL TO VIDEO ONLY)
47 *** NOTES -BW MODES OPERATE SAME AS COLOR MODES, BUT COLOR
48 BURST IS NOT ENABLED
49 -CURSOR IS NOT DISPLAYED IN GRAPHICS MODE
50
51 (AH) = 01H SET CURSOR TYPE
52 (CH) = BITS 4-0 = START LINE FOR CURSOR
53 ** HARDWARE WILL ALWAYS CAUSE BLINK
54 ** SETTING BIT 5 OR 6 WILL CAUSE ERRATIC BLINKING
55 OR NO CURSOR AT ALL
56 (CL) = BITS 4-0 = END LINE FOR CURSOR
57
58 (AH) = 02H SET CURSOR POSITION
59 (DH,DL) = ROW,COLUMN (00H,00H) IS UPPER LEFT
60 (BH) = PAGE NUMBER (MUST BE 00H FOR GRAPHICS MODES)
61
62 (AH) = 03H READ CURSOR POSITION
63 (BH) = PAGE NUMBER (MUST BE 00H FOR GRAPHICS MODES)
64 ON EXIT (DH,DL) = ROW,COLUMN OF CURRENT CURSOR
65 (CH,CL) = CURSOR MODE CURRENTLY SET
66
67 (AH) = 04H READ LIGHT PEN POSITION
68 ON EXIT:
69 (AH) = 00H -- LIGHT PEN SWITCH NOT DOWN/NOT TRIGGERED
70 (AH) = 01H -- VALID LIGHT PEN VALUE IN REGISTERS
71 (DH,DL) = ROW,COLUMN OF CHARACTER LP POSITION
72 (CH) = RASTER LINE (0-199)
73 (BX) = PIXEL COLUMN (0-319,639)
74
75 (AH) = 05H SELECT ACTIVE DISPLAY PAGE (VALID ONLY FOR ALPHA MODES)
76 (AL) = NEW PAGE VALUE (0-7 FOR MODES 0&1, 0-3 FOR MODES 2&3)
77
78 (AH) = 06H SCROLL ACTIVE PAGE UP
79 (AL) = NUMBER OF LINES, ( LINES BLANKED AT BOTTOM OF WINDOW )
80 (AL) = 00H MEANS BLANK ENTIRE WINDOW
81 (CH,CL) = ROW,COLUMN OF UPPER LEFT CORNER OF SCROLL
82 (DH,DL) = ROW,COLUMN OF LOWER RIGHT CORNER OF SCROLL
83 (BH) = ATTRIBUTE TO BE USED ON BLANK LINE
84
85 (AH) = 07H SCROLL ACTIVE PAGE DOWN
86 (AL) = NUMBER OF LINES, INPUT LINES BLANKED AT TOP OF WINDOW
87 (AL) = 00H MEANS BLANK ENTIRE WINDOW
88 (CH,CL) = ROW,COLUMN OF UPPER LEFT CORNER OF SCROLL
89 (DH,DL) = ROW,COLUMN OF LOWER RIGHT CORNER OF SCROLL
90 (BH) = ATTRIBUTE TO BE USED ON BLANK LINE
91
92 CHARACTER HANDLING ROUTINES
93
94 (AH) = 08H READ ATTRIBUTE/CHARACTER AT CURRENT CURSOR POSITION
95 (BH) = DISPLAY PAGE (VALID FOR ALPHA MODES ONLY)
96 ON EXIT:
97 (AL) = CHAR READ
98 (AH) = ATTRIBUTE OF CHARACTER READ (ALPHA MODES ONLY)
99
100 (AH) = 09H WRITE ATTRIBUTE/CHARACTER AT CURRENT CURSOR POSITION
101 (BH) = DISPLAY PAGE (VALID FOR ALPHA MODES ONLY)
102 (CX) = COUNT OF CHARACTERS TO WRITE
103 (AL) = CHAR TO WRITE
104 (BL) = ATTRIBUTE OF CHARACTER (ALPHA)/COLOR OF CHAR (GRAPHICS)
105 SEE NOTE ON WRITE DOT FOR BIT 7 OF BL = 1.
106
107 (AH) = 0AH WRITE CHARACTER ONLY AT CURRENT CURSOR POSITION
108 (BH) = DISPLAY PAGE (VALID FOR ALPHA MODES ONLY)
109 (CX) = COUNT OF CHARACTERS TO WRITE
110 (AL) = CHAR TO WRITE
111 NOTE: USE FUNCTION (AH) = 09H IN GRAPHICS MODES
112 FOR READ/WRITE CHARACTER INTERFACE WHILE IN GRAPHICS MODE, THE
113 CHARACTERS ARE FORMED FROM A CHARACTER GENERATOR IMAGE
114 MAINTAINED IN THE SYSTEM ROM. ONLY THE 1ST 128 CHARS
115 ARE CONTAINED THERE. TO READ/WRITE THE SECOND 128 CHARS,
116 THE USER MUST INITIALIZE THE POINTER AT INTERRUPT 1FH
117 (LOCATION 0007CH) TO POINT TO THE 1K BYTE TABLE CONTAINING
118 THE CODE POINTS FOR THE SECOND 128 CHARS (128-256).
119
120 FOR WRITE CHARACTER INTERFACE IN GRAPHICS MODE, THE REPLICATION FACTOR
121 CONTAINED IN (CX) ON ENTRY WILL PRODUCE VALID RESULTS ONLY
122 FOR CHARACTERS CONTAINED ON THE SAME ROW. CONTINUATION TO
123 SUCCEEDING LINES WILL NOT PRODUCE CORRECTLY.

```

```

115 ; GRAPHICS INTERFACE ;
116 ; ;
117 ; (AH)= 0BH SET COLOR PALETTE ;
118 ; (BH) = PALETTE COLOR ID BEING SET (0-127) ;
119 ; (BL) = COLOR VALUE TO BE USED WITH THAT COLOR ID ;
120 ; NOTE: FOR THE CURRENT COLOR CARD, THIS ENTRY POINT HAS ;
121 ; MEANING ONLY FOR 320X200 GRAPHICS. ;
122 ; COLOR ID = 0 SELECTS THE BACKGROUND COLOR (0-15) ;
123 ; COLOR ID = 1 SELECTS THE PALETTE TO BE USED: ;
124 ; 0 = GREEN(1)/RED(2)/YELLOW(3) ;
125 ; 1 = CYAN(1)/MAGENTA(2)/WHITE(3) ;
126 ; IN 40X25 OR 80X25 ALPHA MODES, THE VALUE SET FOR ;
127 ; PALETTE COLOR 0 INDICATES THE BORDER COLOR ;
128 ; TO BE USED (VALUES 0-31, WHERE 16-31 SELECT ;
129 ; THE HIGH INTENSITY BACKGROUND SET. ;
130 ; ;
131 ; (AH)= 0CH WRITE DOT ;
132 ; (DX) = ROW NUMBER ;
133 ; (CX) = COLUMN NUMBER ;
134 ; (AL) = COLOR VALUE ;
135 ; IF BIT 7 OF AL = 1, THEN THE COLOR VALUE IS EXCLUSIVE ;
136 ; OR'd WITH THE CURRENT CONTENTS OF THE DOT ;
137 ; ;
138 ; (AH)= 0DH READ DOT ;
139 ; (DX) = ROW NUMBER ;
140 ; (CX) = COLUMN NUMBER ;
141 ; (AL) RETURNS THE DOT READ ;
142 ; ;
143 ; ASCII TELETYPE ROUTINE FOR OUTPUT ;
144 ; ;
145 ; (AH)= 0EH WRITE TELETYPE TO ACTIVE PAGE ;
146 ; (AL) = CHAR TO WRITE ;
147 ; (BL) = FOREGROUND COLOR IN GRAPHICS MODE ;
148 ; NOTE -- SCREEN WIDTH IS CONTROLLED BY PREVIOUS MODE SET ;
149 ; (AH)= 0FH CURRENT VIDEO STATE ;
150 ; RETURNS THE CURRENT VIDEO STATE ;
151 ; (AL) = MODE CURRENTLY SET ( SEE (AH)= 00H FOR EXPLANATION) ;
152 ; (AH) = NUMBER OF CHARACTER COLUMNS ON SCREEN ;
153 ; (BH) = CURRENT ACTIVE DISPLAY PAGE ;
154 ; ;
155 ; (AH)= 10H RESERVED ;
156 ; (AH)= 11H RESERVED ;
157 ; (AH)= 12H RESERVED ;
158 ; (AH)= 13H WRITE STRING ;
159 ; ES:BP - POINTER TO STRING TO BE WRITTEN ;
160 ; CX - LENGTH OF CHARACTER STRING TO WRITTEN ;
161 ; DX - CURSOR POSITION FOR STRING TO BE WRITTEN ;
162 ; BH - PAGE NUMBER ;
163 ; (AL)= 00H WRITE CHARACTER STRING ;
164 ; BL - ATTRIBUTE ;
165 ; STRING IS <CHAR,CHAR, ... ,CHAR> ;
166 ; CURSOR NOT MOVED ;
167 ; (AL)= 01H WRITE CHARACTER STRING AND MOVE CURSOR ;
168 ; BL - ATTRIBUTE ;
169 ; STRING IS <CHAR,CHAR, ... ,CHAR> ;
170 ; CURSOR IS MOVED ;
171 ; (AL)= 02H WRITE CHARACTER AND ATTRIBUTE STRING ;
172 ; (VALID FOR ALPHA MODES ONLY) ;
173 ; STRING IS <CHAR,ATTR,CHAR,ATTR .. ,CHAR,ATTR> ;
174 ; CURSOR IS NOT MOVED ;
175 ; (AL)= 03H WRITE CHARACTER AND ATTRIBUTE STRING AND MOVE CURSOR ;
176 ; (VALID FOR ALPHA MODES ONLY) ;
177 ; STRING IS <CHAR,ATTR,CHAR,ATTR .. ,CHAR,ATTR> ;
178 ; CURSOR IS MOVED ;
179 ; NOTE: CARRIAGE RETURN, LINE FEED, BACKSPACE, AND BELL ARE ;
180 ; TREATED AS COMMANDS RATHER THAN PRINTABLE CHARACTERS. ;
181 ; ;
182 ; BX,CX,DX,S1,D1,BP,SP,DS,ES,SS PRESERVED DURING CALLS EXCEPT FOR ;
183 ; BX,CX,DX RETURN VALUES ON FUNCTIONS 03H,04H,0DH AND 0DH. ON ALL CALLS ;
184 ; AX IS MODIFIED. ;
185 ;-----;
186 ; ASSUME CS:CODE,DS:DATA,ES:NOTHING ;
187 ; ;
188 MI DW OFFSET SET_MODE ; TABLE OF ROUTINES WITHIN VIDEO I/O ;
189 DW OFFSET SET_CTYPE ; ;
190 DW OFFSET SET_CPOS ; ;
191 DW OFFSET READ_CURSOR ; ;
192 DW OFFSET READ_LPEN ; ;
193 DW OFFSET ACT_DISP_PAGE ; ;
194 DW OFFSET SCROLL_UP ; ;
195 DW OFFSET SCROLL_DOWN ; ;
196 DW OFFSET READ_AC_CURRENT ; ;
197 DW OFFSET WRITE_AC_CURRENT ; ;
198 DW OFFSET SET_COLOR ; ;
199 DW OFFSET WRITE_DOT ; ;
200 DW OFFSET READ_DOT ; ;
201 DW OFFSET WRITE_TTY ; ;
202 DW OFFSET VIDEO_STATE ; ;
203 DW OFFSET VIDEO_RETURN ; RESERVED ;
204 DW OFFSET VIDEO_RETURN ; RESERVED ;
205 DW OFFSET WRITE_STRING ; CASE 13H, WRITE STRING ;
206 = 0026 ;
207 ;
208 ; VIDEO_IO_1 PROC NEAR ; ENTRY POINT FOR ORG 0F065H ;
209 STI ; INTERRUPTS BACK ON ;
210 CLD ; SET DIRECTION FORWARD ;
211 CMP AH,MIL/2 ; SET DIRECTION FORWARD ;
212 JNB M4 ; TEST FOR WITHIN TABLE RANGE ;
213 ; BRANCH TO EXIT IF NOT A VALID COMMAND ;
214 ; ;
215 PUSH ES ; ;
216 PUSH DS ; SAVE WORK AND PARAMETER REGISTERS ;
217 PUSH DX ; ;
218 PUSH CX ; ;
219 PUSH BX ; ;
220 PUSH SI ; ;
221 PUSH DI ; ;
222 MOV SI,DATA ; POINT DS: TO DATA SEGMENT ;
223 MOV DS,SI ; ;
224 MOV SI,AX ; SAVE COMMAND/DATA INTO (SI) REGISTER ;
225 MOV AL,BYTE PTR @EQUIP_FLAG ; GET THE EQUIPMENT FLAG VIDEO BITS ;
226 AND AL,30H ; ISOLATE CRT SWITCHES ;
227 AND AL,30H ; IS SETTING FOR MONOCHROME CARD? ;
228 MOV DI,0B800H ; GET SEGMENT FOR COLOR CARD ;
    
```

SECTION 5

```

229 0048 76 03      JNE     M2          ; SKIP IF NOT MONOCHROME CARD
230 004A BF B000    MOV     DI,0B000H  ; ELSE GET SEGMENT FOR MONOCHROME CARD
231 004D             ;
232 004D 8E C7      MOV     ES,DI      ; SET UP TO POINT AT VIDEO MEMORY AREAS
233 004F 8A C4      MOV     AL,AH      ; PLACE COMMAND IN LOW BYTE OF (AX)
234 0051 98         CBW             ; AND FORM BYTE OFFSET WITH COMMAND
235 0052 01 E0      SAL     AX,1       ; TIMES 2 FOR WORD TABLE LOOKUP
236 0054 96         XCHG    SI,AX      ; MOVE OFFSET INTO LOOK UP REGISTER (SI)
237             ; AND RESTORE COMMAND/DATA INTO (AX)
238 0055 8A 26 0049 R MOV     AH,*CRT_MODE ; MOVE CURRENT MODE INTO (AH) REGISTER
239             ;
240 0059 2E1 FF A4 0000 R JMP     WORD PTR CS:[SI+OFFSET M1] ; GO TO SELECTED FUNCTION
241             ;
242 005E             M4:             ; COMMAND NOT VALID
243 005E CF             ; DO NOTHING IF NOT IN VALID RANGE
244 005F             ;
245             ;
246             ; SET_MODE
247             ; THIS ROUTINE INITIALIZES THE ATTACHMENT TO
248             ; THE SELECTED MODE. THE SCREEN IS BLANKED.
249             ;
250             ; INPUT
251             ; *EQUIP_FLAG BITS 5-4 = MODE/WIDTH
252             ; 11 = MONOCHROME (FORCES MODE 7)
253             ; 01 = COLOR ADAPTER 40x25 (MODE 0 DEFAULT)
254             ; 10 = COLOR ADAPTER 80x25 (MODE 2 DEFAULT)
255             ; (AL) = COLOR MODE REQUESTED ( RANGE 0 - 6 )
256             ; OUTPUT
257             ; NONE
258             ;
259 005F             SET_MODE PROC NEAR
260 0060 8A 03D4     MOV     DX,03D4H   ; ADDRESS OF COLOR CARD
261 0062 88 3E 0010 R MOV     DI,*EQUIP_FLAG ; GET EQUIPMENT FLAGS SETTING
262 0064 83 FF 30    AND     DI,30H    ; ISOLATE CRT SWITCHES
263 0066 83 FF 30    CMP     DI,30H    ; IS BW CARD INSTALLED AS PRIMARY
264 0068             JNE     M8C       ; SKIP AND CHECK IF COLOR
265 006A 80 07      MOV     AL,7      ; ELSE INDICATE INTERNAL BW CARD MODE
266 0071 B2 B4      MOV     DL,0B4H   ; SET ADDRESS OF BW (MONOCHROME) CARD
267 0073 EB 0D      JMP     SHORT M8  ; CONTINUE WITH FORCED MODE 7
268             ;
269 0075 3C 07      M8C:    CMP     AL,7      ; CHECK FOR VALID COLOR MODES 0-6
270 0077 72 09      JB     M8         ; CONTINUE IF BELOW MODE 7
271 0079 80 00      MOV     AL,0      ; FORCE DEFAULT 40x25 BW MODE
272 007B 83 FF 10    CMP     DI,10H   ; CHECK FOR *EQUIP_FLAG AT 40x25 COLOR
273 007D 74 02      JE     M8         ; CONTINUE WITH MODE 0 IF NOT
274 007F 80 02      MOV     AL,2      ; ELSE FORCE MODE 2
275             ;
276 0082 A2 0049 R     M8:     MOV     *CRT_MODE,AL ; SAVE MODE IN GLOBAL VARIABLE
277 0085 89 16 0063 R MOV     *ADDR_6845,DX ; SAVE ADDRESS OF BASE
278 0088 0E 06 0084 R DS     *ROWS,25-1 ; INITIALIZE DEFAULT ROW COUNT OF 25
279 008B 50         PUSH    AX         ; SAVE POINTER TO DATA SEGMENT
280 008D 50         PUSH    AX         ; SAVE MODE NUMBER (AL)
281 008F 50         PUSH    AX         ; CLEAR HIGH BYTE OF MODE
282 0091 8B F0      MOV     SI,AX      ; SET TABLE POINTER, INDEXED BY MODE
283 0093 2E1 8A 84 0000 E MOV     AL,CS:[SI + OFFSET M7] ; GET THE MODE SET VALUE FROM TABLE
284 0096 A2 0065 R   MOV     *CRT_MODE_SET,AL ; SAVE THE MODE SET VALUE
285 0098 24 37      AND     AL,037H   ; VIDEO OFF, SAVE HIGH RESOLUTION BIT
286 009D 52         PUSH    DX         ; SAVE OUTPUT PORT VALUE
287 009F 83 C2 04   ADD     DX,4       ; POINT TO CONTROL REGISTER
288 00A1 EE         OUT     DX,AL      ; RESET VIDEO TO OFF TO SUPPRESS ROLLING
289 00A3 5A         POP     DX         ; BACK TO BASE REGISTER
290 00A5 2B DB      ASSUME DS:ABS0    ; SET UP FOR ABS0 SEGMENT
291 00A7 8E DB      SUB     DS,BX      ; BX,BX
292 00A9 C5 1E 0074 R LDS     BX,*PARAM_PTR ; ESTABLISH VECTOR TABLE ADDRESSING
293 00AB 5A         POP     DS         ; GET POINTER TO VIDEO PARAMS
294 00AD 56         POP     AX         ; RECOVER MODE NUMBER IN (AL)
295 00AF 89 0010    MOV     CX,16     ; LENGTH OF EACH ROW OF TABLE
296 00B1 0E         CMP     AL,2      ; DETERMINE WHICH ONE TO USE
297 00B3 72 0E     JC     M9         ; MODE IS 0 OR 1
298 00B5 03 D9     ADD     BX,CX     ; NEXT ROW OF INITIALIZATION TABLE
299 00B7 3C 04      CMP     AL,4      ; MODE IS 2 OR 3
300 00B9 72 08     JC     M9         ; MOVE TO GRAPHICS ROW OF INIT_TABLE
301 00BB 03 D9     ADD     BX,CX     ; MODE IS 2 OR 3
302 00BD 3C 07      CMP     AL,7      ; MODE IS 4,5, OR 6
303 00BF 72 02     JC     M9         ; MOVE TO BW CARD ROW OF INIT_TABLE
304 00C1 03 D9     ADD     BX,CX     ;
305             ;
306             ;
307             ; ----- BX POINTS TO CORRECT ROW OF INITIALIZATION TABLE
308             ;
309 00C1             M9:     ; OUT INIT
310 00C3 50         PUSH    AX         ; SAVE MODE IN (AL)
311 00C5 89 47 0A    MOV     AX,[BX+10] ; GET THE CURSOR MODE FROM THE TABLE
312 00C7 1E         XCHG    AH,AL     ; PUT CURSOR MODE IN CORRECT POSITION
313 00C9             PUSH    DS         ; SAVE TABLE SEGMENT POINTER
314 00CB             ASSUME DS:DATA    ;
315 00CD             CALL    DDS        ; POINT DS TO DATA SEGMENT
316 00CF             MOV     *CURSOR_MODE,AX ; PLACE INTO BIOS DATA SAVE AREA
317 00D1             ASSUME DS:CODE    ;
318 00D3             POP     DS         ; RESTORE THE TABLE SEGMENT POINTER
319 00D5             XOR     AH,AH     ; AH IS REGISTER NUMBER DURING LOOP
320             ;
321             ; ----- LOOP THROUGH TABLE, OUTPUTTING REGISTER ADDRESS, THEN VALUE FROM TABLE
322             ;
323 00D1             M10:    ; INITIALIZATION LOOP
324 00D3             MOV     AL,AH     ; GET 6845 REGISTER NUMBER
325 00D5             OUT     DX,AL    ;
326 00D7             INC     DX,AL    ; POINT TO DATA PORT
327 00D9             INC     DX,AL    ; NEXT REGISTER VALUE
328 00DB             MOV     AL,[BX]  ; GET TABLE VALUE
329 00DD             OUT     DX,AL    ; OUT TO CHIP
330 00DF             INC     BX       ; NEXT IN TABLE
331 00E1             DEC     DX       ; BACK TO POINTER REGISTER
332 00E3             LOOP   M10      ; DO THE WHOLE TABLE
333 00E5             POP     AX       ; GET MODE BACK INTO (AL)
334 00E7             POP     DS       ; RECOVER SEGMENT VALUE
335             ;
336             ; ----- FILL REGEN AREA WITH BLANK
337             ;
338 00E0 33 FF      XOR     DI,DI     ; SET UP POINTER FOR REGEN
339 00E2 89 3E 004E R MOV     *CRT_START,DI ; START ADDRESS SAVED IN GLOBAL
340 00E4 C6 06 0062 R MOV     *ACTIVE_PAGE,0 ; SET PAGE VALUE
341 00E6 B9 2000    MOV     CX,8192   ; NUMBER OF WORDS IN COLOR CARD
342 00E8 3C 04      CMP     AL,4      ; TEST FOR GRAPHICS

```

```

343 00F0 72 0A          JC      M12          ; NO GRAPHICS INIT
344 00F2 3C 07          CMP     AL,7        ; TEST FOR SW CARD
345 00F4 74 04          JE      M11         ; SW CARD INIT
346 00F6 33 C0          XOR     AX,AX       ; CLEAR BUFFER
347 00F8 EB 05          JMP     SHORT M13   ; SW CARD INIT
348 00FA B6 08          MOV     CH,08H     ; BUFFER SIZE ON SW CARD (2048)
349 00FC B8 08          MOV     CH,08H     ; NO GRAPHICS INIT
350 00FE B8 08          MOV     CH,08H     ; FILD CHAR FOR ALPHA + ATTRIBUTE
351 00FC B8 0720        MOV     AX,' '*7*H ; CLEAR BUFFER
352 00FF F3/ AB        REP     STOSW      ; FILL THE REGEN BUFFER WITH BLANKS
353
354
355
356
357 0101 8B 16 0063 R   MOV     DX,@ADDR_6845 ; PREPARE TO OUTPUT TO VIDEO ENABLE PORT
358 0105 83 C2 04      ADD     DX,4        ; POINT TO THE MODE CONTROL REGISTER
359 0108 A0 0065 R     MOV     AL,@CRT_MODE_SET ; GET THE MODE SET VALUE
360 010B EE            OUT     DX,AL       ; SET VIDEO ENABLE PORT
361
362
363
364
365 010C 2E1 8A 84 0000 E MOV     AL,CS:[SI + OFFSET M6] ; GET NUMBER OF COLUMNS ON THIS SCREEN
366 0111 98            CBW             ; CLEAR HIGH BYTE
367 0112 A3 004A R     MOV     @CRT_COLS,AX ; INITIALIZE NUMBER OF COLUMNS COUNT
368
369
370
371 0115 81 E6 000E     AND     SI,000EH   ; WORD OFFSET INTO CLEAR LENGTH TABLE
372 0119 2E1 8B 84 0000 E MOV     AX,CS:[SI + OFFSET M5] ; LENGTH TO CLEAR
373 011E A3 004C R     MOV     @CRT_LEN,AX ; SAVE LENGTH OF CRT -- NOT USED FOR BW
374 0121 B9 0008     MOV     CX,8        ; CLEAR ALL CURSOR POSITIONS
375 0124 BF 0080 R     MOV     DI,OFFSET @CURSOR_POSN
376 0127 IE          PUSH    DS         ; ESTABLISH SEGMENT
377 0128 07          POP     ES         ; ADDRESSING
378 0129 33 C0          XOR     AX,AX       ; FILL WITH ZEROES
379 012B F3/ AB        REP     STOSW
380
381
382
383 012D 42            INC     DX          ; SET OVERSCAN PORT TO A DEFAULT
384 012E B0 30          MOV     AL,30H     ; 30H VALUE FOR ALL MODES EXCEPT 640X200
385 0130 8D 2E 0049 R 06 CMP     @CRT_MODE,6 ; SEE IF THE MODE IS 640X200 BW
386 0135 75 02          JNZ     M14        ; IF NOT 640X200, THEN GO TO REGULAR
387 0137 B0 3F          MOV     AL,3FH     ; IF IT IS 640X200, THEN PUT IN 3FH
388 0139
389 0139 EE            OUT     DX,AL      ; OUTPUT THE CORRECT VALUE TO 3D9 PORT
390 013A A2 0066 R     MOV     @CRT_PALETTE,AL ; SAVE THE VALUE FOR FUTURE USE
391
392
393
394 013D
395 013D 5D          POP     BP
396 013E 5F          POP     DI
397 013F 5E          POP     SI
398 0140 5B          POP     BX
399 0141
400 0141 59          POP     CX         ; VIDEO_RETURN_C
401 0142 5A          POP     DX
402 0143 5F          POP     DS
403 0144 07          POP     ES         ; RECOVER SEGMENTS
404 0145 CF          IRET             ; ALL DONE
405 0146
406
407
408
409
410
411
412
413
414 0146
415 0146 B4 0A          MOV     AH,10      ; 6845 REGISTER FOR CURSOR SET
416 0148 89 0E 0060 R  MOV     @CURSOR_MODE,CX ; SAVE IN DATA AREA
417 014C E8 0151 R     CALL    M16        ; OUTPUT CX REGISTER
418 014F EB EC          JMP     VIDEO_RETURN
419
420
421
422
423 0151
424 0151 8B 16 0063 R   MOV     DX,@ADDR_6845 ; ADDRESS REGISTER
425 0155 8A C4          MOV     AL,AH      ; GET VALUE
426 0157 EE            OUT     DX,AL      ; REGISTER SET
427 0158 42            INC     DX          ; DATA REGISTER
428 0159 EB 00          JMP     $+2        ; I/O DELAY
429 015B 8A C5          MOV     AL,CH      ; DATA
430 015D EE            OUT     DX,AL
431 015E 4A            DEC     DX
432 015F 8A C4          MOV     AL,AH      ; POINT TO OTHER DATA REGISTER
433 0161 FE C0          INC     AL         ; SET FOR SECOND REGISTER
434 0163 EE            OUT     DX,AL
435 0164 42            INC     DX
436 0165 EB 00          JMP     $+2        ; I/O DELAY
437 0167 8A C1          MOV     AL,CL      ; SECOND DATA VALUE
438 0169 EE            OUT     DX,AL
439 016A C3            RET
440 016B
441
442
443
444
445
446
447
448
449
450
451
452 016B
453 016B 8A C7          MOV     AL,BH      ; MOVE PAGE NUMBER TO WORK REGISTER
454 016D 98          CBW             ; CONVERT PAGE TO WORD VALUE
455 016E D1 E0          SAL     AX,1       ; WORD OFFSET
456 0170 96          XCHG    AX,SI     ; USE INDEX REGISTER

```

SECTION 5

```

457 0171 89 94 0050 R      MOV     [SI+OFFSET @CURSOR_POSN],DX      ; SAVE THE POINTER
458 0175 38 3E 0062 R      CMP     @ACTIVE_PAGE,BH
459 0179 75 05             JNZ     M17                               ; SET_CPOS_RETURN
460 017B BB C2             MOV     AX,DX                             ; GET_ROW/COLUMN TO AX
461 017D EB 0182 R        CALL    M18                               ; CURSOR SET
462 0180             SET_CPOS:
463 0180 EB BB             JMP     VIDEO_RETURN                       ; SET_CPOS_RETURN
464 0182             SET_CPOS ENDP
465
466
467
468 0182             ;----- SET CURSOR POSITION, AX HAS ROW/COLUMN FOR CURSOR
469 0182 EB 0204 R        M18: PROC NEAR
470 0185 BB C5             CALL    POSITION                            ; DETERMINE LOCATION IN REGEN BUFFER
471 0187 03 0E 004E R      ADD     CX,@CRT_START                      ; ADD IN THE START ADDRESS FOR THIS PAGE
472 018B D1 F9             SAR     CX,1                               ; DIVIDE BY 2 FOR CHAR ONLY COUNT
473 018D B4 0E             MOV     AH,14                             ; REGISTER NUMBER FOR CURSOR
474 018F E8 0181 R        CALL    M16                               ; OUTPUT THE VALUE TO THE 6845
475 0192 C3             RET
476 0193             M18 ENDP
477
478
479             ;----- READ_CURSOR
480             ; THIS ROUTINE READS THE CURRENT CURSOR VALUE FROM THE
481             ; 6845, FORMATS IT, AND SENDS IT BACK TO THE CALLER
482             ; INPUT
483             ; BH - PAGE OF CURSOR
484             ; DX - ROW, COLUMN OF THE CURRENT CURSOR POSITION
485             ; CX - CURRENT CURSOR MODE
486             ;-----
487 0193             READ_CURSOR PROC NEAR
488 0193 8A DF             MOV     BL,BH
489 0195 32 FF             XOR     BH,BH
490 0197 D1 E3             SAL     BX,1                               ; WORD OFFSET
491 0199 BB 97 0050 R      MOV     DX,[BX+OFFSET @CURSOR_POSN]
492 019D BB 0E 0060 R      MOV     CX,@CURSOR_MODE
493 01A1 BD             POP     BP
494 01A2 BF             POP     DI
495 01A3 9E             POP     SI
496 01A4 5B             POP     BX
497 01A5 58             POP     AX                               ; DISCARD SAVED CX AND DX
498 01A6 58             POP     AX
499 01A7 1F             POP     DS
500 01AB 07             POP     ES
501 01A9 CF             IRET
502 01AA             READ_CURSOR ENDP
503
504             ;----- ACT_DISP_PAGE
505             ; THIS ROUTINE SETS THE ACTIVE DISPLAY PAGE, ALLOWING
506             ; THE FULL USE OF THE MEMORY SET ASIDE FOR THE VIDEO ATTACHMENT
507             ; INPUT
508             ; AL HAS THE NEW ACTIVE DISPLAY PAGE
509             ; OUTPUT
510             ; THE 6845 IS RESET TO DISPLAY THAT PAGE
511             ;-----
512 01AA             ACT_DISP_PAGE PROC NEAR
513 01AA A2 0062 R      MOV     @ACTIVE_PAGE,AL                  ; SAVE ACTIVE PAGE VALUE
514 01AD 98             CWD                                       ; CONVERT (AL) TO WORD
515 01AE 50             PUSH   AX                               ; SAVE PAGE VALUE
516 01AF F7 26 004C R      MUL     WORD PTR @CRT_LEN                ; DISPLAY PAGE TIMES REGEN LENGTH
517 01B3 A3 004E R      MOV     @CRT_START,AX                    ; SAVE START ADDRESS FOR LATER
518 01B6 BB C5             MOV     CX,AX                             ; START ADDRESS TO CX
519 01B8 D1 F9             SAR     CX,1                               ; DIVIDE BY 2 FOR 6845 HANDLING
520 01BA B4 0C             MOV     AH,12                             ; 6845 REGISTER FOR START ADDRESS
521 01BC E8 0181 R        CALL    M16                               ; RECOVER PAGE VALUE
522 01BF 58             POP     BX                               ; *2 FOR WORD OFFSET
523 01C0 D1 E3             SAL     BX,1
524 01C2 BB 87 0050 R      MOV     AX,[BX + OFFSET @CURSOR_POSN]    ; GET CURSOR FOR THIS PAGE
525 01C6 E8 0182 R        CALL    M18                               ; SET THE CURSOR POSITION
526 01C9 E9 013D R      JMP     VIDEO_RETURN
527 01CC             ACT_DISP_PAGE ENDP
528
529             ;----- SET_COLOR
530             ; THIS ROUTINE WILL ESTABLISH THE BACKGROUND COLOR, THE OVERSCAN COLOR,
531             ; AND THE FOREGROUND COLOR SET FOR MEDIUM RESOLUTION GRAPHICS
532             ; INPUT
533             ; (BH) HAS COLOR ID
534             ; IF BH=0, THE BACKGROUND COLOR VALUE IS SET
535             ; FROM THE LOW BITS OF BL (0-3)
536             ; IF BH=1, THE PALETTE SELECTION IS MADE
537             ; BASED ON THE LOW BIT OF BL:
538             ; 0 = GREEN, RED, YELLOW FOR COLORS 1,2,3
539             ; 1 = BLUE, CYAN, MAGENTA FOR COLORS 1,2,3
540             ; (BL) HAS THE COLOR VALUE TO BE USED
541             ; OUTPUT
542             ; THE COLOR SELECTION IS UPDATED
543             ;-----
544 01CC             SET_COLOR PROC NEAR
545 01CC BB 16 0063 R      MOV     DX,@ADDR_6845                    ; I/O PORT FOR PALETTE
546 01D0 83 C2 05         ADD     DX,5                               ; OVERSCAN PORT
547 01D3 A0 0066 R      MOV     AL,@CRT_PALETTE                  ; GET THE CURRENT PALETTE VALUE
548 01D6 0A FF             OR     BH,BH                               ; IS THIS COLOR 0?
549 01D8 75 0E             JNZ     M20                               ; OUTPUT COLOR 1
550
551             ;----- HANDLE COLOR 0 BY SETTING THE BACKGROUND COLOR
552
553 01DA 24 E0             AND     AL,0E0H                           ; TURN OFF LOW 5 BITS OF CURRENT
554 01DC 80 E3             AND     BL,01FH                           ; TURN OFF HIGH 9 BITS OF INPUT VALUE
555 01DF 0A C3             OR     AL,BL                               ; PUT VALUE INTO REGISTER
556 01E1             ; OUTPUT THE PALETTE
557 01E1 EE             OUT     DX,AL
558 01E2 A2 0066 R      MOV     @CRT_PALETTE,AL                  ; OUTPUT COLOR SELECTION TO 3D9 PORT
559 01E5 E9 013D R      JMP     VIDEO_RETURN                       ; SAVE THE COLOR VALUE
560
561             ;----- HANDLE COLOR 1 BY SELECTING THE PALETTE TO BE USED
562
563 01E8             M20:
564 01E8 24 DF             AND     AL,0DFH                           ; TURN OFF PALETTE SELECT BIT
565 01EA D0 EB             SHR     BL,1                               ; TEST THE LOW ORDER BIT OF BL
566 01EC 73 F3             JNC     M19                               ; ALREADY DONE
567 01EE 0C 20             OR     AL,20H                             ; TURN ON PALETTE SELECT BIT
568 01F0 EB EF             JMP     M19                               ; GO DO IT
569 01F2             SET_COLOR ENDP
570

```



```

571 ;-----
572 ; VIDEO STATE
573 ; RETURNS THE CURRENT VIDEO STATE IN AX
574 ; AH = NUMBER OF COLUMNS ON THE SCREEN
575 ; AL = CURRENT VIDEO MODE
576 ; BH = CURRENT ACTIVE PAGE
577 ;-----
578 VIDEO_STATE PROC NEAR
579 MOV AH, BYTE PTR #CRT_COLS ; GET NUMBER OF COLUMNS
580 MOV AL, #CRT_MODE ; CURRENT MODE
581 MOV BH, #ACTIVE_PAGE ; GET CURRENT ACTIVE PAGE
582 POP BP ; RECOVER REGISTERS
583 POP DI
584 POP SI
585 POP CX ; DISCARD SAVED BX
586 JMP M16 ; RETURN TO CALLER
587 ENDP
588 ;-----
589 ; POSITION
590 ; THIS SERVICE ROUTINE CALCULATES THE REGEN BUFFER ADDRESS
591 ; OF A CHARACTER IN THE ALPHA MODE
592 ; INPUT
593 ; AX = ROW, COLUMN POSITION
594 ; OUTPUT
595 ; AX = OFFSET OF CHAR POSITION IN REGEN BUFFER
596 ;-----
597 POSITION PROC NEAR
598 PUSH BX ; SAVE REGISTER
599 XCHG BX, AX ; SAVE ROW/COLUMN POSITION IN (BX)
600 MOV AL, BYTE PTR #CRT_COLS ; GET COLUMNS PER ROW COUNT
601 MUL BH ; DETERMINE BYTES TO ROW
602 XOR BH, BH ; DETERMINE BYTES TO ROW
603 ADD AX, BX ; ADD IN COLUMN VALUE
604 SAL AX, 1 ; * 2 FOR ATTRIBUTE BYTES
605 POP BP
606 RET
607 ENDP
608 ;-----
609 ; SCROLL UP
610 ; THIS ROUTINE MOVES A BLOCK OF CHARACTERS UP
611 ; ON THE SCREEN
612 ; INPUT
613 ; (AH) = CURRENT CRT MODE
614 ; (AL) = NUMBER OF ROWS TO SCROLL
615 ; (CX) = ROW/COLUMN OF UPPER LEFT CORNER
616 ; (DX) = ROW/COLUMN OF LOWER RIGHT CORNER
617 ; (BH) = ATTRIBUTE TO BE USED ON BLANKED LINE
618 ; (DS) = DATA SEGMENT
619 ; (ES) = REGEN BUFFER SEGMENT
620 ; OUTPUT
621 ; NONE -- THE REGEN BUFFER IS MODIFIED
622 ;-----
623 ASSUME DS:DATA, ES:DATA
624 SCROLL_UP PROC NEAR
625 CALL TEST_LINE_COUNT
626 CMP AH, 4 ; TEST FOR GRAPHICS MODE
627 JC N1 ; HANDLE SEPARATELY
628 MOV SI, 7 ; TEST FOR BW CARD
629 CMP SI, 7
630 JE N1
631 JMP GRAPHICS_UP
632 N1:
633 PUSH BX ; UP CONTINUE
634 MOV AX, CX ; SAVE FILL ATTRIBUTE IN BH
635 CALL SCROLL_POSITION ; UPPER LEFT POSITION
636 JZ N7 ; DO SETUP FOR SCROLL
637 ADD SI, AX ; BLANK FIELD
638 MOV AH, DH ; FROM ADDRESS
639 SUB AH, BL ; # ROWS IN BLOCK
640 N2: ; # ROWS TO BE MOVED
641 CALL N10 ; ROW LOOP
642 ADD SI, BP ; MOVE ONE ROW
643 ADD DI, BP ; POINT TO NEXT LINE IN BLOCK
644 DEC AH ; COUNT OF LINES TO MOVE
645 JNZ N2 ; ROW LOOP
646 N3: ; CLEAR ENTRY
647 POP AX ; RECOVER ATTRIBUTE IN AH
648 MOV AL, ' ' ; FILL WITH BLANKS
649 N4: ; CLEAR_LOOP
650 CALL N11 ; CLEAR THE ROW
651 ADD DI, BP ; POINT TO NEXT LINE
652 DEC BL ; COUNTER OF LINES TO SCROLL
653 JNZ N4 ; CLEAR_LOOP
654 N5: ; SCROLL_END
655 CALL DDS
656 CMP #CRT_MODE, T ; IS THIS THE BLACK AND WHITE CARD
657 JE N6 ; IF SO, SKIP THE MODE RESET
658 MOV AL, #CRT_MODE_SET ; GET THE VALUE OF THE MODE SET
659 MOV DX, 03DBH ; ALWAYS SET COLOR CARD PORT
660 OUT DX, AL
661 N6: ; VIDEO_RET_HERE
662 JMP VIDEO_RETURN
663 N7: ; BLANK FIELD
664 MOV BL, DH ; GET ROW COUNT
665 JMP N3 ; GO CLEAR THAT AREA
666 SCROLL_UP ENDP
667 ;-----
668 ;----- HANDLE COMMON SCROLL SET UP HERE
669 ;-----
670 SCROLL_POSITION PROC NEAR
671 CALL POSITION ; CONVERT TO REGEN POINTER
672 ADD AX, #CRT_START ; OFFSET OF ACTIVE PAGE
673 MOV D1, AX ; TO ADDRESS FOR SCROLL
674 MOV SI, AX ; FROM ADDRESS FOR SCROLL
675 SUB DX, CX ; DX = #ROWS, #COLS IN BLOCK
676 INC DH ; INCREMENT FOR 0 ORIGIN
677 INC DL ; SET HIGH BYTE OF COUNT TO ZERO
678 XOR CH, CH ; GET NUMBER OF COLUMNS IN DISPLAY
679 MOV BP, #CRT_COLS ; TIMES 2 FOR ATTRIBUTE BYTE
680 ADD BP, BP ; GET CHARACTERS PER LINE COUNT
681 MOV AL, BYTE PTR #CRT_COLS ; DETERMINE OFFSET TO FROM ADDRESS
682 MUL BL ; * 2 FOR ATTRIBUTE BYTE
683 ADD AX, AX ; SAVE LINE COUNT
684 PUSH AX

```

SECTION 5

```

688 0281 A0 0049 R      MOV     AL,6CRT_MODE      ; GET CURRENT MODE
689 0282 06             PUSH    ES                ; ESTABLISH ADDRESSING TO REGEN BUFFER
687 0286 1F             POP     DS                ; FOR BOTH POINTERS
688 0286 3C 02          CMP     AL,2              ; TEST FOR COLOR CARD SPECIAL CASES HERE
689 0286 72 13          JB     N9                 ; HAVE TO HANDLE 80X25 SEPARATELY
690 028A 3C 03          CMP     AL,3
691 028C 77 0F          JA     N9
692                                     ; 80X25 COLOR CARD SCROLL
693 028E 52             PUSH    DX                ; GUARANTEED TO BE COLOR CARD HERE
694 028F BA 03DA        MOV     DX,3DAH
695 0292                                     ; WAIT_DISP_ENABLE
N8: 0292 EC             IN     AL,DX              ; GET PORT
697 0293 A8 08          TEST   AL,RVRT           ; WAIT FOR VERTICAL RETRACE
698 0296 74 F8          JZ     N9                 ; WAIT_DISP_ENABLE
699 0297 80 25          MOV     AL,25H           ; ADDRESS CONTROL PORT
700 0299 82 D8          MOV     DL,0D8H          ; TURN OFF VIDEO DURING VERTICAL RETRACE
701 029B EE             OUT    DX,AL
702 029C 5A             POP     DX
703 029D                                     ; RESTORE LINE COUNT
704 029D 58             POP     AX
705 029E 0A D8          OR     BL,BL             ; 0 SCROLL MEANS BLANK FIELD
706 02A0 C3             RET                                ; RETURN WITH FLAGS SET
707 02A1                                     SCROLL_POSITION ENDP
708
709 ----- MOVE_ROW
710 02A1                                     PROC    NEAR
711 02A1 8A CA          MOV     CL,DL            ; GET # OF COLS TO MOVE
712 02A3 56             PUSH    SI
713 02A4 87             PUSH    DI
714 02A5 F3/ A5        REP     MOVSW            ; SAVE START ADDRESS
715 02A7 5F             POP     DI
716 02A8 5E             POP     SI
717 02A9 C3             RET                                ; RECOVER ADDRESSES
718 02AA                                     N10    ENDP
719 ----- CLEAR_ROW
720 02AA                                     PROC    NEAR
721 02AA 8A CA          MOV     CL,DL            ; GET # COLUMNS TO CLEAR
722 02AC 57             PUSH    DI
723 02AD F1/ AB        REP     STOSW            ; STORE THE FILL CHARACTER
724 02AF 5F             POP     DI
725 02B0 C3             RET                                ; STORE THE FILL CHARACTER
726 02B0 C3
727 02B1                                     N11    ENDP
728 -----
729 ; SCROLL_DOWN
730 ; THIS ROUTINE MOVES THE CHARACTERS WITHIN A DEFINED
731 ; BLOCK DOWN ON THE SCREEN, FILLING THE TOP LINES
732 ; WITH A DEFINED CHARACTER
733 ; INPUT
734 ; (AH) = CURRENT CRT MODE
735 ; (AL) = NUMBER OF LINES TO SCROLL
736 ; (CX) = UPPER LEFT CORNER OF REGION
737 ; (DX) = LOWER RIGHT CORNER OF REGION
738 ; (BH) = FILL CHARACTER
739 ; (DS) = DATA SEGMENT
740 ; (ES) = REGEN SEGMENT
741 ; OUTPUT
742 ; NONE -- SCREEN IS SCROLLED
743 -----
744 02B1 FD             STD                                SCROLL_DOWN PROC NEAR
745 02B2 E8 02EE R      CALL    TEST_LINE_COUNT    ; DIRECTION FOR SCROLL DOWN
746 02B5 80 FC 04       CMP     AH,4              ; TEST FOR GRAPHICS
747 02B8 72 08          JC     N12                ; TEST FOR BW CARD
748 02BA 80 FC 07       CMP     AH,7
749 02BD 74 D3          JE     N12                ; TEST FOR BW CARD
750 02C0 74 D3          JE     N12
751 02B8 E9 0507 R      JMP     GRAPHICS_DOWN
752 02C2                                     N12:
753 02C2 53             PUSH    BX                ; CONTINUE DOWN
754 02C3 8B C2          MOV     AX,DX             ; SAVE ATTRIBUTE IN BH
755 02C5 E8 0260 R      CALL    SCROLL_POSITION    ; LOWER RIGHT CORNER
756 02C8 74 20          JZ     N16                ; GET REGEN LOCATION
757 02CA 2B F0          SUB     SI,AX             ; SI IS FROM ADDRESS
758 02CC 8A E6          MOV     AH,DH             ; GET TOTAL # ROWS
759 02CE 2A E3          SUB     AH,BL             ; COUNT TO MOVE IN SCROLL
760 02D0                                     N13:
761 02D0 E8 02A1 R      CALL    N10                ; MOVE ONE ROW
762 02D3 2B F8          SUB     SI,8P
763 02D5 2B FD          SUB     DI,8P
764 02D7 FE CC          DEC     AH
765 02D9 75 F5          JNZ    N13
766 02DB                                     N14:
767 02DB 58             POP     AX                ; RECOVER ATTRIBUTE IN AH
768 02DC B0 20          MOV     AL,' '
769 02DE                                     N15:
770 02DE E8 02AA R      CALL    N11                ; CLEAR ONE ROW
771 02E1 2B FD          SUB     DI,8P             ; GO TO NEXT ROW
772 02E3 FE CB          DEC     BL
773 02E6 75 F7          JNZ    N15
774 02E7 E9 0248 R      JMP     N6                 ; SCROLL_END
775 02EA                                     N16:
776 02EA 8A DE          MOV     BL,DH
777 02EC EB ED          JMP     N14
778 02EE                                     SCROLL_DOWN ENDP
779 -----
780 ; IF AMOUNT OF LINES TO BE SCROLLED = AMOUNT OF LINES IN WINDOW
781 ; THEN ADJUST AL; ELSE RETURN;
782 -----
783 02EE                                     TEST_LINE_COUNT PROC NEAR
784 02EE 8A D8          MOV     BL,AL            ; SAVE LINE COUNT IN BL
785 02F0 0A C0          OR     AL,AL             ; TEST IF AL IS ALREADY ZERO
786 02F2 74 0E          JZ     BL_SET            ; IF IT IS THEN RETURN...
787 02F4 50             PUSH    AX                ; SAVE AX
788 02F5 8A C6          MOV     AL,DH            ; SUBTRACT LOWER ROW FROM UPPER ROW
789 02F7 8A C5          MOV     AL,CH
790 02F9 FE C0          INC     AL
791 02FB 3A C3          CMP     AL,BL            ; ADJUST DIFFERENCE BY 1
792 02FD 58             POP     AX                ; RESTORE AX
793 02FE 75 02          JNB    BL_SET            ; IF NOT THEN WE'RE ALL SET
794 0300 2A D8          SUB     BL,BL            ; OTHERWISE SET BL TO ZERO
795 0302                                     BL_SET:
796 0302 C3             RET                                ; RETURN
797 0303                                     TEST_LINE_COUNT ENDP
798 0303

```

```

799 PAGE
800 ; READ_AC_CURRENT
801 ; THIS ROUTINE READS THE ATTRIBUTE AND CHARACTER AT THE CURRENT
802 ; CURSOR POSITION AND RETURNS THEM TO THE CALLER
803 ;
804 ; INPUT
805 ; (AH) = CURRENT CRT MODE
806 ; (BH) = DISPLAY PAGE ( ALPHA MODES ONLY )
807 ; (DS) = DATA SEGMENT
808 ; (ES) = REGEN SEGMENT
809 ;
810 ; OUTPUT
811 ; (AL) = CHARACTER READ
812 ; (AH) = ATTRIBUTE READ
813 -----
814 ASSUME DS:DATA,ES:DATA
815
815 0303 READ_AC_CURRENT PROC NEAR
816 0303 80 FC 04 CMP AH,4 ; IS THIS GRAPHICS
817 0306 72 08 JC P10
818
819 0308 80 FC 07 CMP AH,7 ; IS THIS BW CARD
820 0308 74 03 JE P10
821
822 030D E9 0442 R JMP GRAPHICS_READ
823 0310 P10: CALL FIND_POSITION ; READ AC CONTINUE
824 0310 E8 032C R MOV SI,DI ; GET REGEN LOCATION AND PORT ADDRESS
825 0313 8B F7 PUSH ES ; ESTABLISH ADDRESSING IN SI
826 0315 06 ES ; GET REGEN SEGMENT FOR QUICK ACCESS
827 0316 1F POP DS
828
829 ;----- WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
830
831 0317 0A DB OR BL,BL ; CHECK MODE FLAG FOR COLOR CARD IN 80
832 0319 75 0D JNZ P11 ; ELSE SKIP RETRACE WAIT - DO FAST READ
833 031B P11: STI ; WAIT FOR HDRZ RETRACE LOW OR VERTICAL
834 031B FB ; ENABLE INTERRUPTS FIRST
835 031C 90 NOP ; ALLOW FOR SMALL INTERRUPT WINDOW
836 031D FA CLI ; BLOCK INTERRUPTS FOR SINGLE LOOP
837 031E EC IN AL,DX ; GET STATUS FROM THE ADAPTER
838 031F AB 01 TEST AL,RHRZ ; IS HORIZONTAL RETRACE LOW
839 0321 75 F8 JNZ P11 ; WAIT UNTIL IT IS
840 0323 P12: IN AL,DX ; NOW WAIT FOR EITHER RETRACE HIGH
841 0323 EC ; GET STATUS
842 0324 AB 09 TEST AL,RVRT+RHRZ ; IS HORIZONTAL OR VERTICAL RETRACE HIGH
843 0326 74 FB JZ P12 ; WAIT UNTIL EITHER IS ACTIVE
844 0328 P13: LODSW ; GET THE CHARACTER AND ATTRIBUTE
845 0328 AD JMP VIDEO_RETURN ; EXIT WITH (AX)
846 0329 E9 013D R
847
848 032C READ_AC_CURRENT ENDP
849
850
851
852 032C FIND_POSITION PROC NEAR
853 032C 86 E3 XCHG AH,BL ; SETUP FOR BUFFER READ OR WRITE
854 032E 8B E8 MOV BP,AX ; SWAP MODE TYPE WITH ATTRIBUTE
855 0330 80 EB 02 SUB BL,2 ; SAVE CHARACTER/ATTR IN (BP) REGISTER
856 0333 0D EB ; CONVERT DISPLAY MODE TYPE TO A
857 0335 BA C7 SHR BL,1 ; ZERO VALUE FOR COLOR IN 80 COLUMN
858 0337 98 MOV AL,BH ; MOVE DISPLAY PAGE TO LOW BYTE
859 0338 BB F8 CBW ; CLEAR HIGH BYTE FOR BYTE OFFSET
860 033A D1 E7 MOV DI,AX ; MOVE DISPLAY PAGE (COUNT) TO WORK REG
861 033C 8B 95 0050 R MOV DX,[DI+OFFSET #CURSOR_POSN] ; TIMES 2 FOR WORD OFFSET
862 0340 74 09 JZ P21 ; GET ROW/COLUMN OF THAT PAGE
863 ; SKIP BUFFER ADJUSTMENT IF PAGE ZERO
864 0342 33 FF XOR DI,DI ; ELSE SET BUFFER START ADDRESS TO ZERO
865 0344 P20: ADD DI,#CRT_LEN ; ADD LENGTH OF BUFFER FOR ONE PAGE
866 0344 03 3E 004C R DEC AX ; DECREMENT PAGE COUNT
867 0348 48 JNZ P20 ; LOOP TILL PAGE COUNT EXHAUSTED
868 0349 75 F9
869
870 034B P21: MOV AL,BYTE PTR #CRT_COLS ; DETERMINE LOCATION IN REGEN IN PAGE
871 034B A0 004A R ; GET COLUMNS PER ROW COUNT
872 034E 76 E6 MUL DH ; DETERMINE BYTES TO ROW
873 0350 32 F6 XOR DH,DH
874 0352 03 C2 ADD AX,DX ; ADD IN COLUMN VALUE
875 0354 D1 E0 SAL AX,1 ; * 2 FOR ATTRIBUTE BYTES
876 0356 03 F8 ADD DI,AX ; ADD LOCATION TO START OF REGEN PAGE
877 0358 BB 16 0063 R MOV DX,#ADDR_6845 ; GET BASE ADDRESS OF ACTIVE DISPLAY
878 035C 83 C2 06 ADD DX,6 ; DX= STATUS PORT ADDRESS OF ADAPTER
879 035F C3 RET ; BP= ATTRIBUTE/CHARACTER (FROM BL/AL)
880 ; DI= POSITION (OFFSET IN REGEN BUFFER)
881 0360 FIND_POSITION ENDP ; BL= MODE FLAG (ZERO FOR 80X25 COLOR)
    
```

```

882 PAGE
883 -----
884 ; WRITE_AC_CURRENT
885 ; THIS ROUTINE WRITES THE ATTRIBUTE AND CHARACTER
886 ; AT THE CURRENT CURSOR POSITION
887 ;
888 ; INPUT
889 ; (AH) = CURRENT CRT MODE
890 ; (BH) = DISPLAY PAGE
891 ; (CX) = COUNT OF CHARACTERS TO WRITE
892 ; (AL) = CHAR TO WRITE
893 ; (BL) = ATTRIBUTE OF CHAR TO WRITE
894 ; (DS) = DATA SEGMENT
895 ; (ES) = REGEN SEGMENT
896 ; OUTPUT
897 ; DISPLAY REGEN BUFFER UPDATED
898 -----
899
900 WRITE_AC_CURRENT PROC NEAR
901     CMP AH,4 ; IS THIS GRAPHICS
902     JC P30
903     CMP AH,7 ; IS THIS BW CARD
904     JE P30
905     JMP GRAPHICS_WRITE
906
907 P30: CALL FIND_POSITION ; WRITE AC CONTINUE
908     ; GET REGEN LOCATION AND PORT ADDRESS
909     ; ADDRESS IN (DI) REGISTER
910     OR BL,BL ; CHECK MODE FLAG FOR COLOR CARD AT 80
911     JZ P32 ; SKIP TO RETRACE WAIT IF COLOR AT 80
912     XCHG AX,BP ; GET THE ATTR/CHAR SAVED FOR FAST WRITE
913     REP STOSW ; STRING WRITE THE ATTRIBUTE & CHARACTER
914     JMP SHORT P35 ; EXIT FAST WRITE ROUTINE
915
916 ;----- WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
917
918 P31: XCHG BP,AX ; LOOP FOR EACH ATTR/CHAR WRITE
919     ; PLACE ATTR/CHAR BACK IN SAVE REGISTER
920     ; WAIT FOR HORZ RETRACE LOW OR VERTICAL
921     STI ; ENABLE INTERRUPTS FIRST
922     NOP ; ALLOW FOR INTERRUPT WINDOW
923     CLI ; BLOCK INTERRUPTS FOR SINGLE LOOP
924     IN AL,DX ; GET STATUS FROM THE ADAPTER
925     TEST AL,RVRT ; CHECK FOR VERTICAL RETRACE FIRST
926     JNZ P34 ; DO FAST WRITE NOW IF VERTICAL RETRACE
927     TEST AL,RHRZ ; IS HORIZONTAL RETRACE LOW THEN
928     JNZ P33 ; WAIT UNTIL IT IS
929     IN AL,DX ; GET STATUS AGAIN
930     TEST AL,RVRT+RHRZ ; IS HORIZONTAL OR VERTICAL RETRACE HIGH
931     JZ P34 ; WAIT UNTIL EITHER IS ACTIVE
932
933 P34: XCHG AX,BP ; GET THE ATTR/CHAR SAVED IN (BP)
934     STOSW ; WRITE THE ATTRIBUTE AND CHARACTER
935     LOOP P31 ; AS MANY TIMES AS REQUESTED -- TILL CX=0
936
937 P35: JMP VIDEO_RETURN ; EXIT
938
939 WRITE_AC_CURRENT ENDP
940
941 ;-----
942 ; WRITE_C_CURRENT
943 ; THIS ROUTINE WRITES THE CHARACTER AT
944 ; THE CURRENT CURSOR POSITION, ATTRIBUTE UNCHANGED
945 ;
946 ; INPUT
947 ; (AH) = CURRENT CRT MODE
948 ; (BH) = DISPLAY PAGE
949 ; (CX) = COUNT OF CHARACTERS TO WRITE
950 ; (AL) = CHAR TO WRITE
951 ; (DS) = DATA SEGMENT
952 ; (ES) = REGEN SEGMENT
953 ; OUTPUT
954 ; DISPLAY REGEN BUFFER UPDATED
955 -----
956
957 WRITE_C_CURRENT PROC NEAR
958     CMP AH,4 ; IS THIS GRAPHICS
959     JC P40
960     CMP AH,7 ; IS THIS BW CARD
961     JE P40
962     JMP GRAPHICS_WRITE
963
964 P40: CALL FIND_POSITION ; GET REGEN LOCATION AND PORT ADDRESS
965     ; ADDRESS OF LOCATION IN (DI)
966
967 ;----- WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
968
969 P41: STI ; WAIT FOR HORZ RETRACE LOW OR VERTICAL
970     ; ENABLE INTERRUPTS FIRST
971     OR BL,BL ; CHECK MODE FLAG FOR COLOR CARD IN 80
972     JNZ P43 ; ELSE SKIP RETRACE WAIT -- DO FAST WRITE
973     CLI ; BLOCK INTERRUPTS FOR SINGLE LOOP
974     IN AL,DX ; GET STATUS FROM THE ADAPTER
975     TEST AL,RVRT ; CHECK FOR VERTICAL RETRACE FIRST
976     JNZ P42 ; DO FAST WRITE NOW IF VERTICAL RETRACE
977     TEST AL,RHRZ ; IS HORIZONTAL RETRACE LOW THEN
978     JNZ P41 ; WAIT UNTIL IT IS
979     IN AL,DX ; GET STATUS AGAIN
980     TEST AL,RVRT+RHRZ ; IS HORIZONTAL OR VERTICAL RETRACE HIGH
981     JZ P42 ; WAIT UNTIL EITHER RETRACE ACTIVE
982
983 P42: MOV AX,BP ; GET THE CHARACTER SAVED IN (BP)
984     STOSB ; PUT THE CHARACTER INTO REGEN BUFFER
985     INC DI ; BUMP POINTER PAST ATTRIBUTE
986     LOOP P41 ; AS MANY TIMES AS REQUESTED
987
988 P43: JMP VIDEO_RETURN
989
990 WRITE_C_CURRENT ENDP
    
```

```

991 PAGE
992 -----
993 ; WRITE_STRING
994 ; THIS ROUTINE WRITES A STRING OF CHARACTERS TO THE CRT.
995 ; INPUT
996 ; (AL) = WRITE STRING COMMAND 0 - 3
997 ; (BH) = DISPLAY PAGE (ACTIVE PAGE)
998 ; (CX) = COUNT OF CHARACTERS TO WRITE, IF (CX) = 0 THEN RETURN
999 ; (DX) = CURSOR POSITION FOR START OF STRING WRITE
1000 ; (BL) = ATTRIBUTE OF CHARACTER TO WRITE IF (AL) = 0 OR (AL) = 1
1001 ; (BP) = SOURCE STRING OFFSET
1002 ; [DE] = SOURCE STRING SEGMENT (FOR USE IN (ES) IN STACK +14)
1003 ; OUTPUT
1004 ; NONE
1005 -----
1006 03BF PROC NEAR
1007 03BF 55 PUSH BP ; SAVE BUFFER OFFSET (BP) IN STACK
1008 03C0 8B EC MOV BP,SP ; GET POINTER TO STACKED REGISTERS
1009 03C2 8E 46 10 MOV ES,[BP]+14+2 ; RECOVER ENTRY (ES) SEGMENT REGISTER
1010 03C5 5D POP BP ; RESTORE BUFFER OFFSET
1011 03C6 98 CBW ; CLEAR (AH) REGISTER
1012 03C7 8B F8 MOV DI,AX ; SAVE (AL) COMMAND IN (DI) REGISTER
1013 03C9 3C 04 CMP AL,04 ; TEST FOR INVALID WRITE STRING OPTION
1014 03CB 73 73 JNB P59 ; IF OPTION INVALID THEN RETURN
1015
1016 03CD E3 71 JCXZ P59 ; IF ZERO LENGTH STRING THEN RETURN
1017
1018 03CF 8B F3 MOV SI,BX ; SAVE CURRENT CURSOR PAGE
1019 03D1 6A DF MOV BL,BH ; MOVE PAGE TO LOW BYTE
1020 03D3 32 FF XOR BH,BH ; CLEAR HIGH BYTE
1021 03D5 87 F3 XCHG SI,BX ; MOVE OFFSET AND RESTORE PAGE REGISTER
1022 03D7 D1 E6 SAL SI,1 ; CONVERT TO PAGE OFFSET (SI= PAGE)
1023 03D9 FF B4 0050 R PUSH [SI+OFFSET *CURSOR_POSN] ; SAVE CURRENT CURSOR POSITION IN STACK
1024 03DD 8B 0200 MOV AX,0200H ; SET NEW CURSOR POSITION
1025 03E0 CD 10 INT 10H
1026 03E2
1027 03E2 26 8A 46 00 MOV AL,ES:[BP] ; GET CHARACTER FROM INPUT STRING
1028 03E6 45 INC BP ; BUMP POINTER TO CHARACTER
1029
1030 ;----- TEST FOR SPECIAL CHARACTER'S
1031
1032 03E7 3C 08 CMP AL,08H ; IS IT A BACKSPACE
1033 03E9 74 0C JE P51 ; BACK_SPACE
1034 03EB 3C 0D CMP AL,09H ; IS IT CARRIAGE RETURN
1035 03ED 74 08 JE P51 ; CAR_RET
1036 03EF 3C 0A CMP AL,0FH ; IS IT A LINE FEED
1037 03F1 74 04 JE P51 ; LINE_FEED
1038 03F3 3C 07 CMP AL,07H ; IS IT A BELL
1039 03F5 75 0A JNE P52 ; IF NOT THEN DO WRITE CHARACTER
1040 03F7
1041 03F7 B4 0E MOV AH,0EH ; TTY_CHARACTER WRITE
1042 03F9 CD 10 INT 10H ; WRITE TTY CHARACTER TO THE CRT
1043 03FB 8B 94 0050 R MOV DX,[SI+OFFSET *CURSOR_POSN] ; GET CURRENT CURSOR POSITION
1044 03FF EB 2D JMP SHORT P54 ; SET CURSOR POSITION AND CONTINUE
1045
1046 0401
1047 0401 51 PUSH CX
1048 0402 53 PUSH BX
1049 0403 B9 0001 MOV CX,1 ; SET CHARACTER WRITE AMOUNT TO ONE
1050 0406 83 FF 02 CMP DI,2 ; IS THE ATTRIBUTE IN THE STRING
1051 0409 72 05 JB P53 ; IF NOT THEN SKIP
1052 040B 26 8A 9E 00 MOV BL,ES:[BP] ; ELSE GET NEW ATTRIBUTE
1053 040F 45 INC BP ; BUMP STRING POINTER
1054 0410
1055 0410 B4 09 MOV AH,09H ; GOT_CHARACTER
1056 0412 CD 10 INT 10H ; WRITE CHARACTER TO THE CRT
1057 0414 5B POP BX ; RESTORE REGISTERS
1058 0415 59 POP CX
1059 0416 FE C2 INC DL ; INCREMENT COLUMN COUNTER
1060 0418 3A 16 004A R CMP DL,BYTE PTR *CRT_COLS ; IF COLS ARE WITHIN RANGE FOR THIS MODE
1061 041C T2 10 JB P54 ; THEN GO TO COLUMNS SET
1062 041E FE C6 INC DH ; BUMP ROW COUNTER BY ONE
1063 0420 2A D2 SUB DL,DL ; SET COLUMN COUNTER TO ZERO
1064 0422 80 FE 19 CMP DH,25 ; IF ROWS ARE LESS THAN 25 THEN
1065 0425 72 07 JB P54 ; GO TO ROWS_COLUMNS_SET
1066
1067 0427 BB 0E0A MOV AX,0E0AH ; ELSE SCROLL SCREEN ONE LINE
1068 042A CD 10 INT 10H ; RESET ROW COUNTER TO 24
1069 042C FE CE DEC DH
1070 042E
1071 042E B8 0200 MOV AX,0200H ; ROW_COLUMNS_SET
1072 0431 CD 10 INT 10H ; SET NEW CURSOR POSITION COMMAND
1073 0433 E2 AD LOOP P50 ; ESTABLISH NEW CURSOR POSITION
1074 ; DO IT ONCE MORE UNTIL (CX) = ZERO
1075 0435 5A POP DX ; RESTORE OLD CURSOR COORDINATES
1076 0436 97 XCHG AX,DI ; RECOVER WRITE STRING COMMAND
1077 0437 A8 01 TEST AL,01H ; IF CURSOR WAS NOT TO BE MOVED THEN
1078 0439 76 05 JNZ P59 ; THEN EXIT WITHOUT RESETTING OLD VALUE
1079 043B 8B 0200 MOV AX,0200H ; ELSE RESTORE OLD CURSOR POSITION
1080 043E CD 10 INT 10H
1081 0440
1082 0440 E9 013D R JMP VIDEO_RETURN ; DONE - EXIT WRITE STRING
1083 ; RETURN TO CALLER
1084 0443 WRITE_STRING ENDP
    
```

```

1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103 0443
1104 0443 E8 0477 R
1105 0444 26 8A 04
1106 0449 22 C4
1107 044B D2 E0
1108 044D 8A CE
1109 044F D2 C0
1110 0451 E9 013D R
1111 0454
1112
1113 0454
1114 0454 50
1115 0455 50
1116 0456 E8 0477 R
1117 0459 D2 E8
1118 045B 22 C4
1119 045D 26 8A 0C
1120 0460 5B
1121 0461 F6 C3 80
1122 0464 75 0D
1123 0466 F6 D4
1124 0468 22 CC
1125 046A 0A C1
1126 046C
1127 046C 26 88 04
1128 046F 58
1129 0470 E9 013D R
1130 0473
1131 0473 32 C1
1132 0475 EB F5
1133 0477
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147 0477
1148
1149
1150
1151
1152 0477 96
1153 0478 B0 28
1154 047A F6 E2
1155 047C A8 08
1156 047E 74 03
1157 0480 05 1FD8
1158 0483
1159 0483 96
1160 0484 8B D1
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170 0486 B8 02C0
1171 0489 B9 0302
1172 048C 8D 3E 0049 R 06
1173 0491 72 06
1174 0493 B8 0180
1175 0496 B9 0703
1176
1177
1178 0499
1179 0499 22 EA
1180
1181
1182
1183 049B D3 EA
1184 049D 03 F2
1185 049F 8A F7
1186
1187
1188
1189 04A1 2A C9
1190 04A3
1191 04A3 D0 C8
1192 04A5 02 CD
1193 04A7 FE CF
1194 04A9 75 F8
1195 04AB 8A E3
1196 04AD D2 EC
1197 04AF C3
1198 04B0
    
```

```

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)
: 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
READ_DOT PROC NEAR
    CALL R3
    MOV AL,ES:[SI]
    AND AL,AH
    SHL AL,CL
    MOV CL,DH
    ROL AL,CL
    JMP VIDEO_RETURN
: DETERMINE BYTE POSITION OF DOT
: GET THE BYTE
: MASK OFF THE OTHER BITS IN THE BYTE
: LEFT JUSTIFY THE VALUE
: GET NUMBER OF BITS IN RESULT
: RIGHT JUSTIFY THE RESULT
: RETURN FROM VIDEO I/O
ENDP
WRITE_DOT PROC NEAR
    PUSH AX
    CALL R3
    SHR AL,CL
    AND AL,AH
    MOV CL,ES:[SI]
    POP BX
    TEST BL,80H
    JNZ R2
    NOT AH
    AND CL,AH
    OR AL,CL
: OR IN THE NEW VALUE OF THOSE BITS
: FINISH DOT
: RESTORE THE BYTE IN MEMORY
R1: MOV ES:[SI],AL
    POP AX
    JMP VIDEO_RETURN
: RETURN FROM VIDEO I/O
: XOR DOT
: EXCLUSIVE OR THE DOTS
: FINISH UP THE WRITING
R2: XOR AL,CL
    MOV R1,CL
    JMP ENDP
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
-----
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 )
    XCHG SI,AX
    MOV AL,40
    MUL DL
    TEST AL,008H
    JZ R4
    ADD AX,2000H-40
R4: XCHG SI,AX
    MOV DX,CX
:----- 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/COH FOR H/M )
: BH = NUMBER OF VALID BITS IN POINTED BYTE ( 1/2 FOR H/M )
    MOV BX,2C0H
    MOV CX,302H
    CMP @CRT_MODE,6
    JC R5
    MOV BX,180H
    MOV CX,703H
: SET PARMS FOR MED RES
: HANDLE IF MED RES
: SET PARMS FOR HIGH RES
:----- DETERMINE BIT OFFSET IN BYTE FROM COLUMN MASK
R5: AND CH,DL
: ADDRESS OF PEL WITHIN BYTE TO CH
:----- DETERMINE BYTE OFFSET FOR THIS LOCATION IN COLUMN
    SHR DX,CL
    ADD SI,DX
    MOV DH,BH
: SHIFT BY CORRECT AMOUNT
: INCREMENT THE POINTER
: GET THE # OF BITS IN RESULT TO DH
:----- MULTIPLY BH (VALID BITS IN BYTE) BY CH (BIT OFFSET)
    SUB CH,CL
: ZERO INTO STORAGE LOCATION
R6: ROR AL,1
    ADD CH,CH
    DEC BH
    JNZ R6
    MOV AH,BL
    SHR AH,CL
    RET
: ON EXIT, CL HAS COUNT TO RESTORE BITS
: GET MASK TO AH
: MOVE THE MASK TO CORRECT LOCATION
: RETURN WITH EVERYTHING SET UP
R3 ENDP
    
```

```

1199 ;-----
1200 ; SCROLL UP
1201 ; THIS ROUTINE SCROLLS UP THE INFORMATION ON THE CRT
1202 ; ENTRY ---
1203 ; CH,CL = UPPER LEFT CORNER OF REGION TO SCROLL
1204 ; DH,DL = LOWER RIGHT CORNER OF REGION TO SCROLL
1205 ; BOTH OF THE ABOVE ARE IN CHARACTER POSITIONS
1206 ; BH = FILL VALUE FOR BLANKED LINES
1207 ; AL = # LINES TO SCROLL (AL=0 MEANS BLANK THE ENTIRE FIELD)
1208 ; DS = DATA SEGMENT
1209 ; ES = REGEN SEGMENT
1210 ; EXIT ---
1211 ; NOTHING, THE SCREEN IS SCROLLED
1212 ;-----
1213 04B0 GRAPHICS UP PROC NEAR
1214 04B0 MOV BL,AL ; SAVE LINE COUNT IN BL
1215 04B2 8B C1 MOV AX,CX ; GET UPPER LEFT POSITION INTO AX REG
1216
1217 ;----- USE CHARACTER SUBROUTINE FOR POSITIONING
1218 ;----- ADDRESS RETURNED IS MULTIPLIED BY 2 FROM CORRECT VALUE
1219
1220 04B4 E8 06F0 R CALL GRAPH_POSN ; SAVE RESULT AS DESTINATION ADDRESS
1221 04B7 8B F8 MOV DI,AX
1222
1223 ;----- DETERMINE SIZE OF WINDOW
1224
1225 04B9 2B D1 SUB DX,CX
1226 04BB 81 C2 0101 ADD DX,101H ; ADJUST VALUES
1227 04BD D0 E6 SAL DH,1 ; MULTIPLY ROWS BY 4 AT 8 VERT DOTS/CHAR
1228 04C1 D0 E6 SAL DH,1 ; AND EVEN/ODD ROWS
1229
1230 ;----- DETERMINE CRT MODE
1231
1232 04C3 80 3E 0049 R 06 CMP @CRT_MODE,6 ; TEST FOR MEDIUM RES
1233 04C5 73 04 JNC ; FIND_SOURCE
1234
1235 ;----- MEDIUM RES UP
1236 04CA D0 E2 SAL DH,1 ; # COLUMNS * 2, SINCE 2 BYTES/CHAR
1237 04CC D1 E7 SAL DI,1 ; OFFSET *2 SINCE 2 BYTES/CHAR
1238
1239 ;----- DETERMINE THE SOURCE ADDRESS IN THE BUFFER
1240 04CE RTI ; FIND SOURCE
1241 04CE 06 PUSH ES ; GET SEGMENTS BOTH POINTING TO REGEN
1242 04CF 1F POP DS
1243 04D0 2A ED SUB CH,CH
1244 04D2 D0 E3 SAL BL,1 ; ZERO TO HIGH OF COUNT REGISTER
1245 04D4 D0 E3 SAL BL,1 ; MULTIPLY NUMBER OF LINES BY 4
1246 04D6 74 2B JZ R11 ; IF ZERO, THEN BLANK ENTIRE FIELD
1247 04D8 80 50 MOV AL,B0 ; 80 BYTES/ROW
1248 04DA F6 E3 MUL BL ; DETERMINE OFFSET TO SOURCE
1249 04DC 8B F7 MOV SI,DI ; SET UP SOURCE
1250 04DE 03 F0 ADD SI,AX ; ADD IN OFFSET TO IT
1251 04E0 5A E4 MOV AH,DH ; NUMBER OF ROWS IN FIELD
1252 04E2 2A E3 SUB AH,BL ; DETERMINE NUMBER TO MOVE
1253
1254 ;----- LOOP THROUGH, MOVING ONE ROW AT A TIME, BOTH EVEN AND ODD FIELDS
1255 04E4 R8: CALL R17 ; ROW LOOP
1256 04E4 E8 0564 R ; MOVE ONE ROW
1257 04E7 81 EF 1FB0 SUB SI,2000H-80 ; MOVE TO NEXT ROW
1258 04EB 81 EF 1FB0 SUB DI,2000H-80
1259 04EF FE CC DEC AH ; NUMBER OF ROWS TO MOVE
1260 04F1 75 F1 JNZ R8 ; CONTINUE TILL ALL MOVED
1261
1262 ;----- FILL IN THE VACATED LINE(S)
1263 04F3 R9: ; CLEAR ENTRY
1264 04F3 8A C7 MOV AL,BH ; ATTRIBUTE TO FILL WITH
1265 04F5 R10:
1266 04F5 E8 057D R CALL R18 ; CLEAR THAT ROW
1267 04F8 81 EF 1FB0 SUB DI,2000H-80 ; POINT TO NEXT LINE
1268 04FC FE CB DEC BL ; NUMBER OF LINES TO FILL
1269 04FE 75 F5 JNZ R10 ; CLEAR LOOP
1270 0500 E9 013D R JMP VIDEO_RETURN ; EVERYTHING DONE
1271
1272 0503 R11: ; BLANK FIELD
1273 0503 8A DE MOV BL,DH ; SET BLANK COUNT TO EVERYTHING IN FIELD
1274 0505 EB EC JMP R9 ; CLEAR THE FIELD
1275 0507 GRAPHICS_UP ENDP
1276 ;-----
1277 ; SCROLL DOWN
1278 ; THIS ROUTINE SCROLLS DOWN THE INFORMATION ON THE CRT
1279 ; ENTRY ---
1280 ; CH,CL = UPPER LEFT CORNER OF REGION TO SCROLL
1281 ; DH,DL = LOWER RIGHT CORNER OF REGION TO SCROLL
1282 ; BOTH OF THE ABOVE ARE IN CHARACTER POSITIONS
1283 ; BH = FILL VALUE FOR BLANKED LINES
1284 ; AL = # LINES TO SCROLL (AL=0 MEANS BLANK THE ENTIRE FIELD)
1285 ; DS = DATA SEGMENT
1286 ; ES = REGEN SEGMENT
1287 ; EXIT ---
1288 ; NOTHING, THE SCREEN IS SCROLLED
1289 ;-----
1290 0507 GRAPHICS DOWN PROC NEAR
1291 0507 STD ; SET DIRECTION
1292 0507 FD MOV BL,AL ; SAVE LINE COUNT IN BL
1293 0508 8A D8 MOV AX,DX ; GET LOWER RIGHT POSITION INTO AX REG
1294 050A 8B C2 MOV AX,CX
1295
1296 ;----- USE CHARACTER SUBROUTINE FOR POSITIONING
1297 ;----- ADDRESS RETURNED IS MULTIPLIED BY 2 FROM CORRECT VALUE
1298
1299 050C E8 06F0 R CALL GRAPH_POSN ; SAVE RESULT AS DESTINATION ADDRESS
1300 050F 8B F8 MOV DI,AX
1301
1302 ;----- DETERMINE SIZE OF WINDOW
1303
1304 0511 2B D1 SUB DX,CX
1305 0513 81 C2 0101 ADD DX,101H ; ADJUST VALUES
1306 0517 D0 E6 SAL DH,1 ; MULTIPLY ROWS BY 4 AT 8 VERT DOTS/CHAR
1307 0519 D0 E6 SAL DH,1 ; AND EVEN/ODD ROWS
1308
1309 ;----- DETERMINE CRT MODE
1310
1311 051B 80 3E 0049 R 06 CMP @CRT_MODE,6 ; TEST FOR MEDIUM RES
1312 0520 73 05 JNC R12 ; FIND_SOURCE_DOWN
    
```

SECTION 5

```

1313
1314
1315 0522 00 E2          ;----- MEDIUM RES DOWN
1316 0524 01 E7          SAL    DL,1          ; # COLUMNS * 2, SINCE 2 BYTES/CHAR
1317 0526 47            SAL    DI,1          ; OFFSET *2 SINCE 2 BYTES/CHAR
1318                                INC    DI            ; POINT TO LAST BYTE
1319
1320
1321 0527          ;----- DETERMINE THE SOURCE ADDRESS IN THE BUFFER
1322 0527 06          R12:   PUSH   ES          ; FIND_SOURCE_DOWN
1323 0528 1F          POP    DS            ; BOTH SEGMENTS TO REGEN
1324 0529 2A ED       SUB    CH,CH         ; ZERO TO HIGH OF COUNT REGISTER
1325 052B 81 C7 00F0  ADD    DI,240       ; POINT TO LAST ROW OF PIXELS
1326 052F 00 E3       SAL    BL,1         ; MULTIPLY NUMBER OF LINES BY 4
1327 0531 00 E3       SAL    BL,1
1328 0533 74 2B       JZ     R16          ; IF ZERO, THEN BLANK ENTIRE FIELD
1329 0535 80 50       MOV    AL,80        ; 80 BYTES/ROW
1330 0537 F6 E3       MVL   BL           ; DETERMINE OFFSET TO SOURCE
1331 0539 8B F7       MOV    SI,DI        ; SET UP SOURCE
1332 053B 2B F0       SUB    SI,AX        ; SUBTRACT THE OFFSET
1333 053D 8A E6       MOV    AH,DH       ; NUMBER OF ROWS IN FIELD
1334 053F 2A E3       SUB    AH,BL        ; DETERMINE NUMBER TO MOVE
1335
1336
1337
1338 0541          ;----- LOOP THROUGH, MOVING ONE ROW AT A TIME, BOTH EVEN AND ODD FIELDS
1339 0541 E8 0564 R     R13:   CALL   R17          ; ROW LOOP DOWN
1340 0544 81 EE 2050   SUB    SI,2000H+80 ; MOVE ONE ROW
1341 0548 81 EF 2050   SUB    DI,2000H+80 ; MOVE TO NEXT ROW
1342 054C FE CC       DEC    AH           ; NUMBER OF ROWS TO MOVE
1343 054E 75 F1       JNZ   R13          ; CONTINUE TILL ALL MOVED
1344
1345
1346 0550          ;----- FILL IN THE VACATED LINE(S)
1347 0550 8A C7       R14:   MOV    AL,BH        ; CLEAR_ENTRY_DOWN
1348 0552                                ; ATTRIBUTE TO FILL WITH
1349 0552 E8 057D R     R15:   CALL   R18          ; CLEAR_LOOP_DOWN
1350 0555 81 EF 2050   SUB    DI,2000H+80 ; CLEAR_A_ROW
1351 0559 FE CB       DEC    BL           ; POINT TO NEXT LINE
1352 055B 75 F5       JNZ   R15          ; NUMBER OF LINES TO FILL
1353                                ; CLEAR_LOOP_DOWN
1354 055D E9 013D R     JMP    VIDEO_RETURN ; EVERYTHING DONE
1355
1356 0560          ;----- BLANK FIELD DOWN
1357 0560 8A DE       R16:   MOV    BL,DH        ; SET BLANK COUNT TO EVERYTHING IN FIELD
1358 0562 EB EC       JMP    R14         ; CLEAR THE FIELD
1359 0564
1360
1361
1362
1363 0564          ;----- ROUTINE TO MOVE ONE ROW OF INFORMATION
1364 0564 8A CA       R17:   PROC   NEAR        ;
1365 0566 B6           MOV    CL,DL        ; NUMBER OF BYTES IN THE ROW
1366 0567 57           PUSH  DI            ; SAVE POINTERS
1367 0568 F3/ A4       REP   MOVSB         ; MOVE THE EVEN FIELD
1368 056A 5F           POP   DI            ;
1369 056B 5E           POP   SI            ;
1370 056C 81 C6 2000  ADD    SI,2000H     ;
1371 0570 81 C7 2000  ADD    DI,2000H     ; POINT TO THE ODD FIELD
1372 0574 B6           PUSH  SI            ;
1373 0575 57           PUSH  DI            ; SAVE THE POINTERS
1374 0576 8A CA       MOV    CL,DL        ; COUNT BACK
1375 0578 F3/ A4       REP   MOVSB         ; MOVE THE ODD FIELD
1376 057A 5F           POP   DI            ;
1377 057B 5E           POP   SI            ; POINTERS BACK
1378 057C C3           RET                ; RETURN TO CALLER
1379 057D
1380
1381
1382
1383 057D          ;----- CLEAR A SINGLE ROW
1384 057D 8A CA       R18:   PROC   NEAR        ;
1385 057F 57           MOV    CL,DL        ; NUMBER OF BYTES IN FIELD
1386 0580 F3/ AA       PUSH  DI            ; SAVE POINTER
1387 0582 5F           REP   STOSB        ; STORE THE NEW VALUE
1388 0583 81 C7 2000  POP   DI            ; POINTER BACK
1389 0587 57           ADD    DI,2000H     ; POINT TO ODD FIELD
1390 0588 8A CA       PUSH  DI            ;
1391 058A F3/ AA       MOV    CL,DL        ; FILL THE ODD FIELD
1392 058C 5F           REP   STOSB        ;
1393 058D C3           POP   DI            ; RETURN TO CALLER
1394 058E
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
    ;-----
    ; 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
    ;-----
    
```



```

1427          ASSUME DS:DATA,ES:DATA
1428 058E      GRAPHICS WRITE PROC NEAR
1429 058E B4 00      MOV AH,0          ; ZERO TO HIGH OF CODE POINT
1430 0590 50      PUSH AX          ; SAVE CODE POINT VALUE
1431
1432          ;----- DETERMINE POSITION IN REGEN BUFFER TO PUT CODE POINTS
1433
1434 0591 E8 06ED R  CALL 526          ; FIND LOCATION IN REGEN BUFFER
1435 0594 8B F8      MOV DI,AX          ; REGEN POINTER IN DI
1436
1437          ;----- DETERMINE REGION TO GET CODE POINTS FROM
1438
1439 0596 58      POP AX          ; RECOVER CODE POINT
1440 0597 3C 80      CMP AL,80H        ; IS IT IN SECOND HALF
1441 0599 73 06      JAE S1          ; YES
1442
1443          ;----- IMAGE IS IN FIRST HALF, CONTAINED IN ROM
1444
1445 059B BE 0000 E  MOV SI,OFFSET CRT_CHAR_GEN ; OFFSET OF IMAGES
1446 059E 0E      PUSH CS          ; SAVE SEGMENT ON STACK
1447 059F EB 18      JMP SHORT S2     ; DETERMINE_MODE
1448
1449          ;----- IMAGE IS IN SECOND HALF, IN USER MEMORY
1450
1451 05A1          S1:          ; EXTEND CHAR
1452 05A1 2C 80      SUB AL,80H        ; ZERO ORIGIN FOR SECOND HALF
1453 05A3 1E      PUSH DS          ; SAVE DATA POINTER
1454 05A4 2B F6      SUB SI,S1         ; ESTABLISH VECTOR ADDRESSING
1455 05A6 8E DE      MOV DS,S1
1456          ASSUME DS:ABS0
1457 05A8 C5 36 007C R  LDS SI,0EXT_PTR  ; GET THE OFFSET OF THE TABLE
1458 05AC 8C DA      MOV DX,DS        ; GET THE SEGMENT OF THE TABLE
1459          ASSUME DS:DATA
1460 05AE 1F      POP DS          ; RECOVER DATA SEGMENT
1461 05AF 52      PUSH DX         ; SAVE TABLE SEGMENT ON STACK
1462 05B0 0B D4      OR DX,S1         ; CHECK FOR VALID TABLE DEFINED
1463 05B2 75 05      JNZ S2          ; CONTINUE IF DS:S1 NOT 0000:0000
1464
1465 05B4 58      POP AX         ; ELSE SET (AX)= 0000 FOR "NULL"
1466 05B5 8E 0000 E  MOV SI,OFFSET CRT_CHAR_GEN ; POINT TO DEFAULT TABLE OFFSET
1467 05B8 0E      PUSH CS        ; IN THE CODE SEGMENT
1468
1469          ;----- DETERMINE GRAPHICS MODE IN OPERATION
1470
1471 05B9          S2:          ; DETERMINE MODE
1472 05B9 D1 E0      SAL AX,1        ; MULTIPLY CODE POINT VALUE BY 8
1473 05BD D1 E0      SAL AX,1
1474 05BD D1 E0      SAL AX,1
1475 05BF 03 F0      ADD SI,AX       ; SI HAS OFFSET OF DESIRED CODES
1476 05C1 80 3E 0049 R 06  CMP 0CRT_MODE,6
1477 05C6 1F      POP DS         ; RECOVER TABLE POINTER SEGMENT
1478 05C7 72 2C      JC S7          ; TEST FOR MEDIUM RESOLUTION MODE
1479
1480          ;----- HIGH RESOLUTION MODE
1481
1482 05C9          S3:          ; HIGH_CHAR
1483 05C9 57      PUSH DI        ; SAVE REGEN POINTER
1484 05CA 56      PUSH SI        ; SAVE CODE POINTER
1485 05CB B6 04      MOV DH,4       ; NUMBER OF TIMES THROUGH LOOP
1486 05CD
1487 05CD AC          S4:          ; GET BYTE FROM CODE POINTS
1488 05CE F6 C3 80  TEST BL,80H    ; SHOULD WE USE THE FUNCTION
1489 05D1 75 16      JNZ S6         ; TO PUT CHAR IN
1490 05D3 AA      STOSB         ; STORE IN REGEN BUFFER
1491 05D4 AC
1492 05D5
1493 05D5 26 88 85 1FFF  S5:          ; STORE IN SECOND HALF
1494 05DA 83 C7 4F  ADD DI,79     ; MOVE TO NEXT ROW IN REGEN
1495 05DD FE CE      DEC DH        ; DONE WITH LOOP
1496 05DF 75 EC      JNZ S4
1497 05E1 5E      POP SI        ; RECOVER REGEN POINTER
1498 05E2 8F      POP DI        ; POINT TO NEXT CHAR POSITION
1499 05E3 47      INC DI        ; MORE CHARS TO WRITE
1500 05E4 E2 E3      LOOP S3
1501 05E6 E9 013D R  JMP VIDEO_RETURN
1502
1503 05E9          S6:          ; EXCLUSIVE OR WITH CURRENT
1504 05E9 26 32 06  XOR AL,ESI[DI] ; STORE THE CODE POINT
1505 05EC AA      STOSB         ; AGAIN FOR ODD FIELD
1506 05ED AC
1507 05EE 26 32 85 1FFF  XOR AL,ESI[DI+2000H-1]
1508 05F3 EB E0      JMP S5         ; BACK TO MAINSTREAM
1509
1510          ;----- MEDIUM RESOLUTION WRITE
1511
1512 05F5          S7:          ; MED RES WRITE
1513 05F5 8A D3      MOV DL,BL      ; SAVE HIGH COLOR BIT
1514 05F7 D1 E7      SAL DI,1      ; OFFSET*2 SINCE 2 BYTES/CHAR
1515          ; EXPAND BL TO FULL WORD OF COLOR
1516 05F9 80 E3 03  AND BL,3      ; ISOLATE THE COLOR BITS ( LOW 2 BITS )
1517 05FC B0 55      MOV AL,055H   ; GET BIT CONVERSION MULTIPLIER
1518 05FE F6 E3      MUL BL        ; EXPAND 2 COLOR BITS TO 4 REPLICATIONS
1519 0600 8A D8      MOV BL,AL     ; PLACE BACK IN WORK REGISTER
1520 0602 8A F8      MOV BH,AL     ; EXPAND TO 8 REPLICATIONS OF COLOR BITS
1521 0604          S8:          ; MED CHAR
1522 0604 57      PUSH DI       ; SAVE REGEN POINTER
1523 0605 56      PUSH SI       ; SAVE THE CODE POINTER
1524 0606 B6 04      MOV DH,4      ; NUMBER OF LOOPS
1525 0608
1526 0609 AC          S9:          ; GET CODE POINT
1527 0609 E8 06C4 R  CALL S21      ; DOUBLE UP ALL THE BITS
1528 060C 23 C3      AND AX,BX     ; CONVERT TO FOREGROUND COLOR ( 0 BACK )
1529 060E 86 E0      XCHG AH,AL   ; SWAP HIGH/LOW BYTES FOR WORD MOVE
1530 0610 F6 C2 80  TEST DL,80H  ; IS THIS XOR FUNCTION
1531 0613 74 03      JZ S10        ; NO, STORE IT IN AS IT IS
1532 0615 26 33 05  XOR AX,ESI[DI] ; DO FUNCTION WITH LOW/HIGH
1533 0618
1534 0618 26 89 05  S10:         ; STORE FIRST BYTE HIGH, SECOND LOW
1535 061B AC      MOV LDO SB    ; GET CODE POINT
1536 061C E8 06C4 R  CALL S21
1537 061F 23 C3      AND AX,BX     ; CONVERT TO COLOR
1538 0621 86 E0      XCHG AH,AL   ; SWAP HIGH/LOW BYTES FOR WORD MOVE
1539 0623 F6 C2 80  TEST DL,80H  ; AGAIN, IS THIS XOR FUNCTION
1540 0626 74 05      JZ S11        ; NO, JUST STORE THE VALUES
1541 0628 26 33 85 2000  XOR AX,ESI[DI+2000H] ; FUNCTION WITH FIRST HALF LOW
    
```

SECTION 5

```

1541 062D          S11:      MOV     ES:[DI+2000H],AX      ; STORE SECOND PORTION HIGH
1542 062D 26: 89 85 2000    ADD     DI,80                ; POINT TO NEXT LOCATION
1543 0632 83 C7 80          DEC     DH
1544 0635 FE CE           JNZ     S9                   ; KEEP GOING
1546 0639 5E           POP     SI                   ; RECOVER CODE POINTER
1547 063A 5F           POP     DI                   ; RECOVER REGEN POINTER
1548 063B 47           INC     DI                   ; POINT TO NEXT CHAR POSITION
1549 063C 47           INC     DI
1550 063D E2 C5          LOOP   S8                   ; MORE TO WRITE
1551 063F E9 013D R        JMP     VIDEO_RETURN
1552 0642          GRAPHICS_WRITE ENDP
1553          -----
1554          ; GRAPHICS READ
1555          -----
1556 0642          GRAPHICS_READ PROC NEAR
1557 0642 E8 06ED R        CALL   S26                  ; CONVERTED TO OFFSET IN REGEN
1558 0645 8B F0          MOV     SI,AX                ; SAVE IN SI
1559 0647 83 EC 08        SUB     SP,8                 ; ALLOCATE SPACE FOR THE READ CODE POINT
1560 064A 8B EC          MOV     BP,SP                ; POINTER TO SAVE AREA
1561          -----
1562          ;----- DETERMINE GRAPHICS MODES
1563 064C 80 3E 0049 R 06   CMP     CRT_MODE,6
1564 0651 06           PUSH   ES                   ; POINT TO REGEN SEGMENT
1565 0652 1F           POP     DS                   ; MEDIUM RESOLUTION
1566 0653 72 19          JC     S13
1567          -----
1568          ;----- HIGH RESOLUTION READ
1569          ;----- GET VALUES FROM REGEN BUFFER AND CONVERT TO CODE POINT
1570 0655 B6 04          MOV     DH,4                 ; NUMBER OF PASSES
1571 0657          S12:      MOV     AL,[SI]              ; GET FIRST BYTE
1572 0657 8A 04          MOV     [BP],AL              ; SAVE IN STORAGE AREA
1573 0659 88 46 00        INC     BP                    ; NEXT LOCATION
1574 065C 45 F8          INC     BP                    ; GET LOWER REGION BYTE
1575 065D 8A 84 2000     MOV     AL,[SI+2000H]        ; ADJUST AND STORE
1576 0661 88 46 00        MOV     [BP],AL
1577 0664 45 F8          INC     BP                    ; POINTER INTO REGEN
1578 0665 83 C6 50        ADD     SI,80                ; LOOP CONTROL
1579 0668 FE CE         DEC     DH
1580 066A 75 EB          JNZ     S12                  ; DO IT SOME MORE
1581 066C EB 16          JMP     SHORT S15            ; GO MATCH THE SAVED CODE POINTS
1582          -----
1583          ;----- MEDIUM RESOLUTION READ
1584 066E D1 E6          S13:      SAL     SI,1                 ; OFFSET*2 SINCE 2 BYTES/CHAR
1585 0670 B6 04          MOV     DH,4                 ; NUMBER OF PASSES
1586 0672          S14:      CALL   S23                  ; GET BYTES FROM REGEN INTO SINGLE SAVE
1587 0672 E8 06D3 R        CALL   S1,2000H-2           ; GO TO LOWER REGION
1588 0675 81 C6 1FFE       CALL   S23                  ; GET THIS PAIR INTO SAVE
1589 0679 E8 06D3 R        SUB     SI,2000H-80+2       ; ADJUST POINTER BACK INTO UPPER
1590 067C 81 EE 1FB2       DEC     DH
1591 0680 FE CE         JNZ     S14                  ; KEEP GOING UNTIL ALL 8 DONE
1592          -----
1593          ;----- SAVE AREA HAS CHARACTER IN IT, MATCH IT
1594          ;----- FIND_CHAR
1595 0684 BF 0000 E        MOV     DI,OFFSET CRT_CHAR_GEN ; ESTABLISH ADDRESSING
1596 0687 0E           PUSH   CS                    ; CODE POINTS IN CS
1597 0688 07           POP     ES                    ; ADJUST POINTER TO START OF SAVE AREA
1598 0689 83 ED 08        SUB     BP,8                 ; CURRENT CODE POINT BEING MATCHED
1599 068C 8B F8          MOV     SI,BP
1600 068E 8B 00          MOV     AL,0
1601 0690          S16:      PUSH   SS                    ; ESTABLISH ADDRESSING TO STACK
1602 0690          POP     DS                    ; FOR THE STRING COMPARE
1603 0691 1F           MOV     DX,128               ; NUMBER TO TEST AGAINST
1604 0692 BA 0080        PUSH   SI
1605 0695          S17:      PUSH   SI                    ; SAVE SAVE AREA POINTER
1606 0695          PUSH   DI                    ; SAVE CODE POINTER
1607 0696 56           MOV     CX,4                 ; NUMBER OF WORDS TO MATCH
1608 0697 B9 0004        REPE   CMPSW                 ; COMPARE THE 8 BYTES AS WORDS
1609 069A F3/ A7          POP     DI                    ; RECOVER THE POINTERS
1610 069C 8F           POP     SI
1611 069D 5E           JZ     S18                    ; IF ZERO FLAG SET, THEN MATCH OCCURRED
1612 069E 74 1E          INC     AL                    ; NO MATCH, MOVE ON TO NEXT
1613 06A0 FE C0          ADD     DI,8                 ; NEXT CODE POINT
1614 06A2 83 C7 08        DEC     DX                    ; LOOP CONTROL
1615 06A5 4A           JNZ     S17                  ; DO ALL OF THEM
1616          -----
1617          ;----- CHAR NOT MATCHED, MIGHT BE IN USER SUPPLIED SECOND HALF
1618          ;----- AL<= 0 IF ONLY 1ST HALF SCANNED
1619          ;----- IF = 0, THEN ALL HAS BEEN SCANNED
1620 06A8 3C 00          CMP     AL,0
1621 06AA 74 1E          JE     S18                    ; IF ALL 0, THEN DOESN'T EXIST
1622 06AC 2B C0          SUB     AX,AX                ; ESTABLISH ADDRESSING TO VECTOR
1623 06AE 8E D8          MOV     DS,AX
1624          ASSUME DS:ABS0
1625 06B0 C4 3E 007C R    LES     DI,TEXT_PTR         ; GET POINTER
1626 06B4 8C C0          MOV     AX,ES                ; SEE IF THE POINTER REALLY EXISTS
1627 06B6 0B C7          OR     AX,DI                 ; IF ALL 0, THEN DOESN'T EXIST
1628 06B8 74 04          JZ     S18                    ; NO SENSE LOOKING
1629 06BA 80 80          MOV     AL,128               ; ORIGIN FOR SECOND HALF
1630 06BC EB D2          JMP     S16                  ; GO BACK AND TRY FOR IT
1631          ASSUME DS:DATA
1632          -----
1633          ;----- CHARACTER IS FOUND ( AL=0 IF NOT FOUND )
1634 06BE 83 C4 08        ADD     SP,8                 ; READJUST THE STACK, THROW AWAY SAVE
1635 06C1 E9 013D R        JMP     VIDEO_RETURN        ; ALL DONE
1636 06C4          GRAPHICS_READ ENDP
1637          -----
1638          ; EXPAND BYTE
1639          ; THIS ROUTINE TAKES THE BYTE IN AL AND DOUBLES ALL
1640          ; OF THE BITS, TURNING THE 8 BITS INTO 16 BITS.
1641          ; THE RESULT IS LEFT IN AX
1642          -----
1643 06C4          S21:      PROC NEAR
1644 06C4 51           PUSH   CX                    ; SAVE REGISTER
1645 06C5 B9 0008        MOV     CX,8                 ; SHIFT COUNT REGISTER FOR ONE BYTE
1646 06C8          S22:      ROR     AL,1                 ; SHIFT BITS, LOW BIT INTO CARRY FLAG
1647 06C8 D0 C6          RCR     BP,1                 ; MOVE CARRY FLAG (LOW BIT) INTO RESULTS
1648 06CA D1 DD          SAR     BP,1                 ; SIGN EXTEND HIGH BIT (DOUBLE IT)
1649 06CC D1 FD          LOOP   S22                   ; REPEAT FOR ALL 8 BITS
1650 06CE E2 F8          XCHG   AX,BP                ; MOVE RESULTS TO PARAMETER REGISTER
1651 06D0 95           POP     CX                    ; RECOVER REGISTER
1652 06D1 59           RET
1653 06D2 C3          ; ALL DONE
1654 06D3          S21:      ENDP

```

```

1655 ;-----
1656 ; MED READ BYTE
1657 ; THIS ROUTINE WILL TAKE 2 BYTES FROM THE REGEN BUFFER,
1658 ; COMPARE AGAINST THE CURRENT FOREGROUND COLOR, AND PLACE
1659 ; THE CORRESPONDING ON/OFF BIT PATTERN INTO THE CURRENT
1660 ; POSITION IN THE SAVE AREA
1661 ; ENTRY ---
1662 ; SI,DS = POINTER TO REGEN AREA OF INTEREST
1663 ; BX = EXPANDED FOREGROUND COLOR
1664 ; BP = POINTER TO SAVE AREA
1665 ; EXIT --
1666 ; SI AND BP ARE INCREMENTED
1667 ;-----
1668 PROC NEAR
1669 LODSW NEAR ; GET FIRST BYTE AND SECOND BYTES
1670 MOV AL,AX ; SWAP FOR COMPARE
1671 MOV CX,0C000H ; 2 BIT MASK TO TEST THE ENTRIES
1672 MOV DL,0 ; RESULT REGISTER
1673 S24:
1674 TEST AX,CX ; IS THIS SECTION BACKGROUND?
1675 JZ S25 ; IF ZERO, IT IS BACKGROUND (CARRY=0)
1676 STC ; WASN'T, SO SET CARRY
1677 S25:
1678 RCL DL,1 ; MOVE THAT BIT INTO THE RESULT
1679 SHR CX,1 ;
1680 SHR AX,1 ; MOVE THE MASK TO THE RIGHT BY 2 BITS
1681 JNC S24 ; DO IT AGAIN IF MASK DIDN'T FALL OUT
1682 MOV [BP],DL ; STORE RESULT IN SAVE AREA
1683 INC BP ; ADJUST POINTER
1684 RET ; ALL DONE
1685 ENDP
1686 ;-----
1687 ; V4 POSITION
1688 ; THIS ROUTINE TAKES THE CURSOR POSITION CONTAINED IN
1689 ; THE MEMORY LOCATION, AND CONVERTS IT INTO AN OFFSET
1690 ; INTO THE REGEN BUFFER, ASSUMING ONE BYTE/CHAR.
1691 ; FOR MEDIUM RESOLUTION GRAPHICS, THE NUMBER MUST
1692 ; BE DOUBLED.
1693 ; ENTRY -- NO REGISTERS, MEMORY LOCATION @CURSOR_POSN IS USED
1694 ; EXIT --
1695 ; AX CONTAINS OFFSET INTO REGEN BUFFER
1696 ;-----
1697 PROC NEAR
1698 MOV AX,@CURSOR_POSN ; GET CURRENT CURSOR
1699 LABEL NEAR
1700 PUSH BX ; SAVE REGISTER
1701 MOV BX,AX ; SAVE A COPY OF CURRENT CURSOR
1702 MOV AL,BYTE PTR @CRT_COLS ; GET BYTES PER COLUMN
1703 MUL AH ; MULTIPLY BY ROWS
1704 SHL AX,1 ;
1705 SHL AX,1 ; MULTIPLY * 4 SINCE 4 ROWS/BYTE
1706 SUB BH,BH ; ISOLATE COLUMN VALUE
1707 ADD AX,BX ; DETERMINE OFFSET
1708 POP BX ; RECOVER POINTER
1709 RET ; ALL DONE
1710 ENDP
1711 ;----- WRITE_TTY
1712 ;
1713 ; THIS INTERFACE PROVIDES A TELETYPE LIKE INTERFACE TO THE
1714 ; VIDEO CARDS. THE INPUT CHARACTER IS WRITTEN TO THE CURRENT
1715 ; CURSOR POSITION, AND THE CURSOR IS MOVED TO THE NEXT POSITION.
1716 ; IF THE CURSOR LEAVES THE LAST COLUMN OF THE FIELD, THE COLUMN
1717 ; IS SET TO ZERO, AND THE ROW VALUE IS INCREMENTED. IF THE ROW
1718 ; VALUE LEAVES THE FIELD, THE CURSOR IS PLACED ON THE LAST ROW,
1719 ; FIRST COLUMN, AND THE ENTIRE SCREEN IS SCROLLED UP ONE LINE.
1720 ; WHEN THE SCREEN IS SCROLLED UP, THE ATTRIBUTE FOR FILLING THE
1721 ; NEWLY BLANKED LINE IS READ FROM THE CURSOR POSITION ON THE PREVIOUS
1722 ; LINE BEFORE THE SCROLL, IN CHARACTER MODE. IN GRAPHICS MODE,
1723 ; THE 0 COLOR IS USED.
1724 ; ENTRY ---
1725 ; (AH) = CURRENT CRT MODE
1726 ; (AL) = CHARACTER TO BE WRITTEN
1727 ; NOTE THAT BACK SPACE, CARRIAGE RETURN, BELL AND LINE FEED ARE
1728 ; HANDLED AS COMMANDS RATHER THAN AS DISPLAY GRAPHICS CHARACTERS
1729 ; (BL) = FOREGROUND COLOR FOR CHAR WRITE IF CURRENTLY IN A GRAPHICS MODE
1730 ; EXIT --
1731 ; ALL REGISTERS SAVED THROUGH VIDEO_EXIT (INCLUDING AX)
1732 ;-----
1733 ASSUME DS:DATA
1734 PROC NEAR
1735 XCHG DI,AX ; SAVE (AX) REGISTER IN (DI) FOR EXIT
1736 MOV AH,03H ; READ CURSOR POSITION
1737 MOV BH,@ACTIVE_PAGE ; GET CURRENT PAGE SETTING
1738 INT 10H ; READ THE CURRENT CURSOR POSITION
1739 MOV AX,DI ; RECOVER CHARACTER FROM (DI) REGISTER
1740 ;----- DX NOW HAS THE CURRENT CURSOR POSITION
1741 ;-----
1742 CMP AL,CR ; IS IT CARRIAGE RETURN OR CONTROL
1743 JBE UB ; GO TO CONTROL CHECKS IF IT IS
1744 ;----- WRITE THE CHAR TO THE SCREEN
1745 UD: MOV AH,0AH ; WRITE CHARACTER ONLY COMMAND
1746 MOV CX,1 ; ONLY ONE CHARACTER
1747 INT 10H ; WRITE THE CHARACTER
1748 ;----- POSITION THE CURSOR FOR NEXT CHAR
1749 INC DL
1750 CMP DL,BYTE PTR @CRT_COLS ; TEST FOR COLUMN OVERFLOW
1751 JNZ U7 ; SET CURSOR
1752 MOV DL,0 ; COLUMN FOR CURSOR
1753 CMP DH,25-1 ; CHECK FOR LAST ROW
1754 JNZ U6 ; SET_CURSOR_INC
1755 ;----- SCROLL REQUIRED
1756 U1: MOV AH,02H ; SET THE CURSOR
1757 INT 10H ;
1758 ;----- DETERMINE VALUE TO FILL WITH DURING SCROLL
1759 MOV AL,@CRT_MODE ; GET THE CURRENT MODE
1760 CMP AL,5 ;
1761 JC U2 ; READ-CURSOR
1762 ;
1763 ;
1764 ;
1765 ;
1766 MOV AL,@CRT_MODE ; GET THE CURRENT MODE
1767 CMP AL,5 ;
1768 JC U2 ; READ-CURSOR

```

SECTION 5

```

1769 0732 3C 07      CMP     AL,7
1770 0734 87 00      MOV     BH,0
1771 0736 75 06      JNE     U3
1772 0738          U2:
1773 0738 B4 08      MOV     AH,08H
1774 073A CD 10      INT     10H
1775 073C 8A FC      MOV     BH,AH
1776 073E          U3:
1777 073E B8 0601     MOV     AX,0601H
1778 0741 2B C9      SUB     CX,CX
1779 0743 B6 18      MOV     DH,25-1
1780 0745 8A 16 004A R MOV     DL,BYTE PTR @CRT_COLS
1781 0749 FE CA      DEC     DL
1782 074B          U4:
1783 074B CD 10      INT     10H
1784 074D          U5:
1785 074D 97        XCHG   AX,DI
1786 074E E9 013D R   JMP     VIDEO_RETURN
1787
1788 0751          U6:
1789 0751 FE C6      INC     DH
1790 0753          U7:
1791 0753 B4 02      MOV     AH,02H
1792 0755 EB F4      JMP     U4
1793
1794          ;----- CHECK FOR CONTROL CHARACTERS
1795 0757          U8:
1796 0757 74 13      JE     U9
1797 0759 3C 0A      CMP     AL,LF
1798 075B 74 13      JE     U10
1799 075D 3C 07      CMP     AL,07H
1800 075F 74 16      JE     U11
1801 0761 3C 08      CMP     AL,08H
1802 0763 75 AC      JNE     U0
1803
1804          ;----- BACK SPACE FOUND
1805
1806 0765 0A D2      OR     DL,DL
1807 0767 74 EA      JE     UT
1808 0769 4A        DEC     DX
1809 076A EB E7      JMP     UT
1810
1811          ;----- CARRIAGE RETURN FOUND
1812 076C B2 00      MOV     DL,0
1813 076E EB E3      JMP     UT
1814
1815          ;----- LINE FEED FOUND
1816 0770 80 FE 18   CMP     DH,25-1
1817 0773 75 DC      JNE     U6
1818 0775 EB E0      JMP     U1
1819
1820          ;----- BELL FOUND
1821 0777 B9 0533     MOV     CX,1331
1822 077A 93 1F      MOV     BL,31
1823 077C E8 0000 E   CALL    BEEP
1824 077F EB CC      JMP     U5
1825 0781          ENDP
1826
1827          ; LIGHT PEN
1828          ; THIS ROUTINE TESTS THE LIGHT PEN SWITCH AND THE LIGHT
1829          ; PEN TRIGGER. IF BOTH ARE SET, THE LOCATION OF THE LIGHT
1830          ; PEN IS DETERMINED. OTHERWISE, A RETURN WITH NO INFORMATION
1831          ; IS MADE.
1832          ; ON EXIT:
1833          ; (AH) = 0 IF NO LIGHT PEN INFORMATION IS AVAILABLE
1834          ; (AH) = 1 IF LIGHT PEN IS AVAILABLE
1835          ; (DH,DL) = ROW,COLUMN OF CURRENT LIGHT PEN POSITION
1836          ; (CH) = RASTER POSITION
1837          ; (BX) = BEST GUESS AT PIXEL HORIZONTAL POSITION
1838          ;-----
1839
1840          ASSUME DS:DATA
1841 0781 03 03 05 03 03 V1 DB 3,3,5,6,3,3,3,4
1842 0782          ; SUBTRACT_TABLE
1843
1844          ;----- WAIT FOR LIGHT PEN TO BE DEPRESSED
1845 0789          READ_LPEN PROC NEAR
1846 0789 B4 00      MOV     AH,0
1847 078B 8B 16 0063 R MOV     DX,@ADDR_6845
1848 078F 83 C2 06     ADD     DX,5
1849 0792 EC      IN     AL,DX
1850 0793 A8 04      TEST    AL,004H
1851 0795 74 03      JZ     V6_A
1852 0797 E9 081C R   JMP     V6
1853
1854          ;----- NOW TEST FOR LIGHT PEN TRIGGER
1855
1856 079A A8 02      V6_A: TEST    AL,2
1857 079C 75 03      JNZ     V7A
1858 079E E9 0826 R   JMP     V7
1859
1860          ;----- TRIGGER HAS BEEN SET, READ THE VALUE IN
1861
1862 07A1 B4 10      V7A: MOV     AH,16
1863
1864          ;----- INPUT REGISTERS POINTED TO BY AH, AND CONVERT TO ROW COLUMN IN (DX)
1865
1866 07A3 8B 16 0063 R MOV     DX,@ADDR_6845
1867 07A7 8A C4      MOV     AL,AH
1868 07A9 EE      OUT     DX,AL
1869 07AA EB 00      JMP     $+2
1870 07AC 42      INC     DX
1871 07AD EC      IN     AL,DX
1872 07AE 8A E8      MOV     CH,AL
1873 07B0 4A      DEC     DX
1874 07B1 FE C4      INC     AH
1875 07B3 8A C4      MOV     AL,AH
1876 07B5 EE      OUT     DX,AL
1877 07B6 42      INC     DX
1878 07B7 EB 00      JMP     $+2
1879 07B9 EC      IN     AL,DX
1880 07BA 8A E5      MOV     AH,CH
  
```

```

1881
1882 PAGE
1883 ;----- AX HAS THE VALUE READ IN FROM THE 6845
1884 MOV BL,#CRT_MODE
1885 SUB BH,BH ; MODE VALUE TO BX
1886 MOV BL,CS:VI[BX] ; DETERMINE AMOUNT TO SUBTRACT
1887 SUB AX,BX ; TAKE IT AWAY
1888 MOV BX,#CRT_START
1889 SHR BX,1
1890 SUB AX,BX ; CONVERT TO CORRECT PAGE ORIGIN
1891 JNS V2 ; IF POSITIVE, DETERMINE MODE
1892 SUB AX,AX ; <0 PLAYS AS 0
1893
1894 ;----- DETERMINE MODE OF OPERATION
1895
1896 V2:
1897 MOV CL,3 ; DETERMINE MODE
1898 CMP #CRT_MODE,4 ; SET *8 SHIFT COUNT
1899 JB V3 ; DETERMINE IF GRAPHICS OR ALPHA
1900 CMP #CRT_MODE,7 ; ALPHA_PEN
1901 JE V4 ; ALPHA_PEN
1902
1903 ;----- GRAPHICS MODE
1904
1905 MOV DL,40 ; DIVISOR FOR GRAPHICS
1906 DIV DL ; DETERMINE ROW(AL) AND COLUMN(AH)
1907 ; AL RANGE 0-99, AH RANGE 0-39
1908
1909 ;----- DETERMINE GRAPHIC ROW POSITION
1910 MOV CH,AL ; SAVE ROW VALUE IN CH
1911 ADD CH,CH ; *2 FOR EVEN/ODD FIELD
1912 MOV BH,AH ; COLUMN VALUE TO BX
1913 SUB BH,BH ; MULTIPLY BY 8 FOR MEDIUM RES
1914 CMP #CRT_MODE,6 ; DETERMINE MEDIUM OR HIGH RES
1915 JNE V3 ; NOT HIGH RES
1916 MOV CL,4 ; SHIFT VALUE FOR HIGH RES
1917 SAL AH,1 ; COLUMN VALUE TIMES 2 FOR HIGH RES
1918 MOV V3,CL ; NOT HIGH RES
1919 SHL BX,CL ; MULTIPLY *16 FOR HIGH RES
1920
1921 ;----- DETERMINE ALPHA CHAR POSITION
1922
1923 MOV DL,AH ; COLUMN VALUE FOR RETURN
1924 MOV DH,AL ; ROW VALUE
1925 SHR DH,1 ; DIVIDE BY 4
1926 SHR DH,1 ; FOR VALUE IN 0-24 RANGE
1927 JMP SHORT V6 ; LIGHT_PEN_RETURN_SET
1928
1929 ;----- ALPHA MODE ON LIGHT PEN
1930
1931 V4:
1932 DIV BYTE PTR #CRT_COLS ; ALPHA PEN
1933 MOV DH,AL ; DETERMINE ROW,COLUMN VALUE
1934 MOV DL,AH ; ROWS TO DH
1935 SAL AL,CL ; COLS TO DL
1936 MOV CH,AL ; MULTIPLY ROWS * 8
1937 MOV BL,AH ; GET RASTER VALUE TO RETURN REGISTER
1938 XOR BH,BH ; COLUMN VALUE
1939 SAL BX,CL ; TO BX
1940
1941 V5:
1942 MOV AH,1 ; LIGHT PEN RETURN SET
1943 PUSH DX ; INDICATE EVERY THING SET
1944 MOV DX,#ADDR_6845 ; LIGHT PEN RETURN
1945 ADD DX,7 ; SAVE RETURN VALUE (IN CASE)
1946 OUT DX,AL ; GET BASE ADDRESS
1947 POP DX ; POINT TO RESET PARM
1948 ; ADDRESS, NOT DATA, IS IMPORTANT
1949 ; RECOVER VALUE
1950 ; RETURN_NO_RESET
1951
1952 V7:
1953 POP BP
1954 POP DI
1955 POP SI
1956 POP DS
1957 POP DS ; DISCARD SAVED BX,CX,DX
1958 POP DS
1959 POP ES
1960 IRET
1961 READ_LPEN ENDP
1962 CODE ENDS
1963 END
    
```

SECTION 5

```

1 PAGE 110,121
2 TITLE BIOS ----- 06/10/85 BIOS ROUTINES
3 .286C
4 .LIST
5 0000 CODE SEGMENT BYTE PUBLIC
6
7 PUBLIC EQUIPMENT_1
8 PUBLIC MEMORY_SIZE_DET_1
9 PUBLIC NM1_INT_1
10
11 EXTRN C8042:NEAR ; POST SEND 8042 COMMAND ROUTINE
12 EXTRN CMOS_READ:NEAR ; READ CMOS LOCATION ROUTINE
13 EXTRN D1:NEAR ; *PARITY CHECK 1* MESSAGE
14 EXTRN D2:NEAR ; *PARITY CHECK 2* MESSAGE
15 EXTRN DSA:NEAR ; *????* UNKNOWN ADDRESS MESSAGE
16 EXTRN DDS:NEAR ; LOAD (DS) WITH DATA SEGMENT SELECTOR
17 EXTRN OBF_42:NEAR ; POST WAIT 8042 RESPONSE ROUTINE
18 EXTRN PRT_HEX:NEAR ; DISPLAY CHARACTER ROUTINE
19 EXTRN PRT_SEG:NEAR ; DISPLAY FIVE CHARACTER ADDRESS ROUTINE
20 EXTRN P_MSG:NEAR ; DISPLAY MESSAGE STRING ROUTINE
21
22
23 ----- INT 12 H -----
24 MEMORY_SIZE_DETERMINE
25 THIS ROUTINE RETURNS THE AMOUNT OF MEMORY IN THE SYSTEM AS
26 DETERMINED BY THE POST ROUTINES. (UP TO 640K)
27 NOTE THAT THE SYSTEM MAY NOT BE ABLE TO USE I/O MEMORY UNLESS
28 THERE IS A FULL COMPLEMENT OF 512K BYTES ON THE PLANAR.
29
30 INPUT
31 NO REGISTERS
32 THE MEMORY SIZE VARIABLE IS SET DURING POWER ON DIAGNOSTICS
33 ACCORDING TO THE FOLLOWING ASSUMPTIONS:
34
35 1. CONFIGURATION RECORD IN NON-VOLATILE MEMORY EQUALS THE ACTUAL
36 MEMORY SIZE INSTALLED.
37
38 2. ALL INSTALLED MEMORY IS FUNCTIONAL. IF THE MEMORY TEST DURING
39 POST INDICATES LESS, THEN THIS VALUE BECOMES THE DEFAULT.
40 IF NON-VOLATILE MEMORY IS NOT VALID (NOT INITIALIZED OR BATTERY
41 FAILURE) THEN ACTUAL MEMORY DETERMINED BECOMES THE DEFAULT.
42
43 3. ALL MEMORY FROM 0 TO 640K MUST BE CONTIGUOUS.
44
45 OUTPUT
46 (AX) = NUMBER OF CONTIGUOUS 1K BLOCKS OF MEMORY
47
48 -----
49 ASSUME CS:CODE,DS:DATA
50
51 MEMORY_SIZE_DET_1 PROC FAR
52 STI ; INTERRUPTS BACK ON
53 PUSH DS ; SAVE SEGMENT
54 CALL DDS ; ESTABLISH ADDRESSING
55 MOV AX, #MEMORY_SIZE ; GET VALUE
56 POP DS ; RECOVER SEGMENT
57 IRET ; RETURN TO CALLER
58 MEMORY_SIZE_DET_1 ENDP
59
60 ----- INT 11 H -----
61 EQUIPMENT_DETERMINATION
62 THIS ROUTINE ATTEMPTS TO DETERMINE WHAT OPTIONAL
63 DEVICES ARE ATTACHED TO THE SYSTEM.
64
65 INPUT
66 NO REGISTERS
67 THE #EQUIP_FLAG VARIABLE IS SET DURING THE POWER ON
68 DIAGNOSTICS USING THE FOLLOWING HARDWARE ASSUMPTIONS:
69 PORT 03FA = INTERRUPT ID REGISTER OF 8250 (PRIMARY)
70 03FA = INTERRUPT ID REGISTER OF 8250 (SECONDARY)
71 BITS 7-3 ARE ALWAYS 0
72 PORT 0378 = OUTPUT PORT OF PRINTER (PRIMARY)
73 0278 = OUTPUT PORT OF PRINTER (SECONDARY)
74 03BC = OUTPUT PORT OF PRINTER (MONOCHROME-PRINTER)
75
76 OUTPUT
77 (AX) IS SET, BIT SIGNIFICANT, TO INDICATE ATTACHED I/O
78 BIT 15,14 = NUMBER OF PRINTERS ATTACHED
79 BIT 13 = INTERNAL MODEM INSTALLED
80 BIT 12 NOT USED
81 BIT 11,10,9 = NUMBER OF RS232 CARDS ATTACHED
82 BIT 8 = NOT USED
83 BIT 7,6 = NUMBER OF DISKETTE DRIVES
84 00=1, 01=2 ONLY IF BIT 0 = 1
85 BIT 5,4 = INITIAL VIDEO MODE
86 00 - UNUSED
87 01 - 40X25 BW USING COLOR CARD
88 10 - 80X25 BW USING COLOR CARD
89 11 - 80X25 BW USING BW CARD
90
91 BIT 3 = NOT USED
92 BIT 2 = NOT USED
93 BIT 1 = MATH COPROCESSOR
94 BIT 0 = 1 (IPL DISKETTE INSTALLED)
95 NO OTHER REGISTERS AFFECTED
96
97 -----
98 EQUIPMENT_1 PROC FAR
99 STI ; ENTRY POINT FOR ORG OF840H
100 PUSH DS ; INTERRUPTS BACK ON
101 CALL DDS ; SAVE SEGMENT REGISTER
102 MOV AX, #EQUIP_FLAG ; ESTABLISH ADDRESSING
103 POP DS ; GET THE CURRENT SETTINGS
104 IRET ; RECOVER SEGMENT
105 RETURN TO CALLER
106 EQUIPMENT_1 ENDP

```

```

101 PAGE
102 |--- HARDWARE INT 02 H -- ( NMI LEVEL ) -----
103 | NON-MASKABLE INTERRUPT ROUTINE (REAL MODE)
104 | THIS ROUTINE WILL PRINT A "PARITY CHECK 1 OR 2" MESSAGE AND ATTEMPT
105 | TO FIND THE STORAGE LOCATION IN BASE 640K CONTAINING THE BAD PARITY.
106 | IF FOUND, THE SEGMENT ADDRESS WILL BE PRINTED. IF NO PARITY ERROR
107 | CAN BE FOUND (INTERMITTENT READ PROBLEM) ????? WILL BE DISPLAYED
108 | WHERE THE ADDRESS WOULD NORMALLY GO.
109 |
110 | PARITY CHECK 1 = PLANAR BOARD MEMORY FAILURE.
111 | PARITY CHECK 2 = OFF PLANAR BOARD MEMORY FAILURE.
112 |-----
113
114 0014 NMI_INT_1 PROC NEAR
115 0014 50 PUSH AX ; SAVE ORIGINAL CONTENTS OF (AX)
116
117 0015 E4 61 IN AL,PORT_B ; READ STATUS PORT
118 0017 A8 C0 TEST AL,PARITY_ERR ; PARITY CHECK OR I/O CHECK ?
119 0019 75 07 JNZ NMI_1 ; GO TO ERROR HALTS IF HARDWARE ERROR
120
121 001B 80 0F MOV AL,CMOS_SHUT_DOWN ; ELSE ?? - LEAVE NMI ON
122 001D E8 0000 E CALL CMOS_READ ; TOGGLE NMI USING COMMON READ ROUTINE
123 0020 58 POP AX ; RESTORE ORIGINAL CONTENTS OF (AX)
124 0021 CF IRET ; EXIT NMI HANDLER BACK TO PROGRAM
125
126
127 0022 NMI_1: ; HARDWARE ERROR
128 0022 50 PUSH AX ; SAVE INITIAL CHECK MASK IN (AL)
129 0023 80 8F MOV AL,CMOS_SHUT_DOWN+NMI ; MASK TRAP (NMI) INTERRUPTS OFF
130 0025 E8 0000 E CALL CMOS_READ ; OPEN STANDBY LATCH BEFORE POWER DOWN
131 0028 80 AD MOV AL,DIS_KBD ; DISABLE THE KEYBOARD
132 002A E8 0000 E CALL CB042 ; SEND COMMAND TO ADAPTER
133 002D E8 0000 E CALL DDS ; ADDRESS DATA SEGMENT
134 0030 B4 00 MOV AH,0 ; INITIALIZE AND SET MODE FOR VIDEO
135 0032 A0 0049 R MOV AL,%CRT_MODE ; GET CURRENT MODE
136 0035 CD 10 INT 10H ; CALL VIDEO_IO TO CLEAR SCREEN
137
138 |----- DISPLAY "PARITY CHECK ?" ERROR MESSAGES
139
140 0037 58 POP AX ; RECOVER INITIAL CHECK STATUS
141 0038 BE 0000 E MOV SI,OFFSET D1 ; PLANAR ERROR, ADDRESS "PARITY CHECK 1"
142 0039 A8 C0 TEST AL,PARITY_CHECK ; CHECK FOR PLANAR ERROR
143 003D 74 05 JZ NMI_2 ; SKIP IF NOT
144
145 003F 50 PUSH AX ; SAVE STATUS
146 0040 E8 0000 E CALL P_MSG ; DISPLAY "PARITY CHECK 1" MESSAGE
147 0043 58 POP AX ; AND RECOVER STATUS
148
149 0044 BE 0000 E NMI_2: MOV SI,OFFSET D2 ; ADDRESS OF "PARITY CHECK 2" MESSAGE
150 0047 A8 C0 TEST AL,I/O_CHECK ; I/O PARITY CHECK ?
151 0049 74 03 JZ NMI_3 ; SKIP IF CORRECT ERROR DISPLAYED
152 004B E8 0000 E CALL P_MSG ; DISPLAY "PARITY CHECK 2" ERROR
153
154 |----- TEST FOR HOT NMI ON PLANAR PARITY LINE
155
156 004E NMI_3:
157 004E E4 61 IN AL,PORT_B ; TOGGLE PARITY CHECK ENABLES
158 0050 0C 0C OR AL,RAM_PAR_OFF ; TOGGLE PARITY CHECK ENABLES
159 0052 E6 61 OUT PORT_B,AL ; TO CLEAR THE PENDING CHECK
160 0054 24 F3 AND AL,RAM_PAR_ON ; TO CLEAR THE PENDING CHECK
161 0056 E6 61 OUT PORT_B,AL ; TO CLEAR THE PENDING CHECK
162
163 0058 FC CLD ; SET DIRECTION FLAG TO INCREMENT
164 0059 2B D2 SUB DX,DX ; POINT (DX) AT START OF REAL MEMORY
165 005B 2B F6 SUB SI,SI ; SET (SI) TO START OF (DS)
166 005D E4 61 IN AL,PORT_B ; READ CURRENT PARITY CHECK LATCH
167 005F A8 C0 TEST AL,PARITY_ERR ; CHECK FOR HOT NMI SOURCE
168 0061 75 19 JNZ NMI_5 ; SKIP IF ERROR NOT RESET (DISPLAY ???)
169
170 |----- SEE IF LOCATION THAT CAUSED PARITY CHECK CAN BE FOUND IN BASE MEMORY
171
172 0063 8B 1E 0013 R MOV BX,@MEMORY_SIZE ; GET BASE MEMORY SIZE WORD
173 0067 NMI_4:
174 0067 8E DA MOV DS,DX ; POINT TO 64K SEGMENT
175 0069 B9 8000 MOV CX,4000H*2 ; SET WORD COUNT FOR 64 KB SCAN
176 006C F3/ AD REP LODSW ; READ 64 KB OF MEMORY
177 006E E4 61 IN AL,PORT_B ; READ PARITY CHECK LATCHES
178 0070 A8 C0 TEST AL,PARITY_ERR ; CHECK FOR ANY PARITY ERROR PENDING
179 0072 75 10 JNZ NMI_6 ; GO PRINT SEGMENT ADDRESS IF ERROR
180
181 0074 80 C6 10 ADD DH,010H ; POINT TO NEXT 64K BLOCK
182 0077 83 EB 40 SUB BX,100*4 ; DECREMENT COUNT OF 1024 BYTE SEGMENTS
183 007A 77 EB JA NMI_4 ; LOOP TILL ALL 64K SEGMENTS DONE
184
185 007C BE 0000 E NMI_5: MOV SI,OFFSET D2A ; PRINT ROW OF ????? IF PARITY
186 007F E8 0000 E CALL P_MSG ; CHECK COULD NOT BE RE-CREATED
187 0082 FA CLI ; HALT SYSTEM
188 0083 F4 HLT ; HALT SYSTEM
189
190 0084 NMI_6:
191 0084 E8 0000 E CALL PRT_SEG ; PRINT SEGMENT VALUE (IN DX)
192 0087 B0 28 MOV AL,'1' ; PRINT (S)
193 0089 E8 0000 E CALL PRT_HEX ; PRINT (S)
194 008C B0 53 MOV AL,'S'
195 008E E8 0000 E CALL PRT_HEX ; PRINT (S)
196 0091 B0 29 MOV AL,'1'
197 0093 E8 0000 E CALL PRT_HEX ; PRINT (S)
198 0096 FA CLI ; HALT SYSTEM
199 0097 F4 HLT ; HALT SYSTEM
200
201 0098 NMI_INT_1 ENDP
202
203 0098 CODE ENDS
204 END

```

SECTION 5

```

1 PAGE 118,121
2 TITLE BIOS1 ---- 06/10/85 INTERRUPT 15H BIOS ROUTINES
3 .286C
4 .LIST
5 0000 CODE SEGMENT BYTE PUBLIC
6
7 PUBLIC CASSETTE_IO_1
8 PUBLIC GATE_A20
9 PUBLIC SHUT9
10
11 EXTRN CMOS_READ:NEAR ; READ CMOS LOCATION ROUTINE
12 EXTRN CMOS_WRITE:NEAR ; WRITE CMOS LOCATION ROUTINE
13 EXTRN CONF_TBL:NEAR ; SYSTEM/BIOS CONFIGURATION TABLE
14 EXTRN DDS:NEAR ; LOAD (DS) WITH DATA SEGMENT SELECTOR
15 EXTRN PROC_SHUTDOWN:NEAR ; 80286 HARDWARE RESET ROUTINE
16
17 ----- INT 15 H -----
18 INPUT - CASSETTE I/O FUNCTIONS
19 ;
20 ; (AH) = 00H
21 ; (AH) = 01H
22 ; (AH) = 02H
23 ; (AH) = 03H
24 ; RETURNS FOR THESE FUNCTIONS ALWAYS (AH) = 86H, CY = 1
25 ; IF CASSETTE PORT NOT PRESENT
26 -----
27 INPUT - UNUSED FUNCTIONS
28 ; (AH) = 04H THROUGH 7FH
29 ; RETURNS FOR THESE FUNCTIONS ALWAYS (AH) = 86H, CY = 1
30 ; (UNLESS INTERCEPTED BY SYSTEM HANDLERS)
31 ; NOTE: THE KEYBOARD INTERRUPT HANDLER INTERRUPTS WITH AH=4FH
32 -----
33 EXTENSIONS
34 ; (AH) = 80H DEVICE OPEN
35 ; (BX) = DEVICE ID
36 ; (CX) = PROCESS ID
37 ;
38 ; (AH) = 81H DEVICE CLOSE
39 ; (BX) = DEVICE ID
40 ; (CX) = PROCESS ID
41 ;
42 ; (AH) = 82H PROGRAM TERMINATION
43 ; (BX) = DEVICE ID
44 ;
45 ; (AH) = 83H EVENT WAIT
46 ;
47 ; (AL) = 00H SET INTERVAL
48 ; (ES:BX) POINTER TO A BYTE IN CALLERS MEMORY
49 ; THAT WILL HAVE THE HIGH ORDER BIT SET
50 ; AS SOON AS POSSIBLE AFTER THE INTERVAL
51 ; EXPIRES.
52 ; (CX,DX) NUMBER OF MICROSECONDS TO ELAPSE BEFORE
53 ; POSTING.
54 ; (AL) = 01H CANCEL
55 ;
56 ; RETURNS: CARRY IF AL NOT = 00H OR 01H
57 ; OR IF FUNCTION AL=0 ALREADY BUSY
58 ;
59 ; (AH) = 84H JOYSTICK SUPPORT
60 ; (DX) = 00H - READ THE CURRENT SWITCH SETTINGS
61 ; RETURNS AL = SWITCH SETTINGS (BITS 7-4)
62 ; (DX) = 01H - READ THE RESISTIVE INPUTS
63 ; RETURNS AX = A(x) VALUE
64 ; BX = A(y) VALUE
65 ; CX = B(x) VALUE
66 ; DX = B(y) VALUE
67 ;
68 ; (AH) = 85H SYSTEM REQUEST KEY PRESSED
69 ; (AL) = 00H MAKE OF KEY
70 ; (AL) = 01H BREAK OF KEY
71 ;
72 ; (AH) = 86H WAIT
73 ; (CX,DX) NUMBER OF MICROSECONDS TO ELAPSE BEFORE
74 ; RETURN TO CALLER
75 ;
76 ; (AH) = 87H MOVE BLOCK
77 ; (CX) NUMBER OF WORDS TO MOVE
78 ; (ES:SI) POINTER TO DESCRIPTOR TABLE
79 ;
80 ; (AH) = 88H EXTENDED MEMORY SIZE DETERMINE
81 ;
82 ; (AH) = 89H PROCESSOR TO VIRTUAL MODE
83 ;
84 ; (AH) = 90H DEVICE BUSY LOOP
85 ; (AL) SEE TYPE CODE
86 ;
87 ; (AH) = 91H INTERRUPT COMPLETE FLAG SET
88 ; (AL) TYPE CODE
89 ; 00H -> 7FH
90 ; SERIALLY REUSABLE DEVICES
91 ; OPERATING SYSTEM MUST SERIALIZE ACCESS
92 ; 80H -> BFH
93 ; REENRANT DEVICES; ES:BX IS USED TO
94 ; DISTINGUISH DIFFERENT CALLS (MULTIPLE I/O
95 ; CALLS ARE ALLOWED SIMULTANEOUSLY)
96 ; COH -> FFH
97 ; WAIT ONLY CALLS -- THERE IS NO
98 ; COMPLEMENTARY "POST" FOR THESE WAITS.
99 ; THESE ARE TIMEOUT ONLY. TIMES ARE
100 ; FUNCTION NUMBER DEPENDENT.
101 ;
102 ;
103 ;
104 ; TYPE DESCRIPTION TIMEOUT
105 ; 00H = DISK YES
106 ; 01H = DISKETTE YES
107 ; 02H = KEYBOARD NO
108 ; 80H = NETWORK NO
109 ; ES:BX --> NCB
110 ; FDH = DISKETTE MOTOR START YES
111 ; FEH = PRINTER YES

```



```

112 PAGE
113 (AH) = 00H RETURN CONFIGURATION PARAMETERS POINTER
114 RETURNS
115 (AH) = 00H AND CY= 0 (IF PRESENT ELSE 86 AND CY= 1)
116 (ES:BX) = PARAMETER TABLE ADDRESS POINTER
117 WHERE:
118
119 DW 8 LENGTH OF FOLLOWING TABLE
120 DB MODEL_BYTE SYSTEM MODEL BYTE
121 DB TYPE_BYTE SYSTEM MODEL TYPE BYTE
122 DB BIOS_LEVEL BIOS REVISION LEVEL
123 DB ? 10000000 = DMA CHANNEL 3 USE BY BIOS
124 01000000 = CASCADED INTERRUPT LEVEL 2
125 00100000 = REAL TIME CLOCK AVAILABLE
126 00010000 = KEYBOARD SCAN CODE HOOK (AH)
127 DB 0 RESERVED
128 DB 0 RESERVED
129 DB 0 RESERVED
130 DB 0 RESERVED
131
-----
132 ASSUME CS:CODE
133
134 CASSETTE_IO_1 PROC FAR
135 STI
136 0000 CMP AH,080H ; ENABLE INTERRUPTS
137 0000 FB C1 ; CHECK FOR RANGE
138 0001 80 FC 80 JB C1 ; RETURN IF 00-7FH
139 0004 72 4C CMP AH,0C0H ; CHECK FOR CONFIGURATION PARAMETERS
140 0006 80 FC C0 JE CONF_PARMS
141 0009 74 4F JZ DEV_OPEN
142 000B 80 EC 80 SUB AH,080H ; BASE ON 0
143 000E 74 48 DEC AH ; DEVICE OPEN
144 0010 FE CC JZ DEV_CLOSE
145 0012 74 44 DEC AH ; DEVICE CLOSE
146 0014 FE CC JZ PROG_TERM
147 0016 74 40 DEC AH ; PROGRAM TERMINATION
148 0018 FE CC JZ EVENT_WAIT
149 001A 74 47 DEC AH ; EVENT WAIT
150 001C FE CC JNZ NOT_JOYSTICK
151 001E 75 03 JMP JOY_STICK ; JOYSTICK BIOS
152 0020 E9 01D6 R NOT_JOYSTICK:
153 0023 DEC AH
154 0023 FE CC JZ SYS_REQ ; SYSTEM REQUEST KEY
155 0025 74 31 DEC AH
156 0027 FE CC JZ C1_A ; WAIT
157 0029 74 07 DEC AH
158 002B FE CC JNZ C1_B
159 002D 75 06 JMP BLOCKMOVE ; MOVE BLOCK
160 002F E9 01D6 R C1_A:
161 0032 E9 0175 R JMP WAIT
162 0035 FE CC C1_B:
163 DEC AH
164 0037 75 03 JNZ C1_C
165 0039 E9 03FC R JMP EXT_MEMORY ; GO GET THE EXTENDED MEMORY
166 003C FE CC C1_C:
167 003E 75 03 JNZ C1_D
168 0040 E9 0408 R JMP SET_VMODE ; CHECK FOR FUNCTION 89H
169 0043 80 EC 07 SUB AH,7 ; SWAP TO VIRTUAL MODE
170 0045 75 03 JNZ C1_E ; CHECK FOR FUNCTION 90H
171 0047 E9 0491 R JMP DEVICE_BUSY ; GO IF NOT
172 0048 E9 0491 R C1_D:
173 004B FE CC DEC AH ; CHECK FOR FUNCTION 91H
174 004D 75 03 JNZ C1 ; GO IF NOT
175 004F E9 0495 R C1_E:
176 0052 B4 86 MOV AH,86H ; SET BAD COMMAND
177 0054 F9 STC ; SET CARRY FLAG ON
178 0056 CA 0002 C1_F:
179 0058 CA 0002 RET 2 ; FAR RETURN EXIT FROM ROUTINES
180
181 DEV_OPEN:
182 0058 ; NULL HANDLERS
183
184 DEV_CLOSE:
185
186 PROG_TERM:
187
188 SYS_REQ:
189 0058 JMP C1_F ; RETURN
190 005A EB FB CASSETTE_IO_1 ENDP
191
192 CONF_PARMS PROC NEAR
193 005A PUSH CS ; GET CODE SEGMENT
194 005B 07 POP ES ; PLACE IN SELECTOR POINTER
195 005C BB 0000 E MOV BX,OFFSET CONF_TBL ; GET OFFSET OF PARAMETER TABLE
196 005F 3E E4 XOR AH,AH ; CLEAR AH AND SET CARRY OFF
197 0061 EB F2 JMP C1_F ; EXIT THROUGH COMMON RETURN
198 0063 ENDP CONF_PARMS
199
200 EVENT_WAIT PROC NEAR
201 0063 ASSUME DS:DATA
202 0063 IE PUSH DS ; SAVE
203 0064 EB 0000 E CALL DDS
204 0067 0A C0 OR AL,AL
205 0069 74 08 JZ EVENT_WAIT_2 ; GO IF ZERO
206 006B FE C8 DEC AL ; CHECK IF 1
207 006D 74 45 JZ EVENT_WAIT_3
208 006F 1F POP DS ; RESTORE DATA SEGMENT
209 0070 F9 STC ; SET CARRY
210 0071 EB E2 JMP C1_F ; EXIT
211
212 EVENT_WAIT_2:
213 0073 CLI ; NO INTERRUPTS ALLOWED
214 0074 FA TEST ; CHECK FOR FUNCTION ACTIVE
215 0076 06 00A0 R 01 JZ EVENT_WAIT_1
216 0078 FB STI ; ENABLE INTERRUPTS
217 007C 1F POP DS
218 007D F9 STC ; SET ERROR
219 007E EB D5 JMP C1_F ; RETURN
220
221
222
223
224
225

```

SECTION 5

```

226 0080          EVENT_WAIT_1:
227 0080 E4 A1      IN      AL,INTB01          ; ENSURE INTERRUPT UNMASKED
228 0082 E9 00      JMP     $+2
229 0084 24 FE      AND     AL,0FEH          ; ENABLE PIE
230 0086 E6 A1      OUT     INTB01,AL
231 0088 8C 06 009A R  MOV     @USER_FLAG_SEG,ES  ; SET UP TRANSFER TABLE
232 008C 89 1E 009B R  MOV     @USER_FLAG,BX
233 0090 89 0E 009C R  MOV     @RTC_HIGH,CX
234 0094 89 16 009C R  MOV     @RTC_LOW,DX
235 0098 C6 06 00A0 R 01 MOV     @RTC_WAIT_FLAG,01  ; SET ON FUNCTION ACTIVE SWITCH
236 009D 80 0B      MOV     AL,CMOS_REG_B      ; ENABLE PIE
237 009F E8 0000 E    CALL   CMOS_READ          ; READ CMOS LOCATION
238 00A2 24 7F      AND     AL,07FH          ; CLEAR SET
239 00A4 0C 40      OR     AL,040H          ; ENABLE PIE
240 00A6 50          PUSH    AX
241 00A7 8A E0      MOV     AH,AL            ; PLACE DATA INTO DATA REGISTER
242 00A9 B0 0B      MOV     AL,CMOS_REG_B      ; ADDRESS ALARM REGISTER
243 00AB E8 0000 E    CALL   CMOS_WRITE         ; PLACE DATA IN AH INTO ALARM REGISTER
244 00AE 58          POP     AX
245 00AF 1F          POP     DS
246 00B0 FB      STI
247 00B1 F8      CLC
248 00B2 EB A1      JMP     C1_F              ; CLEAR CARRY
249
250
251
252 00B4          EVENT_WAIT_3:
253 00B4 FA      CLI
254 00B5 F6 06 00A0 R 02 TEST    @RTC_WAIT_FLAG,02H  ; DISABLE INTERRUPTS
255 00BA 74 05      JZ     EVENT_WAIT_4      ; CHECK FOR "WAIT" IN PROGRESS
256                                     ; SKIP TO CANCEL CURRENT "EVENT_WAIT"
257 00BC FB      STI
258 00BD 1F      POP     DS
259 00BE F9      STC
260 00BF EB 94      JMP     C1_F              ; AND SET CARRY FLAG FOR ERROR REQUEST
261                                     ; EXIT
262
263 00C1          EVENT_WAIT_4:
264 00C1 50          AX
265 00C2 B8 0B0B E    MOV     AX,*CMOS_REG_B      ; SAVE (WITH INTERRUPTS DISABLED)
266 00C5 E8 0000 E    CALL   CMOS_READ          ; TURN OFF PIE
267 00C8 24 BF      AND     AL,0BFH          ; GET ALARM REGISTER
268 00CA 86 E0      XCHG   AH,AL            ; CLEAR PIE
269 00CC E8 0000 E    CALL   CMOS_WRITE         ; PLACE INTO WRITE REGISTER
270 00CF 58          POP     AX
271 00D0 C6 06 00A0 R 00 MOV     @RTC_WAIT_FLAG,0  ; WRITE BACK TO ALARM REGISTER
272 00D5 FB      STI
273 00D6 1F      POP     DS
274 00D7 F8      CLC
275 00D8 E9 0065 R    JMP     C1_F              ; RESTORE AH
276                                     ; SET FUNCTION ACTIVE FLAG OFF
277                                     ; ENABLE INTERRUPTS
278                                     ; RESTORE DATA SEGMENT
279                                     ; SET CARRY OFF
280                                     ; RETURN
281
282 00DB          EVENT_WAIT_5: ENDP
283
284 00DB 00          JOY_STICK
285 00DB 00          -----
286 00DB 00          THIS ROUTINE WILL READ THE JOYSTICK PORT
287 00DB 00          -----
288 00DB 00          INPUT
289 00DB 00          (DX)=0 READ THE CURRENT SWITCHES
290 00DB 00          RETURNS (AL)= SWITCH SETTINGS IN BITS 7-4
291 00DB 00          -----
292 00DB 00          (DX)=1 READ THE RESISTIVE INPUTS
293 00DB 00          RETURNS (AX)=A(X) VALUE
294 00DB 00          (BX)=B(Y) VALUE
295 00DB 00          (CX)=B(X) VALUE
296 00DB 00          (DX)=B(Y) VALUE
297 00DB 00          -----
298 00DB 00          CY FLAG ON IF NO ADAPTER CARD OR INVALID CALL
299 00DB 00          -----
300
301 00DB 00          JOY_STICK PROC NEAR
302 00DB 00          STI
303 00DB 00          AX,DX
304 00DB 00          MOV     DX,201H
305 00DB 00          OR     AL,AL
306 00DB 00          JZ     JOY_2
307 00DB 00          DEC     AL
308 00DB 00          JZ     JOY_3
309 00DB 00          JMP     C1
310 00DB 00          ; INTERRUPTS BACK ON
311 00DB 00          ; GET SUB FUNCTION CODE
312 00DB 00          ; ADDRESS OF PORT
313 00DB 00          ; READ SWITCHES
314 00DB 00          ; READ RESISTIVE INPUTS
315 00DB 00          ; GO TO ERROR RETURN
316
317 00DB 00          JOY_1: STI
318 00DB 00          JMP     C1_F
319 00DB 00          ; GO TO COMMON RETURN
320
321 00DB 00          JOY_2: IN AL,DX
322 00DB 00          AND AL,0F0H
323 00DB 00          JMP JOY_1
324 00DB 00          ; STRIP UNWANTED BITS OFF
325 00DB 00          ; FINISHED
326
327 00DB 00          JOY_3: MOV BL,1
328 00DB 00          CALL TEST_CORD
329 00DB 00          PUSH CX
330 00DB 00          MOV BL,2
331 00DB 00          CALL TEST_CORD
332 00DB 00          PUSH CX
333 00DB 00          MOV BL,4
334 00DB 00          CALL TEST_CORD
335 00DB 00          MOV BL,8
336 00DB 00          CALL TEST_CORD
337 00DB 00          MOV DX,CX
338 00DB 00          POP CX
339 00DB 00          POP BX
340 00DB 00          POP AX
341 00DB 00          JMP JOY_1
342 00DB 00          ; SAVE A(X) VALUE
343 00DB 00          ; SAVE A(Y) VALUE
344 00DB 00          ; SAVE B(X) VALUE
345 00DB 00          ; SAVE B(Y) VALUE
346 00DB 00          ; GET B(X) VALUE
347 00DB 00          ; GET A(Y) VALUE
348 00DB 00          ; GET A(X) VALUE
349 00DB 00          ; FINISHED - RETURN
350
351 00DB 00          TEST_CORD PROC NEAR
352 00DB 00          PUSH DX
353 00DB 00          CLD
354 00DB 00          MOV AL,0
355 00DB 00          OUT TIMER+3,AL
356 00DB 00          ; SAVE
357 00DB 00          ; BLOCK INTERRUPTS WHILE READING
358 00DB 00          ; SET UP TO LATCH TIMER 0
359 00DB 00          JMP $+2
360 00DB 00          IN AL,TIMER
361 00DB 00          JMP $+2
362 00DB 00          MOV AH,AL
363 00DB 00          IN AL,TIMER
364 00DB 00          XCHG AH,AL
365 00DB 00          ; READ LOW BYTE OF TIMER 0
366 00DB 00          ; READ HIGH BYTE OF TIMER 0
367 00DB 00          ; REARRANGE TO HIGH,LOW
368
369 00DB 00          TEST_CORD ENDP

```

```

340 0125 50          PUSH    AX          ; SAVE
341 0126 B9 04FF    MOV     CX,4FFH    ; SET COUNT
342 0129 EE          OUT     DX,AL      ; FIRE TIMER
343 012A EB 00      JMP     $+2
344 012C            TEST_CORD_1:
345 012C EC          IN      AL,DX      ; READ VALUES
346 012D 84 C3      TEST   AL,BL      ; HAS PULSE ENDED?
347 012F E0 FB      LOOPNZ TEST_CORD_1
348 0131 83 F9 00   CMP     CX,0
349 0134 59          POP     CX          ; ORIGINAL COUNT
350 0135 76 04      JNZ    SHORT TEST_CORD_2
351 0137 2B C9      SUB     CX,CX      ; SET 0 COUNT FOR RETURN
352 0139 EB 26      JMP     SHORT TEST_CORD_3
353 013B            TEST_CORD_2:
354 013B B0 00      MOV     AL,0       ; SET UP TO LATCH TIMER 0
355 013D E6 43      OUT    TIMER+3,AL
356 013F EB 00      JMP     $+2
357 0141 E4 40      IN     AL,TIMER   ; READ LOW BYTE OF TIMER 0
358 0143 8A E0      MOV    AH,AL
359 0145 E2 00      JMP     $+2
360 0147 E4 40      IN     AL,TIMER   ; READ HIGH BYTE OF TIMER 0
361 0149 86 E0      XCHG  AH,AL      ; REARRANGE TO HIGH,LOW
362
363 014B 3B C8      CMP    CX,AX      ; CHECK FOR COUNTER WRAP
364 014D 73 0B      JAE    TEST_CORD_4
365 014F 52          PUSH   DX          ; GO IF NO
366 0150 BA FFFF    MOV    DX,-1
367
368 0153 2B D0      SUB    DX,AX      ; ADJUST FOR WRAP
369 0155 03 CA      ADD    CX,DX
370 0157 5A          POP    DX
371 0158 EB 02      JMP     SHORT TEST_CORD_5
372
373 015A            TEST_CORD_4:
374 015A 2B C8      SUB    CX,AX
375 015C            TEST_CORD_5:
376 015C 81 E1 1FF0  AND    CX,1FF0H   ; ADJUST
377 0160 C1 E9 04      SHR    CX,4
378
379 0163            TEST_CORD_3:
380 0163 FB          STI
381 0164 BA 0201    MOV    DX,201H   ; INTERRUPTS BACK ON
382 0167 51          PUSH   CX          ; FLUSH OTHER INPUTS
383 0168 50          PUSH   AX
384 0169 B9 04FF    MOV    CX,4FFH   ; COUNT
385 016C            TEST_CORD_6:
386 016C EC          IN     AL,DX
387 016D A8 0F      TEST   AL,0FH
388 016F E0 FB      LOOPNZ TEST_CORD_6
389
390 0171 58          POP    AX
391 0172 59          POP    CX
392 0173 5A          POP    DX          ; SET COUNT
393
394 0174 C3          RET
395
396 0175            TEST_CORD
397 0175            JOY_STICK
398
399 0178            WAIT
400 0178 1E          PROC   NEAR
401 0178 E8 0000 E    PUSH  DS          ; SAVE
402 0179 FA          CALL  DDS
403 017A F6 06 00A0 R 01 TEST   %RTC_WAIT_FLAG,01 ; NO INTERRUPTS ALLOWED
404 017F 74 06      JZ     $+2         ; TEST FOR FUNCTION ACTIVE
405 0181 FB          STI
406 0182 1F          POP    DS          ; ENABLE INTERRUPTS PRIOR TO RETURN
407 0183 F9          STC
408 0184 E9 0055 R  JMP    C1_F        ; SET ERROR
409 0187 E4 A1      IN     AL,INTB01  ; RETURN
410 0189 EB 00      JMP     $+2        ; ENSURE INTERRUPT UNMASKED
411 018B 24 FE      AND    AL,0FEH
412 018D E6 A1      OUT    INTB01,AL
413 018F 8C 1E 009A R MOV    %USER_FLAG_SEG,DS ; SET UP TRANSFER TABLE
414 0193 C7 06 009B R 00A0 R MOV    %USER_FLAG,OFFSET %RTC_WAIT_FLAG
415 0199 89 0E 009E R MOV    %RTC_HIGH,CX
416 019D 89 16 009C R MOV    %RTC_LOW,DX
417 01A1 C6 06 00A0 R 03 MOV    %RTC_WAIT_FLAG,03 ; SET ON "WAIT" FUNCTION ACTIVE SWITCHES
418 01A6 50          PUSH  AX
419 01A7 B8 0B0B    MOV    AX,*CMOS_REG_B ; SAVE (AH)
420 01AA E8 0000 E    CALL  CMOS_READ  ; ENABLE PIE
421 01AD 24 7F      AND    AL,07FH    ; READ ALARM BYTE
422 01AF 0C 40      OR     AL,040H    ; CLEAR S1T BIT
423 01B1 86 E0      XCHG  AH,AL      ; ENABLE PIE BIT
424 01B3 E8 0000 E    CALL  CMOS_WRITE ; DATA TO WORK REGISTER
425 01B6 58          POP    AX         ; WRITE NEW ALARM BYTE
426
427
428
429
430 01B7 FB          STI
431 01B8 51          PUSH  CX          ; ENABLE INTERRUPTS
432 01B9 52          PUSH  DX
433 01BA 87 D1      XCHG  DX,CX      ; SAVE CALLERS PARAMETERS
434 01BC            WAIT_21:
435 01BC F6 06 00A0 R 80 TEST   %RTC_WAIT_FLAG,0B0H ; CHECK FOR END OF WAIT - CLEAR CARRY
436 01C1 E1 F9      LOOPZ WAIT_21    ; DECREMENT TIMEOUT DELAY TILL WAIT END
437 01C3 75 05      JNZ   WAIT_9     ; EXIT IF RTC TIMER WAIT ENDED FLAG SET
438 01C5 83 EA 01   SUB    DX,1
439 01C8 73 F2      JNC   WAIT_2     ; DECREMENT ERROR TIMEOUT COUNTER
440 01CA            WAIT_9:
441 01CA C6 06 00A0 R 00 MOV    %RTC_WAIT_FLAG,0 ; SET FUNCTION INACTIVE
442 01CF 5A          POP    DX
443 01D0 59          POP    CX
444 01D1 1F          POP    DS
445 01D2 F8          CLC
446 01D3 E9 0055 R  JMP    C1_F        ; RESTORE CALLERS PARAMETERS
447
448 01D6            WAIT   ENDP

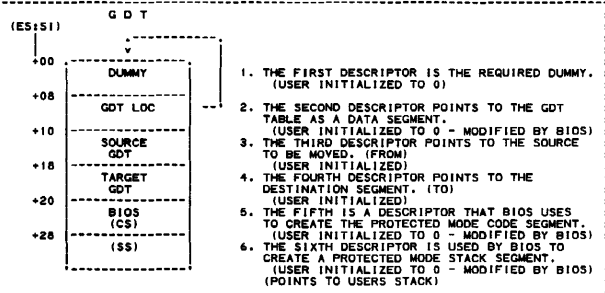
```

SECTION 5

```

449 PAGE
450 INT 15H -- ( FUNCTION 87H - BLOCK MOVE ) -----
451
452 THIS BIOS FUNCTION PROVIDES A MEANS FOR A REAL MODE PROGRAM OR SYSTEM
453 TO TRANSFER A BLOCK OF STORAGE TO AND FROM STORAGE ABOVE THE 1 MEG
454 ADDRESS RANGE IN PROTECTED MODE SPACE BY SWITCHING TO PROTECTED MODE.
455
456 ENTRY:
457 (AH) = 87H (FUNCTION CALL) - BLOCK MOVE.
458 (CX) = WORD COUNT OF STORAGE BLOCK TO BE MOVED.
459 NOTE: MAX COUNT = 8000H FOR 32K WORDS (64K BYTES)
460 ES:SI = LOCATION OF A GDT TABLE BUILT BY ROUTINE USING THIS FUNCTION.
461
462 (ES:SI) POINTS TO A DESCRIPTOR TABLE (GDT) BUILT BEFORE INTERRUPTING
463 TO THIS FUNCTION. THE DESCRIPTORS ARE USED TO PERFORM THE BLOCK
464 MOVE IN THE PROTECTED MODE. THE SOURCE AND TARGET DESCRIPTORS
465 BUILT BY THE USER MUST HAVE A SEGMENT LENGTH = 2 * CX-1 OR GREATER.
466 THE DATA ACCESS RIGHTS BYTE MUST BE SET TO CPL0-R/W (93H). THE
467 24 BIT ADDRESS (BYTE HI, WORD LOW) MUST BE SET TO THE TARGET/SOURCE.
468
469 *** NO INTERRUPTS ARE ALLOWED DURING TRANSFER. LARGE BLOCK MOVES
470 MAY CAUSE LOST INTERRUPTS.
471
472 EXIT:
473 (AH) = 00H IF SUCCESSFUL
474 (AH) = 01H IF MEMORY PARITY (PARITY ERROR REGISTERS ARE CLEARED)
475 (AH) = 02H IF ANY OTHER EXCEPTION INTERRUPT ERROR OCCURRED
476 (AH) = 03H IF GATE ADDRESS LINE 20 FAILED
477 ALL REGISTERS ARE RESTORED EXCEPT (AH).
478
479 IF SUCCESSFUL - CARRY FLAG = 0
480 IF ERROR ----- CARRY FLAG = 1
481
482 DESCRIPTION:
483
484 1. SAVE ENTRY REGISTERS AND SETUP FOR SHUTDOWN EXIT.
485 2. THE REQUIRED ENTRIES ARE BUILT IN THE GDT AT (ES:SI).
486 3. GATE ADDRESS LINE 20 ACTIVE, CLI AND SET SHUTDOWN CODES.
487 4. THE IDTR IS LOADED AND POINTS TO A ROM RESIDENT TABLE.
488 5. THE GDTR IS LOADED FROM THE OFFSET POINTER (ES:SI).
489 6. THE PROCESSOR IS PUT INTO PROTECTED MODE.
490 7. LOAD (DS) AND (ES) WITH SELECTORS FOR THE SOURCE AND TARGET.
491 8. DS:SI (SOURCE) (ES:DI) (TARGET) REP MOVSB IS EXECUTED.
492 9. CHECK MADE FOR PARITY ERRORS.
493 10. REAL MODE RESTORED WHEN SHUTDOWN 09H IS EXECUTED.
494 11. ERRORS ARE CHECKED FOR AND RETURN CODES ARE SET FOR (AH).
495 12. ADDRESS LINE 20 GATE IS DISABLED.
496 13. RETURN WITH REGISTERS RESTORED AND STATUS RETURN CODE.
497 (FOR PC-AT COMPATIBILITY ZF=1 IF SUCCESSFUL, ZF=0 IF ERROR.)
498
499
500 THE FOLLOWING DIAGRAM DEPICTS THE ORGANIZATION OF A BLOCK MOVE GDT.
501
502 G D T
503 (ES:SI)
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562

```



```

531 SOURCE_TARGET_DEF STRUC
532 SEG_LIMIT DW ? ; SEGMENT LIMIT (1-65536 BYTES)
533 LO_WORD DW ? ; 24 BIT SEGMENT PHYSICAL
534 HI_BYTE DB ? ; ADDRESS (0 TO (16M-1))
535 DATA_ACC_RIGHTS DB 93H ; ACCESS RIGHTS BYTE (CPL0-R/W)
536 RESERVED DW 0 ; RESERVED WORD (MUST BE ZERO)
537
538 SOURCE_TARGET_DEF ENDS

```

```

541 THE GLOBAL DESCRIPTOR TABLE (ACTUAL LOCATION POINTED TO BY ES:SI)
542
543 BLOCKMOVE_GDT_DEF STRUC
544
545 0000 ?????????????????? DQ ? ; FIRST DESCRIPTOR NOT ACCESSIBLE
546 0008 ?????????????????? DQ ? ; LOCATION OF CALLING ROUTINE GDT
547 0010 ?????????????????? DQ ? ; SOURCE DESCRIPTOR
548 0018 ?????????????????? DQ ? ; TARGET DESCRIPTOR
549 0020 ?????????????????? DQ ? ; BIOS CODE DESCRIPTOR
550 0028 ?????????????????? DQ ? ; STACK DESCRIPTOR
551 0030
552 BLOCKMOVE_GDT_DEF ENDS

```

```

553 BLOCKMOVE PROC NEAR
554
555 CLD ; SET DIRECTION FORWARD
556 PUSHA ; SAVE GENERAL PURPOSE REGISTERS
557 PUSH ES ; SAVE USERS EXTRA SEGMENT
558 PUSH DS ; SAVE USERS DATA SEGMENT
559
560 I----- SAVE THE CALLING ROUTINE'S STACK
561 CALL DOS ; SET DS TO DATA AREA
562

```

```

563 01DD 8C 16 0069 R      MOV     @10_ROM_SEG,SS      ; SAVE USERS STACK SEGMENT
564 01E1 89 26 0067 R      MOV     @10_ROM_INIT,SP    ; SAVE USERS STACK POINTER
565
566 ;===== SET UP THE PROTECTED MODE DEFINITIONS =====
567 ;----- MAKE A 24 BIT ADDRESS OUT OF THE ES:SI FOR THE GDT POINTER
568
569
570 ASSUME DS:NOTHING      ; POINT (DS) TO USERS CONTROL BLOCK
571 MOV     DS,AX           ; GET THE GDT DATA SEGMENT
572 01E7 BE D8             MOV     DS,AX             ; MOVE THE GDT SEGMENT POINTER TO (DS)
573 01E9 8A F4           MOV     DH,AH            ; BUILD HIGH BYTE OF THE 24 BIT ADDRESS
574 01EB C0 EE 04       SHR     DH,4             ; USE ONLY HIGH NIBBLE SHIFT - RIGHT 4
575 01EE C1 E0 04       SHL     AX,4            ; STRIP HIGH NIBBLE FROM (AX)
576 01F1 03 C6         ADD     AX,SI            ; ADD THE GDT OFFSET TO DEVELOP LOW WORD
577 01F3 80 D6 00     ADC     DH,0            ; ADJUST HIGH BYTE IF CARRY FROM LOW
578
579 ;----- SET THE GDT_LOC
580
581 01F6 C7 44 08 FFFF   MOV     [SI].CGDT_LOC.SEG_LIMIT,MAX_SEG_LEN
582 01FB 89 44 0A       MOV     [SI].CGDT_LOC.BASE_LO_WORD,AX ; SET THE LOW WORD
583 01FE 88 74 0C       MOV     [SI].CGDT_LOC.BASE_HI_BYTE,DH ; SET THE HIGH BYTE
584 0201 C7 44 0E 0000   MOV     [SI].CGDT_LOC.DATA_RESERVED,0 ; RESERVED
585
586 ;----- SET UP THE CODE SEGMENT DESCRIPTOR
587
588 0206 C7 44 20 FFFF   MOV     [SI].BIOS_CS.SEG_LIMIT,MAX_SEG_LEN
589 020B C7 44 22 0000   MOV     [SI].BIOS_CS.BASE_LO_WORD,CSEG@_LO ; LOW WORD OF (CS)= 0
590 0210 C6 44 24 0F    MOV     [SI].BIOS_CS.BASE_HI_BYTE,CSEG@_HI ; HIGH BYTE OF (CS)= 0FH
591 0214 C6 44 26 9B    MOV     [SI].BIOS_CS.DATA_ACC_RIGHTS,CPL0_CODE_ACCESS
592 0218 C7 44 26 0000   MOV     [SI].BIOS_CS.DATA_RESERVED,0 ; RESERVED
593
594 ;----- MAKE A 24 BIT ADDRESS OUT OF THE (SS) - ( SP) REMAINS USER (SP)
595
596 021D 8C D0         MOV     AX,SS           ; GET THE CURRENT STACK SEGMENT
597 021F 8A F4       MOV     DH,AH          ; FORM HIGH BYTE OF 24 BIT ADDRESS
598 0221 C0 EE 04   SHR     DH,4           ; FORM HIGH BYTE - SHIFT RIGHT 4
599 0224 C1 E0 04   SHL     AX,4           ; STRIP HIGH NIBBLE FROM (AX)
600
601 ;----- SS IS NOW IN POSITION FOR A 24 BIT ADDRESS --> SETUP THE (SS) DESCRIPTOR
602
603 0227 C7 44 28 FFFF   MOV     [SI].TEMP_SS.SEG_LIMIT,MAX_SEG_LEN ; SET THE SS SEGMENT LIMIT
604 022C 89 44 2A       MOV     [SI].TEMP_SS.BASE_LO_WORD,AX ; SET THE LOW WORD
605 022F 88 74 2C       MOV     [SI].TEMP_SS.BASE_HI_BYTE,DH ; SET THE HIGH BYTE
606 0232 C6 44 2D 93    MOV     [SI].TEMP_SS.DATA_ACC_RIGHTS,CPL0_DATA_ACCESS ; SET CPL 0
607
608 ;----- GATE ADDRESS BIT 20 ON (DISABLE INTERRUPTS)
609
610 0236 B4 DF         MOV     AH,ENABLE_BIT20 ; GET ENABLE MASK
611 0238 E8 03DA R     CALL   GATE_A20        ; ENABLE A20 AND CLEAR INTERRUPTS
612 023B 3C 00         CMP     AL,0           ; WAS THE COMMAND ACCEPTED?
613 023D 74 06         JZ     BL4             ; GO IF YES
614
615 023F 80 03         MOV     AL,03H        ; SET THE ERROR FLAG IF NOT
616 0241 E6 80         OUT    MFG_PORT,AL    ; EARLY ERROR EXIT
617 0243 EB 51         JMP     SHORT SHUT9
618
619 ;----- SET SHUTDOWN RETURN ADDRESS AND DISABLE NMI
620
621 0245 BL4:          MOV     AX,9*H+CMOS_SHUT_DOWN+NMI ; SET THE SHUTDOWN BYTE LOCATION
622 0248 B8 098F      CALL   CMOS_WRITE     ; TO SHUT DOWN 9 AND DISABLE NMI
623 0248 E8 0000 E
624
625 ;----- CLEAR EXCEPTION ERROR FLAG
626
627 024B 2A C0         SUB    AL,AL          ; SET ERROR FLAG LOCATION TO 0
628 024D E6 80         OUT    MFG_PORT,AL
629
630 ;----- LOAD THE IDT AND GDT
631
632 024F BD 02D4 R     MOV     BP,OFFSET ROM_IDT_LOC ; LOAD THE IDT
633
634 0252 2E +          DB     02EH          ; REGISTER FROM THIS AREA
635 LIDT [BP] +
636 DB     00FH +
637 0254 + 770001 LABEL BYTE ;
638 0254 8B 5E 00 + MOV     BX,WORD PTR [BP] ;
639 0257 + 770002 LABEL BYTE ;
640 0254 + ORG    OFFSET CS:770001 ;
641 0254 01 + DB     001H          ;
642 0257 + ORG    OFFSET CS:770002 ;
643
644 LGDT [SI].CGDT_LOC ; LOAD GLOBAL DESCRIPTOR TABLE REGISTER
645 0257 0F + DB     00FH          ;
646 0258 + 770003 LABEL BYTE ;
647 0258 8B 54 08 + MOV     DX,WORD PTR [SI].CGDT_LOC ;
648 0258 + 770004 LABEL BYTE ;
649 0258 + ORG    OFFSET CS:770003 ;
650 0258 01 + DB     001H          ;
651 0258 + ORG    OFFSET CS:770004 ;
652
653 ;----- SWITCH TO VIRTUAL MODE
654
655 025B B8 0001      MOV     AX,VIRTUAL_ENABLE ; MACHINE STATUS WORD NEEDED TO
656 LMSW AX ; SWITCH TO VIRTUAL MODE
657
658 025E 0F 01 F0 + DB     00FH,001H,0F0H ; PURGE PRE-FETCH QUEUE WITH FAR JUMP
659 0261 EA         DB     0EAH           ;
660 0262 0256 R     DW     OFFSET VIRT ;
661 0264 0020      DW     BIOS_CS       ; - IN SEGMENT -PROTECTED MODE SELECTOR
662 0266
663
664 ;----- IN PROTECTED MODE - SETUP STACK SELECTOR AND SOURCE/TARGET SELECTORS
665
666 0266 B8 0028      MOV     AX,TEMP_SS      ; USER'S SS+SP IS NOT A DESCRIPTOR
667 0269 BE D0       MOV     SS,AX           ; LOAD STACK SELECTOR
668 026B B8 0010      MOV     AX,SOURCE      ; GET THE SOURCE ENTRY
669 026E BE D8       MOV     DS,AX           ; LOAD SOURCE SELECTOR
670 0270 BB 0018      MOV     AX,TARGET      ; GET THE TARGET ENTRY
671 0273 BE C0       MOV     ES,AX           ; LOAD TARGET SELECTOR
672 0275 2B F6       SUB     SI,SI           ; SET SOURCE INDEX REGISTER TO ZERO
673 0277 2B FF       SUB     DI,DI           ; SET TARGET INDEX REGISTER TO ZERO
674
675 0279 F3 / A5     REP    MOVSW           ; MOVE THE BLOCK COUNT PASSED IN (CX)
676

```

SECTION 5

```

677                                     ;----- CHECK FOR MEMORY PARITY BEFORE SHUTDOWN
678
679 027B E4 61                          IN     AL,PORT_B           ; GET THE PARITY LATCHES
680 027D 24 C0                          AND    AL,PARITY_ERR     ; STRIP UNWANTED BITS
681 027F 74 12                          JZ     DONE1             ; GO IF NO PARITY ERROR
682
683                                     ;----- CLEAR PARITY BEFORE SHUTDOWN
684
685 0281 8B 05                          MOV    AX,DS:[DI]        ; FETCH CURRENT SOURCE DATA
686 0283 89 05                          MOV    DS:[DI],AX       ; WRITE IT BACK
687 0285 B0 01                          MOV    AL,01            ; SET PARITY CHECK ERROR = 01
688 0287 E6 80                          OUT    MFG_PORT,AL      ;
689 0289 E4 61                          IN     AL,PORT_B        ;
690 028B 0C 0C                          OR     AL,RAM_PAR_OFF   ; TOGGLE PARITY CHECK LATCHES
691 028D E6 61                          OUT    PORT_B,AL        ; TO CLEAR THE PENDING ERROR
692 028F 24 F3                          AND    AL,RAM_PAR_ON   ; AND ENABLE CHECKING
693 0291 E6 61                          OUT    PORT_B,AL        ;
694
695                                     ;----- CAUSE A SHUTDOWN
696
697 0293                                 DONE1:
698 0295 E9 0000 E                        JMP    PROC_SHUTDOWN    ; GO RESET PROCESSOR AND SHUTDOWN
699
700                                     ;----- RETURN FROM SHUTDOWN
701
702                                     ;-----
703
704 0296                                 SHUT9:
705                                     ; RESTORE USERS STACK
706 0298 B8 ---- R                       ASSUME DS:DATA          ;
707 0299 8E DB                           MOV    AX,DATA          ; SET DS TO DATA AREA
708 029B 8E 16 0069 R                    MOV    DS,AX            ;
709 029F 8B E6 0067 R                    MOV    SS,@10 ROM_SEG  ; GET USER STACK SEGMENT
710                                     ; GET USER STACK POINTER
711
712                                     ;----- GATE ADDRESS BIT 20 OFF
713 02A3 B4 DD                           MOV    AH,DISABLE_BIT20 ; DISABLE MASK
714 02A5 E8 03DA R                       CALL   GATE_A20         ; GATE ADDRESS 20 LINE OFF
715 02A8 3C 00                           CMP    AL,0             ; COMMAND ACCEPTED?
716 02AA 74 0A                           JZ     DONES            ; GO IF YES
717
718 02AC E4 80                          IN     AL,MFG_PORT      ; CHECK FOR ANY OTHER ERROR FIRST
719 02AE 3C 00                           CMP    AL,0             ; WAS THERE AN ERROR?
720 02B0 75 04                           JNZ   DONES            ; REPORT FIRST ERROR IF YES
721 02B2 B0 03                          MOV    AL,03H          ; ELSE SET GATE A20 ERROR FLAG
722 02B4 E6 80                          OUT    MFG_PORT,AL      ;
723
724                                     ;----- RESTORE THE USERS REGISTERS AND SET RETURN CODES
725
726 02B6                                 DONE3:
727 02B8 B8 000F                          MOV    AX,CMOS_SHUT_DOWN ; CLEAR (AH) TO ZERO AND (AL) TO DEFAULT
728 02BA E6 70                          OUT    CMOS_PORT,AL    ; ENABLE NMI INTERRUPTS
729 02BC 1F                              POP    DS               ; RESTORE USER DATA SEGMENT
730 02BE 07                              POP    ES               ; RESTORE USER EXTRA SEGMENT
731 02C0 E4 71                          IN     AL,CMOS_DATA     ; OPEN CMOS STANDBY LATCH
732
733 02C2 E4 80                          IN     AL,MFG_PORT      ; GET THE ENDING STATUS RETURN CODE
734 02C4 8B EC                          MOV    BP,SP           ; POINT TO REGISTERS IN THE STACK
735 02C6 88 46 0F                        MOV    [BP+15],AL      ; PLACE ERROR CODE INTO STACK AT (AH)
736 02C8 3A E0                          CMP    AH,AL            ; SET THE ZF & CY FLAGS WITH RETURN CODE
737 02CA 61                              POPA                    ; RESTORE THE GENERAL PURPOSE REGISTERS
738 02CC 9F                              STI                     ; TURN INTERRUPTS ON
739 02CE CA 0002                          DONE4: FAR             ; RETURN WITH FLAGS SET -- (AH)= CODE
740 02D0 2C 00                          DONE4: ENDP            ; (CF=0,ZF=1)= OK (CF=1,ZF=0)= ERROR
741
742                                     ;----- BLOCK MOVE EXCEPTION INTERRUPT HANDLER
743
744                                     EX_INT:
745 02D2 CD                                MOV    AL,02H          ; GET EXCEPTION ERROR CODE
746 02D4 B0 02                          OUT    MFG_PORT,AL     ; SET EXCEPTION INTERRUPT OCCURRED FLAG
747 02D6 CF E6 80                          JMP    PROC_SHUTDOWN   ; CAUSE A EARLY SHUTDOWN
748 02D8 D1 E9 0000 E
749
750                                     ;----- ROM IDT LOCATION
751
752 02DA                                 ROM_IDT_LOC:
753 02DC 0100                            DW    ROM_IDT_END-ROM_IDT ; LENGTH OF ROM IDT TABLE
754 02DE 02DA R                          DW    ROM_IDT          ; LOW WORD OF BASE ADDRESS
755 02E0 0F                              DB    CSEG@_HI         ; HIGH BYTE OF BASE ADDRESS
756 02E2 00                              DB    0                ; RESERVED
757
758                                     ;----- THE ROM EXCEPTION INTERRUPT VECTOR GATES FOR BLOCK MOVE
759
760 02E4                                 ROM_IDT:
761 02E6 02DA 02CD R                      DW    EX_INT           ; EXCEPTION 00
762 02E8 02D0 0020                       DW    BIOS_CS         ; DESTINATION OFFSET
763 02EA 00                              DB    0                ; DESTINATION SEGMENT SELECTOR
764 02EC 87                              DB    0                ; WORD COPY COUNT
765 02EE 0000                             DB    TRAP_GATE       ; GATE TYPE - ACCESS RIGHTS BYTE
766 02F0 00                              DW    0                ; RESERVED
767 02F2 02E2 02CD R                      DW    EX_INT           ; EXCEPTION 01
768 02F4 02D0 0020                       DW    BIOS_CS         ; DESTINATION OFFSET
769 02F6 00                              DB    0                ; DESTINATION SEGMENT SELECTOR
770 02F8 87                              DB    0                ; WORD COPY COUNT
771 02FA 0000                             DB    TRAP_GATE       ; GATE TYPE - ACCESS RIGHTS BYTE
772 02FC 00                              DW    0                ; RESERVED
773 02FE 02EA 02CD R                      DW    EX_INT           ; EXCEPTION 02
774 0300 02D0 0020                       DW    BIOS_CS         ; DESTINATION OFFSET
775 0302 00                              DB    0                ; DESTINATION SEGMENT SELECTOR
776 0304 87                              DB    0                ; WORD COPY COUNT
777 0306 87                              DB    TRAP_GATE       ; GATE TYPE - ACCESS RIGHTS BYTE
778 0308 00                              DW    0                ; RESERVED
779 030A 02F2 02CD R                      DW    EX_INT           ; EXCEPTION 03
780 030C 02D0 0020                       DW    BIOS_CS         ; DESTINATION OFFSET
781 030E 00                              DB    0                ; DESTINATION SEGMENT SELECTOR
782 0310 87                              DB    0                ; WORD COPY COUNT
783 0312 87                              DB    TRAP_GATE       ; GATE TYPE - ACCESS RIGHTS BYTE
784 0314 00                              DW    0                ; RESERVED
785 0316 02FA 02CD R                      DW    EX_INT           ; EXCEPTION 04
786 0318 02D0 0020                       DW    BIOS_CS         ; DESTINATION OFFSET
787 031A 00                              DB    0                ; DESTINATION SEGMENT SELECTOR
788 031C 87                              DB    0                ; WORD COPY COUNT
789 031E 87                              DB    TRAP_GATE       ; GATE TYPE - ACCESS RIGHTS BYTE
790 0320 0000                             DW    0                ; RESERVED
791                                     ; EXCEPTION 05

```

791	0302 02CD R	DW	EX_INT	DESTINATION OFFSET
792	0304 0020	DW	BIOS_CS	DESTINATION SEGMENT SELECTOR
793	0306 00	DB	0	WORD COPY COUNT
794	0307 87	DB	TRAP_GATE	GATE TYPE - ACCESS RIGHTS BYTE
795	0308 0000	DW	0	RESERVED
796				EXCEPTION 06
797	030A 02CD R	DW	EX_INT	DESTINATION OFFSET
798	030C 0020	DW	BIOS_CS	DESTINATION SEGMENT SELECTOR
799	030E 00	DB	0	WORD COPY COUNT
800	030F 87	DB	TRAP_GATE	GATE TYPE - ACCESS RIGHTS BYTE
801	0310 0000	DW	0	RESERVED
802				EXCEPTION 07
803	0312 02CD R	DW	EX_INT	DESTINATION OFFSET
804	0314 0020	DW	BIOS_CS	DESTINATION SEGMENT SELECTOR
805	0316 00	DB	0	WORD COPY COUNT
806	0317 87	DB	TRAP_GATE	GATE TYPE - ACCESS RIGHTS BYTE
807	0318 0000	DW	0	RESERVED
808				EXCEPTION 08
809	031A 02CD R	DW	EX_INT	DESTINATION OFFSET
810	031C 0020	DW	BIOS_CS	DESTINATION SEGMENT SELECTOR
811	031E 00	DB	0	WORD COPY COUNT
812	031F 87	DB	TRAP_GATE	GATE TYPE - ACCESS RIGHTS BYTE
813	0320 0000	DW	0	RESERVED
814				EXCEPTION 09
815	0322 02CD R	DW	EX_INT	DESTINATION OFFSET
816	0324 0020	DW	BIOS_CS	DESTINATION SEGMENT SELECTOR
817	0326 00	DB	0	WORD COPY COUNT
818	0327 87	DB	TRAP_GATE	GATE TYPE - ACCESS RIGHTS BYTE
819	0328 0000	DW	0	RESERVED
820				EXCEPTION 10
821	032A 02CD R	DW	EX_INT	DESTINATION OFFSET
822	032C 0020	DW	BIOS_CS	DESTINATION SEGMENT SELECTOR
823	032E 00	DB	0	WORD COPY COUNT
824	032F 87	DB	TRAP_GATE	GATE TYPE - ACCESS RIGHTS BYTE
825	0330 0000	DW	0	RESERVED
826				EXCEPTION 11
827	0332 02CD R	DW	EX_INT	DESTINATION OFFSET
828	0334 0020	DW	BIOS_CS	DESTINATION SEGMENT SELECTOR
829	0336 00	DB	0	WORD COPY COUNT
830	0337 87	DB	TRAP_GATE	GATE TYPE - ACCESS RIGHTS BYTE
831	0338 0000	DW	0	RESERVED
832				EXCEPTION 12
833	033A 02CD R	DW	EX_INT	DESTINATION OFFSET
834	033C 0020	DW	BIOS_CS	DESTINATION SEGMENT SELECTOR
835	033E 00	DB	0	WORD COPY COUNT
836	033F 87	DB	TRAP_GATE	GATE TYPE - ACCESS RIGHTS BYTE
837	0340 0000	DW	0	RESERVED
838				EXCEPTION 13
839	0342 02CD R	DW	EX_INT	DESTINATION OFFSET
840	0344 0020	DW	BIOS_CS	DESTINATION SEGMENT SELECTOR
841	0346 00	DB	0	WORD COPY COUNT
842	0347 87	DB	TRAP_GATE	GATE TYPE - ACCESS RIGHTS BYTE
843	0348 0000	DW	0	RESERVED
844				EXCEPTION 14
845	034A 02CD R	DW	EX_INT	DESTINATION OFFSET
846	034C 0020	DW	BIOS_CS	DESTINATION SEGMENT SELECTOR
847	034E 00	DB	0	WORD COPY COUNT
848	034F 87	DB	TRAP_GATE	GATE TYPE - ACCESS RIGHTS BYTE
849	0350 0000	DW	0	RESERVED
850				EXCEPTION 15
851	0352 02CD R	DW	EX_INT	DESTINATION OFFSET
852	0354 0020	DW	BIOS_CS	DESTINATION SEGMENT SELECTOR
853	0356 00	DB	0	WORD COPY COUNT
854	0357 87	DB	TRAP_GATE	GATE TYPE - ACCESS RIGHTS BYTE
855	0358 0000	DW	0	RESERVED
856				EXCEPTION 16
857	035A 02CD R	DW	EX_INT	DESTINATION OFFSET
858	035C 0020	DW	BIOS_CS	DESTINATION SEGMENT SELECTOR
859	035E 00	DB	0	WORD COPY COUNT
860	035F 87	DB	TRAP_GATE	GATE TYPE - ACCESS RIGHTS BYTE
861	0360 0000	DW	0	RESERVED
862				EXCEPTION 17
863	0362 02CD R	DW	EX_INT	DESTINATION OFFSET
864	0364 0020	DW	BIOS_CS	DESTINATION SEGMENT SELECTOR
865	0366 00	DB	0	WORD COPY COUNT
866	0367 87	DB	TRAP_GATE	GATE TYPE - ACCESS RIGHTS BYTE
867	0368 0000	DW	0	RESERVED
868				EXCEPTION 18
869	036A 02CD R	DW	EX_INT	DESTINATION OFFSET
870	036C 0020	DW	BIOS_CS	DESTINATION SEGMENT SELECTOR
871	036E 00	DB	0	WORD COPY COUNT
872	036F 87	DB	TRAP_GATE	GATE TYPE - ACCESS RIGHTS BYTE
873	0370 0000	DW	0	RESERVED
874				EXCEPTION 19
875	0372 02CD R	DW	EX_INT	DESTINATION OFFSET
876	0374 0020	DW	BIOS_CS	DESTINATION SEGMENT SELECTOR
877	0376 00	DB	0	WORD COPY COUNT
878	0377 87	DB	TRAP_GATE	GATE TYPE - ACCESS RIGHTS BYTE
879	0378 0000	DW	0	RESERVED
880				EXCEPTION 20
881	037A 02CD R	DW	EX_INT	DESTINATION OFFSET
882	037C 0020	DW	BIOS_CS	DESTINATION SEGMENT SELECTOR
883	037E 00	DB	0	WORD COPY COUNT
884	037F 87	DB	TRAP_GATE	GATE TYPE - ACCESS RIGHTS BYTE
885	0380 0000	DW	0	RESERVED
886				EXCEPTION 21
887	0382 02CD R	DW	EX_INT	DESTINATION OFFSET
888	0384 0020	DW	BIOS_CS	DESTINATION SEGMENT SELECTOR
889	0386 00	DB	0	WORD COPY COUNT
890	0387 87	DB	TRAP_GATE	GATE TYPE - ACCESS RIGHTS BYTE
891	0388 0000	DW	0	RESERVED
892				EXCEPTION 22
893	038A 02CD R	DW	EX_INT	DESTINATION OFFSET
894	038C 0020	DW	BIOS_CS	DESTINATION SEGMENT SELECTOR
895	038E 00	DB	0	WORD COPY COUNT
896	038F 87	DB	TRAP_GATE	GATE TYPE - ACCESS RIGHTS BYTE
897	0390 0000	DW	0	RESERVED
898				EXCEPTION 23
899	0392 02CD R	DW	EX_INT	DESTINATION OFFSET
900	0394 0020	DW	BIOS_CS	DESTINATION SEGMENT SELECTOR
901	0396 00	DB	0	WORD COPY COUNT
902	0397 87	DB	TRAP_GATE	GATE TYPE - ACCESS RIGHTS BYTE
903	0398 0000	DW	0	RESERVED
904				EXCEPTION 24

SECTION 5

```

905 039A 02CD R      DW      EX_INT      ; DESTINATION OFFSET
906 039C 0020      DW      BIOS_CS     ; DESTINATION SEGMENT SELECTOR
907 039E 00        DB      0             ; WORD COPY COUNT
908 039F 87        DB      TRAP_GATE    ; GATE TYPE - ACCESS RIGHTS BYTE
909 03A0 0000      DW      0             ; RESERVED
910                ; EXCEPTION 25
911 03A2 02CD R      DW      EX_INT      ; DESTINATION OFFSET
912 03A4 0020      DW      BIOS_CS     ; DESTINATION SEGMENT SELECTOR
913 03A6 00        DB      0             ; WORD COPY COUNT
914 03A7 87        DB      TRAP_GATE    ; GATE TYPE - ACCESS RIGHTS BYTE
915 03A8 0000      DW      0             ; RESERVED
916                ; EXCEPTION 26
917 03AA 02CD R      DW      EX_INT      ; DESTINATION OFFSET
918 03AC 0020      DW      BIOS_CS     ; DESTINATION SEGMENT SELECTOR
919 03AE 00        DB      0             ; WORD COPY COUNT
920 03AF 87        DB      TRAP_GATE    ; GATE TYPE - ACCESS RIGHTS BYTE
921 03B0 0000      DW      0             ; RESERVED
922                ; EXCEPTION 27
923 03B2 02CD R      DW      EX_INT      ; DESTINATION OFFSET
924 03B4 0020      DW      BIOS_CS     ; DESTINATION SEGMENT SELECTOR
925 03B6 00        DB      0             ; WORD COPY COUNT
926 03B7 87        DB      TRAP_GATE    ; GATE TYPE - ACCESS RIGHTS BYTE
927 03B8 0000      DW      0             ; RESERVED
928                ; EXCEPTION 28
929 03BA 02CD R      DW      EX_INT      ; DESTINATION OFFSET
930 03BC 0020      DW      BIOS_CS     ; DESTINATION SEGMENT SELECTOR
931 03BE 00        DB      0             ; WORD COPY COUNT
932 03BF 87        DB      TRAP_GATE    ; GATE TYPE - ACCESS RIGHTS BYTE
933 03C0 0000      DW      0             ; RESERVED
934                ; EXCEPTION 29
935 03C2 02CD R      DW      EX_INT      ; DESTINATION OFFSET
936 03C4 0020      DW      BIOS_CS     ; DESTINATION SEGMENT SELECTOR
937 03C6 00        DB      0             ; WORD COPY COUNT
938 03C7 87        DB      TRAP_GATE    ; GATE TYPE - ACCESS RIGHTS BYTE
939 03C8 0000      DW      0             ; RESERVED
940                ; EXCEPTION 30
941 03CA 02CD R      DW      EX_INT      ; DESTINATION OFFSET
942 03CC 0020      DW      BIOS_CS     ; DESTINATION SEGMENT SELECTOR
943 03CE 00        DB      0             ; WORD COPY COUNT
944 03CF 87        DB      TRAP_GATE    ; GATE TYPE - ACCESS RIGHTS BYTE
945 03D0 0000      DW      0             ; RESERVED
946                ; EXCEPTION 31
947 03D2 02CD R      DW      EX_INT      ; DESTINATION OFFSET
948 03D4 0020      DW      BIOS_CS     ; DESTINATION SEGMENT SELECTOR
949 03D6 00        DB      0             ; WORD COPY COUNT
950 03D7 87        DB      TRAP_GATE    ; GATE TYPE - ACCESS RIGHTS BYTE
951 03D8 0000      DW      0             ; RESERVED
952 03DA                ROM_IDT_END;
953
954 03DA                BLOCKMOVE  ENDP
  
```



```

955 PAGE
956 -----
957 ; GATE_A20
958 ; THIS ROUTINE CONTROLS A SIGNAL WHICH GATES ADDRESS BIT 20.
959 ; THE GATE A20 SIGNAL IS AN OUTPUT OF THE 8042 SLAVE PROCESSOR.
960 ; ADDRESS BIT 20 SHOULD BE GATED ON BEFORE ENTERING PROTECTED MODE.
961 ; IT SHOULD BE GATED OFF AFTER ENTERING REAL MODE FROM PROTECTED
962 ; MODE. INTERRUPTS ARE LEFT DISABLED ON EXIT.
963 ; INPUT
964 ; (AH)= DPH ADDRESS BIT 20 GATE OFF. (A20 ALWAYS ZERO)
965 ; (AH)= DPH ADDRESS BIT 20 GATE ON. (A20 CONTROLLED BY 80286)
966 ; OUTPUT
967 ; (AL)= 00H OPERATION SUCCESSFUL. 8042 HAS ACCEPTED COMMAND.
968 ; (AL)= 02H FAILURE--8042 UNABLE TO ACCEPT COMMAND.
969 -----
970 GATE_A20 PROC
971 03DA PUSH CX ; SAVE USERS (CX)
972 03DB CLI ; DISABLE INTERRUPTS WHILE USING 8042
973 03DC EB 03F3 R ; INSURE 8042 INPUT BUFFER EMPTY
974 03DF 75 10 ; EXIT IF 8042 UNABLE TO ACCEPT COMMAND
975 03E1 80 D1 ; 8042 COMMAND TO WRITE OUTPUT PORT
976 03E3 E6 64 ; OUT STATUS_PORT,AL ; OUTPUT COMMAND TO 8042
977 03E5 EB 03F3 R ; CALL EMPTY_8042 ; WAIT FOR 8042 TO ACCEPT COMMAND
978 03E8 75 07 ; JNZ GATE_A20_RETURN ; EXIT IF 8042 UNABLE TO ACCEPT COMMAND
979 03EA 8A C4 ; MOV AL,AH ; 8042 PORT DATA
980 03EC E6 60 ; OUT PORT_A,AL ; OUTPUT PORT DATA TO 8042
981 03EE EB 03F3 R ; CALL EMPTY_8042 ; WAIT FOR 8042 TO ACCEPT PORT DATA
982 ;----- 8042 OUTPUT WILL SWITCH WITHIN 20 MICRO SECONDS OF ACCEPTING PORT DATA
983
984 GATE_A20_RETURN:
985 03F1 POP CX ; RESTORE USERS (CX)
986 03F1 59
987 03F2 C3
988
989 ; EMPTY_8042
990 ; THIS ROUTINE WAITS FOR THE 8042 INPUT BUFFER TO EMPTY.
991 ; INPUT
992 ; NONE
993 ; OUTPUT
994 ; (AL)= 00H 8042 INPUT BUFFER EMPTY (ZERO FLAG SET)
995 ; (AL)= 02H TIME OUT, 8042 INPUT BUFFER FULL (NON-ZERO FLAG SET)
996 ; (CX) - MODIFIED
997 -----
998 EMPTY_8042:
999 03F3 SUB CX,CX ; (CX)=0, WILL BE USED AS TIME OUT VALUE
1000 03F5
1001 03F5 E4 64 ; IN AL,STATUS_PORT ; READ 8042 STATUS PORT
1002 03F7 24 02 ; AND AL,INPT_BUF_FULL ; TEST INPUT BUFFER FULL FLAG (BIT 1)
1003 03F9 E0 FA ; CALL LOPNZ EMPTY_L ; LOOP UNTIL BUFFER EMPTY OR TIME OUT
1004 03FB C3
1005 03FC
1006
1007 ;--- INT 15 H -- ( FUNCTION 80 H - I/O MEMORY SIZE DETERMINE ) -----
1008 ; EXT_MEMORY
1009 ; THIS ROUTINE RETURNS THE AMOUNT OF MEMORY IN THE SYSTEM THAT IS
1010 ; LOCATED STARTING AT THE 1024K ADDRESSING RANGE, AS DETERMINED BY
1011 ; THE POST ROUTINES.
1012 ; NOTE THAT THE SYSTEM MAY NOT BE ABLE TO USE I/O MEMORY UNLESS THERE
1013 ; IS A FULL COMPLEMENT OF 512K OR 640 BYTES ON THE PLANAR. THIS SIZE
1014 ; SIZE IS STORED IN CMOS AT ADDRESS LOCATIONS 30H AND 31H.
1015 ; INPUT
1016 ; AH = 80H
1017 ;
1018 ; THE I/O MEMORY SIZE VARIABLE IS SET DURING POWER ON
1019 ; DIAGNOSTICS ACCORDING TO THE FOLLOWING ASSUMPTIONS:
1020 ;
1021 ; 1. ALL INSTALLED MEMORY IS FUNCTIONAL.
1022 ; 2. ALL MEMORY FROM 0 TO 640K MUST BE CONTIGUOUS.
1023 ;
1024 ; OUTPUT
1025 ; (AX) = NUMBER OF CONTIGUOUS 1K BLOCKS OF MEMORY A
1026 ; AVAILABLE STARTING AT ADDRESS 1024K.
1027 ;
1028 ;-----
1029
1030 EXT_MEMORY PROC
1031 03FC MOV AX,CMOS_U_M_S_LO*H+CMOS_U_M_S_HI ; ADDRESS HIGH/LOW BYTES
1032 03FE
1033 03FC BB 3031 ; CALL CMOS_READ ; GET THE HIGH BYTE OF I/O MEMORY
1034 03FF EB 0000 E ; XCHG AL,AX ; PUT HIGH BYTE IN POSITION (AH)
1035 0402 B6 C4 ; CALL CMOS_READ ; GET THE LOW BYTE OF I/O MEMORY
1036 0404 EB 0000 E ; IRET ; RETURN TO USER
1037 0407 CF
1038
1039 0408 EXT_MEMORY ENDP

```

SECTION 5

```

1040 PAGE
1041 ---- INT 15 H ( FUNCTION 89 H ) -----
1042 |
1043 | PURPOSE:
1044 | THIS BIOS FUNCTION PROVIDES A MEANS TO THE USER TO SWITCH INTO
1045 | VIRTUAL (PROTECTED) MODE. UPON COMPLETION OF THIS FUNCTION THE
1046 | PROCESSOR WILL BE IN VIRTUAL (PROTECTED) MODE AND CONTROL WILL
1047 | BE TRANSFERRED TO THE CODE SEGMENT THAT WAS SPECIFIED BY THE USER.
1048 |
1049 | ENTRY REQUIREMENTS:
1050 | (ES:SI) POINTS TO A DESCRIPTOR TABLE (GDT) BUILT BEFORE INTERRUPTING
1051 | TO THIS FUNCTION. THESE DESCRIPTORS ARE USED BY THIS FUNCTION TO
1052 | INITIALIZE THE IDTR, THE GDTR AND THE STACK SEGMENT SELECTOR. THE
1053 | DATA SEGMENT (DS) SELECTOR AND THE EXTRA SEGMENT (ES) SELECTOR WILL
1054 | BE INITIALIZE TO DESCRIPTORS BUILT BY THE ROUTINE USING THIS FUNCTION.
1055 | BH - OFFSET INTO THE INTERRUPT DESCRIPTOR TABLE STATING WHERE THE
1056 | FIRST EIGHT HARDWARE INTERRUPTS WILL BEGIN. ( INTERRUPT LEVEL 1 )
1057 | BL - OFFSET INTO THE INTERRUPT DESCRIPTOR TABLE STATING WHERE THE
1058 | SECOND EIGHT HARDWARE INTERRUPTS BEGIN. ( INTERRUPT LEVEL 2 )
1059 |
1060 | THE DESCRIPTORS ARE DEFINED AS FOLLOWS:
1061 |
1062 | 1. THE FIRST DESCRIPTOR IS THE REQUIRED DUMMY.
1063 | (USER INITIALIZED TO 0)
1064 | 2. THE SECOND DESCRIPTOR POINTS TO THE GDT TABLE AS
1065 | A DATA SEGMENT.
1066 | (USER INITIALIZED)
1067 | 3. THE THIRD DESCRIPTOR POINTS TO THE USER DEFINED
1068 | INTERRUPT DESCRIPTOR TABLE (IDT).
1069 | (USER INITIALIZED)
1070 | 4. THE FORTH DESCRIPTOR POINTS TO THE USER'S DATA
1071 | SEGMENT (DS).
1072 | (USER INITIALIZED)
1073 | 5. THE FIFTH DESCRIPTOR POINTS TO THE USER'S EXTRA
1074 | SEGMENT (ES).
1075 | (USER INITIALIZED)
1076 | 6. THE SIXTH DESCRIPTOR POINTS TO THE USER'S STACK
1077 | SEGMENT (SS).
1078 | (USER INITIALIZED)
1079 | 7. THE SEVENTH DESCRIPTOR POINTS TO THE CODE SEGMENT
1080 | THAT THIS FUNCTION WILL RETURN TO.
1081 | (USER INITIALIZED TO THE USER'S CODE SEGMENT.)
1082 | 8. THE EIGHTH DESCRIPTOR IS USED BY THIS FUNCTION TO
1083 | ESTABLISH A CODE SEGMENT FOR ITSELF. THIS IS
1084 | NEEDED SO THAT THIS FUNCTION CAN COMPLETE IT'S
1085 | EXECUTION WHILE IN PROTECTED MODE. WHEN CONTROL
1086 | GETS PASSED TO THE USER'S CODE THIS DESCRIPTOR CAN
1087 | BE USED BY HIM IN ANY WAY HE CHOOSES.
1088 |
1089 | NOTE - EACH DESCRIPTOR MUST CONTAIN ALL THE NECESSARY DATA
1090 | I.E. THE LIMIT, BASE ADDRESS AND THE ACCESS RIGHTS BYTE.
1091 |
1092 | AH= 89H (FUNCTION CALL)
1093 | ES:SI = LOCATION OF THE GDT TABLE BUILT BY ROUTINE
1094 | USING THIS FUNCTION.
1095 |
1096 | EXIT PARAMETERS:
1097 |
1098 | AH = 0 IF SUCCESSFUL
1099 | ALL SEGMENT REGISTERS ARE CHANGED, (AX) AND (BP) DESTROYED
1100 |
1101 | CONSIDERATIONS:
1102 |
1103 | 1. NO BIOS AVAILABLE TO USER. USER MUST HANDLE ALL
1104 | I/O COMMANDS.
1105 | 2. INTERRUPTS - INTERRUPT VECTOR LOCATIONS MUST BE
1106 | MOVED, DUE TO THE 286 RESERVED AREAS. THE
1107 | HARDWARE INTERRUPT CONTROLLERS MUST BE REINITIALIZED
1108 | TO DEFINE LOCATIONS THAT DO NOT RESIDE IN THE 286
1109 | RESERVED AREAS.
1110 | 3. EXCEPTION INTERRUPT TABLE AND HANDLER MUST BE
1111 | INITIALIZED BY THE USER.
1112 | 4. THE INTERRUPT DESCRIPTOR TABLE MUST NOT OVERLAP
1113 | THE REAL MODE BIOS INTERRUPT DESCRIPTOR TABLE.
1114 | 5. THE FOLLOWING GIVES AN IDEA OF WHAT THE USER CODE
1115 | SHOULD LOOK LIKE WHEN INVOKING THIS FUNCTION.
1116 |
1117 | REAL MODE ---> "USER CODE"
1118 | " MOV AX,GDT SEGMENT
1119 | " MOV ES,AX
1120 | " MOV SI,GDT OFFSET
1121 | " MOV BH,HARDWARE INT LEVEL 1 OFFSET
1122 | " MOV BL,HARDWARE INT LEVEL 2 OFFSET
1123 | " MOV AH,89H
1124 | " INT 15H
1125 | VIRTUAL MODE ---> "USER CODE"
1126 |
1127 | DESCRIPTION:
1128 |
1129 | 1. CLI (NO INTERRUPTS ALLOWED) WHILE THIS FUNCTION IS EXECUTING.
1130 | 2. ADDRESS LINE 20 IS GATED ACTIVE.
1131 | 3. THE CURRENT USER STACK SEGMENT DESCRIPTOR IS INITIALIZED.
1132 | 4. THE GDTR IS LOADED WITH THE GDT BASE ADDRESS.
1133 | 5. THE IDTR IS LOADED WITH THE IDT BASE ADDRESS.
1134 | 6. THE IDTR IS REINITIALIZED WITH THE NEW INTERRUPT OFFSETS.
1135 | 7. THE PROCESSOR IS PUT IN VIRTUAL MODE WITH THE CODE
1136 | SEGMENT DESIGNATED FOR THIS FUNCTION.
1137 | 8. DATA SEGMENT IS LOADED WITH THE USER DEFINED
1138 | SELECTOR FOR THE DS REGISTER.
1139 | 9. EXTRA SEGMENT IS LOADED WITH THE USER DEFINED
1140 | SELECTOR FOR THE ES REGISTER.
1141 | 10. STACK SEGMENT IS LOADED WITH THE USER DEFINED
1142 | SELECTOR FOR THE SS REGISTER.
1143 | 11. CODE SEGMENT DESCRIPTOR SELECTOR VALUE IS
1144 | SUBSTITUTED ON THE STACK FOR RETURN TO USER.
1145 | 12. WE TRANSFER CONTROL TO THE USER WITH INTERRUPTS DISABLED.
1146 |

```

```

1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189 0000 ??????????????????
1190 0008 ??????????????????
1191 0010 ??????????????????
1192 0018 ??????????????????
1193 0020 ??????????????????
1194 0028 ??????????????????
1195 0030 ??????????????????
1196 0038 ??????????????????
1197 0040
1198
1199
1200
1201 0408
1202 0408
1203
1204
1205
1206 0408 FA
1207 0409 B4 DF
1208 040B EB 03DA R
1209 040E 3C 00
1210 0410 74 04
1211 0412 B4 FF
1212 0414 F9
1213 0415 CF
1214
1215
1216 0416
1217 0416 06
1218 0417 1F
1219
1220
1221
1222
1223
1224 0418 B0 11
1225 041A E6 20
1226 041C EB 00
1227 041E 8A C7
1228 0420 E6 21
1229 0422 EB 00
1230 0424 B0 04
1231 0426 E6 21
1232 0428 EB 00
1233 042A B0 01
1234 042C E6 21
1235 042E EB 00
1236 0430 B0 FF
1237 0432 E6 21
1238
1239
1240
1241
1242
1243 0434 B0 11
1244 0436 E6 A0
1245 0438 EB 00
1246 043A 8A C9
1247 043C E6 A1
1248 043E B0 02
1249 0440 EB 00
1250 0442 E6 A1
1251 0444 EB 00
1252 0446 B0 01
1253 0448 E6 A1
1254 044A EB 00
1255 044C B0 FF
1256 044E E6 A1
    
```

```

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
    DQ  ?           ; FIRST DESCRIPTOR NOT ACCESSIBLE
    DQ  ?           ; GDT DESCRIPTOR
    DQ  ?           ; IDT DESCRIPTOR
    DQ  ?           ; USER DATA SEGMENT DESCRIPTOR
    DQ  ?           ; USER EXTRA SEGMENT DESCRIPTOR
    DQ  ?           ; USER STACK SEGMENT DESCRIPTOR
    DQ  ?           ; USER CODE SEGMENT DESCRIPTOR
    DQ  ?           ; TEMPORARY BIOS DESCRIPTOR
VIRTUAL_ENABLE_GDT_DEF  ENDS

    ASSUME DS:DATA

X_VIRTUAL  PROC  FAR
SET_VMODE:
;-----  ENABLE ADDRESS LATCH BIT 20
    CLI           ; NO INTERRUPTS ALLOWED
    MOV  AH,ENABLE_BIT20  ; ENABLE BIT 20 FOR ADDRESS GATE
    CALL GATE_A20
    CMP  AL,0
    JZ  BIT20_ON  ; WAS THE COMMAND ACCEPTED?
    MOV  AH,0FFH  ; GO IF YES
    STC           ; SET THE ERROR FLAG
    IRET        ; SET CARRY
                ; EARLY EXIT

BIT20_ON:
    PUSH  ES           ; MOVE SEGMENT POINTER
    POP  DS            ; TO THE DATA SEGMENT

;-----
; REINITIALIZE THE 8259 INTERRUPT CONTROLLER #1 TO THE USER SPECIFIED OFFSET ;
;-----
    MOV  AL,11H
    OUT  INTA00,AL    ; START INITIALIZATION SEQUENCE-ICW1
                    ; EDGE,INTERVAL-8,MASTER,ICW4 NEEDED
    JMP  $+2
    MOV  AL,BH
    OUT  INTA01,AL    ; HARDWARE INT'S START AT INT # (BH)
                    ; SEND ICW2
    JMP  $+2
    MOV  AL,04H
    OUT  INTA01,AL    ; SEND ICW3 - MASTER LEVEL 2
                    ; SEND ICW4 - MASTER,8086 MODE
    JMP  $+2
    MOV  AL,01H
    OUT  INTA01,AL    ; SEND ICW4 - SLAVE,8086 MODE
                    ; MASK OFF ALL INTERRUPTS
    JMP  $+2
    MOV  AL,0FFH
    OUT  INTA01,AL    ; MASK OFF ALL INTERRUPTS

;-----
; REINITIALIZE THE 8259 INTERRUPT CONTROLLER #2 TO THE USER SPECIFIED OFFSET ;
;-----
    MOV  AL,11H
    OUT  INTB00,AL    ; INITIALIZE SEQUENCE-ICW1 FOR SLAVE
                    ; EDGE,INTERVAL-8,MASTER,ICW4 NEEDED
    JMP  $+2
    MOV  AL,BL
    OUT  INTB01,AL    ; HARDWARE INT'S START AT INT # (BL)
                    ; SEND ICW2
    MOV  AL,02H
    OUT  INTB01,AL    ; SEND ICW3 - SLAVE LEVEL 2
                    ; SEND ICW4 - SLAVE,8086 MODE
    JMP  $+2
    MOV  AL,01H
    OUT  INTB01,AL    ; SEND ICW4 - SLAVE,8086 MODE
                    ; MASK OFF ALL INTERRUPTS
    JMP  $+2
    MOV  AL,0FFH
    OUT  INTB01,AL    ; MASK OFF ALL INTERRUPTS
    
```

SECTION 5

```

1257 PAGE
1258 -----
1259 ; SETUP BIOS CODE SEGMENT DESCRIPTOR ;
1260 -----
1261
1262 0480 C7 44 38 FFFF      MOV     [SI],BIO_CS.SEG_LIMIT,MAX_SEG_LEN ; SET LENGTH
1263 0485 C6 44 3C 0F      MOV     [SI],BIO_CS.BASE_HI_BYTE,CSEG@_HI ; SET HIGH BYTE OF CS=0F
1264 0489 C7 44 3A 0000    MOV     [SI],BIO_CS.BASE_LO_WORD,CSEG@_LO ; SET LOW WORD OF CS=0
1265 048E C6 44 3D 9B      MOV     [SI],BIO_CS.DATA_ACC_RIGHTS,CPC@_CODE_ACCESS
1266 0462 C7 44 3E 0000    MOV     [SI],BIO_CS.DATA_RESERVED,0 ; ZERO RESERVED AREA
1267
1268 -----
1269 ; ENABLE PROTECTED MODE ;
1270 -----
1271
1272 0467 0F                + LGDT  [SI],GDTPTR ; LOAD GLOBAL DESCRIPTOR TABLE REGISTER
1273 0468                + DB    00FH
1274 0468 8B 54 08        + LABEL BYTE
1275 0468                + MOV   DX,WORD PTR [SI],GDTPTR
1276 0468                + LABEL BYTE
1277 0468 01                + ORG  OFFSET CS:??0005
1278 046B                + DB    001H
1279                + ORG  OFFSET CS:??0006 ; INTERRUPT DESCRIPTOR TABLE REGISTER
1280 046B 0F                + LGDT  [SI],IDTPTR
1281 046C                + DB    00FH
1282 046C 8B 5C 10        + LABEL BYTE
1283 046F                + MOV   BX,WORD PTR [SI],IDTPTR
1284 046C                + LABEL BYTE
1285 046C 01                + ORG  OFFSET CS:??0007
1286 046F                + DB    001H
1287                + ORG  OFFSET CS:??0008
1288 046F B8 0001        MOV     AX,VIRTUAL_ENABLE ; MACHINE STATUS WORD NEEDED TO
1289                LMSW  AX ; SWITCH TO VIRTUAL MODE
1290 0472 0F 01 F0        DB     00FH,001H,0F0H
1291 0475 EA                DB     0EAH ; PURGE PRE-FETCH QUEUE WITH FAR JUMP
1292 0476 047A R          DW     OFFSET VMODE ; - TO OFFSET
1293 0476 0038            DW     BIO_CS ; - IN SEGMENT -PROTECTED MODE SELECTOR
1294
1295 047A                VMODE:
1296 -----
1297 ; SETUP USER SEGMENT REGISTERS ;
1298 -----
1299 047A B8 0018        MOV     AX,USER_DS ; SETUP USER'S DATA SEGMENT
1300 047D 8E D8          MOV     DS,AX ; TO PROTECTED MODE SELECTORS
1301 047F B8 0020        MOV     AX,USER_ES ; SETUP USER'S EXTRA SEGMENT
1302 0482 8E CD          MOV     ES,AX
1303 0484 B8 0028        MOV     AX,USER_SS ; SETUP USER'S STACK SEGMENT
1304 0487 8E D0          MOV     SS,AX
1305
1306 -----
1307 ; PUT TRANSFER ADDRESS ON STACK ;
1308 ; AND RETURN TO THE USER ;
1309 -----
1309 0489 5B            POP     BX ; GET RETURN IP FROM THE STACK
1310 048A 83 C4 04      ADD     SP,4 ; NORMALIZE STACK POINTER
1311 048D 6A 30        PUSH   USER_CS ; SET STACK FOR A RETURN FAR
1312 048F 53            PUSH   BX
1313 0490 CB            RET     ; RETURN TO USER IN VIRTUAL MODE
1314
1315 0491                X_VIRTUAL ENDP
1316
1317 ;--- DEVICE BUSY AND INTERRUPT COMPLETE ---
1318 ;
1319 ; THIS ROUTINE IS A TEMPORARY HANDLER FOR DEVICE BUSY ;
1320 ; AND INTERRUPT COMPLETE ;
1321 ;
1322 ; INPUT - SEE PROLOGUE ;
1323 -----
1324
1325 0491                DEVICE_BUSY PROC NEAR
1326 0491 F8            CLC ; TURN CARRY OFF
1327 0492 E9 0055 R    JMP     C1,F NEAR ; RETURN WITH CARRY FLAG
1328 0495                DEVICE_BUSY ENDP
1329
1330 0495                INT_COMPLETE PROC NEAR
1331 0495 CF            IRET ; RETURN
1332 0496                INT_COMPLETE ENDP
1333
1334 0496                CODE ENDS
1335 END

```

```

1 PAGE 118,121
2 TITLE BIOS2 ---- 06/10/85 BIOS INTERRUPT ROUTINES
3 .286C
4 .LIST
5 0000 CODE SEGMENT BYTE PUBLIC
6
7 PUBLIC PRINT_SCREEN_1
8 PUBLIC RTC_INT
9 PUBLIC TIME_OF_DAY_1
10 PUBLIC TIMER_INT_1
11
12 EXTRN CMOS_READ:NEAR ; READ CMOS LOCATION ROUTINE
13 EXTRN CMOS_WRITE:NEAR ; WRITE CMOS LOCATION ROUTINE
14 EXTRN DDS:NEAR ; LOAD (DS) WITH DATA SEGMENT SELECTOR
15
16 ;--- INT 1A H -- (TIME OF DAY) -----
17 ; THIS BIOS ROUTINE ALLOWS THE CLOCKS TO BE SET OR READ
18 ;
19 ;
20 ; PARAMETERS:
21 ; (AH) = 00H READ THE CURRENT CLOCK SETTING AND RETURN WITH,
22 ; (CX) = HIGH PORTION OF COUNT
23 ; (DX) = LOW PORTION OF COUNT
24 ; (AL) = 0 TIMER HAS NOT PASSED 24 HOURS SINCE LAST READ
25 ; ; IF ON ANOTHER DAY. (RESET TO ZERO AFTER READ)
26 ;
27 ; (AH) = 01H SET THE CURRENT CLOCK USING,
28 ; (CX) = HIGH PORTION OF COUNT
29 ; (DX) = LOW PORTION OF COUNT
30 ;
31 ; NOTE: COUNTS OCCUR AT THE RATE OF 1193180/65536 COUNTS/SECOND
32 ; (OR ABOUT 18.2 PER SECOND -- SEE EQUATES)
33 ;
34 ; (AH) = 02H READ THE REAL TIME CLOCK AND RETURN WITH,
35 ; (CH) = HOURS IN BCD (00-23)
36 ; (CL) = MINUTES IN BCD (00-59)
37 ; (DH) = SECONDS IN BCD (00-59)
38 ; (DL) = DAYLIGHT SAVINGS ENABLE (00-01).
39 ;
40 ; (AH) = 03H SET THE REAL TIME CLOCK USING,
41 ; (CH) = HOURS IN BCD (00-23)
42 ; (CL) = MINUTES IN BCD (00-59)
43 ; (DH) = SECONDS IN BCD (00-59)
44 ; (DL) = 01 IF DAYLIGHT SAVINGS ENABLE OPTION, ELSE 00.
45 ;
46 ; NOTE: (DL) = 00 IF DAYLIGHT SAVINGS TIME ENABLE IS NOT ENABLED.
47 ; (DL) = 01 ENABLES TWO SPECIAL UPDATES THE LAST SUNDAY IN
48 ; APRIL (1:59:59 --> 3:00:00 AM) AND THE LAST SUNDAY IN
49 ; OCTOBER (1:59:59 --> 1:00:00 AM) THE FIRST TIME.
50 ;
51 ; (AH) = 04H READ THE DATE FROM THE REAL TIME CLOCK AND RETURN WITH,
52 ; (CH) = CENTURY IN BCD (19 OR 20)
53 ; (CL) = YEAR IN BCD (00-99)
54 ; (DH) = MONTH IN BCD (01-12)
55 ; (DL) = DAY IN BCD (01-31).
56 ;
57 ; (AH) = 05H SET THE DATE INTO THE REAL TIME CLOCK USING,
58 ; (CH) = CENTURY IN BCD (19 OR 20)
59 ; (CL) = YEAR IN BCD (00-99)
60 ; (DH) = MONTH IN BCD (01-12)
61 ; (DL) = DAY IN BCD (01-31).
62 ;
63 ; (AH) = 06H SET THE ALARM TO INTERRUPT AT SPECIFIED TIME,
64 ; (CL) = MINUTES IN BCD (00-59 (OR FFH))
65 ; (DH) = SECONDS IN BCD (00-59 (OR FFH)).
66 ;
67 ; (AH) = 07H RESET THE ALARM INTERRUPT FUNCTION.
68 ;
69 ; NOTES: FOR ALL RETURNS CY= 0 FOR SUCCESSFUL OPERATION.
70 ; FOR (AH)= 2, 4, 6 - CARRY FLAG SET IF REAL TIME CLOCK NOT OPERATING.
71 ; FOR (AH)= 6 - CARRY FLAG SET IF ALARM ALREADY ENABLED.
72 ; FOR THE ALARM FUNCTION (AH = 6) THE USER MUST SUPPLY A ROUTINE AND
73 ; INTERCEPT THE CORRECT ADDRESS IN THE VECTOR TABLE FOR INTERRUPT 4AH.
74 ; USE 0FTH FOR ANY "DO NOT CARE" POSITION FOR INTERVAL INTERRUPTS.
75 ; INTERRUPTS ARE DISABLED DURING DATA MODIFICATION.
76 ; AH & AL ARE RETURNED MODIFIED AND NOT DEFINED EXCEPT WHERE INDICATED.
77 ;-----
78 ASSUME CS:CODE,DS:DATA
79
80 0000 TIME_OF_DAY_1 PROC FAR
81 0000 FB ; INTERRUPTS BACK ON
82 0001 80 FC 08 ; CHECK IF COMMAND IN VALID RANGE (0-7)
83 0004 F5 ; COMPLEMENT CARRY FOR ERROR EXIT
84 0006 72 17 ; EXIT WITH CARRY = 1 IF NOT VALID
85
86 0007 1E ; SAVE USERS (DS) SEGMENT
87 0008 E8 0000 E ; GET DATA SEGMENT SELECTOR
88 000B 56 ; SAVE WORK REGISTER
89 000C C1 E8 08 ; CONVERT FUNCTION TO BYTE OFFSET
90 000F 03 C0 ; CONVERT FUNCTION TO WORD OFFSET (CY=0)
91 0011 8B F0 ; PLACE INTO ADDRESSING REGISTER
92 0013 FA ; NO INTERRUPTS DURING TIME FUNCTIONS
93 0014 2E1 F4 0021 R ; VECTOR TO FUNCTION REQUESTED WITH CY=0
94 ; RETURN WITH CARRY FLAG SET FOR RESULT
95 0019 FB ; INTERRUPTS BACK ON
96 001A B4 00 ; CLEAR (AH) TO ZERO
97 001C 5E ; RECOVER USERS REGISTER
98 001D 1F ; RECOVER USERS SEGMENT SELECTOR
99 001E ; RETURN WITH CY= 0 IF NO ERROR
100 001E CA 0002 TIME_9: RET 2
101
102 ; ROUTINE VECTOR TABLE (AH)=
103 0021 0031 R ; 0 = READ CURRENT CLOCK COUNT
104 0023 0042 R ; 1 = SET CLOCK COUNT
105 0025 0050 R ; 2 = READ THE REAL TIME CLOCK TIME
106 0027 0075 R ; 3 = SET REAL TIME CLOCK TIME
107 0029 00A8 R ; 4 = READ THE REAL TIME CLOCK DATE
108 002B 00CB R ; 5 = SET REAL TIME CLOCK DATE
109 002D 0104 R ; 6 = SET THE REAL TIME CLOCK ALARM
110 002F 0145 R ; 7 = RESET ALARM
111 = 0031
112
113 0031 TIME_OF_DAY_1 ENDP
    
```

SECTION 5

```

114                                     PAGE
115 0031                                RTC_00 PROC NEAR
116 0031 A0 0070 R                      MOV AL,®TIMER_OFL
117 0034 C6 06 0070 R 00                MOV ®TIMER_OFL,0
118 0039 BB 0E 006E R                    MOV CX,®TIMER_HIGH
119 003D BB 16 006C R                    MOV DX,®TIMER_LOW
120 0041 C3                               RET
121
122 0042                                RTC_10:
123 0042 89 16 006C R                    MOV ®TIMER_LOW,DX
124 0046 89 0E 006E R                    MOV ®TIMER_HIGH,CX
125 004A C6 06 0070 R 00                MOV ®TIMER_OFL,0
126 004F C3                               RET
127
128 0050                                RTC_20:
129 0050 E8 016B R                        CALL UPD_IPR
130 0053 T2 1F                            JC RTC_29
131
132 0055 B0 00                            MOV AL,CMOS_SECONDS
133 0057 EB 0000 E                        CALL CMOS_READ
134 005A BA F0                            MOV DH,AL
135 005C B0 0B                            MOV AL,CMOS_REG_B
136 005E EB 0000 E                        CALL CMOS_READ
137 0061 24 01                            AND AL,00000001B
138 0063 EA 00                            MOV DL,AL
139 0065 B0 02                            MOV AL,CMOS_MINUTES
140 0067 EB 0000 E                        CALL CMOS_READ
141 006A BA C8                            MOV CL,AL
142 006C B0 04                            MOV AL,CMOS_HOURS
143 006E EB 0000 E                        CALL CMOS_READ
144 0071 BA E8                            MOV CH,AL
145 0073 F8                               CLC
146 0074                                     RTC_29:
147 0074 C3                               RET
148
149 0075                                RTC_30:
150 0075 E8 016B R                        CALL UPD_IPR
151 0078 T3 03                            JNC RTC_35
152 007A EB 0154 R                        CALL RTC_STA
153 007D
154 007D BA E6                            MOV AH,DH
155 007F B0 00                            MOV AL,CMOS_SECONDS
156 0081 EB 0000 E                        CALL CMOS_WRITE
157 0084 BA E1                            MOV AH,CL
158 0086 B0 02                            MOV AL,CMOS_MINUTES
159 0088 EB 0000 E                        CALL CMOS_WRITE
160 008B BA E5                            MOV AH,CH
161 008D B0 04                            MOV AL,CMOS_HOURS
162 008F EB 0000 E                        CALL CMOS_WRITE
163 0092 BB 080B                            MOV AX,®CMOS_REG_B
164 0095 EB 0000 E                        CALL CMOS_READ
165 0098 24 62                            AND AL,0100010B
166 009A 0C 02                            OR AL,00000010B
167 009C B0 E2 01                            AND DL,00000001B
168 009F 0A C2                            OR AL,DL
169 00A1 B6 E0                            XCHG AH,AL
170 00A3 EB 0000 E                        CALL CMOS_WRITE
171 00A6 F8                               CLC
172 00A7 C3                               RET
173
174 00A8                                RTC_40:
175 00A8 E8 016B R                        CALL UPD_IPR
176 00AB T2 1D                            JC RTC_49
177
178 00AD B0 07                            MOV AL,CMOS_DAY_MONTH
179 00AF EB 0000 E                        CALL CMOS_READ
180 00B2 BA D0                            MOV DL,AL
181 00B4 B0 08                            MOV AL,CMOS_MONTH
182 00B6 EB 0000 E                        CALL CMOS_READ
183 00B9 BA F0                            MOV DH,AL
184 00BB B0 09                            MOV AL,CMOS_YEAR
185 00BD EB 0000 E                        CALL CMOS_READ
186 00C0 BA C5                            MOV CL,AL
187 00C2 B0 32                            MOV AL,CMOS_CENTURY
188 00C4 EB 0000 E                        CALL CMOS_READ
189 00C7 BA E8                            MOV CH,AL
190 00C9 F8                               CLC
191 00CA                                     RTC_49:
192 00CA C3                               RET
193
194 00CB                                RTC_50:
195 00CB E8 016B R                        CALL UPD_IPR
196 00CE T3 03                            JNC RTC_55
197 00D0 EB 0154 R                        CALL RTC_STA
198 00D3
199 00D3 BB 0006                            MOV AX,CMOS_DAY_WEEK
200 00D6 EB 0000 E                        CALL CMOS_WRITE
201 00D9 BA E2                            MOV AH,DL
202 00DB B0 07                            MOV AL,CMOS_DAY_MONTH
203 00DD EB 0000 E                        CALL CMOS_WRITE
204 00E0 BA E6                            MOV AH,DH
205 00E2 B0 08                            MOV AL,CMOS_MONTH
206 00E4 EB 0000 E                        CALL CMOS_WRITE
207 00E7 BA E1                            MOV AH,CL
208 00E9 B0 09                            MOV AL,CMOS_YEAR
209 00EB EB 0000 E                        CALL CMOS_WRITE
210 00EE BA E5                            MOV AH,CH
211 00F0 B0 32                            MOV AL,CMOS_CENTURY
212 00F2 EB 0000 E                        CALL CMOS_WRITE
213 00F5 BB 080B                            MOV AX,®CMOS_REG_B
214 00F8 EB 0000 E                        CALL CMOS_READ
215 00FB 24 7F                            AND AL,07FH
216 00FD B6 E0                            XCHG AH,AL
217 00FF EB 0000 E                        CALL CMOS_WRITE
218 0102 F8                               CLC
219 0103 C3                               RET

```

```

220 PAGE
221 RTC_60:
222 0104 B0 0B MOV AL,CMOS_REG_B ; SET RTC ALARM
223 0106 E9 0000 E CALL CMOS_READ ; ADDRESS ALARM
224 0109 A8 20 TEST AL,20H ; READ ALARM REGISTER
225 010B F9 STC ; CHECK FOR ALARM ALREADY ENABLED
226 010C 75 33 JNZ RTC_69 ; SET CARRY IN CASE OF ERROR
227 ; ERROR EXIT IF ALARM SET
228 010E E8 016B R CALL UPD_IPR ; CHECK FOR UPDATE IN PROCESS
229 0111 73 03 JNC RTC_65 ; SKIP INITIALIZATION IF NO ERROR
230 0113 E8 0154 R CALL RTC_STA ; ELSE INITIALIZE CLOCK
231 0116
232 0116 8A E6 MOV AH,DH ; GET SECONDS BYTE
233 0118 B0 01 MOV AL,CMOS_SEC_ALARM ; ADDRESS THE SECONDS ALARM REGISTER
234 011A E8 0000 E CALL CMOS_WRITE ; INSERT SECONDS
235 011D 5A E1 MOV AH,CL ; GET MINUTES PARAMETER
236 011F B0 03 MOV AL,CMOS_MIN_ALARM ; ADDRESS MINUTES ALARM REGISTER
237 0121 E8 0000 E CALL CMOS_WRITE ; INSERT MINUTES
238 0124 5A E5 MOV AH,CH ; GET HOURS PARAMETER
239 0126 B0 05 MOV AL,CMOS_HR_ALARM ; ADDRESS HOUR ALARM REGISTER
240 0128 E8 0000 E CALL CMOS_WRITE ; INSERT HOURS
241 012B 54 A1 IN AL,IRTB01 ; READ SECOND INTERRUPT MASK REGISTER
242 012D 34 FE AND AL,0FEH ; ENABLE ALARM TIMER BIT (CY= 0)
243 012F E6 A1 OUT INTB01,AL ; WRITE UPDATED MASK
244 0131 B8 0B0B MOV AX,X*CMOS_REG_B ; ADDRESS ALARM REGISTER
245 0134 E8 0000 E CALL CMOS_READ ; READ CURRENT ALARM REGISTER
246 0137 24 7F AND AL,07FH ; ENSURE SET BIT TURNED OFF
247 0139 0C 20 OR AL,20H ; TURN ON ALARM ENABLE
248 013B 86 E0 XCHG AH,AL ; MOVE MASK TO OUTPUT REGISTER
249 013D E8 0000 E CALL CMOS_WRITE ; WRITE NEW ALARM MASK
250 0140 F8 CLC ; SET CY= 0
251 0141
252 0141 B8 0000 RTC_69: MOV AX,0 ; CLEAR AX REGISTER
253 0144 C3 RET ; RETURN WITH RESULTS IN CARRY FLAG
254
255 0148 RTC_70: ; RESET ALARM
256 0148 B8 0B0B MOV AX,X*CMOS_REG_B ; ADDRESS ALARM REGISTER (TO BOTH AH,AL)
257 014B E8 0000 E CALL CMOS_READ ; READ ALARM REGISTER
258 014B 24 57 AND AL,57H ; TURN OFF ALARM ENABLE
259 014D 86 E0 XCHG AH,AL ; SAVE DATA AND RECOVER ADDRESS
260 014F E8 0000 E CALL CMOS_WRITE ; RESTORE NEW VALUE
261 0152 F8 CLC ; SET CY= 0
262 0153 C3 RET ; RETURN WITH NO CARRY
263
264 0154 RTC_00 ENDP
265
266 0154 RTC_STA PROC NEAR ; INITIALIZE REAL TIME CLOCK
267 0154 B8 260A MOV AX,26H*H+CMOS_REG_A ; ADDRESS REGISTER A AND LOAD DATA MASK
268 0157 E8 0000 E CALL CMOS_WRITE ; INITIALIZE STATUS REGISTER A
269 015A B8 820B MOV AX,82H*H+CMOS_REG_B ; SET "SET BIT" FOR CLOCK INITIALIZATION
270 015D E8 0000 E CALL CMOS_WRITE ; AND 24 HOUR MODE TO REGISTER B
271 0160 B0 0C MOV AL,CMOS_REG_C ; ADDRESS REGISTER C
272 0162 E8 0000 E CALL CMOS_READ ; READ REGISTER C TO INITIALIZE
273 0165 B0 0D MOV AL,CMOS_REG_D ; ADDRESS REGISTER D
274 0167 E8 0000 E CALL CMOS_READ ; READ REGISTER D TO INITIALIZE
275 016A C3 RET
276
277 016B RTC_STA ENDP
278
279 016B UPD_IPR PROC NEAR ; WAIT TILL UPDATE NOT IN PROGRESS
280 016B 51 PUSH CX ; SAVE CALLERS REGISTER
281 016C B9 0320 MOV CX,800 ; SET TIMEOUT LOOP COUNT
282 016F UPD_10: MOV AL,CMOS_REG_A ; ADDRESS STATUS REGISTER A
283 016F B0 0A CL I ; NO TIMER INTERRUPTS DURING UPDATES
284 0171 FA CALL CMOS_READ ; READ UPDATE IN PROCESS FLAG
285 0172 E8 0000 E CALL AL,80H ; IF UIP BIT IS ON ( CANNOT READ TIME )
286 0175 A8 80 TEST ; EXIT WITH CY= 0 IF CAN READ CLOCK NOW
287 0177 74 06 JZ UPD_90 ; ALLOW INTERRUPTS WHILE WAITING
288 0179 FB STJ ; LOOP TILL READY OR TIMEOUT
289 017A E2 F3 LOOP UPD_10 ; CLEAR RESULTS IF ERROR
290 017C 33 C0 XOR AX,AX ; SET CARRY FOR ERROR
291 017E F9 STC
292 017F UPD_90: POP CX ; RESTORE CALLERS REGISTER
293 017F 59 CL I ; INTERRUPTS OFF DURING SET
294 0180 FA RET ; RETURN WITH CY FLAG SET
295 0181 C3
296
297 0182 UPD_IPR ENDP

```

SECTION 5

```

298 PAGE
299 ;--- HARDWARE INT 70 H -- ( IRQ LEVEL 8 ) -----
300 ; ALARM INTERRUPT HANDLER (RTC)
301 ; THIS ROUTINE HANDLES THE PERIODIC AND ALARM INTERRUPTS FROM THE CMOS
302 ; TIMER. INPUT FREQUENCY IS 1.024 KHZ OR APPROXIMATELY 1024 INTERRUPTS
303 ; EVERY SECOND FOR THE PERIODIC INTERRUPT. FOR THE ALARM FUNCTION,
304 ; THE INTERRUPT WILL OCCUR AT THE DESIGNATED TIME.
305 ;
306 ; INTERRUPTS ARE ENABLED WHEN THE EVENT OR ALARM FUNCTION IS ACTIVATED.
307 ; FOR THE EVENT INTERRUPT, THE HANDLER WILL DECREMENT THE WAIT COUNTER
308 ; AND WHEN IT EXPIRES WILL SET THE DESIGNATED LOCATION TO 80H. FOR
309 ; THE ALARM INTERRUPT, THE USER MUST PROVIDE A ROUTINE TO INTERCEPT
310 ; THE CORRECT ADDRESS FROM THE VECTOR TABLE INVOKED BY INTERRUPT 4AH
311 ; PRIOR TO SETTING THE REAL TIME CLOCK ALARM (INT 1AH, AH= 06H).
312 ;-----
313
314 0182 RTC_INT PROC FAR ; ALARM INTERRUPT
315 0182 1E PUSH DS ; LEAVE INTERRUPTS DISABLED
316 0183 50 PUSH AX ; SAVE REGISTERS
317 0184 57 PUSH DI
318
319 0185 RTC_I_1: ; CHECK FOR SECOND INTERRUPT
320 0185 88 88BC MOY AX, (CMOS_REG_B+NMI)*H+CMOS_REG_C+NMI ; ALARM AND STATUS
321 0186 E6 70 OUT CMOS_PORT,AL ; WRITE ALARM FLAG MASK ADDRESS
322 018A 90 NOP ; I/O DELAY
323 018B E4 71 IN AL,CMOS_DATA ; READ AND RESET INTERRUPT REQUEST FLAGS
324 018D A8 60 TEST AL,0100000B ; CHECK FOR EITHER INTERRUPT PENDING
325 018F 74 5C JZ RTC_I_9 ; EXIT IF NOT A VALID RTC INTERRUPT
326
327 0191 86 E0 XCHG AH,AL ; SAVE FLAGS AND GET ENABLE ADDRESS
328 0193 E6 70 OUT CMOS_PORT,AL ; WRITE ALARM ENABLE MASK ADDRESS
329 0195 90 NOP ; I/O DELAY
330 0196 E4 71 IN AL,CMOS_DATA ; READ CURRENT ALARM ENABLE MASK
331 0198 22 C4 AND AL,AH ; ALLOW ONLY SOURCES THAT ARE ENABLED
332 019A A8 40 TEST AL,01000000B ; CHECK FOR PERIODIC INTERRUPT
333 019C 74 3F JZ RTC_I_5 ; SKIP IF NOT A PERIODIC INTERRUPT
334
335 ;----- DECREMENT WAIT COUNT BY INTERRUPT INTERVAL
336
337 019E E8 0000 E CALL DDS ; ESTABLISH DATA SEGMENT ADDRESSABILITY
338 01A1 81 2E 009C R 03DD SUB @RTC_LOW,0976 ; DECREMENT COUNT LOW BY 1/1024
339 01A7 83 1E 009E R 00 SBB @RTC_HIGH,0 ; ADJUST HIGH WORD FOR LOW WORD BORROW
340 01AC 73 2F JNC RTC_I_5 ; SKIP TILL 32 BIT WORD LESS THAN ZERO
341
342 ;----- TURN OFF PERIODIC INTERRUPT ENABLE
343
344 01AE 50 PUSH AX ; SAVE INTERRUPT FLAG MASK
345 01AF 88 88BB MOY AX,*(CMOS_REG_B+NMI) ; INTERRUPT ENABLE REGISTER
346 01B2 E6 70 OUT CMOS_PORT,AL ; WRITE ADDRESS TO CMOS CLOCK
347 01B4 90 NOP ; I/O DELAY
348 01B5 E4 71 IN AL,CMOS_DATA ; READ CURRENT ENABLER
349 01B7 24 BF AND AL,0BFH ; TURN OFF PIE
350 01B9 86 C4 XCHG AL,AH ; GET CMOS ADDRESS AND SAVE VALUE
351 01BB E6 70 OUT CMOS_PORT,AL ; ADDRESS REGISTER B
352 01BD 86 C4 XCHG AL,AH ; GET NEW INTERRUPT ENABLE MASK
353 01BF E6 71 OUT CMOS_DATA,AL ; SET MASK IN INTERRUPT ENABLE REGISTER
354 01C1 88 POP AX ; GET INTERRUPT SOURCE BACK
355 01C2 F6 06 00A0 R 02 TEST @RTC_WAIT_FLAG,02H ; CHECK FOR "WAIT" FUNCTION ACTIVE
356 01C7 C6 06 00A0 R 00 MOY @RTC_WAIT_FLAG,0 ; SET FUNCTION ACTIVE FLAGS OFF
357 01CC C5 3E 0098 R LDS DI,WORD PTR @USER_FLAG ; SET UP (DS:DI) TO POINT TO USER FLAG
358 01DD C6 05 80 MOV BYTE PTR [DI],80H ; TURN ON USERS POSTED FLAG
359 01D3 74 08 JZ RTC_I_5 ; SKIP IF "EVENT_WAIT" FUNCTION
360
361 01D5 E8 0000 E CALL DDS ; ESTABLISH DATA SEGMENT ADDRESSABILITY
362 01D8 C6 06 00A0 R 83 MOY @RTC_WAIT_FLAG,083H ; AND SET "WAIT" BACK TO BUSY & POSTED
363 01DD TEST AL,00100000B ; TEST FOR ALARM INTERRUPT
364 01DF 74 0A JZ RTC_I_7 ; SKIP USER INTERRUPT CALL IF NOT ALARM
365
366 01E1 80 0F MOY AL,CMOS_SHUT_DOWN ; POINT TO DEFAULT READ ONLY REGISTER
367 01E3 E6 70 OUT CMOS_PORT,AL ; ENABLE NMI AND CMOS ADDRESS TO DEFAULT
368 01E5 FB STI ; INTERRUPTS BACK ON NOW
369 01E6 62 PUSH DX
370 01E7 CD 4A INT 4AH ; TRANSFER TO USER ROUTINE
371 01E9 5A POP DX
372 01EA FA CLI ; BLOCK INTERRUPT FOR RETRY
373 01EB EB 98 RTC_I_7: JMP RTC_I_1 ; RESTART ROUTINE TO HANDLE DELAYED
374 ; ENTRY AND SECOND EVENT BEFORE DONE
375
376
377
378 01ED RTC_I_9: ; EXIT - NO PENDING INTERRUPTS
379 01ED 80 0F MOY AL,CMOS_SHUT_DOWN ; POINT TO DEFAULT READ ONLY REGISTER
380 01EF E6 70 OUT CMOS_PORT,AL ; ENABLE NMI AND CMOS ADDRESS TO DEFAULT
381 01F1 90 NOP ; I/O DELAY
382 01F2 E4 71 IN AL,CMOS_DATA ; OPEN STANDBY LATCH
383 01F4 80 20 MOV AL,ED1 ; END OF INTERRUPT MASK TO 8259 - 2
384 01F6 E6 A0 OUT INTB00,AL ; TO 8259 - 2
385 01F8 E6 20 OUT INTA00,AL ; TO 8259 - 1
386 01FA 5F POP DI ; RESTORE REGISTERS
387 01FB 58 POP AX
388 01FC 1F POP DS
389 01FD CF IRET ; END OF INTERRUPT
390
391 01FE RTC_INT ENDP

```



```

392 PAGE
393 ;--- INT 05H -----
394 ; PRINT_SCREEN
395 ; THIS LOGIC WILL BE INVOKED BY INTERRUPT 05H TO PRINT THE SCREEN.
396 ; THE CURSOR POSITION AT THE TIME THIS ROUTINE IS INVOKED WILL BE
397 ; SAVED AND RESTORED UPON COMPLETION. THE ROUTINE IS INTENDED TO
398 ; RUN WITH INTERRUPTS ENABLED. IF A SUBSEQUENT PRINT_SCREEN KEY
399 ; IS DEPRESSED WHILE THIS ROUTINE IS PRINTING IT WILL BE IGNORED.
400 ; THE BASE PRINTERS STATUS IS CHECKED FOR NOT BUSY AND NOT OUT OF
401 ; PAPER. AN INITIAL STATUS ERROR WILL ABEND THE PRINT REQUEST.
402 ; ADDRESS 0050:0000 CONTAINS THE STATUS OF THE PRINT SCREEN:
403 ;
404 ; 50:0 = 0 PRINT_SCREEN HAS NOT BEEN CALLED OR UPON RETURN
405 ; FROM A CALL THIS INDICATES A SUCCESSFUL OPERATION.
406 ; = 1 PRINT_SCREEN IS IN PROGRESS IGNORE THIS REQUEST.
407 ; = 255 ERROR ENCOUNTERED DURING PRINTING.
408 ;-----
409
410 PRINT_SCREEN_1 PROC FAR ; DELAY INTERRUPT ENABLE TILL FLAG SET
411
412 01FE IE PUSH DS ; SAVE WORK REGISTERS
413 01FF 50 PUSH AX
414 0200 53 PUSH BX
415 0201 51 PUSH CX
416 0202 52 PUSH DX ; USE 0040:0100 FOR STATUS AREA STORAGE
417 0203 E8 0000 E CALL DDS ; GET STATUS BYTE DATA SEGMENT
418 0206 80 3E 0100 R 01 CMP #STATUS_BYTE,1 ; SEE IF PRINT ALREADY IN PROGRESS
419 020B 74 74 JE PR190 ; EXIT IF PRINT ALREADY IN PROGRESS
420 020D C6 06 0100 R 01 MOV #STATUS_BYTE,1 ; INDICATE PRINT NOW IN PROGRESS
421 0212 FB STI ; MUST RUN WITH INTERRUPTS ENABLED
422 0213 94 0F MOV AH,0FH ; WILL REQUEST THE CURRENT SCREEN MODE
423 0215 CD 10 INT 10H ; (AL)= MODE
424 ; (AH)= NUMBER COLUMNS/LINE
425 ; (BH)= VISUAL PAGE
426 ; WILL MAKE USE OF (CX) REGISTER TO
427 0217 8A CC MOV CL,AH ; CONTROL ROWS ON SCREEN & COLUMNS
428 0219 8A 2E 0084 R MOV CH,#ROWS ; ADJUST ROWS ON DISPLAY COUNT
429 021D FE C5 INC CH ; (CL)= NUMBER COLUMNS/LINE
430 ; (CH)= NUMBER OF ROWS ON DISPLAY
431
432 ;-----
433 ; AT THIS POINT WE KNOW THE COLUMNS/LINE COUNT IS IN (CL) ;
434 ; AND THE NUMBER OF ROWS ON THE DISPLAY IS IN (CH) ;
435 ; THE PAGE IF APPLICABLE IS IN (BH). THE STACK HAS ;
436 ; (DS),(AX),(BX),(CX),(DX) PUSHED. ;
437 ;-----
438 021F 33 02 XOR DX,DX ; FIRST PRINTER
439 0221 B4 02 MOV AH,02H ; SET PRINTER STATUS REQUEST COMMAND
440 0223 CD 17 INT 17H ; REQUEST CURRENT PRINTER STATUS
441 0225 80 F4 80 XOR AH,080H ; CHECK FOR PRINTER BUSY (NOT CONNECTED)
442 0228 F6 C4 A0 TEST AH,0A0H ; OR OUT OF PAPER
443 022B 75 4E JNZ PR180 ; ERROR EXIT IF PRINTER STATUS ERROR
444 022D E8 0287 R CALL CRLF ; CARRIAGE RETURN LINE FEED TO PRINTER
445
446 0230 51 PUSH CX ; SAVE SCREEN BOUNDS
447 0231 B4 03 MOV AH,03H ; NOW READ THE CURRENT CURSOR POSITION
448 0233 CD 10 INT 10H ; AND RESTORE AT END OF ROUTINE
449 0235 59 POP CX ; RECALL SCREEN BOUNDS
450 0236 52 PUSH DX ; PRESERVE THE ORIGINAL POSITION
451 0237 33 02 XOR DX,DX ; INITIAL CURSOR (0,0) AND FIRST PRINTER
452 ;-----
453 ; THIS LOOP IS TO READ EACH CURSOR POSITION FROM THE ;
454 ; SCREEN AND PRINT IT. (BH)= VISUAL PAGE (CH)= ROWS ;
455 ;-----
456
457 0239 PR110: MOV AH,02H ; INDICATE CURSOR SET REQUEST
458 023B CD 10 INT 10H ; NEW CURSOR POSITION ESTABLISHED
459 023D B4 08 MOV AH,08H ; INDICATE READ CHARACTER FROM DISPLAY
460 023F CD 10 INT 10H ; CHARACTER NOW IN (AL)
461 0241 0A C0 OR AL,AL ; SEE IF VALID CHAR
462 0243 75 02 JNZ PR120 ; JUMP IF VALID CHAR
463 0245 B0 20 MOV AL,' ' ; ELSE MAKE IT A BLANK
464 0247 PR120:
465 0247 52 PUSH DX ; SAVE CURSOR POSITION
466 0248 33 02 XOR DX,DX ; INDICATE FIRST PRINTER (DX= 0)
467 024A 32 E4 XOR AH,AH ; INDICATE PRINT CHARACTER IN (AL)
468 024C CD 17 INT 17H ; PRINT THE CHARACTER
469 024E 5A POP DX ; RECALL CURSOR POSITION
470 024F F6 C4 29 TEST AH,29H ; TEST FOR PRINTER ERROR
471 0252 75 22 JNZ PR170 ; EXIT IF ERROR DETECTED
472 0254 FE C2 INC DL ; ADVANCE TO NEXT COLUMN
473 0256 3A CA CMP CL,DL ; SEE IF AT END OF LINE
474 0258 75 0F JNZ PR110 ; IF NOT LOOP FOR NEXT COLUMN
475 025A 32 02 XOR DL,DL ; BACK TO COLUMN 0
476 025C 8A E2 MOV AH,DL ; (AH)=0
477 025E 52 PUSH DX ; SAVE NEW CURSOR POSITION
478 025F E8 0287 R CALL CRLF ; LINE FEED CARRIAGE RETURN
479 0262 5A POP DX ; RECALL CURSOR POSITION
480 0263 FE C6 INC DH ; ADVANCE TO NEXT LINE
481 0265 3A EE CMP CH,DH ; FINISHED?
482 0267 75 00 JNZ PR110 ; IF NOT LOOP FOR NEXT LINE
483
484 0269 5A POP DX ; GET CURSOR POSITION
485 026A B4 02 MOV AH,02H ; INDICATE REQUEST CURSOR SET
486 026C CD 10 INT 10H ; CURSOR POSITION RESTORED
487 026E FA CLI ; BLOCK INTERRUPTS TILL STACK CLEARED
488 026F C6 06 0100 R 00 MOV #STATUS_BYTE,0 ; MOVE OK RESULTS FLAG TO STATUS_BYTE
489 0274 EB 0B JMP SHORT PR190 ; EXIT PRINTER ROUTINE
490
491 0276 PR170: ; ERROR EXIT
492 0276 5A POP DX ; GET CURSOR POSITION
493 0277 B4 02 MOV AH,02H ; INDICATE REQUEST CURSOR SET
494 0279 CD 10 INT 10H ; CURSOR POSITION RESTORED
495 027B PR180:
496 027B FA CLI ; BLOCK INTERRUPTS TILL STACK CLEARED
497 027C C6 06 0100 R FF MOV #STATUS_BYTE,0FFH ; SET ERROR FLAG
498 0281 PR190:
499 0281 5A POP DX ; EXIT ROUTINE
500 0282 59 POP CX ; RESTORE ALL THE REGISTERS USED
501 0283 5B POP BX
502 0284 58 POP AX
503 0285 5F POP DS
504 0286 CF IRET ; RETURN WITH INITIAL INTERRUPT MASK
505 0287 PRINT_SCREEN_1 ENDP

```

SECTION 5

```

506
507
508
509 0287          ;----- CARRIAGE RETURN, LINE FEED SUBROUTINE
510          CRLF PROC NEAR
511          XOR DX,DX          ; SEND CR,LF TO FIRST PRINTER
512          MOV AX,CR          ; ASSUME FIRST PRINTER (DX= 0)
513          INT 17H           ; GET THE PRINT CHARACTER COMMAND AND
514          MOV AX,LF          ; THE CARRIAGE RETURN CHARACTER
515          INT 17H           ; NOW GET THE LINE FEED AND
516          RET               ; SEND IT TO THE BIOS PRINTER ROUTINE
517 0294          CRLF ENDP
518
519
520
521          ;-- HARDWARE INT 08 H -- ( IRQ LEVEL 0 ) -----
522          ;
523          ; THIS ROUTINE HANDLES THE TIMER INTERRUPT FROM CHANNEL 0 OF
524          ; THE 8254 TIMER. INPUT FREQUENCY IS 1.19318 MHZ AND THE DIVISOR
525          ; IS 65536, RESULTING IN APPROXIMATELY 18.2 INTERRUPTS EVERY SECOND.
526          ;
527          ; THE INTERRUPT HANDLER MAINTAINS A COUNT (40:16C) OF INTERRUPTS SINCE
528          ; POWER ON TIME, WHICH MAY BE USED TO ESTABLISH TIME OF DAY.
529          ; THE INTERRUPT HANDLER ALSO DECREMENTS THE MOTOR CONTROL COUNT (40:40)
530          ; OF THE DISKETTE, AND WHEN IT EXPIRES, WILL TURN OFF THE
531          ; DISKETTE MOTOR(S), AND RESET THE MOTOR RUNNING FLAGS.
532          ; THE INTERRUPT HANDLER WILL ALSO INVOKE A USER ROUTINE THROUGH
533          ; INTERRUPT ICH AT EVERY TIME TICK. THE USER MUST CODE A
534          ; ROUTINE AND PLACE THE CORRECT ADDRESS IN THE VECTOR TABLE.
535          ;-----
536
537 0294          TIMER_INT_1 PROC FAR
538          STI                ; INTERRUPTS BACK ON
539          PUSH DS
540          PUSH AX
541          CALL DOS            ; SAVE MACHINE STATE
542          INC *TIMER_LOW     ; ESTABLISH ADDRESSABILITY
543          INC *TIMER_HIGH    ; INCREMENT TIME
544          JNZ T4             ; GO TO TEST DAY
545          INC *TIMER_HIGH    ; INCREMENT HIGH WORD OF TIME
546          CMP *TIMER_HIGH,010H ; TEST DAY
547          JNZ T5             ; TEST FOR COUNT EQUALING 24 HOURS
548          CMP *TIMER_LOW,000H ; GO TO DISKETTE_CTL
549          JNZ T5             ; GO TO DISKETTE_CTL
550          ;
551          ;----- TIMER HAS GONE 24 HOURS
552          ;
553          SUB AX,AX
554          MOV *TIMER_HIGH,AX
555          MOV *TIMER_LOW,AX
556          MOV *TIMER_CTL,1
557          ;
558          ;----- TEST FOR DISKETTE TIME OUT
559          ;
560          ;
561          ;
562          ;
563          ;
564          ;
565          ;
566          ;
567          ;
568          ;
569          ;
570          ;
571          ;
572          ;
573          ;
574          ;
575          ;
576          ;
577          ;
578          ;
579          ;
580          ;
581          ;
582          ;
583          ;
584          ;
585          ;
586          ;
587          ;
588          ;
589          ;
590          ;
591          ;
592          ;
593          ;
594          ;
595          ;
596          ;
597          ;
598          ;
599          ;
600          ;
601          ;
602          ;
603          ;
604          ;
605          ;
606          ;
607          ;
608          ;
609          ;
610          ;
611          ;
612          ;
613          ;
614          ;
615          ;
616          ;
617          ;
618          ;
619          ;
620          ;
621          ;
622          ;
623          ;
624          ;
625          ;
626          ;
627          ;
628          ;
629          ;
630          ;
631          ;
632          ;
633          ;
634          ;
635          ;
636          ;
637          ;
638          ;
639          ;
640          ;
641          ;
642          ;
643          ;
644          ;
645          ;
646          ;
647          ;
648          ;
649          ;
650          ;
651          ;
652          ;
653          ;
654          ;
655          ;
656          ;
657          ;
658          ;
659          ;
660          ;
661          ;
662          ;
663          ;
664          ;
665          ;
666          ;
667          ;
668          ;
669          ;
670          ;
671          ;
672          ;
673          ;
674          ;
675          ;
676          ;
677          ;
678          ;
679          ;
680          ;
681          ;
682          ;
683          ;
684          ;
685          ;
686          ;
687          ;
688          ;
689          ;
690          ;
691          ;
692          ;
693          ;
694          ;
695          ;
696          ;
697          ;
698          ;
699          ;
700          ;
701          ;
702          ;
703          ;
704          ;
705          ;
706          ;
707          ;
708          ;
709          ;
710          ;
711          ;
712          ;
713          ;
714          ;
715          ;
716          ;
717          ;
718          ;
719          ;
720          ;
721          ;
722          ;
723          ;
724          ;
725          ;
726          ;
727          ;
728          ;
729          ;
730          ;
731          ;
732          ;
733          ;
734          ;
735          ;
736          ;
737          ;
738          ;
739          ;
740          ;
741          ;
742          ;
743          ;
744          ;
745          ;
746          ;
747          ;
748          ;
749          ;
750          ;
751          ;
752          ;
753          ;
754          ;
755          ;
756          ;
757          ;
758          ;
759          ;
760          ;
761          ;
762          ;
763          ;
764          ;
765          ;
766          ;
767          ;
768          ;
769          ;
770          ;
771          ;
772          ;
773          ;
774          ;
775          ;
776          ;
777          ;
778          ;
779          ;
780          ;
781          ;
782          ;
783          ;
784          ;
785          ;
786          ;
787          ;
788          ;
789          ;
790          ;
791          ;
792          ;
793          ;
794          ;
795          ;
796          ;
797          ;
798          ;
799          ;
800          ;
801          ;
802          ;
803          ;
804          ;
805          ;
806          ;
807          ;
808          ;
809          ;
810          ;
811          ;
812          ;
813          ;
814          ;
815          ;
816          ;
817          ;
818          ;
819          ;
820          ;
821          ;
822          ;
823          ;
824          ;
825          ;
826          ;
827          ;
828          ;
829          ;
830          ;
831          ;
832          ;
833          ;
834          ;
835          ;
836          ;
837          ;
838          ;
839          ;
840          ;
841          ;
842          ;
843          ;
844          ;
845          ;
846          ;
847          ;
848          ;
849          ;
850          ;
851          ;
852          ;
853          ;
854          ;
855          ;
856          ;
857          ;
858          ;
859          ;
860          ;
861          ;
862          ;
863          ;
864          ;
865          ;
866          ;
867          ;
868          ;
869          ;
870          ;
871          ;
872          ;
873          ;
874          ;
875          ;
876          ;
877          ;
878          ;
879          ;
880          ;
881          ;
882          ;
883          ;
884          ;
885          ;
886          ;
887          ;
888          ;
889          ;
890          ;
891          ;
892          ;
893          ;
894          ;
895          ;
896          ;
897          ;
898          ;
899          ;
900          ;
901          ;
902          ;
903          ;
904          ;
905          ;
906          ;
907          ;
908          ;
909          ;
910          ;
911          ;
912          ;
913          ;
914          ;
915          ;
916          ;
917          ;
918          ;
919          ;
920          ;
921          ;
922          ;
923          ;
924          ;
925          ;
926          ;
927          ;
928          ;
929          ;
930          ;
931          ;
932          ;
933          ;
934          ;
935          ;
936          ;
937          ;
938          ;
939          ;
940          ;
941          ;
942          ;
943          ;
944          ;
945          ;
946          ;
947          ;
948          ;
949          ;
950          ;
951          ;
952          ;
953          ;
954          ;
955          ;
956          ;
957          ;
958          ;
959          ;
960          ;
961          ;
962          ;
963          ;
964          ;
965          ;
966          ;
967          ;
968          ;
969          ;
970          ;
971          ;
972          ;
973          ;
974          ;
975          ;
976          ;
977          ;
978          ;
979          ;
980          ;
981          ;
982          ;
983          ;
984          ;
985          ;
986          ;
987          ;
988          ;
989          ;
990          ;
991          ;
992          ;
993          ;
994          ;
995          ;
996          ;
997          ;
998          ;
999          ;
1000         ;
    
```

```

1      PAGE 118,121
2      TITLE ORGS ----- 04/21/86 COMPATIBILITY MODULE
3      .LIST
4      0000      CODE      SEGMENT BYTE PUBLIC
5
6      PUBLIC   A1
7      PUBLIC   CONF_TBL
8      PUBLIC   CRT_CHAR_GEN
9      PUBLIC   D1
10     PUBLIC   D2
11     PUBLIC   D2A
12     PUBLIC   DISK_BASE
13     PUBLIC   DUMMY_RETURN
14     PUBLIC   E101
15     PUBLIC   E102
16     PUBLIC   E103
17     PUBLIC   E104
18     PUBLIC   E105
19     PUBLIC   E106
20     PUBLIC   E107
21     PUBLIC   E108
22     PUBLIC   E109
23     PUBLIC   E161
24     PUBLIC   E162
25     PUBLIC   E163
26     PUBLIC   E164
27     PUBLIC   E201
28     PUBLIC   E202
29     PUBLIC   E203
30     PUBLIC   E301
31     PUBLIC   E302
32     PUBLIC   E303
33     PUBLIC   E304
34     PUBLIC   E401
35     PUBLIC   E501
36     PUBLIC   E601
37     PUBLIC   E602
38     PUBLIC   F1780
39     PUBLIC   F1781
40     PUBLIC   F1782
41     PUBLIC   F1790
42     PUBLIC   F1791
43     PUBLIC   F31
44     PUBLIC   F3D
45     PUBLIC   F3D1
46     PUBLIC   FD_TBL
47     PUBLIC   FLOPPY
48     PUBLIC   HRD
49     PUBLIC   K6
50     PUBLIC   K6L
51     PUBLIC   K7
52     PUBLIC   K8
53     PUBLIC   K10
54     PUBLIC   K11
55     PUBLIC   K12
56     PUBLIC   K14
57     PUBLIC   K15
58     PUBLIC   M4
59     PUBLIC   M5
60     PUBLIC   M6
61     PUBLIC   NT
62     PUBLIC   NMI_INT
63     PUBLIC   PRINT_SCREEN
64     PUBLIC   PDR
65     PUBLIC   SEGS_I
66     PUBLIC   SLAVE_VECTOR_TABLE
67     PUBLIC   TUTOR
68     PUBLIC   VECTOR_TABLE
69     PUBLIC   VIDEO_FARMS
70
71     EXTRN   BOOT_STRAP_1:NEAR
72     EXTRN   CASSETTE_IO_1:NEAR
73     EXTRN   D11:NEAR
74     EXTRN   DISK_INT_1:NEAR
75     EXTRN   DISK_SETUP:NEAR
76     EXTRN   DISKETTE_IO_1:NEAR
77     EXTRN   DSKETTE_SETUP:NEAR
78     EXTRN   EQUIPMENT_1:NEAR
79     EXTRN   INT_281:NEAR
80     EXTRN   K16:NEAR
81     EXTRN   KEYBOARD_IO_1:NEAR
82     EXTRN   KB_INT_1:NEAR
83     EXTRN   MEMORY_SIZE_DET_1:NEAR
84     EXTRN   NMI_INT_1:NEAR
85     EXTRN   PRINT_SCREEN_1:NEAR
86     EXTRN   PRINTER_IO_1:NEAR
87     EXTRN   RE_DIRECT:NEAR
88     EXTRN   RS232_IO_1:NEAR
89     EXTRN   RTC_INT:NEAR
90     EXTRN   SEGS:NEAR
91     EXTRN   START_1:NEAR
92     EXTRN   TIME_OF_DAY_1:NEAR
93     EXTRN   TIMER_INT_1:NEAR
94     EXTRN   VIDEO_IO_T:NEAR
95
96     ASSUME  CS:CODE,DS:DATA
97
98     -----
99
100    ; THIS MODULE HAS BEEN ADDED TO FACILITATE THE EXPANSION OF THIS PROGRAM.
101    ; IT ALLOWS FOR THE FIXED ORG STATEMENT ENTRY POINTS THAT HAVE TO REMAIN
102    ; AT THE SAME ADDRESSES. THE USE OF ENTRY POINTS AND TABLES WITHIN THIS
103    ; MODULE SHOULD BE AVOIDED AND ARE INCLUDED ONLY TO SUPPORT EXISTING CODE
104    ; THAT VIOLATE THE STRUCTURE AND DESIGN OF BIOS. ALL BIOS ACCESS SHOULD
105    ; USE THE DOCUMENTED INTERRUPT VECTOR INTERFACE FOR COMPATIBILITY.
106    ;
107

```

SECTION 5

```

108                                     PAGE
109                                     |-----|
110                                     |  COPYRIGHT NOTICE  |
111                                     |-----|
112                                     |*--  ORG  0E000H  |
113 0000                                |  ORG  00000H  |
114
115 0000 37 38 58 37 34 36              DB  '78X7462 COPR. IBM 1981, 1986  '
116      32 20 43 4F 50 52
117      2E 20 49 42 4D 20
118      31 39 38 31 2C 20
119      31 39 38 36 20 20
120      20 20
121
122                                     |-----|
123                                     |  PARITY ERROR MESSAGES  |
124                                     |-----|
125
126 0020 60 41 52 49 54 59              D1  DB  'PARITY CHECK 1',CR,LF ; PLANAR BOARD PARITY CHECK LATCH SET
127      20 43 48 45 43 4B
128      20 31 0D 0A
129 0030 50 41 52 49 54 59              D2  DB  'PARITY CHECK 2',CR,LF ; I/O CHANNEL CHECK LATCH SET
130      20 43 48 45 43 4B
131      20 32 0D 0A
132 0040 3F 3F 3F 3F 3F 0D              D2A DB  '?????',CR,LF
133      0A
134 = 0047                                IP  =  $
135      79 73 74 65 6D 20              |*--  ORG  0E05BH
136 005B                                |*--  ORG  0005BH
137 005B                                RESETE JUMP  START_1 ; RESET START
138 005B E9 0000 E                       ; VECTOR ON TO THE MOVED POST CODE
139
140                                     |-----|
141                                     |  POST ERROR MESSAGES  |
142                                     |-----|
143
144 005E 20 31 30 31 2D 53              E101 DB  ' 101-System Board Error',CR,LF ; INTERRUPT FAILURE
145      79 73 74 65 6D 20
146      42 6F 61 72 64 20
147      45 72 72 6F 72 0D
148      0A
149 0077 20 31 30 32 2D 53              E102 DB  ' 102-System Board Error',CR,LF ; TIMER FAILURE
150      79 73 74 65 6D 20
151      42 6F 61 72 64 20
152      45 72 72 6F 72 0D
153      0A
154 0090 20 31 30 33 2D 53              E103 DB  ' 103-System Board Error',CR,LF ; TIMER INTERRUPT FAILURE
155      79 73 74 65 6D 20
156      42 6F 61 72 64 20
157      45 72 72 6F 72 0D
158      0A
159 00A9 20 31 30 34 2D 53              E104 DB  ' 104-System Board Error',CR,LF ; PROTECTED MODE FAILURE
160      79 73 74 65 6D 20
161      42 6F 61 72 64 20
162      45 72 72 6F 72 0D
163      0A
164 00C2 20 31 30 35 2D 53              E105 DB  ' 105-System Board Error',CR,LF ; LAST 8042 COMMAND NOT ACCEPTED
165      79 73 74 65 6D 20
166      42 6F 61 72 64 20
167      45 72 72 6F 72 0D
168      0A
169 00DB 20 31 30 36 2D 53              E106 DB  ' 106-System Board Error',CR,LF ; CONVERTING LOGIC TEST
170      79 73 74 65 6D 20
171      42 6F 61 72 64 20
172      45 72 72 6F 72 0D
173      0A
174 00F4 20 31 30 37 2D 53              E107 DB  ' 107-System Board Error',CR,LF ; HOT NMI TEST
175      79 73 74 65 6D 20
176      42 6F 61 72 64 20
177      45 72 72 6F 72 0D
178      0A
179 010D 20 31 30 38 2D 53              E108 DB  ' 108-System Board Error',CR,LF ; TIMER BUS TEST
180      79 73 74 65 6D 20
181      42 6F 61 72 64 20
182      45 72 72 6F 72 0D
183      0A
184 0126 20 31 30 39 2D 53              E109 DB  ' 109-System Board Error',CR,LF ; LOW MEG CHIP SELECT TEST
185      79 73 74 65 6D 20
186      42 6F 61 72 64 20
187      45 72 72 6F 72 0D
188      0A
189 013F 20 31 36 31 2D 53              E161 DB  ' 161-System Options Not Set-(Run SETUP)',CR,LF ; DEAD BATTERY
190      79 73 74 65 6D 20
191      4F 70 74 69 6F 6E
192      73 20 4E 6F 74 20
193      68 68 74 2D 28 52
194      75 6E 20 53 48 84
195      65 60 29 0D 0A
196 0168 20 31 36 32 2D 53              E162 DB  ' 162-System Options Not Set-(Run SETUP)',CR,LF ;CHECKSUM/CONFIG
197      79 73 74 65 6D 20
198      4F 70 74 69 6F 6E
199      73 20 4E 6F 74 20
200      53 65 74 2D 28 52
201      75 6E 20 53 48 84
202      55 50 29 0D 0A
203 0191 20 31 36 33 2D 54              E163 DB  ' 163-Time & Date Not Set-(Run SETUP)',CR,LF ;CLOCK NOT UPDATING
204      69 6D 65 20 26 20
205      44 61 74 65 20 4E
206      6F 74 20 53 65 74
207      2D 28 52 75 6E 20
208      53 45 54 65 50 29
209      0D 0A
210 01B7 20 31 36 34 2D 4D              E164 DB  ' 164-Memory Size Error-(Run SETUP)',CR,LF ; CMOS DOES NOT MATCH
211      65 6D 6F 72 79 20
212      53 69 74 65 20 45
213      72 72 6F 72 2D 28
214      52 75 6E 20 53 45
215      54 55 50 29 0D 0A
216 01DB 20 32 30 31 2D 4D              E201 DB  ' 201-Memory Error',CR,LF
217      65 6D 6F 72 79 20
218      45 72 72 6F 72 0D
219      0A
220 01EE 20 32 30 32 2D 4D              E202 DB  ' 202-Memory Address Error',CR,LF ; LINE ERROR 00->15
221      65 6D 6F 72 79 20
  
```

```

222      41 64 64 72 65 73
223      73 20 45 72 72 6F
224      72 0D 0A
225 0209 20 32 30 33 2D 4D E203 DB ' 203-Memory Address Error',CR,LF ; LINE ERROR 16->23
226      65 6D 6F 72 79 20
227      41 64 64 72 65 73
228      73 20 45 72 72 6F
229      72 0D 0A
230 0224 20 33 30 31 2D 4B E301 DB ' 301-Keyboard Error',CR,LF ; KEYBOARD ERROR
231      65 79 62 6F 61 72
232      64 20 45 72 72 6F
233      72 0D 0A
234 0239 20 33 30 32 2D 53 E302 DB ' 302-System Unit Keylock is Locked',CR,LF ; KEYBOARD LOCK ON
235      79 73 74 65 6D 20
236      55 6E 69 74 20 4B
237      65 79 6C 6F 63 6B
238      20 69 73 20 4C 6F
239      63 6B 64 0D 0A
240 025D 20 28 52 45 53 55 F3D DB ' (RESUME = 'F1' KEY)',CR,LF
241      4D 45 20 3D 20 22
242      46 31 22 20 4B 45
243      59 29 0D 0A
244
245      ;----- NMI ENTRY
246
247      = 0273
248
249 02C3
250 = 02C3
251 02C3 E9 0000 E
252
253 02C6 20 33 30 33 2D 4B E303 DB ' 303-Keyboard Or System Unit Error',CR,LF
254      65 79 62 6F 61 72
255      64 20 4F 72 20 53
256      79 73 74 65 6D 20
257      55 6E 69 74 20 45
258      72 72 6F 72 0D 0A
259
260 02EA 20 33 30 34 2D 4B E304 DB ' 304-Keyboard Or System Unit Error',CR,LF ; KEYBOARD CLOCK HIGH
261      65 79 62 6F 61 72
262      64 20 4F 72 20 53
263      79 73 74 65 6D 20
264      55 6E 69 74 20 45
265      72 72 6F 72 0D 0A
266 030E 20 34 30 31 2D 43 E401 DB ' 401-CRT Error',CR,LF ; MONOCHROME
267      52 54 20 45 72 72
268      6F 72 0D 0A
269 031E 20 35 30 31 2D 43 E501 DB ' 501-CRT Error',CR,LF ; COLOR
270      52 54 20 45 72 72
271      6F 72 0D 0A
272 032E 20 36 30 31 2D 44 E601 DB ' 601-Diskette Error',CR,LF ; DISKETTE ERROR
273      69 73 6B 65 74 74
274      65 20 45 72 72 6F
275      72 0D 0A
276
277 0343 20 36 30 32 2D 44 E602 DB ' 602-Diskette Boot Record Error',CR,LF
278      69 73 6B 65 74 74
279      65 20 42 6F 6F 74
280      20 52 65 63 6F 72
281      64 20 45 72 72 6F
282      72 0D 0A
283
284 0364 31 37 38 30 2D 44 F1780 DB '1780-Disk 0 Failure',CR,LF
285      69 73 6B 20 30 20
286      46 61 69 6C 75 72
287      65 0D 0A
288 0379 31 37 38 31 2D 44 F1781 DB '1781-Disk 1 Failure',CR,LF
289      69 73 6B 20 31 20
290      46 61 69 6C 75 72
291      65 0D 0A
292 038E 31 37 38 32 2D 44 F1782 DB '1782-Disk Controller Failure',CR,LF
293      69 73 6B 20 43 6F
294      6E 74 72 6F 6C 6C
295      65 72 20 46 61 69
296      6C 75 72 65 0D 0A
297 03AC 31 37 39 30 2D 44 F1790 DB '1790-Disk 0 Error',CR,LF
298      69 73 6B 20 30 20
299      45 72 72 6F 72 0D
300      0A
301 03BF 31 37 39 31 2D 44 F1791 DB '1791-Disk 1 Error',CR,LF
302      69 73 6B 20 31 20
303      45 72 72 6F 72 0D
304      0A
305
306 03D2 52 4F 4D 20 2D 45 F3A DB 'ROM Error ',CR,LF ; ROM CHECKSUM
307      72 72 6F 72 20 0D
308      0A
309 03DF 20 20 20 2D 55 F3D1 DB ' -Unlock System Unit Keylock ',CR,LF
310      6E 6C 6F 63 6B 20
311      53 79 73 74 65 6D
312      20 55 6E 69 74 20
313      4B 65 79 6C 6F 63
314      6B 20 0D 0A

```

SECTION 5

```

315 PAGE
316 -----
317 | INITIALIZE DRIVE CHARACTERISTICS |
318 | |
319 | FIXED DISK PARAMETER TABLE |
320 | |
321 | - THE TABLE IS COMPOSED OF A BLOCK DEFINED AS: |
322 | |
323 | +0 (1 WORD) - MAXIMUM NUMBER OF CYLINDERS |
324 | +2 (1 BYTE) - MAXIMUM NUMBER OF HEADS |
325 | +3 (1 WORD) - NOT USED/SEE PC-XT |
326 | +5 (1 WORD) - STARTING WRITE PRECOMPENSATION CYL |
327 | +7 (1 BYTE) - NOT USED/SEE PC-XT |
328 | +8 (1 BYTE) - CONTROL BYTE |
329 | |
330 | BIT 7 DISABLE RETRIES -OR- |
331 | BIT 6 DISABLE RETRIES |
332 | +9 (3 BYTES) - NOT USED/SEE PC-XT |
333 | +12 (1 WORD) - LANDING ZONE |
334 | +14 (1 BYTE) - NUMBER OF SECTORS/TRACK |
335 | +15 (1 BYTE) - RESERVED FOR FUTURE USE |
336 | |
337 | - TO DYNAMICALLY DEFINE A SET OF PARAMETERS |
338 | BUILD A TABLE FOR UP TO 15 TYPES AND PLACE |
339 | THE CORRESPONDING VECTOR INTO INTERRUPT 41 |
340 | FOR DRIVE 0 AND INTERRUPT 46 FOR DRIVE 1. |
341 | |
342 |-----|
343
344 0401 FD_TBL:
345
346 |----- DRIVE TYPE 01
347
348 0401 0132 DW 0306D ; CYLINDERS
349 0403 04 DB 04D ; HEADS
350 0404 0000 DW 0 ;
351 0406 0080 DW 0126D ; WRITE PRE-COMPENSATION CYLINDER
352 0408 00 DB 0 ;
353 0409 00 DB 0 ; CONTROL BYTE
354 040A 00 00 00 DB 0,0,0 ;
355 040D 0131 DW 0305D ; LANDING ZONE
356 040F 11 DB 17D ; SECTORS/TRACK
357 0410 00 DB 0 ;
358
359 |----- DRIVE TYPE 02
360
361 0411 0267 DW 0615D ; CYLINDERS
362 0413 04 DB 04D ; HEADS
363 0414 0000 DW 0 ;
364 0416 012C DW 0300D ; WRITE PRE-COMPENSATION CYLINDER
365 0418 00 DB 0 ;
366 0419 00 DB 0 ; CONTROL BYTE
367 041A 00 00 00 DB 0,0,0 ;
368 041D 0267 DW 0615D ; LANDING ZONE
369 041F 11 DB 17D ; SECTORS/TRACK
370 0420 00 DB 0 ;
371
372 |----- DRIVE TYPE 03
373
374 0421 0267 DW 0615D ; CYLINDERS
375 0423 06 DB 06D ; HEADS
376 0424 0000 DW 0 ;
377 0426 012C DW 0300D ; WRITE PRE-COMPENSATION CYLINDER
378 0428 00 DB 0 ;
379 0429 00 DB 0 ; CONTROL BYTE
380 042A 00 00 00 DB 0,0,0 ;
381 042D 0267 DW 0615D ; LANDING ZONE
382 042F 11 DB 17D ; SECTORS/TRACK
383 0430 00 DB 0 ;
384
385 |----- DRIVE TYPE 04
386
387 0431 03AC DW 0940D ; CYLINDERS
388 0433 08 DB 08D ; HEADS
389 0434 0000 DW 0 ;
390 0436 0200 DW 0512D ; WRITE PRE-COMPENSATION CYLINDER
391 0438 00 DB 0 ;
392 0439 00 DB 0 ; CONTROL BYTE
393 043A 00 00 00 DB 0,0,0 ;
394 043D 03AC DW 0940D ; LANDING ZONE
395 043F 11 DB 17D ; SECTORS/TRACK
396 0440 00 DB 0 ;
397
398 |----- DRIVE TYPE 05
399
400 0441 03AC DW 0940D ; CYLINDERS
401 0443 06 DB 06D ; HEADS
402 0444 0000 DW 0 ;
403 0446 0200 DW 0512D ; WRITE PRE-COMPENSATION CYLINDER
404 0448 00 DB 0 ;
405 0449 00 DB 0 ; CONTROL BYTE
406 044A 00 00 00 DB 0,0,0 ;
407 044D 03AC DW 0940D ; LANDING ZONE
408 044F 11 DB 17D ; SECTORS/TRACK
409 0450 00 DB 0 ;
410
411 |----- DRIVE TYPE 06
412
413 0451 0267 DW 0615D ; CYLINDERS
414 0453 04 DB 04D ; HEADS
415 0454 0000 DW 0 ;
416 0456 FFFF DW 0FFFFH ; NO WRITE PRE-COMPENSATION
417 0458 00 DB 0 ;
418 0459 00 DB 0 ; CONTROL BYTE
419 045A 00 00 00 DB 0,0,0 ;
420 045D 0267 DW 0615D ; LANDING ZONE
421 045F 11 DB 17D ; SECTORS/TRACK
422 0460 00 DB 0 ;

```

```

423                                     PAGE
424                                     |-----
425                                     |----- DRIVE TYPE 07
426 0461 01CE                            DW 0462D                | CYLINDERS
427 0463 08                              DB 08D                 | HEADS
428 0464 0000                            DW 0                   | NO WRITE PRE-COMPENSATION
429 0466 0100                            DW 0256D              | WRITE PRE-COMPENSATION CYLINDER
430 0468 00                              DB 0                   | CONTROL BYTE
431 0469 00                              DB 0,0,0              | CONTROL BYTE
432 046A 00 00 00                        DB 0,0,0              | CONTROL BYTE
433 046D 01FF                            DW 0511D              | LANDING ZONE
434 046F 11                              DB 17D                | SECTORS/TRACK
435 0470 00                              DB 0                   | SECTORS/TRACK
436
437                                     |----- DRIVE TYPE 08
438
439 0471 02DD                            DW 0733D              | CYLINDERS
440 0473 05                              DB 05D                | HEADS
441 0474 0000                            DW 0                   | NO WRITE PRE-COMPENSATION
442 0476 FFFF                            DB 0FFFFH            | NO WRITE PRE-COMPENSATION
443 0478 00                              DB 0                   | CONTROL BYTE
444 0479 00                              DB 0                   | CONTROL BYTE
445 047A 00 00 00                        DB 0,0,0              | CONTROL BYTE
446 047D 02DD                            DW 0733D              | LANDING ZONE
447 047F 11                              DB 17D                | SECTORS/TRACK
448 0480 00                              DB 0                   | SECTORS/TRACK
449
450                                     |----- DRIVE TYPE 09
451
452 0481 0364                            DW 0900D              | CYLINDERS
453 0483 0F                              DB 15D                | HEADS
454 0484 0000                            DW 0                   | NO WRITE PRE-COMPENSATION
455 0486 FFFF                            DB 0FFFFH            | NO WRITE PRE-COMPENSATION
456 0488 00                              DB 0                   | CONTROL BYTE
457 0489 08                              DB 008H               | CONTROL BYTE
458 048A 00 00 00                        DB 0,0,0              | CONTROL BYTE
459 048D 0365                            DW 0901D              | LANDING ZONE
460 048F 11                              DB 17D                | SECTORS/TRACK
461 0490 00                              DB 0                   | SECTORS/TRACK
462
463                                     |----- DRIVE TYPE 10
464
465 0491 0334                            DW 0820D              | CYLINDERS
466 0493 03                              DB 03D                | HEADS
467 0494 0000                            DW 0                   | NO WRITE PRE-COMPENSATION
468 0496 FFFF                            DB 0FFFFH            | NO WRITE PRE-COMPENSATION
469 0498 00                              DB 0                   | CONTROL BYTE
470 0499 00                              DB 0                   | CONTROL BYTE
471 049A 00 00 00                        DB 0,0,0              | CONTROL BYTE
472 049D 0334                            DW 0820D              | LANDING ZONE
473 049F 11                              DB 17D                | SECTORS/TRACK
474 04A0 00                              DB 0                   | SECTORS/TRACK
475
476                                     |----- DRIVE TYPE 11
477
478 04A1 0357                            DW 0855D              | CYLINDERS
479 04A3 05                              DB 05D                | HEADS
480 04A4 0000                            DW 0                   | NO WRITE PRE-COMPENSATION
481 04A6 FFFF                            DB 0FFFFH            | NO WRITE PRE-COMPENSATION
482 04A8 00                              DB 0                   | CONTROL BYTE
483 04A9 00                              DB 0                   | CONTROL BYTE
484 04AA 00 00 00                        DB 0,0,0              | CONTROL BYTE
485 04AD 0357                            DW 0855D              | LANDING ZONE
486 04AF 11                              DB 17D                | SECTORS/TRACK
487 04B0 00                              DB 0                   | SECTORS/TRACK
488
489                                     |----- DRIVE TYPE 12
490
491 04B1 0357                            DW 0855D              | CYLINDERS
492 04B3 07                              DB 07D                | HEADS
493 04B4 0000                            DW 0                   | NO WRITE PRE-COMPENSATION
494 04B6 FFFF                            DB 0FFFFH            | NO WRITE PRE-COMPENSATION
495 04B8 00                              DB 0                   | CONTROL BYTE
496 04B9 00                              DB 0                   | CONTROL BYTE
497 04BA 00 00 00                        DB 0,0,0              | CONTROL BYTE
498 04BD 0357                            DW 0855D              | LANDING ZONE
499 04BF 11                              DB 17D                | SECTORS/TRACK
500 04C0 00                              DB 0                   | SECTORS/TRACK
501
502                                     |----- DRIVE TYPE 13
503
504 04C1 0132                            DW 0306D              | CYLINDERS
505 04C3 08                              DB 08D                | HEADS
506 04C4 0000                            DW 0                   | NO WRITE PRE-COMPENSATION
507 04C6 0080                            DW 0128D              | WRITE PRE-COMPENSATION CYLINDER
508 04C8 00                              DB 0                   | CONTROL BYTE
509 04C9 00                              DB 0                   | CONTROL BYTE
510 04CA 00 00 00                        DB 0,0,0              | CONTROL BYTE
511 04CD 013F                            DW 0319D              | LANDING ZONE
512 04CF 11                              DB 17D                | SECTORS/TRACK
513 04D0 00                              DB 0                   | SECTORS/TRACK
514
515                                     |----- DRIVE TYPE 14
516
517 04D1 02DD                            DW 0733D              | CYLINDERS
518 04D3 07                              DB 07D                | HEADS
519 04D4 0000                            DW 0                   | NO WRITE PRE-COMPENSATION
520 04D6 FFFF                            DB 0FFFFH            | NO WRITE PRE-COMPENSATION
521 04D8 00                              DB 0                   | CONTROL BYTE
522 04D9 00                              DB 0                   | CONTROL BYTE
523 04DA 00 00 00                        DB 0,0,0              | CONTROL BYTE
524 04DD 02DD                            DW 0733D              | LANDING ZONE
525 04DF 11                              DB 17D                | SECTORS/TRACK
526 04E0 00                              DB 0                   | SECTORS/TRACK

```

SECTION 5

```

527                                     PAGE
528                                     |----- DRIVE TYPE 15   RESERVED   **** DO NOT USE****
529
530 04E1 0000                            DW 0000D           | CYLINDERS
531 04E3 00                              DB 00D            | HEADS
532 04E4 0000                            DW 0              |
533 04E6 0000                            DW 0000D         | WRITE PRE-COMPENSATION CYLINDER
534 04E8 00                              DB 0              |
535 04E9 00                              DB 0              | CONTROL BYTE
536 04EA 00 00 00                        DB 0,0,0         |
537 04ED 0000                            DW 0000D         | LANDING ZONE
538 04EF 00                              DB 00D           | SECTORS/TRACK
539 04F0 00                              DB 0              |
540
541                                     |----- DRIVE TYPE 16
542
543 04F1 0264                            DW 0612D         | CYLINDERS
544 04F3 04                              DB 04D           | HEADS
545 04F4 0000                            DW 0              |
546 04F6 0000                            DW 0000D         | WRITE PRE-COMPENSATION ALL CYLINDER
547 04F8 00                              DB 0              |
548 04F9 00                              DB 0              | CONTROL BYTE
549 04FA 00 00 00                        DB 0,0,0         |
550 04FD 0297                            DW 0663D         | LANDING ZONE
551 04FF 11                              DB 17D           | SECTORS/TRACK
552 0500 00                              DB 0              |
553
554                                     |----- DRIVE TYPE 17
555
556 0501 03D1                            DW 0977D         | CYLINDERS
557 0503 05                              DB 05D           | HEADS
558 0504 0000                            DW 0              |
559 0506 012C                            DW 0300D         | WRITE PRE-COMPENSATION CYL
560 0508 00                              DB 0              |
561 0509 00                              DB 0              | CONTROL BYTE
562 050A 00 00 00                        DB 0,0,0         |
563 050D 03D1                            DW 0977D         | LANDING ZONE
564 050F 11                              DB 17D           | SECTORS/TRACK
565 0510 00                              DB 0              |
566
567                                     |----- DRIVE TYPE 18
568
569 0511 03D1                            DW 0977D         | CYLINDERS
570 0513 07                              DB 07D           | HEADS
571 0514 0000                            DW 0              |
572 0516 FFFF                            DW 0FFFFFFH      | NO WRITE PRE-COMPENSATION
573 0518 00                              DB 0              |
574 0519 00                              DB 0              | CONTROL BYTE
575 051A 00 00 00                        DB 0,0,0         |
576 051D 03D1                            DW 0977D         | LANDING ZONE
577 051F 11                              DB 17D           | SECTORS/TRACK
578 0520 00                              DB 0              |
579
580                                     |----- DRIVE TYPE 19
581
582 0521 0400                            DW 1024D         | CYLINDERS
583 0523 07                              DB 07D           | HEADS
584 0524 0000                            DW 0              |
585 0526 0200                            DW 0512D         | WRITE PRE-COMPENSATION CYLINDER
586 0528 00                              DB 0              |
587 0529 00                              DB 0              | CONTROL BYTE
588 052A 00 00 00                        DB 0,0,0         |
589 052D 03FF                            DW 1023D         | LANDING ZONE
590 052F 11                              DB 17D           | SECTORS/TRACK
591 0530 00                              DB 0              |
592
593                                     |----- DRIVE TYPE 20
594
595 0531 02DD                            DW 0733D         | CYLINDERS
596 0533 05                              DB 05D           | HEADS
597 0534 0000                            DW 0              |
598 0536 012C                            DW 0300D         | WRITE PRE-COMPENSATION CYL
599 0538 00                              DB 0              |
600 0539 00                              DB 0              | CONTROL BYTE
601 053A 00 00 00                        DB 0,0,0         |
602 053D 02DC                            DW 0732D         | LANDING ZONE
603 053F 11                              DB 17D           | SECTORS/TRACK
604 0540 00                              DB 0              |
605
606                                     |----- DRIVE TYPE 21
607
608 0541 02DD                            DW 0733D         | CYLINDERS
609 0543 07                              DB 07D           | HEADS
610 0544 0000                            DW 0              |
611 0546 012C                            DW 0300D         | WRITE PRE-COMPENSATION CYL
612 0548 00                              DB 0              |
613 0549 00                              DB 0              | CONTROL BYTE
614 054A 00 00 00                        DB 0,0,0         |
615 054D 02DC                            DW 0732D         | LANDING ZONE
616 054F 11                              DB 17D           | SECTORS/TRACK
617 0550 00                              DB 0              |
618
619                                     |----- DRIVE TYPE 22
620
621 0551 02DD                            DW 0733D         | CYLINDERS
622 0553 05                              DB 05D           | HEADS
623 0554 0000                            DW 0              |
624 0556 012C                            DW 0300D         | WRITE PRE-COMPENSATION CYL
625 0558 00                              DB 0              |
626 0559 00                              DB 0              | CONTROL BYTE
627 055A 00 00 00                        DB 0,0,0         |
628 055D 02DD                            DW 0733D         | LANDING ZONE
629 055F 11                              DB 17D           | SECTORS/TRACK
630 0560 00                              DB 0              |

```



```

631          PAGE
632          ;----- DRIVE TYPE 23
633
634 0561 0132      DW 0306D      ; CYLINDERS
635 0563 04        DB 04D       ; HEADS
636 0564 0000      DW 0         ; WRITE PRE-COMPENSATION ALL CYL
637 0566 0000      DW 0000D      ;
638 0568 00        DB 0         ;
639 0569 00        DB 0         ; CONTROL BYTE
640 056A 00 00 00  DB 0,0,0     ;
641 056D 0160      DW 0336D      ; LANDING ZONE
642 056F 11        DB 17D       ; SECTORS/TRACK
643 0570 00        DB 0
644
645          ;----- DRIVE TYPE 24
646
647 0571 0264      DW 0612D      ; CYLINDERS
648 0573 04        DB 04D       ; HEADS
649 0574 0000      DW 0         ; WRITE PRE-COMPENSATION CYL
650 0576 0131      DW 0305D      ;
651 0578 00        DB 0         ;
652 0579 00        DB 0         ; CONTROL BYTE
653 057A 00 00 00  DB 0,0,0     ;
654 057D 0297      DW 0663D      ; LANDING ZONE
655 057F 11        DB 17D       ; SECTORS/TRACK
656 0580 00        DB 0
657
658          ;----- DRIVE TYPE 25 *** RESERVED***
659
660 0581 0000      DW 0000D      ; CYLINDERS
661 0583 00        DB 00D       ; HEADS
662 0584 0000      DW 0         ; WRITE PRE-COMPENSATION CYL
663 0586 0000      DW 0000D      ;
664 0588 00        DB 0         ;
665 0589 00        DB 0         ; CONTROL BYTE
666 058A 00 00 00  DB 0,0,0     ;
667 058D 0000      DW 0000D      ; LANDING ZONE
668 058F 00        DB 00D       ; SECTORS/TRACK
669 0590 00        DB 0
670
671          ;----- DRIVE TYPE 26 *** RESERVED***
672
673 0591 0000      DW 0000D      ; CYLINDERS
674 0593 00        DB 00D       ; HEADS
675 0594 0000      DW 0         ; WRITE PRE-COMPENSATION CYL
676 0596 0000      DW 0000D      ;
677 0598 00        DB 0         ;
678 0599 00        DB 0         ; CONTROL BYTE
679 059A 00 00 00  DB 0,0,0     ;
680 059D 0000      DW 0000D      ; LANDING ZONE
681 059F 00        DB 00D       ; SECTORS/TRACK
682 05A0 00        DB 0
683
684          ;----- DRIVE TYPE 27 *** RESERVED***
685
686 05A1 0000      DW 0000D      ; CYLINDERS
687 05A3 00        DB 00D       ; HEADS
688 05A4 0000      DW 0         ; WRITE PRE-COMPENSATION CYL
689 05A6 0000      DW 0000D      ;
690 05A8 00        DB 0         ;
691 05A9 00        DB 0         ; CONTROL BYTE
692 05AA 00 00 00  DB 0,0,0     ;
693 05AD 0000      DW 0000D      ; LANDING ZONE
694 05AF 00        DB 00D       ; SECTORS/TRACK
695 05B0 00        DB 0
696
697          ;----- DRIVE TYPE 28 *** RESERVED***
698
699 05B1 0000      DW 0000D      ; CYLINDERS
700 05B3 00        DB 00D       ; HEADS
701 05B4 0000      DW 0         ; WRITE PRE-COMPENSATION CYL
702 05B6 0000      DW 0000D      ;
703 05B8 00        DB 0         ;
704 05B9 00        DB 0         ; CONTROL BYTE
705 05BA 00 00 00  DB 0,0,0     ;
706 05BD 0000      DW 0000D      ; LANDING ZONE
707 05BF 00        DB 00D       ; SECTORS/TRACK
708 05C0 00        DB 0
709
710          ;----- DRIVE TYPE 29 *** RESERVED***
711
712 05C1 0000      DW 0000D      ; CYLINDERS
713 05C3 00        DB 00D       ; HEADS
714 05C4 0000      DW 0         ; WRITE PRE-COMPENSATION CYL
715 05C6 0000      DW 0000D      ;
716 05C8 00        DB 0         ;
717 05C9 00        DB 0         ; CONTROL BYTE
718 05CA 00 00 00  DB 0,0,0     ;
719 05CD 0000      DW 0000D      ; LANDING ZONE
720 05CF 00        DB 00D       ; SECTORS/TRACK
721 05D0 00        DB 0
722
723          ;----- DRIVE TYPE 30 *** RESERVED***
724
725 05D1 0000      DW 0000D      ; CYLINDERS
726 05D3 00        DB 00D       ; HEADS
727 05D4 0000      DW 0         ; WRITE PRE-COMPENSATION CYL
728 05D6 0000      DW 0000D      ;
729 05D8 00        DB 0         ;
730 05D9 00        DB 0         ; CONTROL BYTE
731 05DA 00 00 00  DB 0,0,0     ;
732 05DD 0000      DW 0000D      ; LANDING ZONE
733 05DF 00        DB 00D       ; SECTORS/TRACK
734 05E0 00        DB 0
  
```

SECTION 5

```

PAGE
736 |----- DRIVE TYPE 31 *** RESERVED***
737
738 05E1 0000 DF 0000D ; CYLINDERS
739 05E3 00 DB 00D ; HEADS
740 05E4 0000 DF 0 ;
741 05E5 0000 DF 0000D ; WRITE PRE-COMPENSATION CYL
742 05E8 00 DB 0 ;
743 05E9 00 DB 0 ; CONTROL BYTE
744 05EA 00 00 00 DB 0,0,0
745 05ED 0000 DF 0000D ; LANDING ZONE
746 05EF 00 DB 00D ; SECTORS/TRACK
747 05F0 00 DB 0
748
749 |----- DRIVE TYPE 32 *** RESERVED***
750
751 05F1 0000 DF 0000D ; CYLINDERS
752 05F3 00 DB 00D ; HEADS
753 05F4 0000 DF 0 ;
754 05F6 0000 DF 0000D ; WRITE PRE-COMPENSATION CYL
755 05F8 00 DB 0 ;
756 05F9 00 DB 0 ; CONTROL BYTE
757 05FA 00 00 00 DB 0,0,0
758 05FD 0000 DF 0000D ; LANDING ZONE
759 05FF 00 DB 00D ; SECTORS/TRACK
760 0600 00 DB 0
761
762 |----- DRIVE TYPE 33 *** RESERVED***
763
764 0601 0000 DF 0000D ; CYLINDERS
765 0603 00 DB 00D ; HEADS
766 0604 0000 DF 0 ;
767 0606 0000 DF 0000D ; WRITE PRE-COMPENSATION CYL
768 0608 00 DB 0 ;
769 0609 00 DB 0 ; CONTROL BYTE
770 060A 00 00 00 DB 0,0,0
771 060D 0000 DF 0000D ; LANDING ZONE
772 060F 00 DB 00D ; SECTORS/TRACK
773 0610 00 DB 0
774
775 |----- DRIVE TYPE 34 *** RESERVED***
776
777 0611 0000 DF 0000D ; CYLINDERS
778 0613 00 DB 00D ; HEADS
779 0614 0000 DF 0 ;
780 0616 0000 DF 0000D ; WRITE PRE-COMPENSATION CYL
781 0618 00 DB 0 ;
782 0619 00 DB 0 ; CONTROL BYTE
783 061A 00 00 00 DB 0,0,0
784 061D 0000 DF 0000D ; LANDING ZONE
785 061F 00 DB 00D ; SECTORS/TRACK
786 0620 00 DB 0
787
788 |----- DRIVE TYPE 35 *** RESERVED***
789
790 0621 0000 DF 0000D ; CYLINDERS
791 0623 00 DB 00D ; HEADS
792 0624 0000 DF 0 ;
793 0626 0000 DF 0000D ; WRITE PRE-COMPENSATION CYL
794 0628 00 DB 0 ;
795 0629 00 DB 0 ; CONTROL BYTE
796 062A 00 00 00 DB 0,0,0
797 062D 0000 DF 0000D ; LANDING ZONE
798 062F 00 DB 00D ; SECTORS/TRACK
799 0630 00 DB 0
800
801 |----- DRIVE TYPE 36 *** RESERVED***
802
803 0631 0000 DF 0000D ; CYLINDERS
804 0633 00 DB 00D ; HEADS
805 0634 0000 DF 0 ;
806 0636 0000 DF 0000D ; WRITE PRE-COMPENSATION CYL
807 0638 00 DB 0 ;
808 0639 00 DB 0 ; CONTROL BYTE
809 063A 00 00 00 DB 0,0,0
810 063D 0000 DF 0000D ; LANDING ZONE
811 063F 00 DB 00D ; SECTORS/TRACK
812 0640 00 DB 0
813
814 |----- DRIVE TYPE 37 *** RESERVED***
815
816 0641 0000 DF 0000D ; CYLINDERS
817 0643 00 DB 00D ; HEADS
818 0644 0000 DF 0 ;
819 0646 0000 DF 0000D ; WRITE PRE-COMPENSATION CYL
820 0648 00 DB 0 ;
821 0649 00 DB 0 ; CONTROL BYTE
822 064A 00 00 00 DB 0,0,0
823 064D 0000 DF 0000D ; LANDING ZONE
824 064F 00 DB 00D ; SECTORS/TRACK
825 0650 00 DB 0
826
827 |----- DRIVE TYPE 38 *** RESERVED***
828
829 0651 0000 DF 0000D ; CYLINDERS
830 0653 00 DB 00D ; HEADS
831 0654 0000 DF 0 ;
832 0656 0000 DF 0000D ; WRITE PRE-COMPENSATION CYL
833 0658 00 DB 0 ;
834 0659 00 DB 0 ; CONTROL BYTE
835 065A 00 00 00 DB 0,0,0
836 065D 0000 DF 0000D ; LANDING ZONE
837 065F 00 DB 00D ; SECTORS/TRACK
838 0660 00 DB 0
  
```

```

839          PAGE
840          |----- DRIVE TYPE 39      *** RESERVED***
841
842          DW 0000D          ; CYLINDERS
843          DB 00D           ; HEADS
844          DW 0000D          ; WRITE PRE-COMPENSATION CYL
845          DB 0             ; CONTROL BYTE
846          DB 0,0,0         ; LANDING ZONE
847          DB 00D          ; SECTORS/TRACK
848          DB 0
849          DB 0,0,0
850          DW 0000D
851          DB 00D
852          DB 0
853          |----- DRIVE TYPE 40      *** RESERVED***
854
855          DW 0000D          ; CYLINDERS
856          DB 00D           ; HEADS
857          DW 0             ; WRITE PRE-COMPENSATION CYL
858          DB 0000D          ; CONTROL BYTE
859          DB 0             ; LANDING ZONE
860          DB 0,0,0         ; SECTORS/TRACK
861          DB 0
862          DW 0000D
863          DB 00D
864          DB 0
865          |----- DRIVE TYPE 41      *** RESERVED***
866
867          DW 0000D          ; CYLINDERS
868          DB 00D           ; HEADS
869          DW 0             ; WRITE PRE-COMPENSATION CYL
870          DB 0000D          ; CONTROL BYTE
871          DB 0             ; LANDING ZONE
872          DB 0,0,0         ; SECTORS/TRACK
873          DW 0000D
874          DB 00D
875          DB 0
876          DW 0000D
877          DB 00D
878          DB 0
879          |----- DRIVE TYPE 42      *** RESERVED***
880
881          DW 0000D          ; CYLINDERS
882          DB 00D           ; HEADS
883          DW 0             ; WRITE PRE-COMPENSATION CYL
884          DB 0000D          ; CONTROL BYTE
885          DB 0             ; LANDING ZONE
886          DB 0,0,0         ; SECTORS/TRACK
887          DW 0000D
888          DB 00D
889          DB 0
890          DW 0000D
891          DB 00D
892          DB 0
893          |----- DRIVE TYPE 43      *** RESERVED***
894
895          DW 0000D          ; CYLINDERS
896          DB 00D           ; HEADS
897          DW 0             ; WRITE PRE-COMPENSATION CYL
898          DB 0000D          ; CONTROL BYTE
899          DB 0             ; LANDING ZONE
900          DB 0,0,0         ; SECTORS/TRACK
901          DW 0000D
902          DB 00D
903          DB 0
904          |----- DRIVE TYPE 44      *** RESERVED***
905
906          DW 0000D          ; CYLINDERS
907          DB 00D           ; HEADS
908          DW 0             ; WRITE PRE-COMPENSATION CYL
909          DB 0000D          ; CONTROL BYTE
910          DB 0             ; LANDING ZONE
911          DB 0,0,0         ; SECTORS/TRACK
912          DW 0000D
913          DB 00D
914          DB 0
915          DW 0000D
916          DB 00D
917          DB 0
918          |----- DRIVE TYPE 45      *** RESERVED***
919
920          DW 0000D          ; CYLINDERS
921          DB 00D           ; HEADS
922          DW 0             ; WRITE PRE-COMPENSATION CYL
923          DB 0000D          ; CONTROL BYTE
924          DB 0             ; LANDING ZONE
925          DB 0,0,0         ; SECTORS/TRACK
926          DW 0000D
927          DB 00D
928          DB 0
929          DW 0000D
930          DB 00D
931          DB 0
932          |----- DRIVE TYPE 46      *** RESERVED***
933
934          DW 0000D          ; CYLINDERS
935          DB 00D           ; HEADS
936          DW 0             ; WRITE PRE-COMPENSATION CYL
937          DB 0000D          ; CONTROL BYTE
938          DB 0             ; LANDING ZONE
939          DB 0,0,0         ; SECTORS/TRACK
940          DW 0000D
941          DB 00D
942          DB 0
  
```

SECTION 5

```

943 PAGE
944 |----- DRIVE TYPE 47 *** RESERVED***
945
946 06E1 0000 DW 0000D ; CYLINDERS
947 06E3 00 DW 00D ; HEADS
948 06E4 0000 DW 0 ;
949 06E6 0000 DW 0000D ; WRITE PRE-COMPENSATION CYL
950 06E8 00 DW 0 ;
951 06E9 00 DW 0 ; CONTROL BYTE
952 06EA 00 00 00 DW 0,0,0
953 06ED 0000 DW 0000D ; LANDING ZONE
954 06EF 00 DW 00D ; SECTORS/TRACK
955 06F0 00 DW 0
956
957
958 |----- BOOT LOADER INTERRUPT
959
960 = 06F1 IP = $
961 06F2 02 II- ORG 06F2H
962 06F2 EQU $
963 = 06F2 BOOT_STRAP EQU $
964 06F2 E9 0000 E JMP BOOT_STRAP_1 ; VECTOR ON TO MOVED BOOT CODE
965
966 | USE INT 15 H AHa 0C0H
967 06F5 CONF_TBL1 CONF_E-CNF_TBL-2 ; CONFIGURATION TABLE FOR THIS SYSTEM
968 06F5 0008 DW CONF_E-CNF_TBL-2 ; LENGTH OF FOLLOWING TABLE
969 06F7 FC DB MODEL_BYTE ; SYSTEM MODEL BYTE
970 06F8 02 DB SUB_MODEL_BYTE ; SYSTEM SUB MODEL TYPE BYTE
971 06F9 00 DB BIOS_LEVEL ; BIOS REVISION LEVEL
972 06FA 70 DB 0110000B ; 10000000 = DMA CHANNEL 3 USE BY BIOS
973 ; 01000000 = CASCADED INTERRUPT LEVEL 2
974 ; 00100000 = REAL TIME CLOCK AVAILABLE
975 ; 00010000 = KEYBOARD SCAN CODE HOOK 1AH
976 06FB 00 DW 0 ; RESERVED
977 06FC 00 DW 0 ; RESERVED
978 06FD 00 DW 0 ; RESERVED
979 06FE 00 DW 0 ; RESERVED
980 = 06FF CONF_E EQU $ ; RESERVED FOR EXPANSION
981
982 |----- BAUD RATE INITIALIZATION TABLE
983
984 = 06FF IP = $
985 0729 II- ORG 0729H
986 0729 ORG 00729H
987 0729 0417 A1 DW 1047 ; 110 BAUD ; TABLE OF VALUES
988 072B 0300 DW 768 ; FOR INITIALIZATION
989 072D 0180 DW 384 ; 300
990 072F 00C0 DW 192 ; 600
991 0731 0060 DW 96 ; 1200
992 0733 0030 DW 48 ; 2400
993 0735 0018 DW 24 ; 4800
994 0737 000C DW 12 ; 9600
995
996 |----- RS232
997
998 0739 II- ORG 0739H
999 0739 ORG 00739H
1000 = 0739 RS232_IO EQU $
1001 0739 E9 0000 E JMP RS232_IO_1 ; VECTOR ON TO MOVED RS232 CODE
1002
1003 |----- KEYBOARD
1004
1005 082E II- ORG 082EH
1006 082E ORG 0082EH
1007 = 082E KEYBOARD_IO EQU $
1008 082E E9 0000 E JMP KEYBOARD_IO_1 ; VECTOR ON TO MOVED KEYBOARD CODE
1009
1010 |----- KEY IDENTIFICATION SCAN TABLES
1011
1012 |----- TABLE OF SHIFT KEYS AND MASK VALUES
1013
1014 087E II- ORG 087EH
1015 |----- KEY TABLE
1016 K6 LABEL BYTE
1017 087E LABEL BYTE
1018 087E 52 DB INS_KEY ; INSERT KEY
1019 087F 3A 45 44 38 1D DB CAPS_KEY,NUM_KEY,SCROLL_KEY,ALT_KEY,CTL_KEY
1020 0884 2A 36 DB LEFT_KEY,RIGHT_KEY
1021 = 0008 EQU $-K6
1022
1023 |----- MASK TABLE
1024 K7 LABEL BYTE
1025 0886 LABEL BYTE
1026 0886 80 DB INS_SHIFT ; INSERT MODE SHIFT
1027 0887 40 20 10 08 04 DB CAPS_SHIFT,NUM_SHIFT,SCROLL_SHIFT,ALT_SHIFT,CTL_SHIFT
1028 088C 02 01 DB LEFT_SHIFT,RIGHT_SHIFT
1029
1030 |----- TABLES FOR CTRL CASE
1031
1032 088E K8 LABEL BYTE
1033 088E 1B FF 00 FF FF FF DB 27,-1,00,-1,-1,-1 ; Esc, 1, 2, 3, 4, 5
1034 0891 FF FF FF FF FF FF DB 30,-1,-1,-1,-1,31 ; 6, 7, 8, 9, 0, -
1035 0894 FF 7F 94 11 17 05 DB -1,127,148,17,23,5 ; =, Bksp, Tab, Q, W, E
1036 08A0 12 14 19 15 0F DB 18,20,25,21,09,16 ; R, T, Y, U, I, O
1037 08A6 10 18 1D 0A FF 01 DB 16,27,29,10,-1,01 ; P, [, ], Enter, Ctrl, A
1038 08AC 13 04 06 07 08 0A DB 19,04,06,07,08,10 ; S, D, F, G, H, J
1039 08B2 0B 0C FF FF FF FF DB 11,12,-1,-1,-1,-1 ; K, L, ;, ', ~, LShift
1040 08B8 1C 1A 18 03 16 02 DB 28,26,24,02,25,02 ; Backslash, Z, X, C, V, B
1041 08BE 0E 0D FF FF FF FF DB 14,13,-1,-1,-1,-1 ; N, M, ,, ., /, RShift
1042 08C4 96 FF 20 FF DB 180,-1,'',-1 ; , Alt, Space, CL
1043
1044 08C8 5E 5F 60 61 62 63 DB 94,96,96,97,98,99 ; F1 - F6
1045 08CE 64 66 66 67 FF FF DB 100,101,102,103,-1,-1 ; F7 - F10, NL, SL
1046 08D4 77 8D 84 8E 73 8F DB 119,141,132,142,115,143 ; Home, Up, PgUp, -, Left, Pad5
1047 08DA 74 90 75 91 76 92 DB 116,144,117,145,118,146 ; Right, *, End, Down, PgDn, Ins
1048 08E0 93 FF FF FF 89 8A DB 147,-1,-1,-1,137,138 ; Del, SysReq, Undef. Wt, F11, F12

```

```

1048          PAGE
1049          ;----- TABLES FOR LOWER CASE -----
1050
1051 08E6          K10 LABEL BYTE
1052 08E6 1B 31 32 33 34 35      DB 27,'12345'
1053 08EC 36 37 38 39 30 2D      DB '67890-'
1054 08F2 3D 08 09 71 77 85      DB '=,08,09,'qwe'
1055 08F8 72 74 79 75 69 6F      DB 'ryuiop'
1056 08FE 70 5B 5D 0D FF 61      DB 'p[',0DH,-1,'a'      ; LETTERS, Return, Ctrl
1057 0904 73 64 66 67 68 6A      DB 'sd fghj'
1058 090A 6B 5C 3B 27 60 FF      DB 'kl;',-1              ; LETTERS, L Shift
1059 0910 5C 7A 78 62 76 62      DB 92,'zxcv'
1060 0916 6E 6D 2C 2E 2F        DB 'nm,/'
1061 091B FF 2A FF 20 FF        DB -1,'*',-1,' ',-1      ; R Shift,*, Alt, Space, CL
1062
1063          ;----- LC TABLE SCAN
1064 0920 3B 3C 3D 3E 3F        DB 59,60,61,62,63      ; BASE STATE OF F1 - F10
1065 0925 40 41 42 43 44        DB 64,65,66,67,68      ; NL, SL
1066 092A FF FF
1067
1068          ;----- KEYPAD TABLE
1069 092C          K15 LABEL BYTE
1070 092C 47 48 49 FF 4B FF      DB 71,72,73,-1,75,-1   ; BASE STATE OF KEYPAD KEYS
1071 0932 4D FF 4F 50 51 52      DB 77,-1,79,80,81,82
1072 0938 53                      DB 83
1073 0939 FF FF 5C 85 86        DB -1,-1,92,133,134    ; SysRq, Underf, WT, F11, F12
1074
1075          ;----- KEYBOARD INTERRUPT
1076
1077          ORG 0E987H
1078 0987          ORG 00987H
1079 = 0987      EQU $
1080 0987 E9 0000 E          JMP KB_INT_1           ; VECTOR ON TO MOVED KEYBOARD HANDLER
1081
1082          ;----- TABLES FOR UPPER CASE -----
1083
1084 098A          K11 LABEL BYTE
1085 098A 1B 21 40 23 24 25      DB 27,'1@##X'
1086 0990 5E 26 2A 28 29 5F      DB 94,'4*()'
1087 0996 2B 08 00 51 57 45      DB '+,08,00,'QWE'
1088 099C 52 54 59 58 49 4F      DB 'RTYUIO'
1089 09A2 50 7B 7D 0D FF 41      DB 'P[',0DH,-1,'A'      ; LETTERS, Return, Ctrl
1090 09A8 53 44 46 47 48 4A      DB 'SDFGHJ'
1091 09AE 4B 4C 3A 22 7E FF      DB 'KL;',126,-1        ; LETTERS, L Shift
1092 09B4 7C 5A 58 43 56 42      DB 124,'zxcv'
1093 09BA 4E 4D 3C 3E 3F        DB 'NM<?'
1094 09BF FF 2A FF 20 FF        DB -1,'*',-1,' ',-1      ; R Shift,*, Alt, Space, CL
1095
1096          ;----- LC TABLE SCAN
1097 09C4          K12 LABEL BYTE
1098 09C4 54 55 56 57 58        DB 84,85,86,87,88      ; SHIFTED STATE OF F1 - F10
1099 09C9 59 5A 5B 5C 5D        DB 89,90,91,92,93      ; NL, SL
1100 09CE FF FF
1101
1102          ;----- NUM STATE TABLE
1103 09D0          K14 LABEL BYTE
1104 09D0 37 38 39 2D 34 35      DB '789-456+1230.'    ; NUMLOCK STATE OF KEYPAD KEYS
1105 36 2B 31 32 33 30
1106 2E
1107 09DD FF FF 7C 87 88        DB -1,-1,124,135,136    ; SysRq, Underf, WT, F11, F12

```

SECTION 5

```

1108 PAGE
1109 ;----- DISKETTE I/O
1110
1111 ;-- ORG 0EC29H
1112 OC59 ORG 00C59H
1113 = OC59 DISKETTE_IO EQU $
1114 OC59 E9 0000 E JMP DISKETTE_IO_1 ; VECTOR ON TO MOVED DISKETTE CODE
1115
1116 ;----- DISKETTE INTERRUPT
1117
1118 ;-- ORG 0EF57H
1119 OF57 ORG 00F57H
1120 = OF57 DISK_INT EQU $
1121 OF57 E9 0000 E JMP DISK_INT_1 ; VECTOR ON TO MOVED DISKETTE HANDLER
1122
1123 ;----- DISKETTE PARAMETERS
1124
1125 ;-- ORG 0EFCTH
1126 OFC7 ORG 00FCTH
1127
1128 ;-----
1129 ; DISK_BASE
1130 ; THIS IS THE SET OF PARAMETERS REQUIRED FOR
1131 ; DISKETTE OPERATION. THEY ARE POINTED AT BY THE
1132 ; DATA VARIABLE @DISK_POINTER. TO MODIFY THE PARAMETERS,
1133 ; BUILD ANOTHER PARAMETER BLOCK AND POINT AT IT
1134 ;-----
1135
1136 OFC7 DISK_BASE LABEL BYTE
1137
1138 OFC7 DF DB 11011111B ; SRT=0, HD UNLOAD=0F - 1ST SPECIFY BYTE
1139 OFC8 02 DB 2 ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
1140 OFC9 26 DB MOTOR_WAIT ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
1141 OFCA 02 DB 2 ; 512 BYTES/SECTOR
1142 OFCB 0F DB 15 ; 1ST SECTOR ON TRACK)
1143 OFCC 1B DB 01BH ; GAP LENGTH
1144 OFCD FF DB 0FFH ; DTL
1145 OFCE 54 DB 054H ; GAP LENGTH FOR FORMAT
1146 OFCF F6 DB 0F6H ; FILL BYTE FOR FORMAT
1147 OFD0 0F DB 15 ; HEAD SETTLE TIME (MILLISECONDS)
1148 OFD1 08 DB 8 ; MOTOR START TIME (1/8 SECONDS)
1149
1150 ;----- PRINTER I/O
1151
1152 ;-- ORG 0EFD2H
1153 OFD2 ORG 00FD2H
1154 = OFD2 PRINTER_IO EQU $
1155 OFD2 E9 0000 E JMP PRINTER_IO_1 ; VECTOR ON TO MOVED PRINTER CODE
1156
1157 ;----- FOR POSSIBLE COMPATIBILITY ENTRY POINTS
1158
1159 ;-- ORG 0F045H
1160 1045 ORG 01045H
1161 ASSUME CS:CODE,DS:DATA
1162
1163 EXTRN SET_MODE:NEAR
1164 EXTRN SET_CTYPE:NEAR
1165 EXTRN SET_CPOS:NEAR
1166 EXTRN READ_CURSOR:NEAR
1167 EXTRN READ_LPEN:NEAR
1168 EXTRN ACT_DISP_PAGE:NEAR
1169 EXTRN SCROLL_UP:NEAR
1170 EXTRN SCROLL_DOWN:NEAR
1171 EXTRN READ_AC_CURRENT:NEAR
1172 EXTRN WRITE_AC_CURRENT:NEAR
1173 EXTRN WRITE_C_CURRENT:NEAR
1174 EXTRN SET_COLOR:NEAR
1175 EXTRN WRITE_DOT:NEAR
1176 EXTRN READ_DOT:NEAR
1177 EXTRN WRITE_TTY:NEAR
1178 EXTRN VIDEO_STATE:NEAR
1179
1180 1045 0000 E MI DW OFFSET SET_MODE ; TABLE OF ROUTINES WITHIN VIDEO I/O
1181 1047 0000 E DW OFFSET SET_CTYPE ; EXIT STACK VALUES MAY BE
1182 1049 0000 E DW OFFSET SET_CPOS ; DIFFERENT DEPENDING ON THE
1183 104B 0000 E DW OFFSET READ_CURSOR ; SYSTEM AND MODEL
1184 104D 0000 E DW OFFSET READ_LPEN
1185 104F 0000 E DW OFFSET ACT_DISP_PAGE
1186 1051 0000 E DW OFFSET SCROLL_UP
1187 1053 0000 E DW OFFSET SCROLL_DOWN
1188 1055 0000 E DW OFFSET READ_AC_CURRENT
1189 1057 0000 E DW OFFSET WRITE_AC_CURRENT
1190 1059 0000 E DW OFFSET WRITE_C_CURRENT
1191 105B 0000 E DW OFFSET SET_COLOR
1192 105D 0000 E DW OFFSET WRITE_DOT
1193 105F 0000 E DW OFFSET READ_DOT
1194 1061 0000 E DW OFFSET WRITE_TTY
1195 1063 0000 E DW OFFSET VIDEO_STATE
1196 = 0020 MIL EQU $-MI
1197
1198 ;-- ORG 0F065H
1199 1065 ORG 01065H
1200 = 1065 VIDEO_IO EQU $
1201 1065 E9 0000 E JMP VIDEO_IO_1 ; VECTOR ON TO MOVED VIDEO CODE
1202
1203 ;----- VIDED PARAMETERS --- INIT_TABLE
1204
1205 ;-- ORG 0F0A4H
1206 10A4 ORG 010A4H
1207
1208 10A4 LABEL BYTE
1209 10A4 38 28 2D 0A 1F 06 VIDEO_PARAMS DB 38H,28H,2DH,0AH,1FH,6,19H ; SET UP FOR 40X25
1210 19 DB 0,0,0,0
1211 10AB 1C 02 07 06 07 DB 1CH,2,7,6,7
1212 10B0 00 00 00 00 DB 0,0,0,0
1213 = 0010 M4 EQU $-VIDEO_PARAMS
1214
1215 10B4 71 50 5A 0A 1F 06 DB 71H,50H,5AH,0AH,1FH,6,19H ; SET UP FOR 80X25
1216 19 DB 0,0,0,0
1217 10BB 1C 02 07 06 07 DB 1CH,2,7,6,7
1218 10C0 00 00 00 00 DB 0,0,0,0
1219
1220 10C4 38 28 2D 0A 7F 06 DB 38H,28H,2DH,0AH,7FH,6,64H ; SET UP FOR GRAPHICS
1221 64

```

```

1222 10CB 70 02 01 06 07      DB      70H,2,1,6,7
1223 10DD 00 00 00 00      DB      0,0,0,0
1224
1225 10D4 61 50 52 0F 19 06  DB      61H,60H,52H,0FH,19H,6,19H      ; SET UP FOR 80X25 B&W CARD
1226      19
1227 10DB 19 02 0D 0B 0C      DB      19H,2,0DH,0BH,0CH
1228 10E0 00 00 00 00      DB      0,0,0,0
1229
1230 10E4 0800      M5      DW      2048      ; TABLE OF REGEN LENGTHS
1231 10E6 1000      DW      4096      ; 40X25
1232 10E8 4000      DW      16384     ; 80X25
1233 10EA 4000      DW      16384     ; GRAPHICS
1234
1235 ;----- COLUMNS
1236
1237 10EC 28 28 50 50 28 28  M6      DB      40,40,80,80,40,40,80,80
1238      50 50
1239 ;----- C_REG_TAB
1240
1241 10F4 2C 28 2D 29 2A 2E  M7      DB      2CH,28H,2DH,29H,2AH,2EH,1EH,29H ; TABLE OF MODE SETS
1242      1E 29
1243 ;----- MEMORY SIZE
1244
1245 ;-- ORG 0F841H
1246 1841      ORG 01841H
1247 = 1841      MEMORY_SIZE_DET EQU $
1248 1841 E9 0000 E      JMP     MEMORY_SIZE_DET_1      ; VECTOR ON TO MOVED BIOS CODE
1249
1250 ;----- EQUIPMENT DETERMINE
1251
1252 ;-- ORG 0F840H
1253 184D      ORG 01840H
1254 = 184D      EQUIPMENT EQU $
1255 184D E9 0000 E      JMP     EQUIPMENT_1            ; VECTOR ON TO MOVED BIOS CODE
1256
1257 ;----- CASSETTE (NO BIOS SUPPORT)
1258
1259 ;-- ORG 0F859H
1260 1859      ORG 01859H
1261 = 1859      CASSETTE_10 EQU $
1262 1859 E9 0000 E      JMP     CASSETTE_10_1        ; VECTOR ON TO MOVED BIOS CODE
1263
1264 ;----- CHARACTER GENERATOR GRAPHICS FOR 320X200 AND 640X200 GRAPHICS
1265 ;-----
1266 ;-- ORG 0FA6EH
1267 1A6E      ORG 01A6EH
1268 1A6E      LABEL BYTE
1269 1A6E      CRT_CHAR_GEN
1270 1A6E 00 00 00 00 00 00  DB      00H,00H,00H,00H,00H,00H,00H ; D_00 BLANK
1271      00 00
1272 1A76 7E 81 A5 81 B0 99  DB      07EH,081H,0A5H,081H,0BDH,099H,081H,07EH ; D_01 SMILING FACE
1273      81 7E
1274 1A7E 7E FF DB FF C3 E7  DB      07EH,0FFH,0DBH,0FFH,0C3H,0E7H,0FFH,07EH ; D_02 SMILING FACE N
1275      FF 7E
1276 1A86 6C FE FE FE 7C 38  DB      06CH,0FEH,0FEH,0FEH,07CH,038H,010H,000H ; D_03 HEART
1277      10 00
1278 1A8E 10 38 7C FE 7C 38  DB      010H,038H,07CH,0FEH,07CH,038H,010H,000H ; D_04 DIAMOND
1279      10 00
1280 1A96 38 7C 38 FE FE 7C  DB      038H,07CH,038H,0FEH,0FEH,07CH,038H,07CH ; D_05 CLUB
1281      38 7C
1282 1A9E 10 10 38 7C FE 7C  DB      010H,010H,038H,07CH,0FEH,07CH,038H,07CH ; D_06 SPADE
1283      38 7C
1284 1AA6 00 00 18 3C 3C 18  DB      000H,000H,018H,03CH,03CH,018H,000H,000H ; D_07 BULLET
1285      00 00
1286 1AAE FF FF ET C3 C3 E7  DB      0FFH,0FFH,0E7H,0C3H,0C3H,0E7H,0FFH,0FFH ; D_08 BULLET NEG
1287      FF FF
1288 1AB6 00 0C 66 42 42 66  DB      000H,03CH,066H,042H,042H,066H,03CH,000H ; D_09 CIRCLE
1289      3C 00
1290 1ABE FF C3 99 B0 B0 99  DB      0FFH,0C3H,099H,0BDH,0BDH,099H,0C3H,0FFH ; D_0A CIRCLE NEG
1291      C3 FF
1292 1AC6 0F 07 0F 7D CC CC  DB      0FFH,007H,00FH,07DH,0CCH,0CCH,0CCH,078H ; D_0B MALE
1293      CC 78
1294 1ACE 3C 66 66 66 3C 18  DB      03CH,066H,066H,066H,03CH,018H,07EH,018H ; D_0C FEMALE
1295      7E 18
1296 1AD6 3F 33 3F 30 30 70  DB      03FH,033H,03FH,030H,030H,070H,0F0H,0E0H ; D_0D EIGHTH NOTE
1297      F0 E0
1298 1ADE 7F 63 7F 63 63 67  DB      07FH,063H,07FH,063H,063H,067H,0E6H,0C0H ; D_0E TWO 1/16 NOTE
1299      E6 C0
1300 1AE6 98 5A 3C ET ET 3C  DB      099H,05AH,03CH,0E7H,0E7H,03CH,05AH,099H ; D_0F SUN
1301      5A 99
1302
1303 1AE6 80 E0 F8 FE F8 E0  DB      080H,0E0H,0F8H,0FEH,0F8H,0E0H,080H,000H ; D_10 R ARROWHEAD
1304      80 00
1305 1AF6 02 0E 3E FE 3E 0E  DB      002H,00EH,03EH,0FEH,03EH,00EH,002H,000H ; D_11 L ARROWHEAD
1306      02 00
1307 1AFE 18 3C 7E 18 18 7E  DB      018H,03CH,07EH,018H,018H,07EH,03CH,018H ; D_12 ARROW 2 VERT
1308      3C 18
1309 1B06 66 66 66 66 66 00  DB      066H,066H,066H,066H,066H,000H,066H,000H ; D_13 2 EXCLAMATIONS
1310      66 00
1311 1B0E 7F DB DB 7B 1B 1B  DB      07FH,0DBH,0DBH,07BH,01BH,01BH,01BH,000H ; D_14 PARAGRAPH
1312      1B 00
1313 1B16 3E 63 38 6C 6C 38  DB      03EH,063H,038H,06CH,06CH,038H,0CCH,078H ; D_15 SECTION
1314      CC 78
1315 1B1E 00 00 00 00 7E 7E  DB      000H,000H,000H,000H,07EH,07EH,07EH,000H ; D_16 RECTANGLE
1316      7E 00
1317 1B26 18 3C 7E 18 7E 3C  DB      018H,03CH,07EH,018H,07EH,03CH,018H,0FFH ; D_17 ARROW 2 VRT UP
1318      18 FF
1319 1B2E 18 3C 7E 18 18 18  DB      018H,03CH,07EH,018H,018H,018H,018H,000H ; D_18 ARROW VRT DOWN
1320      18 00
1321 1B36 18 18 18 18 7E 3C  DB      018H,018H,018H,018H,07EH,03CH,018H,000H ; D_19 ARROW VRT DOWN
1322      18 00
1323 1B3E 00 18 0C FE 0C 18  DB      000H,018H,00CH,0FEH,00CH,018H,000H,000H ; D_1A ARROW RIGHT
1324      00 00
1325 1B46 00 30 60 FE 60 30  DB      000H,030H,060H,0FEH,060H,030H,000H,000H ; D_1B ARROW LEFT
1326      00 00
1327 1B4E 00 00 C0 C0 C0 FE  DB      000H,000H,0C0H,0C0H,0C0H,0FEH,000H,000H ; D_1C NOT INVERTED
1328      00 00
1329 1B56 00 24 66 FF 66 24  DB      000H,024H,066H,0FFH,066H,024H,000H,000H ; D_1D ARROW 2 HORZ
1330      00 00
1331 1B5E 00 18 3C 7E FF FF  DB      000H,018H,03CH,07EH,0FFH,0FFH,000H,000H ; D_1E ARROWHEAD UP
1332      00 00
1333 1B66 00 FF FF 7E 3C 18  DB      000H,0FFH,0FFH,07EH,03CH,018H,000H,000H ; D_1F ARROWHEAD DOWN
1334      00 00
1335

```

SECTION 5

1336	1B6E	00 00 00 00 00 00	DB	000H,000H,000H,000H,000H,000H,000H,000H ; D_20	SPACE
1337		00 00			
1338	1B7E	30 78 78 30 30 00	DB	030H,078H,078H,030H,030H,000H,030H,000H ; D_21	!
1339		30 00			
1340	1B7E	6C 6C 6C 00 00 00	DB	06CH,06CH,06CH,000H,000H,000H,000H,000H ; D_22	"
1341		00 00			
1342	1B8E	6C 6C FE 6C FE 6C	DB	06CH,06CH,0FEH,06CH,0FEH,06CH,06CH,000H ; D_23	#
1343		6C 00			
1344	1B8E	30 7C C0 78 0C F8	DB	030H,07CH,0C0H,078H,00CH,0F8H,030H,000H ; D_24	\$
1345		30 00			
1346	1B9E	00 C6 CC 18 30 66	DB	000H,0C6H,0CCH,018H,030H,066H,0C6H,000H ; D_25	X
1347		C6 00			
1348	1B9E	38 6C 38 76 DC CC	DB	038H,06CH,038H,076H,0DCH,0CCH,076H,000H ; D_26	&
1349		76 00			
1350	1BAE	60 60 C0 00 00 00	DB	060H,060H,0C0H,000H,000H,000H,000H,000H ; D_27	'
1351		00 00			
1352	1BAE	18 30 60 60 60 30	DB	018H,030H,060H,060H,060H,030H,018H,000H ; D_28	(
1353		18 00			
1354	1BBE	60 30 18 18 18 30	DB	060H,030H,018H,018H,018H,030H,060H,000H ; D_29	)
1355		60 00			
1356	1BBE	00 66 3C FF 3C 66	DB	000H,066H,03CH,0FFH,03CH,066H,000H,000H ; D_2A	*
1357		00 00			
1358	1BCA	00 30 30 FC 30 30	DB	000H,030H,030H,0FCH,030H,030H,000H,000H ; D_2B	+
1359		00 00			
1360	1BCA	00 00 00 00 00 30	DB	000H,000H,000H,000H,000H,030H,030H,060H ; D_2C	,
1361		30 00			
1362	1BDE	00 00 00 FC 00 00	DB	000H,000H,000H,0FCH,000H,000H,000H,000H ; D_2D	-
1363		00 00			
1364	1BDE	00 00 00 00 00 30	DB	000H,000H,000H,000H,000H,030H,030H,000H ; D_2E	.
1365		30 00			
1366	1BE0	06 0C 18 30 60 C0	DB	006H,00CH,018H,030H,060H,0C0H,080H,000H ; D_2F	/
1367		80 00			
1368					
1369	1BEE	7C C6 CE DE F6 E6	DB	07CH,0C6H,0CEH,0DEH,0F6H,0E6H,07CH,000H ; D_30	0
1370		7C 00			
1371	1BF6	30 70 30 30 30 30	DB	030H,070H,030H,030H,030H,030H,0FCH,000H ; D_31	1
1372		FC 00			
1373	1BF6	78 CC 0C 38 60 CC	DB	078H,0CCH,00CH,038H,060H,0CCH,0FCH,000H ; D_32	2
1374		FC 00			
1375	1C06	78 CC 0C 38 0C CC	DB	078H,0CCH,00CH,038H,00CH,0CCH,078H,000H ; D_33	3
1376		78 00			
1377	1C0E	1C 3C 6C CC FE 0C	DB	01CH,03CH,06CH,0CCH,0FEH,00CH,01EH,000H ; D_34	4
1378		1E 00			
1379	1C16	FC C0 F8 0C 0C CC	DB	0FCH,0C0H,0F8H,00CH,00CH,0CCH,078H,000H ; D_35	5
1380		78 00			
1381	1C1E	38 60 C0 F8 CC CC	DB	038H,060H,0C0H,0F8H,0CCH,0CCH,078H,000H ; D_36	6
1382		78 00			
1383	1C26	FC CC 0C 18 30 30	DB	0FCH,0CCH,00CH,018H,030H,030H,030H,000H ; D_37	7
1384		30 00			
1385	1C2E	78 CC CC 78 CC CC	DB	078H,0CCH,0CCH,078H,0CCH,0CCH,078H,000H ; D_38	8
1386		78 00			
1387	1C36	78 CC CC 7C 0C 18	DB	078H,0CCH,0CCH,07CH,00CH,018H,070H,000H ; D_39	9
1388		70 00			
1389	1C3E	00 30 30 00 00 30	DB	000H,030H,030H,000H,000H,030H,030H,000H ; D_3A	:
1390		30 00			
1391	1C46	00 30 30 00 00 30	DB	000H,030H,030H,000H,000H,030H,030H,060H ; D_3B	;
1392		30 60			
1393	1C4E	18 30 60 C0 60 30	DB	018H,030H,060H,0C0H,060H,030H,018H,000H ; D_3C	<
1394		18 00			
1395	1C56	00 00 FC 00 00 FC	DB	000H,000H,0FCH,000H,000H,0FCH,000H,000H ; D_3D	=
1396		00 00			
1397	1C5E	40 30 18 0C 18 30	DB	060H,030H,018H,00CH,018H,030H,060H,000H ; D_3E	>
1398		60 00			
1399	1C66	78 CC 0C 18 30 00	DB	078H,0CCH,00CH,018H,030H,000H,030H,000H ; D_3F	?
1400		30 00			
1401					
1402	1C6E	7C C6 DE DE DE C0	DB	07CH,0C6H,0DEH,0DEH,0DEH,0C0H,078H,000H ; D_40	@
1403		78 00			
1404	1C7E	30 78 CC CC FC CC	DB	030H,078H,0CCH,0CCH,0FCH,0CCH,0CCH,000H ; D_41	A
1405		CC 00			
1406	1C7E	FC 66 66 7C 66 66	DB	0FCH,066H,066H,07CH,066H,066H,0FCH,000H ; D_42	B
1407		FC 00			
1408	1C86	3C 66 C0 C0 C0 66	DB	03CH,066H,0C0H,0C0H,0C0H,066H,03CH,000H ; D_43	C
1409		3C 00			
1410	1C8E	F8 6C 66 66 66 6C	DB	0F8H,06CH,066H,066H,066H,06CH,0F8H,000H ; D_44	D
1411		F8 00			
1412	1C96	FE 62 68 78 68 62	DB	0FEH,062H,068H,078H,068H,062H,0FEH,000H ; D_45	E
1413		FE 00			
1414	1C9E	FE 62 68 78 68 60	DB	0FEH,062H,068H,078H,068H,060H,0F0H,000H ; D_46	F
1415		F0 00			
1416	1CA6	3C 66 C0 C0 CE 66	DB	03CH,066H,0C0H,0C0H,0CEH,066H,03EH,000H ; D_47	G
1417		3E 00			
1418	1CAE	CC CC CC FC CC CC	DB	0CCH,0CCH,0CCH,0FCH,0CCH,0CCH,0CCH,000H ; D_48	H
1419		CC 00			
1420	1CB6	78 30 30 30 30 30	DB	078H,030H,030H,030H,030H,030H,078H,000H ; D_49	I
1421		78 00			
1422	1CBE	1E 0C 0C CC CC CC	DB	01EH,00CH,00CH,00CH,0CCH,0CCH,078H,000H ; D_4A	J
1423		78 00			
1424	1CC6	E6 66 6C 78 6C 66	DB	0E6H,066H,06CH,078H,06CH,066H,0E6H,000H ; D_4B	K
1425		E6 00			
1426	1CCE	F0 60 60 60 62 66	DB	0F0H,060H,060H,060H,062H,066H,0FEH,000H ; D_4C	L
1427		FE 00			
1428	1CD6	C6 EE FE D6 C6	DB	0C6H,0EEH,0FEH,0FEH,0D6H,0C6H,0C6H,000H ; D_4D	M
1429		C6 00			
1430	1CDE	C6 E6 F6 DE CE C6	DB	0C6H,0E6H,0F6H,0DEH,0CEH,0C6H,0C6H,000H ; D_4E	N
1431		C6 00			
1432	1CE6	38 6C C6 C6 C6 6C	DB	038H,06CH,0C6H,0C6H,0C6H,038H,000H ; D_4F	O
1433		38 00			
1434					
1435	1CEE	FC 66 66 7C 60 60	DB	0FCH,066H,066H,07CH,060H,060H,0F0H,000H ; D_50	P
1436		F0 00			
1437	1CF6	78 CC CC CC DC 78	DB	078H,0CCH,0CCH,0CCH,0DCH,078H,01CH,000H ; D_51	Q
1438		1C 00			
1439	1CFE	FC 66 66 7C 6C 66	DB	0FCH,066H,066H,07CH,06CH,066H,0E6H,000H ; D_52	R
1440		E6 00			
1441	1DD6	78 CC E0 70 1C CC	DB	078H,0CCH,0E0H,070H,01CH,0CCH,078H,000H ; D_53	S
1442		78 00			
1443	1DDE	FC 84 30 30 30 30	DB	0FCH,0B4H,030H,030H,030H,030H,078H,000H ; D_54	T
1444		78 00			
1445	1D16	CC CC CC CC CC CC	DB	0CCH,0CCH,0CCH,0CCH,0CCH,0CCH,0FCH,000H ; D_55	U
1446		FC 00			
1447	1D1E	CC CC CC CC 78	DB	0CCH,0CCH,0CCH,0CCH,0CCH,078H,030H,000H ; D_56	V
1448		30 00			
1449	1D26	C6 C6 C6 D6 FE EE	DB	0C6H,0C6H,0C6H,0D6H,0FEH,0EEH,0C6H,000H ; D_57	W



```

1450 C6 00
1451 ID2E C6 C6 6C 3B 3B 6C DB 0C6H,0C6H,06CH,038H,038H,06CH,0C6H,000H ; D_58 X
1452 C6 00
1453 ID36 CC CC CC 7B 30 30 DB 0CCH,0CCH,0CCH,078H,030H,030H,078H,000H ; D_59 Y
1454 78 00
1455 ID3E FE C6 8C 1B 32 66 DB 0FEH,0C6H,08CH,018H,032H,066H,0FEH,000H ; D_5A Z
1456 FE 00
1457 ID46 78 60 60 60 60 60 DB 078H,060H,060H,060H,060H,060H,078H,000H ; D_5B [ LEFT BRACKET
1458 78 00
1459 ID4E C0 60 30 18 0C 06 DB 0C0H,060H,030H,018H,00CH,066H,002H,000H ; D_5C $ BACKSLASH
1460 02 00
1461 ID56 78 18 18 18 18 18 DB 078H,018H,018H,018H,018H,018H,078H,000H ; D_5D ] RIGHT BRACKET
1462 78 00
1463 ID6E 10 38 6C C6 00 00 DB 010H,038H,06CH,0C6H,000H,000H,000H,000H ; D_5E ¢ CIRCUMFLEX
1464 00 00
1465 ID66 00 00 00 00 00 00 DB 000H,000H,000H,000H,000H,000H,000H,000H ; D_5F _ UNDERSCORE
1466 00 FF
1467
1468 ID6E 30 30 18 00 00 00 DB 030H,030H,018H,000H,000H,000H,000H,000H ; D_60 ' APOSTROPHE REV
1469 00 00
1470 ID76 00 00 78 0C 7C CC DB 000H,000H,078H,00CH,07CH,0CCH,076H,000H ; D_61 ¢
1471 76 00
1472 ID7E E0 60 60 7C 66 66 DB 0E0H,060H,060H,07CH,066H,066H,0DCH,000H ; D_62 b
1473 DC 00
1474 ID86 00 00 78 CC C0 CC DB 000H,000H,078H,0CCH,0C0H,0CCH,078H,000H ; D_63 c
1475 78 00
1476 ID8E 1C 0C 0C 7C CC CC DB 01CH,00CH,00CH,07CH,0CCH,0CCH,076H,000H ; D_64 d
1477 76 00
1478 ID96 00 00 78 CC FC C0 DB 000H,000H,078H,0CCH,0FCH,0C0H,078H,000H ; D_65 e
1479 78 00
1480 ID9E 38 6C 60 F0 60 60 DB 038H,06CH,060H,0F0H,060H,060H,0F0H,000H ; D_66 f
1481 F0 00
1482 IDA6 00 00 76 CC CC 7C DB 000H,000H,076H,0CCH,0CCH,07CH,00CH,0F8H ; D_67 g
1483 0C F8
1484 IDAE E0 60 6C 76 66 66 DB 0E0H,060H,06CH,076H,066H,066H,0E6H,000H ; D_68 h
1485 E6 00
1486 IDB6 30 00 70 30 30 30 DB 030H,000H,070H,030H,030H,030H,078H,000H ; D_69 i
1487 78 00
1488 IDBE 0C 00 0C 0C 0C CC DB 00CH,000H,00CH,00CH,00CH,0CCH,0CCH,078H ; D_6A j
1489 CC 78
1490 IDC6 E0 60 66 6C 78 6C DB 0E0H,060H,066H,06CH,078H,06CH,0E6H,000H ; D_6B k
1491 E6 00
1492 IDCE 70 30 30 30 30 30 DB 070H,030H,030H,030H,030H,030H,078H,000H ; D_6C l
1493 78 00
1494 IDD6 00 00 CC FE FE D6 DB 000H,000H,0CCH,0FEH,0FEH,0D6H,0C6H,000H ; D_6D m
1495 C6 00
1496 IDDE 00 00 F8 CC CC CC DB 000H,000H,0F8H,0CCH,0CCH,0CCH,0CCH,000H ; D_6E n
1497 CC 00
1498 IDE6 00 00 78 CC CC CC DB 000H,000H,078H,0CCH,0CCH,0CCH,078H,000H ; D_6F o
1499 78 00
1500
1501 IDEE 00 00 DC 66 66 7C DB 000H,000H,0DCH,066H,066H,07CH,060H,0F0H ; D_70 p
1502 60 F0
1503 IDF6 00 00 76 CC CC 7C DB 000H,000H,076H,0CCH,0CCH,07CH,00CH,01EH ; D_71 q
1504 0C 1E
1505 IDFE 00 00 DC 76 66 60 DB 000H,000H,0DCH,076H,066H,060H,0F0H,000H ; D_72 r
1506 F0 00
1507 IE06 00 00 7C C0 78 0C DB 000H,000H,07CH,0C0H,078H,00CH,0F8H,000H ; D_73 s
1508 F8 00
1509 IE0E 10 30 7C 30 30 34 DB 010H,030H,07CH,030H,030H,034H,018H,000H ; D_74 t
1510 18 00
1511 IE16 00 00 CC CC CC CC DB 000H,000H,0CCH,0CCH,0CCH,0CCH,076H,000H ; D_75 u
1512 76 00
1513 IE1E 00 00 CC CC CC 78 DB 000H,000H,0CCH,0CCH,0CCH,078H,030H,000H ; D_76 v
1514 30 00
1515 IE26 00 00 C6 D6 FE FE DB 000H,000H,0C6H,0D6H,0FEH,0FEH,06CH,000H ; D_77 w
1516 60 00
1517 IE2E 00 00 C6 6C 3B 6C DB 000H,000H,0C6H,06CH,038H,06CH,0C6H,000H ; D_78 x
1518 C6 00
1519 IE36 00 00 CC CC CC 7C DB 000H,000H,0CCH,0CCH,0CCH,07CH,00CH,0F8H ; D_79 y
1520 0C F8
1521 IE3E 00 00 FC 9B 30 64 DB 000H,000H,0FCH,098H,030H,064H,0FCH,000H ; D_7A z
1522 FC 00
1523 IE46 1C 30 30 E0 30 30 DB 01CH,030H,030H,0E0H,030H,030H,01CH,000H ; D_7B { LEFT BRACE
1524 1C 00
1525 IE4E 18 18 18 00 18 18 DB 018H,018H,018H,000H,018H,018H,018H,000H ; D_7C | BROKEN STROKE
1526 18 00
1527 IE56 E0 30 30 1C 30 30 DB 0E0H,030H,030H,01CH,030H,030H,0E0H,000H ; D_7D } RIGHT BRACE
1528 E0 00
1529 IE5E 76 DC 00 00 00 00 DB 076H,0DCH,000H,000H,000H,000H,000H ; D_7E ¢ TILDE
1530 00 00
1531 IE66 00 10 38 6C C6 C6 DB 000H,010H,038H,06CH,0C6H,0C6H,0FEH,000H ; D_7F DELTA
1532 FE 00
1533
1534
1535
1536
1537 IE6E
1538 IE6E
1539 IE6E E9 0000 E
1540
1541
1542
1543
1544 IEA5
1545 IEA5
1546 IEA5 E9 0000 E

```

```

;----- TIME OF DAY
;-- ORG OFE6EH
;-- ORG 01E6EH
TIME_OF_DAY EQU $
;-----
;-- JUMP TIME_OF_DAY_! ; VECTOR ON TO MOVED BIOS CODE

;----- TIMER INTERRUPT
;-- ORG OFEASH
;-- ORG 01EASH
TIMER_INT EQU $
;-----
;-- JUMP TIMER_INT_! ; VECTOR ON TO MOVED BIOS CODE

```

SECTION 5

```

1547
1548
1549
1550
1551 1EF3
1552 1EF3
1553 1EF3 1EA5 R
1554 1EF6 0987 R
1555 1EF7 0000 E
1556 1EF9 0000 E
1557 1EF8 0000 E
1558 1EFD 0000 E
1559 1EFF 0F57 R
1560 1F01 0000 E
1561
1562
1563
1564 1F03 1065 R
1565 1F05 1840 R
1566 1F07 1841 R
1567 1F09 0C59 R
1568 1F0B 0789 R
1569 1F0D 1859 R
1570 1F0F 082E R
1571 1F11 0FD2 R
1572 1F13 0000
1573 1F16 04F2 R
1574 1F17 1E5E R
1575 1F19 1F53 R
1576 1F1B 1F53 R
1577 1F1D 10A4 R
1578 1F1F 0FC7 R
1579 1F21 0000
1580
1581 1F23
1582
1583 1F23 0000 E
1584 1F26 0000 E
1585 1F27 0000 E
1586 1F29 0000 E
1587 1F2B 0000 E
1588 1F2D 0000 E
1589 1F2F 0000 E
1590 1F31 0000 E
1591
1592
1593
1594
1595 1F53
1596
1597 = 1F53
1598
1599 1F53 CF
1600
1601
1602
1603
1604 1F54
1605 = 1F54
1606 1F54 E9 0000 E
1607
1608
1609
1610
1611
1612
1613
1614 1FF0
1615
1616
1617
1618 1FF0
1619
1620 1FF0 EA
1621 1FF1 008B R
1622 1FF3 0000
1623
1624 1FF6 30 34 2F 32 31 2F
1625 38 36
1626
1627 1FFE
1628 1FFE FC
1629
1630 1FFF
1631

```

```

PAGE
|----- VECTOR TABLE
|
| 11- ORG 0FEF3H
| VECTOR_TABLE LABEL WORD | AT LOCATION 0FEF3H
| DW OFFSET TIMER_INT | VECTOR TABLE VALUES FOR POST TESTS
| DW OFFSET KS_INT | INT 08H - HARDWARE TIMER 0 IRQ 0
| DW OFFSET DT1 | INT 09H - KEYBOARD IRQ 1
| DW OFFSET D11 | INT 0AH - SLAVE INTERRUPT INPUT
| DW OFFSET D11 | INT 0BH - IRQ 3
| DW OFFSET D11 | INT 0CH - IRQ 4
| DW OFFSET D11 | INT 0DH - IRQ 5
| DW OFFSET DISK_INT | INT 0EH - DISKETTE IRQ 6
| DW OFFSET D11 | INT 0FH - IRQ 7
|
|----- SOFTWARE INTERRUPTS ( BIOS CALLS AND POINTERS )
|
| DW OFFSET VIDEO_IO | INT 10H -- VIDEO DISPLAY
| DW OFFSET EQUIPMENT | INT 11H -- GET EQUIPMENT FLAG WORD
| DW OFFSET MEMORY_SIZE_DET | INT 12H -- GET REAL MODE MEMORY SIZE
| DW OFFSET DISKETTE_IO | INT 13H -- DISKETTE
| DW OFFSET RS232_IO | INT 14H -- COMMUNICATION ADAPTER
| DW OFFSET CASSETTE_IO | INT 15H -- EXPANDED BIOS FUNCTION CALL
| DW OFFSET KEYBOARD_IO | INT 16H -- KEYBOARD INPUT
| DW OFFSET PRINTER_IO | INT 17H -- PRINTER OUTPUT
| DW 00000H | INT 18H -- 0F600H INSERTED FOR BASIC
| DW OFFSET BOOT_STRAP | INT 19H -- BOOT FROM SYSTEM MEDIA
| DW OFFSET TIME_OF_DAY | INT 1AH -- TIME OF DAY
| DW OFFSET DUMMY_RETURN | INT 1BH -- KEYBOARD BREAK ADDRESS
| DW OFFSET DUMMY_RETURN | INT 1CH -- TIMER BREAK ADDRESS
| DW OFFSET VIDEO_PARAMS | INT 1DH -- VIDEO PARAMETERS
| DW OFFSET DISK_BASE | INT 1EH -- DISKETTE PARAMETERS
| DW 00000H | INT 1FH -- POINTER TO VIDEO EXTENSION
|
| SLAVE_VECTOR_TABLE LABEL WORD | ( INTERRUPT 70H THRU 7FH )
| DW OFFSET RTC_INT | INT 70H - REAL TIME CLOCK IRQ 8
| DW OFFSET RE_DIRECT | INT 71H - REDIRECT TO INT 0AH IRQ 9
| DW OFFSET DT1 | INT 72H - IRQ 10
| DW OFFSET D11 | INT 73H - IRQ 11
| DW OFFSET D11 | INT 74H - IRQ 12
| DW OFFSET INT_287 | INT 75H - -MATH COPROCESSOR IRQ 13
| DW OFFSET D11 | INT 76H - -FIXED DISK IRQ 14
| DW OFFSET D11 | INT 77H - IRQ 15
|
|----- DUMMY INTERRUPT HANDLER
|
| 11- ORG 0FF53H
| ORG 01F53H
|
| DUMMY_RETURN EQU $ | BIOS DUMMY (NULL) INTERRUPT RETURN
|
| IRET
|
|----- PRINT SCREEN
|
| 11- ORG 0FF54H
| ORG 01F54H
| PRINT_SCREEN EQU $ |
| .LIST JMP PRINT_SCREEN_1 | VECTOR ON TO MOVED BIOS CODE
| | TUTOR
| |-----
| |
| | POWER ON RESET VECTOR
| |-----
| |
| 11- ORG 0FFF0H
| ORG 01FF0H
|
|----- POWER ON RESET
|
| P_O_R LABEL FAR | POWER ON RESTART EXECUTION LOCATION
|
| DB 0EAH | HARD CODE FAR JUMP TO SET
| DW OFFSET RESET | OFFSET
| DW 0F000H | SEGMENT
|
| DB '04/21/86' | RELEASE MARKER
|
| ORG 01FFE H
| DB MODEL_BYTE | THIS PC'S ID ( MODEL BYTE )
|
| CODE ENDS | CHECKSUM AT LAST LOCATION
| END

```

## SECTION 6. INSTRUCTION SET

80286 Instruction Set .....	6-3
Data Transfer .....	6-3
Arithmetic .....	6-6
Logic .....	6-9
String Manipulation .....	6-11
Control Transfer .....	6-13
Processor Control .....	6-17
Protection Control .....	6-18
80287 Coprocessor Instruction Set .....	6-22
Data Transfer .....	6-22
Comparison .....	6-23
Constants .....	6-24
Arithmetic .....	6-25
Transcendental .....	6-26

**6-2 Instruction Set**

# 80286 Instruction Set

## Data Transfer

### MOV = move

Register to Register/Memory

1000100w	mod reg r/w
----------	-------------

Register/Memory to Register

1000101w	mod reg r/w
----------	-------------

Immediate to Register/Memory

1100011w	mod 000 r/w	data	data if w = 1
----------	-------------	------	---------------

Immediate to Register

1011wreg	data	data if w = 1
----------	------	---------------

Memory to Accumulator

1010000w	addr-low	addr-high
----------	----------	-----------

Accumulator to Memory

1010001w	addr-low	addr-high
----------	----------	-----------

Register/Memory to Segment Register

10001110	mod0reg r/w	reg ≠ 01
----------	-------------	----------

Segment Register to Register/Memory

10001100	mod0reg r/w
----------	-------------

### PUSH = Push

Memory

11111111	mod110 r/w
----------	------------

Register

01010reg

Segment Register

000reg110

Immediate

011010s0

data

data if s = 0

**PUSHA = Push All**

01100000

**POP = Pop**

Memory

10001111

mod000 r/m

Register

01011reg

Segment Register

000reg111

reg ≠ 01

**POPA = Pop All**

01100001

**XCHG = Exchange**

Register/Memory with Register

1000011w

mod reg r/m

Register with Accumulator

10010reg

**IN = Input From**

Fixed Port

1110010w	port
----------	------

Variable Port

1110110w
----------

**OUT = Output To**

Fixed Port

1110011w	port
----------	------

Variable Port

1110111w
----------

**XLAT = Translate Byte to AL**

11010111
----------

**LEA = Load EA to Register**

10001101	mod reg r/m
----------	-------------

**LDS = Load Pointer to DS**

11000101	mod reg r/m	mod $\neq$ 11
----------	-------------	---------------

**LES = Load Pointer to ES**

11000100	mod reg r/m	mod $\neq$ 11
----------	-------------	---------------

**LAHF = Load AH with Flags**

10011111
----------

**SAHF = Store AH with Flags**

10011110
----------

**PUSHF = Push Flags**

10011100
----------

**POPF = Pop Flags**

10011101
----------

**Arithmetic**

**ADD = Add**

Register/Memory with Register to Either

0000000w	mod reg r/m
----------	-------------

Immediate to Register Memory

100000sw	mod000 r/m	data	data if sw = 01
----------	------------	------	-----------------

Immediate to Accumulator

0000010w	data	data if w = 1
----------	------	---------------

**ADC = Add with Carry**

Register/Memory with Register to Either

000100dw	mod reg r/m
----------	-------------

Immediate to Register/Memory

100000sw	mod000 r/m	data	data if sw = 01
----------	------------	------	-----------------

Immediate to Accumulator

0001010w	data	data if w = 1
----------	------	---------------

**INC = Increment**

Register/Memory

1111111w	mod000 r/m
----------	------------



Register

01000reg
----------

**SUB = Subtract**

Register/Memory with Register to Either

001010dw	mod reg r/m
----------	-------------

Immediate from Register/Memory

100000sw	mod101 r/m	data	data if sw = 01
----------	------------	------	-----------------

Immediate from Accumulator

0010110w	data	data if w = 1
----------	------	---------------

**SBB = Subtract with Borrow**

Register/Memory with Register to Either

000110dw	mod reg r/m
----------	-------------

Immediate to Register/Memory

100000sw	mod011 r/m	data	data if sw = 01
----------	------------	------	-----------------

Immediate to Accumulator

0001110w	data	data if w = 1
----------	------	---------------

**DEC = Decrement**

Register/Memory

1111111w	mod001 r/m
----------	------------

Register

01001reg
----------

**CMP = Compare**

Register/Memory with Register

0011101w	mod reg r/m
----------	-------------

Register with Register/Memory

0011100w	mod reg r/m
----------	-------------

Immediate with Register/Memory

100000sw	mod111 r/m	data	data if sw = 01
----------	------------	------	-----------------

Immediate with Accumulator

0001110w	data	data if w = 1
----------	------	---------------

**NEG = Change Sign**

1111011w	mod011 r/m
----------	------------

**AAA = ASCII Adjust for Add**

00110111
----------

**DAA = Decimal Adjust for Add**

00100111
----------

**AAS = ASCII Adjust for Subtract**

00111111
----------

**DAS = Decimal Adjust for Subtract**

00110111
----------

**MUL = Multiply (Unsigned)**

1111011w	mod100 r/m
----------	------------

**IMUL = Integer Multiply (Signed)**

1111011w	mod101 r/m
----------	------------

**IMUL = Integer Immediate Multiply (Signed)**

011010s1	mod reg r/m	Data	Data if s = 0
----------	-------------	------	---------------

**DIV = Divide (Unsigned)**

1111011w	mod110 r/m
----------	------------

**IDIV = Integer Divide (Signed)**

1111011w	mod111 r/m
----------	------------

**AAM = ASCII Adjust for Multiply**

11010100	00001010
----------	----------

**AAD = ASCII Adjust for Divide**

11010101	00001010
----------	----------

**CBW = Convert Byte to Word**

10011000
----------

**CWD = Convert Word to Double Word**

10011001
----------

**Logic**

**Shift/Rotate Instructions**

Register/Memory by 1

1101000w	mod TTT r/m
----------	-------------

Register/Memory by CL

1101001w	mod TTT r/m
----------	-------------

Register/Memory by Count

1100000w	mod TTT r/m	count
----------	-------------	-------

T T T	Instruction
000	ROL
001	ROR
010	RCL
011	RCR
100	SHL/SAL
101	SHR
111	SAR

### AND = And

Register/Memory and Register to Either

001000dw	mod reg r/m
----------	-------------

Immediate to Register/Memory

1000000w	mod000 r/m	data	data if w = 1
----------	------------	------	---------------

Immediate to Accumulator

0010010w	data	data if w = 1
----------	------	---------------

### TEST = AND Function to Flags; No Result

Register/Memory and Register

1000010w	mod reg r/m
----------	-------------

Immediate Data and Register/Memory

1111011w	mod000 r/m	data	data if w = 1
----------	------------	------	---------------

Immediate to Accumulator

0000110w	data	data if w = 1
----------	------	---------------

### Or = Or

Register/Memory and Register to Either

000010dw	mod reg r/m
----------	-------------

Immediate to Register/Memory

1000000w	mod001 r/m	data	data if w = 1
----------	------------	------	---------------

Immediate to Accumulator

0000110w	data	data if w = 1
----------	------	---------------

### **XOR = Exclusive OR**

Register/Memory and Register to Either

001100dw	mod reg r/m
----------	-------------

Immediate to Register/Memory

1000000w	mod110 r/m	data	data if w = 1
----------	------------	------	---------------

Immediate to Accumulator

0010010w	data	data if w = 1
----------	------	---------------

### **NOT = Invert Register/Memory**

1111011w	mod010 r/m
----------	------------

## **String Manipulation**

### **MOVS = Move Byte Word**

1010010w
----------

### **CMPS = Compare Byte Word**

1010011w
----------

### **SCAS = Scan Byte Word**

1010111w
----------

### **LODS = Load Byte Word to AL/AX**

1010110w
----------

**STOS = Store Byte Word from AL/AX**

1010101w
----------

**INS = Input Byte from DX Port**

0110110w
----------

**OUTS = Output Byte Word to DX Port**

0110111w
----------

**REP/REPNE, REPZ/REPNZ = Repeat String**

Repeat Move String

11110011	1010010w
----------	----------

Repeat Compare String (z/Not z)

1111001z	1010011w
----------	----------

Repeat Scan String (z/Not z)

111100iz	1010111w
----------	----------

Repeat Load String

11110011	1010110w
----------	----------

Repeat Store String

11110011	1010101w
----------	----------

Repeat Input String

11110011	0110110w
----------	----------

Repeat Output String

11110011	1010011w
----------	----------

## Control Transfer

### CALL = Call

Direct Within Segment

11101000	disp-low	disp-high
----------	----------	-----------

Register/Memory Indirect Within Segment

11111111	mod010 r/m
----------	------------

Direct Intersegment

10011010	Segment Offset	Segment Selector
----------	----------------	------------------

Indirect Intersegment

11111111	mod011 r/m (mod ≠ 11)
----------	-----------------------

### JMP = Unconditional Jump

Short/Long

11101011	disp-low
----------	----------

Direct within Segment

11101001	disp-low	disp-high
----------	----------	-----------

Register/Memory Indirect Within Segment

11111111	mod100 r/m
----------	------------

Direct Intersegment

11101010	Segment Offset	Segment Selector
----------	----------------	------------------

Indirect Intersegment

11111111	mod101 r/m (mod ≠ 11)
----------	-----------------------

### RET = Return from Call

Within Segment

11000011
----------

Within Segment Adding Immediate to SP

11000010	data-low	data-high
----------	----------	-----------

Intersegment

11001011
----------

Intersegment Adding Immediate to SP

11001010	data-low	data-high
----------	----------	-----------

**JE/JZ = Jump on Equal/Zero**

01110100	disp
----------	------

**JL/JNGE = Jump on Less/Not Greater, or Equal**

01111100	disp
----------	------

**JLE/JNG = Jump on Less, or Equal/Not Greater**

01111110	disp
----------	------

**JB/JNAE = Jump on Below/Not Above, or Equal**

01110010	disp
----------	------

**JBE/JNA = Jump on Below, or Equal/Not Above**

01110110	disp
----------	------

**JP/JPE = Jump on Parity/Parity Even**

01111010	disp
----------	------

**JO = Jump on Overflow**

01110000	disp
----------	------

**JS = Jump on Sign**

01111000	disp
----------	------



**JNE/JNZ = Jump on Not Equal/Not Zero**

01110101	disp
----------	------

**JNL/JGE = Jump on Not Less/Greater, or Equal**

01111101	disp
----------	------

**JNLE/JG = Jump on Not Less, or Equal/Greater**

01111111	disp
----------	------

**JNB/JAE = Jump on Not Below/Above, or Equal**

01110011	disp
----------	------

**JNBE/JA = Jump on Not Below, or Equal/Above**

01110111	disp
----------	------

**JNP/JPO = Jump on Not Parity/Parity Odd**

01111011	disp
----------	------

**JNO = Jump on Not Overflow**

01110001	disp
----------	------

**JNS = Jump on Not Sign**

01111011	disp
----------	------

**LOOP = Loop CX Times**

11100010	disp
----------	------

**LOOPZ/LOOPE = Loop while Zero/Equal**

11100001	disp
----------	------

**LOOPNZ/LOOPNE = Loop while Not Zero/Not Equal**

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  $\pi$  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  $\text{Log}_2$  [ST(0)]**

escape 001	11110001
------------	----------

**FYL2XP1 = ST(1) x Log<sub>2</sub> [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

Character Codes ..... 7-3  
Quick Reference ..... 7-14

|

⌋

⌋

⌋

**7-2 Characters, Keystrokes, and Colors**

# Character Codes

Value		As Characters			As Text Attributes		
					Color/Graphics Monitor Adapter		IBM Monochrome Display Adapter
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground	
00	0	Blank (Null)	Ctrl 2		Black	Black	Non-Display
01	1	☺	Ctrl A		Black	Blue	Underline
02	2	☹	Ctrl B		Black	Green	Normal
03	3	♥	Ctrl C		Black	Cyan	Normal
04	4	♦	Ctrl D		Black	Red	Normal
05	5	♣	Ctrl E		Black	Magenta	Normal
06	6	♠	Ctrl F		Black	Brown	Normal
07	7	●	Ctrl G		Black	Light Grey	Normal
08	8	•	Ctrl H, Backspace, Shift Backspace		Black	Dark Grey	Non-Display
09	9	○	Ctrl I		Black	Light Blue	High Intensity Underline
0A	10	◐	Ctrl J, Ctrl ←		Black	Light Green	High Intensity
0B	11	♂	Ctrl K		Black	Light Cyan	High Intensity
0C	12	♀	Ctrl L		Black	Light Red	High Intensity
0D	13	♪	Ctrl M, ←, Shift ←		Black	Light Magenta	High Intensity
0E	14	♫	Ctrl N		Black	Yellow	High Intensity
0F	15	⚙	Ctrl O		Black	White	High Intensity
10	16	▶	Ctrl P		Blue	Black	Normal
11	17	◀	Ctrl Q		Blue	Blue	Underline
12	18	↕	Ctrl R		Blue	Green	Normal
13	19	!!	Ctrl S		Blue	Cyan	Normal
14	20	¶	Ctrl T		Blue	Red	Normal
15	21	§	Ctrl U		Blue	Magenta	Normal
16	22	■	Ctrl V		Blue	Brown	Normal
17	23	↕	Ctrl W		Blue	Light Grey	Normal

Value		As Characters			As Text Attributes		
					Color/Graphics Monitor Adapter		IBM Monochrome Display Adapter
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground	
18	24	↑	Ctrl X		Blue	Dark Grey	High Intensity
19	25	↓	Ctrl Y		Blue	Light Blue	High Intensity Underline
1A	26	→	Ctrl Z		Blue	Light Green	High Intensity
1B	27	←	Ctrl [, Esc, Shift Esc, Ctrl 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

#### 7-4 Characters, Keystrokes, and Colors

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

## 7-6 Characters, Keystrokes, and Colors

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	Δ	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
















## 7-8 Characters, Keystrokes, and Colors



Value		As Characters			As Text Attributes		
					Color/Graphics Monitor Adapter		IBM Monochrome Display Adapter
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground	
9B	155	¢	Alt 155	Note 6	Blue	Light Cyan	High Intensity
9C	156	£	Alt 156	Note 6	Blue	Light Red	High Intensity
9D	157	¥	Alt 157	Note 6	Blue	Light Magenta	High Intensity
9E	158	Pt	Alt 158	Note 6	Blue	Yellow	High Intensity
9F	159	f	Alt 159	Note 6	Blue	White	High Intensity
A0	160	á	Alt 160	Note 6	Green	Black	Normal
A1	161	í	Alt 161	Note 6	Green	Blue	Underline
A2	162	ó	Alt 162	Note 6	Green	Green	Normal
A3	163	ú	Alt 163	Note 6	Green	Cyan	Normal
A4	164	ñ	Alt 164	Note 6	Green	Red	Normal
A5	165	Ñ	Alt 165	Note 6	Green	Magenta	Normal
A6	166	<u>a</u>	Alt 166	Note 6	Green	Brown	Normal
A7	167	<u>o</u>	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	i	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

7-10 Characters, Keystrokes, and Colors

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	$\alpha$	Alt 224	Note 6	Brown	Black	Normal
E1	225	$\beta$	Alt 225	Note 6	Brown	Blue	Underline
E2	226	$\Gamma$	Alt 226	Note 6	Brown	Green	Normal
E3	227	$\pi$	Alt 227	Note 6	Brown	Cyan	Normal
E4	228	$\Sigma$	Alt 228	Note 6	Brown	Red	Normal
E5	229	$\sigma$	Alt 229	Note 6	Brown	Magenta	Normal
E6	230	$\mu$	Alt 230	Note 6	Brown	Brown	Normal
E7	231	$\tau$	Alt 231	Note 6	Brown	Light Grey	Normal
E8	232	$\Phi$	Alt 232	Note 6	Brown	Dark Grey	High Intensity
E9	233	$\theta$	Alt 233	Note 6	Brown	Light Blue	High Intensity Underline
EA	234	$\Omega$	Alt 234	Note 6	Brown	Light Green	High Intensity
EB	235	$\delta$	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	<sup>n</sup>	Alt 252	Note 6	Light Grey	Light Red	High Intensity
FD	253	<sup>2</sup>	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

## 7-12 Characters, Keystrokes, and Colors

## 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	â	û	à				μ	÷
7	7	ç	ù	ó				γ	≈
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	f	»				∩	BLANK 'FF'

**Notes:**

—

—

—



# SECTION 8. IBM PERSONAL COMPUTER COMPATIBILITY

Hardware Considerations .....	8-3
System Board .....	8-3
Fixed Disk Drive .....	8-5
Diskette Drive Compatibility .....	8-5
Copy Protection .....	8-5
Bypassing BIOS .....	8-6
Diskette Drive Controls .....	8-6
Write Current Control .....	8-6
Application Guidelines .....	8-7
High-Level Language Considerations .....	8-7
Assembler Language Programming Considerations ..	8-8
Multitasking Provisions .....	8-15
Interfaces .....	8-15
Classes .....	8-17
Time-Outs .....	8-18
Machine-Sensitive Code .....	8-19

|

)

)

)

**8-2 Compatibility**

This section describes the differences among the members of the IBM Personal Computer family. It also contains information necessary to design hardware and programs that will be compatible with all members of the IBM Personal Computer family.

## Hardware Considerations

To design compatible hardware or programs, you must consider hardware differences among the IBM Personal Computers. The following are hardware features of the IBM Personal Computer XT Model 286 that are not supported by all of the IBM Personal Computer family.

### System Board

The IBM Personal Computer XT Model 286 system board uses an Intel 80286-6 Microprocessor. This microprocessor uses the 80287 Math Coprocessor and is generally compatible with the Intel 8088 Microprocessor used in other IBM Personal Computers.

The following table identifies the microprocessor and describes the I/O channel used with each type of IBM Personal Computer.

System Name	System Unit Microprocessor	I/O Channel Description
Personal Computer	8088	5 62-Pin
PCjr	8088	Not Compatible
Personal Computer XT	8088	8 62-Pin
Portable Personal Computer	8088	8 62-Pin
Personal Computer XT Model 286	80286-6	3 62-pin 5 98-Pin (62 Pin + 36 Pin)
Personal Computer AT	80286(-6 or -8)	2 62-pin 6 98-Pin (62 Pin + 36 Pin)

### System Hardware Identification Chart

The faster processing capability of the 80286, compared to the 8088, creates special programming considerations, which are discussed later in this section under "Application Guidelines."

Adapters designed to use all 98 pins on the 98-pin connectors are not compatible with all members of the IBM Personal Computer family. Some options not supported by the IBM Personal Computer XT Model 286 are:

- Expansion Unit
- AT 128KB Memory Expansion
- AT 512KB Memory Expansion
- AT 128/640KB Memory Expansion
- AT Fixed Disk And Diskette Drive Adapter
- 256KB Memory Expansion
- 64/256KB Memory Expansion
- 64KB Memory Module Kit
- Full-high diskette drives
- AT 30MB Fixed Disk Drive
- AT 20MB Fixed Disk Drive
- 10MB Fixed Disk Drive
- AT Prototype Card
- Diskette Drive Adapter
- Fixed Disk Adapter
- 8087 Math Coprocessor
- Professional Graphics Adapter and Display
- Game Control Adapter
- Color Printer
- Other keyboards

On the I/O channel:

- The system clock signal should be used only for synchronization and not for applications requiring a fixed frequency.
- The 14.31818-MHz oscillator is not synchronous with the system clock.
- The ALE signal remains high during DMA cycles.
- Pin B04 supports IRQ 9.

#### 8-4 Compatibility

## Fixed Disk Drive

Reading from and writing to this drive is initiated in the same way as with other IBM Personal Computers; however, the Fixed Disk and Diskette Drive Adapter may be addressed from different BIOS locations.

## Diskette Drive Compatibility

The following chart shows the read, write, and format capabilities for each of the diskette drives used by IBM Personal Computers.

Diskette Drive Name	160/180K Mode	320/360K Mode	1.2M Mode	720K Mode
5-1/4 In. Diskette Drive				
Type 1	R W F	---	---	---
Type 2	R W F	R W F	---	---
Type 3	R W F	R W F	---	---
Slimline Diskette Drive	R W F	R W F	---	---
Double Sided Diskette Drive	R W F	R W F	---	---
High Capacity Diskette Drive	R W*	R W*	R W F	---
3-1/2 In.- 720K Drive	---	---	---	R W F

R=Read W=Write F=Format W\*= If a diskette is formatted in either 160/180K mode or 320/360K mode and written on by a High Capacity Drive, that diskette may be read by only a High Capacity Drive.

**Diskette Drive Compatibility Chart**

**Note:** Diskettes designed for the 1.2M mode may not be used in either a 160/180K or a 320/360K diskette drive.

## Copy Protection

The following methods of copy protection may not work on systems using the High Capacity Diskette Drive:

- Bypassing BIOS
- Diskette drive controls
- Write current control

## **Bypassing BIOS**

Copy protection that tries to bypass the following BIOS routines will not work on the High Capacity Diskette Drive:

**Track Density:** The High Capacity Diskette Drive records tracks at a density of 96 tracks per inch (TPI). This drive has to double-step in the 48 TPI mode, which is performed by BIOS.

**Data Transfer Rate:** BIOS selects the proper data transfer rate for the media being used.

**Disk \_\_ Base:** Copy protection, which creates its own disk \_\_ base will not work on the High Capacity Diskette Drive.

## **Diskette Drive Controls**

Copy protection that uses the following will not work on the High Capacity Diskette Drive:

**Rotational Speed:** The time between two events on a diskette is controlled by the Fixed Disk and Diskette Drive Adapter.

**Access Time:** Diskette BIOS routines must set the track-to-track access time for the different types of media used on the IBM Personal Computer XT Model 286.

**Head Geometry:** See "Diskette Drive Compatibility" on page 8-5.

**Diskette Change Signal:** Copy protection may not be able to reset this signal.

## **Write Current Control**

Copy protection that uses write current control will not work because the Fixed Disk and Diskette Drive Adapter selects the proper write current for the media being used.

## Application Guidelines

The following information should be used to develop application programs for the IBM Personal Computer family.

### High-Level Language Considerations

The IBM-supported languages of BASIC, FORTRAN, COBOL, Pascal, and APL are the best choices for writing compatible programs.

If a program uses specific features of the hardware, that program may not be compatible with all IBM Personal Computers. Specifically, the use of assembler language subroutines or hardware-specific commands (In, Out, Peek, Poke, ...) must follow the assembler language rules (see "Assembler Language Programming Considerations" on page 8-8).

Any program that requires precise timing information should obtain it through a DOS or language interface; for example, TIME\$ in BASIC. If greater precision is required, the assembler techniques in "Assembler Language Programming Considerations" are available. The use of programming loops may prevent a program from being compatible with other IBM Personal Computers.

## Assembler Language Programming Considerations

The following OP codes work differently on systems using the 80286 microprocessor than they do on systems using the 8088 microprocessor.

- If the system microprocessor executes a POPF instruction in either the real or the virtual address mode with  $CPL \leq IOPL$ , 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
OE         PUSH CS
E8 FB FF   CALL $-2  ; CALL within segment
              ; program will continue here
```

- PUSH SP

80286 microprocessor pushes the current stack pointer.

8088 microprocessor pushes the new stack pointer.

- Single step interrupt (when  $TF=1$ ) on the interrupt instruction (OP code hex CC,CD):

80286 microprocessor does **not** interrupt on the INT instruction.



8088 microprocessor does interrupt on the INT instruction.

- The divide error exception (interrupt 0):

80286 microprocessor pushes the CS:IP of the instruction, causing the exception.

8088 microprocessor pushes the CS:IP **following** the instruction, causing the exception.

- Shift counts are masked to five bits. Shift counts greater than 31 are treated mod 32. For example, a shift count of 36, shifts the operand four places.

The following describes anomalies which may occur in systems which contain 80286 processors with 1983 and 1984 date codes (S40172, S54036, S40093, S54012).

In protected mode, the contents of the CX register may be unexpectedly altered under the following conditions:

**Note:** The value in parenthesis indicates the type of error code pushed onto the exception handler's stack.

**Exception #NP() = Exception #11 = Not-present Fault**

**Exception #SS() = Exception #12 = Stack Fault**

**Exception #GP() = Exception #13 = General Protection Fault**

- Exception #GP(0) from attempted access to data segment or extra segment when the corresponding segment register holds a null selector.
- Exception #GP(0) from attempted data read from code segment when code segment has the "execute only" attribute.
- Exception #GP(0) from attempted write to code segment (code segments are not writable in protected mode), or to data segment of extra segment if the data or extra segment has the read only attribute.
- Exception #GP(0) from attempted load of a selector referencing the local descriptor table into CS, DS, ES or SS, when the LDT is not present.

- Exception #GP(0) from attempted input or output instruction when  $CPL > IOPL$ .
- Exception #GP(selector) from attempted access to a descriptor is GDT, LDT, or IDT, beyond the defined limit of the descriptor table.
- Exception #GP(0) from attempted read or write (except for "PUSH" onto stack) beyond the defined limit of segment.
- Exception #SS(0) from attempted "PUSH" below the defined limit of the stack segment.

Restarting applications which generate the above exceptions may result in errors.

In the protected mode, when any of the null selector values (0000H, 0001H, 0002H, 0003H) are loaded into the DS or ES registers via a MOV or POP instruction or a task switch, the 80286 always loads the null selector 0000H into the corresponding register.

If a coprocessor (80287) operand is read from an "executable and readable" and conforming (ERC) code segment, and the coprocessor operand is sufficiently near the segment's limit that the second or subsequent byte lies outside the limit, no protection exception #9 will be generated.

The following correctly describes the operation of all 80286 parts:

- Instructions longer than 10 bytes (instructions using multiple redundant prefixes) generate exception #13 (General Purpose Exception) in both the real and protected modes.
- If the second operand of an ARPL instruction is a null selector, the instruction generates an exception #13.

Assembler language programs should perform all I/O operations through ROM BIOS or DOS function calls.

- Program interrupts are used for access to these functions. This practice removes the absolute addressing from the program. Only the interrupt number is required.

## 8-10 Compatibility

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

The power-on-self-test code in the system ROM enables hardware IRQ 13 and sets up its vector to point to a routine in ROM. The ROM routine clears the 'busy' signal latch and then transfers control to the address pointed to by the NMI interrupt vector. This allows code written for any IBM Personal Computer to work on an IBM Personal Computer XT Model 286. The NMI interrupt handler should read the coprocessor's status to determine if the NMI was caused by the coprocessor. If the interrupt was not generated by the coprocessor, control should be passed to the original NMI interrupt handler.

- Back to back I/O commands to the same I/O ports will not permit enough recovery time for I/O chips. To ensure enough time, a `JMP SHORT $+2` must be inserted between IN/OUT instructions to the same I/O chip.

**Note:** `MOV AL,AH` type instruction does not allow enough recovery time. An example of the correct procedure follows:

```
OUT  IO_ADD,AL
JMP  SHORT $+2
MOV  AL,AH
OUT  IO_ADD,AL
```

- In systems using the 80286 microprocessor, IRQ 9 is redirected to INT hex 0A (hardware IRQ 2). This insures that hardware designed to use IRQ 2 will operate in the IBM Personal Computer XT Model 286.
- The system can mask hardware sensitivity. New devices can change the ROM BIOS to accept the same programming interface on the new device.

- In cases where BIOS provides parameter tables, such as for video or diskette, a program may substitute new parameter values by building a new copy of the table and changing the vector to point to that table. However, the program should copy the current table, using the current vector, and then modify those locations in the table that need to be changed. In this way, the program will not inadvertently change any values that should be left the same.
- **Disk\_\_Base** consists of 11 parameters required for diskette operation. They are pointed at by the data variable, **Disk\_\_Pointer**, at absolute address 0:78. It is strongly recommended that the values supplied in ROM be used. If it becomes necessary to modify any of the parameters, build another parameter block and modify the address in **Disk\_\_Pointer** to point to the new block.

The parameters were established to operate both the High Capacity Diskette Drive and the Double Sided Diskette Drive. Three of the parameters in this table are under control of BIOS in the following situations.

The Gap Length Parameter is no longer retrieved from the parameter block.

The gap length used during diskette read, write, and verify operations is derived from within diskette BIOS.

The gap length for format operations is still obtained from the parameter block.

Special considerations are required for formatting operations. See the prolog of Diskette BIOS for the required details. If a parameter block contains a head settle time parameter value of 0 milliseconds, and a write operation is being performed, at least 15 milliseconds of head settle time will be enforced for a High Capacity Diskette Drive and 20 milliseconds will be enforced for a Double Sided Diskette Drive. If a parameter block contains a motor start wait parameter of less than 1 second for a write or format operation of 625 milliseconds for a read or verify operation, Diskette BIOS will enforce those times listed above.

- The following procedure is used to determine the type of media inserted in the High Capacity Diskette Drive:
  1. Read Track 0, Head 0, Sector 1 to allow diskette BIOS to establish the media/drive combination. If this is successful, continue with the next step.
  2. Read Track 0, Sector 15. If an error occurs, a double sided diskette is in the drive.

**Note:** Refer to the *DOS Technical Reference* manual for the File Allocation Table (FAT) parameters for single- and double-sided diskettes.  
If a successful read occurs, a high capacity diskette is in the drive.
  3. If Step 1 fails, issue the reset function (AH=0) to diskette BIOS and retry. If a successful read cannot be done, the media needs to be formatted or is defective.

ROM BIOS and DOS do not provide for all functions. The following are the allowable I/O operations with which IBM will maintain compatibility in future systems.

- Control of the sound, using port hex 61, and the sound channel of the timer/counter. A program can control timer/counter channels 0 and 2, ports hex 40, 42, and 43. A program must not change the value in port hex 41, because this port controls the dynamic-memory refresh. Channel 0 provides the time-of-day interrupt, and can also be used for timing short intervals. Channel 2 of the timer/counter is the output for the speaker and cassette ports. This channel may also be used for timing short intervals, although it cannot interrupt at the end of the period.

**Note:** Programs should use the timer for delay on the paddle input rather than a program loop.
- Interrupt Mask Register (IMR), port hex 21, can be used to selectively mask and unmask the hardware features.

The following information pertains to absolute memory locations.

- **Interrupt Vectors Segment (hex 0)**--A program may change these to point at different processing routines. When an interrupt vector is modified, the original value should be retained. If the interrupt, either hardware or program, is not directed toward this device handler, the request should be passed to the next item in the list.
- **Video Display Buffers (hex B000 and B8000)**-- For each mode of operation defined in the video display BIOS, the memory map will remain the same. For example, the bit map for the 320 x 200 medium-resolution graphics mode of the Color/Graphics Monitor adapter will be retained on any future adapter that supports that mode. If the bit map is modified, a different mode number will be used.
- **ROM BIOS Data Area (hex 40:0)**--Any variables in this area will retain their current definition, whenever it is reasonable to do so. IBM may use these data areas for other purposes when the variable no longer has meaning in the system. In general, ROM BIOS data variables should be read or modified through BIOS calls whenever possible, and not with direct access to the variable.

A program that requires timing information should use either the time-of-day clock or the timing channels of the timer/counter. The input frequency to the timer will be maintained at 1.19 MHz, providing a constant time reference. Program loops should be avoided.

Programs that use copy protection schemes should use the ROM BIOS diskette calls to read and verify the diskette and should not be timer dependent. Any method can be used to create the diskette, although manufacturing capability should be considered. The verifying program can look at the diskette controller's status bytes in the ROM BIOS data area for additional information about embedded errors. More information about copy protection may be found on page 8-5 under "Copy Protection".

Any DOS program must be relocatable and insensitive to the size of DOS or its own load addresses. A program's memory requirement should be identified and contiguous with the load module. A program should not assume that all of memory is available to it.

There are several 80286 instructions that, when executed, lock out external bus signals. DMA requests are not honored during the execution of these instructions. Consecutive instructions of this type prevent DMA activity from the start of the first instruction to the end of the last instruction. To allow for necessary DMA cycles, as required by the diskette controller in a multitasking system, multiple lock-out instructions must be separated by `JMP SHORT $+2`.

## Multitasking Provisions

The IBM Personal Computer XT Model 286 BIOS contains a feature to assist multitasking implementation. "Hooks" are provided for a multitasking dispatcher. Whenever a busy (wait) loop occurs in the BIOS, a hook is provided for the program to break out of the loop. Also, whenever BIOS services an interrupt, a corresponding wait loop is exited, and another hook is provided. Thus a program may be written that employs the bulk of the device driver code. The following is valid only in the microprocessor's real address mode and must be taken by the code to allow this support.

The program is responsible for the serialization of access to the device driver. The BIOS code is not reentrant.

The program is responsible for matching corresponding wait and post calls.

## Interfaces

There are four interfaces to be used by the multitasking dispatcher:

## Startup

First, the startup code hooks interrupt hex 15. The dispatcher is responsible to check for function codes of AH= hex 90 or 91. The "Wait" and "Post" sections describe these codes. The dispatcher must pass all other functions to the previous user of interrupt hex 15. This can be done by a JMP or a CALL. If the function code is hex 90 or 91, the dispatcher should do the appropriate processing and return by the IRET instruction.

## Serialization

It is up to the multitasking system to ensure that the device driver code is used serially. Multiple entries into the code can result in serious errors.

## Wait (Busy)

Whenever the BIOS is about to enter a busy loop, it first issues an interrupt hex 15 with a function code of hex 90 in AH. This signals a wait condition. At this point, the dispatcher should save the task status and dispatch another task. This allows overlapped execution of tasks when the hardware is busy. The following is an outline of the code that has been added to the BIOS to perform this function.

```
MOV AX, 90XXH      ; wait code in AH and
                   ; type code in AL
INT 15H           ; issue call
JC TIMEOUT        ; optional: for time-out or
                   ; if carry is set, time-out
                   ; occurred
NORMAL TIMEOUT LOGIC ; normal time-out
```



## Post (Interrupt)

Whenever the BIOS has set an interrupt flag for a corresponding busy loop, an interrupt 15 occurs with a function code of hex 91 in AH. This signals a post condition. At this point, the dispatcher should set the task status to "ready to run" and return to the interrupt routine. The following is an outline of the code added to BIOS that performs this function.

```
MOV AX, 91XXH      ; post code AH and
                   ; type code AL
INT 15H           ; issue call
```

## Classes

The following types of wait loops are supported:

- The class for hex 0 to 7F is serially reusable. This means that for the devices that use these codes, access to the BIOS must be restricted to only one task at a time.
- The class for hex 80 to BF is reentrant. There is no restriction on the number of tasks that may access the device.
- The class for hex C0 to FF is non-interrupt. There is no corresponding interrupt for the wait loop. Therefore, it is the responsibility of the dispatcher to determine what satisfies this condition to exit the loop.

## Function Code Classes

Type Code (AL)	Description
00H->7FH	Serially reusable devices; operating system must serialize access
80H->0BFH	Reentrant devices; ES:BX is used to distinguish different calls (multiple I/O calls are allowed simultaneously)

0C0H->0FH      Wait only calls; there is no complementary POST for these waits--these are time-out only. Times are function-number dependent.

### Function Code Assignments

The following are specific assignments for the IBM Personal Computer XT Model 286 BIOS. Times are approximate. They are grouped according to the classes described under "Function Code Classes".

Type Code (AL)	Time-out	Description
00H	yes (6 second)	fixed disk
01H	yes (2 second)	diskette
02H	no (2 second)	keyboard
0FDH	yes (1 second-write)	diskette motor start
--	(625 ms-read)	--
0FEH	yes (18 second)	printer

The asynchronous support has been omitted. The Serial/Parallel Adapter will generate interrupts, but BIOS does not support it in the interrupt mode. Therefore, the support should be included in the multitasking system code if that device is to be supported.

### Time-Outs

To support time-outs properly, the multitasking dispatcher must be aware of time. If a device enters a busy loop, it generally should remain there for a specific amount of time before indicating an error. The dispatcher should return to the BIOS wait loop with the carry bit set if a time-out occurs.

## Machine-Sensitive Code

Programs may select machine specific features, but they must test for specific machine type. Location of the specific machine identification codes can be found through interrupt 15 function code AH (See 'Configuration Parameters' in BIOS Listing). The code is two bytes. The first byte shows the machine type and the second byte shows the series type. They are as follows:

First Byte	Second Byte	Machine Identification
FF	00	IBM Personal Computer
FE	00	IBM Personal Computer XT
FE	00	IBM Portable Personal Computer
FD	00	IBM PCjr
FC	00	IBM Personal Computer AT
FC	02	IBM Personal Computer XT Model 286
FB	00	IBM Personal Computer XT with 256/640 system board

### Machine Identification Code

IBM will define methods for uniquely determining the specific machine type or I/O feature for any new device.

## Notes:

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

---

## Glossary

This glossary includes terms and definitions from the *IBM Vocabulary for Data Processing, Telecommunications, and Office Systems*, GC20-1699.

$\mu$ . Prefix micro; 0.000 001.

$\mu$ s. Microsecond; 0.000 001 second.

A. Ampere.

ac. Alternating current.

**accumulator.** A register in which the result of an operation is formed.

**active high.** Designates a signal that has to go high to produce an effect. Synonymous with positive true.

**active low.** Designates a signal that has to go low to produce an effect. Synonymous with negative true.

**adapter.** An auxiliary device or unit used to extend the operation of another system.

**address bus.** One or more conductors used to carry the binary-coded address from the processor throughout the rest of the system.

**algorithm.** A finite set of well-defined rules for the solution of a problem in a finite number of steps.

**all points addressable (APA).** A mode in which all points of a displayable image can be controlled by the user.

**alphameric.** Synonym for alphanumeric.

**alphanumeric (A/N).** Pertaining to a character set that contains letters, digits, and usually other characters, such as punctuation marks. Synonymous with alphameric.

**alternating current (ac).** A current that periodically reverses its direction of flow.

**American National Standard Code for Information Interchange (ASCII).** The standard code, using a coded character set consisting of 7-bit coded characters (8 bits including parity check), used for information exchange between data processing systems, data communication systems, and associated equipment. The ASCII set consists of control characters and graphic characters.

**ampere (A).** The basic unit of electric current.

**A/N.** Alphanumeric

**analog.** (1) Pertaining to data in the form of continuously variable physical quantities. (2) Contrast with digital.

**AND.** A logic operator having the property that if P is a statement, Q is a statement, R is a statement,..., then the AND of P, Q, R,...is true if all statements are true, false if any statement is false.

**AND gate.** A logic gate in which the output is 1 only if all inputs are 1.

**AND operation.** The boolean operation whose result has the boolean value 1, if and only if, each operand has the boolean value 1. Synonymous with conjunction.

**APA.** All points addressable.

**ASCII.** American National Standard Code for Information Interchange.

**assemble.** To translate a program expressed in an assembler language into a computer language.

**assembler.** A computer program used to assemble.

**assembler language.** A computer-oriented language whose instructions are usually in one-to-one correspondence with computer instructions.

**asynchronous transmission.** (1) Transmission in which the time of occurrence of the start of each character, or block of characters, is arbitrary; once started, the time of occurrence of each signal representing a bit within a character, or block, has the same relationship to significant instants of a fixed time frame. (2) Transmission in which each information character is individually transmitted (usually timed by the use of start elements and stop elements).

**audio frequencies.** Frequencies that can be heard by the human ear (approximately 15 hertz to 20,000 hertz).

**auxiliary storage.** (1) A storage device that is not main storage. (2) Data storage other than main storage; for example, storage on magnetic disk. (3) Contrast with main storage.

**BASIC.** Beginner's all-purpose symbolic instruction code.

**basic input/output system (BIOS).** The feature of the IBM Personal Computer that provides the level control of the major I/O devices, and relieves the programmer from concern about hardware device characteristics.

**baud.** (1) A unit of signaling speed equal to the number of discrete conditions or signal events per second. For example, one baud equals one bit per second in a train of binary signals, one-half dot cycle per second in Morse code, and one 3-bit value per second in a train of signals each of which can assume one of eight different states. (2) In asynchronous transmission, the unit of modulation rate corresponding to one unit of interval per second; that is, if the duration of the unit interval is 20 milliseconds, the modulation rate is 50 baud.

**BCC.** Block-check character.

**beginner's all-purpose symbolic instruction code (BASIC).** A programming language with a small repertoire of commands and a simple syntax, primarily designed for numeric applications.

**binary.** (1) Pertaining to a selection, choice, or condition that has two possible values or states. (2) Pertaining to a fixed radix numeration system having a radix of 2.

**binary digit.** (1) In binary notation, either of the characters 0 or 1. (2) Synonymous with bit.

**binary notation.** Any notation that uses two different characters, usually the binary digits 0 and 1.

**binary synchronous communications (BSC).** A uniform procedure, using a standardized set of control characters and control character sequences for synchronous transmission of binary-coded data between stations.

**BIOS.** Basic input/output system.

**bit.** Synonym for binary digit

**bits per second (bps).** A unit of measurement representing the number of discrete binary digits transmitted by a device in one second.

**block.** (1) A string of records, a string of words, or a character string formed for technical or logic reasons to be treated as an entity. (2) A set of things, such as words, characters, or digits, treated as a unit.

**block-check character (BCC).** In cyclic redundancy checking, a character that is transmitted by the sender after each message block and is compared with a block-check character computed by the receiver to determine if the transmission was successful.

**boolean operation.** (1) Any operation in which each of the operands and the result take one of two values. (2) An operation that follows the rules of boolean algebra.

**bootstrap.** A technique or device designed to bring itself into a desired state by means of its own action; for example, a machine routine whose first few instructions are sufficient to bring the rest of itself into the computer from an input device.

**bps.** Bits per second.



**BSC.** Binary synchronous communications.

**buffer.** (1) An area of storage that is temporarily reserved for use in performing an input/output operation, into which data is read or from which data is written. Synonymous with I/O area. (2) A portion of storage for temporarily holding input or output data.

**bus.** One or more conductors used for transmitting signals or power.

**byte.** (1) A sequence of eight adjacent binary digits that are operated upon as a unit. (2) A binary character operated upon as a unit. (3) The representation of a character.

**C.** Celsius.

**capacitor.** An electronic circuit component that stores an electric charge.

**Cartesian coordinates.** A system of coordinates for locating a point on a plane by its distance from each of two intersecting lines, or in space by its distance from each of three mutually perpendicular planes.

**CAS.** Column address strobe.

**cathode ray tube (CRT).** A vacuum tube in which a stream of electrons is projected onto a fluorescent screen producing a luminous spot. The location of the spot can be controlled.

**cathode ray tube display (CRT display).** (1) A CRT used for displaying data. For example, the electron beam can be controlled to form alphanumeric data by use of a dot matrix. (2) Synonymous with monitor.

**CCITT.** International Telegraph and Telephone Consultative Committee.

**Celsius (C).** A temperature scale. Contrast with Fahrenheit (F).

**central processing unit (CPU).** Term for processing unit.

**channel.** A path along which signals can be sent; for example, data channel, output channel.

**character generator.** (1) In computer graphics, a functional unit that converts the coded representation of a graphic character into the shape of the character for display. (2) In word processing, the means within equipment for generating visual characters or symbols from coded data.

**character set.** (1) A finite set of different characters upon which agreement has been reached and that is considered complete for some purpose. (2) A set of unique representations called characters. (3) A defined collection of characters.

**characters per second (cps).** A standard unit of measurement for the speed at which a printer prints.

**check key.** A group of characters, derived from and appended to a data item, that can be used to detect errors in the data item during processing.

**clipping.** In computer graphics, removing parts of a display image that lie outside a window.

**closed circuit.** A continuous unbroken circuit; that is, one in which current can flow. Contrast with open circuit.

**CMOS.** Complementary metal oxide semiconductor.

**code.** (1) A set of unambiguous rules specifying the manner in which data may be represented in a discrete form. Synonymous with coding scheme. (2) A set of items, such as abbreviations, representing the members of another set. (3) To represent data or a computer program in a symbolic form that can be accepted by a data processor. (4) Loosely, one or more computer programs, or part of a computer program.

**coding scheme.** Synonym for code.

**collector.** An element in a transistor toward which current flows.

**color cone.** An arrangement of the visible colors on the surface of a double-ended cone where lightness varies along the axis of

the cone, and hue varies around the circumference. Lightness includes both the intensity and saturation of color.

**column address strobe (CAS).** A signal that latches the column addresses in a memory chip.

**compile.** (1) To translate a computer program expressed in a problem-oriented language into a computer-oriented language. (2) To prepare a machine-language program from a computer program written in another programming language by making use of the overall logic structure of the program, or generating more than one computer instruction for each symbolic statement, or both, as well as performing the function of an assembler.

**complement.** A number that can be derived from a specified number by subtracting it from a second specified number.

**complementary metal oxide semiconductor (CMOS).** A logic circuit family that uses very little power. It works with a wide range of power supply voltages.

**computer.** A functional unit that can perform substantial computation, including numerous arithmetic operations or logic operations, without human intervention during a run.

**computer instruction code.** A code used to represent the instructions in an instruction set. Synonymous with machine code.

**computer program.** A sequence of instructions suitable for processing by a computer.

**computer word.** A word stored in one computer location and capable of being treated as a unit.

**configuration.** (1) The arrangement of a computer system or network as defined by the nature, number, and the chief characteristics of its functional units. More specifically, the term configuration may refer to a hardware configuration or a software configuration. (2) The devices and programs that make up a system, subsystem, or network.

**conjunction.** Synonym for AND operation.

**contiguous.** Touching or joining at the edge or boundary; adjacent.

**control character.** A character whose occurrence in a particular context initiates, modifies, or stops a control operation.

**control operation.** An action that affects the recording, processing, transmission, or interpretation of data; for example, starting or stopping a process, carriage return, font change, rewind, and end of transmission.

**control storage.** A portion of storage that contains microcode.

**coordinate space.** In computer graphics, a system of Cartesian coordinates in which an object is defined.

**cps.** Characters per second.

**CPU.** Central processing unit.

**CRC.** Cyclic redundancy check.

**CRT.** Cathode ray tube.

**CRT display.** Cathode ray tube display.

**CTS.** Clear to send. Associated with modem control.

**cursor.** (1) In computer graphics, a movable marker that is used to indicate position on a display. (2) A displayed symbol that acts as a marker to help the user locate a point in text, in a system command, or in storage. (3) A movable spot of light on the screen of a display device, usually indicating where the next character is to be entered, replaced, or deleted.

**cyclic redundancy check (CRC).** (1) A redundancy check in which the check key is generated by a cyclic algorithm. (2) A system of error checking performed at both the sending and receiving station after a block-check character has been accumulated.

**cylinder.** (1) The set of all tracks with the same nominal distance from the axis about which the disk rotates. (2) The

tracks of a disk storage device that can be accessed without repositioning the access mechanism.

**daisy-chained cable.** A type of cable that has two or more connectors attached in series.

**data.** (1) A representation of facts, concepts, or instructions in a formalized manner suitable for communication, interpretation, or processing by human or automatic means. (2) Any representations, such as characters or analog quantities, to which meaning is, or might be assigned.

**data base.** A collection of data that can be immediately accessed and operated upon by a data processing system for a specific purpose.

**data processing system.** A system that performs input, processing, storage, output, and control functions to accomplish a sequence of operations on data.

**data transmission.** Synonym for transmission.

**dB.** Decibel.

**dBa.** Adjusted decibels.

**dc.** Direct current.

**debounce.** (1) An electronic means of overcoming the make/break bounce of switches to obtain one smooth change of signal level. (2) The elimination of undesired signal variations caused by mechanically generated signals from contacts.

**decibel.** (1) A unit that expresses the ratio of two power levels on a logarithmic scale. (2) A unit for measuring relative power.

**decoupling capacitor.** A capacitor that provides a low impedance path to ground to prevent common coupling between circuits.

**Deutsche Industrie Norm (DIN).** (1) German Industrial Norm. (2) The committee that sets German dimension standards.

**digit.** (1) A graphic character that represents an integer; for example, one of the characters 0 to 9. (2) A symbol that

represents one of the non-negative integers smaller than the radix. For example, in decimal notation, a digit is one of the characters 0 to 9.

**digital.** (1) Pertaining to data in the form of digits.  
(2) Contrast with analog.

**DIN.** Deutsche Industrie Norm.

**DIN connector.** One of the connectors specified by the DIN committee.

**DIP.** Dual in-line package.

**DIP switch.** One of a set of small switches mounted in a dual in-line package.

**direct current (dc).** A current that always flows in one direction.

**direct memory access (DMA).** A method of transferring data between main storage and I/O devices that does not require processor intervention.

**disable.** To stop the operation of a circuit or device.

**disabled.** Pertaining to a state of a processing unit that prevents the occurrence of certain types of interruptions. Synonymous with masked.

**disk.** Loosely, a magnetic disk.

**diskette.** A thin, flexible magnetic disk and a semirigid protective jacket, in which the disk is permanently enclosed. Synonymous with flexible disk.

**diskette drive.** A device for storing data on and retrieving data from a diskette.

**display.** (1) A visual presentation of data. (2) A device for visual presentation of information on any temporary character imaging device. (3) To present data visually. (4) See cathode ray tube display.

**display attribute.** In computer graphics, a particular property that is assigned to all or part of a display; for example, low intensity, green color, blinking status.

**display element.** In computer graphics, a basic graphic element that can be used to construct a display image; for example, a dot, a line segment, a character.

**display group.** In computer graphics, a collection of display elements that can be manipulated as a unit and that can be further combined to form larger groups.

**display image.** In computer graphics, a collection of display elements or display groups that are represented together at any one time in a display space.

**display space.** In computer graphics, that portion of a display surface available for a display image. The display space may be all or part of a display surface.

**display surface.** In computer graphics, that medium on which display images may appear; for example, the entire screen of a cathode ray tube.

**DMA.** Direct memory access.

**dot matrix.** (1) In computer graphics, a two-dimensional pattern of dots used for constructing a display image. This type of matrix can be used to represent characters by dots. (2) In word processing, a pattern of dots used to form characters. This term normally refers to a small section of a set of addressable points; for example, a representation of characters by dots.

**dot printer.** Synonym for matrix printer.

**dot-matrix character generator.** In computer graphics, a character generator that generates character images composed of dots.

**drawing primitive.** A group of commands that draw defined geometric shapes.

**DSR.** Data set ready. Associated with modem control.

**DTR.** In the IBM Personal Computer, data terminal ready. Associated with modem control.

**dual in-line package (DIP).** A widely used container for an integrated circuit. DIPs have pins in two parallel rows. The pins are spaced 1/10 inch apart. See also DIP switch.

**duplex.** (1) In data communication, pertaining to a simultaneous two-way independent transmission in both directions. (2) Contrast with half-duplex.

**duty cycle.** In the operation of a device, the ratio of on time to idle time. Duty cycle is expressed as a decimal or percentage.

**dynamic memory.** RAM using transistors and capacitors as the memory elements. This memory requires a refresh (recharge) cycle every few milliseconds. Contrast with static memory.

**EBCDIC.** Extended binary-coded decimal interchange code.

**ECC.** Error checking and correction.

**edge connector.** A terminal block with a number of contacts attached to the edge of a printed-circuit board to facilitate plugging into a foundation circuit.

**EIA.** Electronic Industries Association.

**electromagnet.** Any device that exhibits magnetism only while an electric current flows through it.

**enable.** To initiate the operation of a circuit or device.

**end of block (EOB).** A code that marks the end of a block of data.

**end of file (EOF).** An internal label, immediately following the last record of a file, signaling the end of that file. It may include control totals for comparison with counts accumulated during processing.

**end-of-text (ETX).** A transmission control character used to terminate text.



**end-of-transmission (EOT).** A transmission control character used to indicate the conclusion of a transmission, which may have included one or more texts and any associated message headings.

**end-of-transmission-block (ETB).** A transmission control character used to indicate the end of a transmission block of data when data is divided into such blocks for transmission purposes.

**EOB.** End of block.

**EOF.** End of file.

**EOT.** End-of-transmission.

**EPROM.** Erasable programmable read-only memory.

**erasable programmable read-only memory (EPROM).** A PROM in which the user can erase old information and enter new information.

**error checking and correction (ECC).** The detection and correction of all single-bit errors, plus the detection of double-bit and some multiple-bit errors.

**ESC.** The escape character.

**escape character (ESC).** A code extension character used, in some cases, with one or more succeeding characters to indicate by some convention or agreement that the coded representations following the character or the group of characters are to be interpreted according to a different code or according to a different coded character set.

**ETB.** End-of-transmission-block.

**ETX.** End-of-text.

**extended binary-coded decimal interchange code (EBCDIC).** A set of 256 characters, each represented by eight bits.

**F.** Fahrenheit.

**Fahrenheit (F).** A temperature scale. Contrast with Celsius (C).

**falling edge.** Synonym for negative-going edge.

**FCC.** Federal Communications Commission.

**fetch.** To locate and load a quantity of data from storage.

**FF.** The form feed character.

**field.** (1) In a record, a specified area used for a particular category of data. (2) In a data base, the smallest unit of data that can be referred to.

**field-programmable logic sequencer (FPLS).** An integrated circuit containing a programmable, read-only memory that responds to external inputs and feedback of its own outputs.

**FIFO (first-in-first out).** A queuing technique in which the next item to be retrieved is the item that has been in the queue for the longest time.

**fixed disk drive.** In the IBM Personal Computer, a unit consisting of nonremovable magnetic disks, and a device for storing data on and retrieving data from the disks.

**flag.** (1) Any of various types of indicators used for identification. (2) A character that signals the occurrence of some condition, such as the end of a word. (3) Deprecated term for mark.

**flexible disk.** Synonym for diskette.

**flip-flop.** A circuit or device containing active elements, capable of assuming either one of two stable states at a given time.

**font.** A family or assortment of characters of a given size and style; for example, 10 point Press Roman medium.

**foreground.** (1) In multiprogramming, the environment in which high-priority programs are executed. (2) On a color display screen, the characters as opposed to the background.

**form feed.** (1) Paper movement used to bring an assigned part of a form to the printing position. (2) In word processing, a

function that advances the typing position to the same character position on a predetermined line of the next form or page.

**form feed character.** A control character that causes the print or display position to move to the next predetermined first line on the next form, the next page, or the equivalent.

**format.** The arrangement or layout of data on a data medium.

**FPLS.** Field-programmable logic sequencer.

**frame.** (1) In SDLC, the vehicle for every command, every response, and all information that is transmitted using SDLC procedures. Each frame begins and ends with a flag. (2) In data transmission, the sequence of contiguous bits bracketed by and including beginning and ending flag sequences.

**g.** Gram.

**G.** (1) Prefix giga; 1,000,000,000. (2) When referring to computer storage capacity, 1,073,741,824. ( $1,073,741,824 = 2$  to the 30th power.)

**gate.** (1) A combinational logic circuit having one output channel and one or more input channels, such that the output channel state is completely determined by the input channel states. (2) A signal that enables the passage of other signals through a circuit.

**Gb.** 1,073,741,824 bytes.

**general-purpose register.** A register, usually explicitly addressable within a set of registers, that can be used for different purposes; for example, as an accumulator, as an index register, or as a special handler of data.

**giga (G).** Prefix 1,000,000,000.

**gram (g).** A unit of weight (equivalent to 0.035 ounces).

**graphic.** A symbol produced by a process such as handwriting, drawing, or printing.

**graphic character.** A character, other than a control character, that is normally represented by a graphic.

**half-duplex.** (1) In data communication, pertaining to an alternate, one way at a time, independent transmission. (2) Contrast with duplex.

**hardware.** (1) Physical equipment used in data processing, as opposed to programs, procedures, rules, and associated documentation. (2) Contrast with software.

**head.** A device that reads, writes, or erases data on a storage medium; for example, a small electromagnet used to read, write, or erase data on a magnetic disk.

**hertz (Hz).** A unit of frequency equal to one cycle per second.

**hex.** Common abbreviation for hexadecimal. Also, hexadecimal can be noted as X' '.

**hexadecimal.** (1) Pertaining to a selection, choice, or condition that has 16 possible different values or states. These values or states are usually symbolized by the ten digits 0 through 9 and the six letters A through F. (2) Pertaining to a fixed radix numeration system having a radix of 16.

**high impedance state.** A state in which the output of a device is effectively isolated from the circuit.

**highlighting.** In computer graphics, emphasizing a given display group by changing its attributes relative to other display groups in the same display field.

**high-order position.** The leftmost position in a string of characters. See also most-significant digit.

**hither plane.** In computer graphics, a plane that is perpendicular to the line joining the viewing reference point and the view point and that lies between these two points. Any part of an object between the hither plane and the view point is not seen. See also yon plane.

**housekeeping.** Operations or routines that do not contribute directly to the solution of the problem but do contribute directly to the operation of the computer.

**Hz.** Hertz

**image.** A fully processed unit of operational data that is ready to be transmitted to a remote unit; when loaded into control storage in the remote unit, the image determines the operations of the unit.

**immediate instruction.** An instruction that contains within itself an operand for the operation specified, rather than an address of the operand.

**index register.** A register whose contents may be used to modify an operand address during the execution of computer instructions.

**indicator.** (1) A device that may be set into a prescribed state, usually according to the result of a previous process or on the occurrence of a specified condition in the equipment, and that usually gives a visual or other indication of the existence of the prescribed state, and that may in some cases be used to determine the selection among alternative processes; for example, an overflow indicator. (2) An item of data that may be interrogated to determine whether a particular condition has been satisfied in the execution of a computer program; for example, a switch indicator, an overflow indicator.

**inhibited.** (1) Pertaining to a state of a processing unit in which certain types of interruptions are not allowed to occur. (2) Pertaining to the state in which a transmission control unit or an audio response unit cannot accept incoming calls on a line.

**initialize.** To set counters, switches, addresses, or contents of storage to 0 or other starting values at the beginning of, or at prescribed points in, the operation of a computer routine.

**input/output (I/O).** (1) Pertaining to a device or to a channel that may be involved in an input process, and, at a different time, in an output process. In the English language, "input/output" may be used in place of such terms as "input/output data," "input/output signal," and "input/output terminals," when such usage is clear in a given context. (2) Pertaining to a device

whose parts can be performing an input process and an output process at the same time. (3) Pertaining to either input or output, or both.

**instruction.** In a programming language, a meaningful expression that specifies one operation and identifies its operands, if any.

**instruction set.** The set of instructions of a computer, of a programming language, or of the programming languages in a programming system.

**intensity.** In computer graphics, the amount of light emitted at a display point

**interface.** A device that alters or converts actual electrical signals between distinct devices, programs, or systems.

**interleave.** To arrange parts of one sequence of things or events so that they alternate with parts of one or more other sequences of the same nature and so that each sequence retains its identity.

**interrupt.** (1) A suspension of a process, such as the execution of a computer program, caused by an event external to that process, and performed in such a way that the process can be resumed. (2) In a data transmission, to take an action at a receiving station that causes the transmitting station to terminate a transmission. (3) Synonymous with interruption.

**I/O.** Input/output.

**I/O area.** Synonym for buffer.

**irrecoverable error.** An error that makes recovery impossible without the use of recovery techniques external to the computer program or run.

**joystick.** In computer graphics, a lever that can pivot in all directions and that is used as a locator device.

**k.** Prefix kilo; 1000.

**K.** When referring to storage capacity, 1024. (1024 = 2 to the 10th power.)

**KB.** 1024 bytes.

**key lock.** A device that deactivates the keyboard and locks the cover on for security.

**kg.** Kilogram; 1000 grams.

**kHz.** Kilohertz; 1000 hertz.

**kilo (k).** Prefix 1000

**kilogram (kg).** 1000 grams.

**kilohertz (kHz).** 1000 hertz

**latch.** (1) A simple logic-circuit storage element. (2) A feedback loop in sequential digital circuits used to maintain a state.

**least-significant digit.** The rightmost digit. See also low-order position.

**LED.** Light-emitting diode.

**light-emitting diode (LED).** A semiconductor device that gives off visible or infrared light when activated.

**load.** In programming, to enter data into storage or working registers.

**look-up table (LUT).** (1) A technique for mapping one set of values into a larger set of values. (2) In computer graphics, a table that assigns a color value (red, green, blue intensities) to a color index.

**low power Schottky TTL.** A version (LS series) of TTL giving a good compromise between low power and high speed. See also transistor-transistor logic and Schottky TTL.

**low-order position.** The rightmost position in a string of characters. See also least-significant digit.

**luminance.** The luminous intensity per unit projected area of a given surface viewed from a given direction.

**LUT.** Look-up table.

**m.** (1) Prefix milli; 0.001. (2) Meter.

**M.** (1) Prefix mega; 1,000,000. (2) When referring to computer storage capacity, 1,048,576. ( $1,048,576 = 2$  to the 20th power.)

**mA.** Milliampere; 0.001 ampere.

**machine code.** The machine language used for entering text and program instructions onto the recording medium or into storage and which is subsequently used for processing and printout.

**machine language.** (1) A language that is used directly by a machine. (2) Deprecated term for computer instruction code.

**magnetic disk.** (1) A flat circular plate with a magnetizable surface layer on which data can be stored by magnetic recording. (2) See also diskette.

**main storage.** (1) Program-addressable storage from which instructions and other data can be loaded directly into registers for subsequent execution or processing. (2) Contrast with auxiliary storage.

**mark.** A symbol or symbols that indicate the beginning or the end of a field, of a word, of an item of data, or of a set of data such as a file, a record, or a block.

**mask.** (1) A pattern of characters that is used to control the retention or elimination of portions of another pattern of characters. (2) To use a pattern of characters to control the retention or elimination of portions of another pattern of characters.

**masked.** Synonym for disabled.

**matrix.** (1) A rectangular array of elements, arranged in rows and columns, that may be manipulated according to the rules of matrix algebra. (2) In computers, a logic network in the form of an array of input leads and output leads with logic elements connected at some of their intersections.



**matrix printer.** A printer in which each character is represented by a pattern of dots; for example, a stylus printer, a wire printer. Synonymous with dot printer.

**MB.** 1,048,576 bytes.

**mega (M).** Prefix 1,000,000.

**megahertz (MHz).** 1,000,000 hertz.

**memory.** Term for main storage.

**meter (m).** A unit of length (equivalent to 39.37 inches).

**MFM.** Modified frequency modulation.

**MHz.** Megahertz; 1,000,000 hertz.

**micro ( $\mu$ ).** 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 ( $\mu$ s).** 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 (RGBO).** 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.

**RGBO.** Red-green-blue-intensity.

**rising edge.** Synonym for positive-going edge.

**ROM.** Read-only memory.

**ROM/BIOS.** The ROM resident basic input/output system, which provides the level control of the major I/O devices in the computer system.

**row address strobe (RAS).** A signal that latches the row address in a memory chip.

**RS-232C.** A standard by the EIA for communication between computers and external equipment.

**RTS.** Request to send. Associated with modem control.

**run.** A single continuous performance of a computer program or routine.

**saturation.** In computer graphics, the purity of a particular hue. A color is said to be saturated when at least one primary color (red, blue, or green) is completely absent.

**scaling.** In computer graphics, enlarging or reducing all or part of a display image by multiplying the coordinates of the image by a constant value.

**schematic.** The representation, usually in a drawing or diagram form, of a logical or physical structure.

**Schottky TTL.** A version (S series) of TTL with faster switching speed, but requiring more power. See also transistor-transistor logic and low power Schottky TTL.

**SDL.** Shielded Data Link

**SDLC.** Synchronous Data Link Control.

**sector.** That part of a track or band on a magnetic drum, a magnetic disk, or a disk pack that can be accessed by the magnetic heads in the course of a predetermined rotational displacement of the particular device.

**SERDES.** Serializer/deserializer.

**serial.** (1) Pertaining to the sequential performance of two or more activities in a single device. In English, the modifiers serial and parallel usually refer to devices, as opposed to sequential and consecutive, which refer to processes. (2) Pertaining to the sequential or consecutive occurrence of two or more related activities in a single device or channel. (3) Pertaining to the sequential processing of the individual parts of a whole, such as the bits of a character or the characters of a word, using the same facilities for successive parts. (4) Contrast with parallel.

**serializer/deserializer (SERDES).** A device that serializes output from, and deserializes input to, a business machine.

**setup.** (1) In a computer that consists of an assembly of individual computing units, the arrangement of interconnections between the units, and the adjustments needed for the computer to operate. (2) The preparation of a computing system to perform a job or job step. Setup is usually performed by an operator and often involves performing routine functions, such as mounting tape reels. (3) The preparation of the system for normal operation.

**short circuit.** A low-resistance path through which current flows, rather than through a component or circuit.

**signal.** A variation of a physical quantity, used to convey data.

**sink.** A device or circuit into which current drains.

**software.** (1) Computer programs, procedures, and rules concerned with the operation of a data processing system. (2) Contrast with hardware.

**source.** The origin of a signal or electrical energy.

**square wave.** An alternating or pulsating current or voltage whose waveshape is square.

**square wave generator.** A signal generator delivering an output signal having a square waveform.

**SS.** Start-stop.

---

**start bit.** (1) A signal to a receiving mechanism to get ready to receive data or perform a function. (2) In a start-stop system, a signal preceding a character or block that prepares the receiving device for the reception of the code elements.

**start-of-text (STX).** A transmission control character that precedes a text and may be used to terminate the message heading.

**start-stop system.** A data transmission system in which each character is preceded by a start bit and is followed by a stop bit.

**start-stop (SS) transmission.** (1) Asynchronous transmission such that a group of signals representing a character is preceded by a start bit and followed by a stop bit. (2) Asynchronous transmission in which a group of bits is preceded by a start bit that prepares the receiving mechanism for the reception and registration of a character and is followed by at least one stop bit that enables the receiving mechanism to come to an idle condition pending the reception of the next character.

**static memory.** RAM using flip-flops as the memory elements. Data is retained as long as power is applied to the flip-flops. Contrast with dynamic memory.

**stop bit.** (1) A signal to a receiving mechanism to wait for the next signal. (2) In a start-stop system, a signal following a character or block that prepares the receiving device for the reception of a subsequent character or block.

**storage.** (1) A storage device. (2) A device, or part of a device, that can retain data. (3) The retention of data in a storage device. (4) The placement of data into a storage device.

**strobe.** An instrument that emits adjustable-rate flashes of light. Used to measure the speed of rotating or vibrating objects.

**STX.** Start-of-text.

**symbol.** (1) A conventional representation of a concept. (2) A representation of something by reason of relationship, association, or convention.

**synchronization.** The process of adjusting the corresponding significant instants of two signals to obtain the desired phase relationship between these instants.

**Synchronous Data Link Control (SDLC).** A protocol for management of data transfer over a data link.

**synchronous transmission.** (1) Data transmission in which the time of occurrence of each signal representing a bit is related to a fixed time frame. (2) Data transmission in which the sending and receiving devices are operating continuously at substantially the same frequency and are maintained, by means of correction, in a desired phase relationship.

**syntax.** (1) The relationship among characters or groups of characters, independent of their meanings or the manner of their interpretation and use. (2) The structure of expressions in a language. (3) The rules governing the structure of a language. (4) The relationships among symbols.

**text.** In ASCII and data communication, a sequence of characters treated as an entity if preceded and terminated by one STX and one ETX transmission control character, respectively.

**time-out.** (1) A parameter related to an enforced event designed to occur at the conclusion of a predetermined elapsed time. A time-out condition can be cancelled by the receipt of an appropriate time-out cancellation signal. (2) A time interval allotted for certain operations to occur; for example, response to polling or addressing before system operation is interrupted and must be restarted.

**track.** (1) The path or one of the set of paths, parallel to the reference edge on a data medium, associated with a single reading or writing component as the data medium moves past the component. (2) The portion of a moving data medium such as a drum, or disk, that is accessible to a given reading head position.

**transistor-transistor logic (TTL).** A popular logic circuit family that uses multiple-emitter transistors.

**translate.** To transform data from one language to another.

---

**transmission.** (1) The sending of data from one place for reception elsewhere. (2) In ASCII and data communication, a series of characters including headings and text. (3) The dispatching of a signal, message, or other form of intelligence by wire, radio, telephone, or other means. (4) One or more blocks or messages. For BSC and start-stop devices, a transmission is terminated by an EOT character. (5) Synonymous with data transmission.

**TTL.** Transistor-transistor logic.

**typematic key.** A keyboard key that repeats its function when held pressed.

**V.** Volt.

**vector.** In computer graphics, a directed line segment.

**video.** Computer data or graphics displayed on a cathode ray tube, monitor, or display.

**view point.** In computer graphics, the origin from which angles and scales are used to map virtual space into display space.

**viewing reference point.** In computer graphics, a point in the modeling coordinate space that is a defined distance from the view point.

**viewing transformation.** Operations on the coordinates of an object (usually matrix multiplications) that cause the view of the object to be rotated about any axis, translated (moved without rotating), and/or scaled (changed in size along any or all dimensions). Viewing transformation differs from modeling transformation in that perspective is considered. See also modeling transformation.

**viewplane.** The visible plane of a CRT display screen that completely contains a defined window.

**viewport.** In computer graphics, a predefined part of the CRT display space.

**volt.** The basic practical unit of electric pressure. The potential that causes electrons to flow through a circuit.

**W. Watt.**

**watt.** The practical unit of electric power.

**window.** (1) A predefined part of the virtual space. (2) The visible area of a viewplane.

**word.** (1) A character string or a bit string considered as an entity. (2) See computer word.

**write.** To make a permanent or transient recording of data in a storage device or on a data medium.

**write precompensation.** The varying of the timing of the head current from the outer tracks to the inner tracks of the diskette to keep a constant 'write' signal.

**yon plane.** In computer graphics, a plane that is perpendicular to the line joining the viewing reference point and the view point, and that lies beyond the viewing reference point. Any part of an object beyond the yon plane is not seen. See also hither plane.





## Bibliography

- Microprocessor and Peripheral Handbook
  - INTEL Corporation. *210844.001*
- Introduction to the iAPX 286
  - INTEL Corporation. *210308.001*
- iAPX 286 Operating Systems Writer's Guide
  - INTEL Corporation. *121960.001*
- iAPX 286 Programmer's Reference Manual
  - INTEL Corporation. *210498.001*
- iAPX 286 Hardware Reference Manual
  - INTEL Corporation. *210760.001*
- Numeric Processor Extension Data Sheet
  - INTEL Corporation. *210920*
- 80287 Support Library Reference Manual
  - INTEL Corporation. *122129*
- National Semiconductor Corporation. *NS16450*
- Motorola Microprocessor's Data Manual
  - Motorola Inc. *Series B*

**Notes:**

—

—

—

# Index

## A

AAA 6-8  
AAD 6-9  
AAM 6-9  
AAS 6-8  
access time,  
  track-to-track 8-6  
ACK command 4-13  
Acknowledge (ACK)  
  command 4-13  
ADC 6-6  
ADD 6-6  
address generation,  
  DMA 1-9  
address latch enable 1-35  
address latch enable,  
  buffered 1-32  
address mode  
  real 1-4  
  virtual 1-4  
address space, I/O 1-24  
address, segment 1-4  
addresses, CMOS RAM 1-59  
addresses, page register 1-11  
AEN 1-35  
ALE 8-4  
alternate key 4-37  
AND 6-10  
APL 8-7  
application guidelines 8-7  
arithmetic instructions 6-6,  
  6-25  
ARPL 6-19  
ASCII characters 7-3  
ASCII, extended 4-30

## B

BALE 1-32  
bandwidth 1-7  
BASIC 8-7  
basic assurance test 4-5  
BASIC interrupts 5-6  
BAT (basic assurance  
  test) 4-5  
BAT Completion Code  
  command 4-13  
BAT Failure Code  
  command 4-13  
battery connector 1-75  
BHE 1-9  
BIOS fixed disk  
  parameters 1-65  
BIOS memory map 5-10  
BIOS programming  
  hints 5-10  
BIOS quick reference 5-14  
block diagram  
  keyboard interface 1-54  
  system xiv  
  system board 1-6  
  system timer 1-22  
board, system 1-3  
BOUND 6-16  
break code 4-4  
break key 4-38  
buffer, keyboard 4-4  
buffered address latch  
  enable 1-32  
buffers, video display 8-14

bus cycle 1-7  
busy loop 8-17  
bypassing BIOS 8-6  
byte high enable 1-9

**C**

cabling 4-3  
CALL 6-13  
caps lock key 4-37  
CBW 6-9  
channel, I/O 1-24  
    connectors 1-25  
    pin assignments 1-28  
    signals 1-31  
channels, DMA 1-9  
character codes 4-30  
characters 7-3  
characters keystrokes, and  
    colors 7-14  
classes, wait loop 8-17  
CLC 6-17  
CLD 6-17  
CLEX 6-27  
CLI 6-17  
CLK 1-31  
clock and data signals 4-27  
    data input 4-29  
    data output 4-28  
    data stream 4-27  
clock cycle 1-7  
clock line, keyboard 1-58  
clock, real-time 1-59  
clock, system 1-7  
CMC 6-17  
CMOS RAM 1-59  
CMOS RAM addresses 1-59  
CMOS RAM  
    configuration 1-63  
CMOS RAM I/O  
    operations 1-70

CMP 6-7  
CMPS 6-11  
COBOL 8-7  
code  
    device driver 8-15  
    machine  
        identification 8-19  
        machine-sensitive 8-19  
codes  
    character 4-30  
    extended 4-34  
    multitasking  
        function 8-17  
command codes, DMA  
    controller 1 1-10  
command codes, DMA  
    controller 2 1-11  
commands  
    I/O 8-11  
    keyboard controller 1-56  
commands from the  
    system 4-6  
    Default Disable 4-7  
    Echo 4-7  
    Enable 4-7  
    Read ID 4-7  
    Resend 4-8  
    Reset 4-8  
    Select Alternate Scan  
        Codes 4-8  
    Set All Keys 4-9  
    Set Default 4-9  
    Set Key Type 4-9  
    Set Typematic  
        Rate/Delay 4-11  
    Set/Reset Status  
        Indicators 4-10  
commands to the system  
    ACK (acknowledge) 4-13  
    BAT (basic assurance test)  
        Completion Code 4-13  
    BAT Failure 4-13  
commands to the  
    system 4-13

- Echo 4-13
- Key Detection Error 4-14
- Keyboard ID 4-14
- Overrun 4-14
- Resend 4-14
- comparison instructions 6-23
- compatibility, hardware 8-3
- condition, wait 8-16
- configuration record 1-60
- configuration, CMOS RAM 1-63
- connectors
  - battery 1-75
  - I/O channel 1-25
  - J-1 through J-8 1-27
  - J-1, J-7 and J-8 1-26
  - keyboard 1-75
  - power supply 1-74
  - power supply output 3-6
  - speaker 1-75
  - system board 1-74
- constants instructions 6-24
- control key 4-36
- control transfer
  - instructions 6-13
- control, sound 8-13
- controller, keyboard 1-44
- controllers
  - DMA 1-7, 1-9, 1-10
  - interrupt 1-12
  - refresh 1-7
- Coprocessor controls 1-42
- coprocessor
  - programming 2-3
- coprocessor, math 2-3
- copy protection 8-5, 8-14
- Ctrl state 4-34
- CTS 6-18
- CWD 6-9
- cycle
  - bus 1-7
  - clock 1-7
  - microprocessor 1-7

D

- DACK 0-3 and 5-7 1-35
- DAS 6-8
- data area, ROM BIOS 8-14
- data input 4-29
- data line, keyboard 1-58
- data output 4-28
- data stream 4-27
- data transfer
  - instructions 6-3, 6-22
- data transfer rate,
  - diskette 8-6
- DEC 6-7, 6-8
- decodes, memory 1-12
- DECSTP 6-28
- Default Disable
  - command 4-7
- default segment
  - workspace 5-10
- delay, typematic 4-4
- description 4-3
  - buffer 4-4
  - cabling 4-3
  - key-code scanning 4-4
  - keys 4-4
  - sequencing key-code scanning 4-4
- descriptors 1-5
- device driver code 8-15
- diagnostic checkpoint
  - port 1-42
- direct memory access 1-9
- disk pointer 8-12
- disk\_base 8-6, 8-12
- diskette change signal 8-6
- diskette data transfer
  - rate 8-6
- diskette rotational speed 8-6
- diskette track density 8-6

INDEX

diskette write current 8-6  
DIV 6-9  
divide error exception 8-9  
DMA address generation 1-9  
DMA channels 1-9  
DMA controller 1-7  
DMA controller 1 1-9  
DMA controller 1 command codes 1-10  
DMA controller 2 1-10  
DMA controller 2 command codes 1-11  
DMA controllers 1-9  
DOS 8-7  
DOS function calls 8-10  
DOS interrupts 5-6  
DRQ0-DRQ3 1-34  
DRQ5-DRQ7 1-34

## E

Echo command 4-7, 4-13  
Enable command 4-7  
enable NMI 1-39  
encoding, keyboard 4-30  
ENTER 6-16  
ESC 6-18  
exception, divide error 8-9  
extended ASCII 4-30  
extended codes 4-34

## F

FABS 6-26  
FADD 6-25  
FCHS 6-26  
FCOM 6-23  
FCOMP 6-23

FCOMPP 6-23  
FDIV 6-25  
FIFO 4-4  
FLD 6-22  
FLDLG2 6-24  
FLDLN2 6-24  
FLDL2T 6-24  
FLDP1 6-24  
FLDZ 6-24  
FLD1 6-24  
FMUL 6-25  
FORTRAN 8-7  
FPATAN 6-26  
FPREM 6-26  
FREE 6-28  
French keyboard 4-41  
FRNDINT 6-26  
FSCALE 6-26  
FSQRT 6-25  
FST 6-22  
FSTP 6-22  
FSUB 6-25  
FTST 6-24  
function calls, DOS 8-10  
function codes, multitasking 8-17  
FXAM 6-24  
FXCH 6-23  
FXTRACT 6-26

## G

gap length parameter 8-12  
generator, refresh request 1-22  
German keyboard 4-42  
graphics modes 5-8  
guidelines, application 8-7

## H

hard code 5-10  
hardware compatibility 8-3  
hardware interrupts 5-6  
HLT 6-17  
hooks 8-15

## I

I/O address map 1-38  
I/O address space 1-24  
I/O addresses 1-38  
I/O CH CK 1-32, 1-39  
I/O CH RDY 1-33  
I/O channel 1-24  
    connectors 1-25  
    pin assignments 1-28  
    signals 1-31  
I/O channel check 1-32  
I/O channel connectors 1-28  
I/O channel ready 1-33  
I/O chip select 1-36  
I/O commands 8-11  
I/O CS16 1-36  
I/O port (read/write) 1-40  
I/O ports, keyboard  
    controller 1-58  
I/O read 1-33  
I/O write 1-33  
IDIV 6-9  
IIMUL 6-9  
IMR 8-13  
IMUL 6-8  
IN 6-5  
INC 6-6  
INCSTP 6-28  
input buffer, keyboard  
    controller 1-56

input port, keyboard  
    controller 1-58  
input requirements 3-3  
input, keyboard 4-29  
inputs, power supply 3-3  
INS 6-12  
instructions  
    arithmetic 6-6, 6-25  
    comparison 6-23  
    constants 6-24  
    control transfer 6-13  
    data transfer 6-3, 6-22  
    logic 6-9  
    processor control 6-17  
    protection control 6-18  
    rotate 6-9  
    shift 6-9  
    string manipulation 6-11  
INT 6-16, 6-27  
interfaces, multitasking 8-15  
interrupt controller 1-12  
interrupt mask register 8-13  
interrupt service routine 1-33  
interrupt sharing 1-14  
interrupt vectors 8-14  
interrupt, single step 8-8  
interrupts  
    BASIC 5-6  
    DOS 5-6  
    hardware 5-6  
    program 5-3  
    program interrupt listing  
        (real mode) 5-5  
    sharing 1-14  
    system 1-12  
interrupts, program (real  
    mode) 5-5  
INTO 6-16  
IOR 1-33  
IOW 1-33  
IRET 6-16  
IRQ 2 8-11  
IRQ 9 8-4, 8-11

IRQ3-IRQ15 1-33  
Italian keyboard 4-43

## J

JB/JNAE 6-14  
JBE/JNA 6-14  
JCXZ 6-16  
JE/JZ 6-14  
JL/JNGE 6-14  
JLE/JNG 6-14  
JMP 6-13  
JNB/JAE 6-15  
JNBE/JA 6-15  
JNE/JNZ 6-15  
JNL/JGE 6-15  
JNLE/JG 6-15  
JNO 6-15  
JNP/JPO 6-15  
JNS 6-15  
JO 6-14  
joystick support 5-6  
JP/JPE 6-14  
JS 6-14  
jumper, RAM 1-43

## K

Key Detection Error  
command 4-14  
key-code scanning 4-4  
keyboard 4-48  
clock line 1-58  
connector 1-75  
controller 1-44  
controller  
commands 1-56  
controller I/O ports 1-58  
controller input  
buffer 1-56

controller input port 1-58  
controller output  
buffer 1-56  
controller output  
port 1-58  
controller status  
register 1-54  
controller test inputs 1-58  
data line 1-58  
encoding 4-30  
interface block  
diagram 1-54  
layout 1-47, 4-31  
routine 4-39  
keyboard buffer 4-4  
keyboard data input 4-29  
keyboard data output 4-28  
Keyboard ID command 4-14  
keyboard layouts  
French 4-41  
German 4-42  
Italian 4-43  
keyboard layouts 4-40  
Spanish 4-44  
U.K. English 4-45  
U.S. English 4-46  
keyboard logic diagrams 4-48  
keyboard scan codes 4-15  
keyboard scan-code outputs  
scan code set 1 4-16  
scan code set 2 4-20  
scan code set 3 4-24  
keyboard, French 4-41  
keyboard, German 4-42  
keyboard, Italian 4-43  
keyboard, Spanish 4-44  
keyboard, U.K. English 4-45  
keyboard, U.S. English 4-46  
keys 4-4  
alternate 4-37  
break 4-38  
caps lock 4-37  
combinations 4-37



- control 4-36
- number lock 4-37
- pause 4-38
- print screen 4-38
- scroll lock 4-37
- shift 4-36
- system request 4-38, 5-6

keys, typematic 4-4

**L**

- LAHF 6-5
- LAR 6-19
- layout system board 1-76
- layout, keyboard 1-47, 4-31
- layouts
  - French 4-41
  - German 4-42
  - Italian 4-43
  - layouts 4-40
  - Spanish 4-44
  - U.K. English 4-45
  - U.S. English 4-46
- LA17-LA23 1-31
- LDCW 6-27
- LDENV 6-27
- LDS 6-5
- LEA 6-5
- LEAVE 6-16
- LES 6-5
- LGDT 6-18
- LIDT 6-18
- line contention 4-28
- line protocol 4-6
- LLDT 6-18
- LMSW 6-19
- load current 3-3
- LOCK 6-17
- LODS 6-11
- logic diagrams
  - system board 1-77
- logic instructions 6-9

- LOOP 6-15
- loop, busy 8-17
- LOOPNZ/LOOPNE 6-16
- loops, program 8-14
- LOOPZ/LOOPE 6-15
- LSL 6-19
- LTR 6-18

**M**

- machine identification
  - code 8-19
- machine-sensitive code 8-19
- make code 4-4
- make/break 4-4
- mask on and off 1-39
- MASTER (I) 1-36
- math coprocessor 2-3, 8-11
- math coprocessor
  - controls 1-42
- MEM chip select 1-36
- MEM CS16 1-36
- memory 1-4
- memory decodes 1-12
- memory locations,
  - reserved 5-9
- memory map, BIOS 5-10
- memory module
  - packages 1-24
- MEMR 1-34
- MEMW 1-34
- microprocessor 1-4, 1-7
- microprocessor cycle 1-7
- mode, data stream 1-45, 4-6, 4-27
- modes, graphic 5-8
- modules, RAM 1-24
- modules,
  - ROM/EPROM 1-23
- MOV 6-3
- MOVS 6-11

INDEX

MUL 6-8  
multi-tasking  
  function codes 8-17  
  interfaces 8-15  
  provisions 8-15  
  serialization 8-16  
  startup 8-16

## N

NEG 6-8  
NMI 1-12, 1-39  
NMI controls 1-39  
no load protection 3-4  
non-maskable interrupt 1-39  
NOP 6-26, 6-28  
NOT 6-11  
Num Lock state 4-34  
number lock key 4-37

## O

operations, CMOS RAM  
  I/O 1-70  
OR 6-10  
OSC 1-36  
oscillator 1-36  
OUT 6-5  
output buffer, keyboard  
  controller 1-56  
output port, keyboard  
  controller 1-58  
output voltage sense  
  levels 3-5  
output voltage  
  sequencing 3-4  
output, keyboard 4-28  
outputs, power supply 3-3  
OUTS 6-12

Overrun command 4-14

## P

packages, memory  
  module 1-24  
page register addresses 1-11  
parameter  
  gap length 8-12  
  passing 5-4  
  tables 8-12  
parameters, BIOS fixed  
  disk 1-65  
Pascal 8-7  
pause key 4-38  
performance, system 1-7  
POP 6-4  
POPA 6-4  
POPF 6-6, 8-8  
POR 4-5  
port, diagnostic  
  checkpoint 1-42  
post 8-17  
power good signal 3-4  
power requirements 4-47  
power supply  
  connectors 1-74  
  inputs 3-3  
  output connectors 3-6  
  outputs 3-3  
power-on reset 4-5  
power-on routine 4-5  
  basic assurance test 4-5  
  BAT (basic assurance  
  test) 4-5  
  POR (power-on  
  reset) 4-5  
  power-on reset 4-5  
print screen key 4-38  
priorities, shift key 4-37  
processor control  
  instructions 6-17

program interrupts 5-3  
 program loops 8-14  
 programming hints,  
   BIOS 5-10  
 programming,  
   coprocessor 2-3  
 protected mode 1-5, 5-6  
 protection control  
   instructions 6-18  
 protection, no load 3-4  
 protocol 4-6  
 provisions, multitasking 8-15  
 PTAN 6-26  
 PUSH 6-3  
 PUSH SP 8-8  
 PUSHA 6-4  
 PUSHF 6-6

## Q

quick reference charts  
   BIOS 5-14  
   characters keystrokes, and  
   colors 7-14

## R

R/W memory 1-24  
 RAM jumper 1-43  
 RAM modules 1-24  
 RAM subsystem 1-24  
 RAM, CMOS 1-59  
 rate, typematic 4-4, 4-11  
 Read ID command 4-7  
 real address mode 1-4, 2-5  
 real mode 5-3  
 real-time clock 1-59, 1-60  
 record, configuration 1-60  
 REFRESH 1-35

refresh controller 1-7  
 refresh request  
   generator 1-22  
 regulation tolerance 3-3  
 REP/REPNE,  
   REPZ/REPNZ 6-12  
 requirements, input 3-3  
 Resend command 4-8, 4-14  
 reserved memory  
   locations 5-9  
 reserved scan codes 1-52  
 Reset command 4-8  
 RESET DRV 1-32  
 reset, power-on 4-5  
 reset, system 4-38  
 RET 6-13  
 ROM BIOS 8-10  
 ROM BIOS data area 8-14  
 ROM scan codes 4-30  
 ROM subsystem 1-23  
 ROM/EPROM  
   modules 1-23  
 rotate instructions 6-9  
 rotational, speed 8-6  
 routine, interrupt  
   service 1-33  
 routine, keyboard 4-39  
 RSTOR 6-28

## S

SAHF 6-5  
 SAVE 6-28  
 SA0-SA19 1-31  
 SBB 6-7  
 SBHE 1-35  
 scan code set 1 4-16  
 scan code set 2 4-20  
 scan code set 3 4-24  
 scan code tables (set 1) 4-16  
 scan code tables (set 2) 4-20  
 scan code tables (set 3) 4-24

INDEX

scan code translation 1-46  
 scan codes, keyboard 4-15  
 scan codes, ROM 4-30  
 scanning, key-code  
   sequencing 4-4  
 SCAS 6-11  
 scroll lock key 4-37  
 SDO-SD15 1-32  
 segment address 1-4  
 segments 1-4  
 Select Alternate Scan Codes  
   command 4-8  
 sense levels, output  
   voltage 3-5  
 sequencing key-code  
   scanning 4-4  
 sequencing, output  
   voltage 3-4  
 serialization,  
   multitasking 8-16  
 Set All Keys commands 4-9  
 Set Default command 4-9  
 Set Key Type commands 4-9  
 Set Typematic Rate/Delay  
   command 4-11  
 Set/Reset Status Indicators  
   command 4-10  
 SETPM 6-27  
 SGDT 6-18  
 shift counts 8-9  
 shift instructions 6-9  
 shift key 4-36  
 shift key priorities 4-37  
 Shift state 4-34  
 shift states 4-36  
 SIDT 6-18  
 signals  
   diskette change 8-6  
   I/O channels 1-31  
   power good 3-4  
   system clock 8-4  
 signals, clock and data 4-27  
 single step interrupt 8-8  
 SLDT 6-18  
 SMEMR 1-34  
 SMEMW 1-34  
 SMSW 6-19  
 sound control 8-13  
 Spanish keyboard 4-44  
 speaker 1-43  
 speaker connector 1-75  
 speaker tone generation 1-22  
 special vectors 5-6  
 specifications 4-47  
   power requirements 4-47  
   size 4-47  
   system unit 1-72  
   weight 4-47  
 startup, multitasking 8-16  
 states  
   Ctrl 4-34  
   Num Lock 4-34  
   Shift 4-34, 4-36  
 status register, keyboard  
   controller 1-54  
 STC 6-17  
 STCW 6-27  
 STD 6-17  
 STENV 6-27  
 STI 6-17  
 STOS 6-12  
 STR 6-19  
 stream, data 4-27  
 string manipulation  
   instructions 6-11  
 STSW 6-27  
 STSWAX 6-27  
 SUB 6-7  
 subsystem, RAM 1-24  
 subsystem, ROM 1-23  
 support joystick 5-6  
 switch, display 1-44  
 system BIOS usage 5-3  
 system block diagram xiv  
 system board 1-3  
 system board block  
   diagram 1-6

system board  
  connectors 1-74  
system board layout 1-76  
system board logic  
  diagrams 1-77  
system bus high enable 1-35  
system clock 1-7  
system clock signal 8-4  
system interrupts 1-12  
system performance 1-7  
system request key 4-38, 5-6  
system reset 4-38  
system timer block  
  diagram 1-22  
system timers 1-22

## T

T/C 1-35  
table, translation 1-49  
tables, parameter 8-12  
terminal count 1-35  
TEST 6-10  
test inputs, keyboard  
  controller 1-58  
time-outs 8-18  
timer/counter 1-22  
timer/counters 1-22  
timers, system 1-22  
tone generation,  
  speaker 1-22  
track density, diskette 8-6  
track-to-track access  
  time 8-6  
translation table 1-49  
translation, scan code 1-46  
tri-state 1-36  
type of display switch 1-44  
typematic delay 4-4  
typematic keys 4-4  
typematic rate 4-4, 4-11

## U

U.K. English keyboard 4-45  
U.S. English keyboard 4-46

## V

vectors, special 5-6  
VERR 6-19  
video display buffers 8-14  
virtual address mode 1-4, 2-5

## W

WAIT 6-17  
wait condition 8-16  
wait loop classes 8-17  
workspace, default  
  segment 5-10  
write current, diskette 8-6

## X

XCHG 6-4  
XLAT 6-5  
XOR 6-11

## Y

YL2XP1 6-27

**Z**

zero wait state 1-37

**Numerics**

OVS 1-37  
2XM1 6-26  
80286 1-4  
8042 1-44  
8237A-5 1-9  
8254-2 1-22  
8259A Interrupt 1-12

## Notes:

⌋

⌋

⌋

**Notes:**

—  
)

)

)