



*Personal Computer  
Hardware Reference  
Library*

---

# Technical Reference

6139362

Q

Q

Q



*Personal Computer  
Hardware Reference  
Library*

---

# Technical Reference

## **First Edition (September, 1985)**

**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.

Products are not stocked at the address below. 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, IBM Product Center, or your IBM Marketing Representative.

**The following paragraph applies only to the United States and Puerto Rico:** A Reader's Comment Form is provided at the back of this publication. If the form has been removed, address comments to: IBM Corporation, Personal Computer, P.O. Box 1328-C, Boca Raton, Florida 33432. IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligations whatever.

© Copyright International Business Machines Corporation 1985

# **Federal Communications Commission Radio Frequency Interference Statement**

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

## **CAUTION**

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

# Notes:

# Preface

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

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

This manual consists of nine sections:

- The first three sections describe the hardware aspects of the IBM Personal Computer AT including signal charts and register information.
- Section 4 describes keyboard operation, the commands to and from the system, and the various keyboard layouts.
- Section 5 contains information about the usage of BIOS and a system BIOS listing.
- Section 6 contains instruction sets for the 80286 microprocessor and the 80287 math coprocessor.
- Section 7 provides information about characters, keystrokes, and colors.
- Section 8 has general communications information.
- Section 9 contains information about the compatibility of the IBM Personal Computer AT and the rest of the IBM Personal Computer family.

A glossary of terms and a bibliography of related publications are included.

## Prerequisite Publications

*Guide to Operations* for the IBM Personal Computer AT

## Suggested Reading

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

# Contents

<b>SECTION 1. SYSTEM BOARD</b>	.....	<b>1-1</b>
Memory	.....	1-4
Microprocessor	.....	1-4
System Performance	.....	1-7
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
NMI and Coprocessor Controls	.....	1-38
Other Circuits	.....	1-40
Speaker	.....	1-40
RAM Jumpers	.....	1-40
Display Switch	.....	1-41
Variable Capacitor	.....	1-41
Keyboard Controller	.....	1-42
Real-Time Clock/CMOS RAM Information	.....	1-56
Specifications	.....	1-69
System Unit	.....	1-69
Connectors	.....	1-71
Logic Diagrams - Type 1	.....	1-76
Logic Diagrams - Type 2	.....	1-98
<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 .....	3-4
DC Output Protection .....	3-4
Output Voltage Sequencing .....	3-4
No-Load Operation .....	3-5
Power-Good Signal .....	3-5
Connectors .....	3-7
<b>SECTION 4. KEYBOARD .....</b>	<b>4-1</b>
Description .....	4-3
Power-On Routine .....	4-4
Commands from the System .....	4-5
Commands to the System .....	4-9
Keyboard Scan-Code Outputs .....	4-11
Clock and Data Signals .....	4-12
Keyboard Layouts .....	4-15
Specifications .....	4-22
Logic Diagram .....	4-23
<b>SECTION 5. SYSTEM BIOS .....</b>	<b>5-1</b>
System BIOS Usage .....	5-3
Keyboard Encoding and Usage .....	5-13
Quick Reference .....	5-24
<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 COLORS .....</b>	<b>7-1</b>
Character Codes .....	7-3
Quick Reference .....	7-14

<b>SECTION 8. COMMUNICATIONS .....</b>	<b>8-1</b>
Hardware .....	8-3
Establishing a Communications Link .....	8-5
<b>SECTION 9. IBM PERSONAL COMPUTER</b>	
<b>COMPATIBILITY .....</b>	<b>9-1</b>
Hardware Considerations .....	9-3
System Board .....	9-3
Fixed Disk Drive .....	9-5
Diskette Drive Compatibility .....	9-5
Copy Protection .....	9-5
Application Guidelines .....	9-7
High-Level Language Considerations .....	9-7
Assembler Language Programming Considerations	9-8
Multitasking Provisions .....	9-16
Machine-Sensitive Code .....	9-19
<b>Glossary .....</b>	<b>Glossary-1</b>
<b>Bibliography .....</b>	<b>Bibliography-1</b>
<b>Index .....</b>	<b>Index-1</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 8: Communications .....

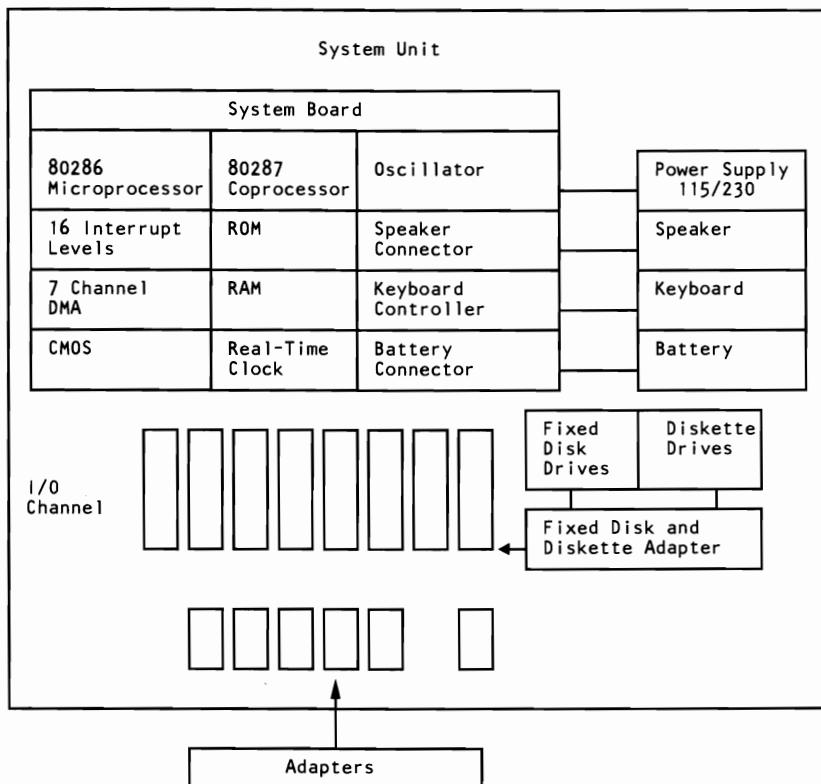
Section 9: Compatibility .....

Glossary .....

Bibliography .....

Index .....

# System Block Diagram



# SECTION 1. SYSTEM BOARD

## Contents

<b>Memory</b> .....	<b>1-4</b>
<b>Microprocessor</b> .....	<b>1-4</b>
Real Address Mode .....	1-4
Protected (Virtual Address) Mode .....	1-5
<b>System Performance</b> .....	<b>1-7</b>
<b>Direct Memory Access</b> .....	<b>1-9</b>
<b>System Interrupts</b> .....	<b>1-12</b>
Hardware Interrupt Listing .....	1-13
Interrupt Sharing .....	1-14
Design Overview .....	1-14
Program Support .....	1-15
Precautions .....	1-17
Examples .....	1-18
<b>System Timers</b> .....	<b>1-22</b>
<b>System Clock</b> .....	<b>1-23</b>
<b>ROM Subsystem</b> .....	<b>1-23</b>
<b>RAM Subsystem</b> .....	<b>1-24</b>
<b>I/O Channel</b> .....	<b>1-24</b>
Connectors .....	1-25
I/O Channel Signal Description .....	1-31
NMI and Coprocessor Controls .....	1-38
<b>Other Circuits</b> .....	<b>1-40</b>
Speaker .....	1-40
RAM Jumpers .....	1-40

Display Switch .....	1-41
Variable Capacitor .....	1-41
Keyboard Controller .....	1-42
Keyboard Controller Initialization .....	1-42
Receiving Data from the Keyboard .....	1-43
Scan Code Translation .....	1-43
Sending Data to the Keyboard .....	1-48
Inhibit .....	1-48
Keyboard Controller System Interface .....	1-48
Status Register .....	1-49
Status-Register Bit Definition .....	1-49
Output Buffer .....	1-50
Input Buffer .....	1-51
Commands (I/O Address Hex 64) .....	1-51
I/O Ports .....	1-54
Real-Time Clock/CMOS RAM Information .....	1-56
Real-Time Clock Information .....	1-57
CMOS RAM Configuration Information .....	1-59
I/O Operations .....	1-68
<b>Specifications .....</b>	<b>1-69</b>
System Unit .....	1-69
Size .....	1-69
Weight .....	1-69
Power Cables .....	1-69
Environment .....	1-69
Heat Output .....	1-70
Noise Level .....	1-70
Electrical .....	1-70
Connectors .....	1-71
Logic Diagrams - Type 1 .....	1-76
Logic Diagrams - Type 2 .....	1-98

The type 1 system board is approximately 30.5 by 35 centimeters (12 by 13.8 inches). The type 2 system board is approximately 23.8 by 35 centimeters (9.3 by 13.8 inches). Both types of system boards use very large scale integration (VLSI) technology and have the following components:

- Intel 80286 Microprocessor
- System support function:
  - Seven-Channel Direct Memory Access (DMA)
  - Sixteen-level interrupt
  - Three programmable timers
  - System clock
- 64K read-only memory (ROM) subsystem, expandable to 128K
- A 512K random-access memory (RAM) Subsystem
- Eight input/output (I/O) slots:
  - Six with a 36-pin and a 62-pin card-edge socket
  - Two with only the 62-pin card-edge socket
- Speaker attachment
- Keyboard attachment
- Complementary metal oxide semiconductor (CMOS) memory RAM to maintain system configuration
- Real-Time Clock
- Battery backup for CMOS configuration table and Real-Time Clock

# Memory

The type 1 system board has four banks of memory sockets, each supporting 9 128K-by-1-bit modules for a total memory size of 512K, with parity checking.

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

# Microprocessor

The Intel 80286 microprocessor has a 24-bit address, 16-bit memory interface<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 or virtual address mode. More detailed descriptions of the microprocessor may be found in the publications listed in the Bibliography of this manual.

## Real Address Mode

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

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

---

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

All segments in the real address mode are 64K in size and may be read, written, or executed. An exception or interrupt can occur if data operands or instructions attempt to wrap around the end of a segment. For example, a word with its low-order byte at offset FFFF and its high-order byte at 0000. If, in the real address mode, the information contained in the segment does not use the full 64K, the unused end of the segment may be overlayed by another segment to reduce physical memory requirements.

## Protected (Virtual Address) Mode

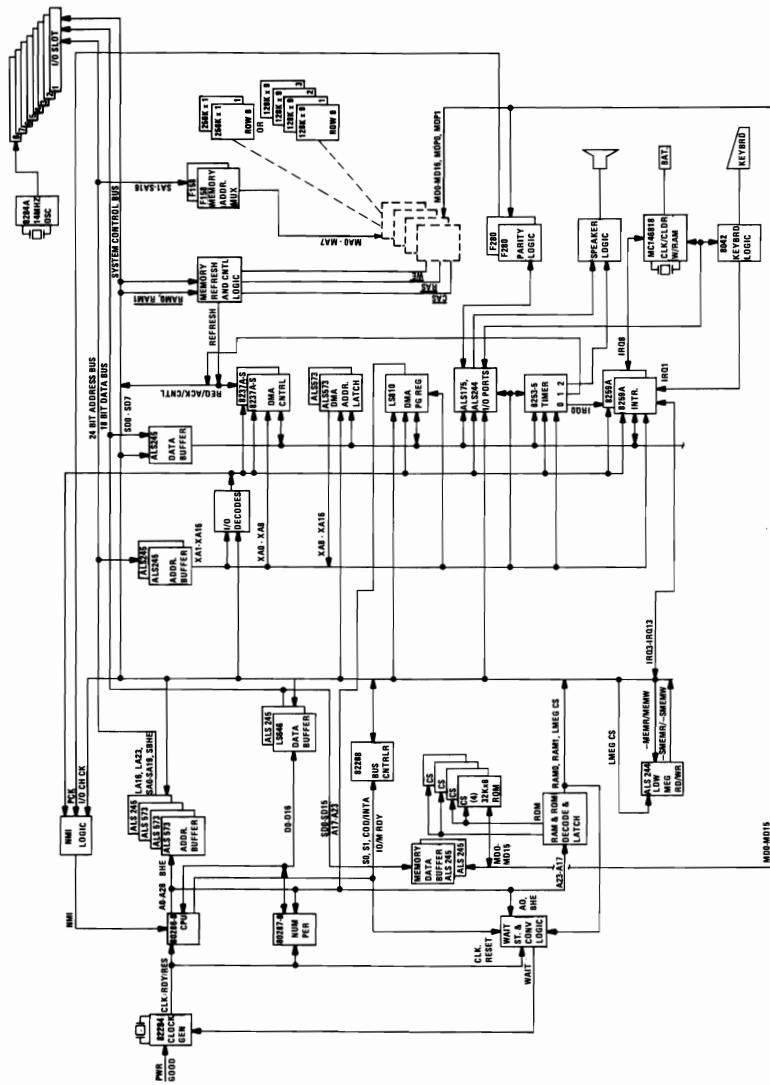
The protected mode offers extended physical and virtual memory address space, memory protection mechanisms, and new operations to support operating systems and virtual memory.

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

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

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

Following is a block diagram of the system board.



# System Performance

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

A bus cycle requires 3 clock cycles (which includes 1 wait state) so that a 500-nanosecond, 16-bit, microprocessor cycle time is achieved. Eight-bit bus operations to 8-bit devices take 6 clock cycles (which include 4 wait states), resulting in a 1000-nanosecond 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 5 clock cycles to refresh all of the system's dynamic memory; 256 refresh cycles are required every 4 milliseconds. The following formula determines the percentage of bandwidth used for refresh.

$$\begin{aligned} \text{\% Bandwidth used for Refresh} &= \frac{5 \text{ cycles} \times 256}{4 \text{ ms}/167 \text{ ns}} = \frac{1280}{24000} = 5.3\% \end{aligned}$$

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

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

The following figure is a system memory map.

Address	Name	Function
000000 to 07FFFF	512K system board	System board memory
080000 to 09FFFF	128K	I/O channel memory - IBM Personal Computer AT 128K Memory Expansion Option
0A0000 to 0BFFFF	128K video RAM	Reserved for graphics display buffer
0C0000 to 0DFFFF	128K I/O expansion ROM	Reserved for ROM on I/O adapters
0E0000 to 0EFFFF	64K reserved on system board	Duplicated code assignment at address FE0000
0F0000 to 0FFFFF	64K ROM on the system board	Duplicated code assignment at address FF0000
100000 to FDFFFF	Maximum memory 15M	I/O channel memory - 512K to 15M installed on memory expansion options
FE0000 to FEFFFF	64K reserved on system board	Duplicated code assignment at address OE0000
FF0000 to FFFFFF	64K ROM on the system board	Duplicated code assignment at address OF0000

### System Memory Map

# Direct Memory Access

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

Controller 1	Controller 2
Ch 0 - Reserved	Ch 4 - Cascade for Ctlr 1
Ch 1 - SDLC	Ch 5 - Reserved
Ch 2 - Diskette (IBM Personal Computer)	Ch 6 - Reserved
Ch 3 - Reserved	Ch 7 - Reserved

## DMA Channels

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

The following figures show address generation for the DMA channels.

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

## Address Generation for DMA Channels 0 through 3

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

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

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

### Address Generation for DMA Channels 5 through 7

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

The following figure shows the addresses for the page register.

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

### Page Register Addresses

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

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

The DMA controller command code addresses follow.

Hex Address	Register Function
0C0	CHO base and current address
0C2	CHO base and current word count
0C4	CH1 base and current address
0C6	CH1 base and current word count
0C8	CH2 base and current address
0CA	CH2 base and current word count
0CC	CH3 base and current address
0CE	CH3 base and current word count
0D0	Read Status Register/Write Command Register
0D2	Write Request Register
0D4	Write Single Mask Register Bit
0D6	Write Mode Register
0D8	Clear Byte Pointer Flip-Flop
0DA	Read Temporary Register/Write Master Clear
0DC	Clear Mask Register
0DE	Write All Mask Register Bits

## DMA Controller

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

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

# System Interrupts

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

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

# Hardware Interrupt Listing

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

Level	Function
Microprocessor NMI	Parity or I/O Channel Check
Interrupt Controllers CTRL 1      CTRL 2	
IRQ 0	Timer Output 0
IRQ 1	Keyboard (Output Buffer Full)
IRQ 2	Interrupt from CTRL 2
IRQ 3	Realtime Clock Interrupt Software Redirected to INT 0AH PC Network * PC Network(Alt.) * Reserved Reserved Reserved Coprocessor Fixed Disk Controller Reserved
IRQ 4	Serial Port 2 BSC BSC (Alt.) Cluster (Primary) PC Network * PC Network (Alt.) * SDLC Serial Port 1 BSC BSC (Alt.) SDLC
IRQ 5	Parallel Port 2
IRQ 6	Diskette Controller
IRQ 7	Fixed Disk and Diskette Drive Parallel Port 1 Data Aquisition and Control *** GPIB ** Cluster (Secondary)

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

## Hardware Interrupt Listing

## Interrupt Sharing

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

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

### Design Overview

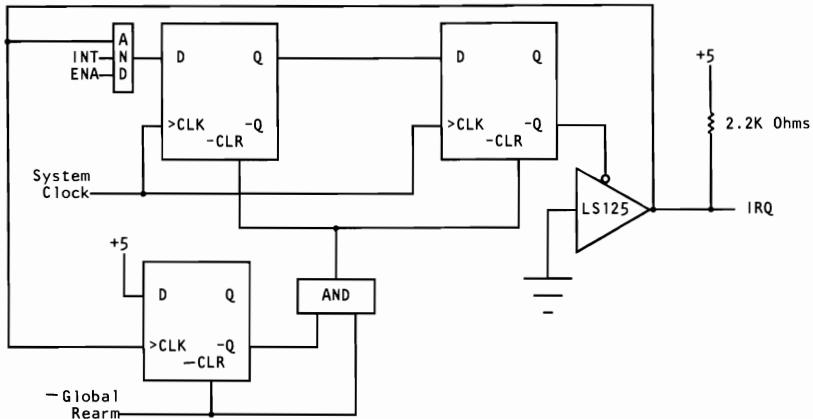
Most interrupt-supporting adapters hold the 'interrupt request' line (IRQ) at a low level and then drive the line high to cause an interrupt. In contrast, the shared-interrupt hardware design allows IRQ to float high through pull-up resistors on each adapter. Each adapter on the line may cause an interrupt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

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

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

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

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



## Shared Interrupt Logic Diagram

## Program Support

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

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

## Linking to a Chain

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

## Sharing the Interrupt Level

When the new task's handler gains control as a result of an interrupt, the handler reads the contents of the adapter's interrupt status register to determine if its adapter caused the interrupt. If it did, the handler services the interrupt, disables the interrupts (CLI), issues a non-specific End of Interrupt (EOI), and then, to rearm the interrupt hardware, writes to address 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 Rarm. An adapter is required to decode the least significant 11 bits for this Global Rarm 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 Rarm.
  - 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      ; Future expansion
        RES_BYTES DB      DUP 7 (0) ; Actual start of code
PAST:   ...
```

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
          ; control status register
          ; Global Rearm location for interrupt
          ; level 7
SPC_EOI  EQU      67H          ; Specific EOI for 8259's interrupt
          ; level 7
EOI       EQU      20H          ; Non-specific EOI
OCR      EQU      20H          ; Location of 8259 operational control
          ; register
IMR      EQU      21H          ; Location of 8259 interrupt mask
          ; register

MYCSEG   SEGMENT PARA
ASSUME  CS:MYCSEG,DS:DSEG
ENTRY    PROC  FAR
          JMP  SHORT PAST          ; Entry point of handler
FPTR     DD      0             ; Forward Pointer
SIGNATURE DW      424BH         ; Used when unlinking to identify
          ; compatible interrupt handlers
FLAGS    DB      0             ; Flags
FIRST   EQU      80H
JMP     SHORT  RESET
RES_BYTES DB      DUP 7 (0)    ; Future expansion
PAST:    STI
          PUSH  ...
          MOV   DX, YOUR_CARD     ; Select your status register
          IN    AL, DX             ; Read the status register
TEST:    TEST  AL, ISB          ; Your card caused the interrupt?
JNZ     SERVICE          ; Yes, branch to service logic
TEST:    TEST  CS:FLAGS, FIRST ; Are we the first ones in?
JNZ     EXIT             ; If yes, branch for EOI and Rearm
POP     ...
CLI
JMP     DWORD PTR CS:FPTR    ; Pass control to next guy on chain

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

```

## Linking Code Example

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

## Unlinking Code Example

```

PUSH    DS
PUSH    ES
CLI     ; Disable interrupts
MOV     AX,350FH ; DOS get interrupt vector
INT     21H      ; ES:BX points to first of chain
MOV     CX,ES    ; Pickup segment part of interrupt vector
; Are we the first handler in the chain?
MOV     AX,CS    ; Get code seg into comparable register
CMP     BX,OFFSET ENTRY ; Interrupt vector in low memory
; pointing to your handler's offset?
JNE     UNCHAIN_A ; No, branch
CMP     AX,CX    ; Vector pointing to your
; handler's segment?
JNE     UNCHAIN_A ; No, branch
; Set interrupt vector in low memory to point to the handler
; pointed to by your pointer

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

UNCHAIN_A: ; BX = FPTR offset, ES = FPTR segment, CX = CS
CMP     ES:[BX+6],4B42H ; Is handler using the appropriate
; conventions (is SIGNATURE present in
; the interrupt chaining structure)?
JNE     exception ; No, invoke error exception handler
LDS     SI,ES:[BX+2] ; Get FPTR's segment and offset
CMP     SI,OFFSET ENTRY ; Is this forward pointer pointing to
; your handler's offset?
JNE     UNCHAIN_B ; No, branch
MOV     CX,DS ; Move to compare
CMP     AX,CX ; Is this forward pointer pointing to
; your handler's segment?
JNE     UNCHAIN_B ; No, branch
; Located your handler in the chain
MOV     AX,WORD PTR CS:FPTR ; Get your FPTR's offset
MOV     ES:[BX+2],AX ; Replace offset of FPTR of handler
; that points to you
MOV     AX,WORD PTR CS:FPTR[2] ; Get your FPTR's segment
MOV     ES:[BX+4],AX ; Replace segment of FPTR of handler
; that points to you
MOV     AL,CS:FLAGS ; Get your flags
AND     AL,FIRST ; Isolate FIRST flag
OR     ES:[BX + 6],AL ; Set your first flag into prior routine
JMP     UNCHAIN_X

UNCHAIN_B: MOV     BX,SI ; Move new offset to BX
PUSH    DS
PUSH    ES
JMP     UNCHAIN_A ; Examine next handler in chain
UNCHAIN_X: STI    ; Enable interrupts
POP     ES
POP     DS

```

# System Timers

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

## Channel 0      **System Timer**

GATE 0      Tied on  
CLK IN 0      1.190 MHz OSC  
CLK OUT 0      8259A IRQ 0

## Channel 1      **Refresh Request Generator**

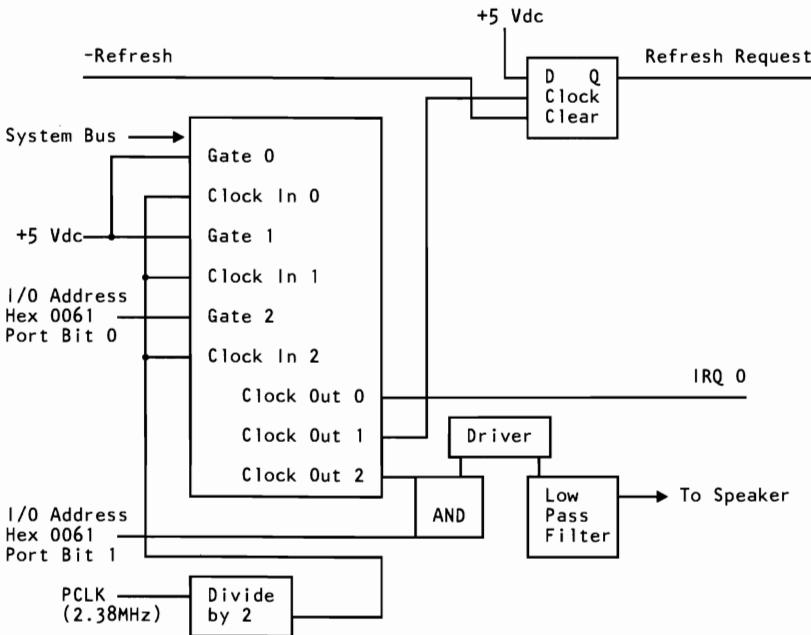
GATE 1      Tied on  
CLK IN 1      1.190 MHz OSC  
CLK OUT 1      Request refresh cycle

**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      1.190 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, the coprocessor, and I/O channel.

## ROM Subsystem

The system board's ROM subsystem consists of two 32K by 8-bit ROM/EPROM modules in a 32K-by-16-bit arrangement. The code for odd and even addresses resides in separate modules. ROM is assigned at the top of the first and last 1M address space (0F0000 and FF0000). ROM is not parity-checked. Its access time is 150 nanoseconds and its cycle time is 230 nanoseconds.

## RAM Subsystem

The system board's RAM subsystem starts at address 000000 of the 16M address space. It is 512K of 128K-by-1-bit RAM modules (type 1 system board) or 512K of 256K-by-1-bit RAM modules (type 2 system board). Memory access time is 150 nanoseconds and the cycle time is 275 nanoseconds.

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

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

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

## I/O Channel

The I/O channel supports:

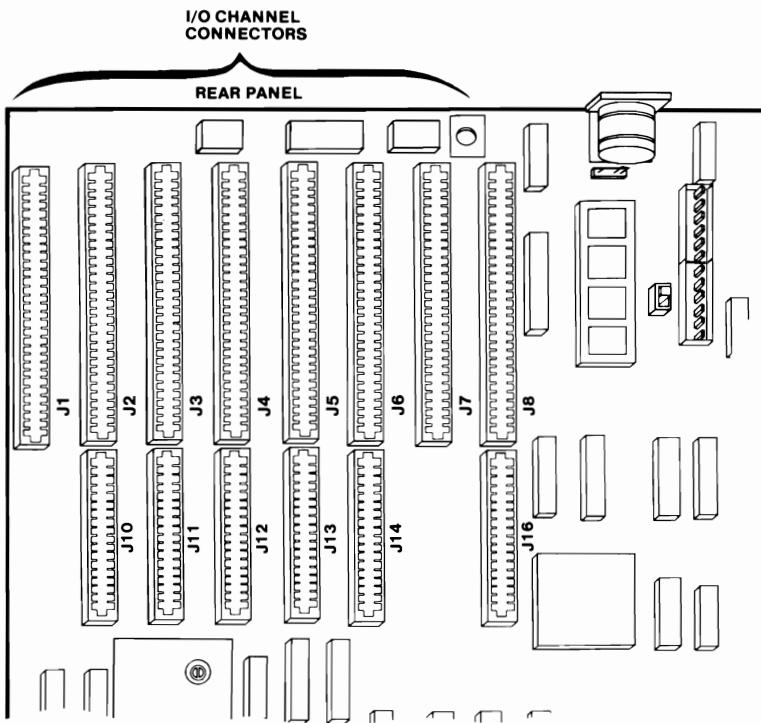
- I/O address space hex 100 to hex 3FF
- 24-bit memory addresses (16M)
- Selection of data accesses (either 8- or 16-bit)
- Interrupts
- DMA channels
- I/O wait-state generation

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

## Connectors

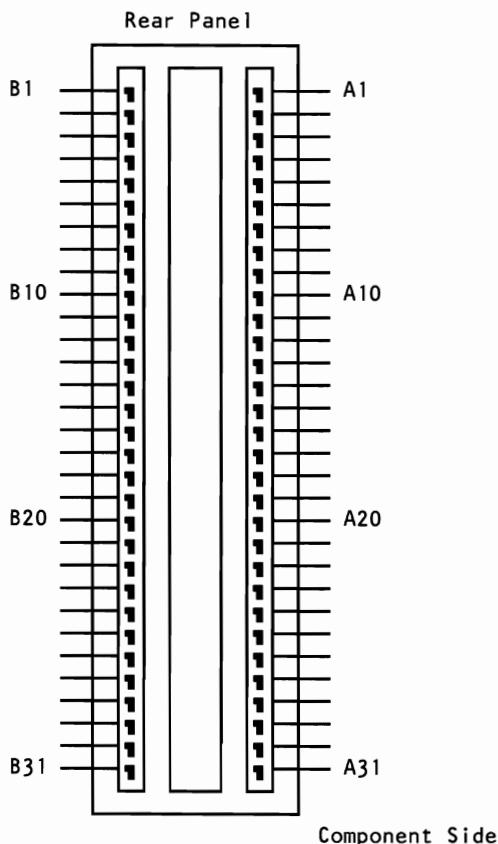
The following figure shows the location and the numbering of the I/O channel connectors. These connectors consist of six 36-pin and eight 62-pin edge connector sockets.

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



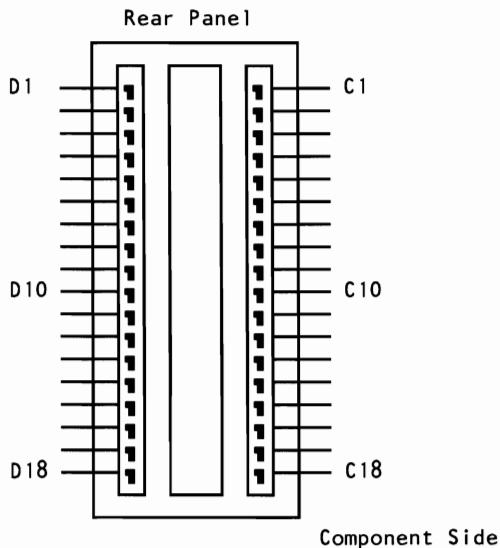
### I/O Channel Connector Location

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



**I/O Channel Pin Numbering (J1-J8)**

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



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

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

I/O Pin	Signal Name	I/O
A1	-I/O CH CK	I
A2	SD7	I/O
A3	SD6	I/O
A4	SD5	I/O
A5	SD4	I/O
A6	SD3	I/O
A7	SD2	I/O
A8	SD1	I/O
A9	SD0	I/O
A10	-I/O CH RDY	I
A11	AEN	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, J1 through J8)**

I/O Pin	Signal Name	I/O
B1	GND	Ground
B2	RESET DRV	0
B3	+5 Vdc	Power
B4	IRQ 9	1
B5	-5 Vdc	Power
B6	DRQ2	1
B7	-12 Vdc	Power
B8	OWS	1
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	1
B17	-DACK1	0
B18	DRQ1	1
B19	-REFRESH	I/O
B20	CLK	0
B21	IRQ7	1
B22	IRQ6	1
B23	IRQ5	1
B24	IRQ4	1
B25	IRQ3	1
B26	-DACK2	0
B27	T/C	0
B28	BALE	0
B29	+5Vdc	Power
B30	OSC	0
B31	GND	Ground

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

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

#### I/O Channel (C-Side, J10 through J14 and 16)

I/O Pin	Signal Name	I/O
D1	-MEM CS16	—
D2	-I/O CS16	—
D3	IRQ10	—
D4	IRQ11	—
D5	IRQ12	—
D6	IRQ15	—
D7	IRQ14	—
D8	-DACK0	0
D9	DRQ0	—
D10	-DACK5	0
D11	DRQ5	—
D12	-DACK6	0
D13	DRQ6	—
D14	-DACK7	0
D15	DRQ7	—
D16	+5 Vdc	POWER
D17	-MASTER	—
D18	GND	GROUND

#### I/O Channel (D-Side, J10 through J14 and 16)

# I/O Channel Signal Description

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

## SA0 through SA19 (I/O)

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

## LA17 through LA23 (I/O)

These signals (unlatched) are used to address memory and I/O devices within the system. They give the system up to 16M of addressability. These signals are valid when BALE is high. LA17 through LA23 are not latched during microprocessor cycles and therefore do not stay valid for the whole cycle. Their purpose is to generate memory decodes for 16-bit, 1 wait-state, memory cycles. These decodes should be latched by I/O adapters on the falling edge of BALE.

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

## CLK (O)

This is the 6-MHz 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 bus bits 0 through 15 for the microprocessor, memory, and I/O devices. D0 is the least-significant bit and D15 is the most-significant bit. All 8-bit devices on the I/O channel should use D0 through D7 for communications to the microprocessor. The 16-bit devices will use D0 through D15. To support 8-bit devices, the data on D8 through D15 will be gated to D0 through D7 during 8-bit transfers to these devices; 16-bit microprocessor transfers to 8-bit devices will be converted to two 8-bit transfers.

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

The 'buffered address latch enable' signal is provided by the 82288 Bus Controller and is used on the system board to latch valid addresses and memory decodes from the microprocessor. It is available to the I/O channel as an indicator of a valid microprocessor or DMA address (when used with 'address enable' signal, AEN). Microprocessor addresses SA0 through SA19 are latched with the falling edge of BALE. BALE is forced high (active) during DMA cycles.

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

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

## I/O CH RDY (I)

The 'I/O channel ready' signal is pulled low (not ready) by a memory or I/O device to lengthen I/O or memory cycles. Any slow device using this line should drive it low immediately upon detecting its valid address and a Read or Write command.

Machine cycles are extended by an integral number of clock cycles (167 nanoseconds). This signal should be held low for no more than 2.5 microseconds.

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

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

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

## -IOR (I/O)

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

## -IOW (I/O)

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

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

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

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

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

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

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

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

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

### **AEN (O)**

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

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

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

### **T/C (O)**

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

### **SBHE (I/O)**

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

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

This signal is used with a DRQ line to gain control of the system. A processor or DMA controller on the I/O channel may issue a DRQ to a DMA channel in cascade mode and receive a -DACK. Upon receiving the -DACK, a microprocessor may pull

-MASTER active (low), which will allow it to control the system address, data, and control lines (a condition known as *tri-state*). After -MASTER is low, the microprocessor must wait one clock cycle before driving the address and data lines, and two clock cycles before issuing a Read or Write command. If this signal is held low for more than 15 microseconds, the system memory may be lost because of a lack of refresh.

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

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

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

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

### **OSC (O)**

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

### **0WS (I)**

The 'zero wait state' signal tells the microprocessor that it can complete the present bus cycle without inserting any additional wait cycles. In order to run a memory cycle to a 16-bit device without wait cycles, 0WS is derived from an address decode gated with a Read or Write command. In order to run a memory cycle to an 8-bit device with a minimum of two wait states, 0WS should

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

The following figure is an I/O address map.

Hex Range	Device
000-01F	DMA controller 1, 8237A-5
020-03F	Interrupt controller 1, 8259A, Master
040-05F	Timer, 8254-2
060-06F	8042 (Keyboard)
070-07F	Real-time clock, NMI (non-maskable interrupt) mask
080-09F	DMA page register, 74LS612
0A0-0BF	Interrupt Controller 2, 8259A
0C0-0DF	DMA controller 2, 8237A-5
0F0	Clear Math Coprocessor Busy
0F1	Reset Math Coprocessor
0F8-0FF	Math Coprocessor

**Note:** I/O Addresses, hex 000 to OFF, are reserved for the system board I/O. Hex 100 to 3FF are available on the I/O channel.

### I/O Address Map (Part 1 of 2)

Hex Range	Device
1F0-1F8	Fixed Disk
200-207	Game I/O
20C-20D	Reserved
21F	Reserved
278-27F	Parallel printer port 2
2B0-2DF	Alternate Enhanced Graphics Adapter
2E1	GPIB (Adapter 0)
2E2 & 2E3	Data Acquisition (Adapter 0)
2F8-2FF	Serial port 2
300-31F	Prototype card
360-363	PC Network (low address)
364-367	Reserved
368-36B	PC Network (high address)
36C-36F	Reserved
378-37F	Parallel printer port 1
380-38F	SDLC, bisynchronous 2
390-393	Cluster
3A0-3AF	Bisynchronous 1
3B0-3BF	Monochrome Display and Printer Adapter
3C0-3CF	Enhanced Graphics Adapter
3D0-3DF	Color/Graphics Monitor Adapter
3F0-3F7	Diskette controller
3F8-3FF	Serial port 1
6E2 & 6E3	Data Acquisition (Adapter 1)
790-793	Cluster (Adapter 1)
AE2 & AE3	Data Acquisition (Adapter 2)
B90-B93	Cluster (Adapter 2)
EE2 & EE3	Data Acquisition (Adapter 3)
1390-1393	Cluster (Adapter 3)
22E1	GPIB (Adapter 1)
2390-2393	Cluster (Adapter 4)
42E1	GPIB (Adapter 2)
62E1	GPIB (Adapter 3)
82E1	GPIB (Adapter 4)
A2E1	GPIB (Adapter 5)
C2E1	GPIB (Adapter 6)
E2E1	GPIB (Adapter 7)

Note: I/O Addresses, hex 000 to OFF, are reserved for the system board I/O. Hex 100 to 3FF are available on the I/O channel.

## I/O Address Map (Part 2 of 2)

## NMI and Coprocessor Controls

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

- |                 |   |
|-----------------|---|
| <b>Mask On</b>  | Write to I/O address hex 070, with data bit 7 equal to a logic 0. |
| <b>Mask Off</b> | Write to I/O address hex 070, with data bit 7 equal to a logic 1. |

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

The following is a description of the Math Coprocessor controls.

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

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

The '-I/O channel check' signal (-I/O CH CK) is used to report non-correctable errors on RAM adapters on the I/O channel. This check will create an NMI if the NMI is enabled. At power-on time, the NMI is masked off and -I/O CH CK is disabled. Follow these steps when enabling -I/O CH CK and the NMI.

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

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

When a check occurs, an interrupt (NMI) will result. Read the status bits to determine the source of the NMI (see the figure, "I/O Address Map", on page 1-37). To determine the location of the failing adapter, write to any memory location within a given

adapter. If the parity check was from that adapter, -I/O CH CK will be reset to inactive.

## Other Circuits

### Speaker

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

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

### RAM Jumpers

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

Pin	Assignments
1	No Connection
2	- RAM SEL
3	Ground

**RAM Jumper Connector (J18)**

The following shows how the jumpers affect RAM.

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

### RAM Jumper

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

## Display Switch

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

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

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

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

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

## Variable Capacitor

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

## Keyboard Controller

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

Data is sent to 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. 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.

## Receiving Data from the Keyboard

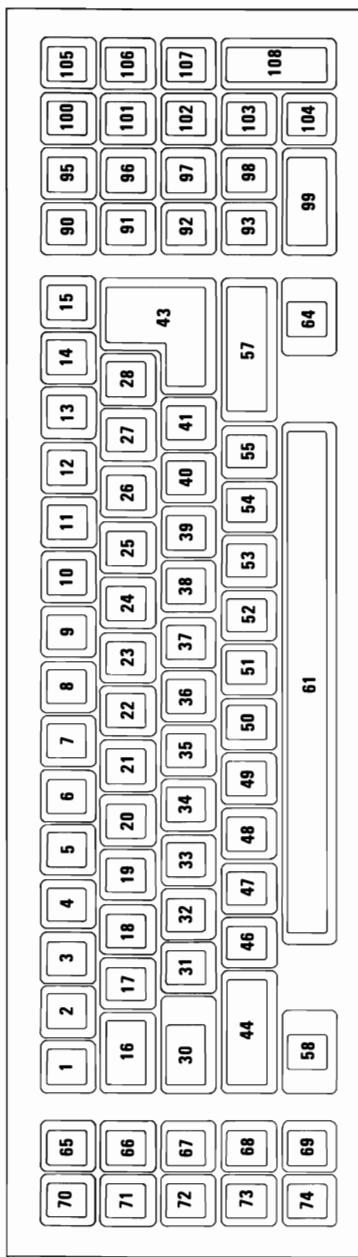
The keyboard sends data in a serial format using an 11-bit frame. The first bit is a start bit, and is followed by eight data bits, an odd parity bit, and a stop bit. Data sent is synchronized by a clock supplied by the keyboard. At the end of a transmission, the keyboard controller disables the interface 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 two milliseconds, a hex FF is placed in the keyboard controller's output buffer, and the receive time-out bit in the status register is set. No retries will be attempted on a receive time-out error.

**Note:** When a receive error occurs in the default mode (bits 5, 6, and 7 of the command byte set to 0), hex 00 is placed in the output buffer instead of hex FF. See "Commands (I/O Address Hex 64)" on page 1-51 for a detailed description of the command byte.

## Scan Code Translation

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

# Keyboard



The following figure is the scan-code translation table.

System Scan Code	Keyboard Scan Code	Key
01	76	90
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	30
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	14
2C	1A	46
2D	22	47
2E	21	48
2F	2A	49

**Scan-Code Translation Table (Part 1 of 2)**

System Scan Code	Keyboard Scan Code	Key
30	32	50
31	31	51
32	3A	52
33	41	53
34	49	54
35	4A	55
36	59	57
38	11	58
39	29	61
3A	58	64
3B	05	70
3C	06	65
3D	04	71
3E	0C	66
3F	03	72
40	08	67
41	02 or 83	73
42	0A	68
43	01	74
44	09	69
45	77	95
46	7E	100
47	6C	91
48	75	96
49	7D	101
4A	78	107
4B	68	92
4C	73	97
4D	74	102
4E	79	108
4F	69	93
50	72	98
51	7A	103
52	70	99
53	71	104
54	7F or 84	105

**Scan-Code Translation Table (Part 2 of 2)**

The following scan codes are reserved.

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

**Reserved Scan-Code Translation Table**

## Sending Data to the Keyboard

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

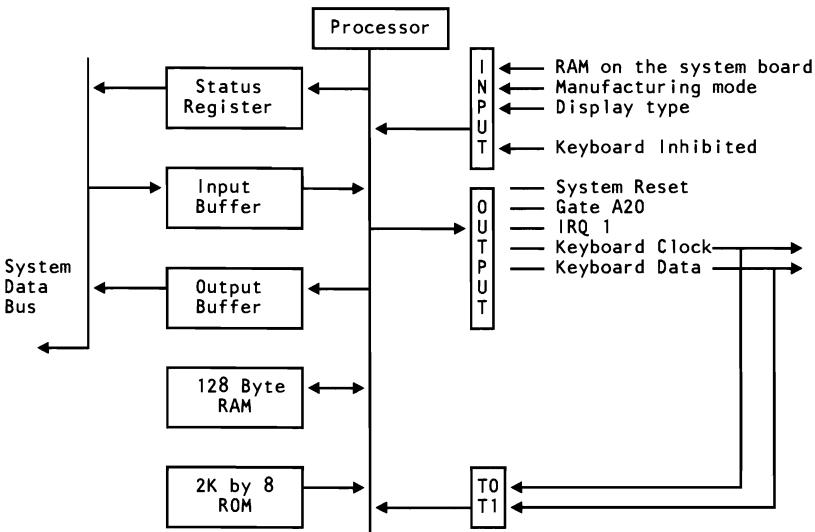
The keyboard is required to respond to all transmissions. The keyboard responds to any valid command and parameter, other than Echo and Resend, with an Acknowledge (ACK) response, hex FA. If the response contains a parity error, the keyboard controller places a hex FE in its output buffer, and the transmit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

## Inhibit

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

## Keyboard Controller System Interface

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



**Keyboard Controller Interface Block Diagram**

## Status Register

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

## Status-Register Bit Definition

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

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

- Bit 4** Inhibit Switch—This bit is updated whenever data is placed in the keyboard controller's output buffer. It reflects the state of the keyboard-inhibit switch. A 0 indicates the keyboard is inhibited.
- Bit 3** Command/Data—The keyboard controller's input buffer may be addressed as either I/O address hex 60 or 64. Address hex 60 is defined as the data port, and address hex 64 is defined as the command port. Writing to address hex 64 sets this bit to 1; writing to address hex 60 sets this bit to 0. The controller uses this bit to determine if the byte in its input buffer should be interpreted as a command byte or a data byte.
- Bit 2** System Flag—This bit is monitored by the system during the reset routine. If it is a 0, the reset was caused by a power on. The controller sets this bit to 0 at power on and it is set to 1 after a successful self test. This bit can be changed by writing to the system flag bit in the command byte (hex 64).
- Bit 1** Input Buffer Full—A 0 indicates that the keyboard controller's input buffer (I/O address hex 60 or 64) is empty. A 1 indicates that data has been written into the buffer but the controller has not read the data. When the controller reads the input buffer, this bit will return to 0.
- Bit 0** Output Buffer Full—A 0 indicates that the keyboard controller's output buffer has no data. A 1 indicates that the controller has placed data into its output buffer but the system has not yet read the data. When the system reads the output buffer (I/O address hex 60), this bit will return to a 0.

## Output Buffer

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

## Input Buffer

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

## Commands (I/O Address Hex 64)

- 20**      Read Keyboard Controller's Command Byte—The controller sends its current command byte to its output buffer.
- 60**      Write Keyboard Controller's Command Byte—The next byte of data written to I/O address hex 60 is placed in the controller's command byte. Bit definitions of the command byte are as follows:
  - Bit 7**    Reserved—Should be written as a 0.
  - Bit 6**    IBM Personal Computer Compatibility Mode—Writing a 1 to this bit causes the controller to convert the scan codes received from the keyboard to those used by the IBM Personal Computer. This includes converting a 2-byte break sequence to the 1-byte IBM Personal Computer format.

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

- AC** Diagnostic Dump—Sends 16 bytes of the controller's RAM, the current state of the input port, the current state of the output port, and the controller's program status word to the system. All items are sent in scan-code format.
- AD** Disable Keyboard Feature—This command sets bit 4 of the controller's command byte. This disables the keyboard interface by driving the clock line low. Data will not be sent or received.
- AE** Enable Keyboard Interface—This command clears bit 4 of the command byte, which releases the keyboard interface.
- C0** Read Input Port—This commands the controller to read its input port and place the data in its output buffer. This command should be used only if the output buffer is empty.
- D0** Read Output Port—This command causes the controller to read its output port and place the data in its output buffer. This command should be issued only if the output buffer is empty.
- D1** Write Output Port—The next byte of data written to I/O address hex 60 is placed in the controller's output port.

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

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

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

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

## I/O Ports

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

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

Bit 7	Keyboard inhibit switch 0 = Keyboard inhibited 1 = Keyboard not inhibited
Bit 6	Display switch - Primary display attached to: 0 = Color/Graphics adapter 1 = Monochrome adapter
Bit 5	Manufacturing Jumper 0 = Manufacturing jumper installed 1 = Jumper not installed
Bit 4	RAM on the system board 0 = Enable 512K of system board RAM 1 = Enable 256K of system board RAM
Bit 3	Reserved
Bit 2	Reserved
Bit 1	Reserved
Bit 0	Reserved

### Input-Port Bit Definitions

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

### Output-Port Bit Definitions

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

### Test-Input Bit Definitions

## Real-Time Clock/CMOS RAM Information

The RT/CMOS RAM chip (Motorola MC146818) contains the real-time clock and 64 bytes of CMOS RAM. The internal clock circuitry uses 14 bytes of this RAM, and the rest is allocated to configuration information. The following figure shows the CMOS RAM addresses.

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

### CMOS RAM Address Map

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

## Real-Time Clock Information

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

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

### Real-Time Clock Information (Addresses 00 - 0D)

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

## Status Register A

- Bit 7** Update in Progress (UIP)—A 1 indicates the time update cycle is in progress. A 0 indicates the current date and time are available to read.
- Bit 6–Bit 4** 22-Stage Divider (DV2 through DV0)—These three divider-selection bits identify which time-base frequency is being used. The system initializes the stage divider to 010, which selects a 32.768-kHz time base.

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

## Status Register B

- Bit 7** Set—A 0 updates the cycle normally by advancing the counts at one-per-second. A 1 aborts any update cycle in progress and the program can initialize the 14 time-bytes without any further updates occurring until a 0 is written to this bit.
- Bit 6** Periodic Interrupt Enable (PIE)—This bit is a read/write bit that allows an interrupt to occur at a rate specified by the rate and divider bits in register A. A 1 enables an interrupt, and a 0 disables it. The system initializes this bit to 0.
- Bit 5** Alarm Interrupt Enable (AIE)—A 1 enables the alarm interrupt, and a 0 disables it. The system initializes this bit to 0.
- Bit 4** Update-Ended Interrupt Enabled (UIE)—A 1 enables the update-ended interrupt, and a 0 disables it. The system initializes this bit to 0.
- Bit 3** Square Wave Enabled (SQWE)—A 1 enables the square-wave frequency as set by the rate selection bits in register A, and a 0 disables the square wave. The system initializes this bit to 0.
- Bit 2** Date Mode (DM)—This bit indicates whether the time and date calendar updates are to use binary or binary coded decimal (BCD) formats. A 1 indicates binary, and a 0 indicates BCD. The system initializes this bit to 0.

- Bit 1** 24/12—This bit indicates whether the hours byte is in the 24-hour or 12-hour mode. A 1 indicates the 24-hour mode and a 0 indicates the 12-hour mode. The system initializes this bit to 1.
- Bit 0** Daylight Savings Enabled (DSE)—A 1 enables daylight savings and a 0 disables daylight savings (standard time). The system initializes this bit to 0.

## Status Register C

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

## Status Register D

- Bit 7** Valid RAM Bit (VRB)—This bit is read-only and indicates the status of the power-sense pin (battery level). A 1 indicates battery power to the real-time clock is good. A 0 indicates the battery is dead, so RAM is not valid.
- Bits 6–Bit 0** Reserved—Should be written as a 0.

## CMOS RAM Configuration Information

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

### Diagnostic Status Byte (Hex 0E)

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

<b>Bit 6</b>	Configuration Record (Checksum Status Indicator)—A 0 indicates that checksum is good, and a 1 indicates it is bad.
<b>Bit 5</b>	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:
	<ul style="list-style-type: none"><li>• At least one diskette drive to be installed (bit 0 of the equipment byte set to 1).</li><li>• The primary display adapter setting in configuration matches the system board's display switch setting and the actual display adapter hardware in the system.</li></ul>
<b>Bit 4</b>	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.
<b>Bit 3</b>	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."
<b>Bit 2</b>	Time Status Indicator (POST validity check)—A 0 indicates that the time is valid, and a 1 indicates that it is invalid.
<b>Bit 1–Bit 0</b>	Reserved

## Shutdown Status Byte (Hex 0F)

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

### Diskette Drive Type Byte (Hex 10)

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

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

**Note:** 0011 through 1111 are reserved.

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

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

**Note:** 0011 through 1111 are reserved.

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

## Fixed Disk Type Byte (Hex 12)

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

- 0000** No fixed disk drive is present.
- 0001** Define type 1 through type 14 as shown to in the following table (also see BIOS listing at label FD\_TBL)
- 1110** Type 16 through 255. See “Drive C Extended Byte (Hex 19)” on page 1-65 .

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

- 0000** No fixed disk drive is present.
- 0001** Define type 1 through type 14 as shown to in the following table (also see BIOS listing at label FD\_TBL)
- 1110** Type 16 through 255. See “Drive D Extended Byte (Hex 1A)” on page 1-65 .

The following figure shows the BIOS fixed disk parameters.

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

### BIOS Fixed Disk Parameters

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

### Equipment Byte (Hex 14)

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

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

**Bit 5–Bit 4**      Primary display

**00**    Primary display is attached to an adapter that has its own BIOS, such as one of the following:

- the Enhanced Graphics Adapter
- the Professional Graphics Controller.

- 01** Primary display is in the 40-column mode and attached to the Color/Graphics Monitor Adapter.
- 10** Primary display is in the 80-column mode and attached to the Color/Graphics Monitor Adapter.
- 11** Primary display is attached to the Monochrome Display and Printer Adapter.

**Bit 3–Bit 2** Not used.

**Bit 1** Math Coprocessor presence bit:

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

**Bit 0** Diskette drive presence bit:

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

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

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

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

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

Valid Sizes:

- 0100H** 256K–system board RAM
- 0200H** 512K–system board RAM
- 0280H** 640K–512K system board RAM and the IBM Personal Computer AT 128KB Memory Expansion Option

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

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

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

### Valid Sizes:

**0200H** 512K-I/O adapter

**0400H** 1024K-I/O adapter (2 adapters)

**0600H** 1536K-I/O adapter (3 adapters)  
through

**3C00H** 15360K I/O adapter (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 figure shows the BIOS fixed disk parameters for fixed disk drive types 16 through 22.

**Note:** Types 23 through 255 are reserved.

Type	Cylinders	Heads	Write Pre-Comp	Landing Zone
16	612	4	A11 Cyl	663
17	977	5	300	977
18	977	7	None	977
19	1024	7	512	1023
20	733	5	300	732
21	733	7	300	732
22	733	7	300	733
23	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-I/O adapter

**0400H** 1024K-I/O adapter

**0600H** 1536K-I/O adapter  
through

**3C00H** 15360K I/O adapter (15M  
maximum).

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

## Date Century Byte (Hex 32)

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

## Information Flag (Hex 33)

**Bit 7** When set, this bit indicates that the top 128K of base memory is installed.

**Bit 6** This bit is set to instruct the Setup utility to put out a first user message after initial setup.

**Bit 5–Bit 0** Reserved

**Hex addresses 34 through 3F are reserved.**

## I/O Operations

Writing to CMOS RAM involves two steps:

1. OUT to port hex 70 with the CMOS address that will be written to.
2. OUT to port hex 71 with the data to be written.

Reading CMOS RAM also requires two steps:

1. OUT to port hex 70 with the CMOS address that is to be read from.
2. IN from port hex 71, and the data read is returned in the AL register.

# Specifications

## System Unit

### Size

- Length: 540 millimeters (21.3 inches)
- Depth: 439 millimeters (17.3 inches)
- Height: 162 millimeters (6.8 inches)

### Weight

- 20.0 kilograms (44 pounds)

## Power Cables

- Length: 1.8 meters (6 feet)

## Environment

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

- Humidity
  - System On: 8% to 80%
  - System Off: 20% to 80%
- Altitude
  - Maximum altitude: 2133.6 meters (7000 feet)

## **Heat Output**

- 1229 British Thermal Units (BTU) per hour

## **Noise Level**

- Meets Class 3; 59 decibels average-noise rating (without printer)

## **Electrical**

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

# Connectors

The system board has the following additional connectors:

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

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

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

## Power Supply Connectors (PS8, PS9)

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

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

### **Speaker Connector (J19)**

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

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

### **Power LED and Key Lock Connector (J20)**

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

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

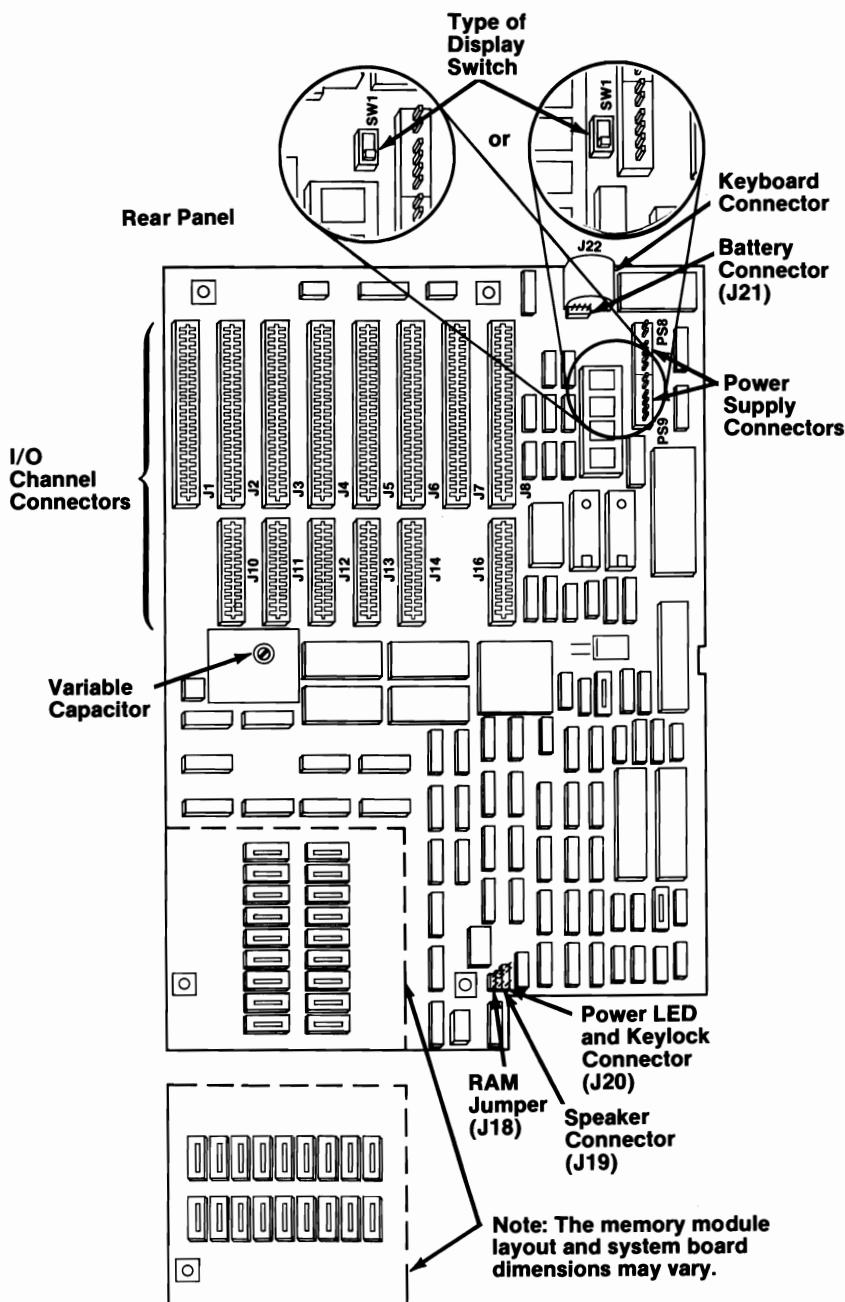
### **Battery Connector (J21)**

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

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

### Keyboard Connector (J22)

The following figure shows the layout of the system board.



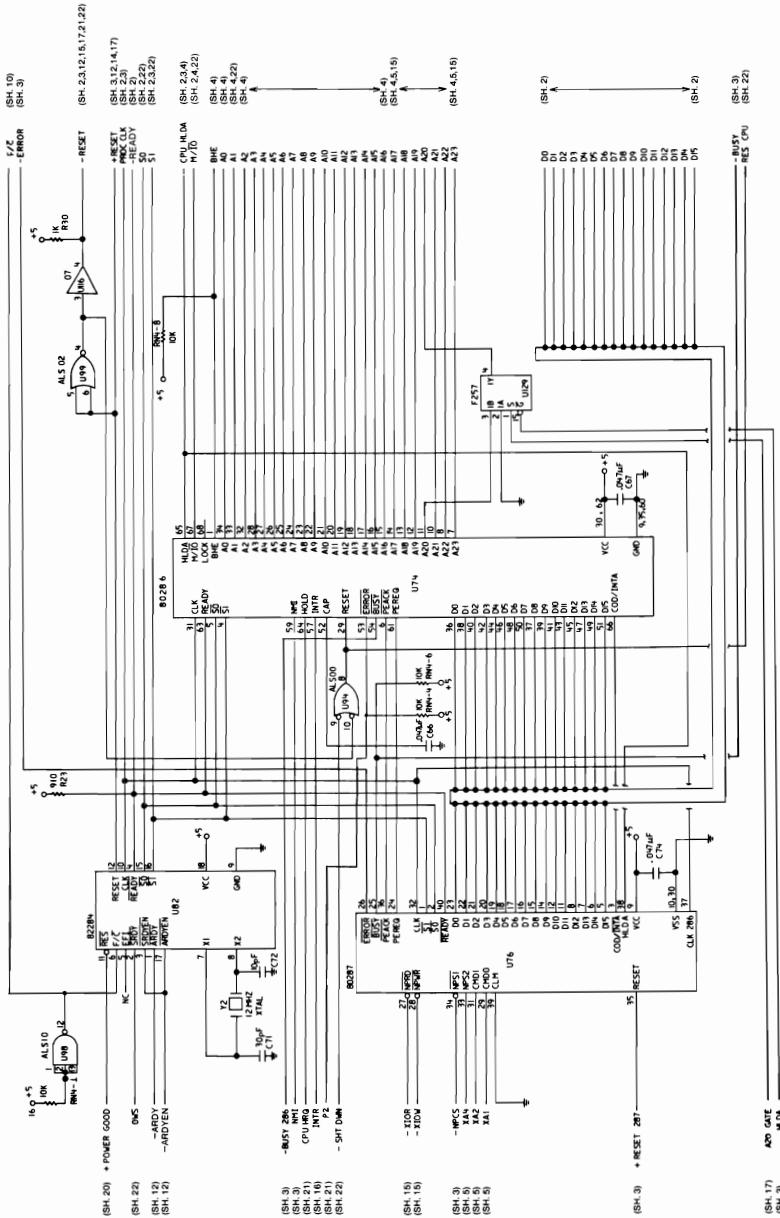
## Notes:

(

(

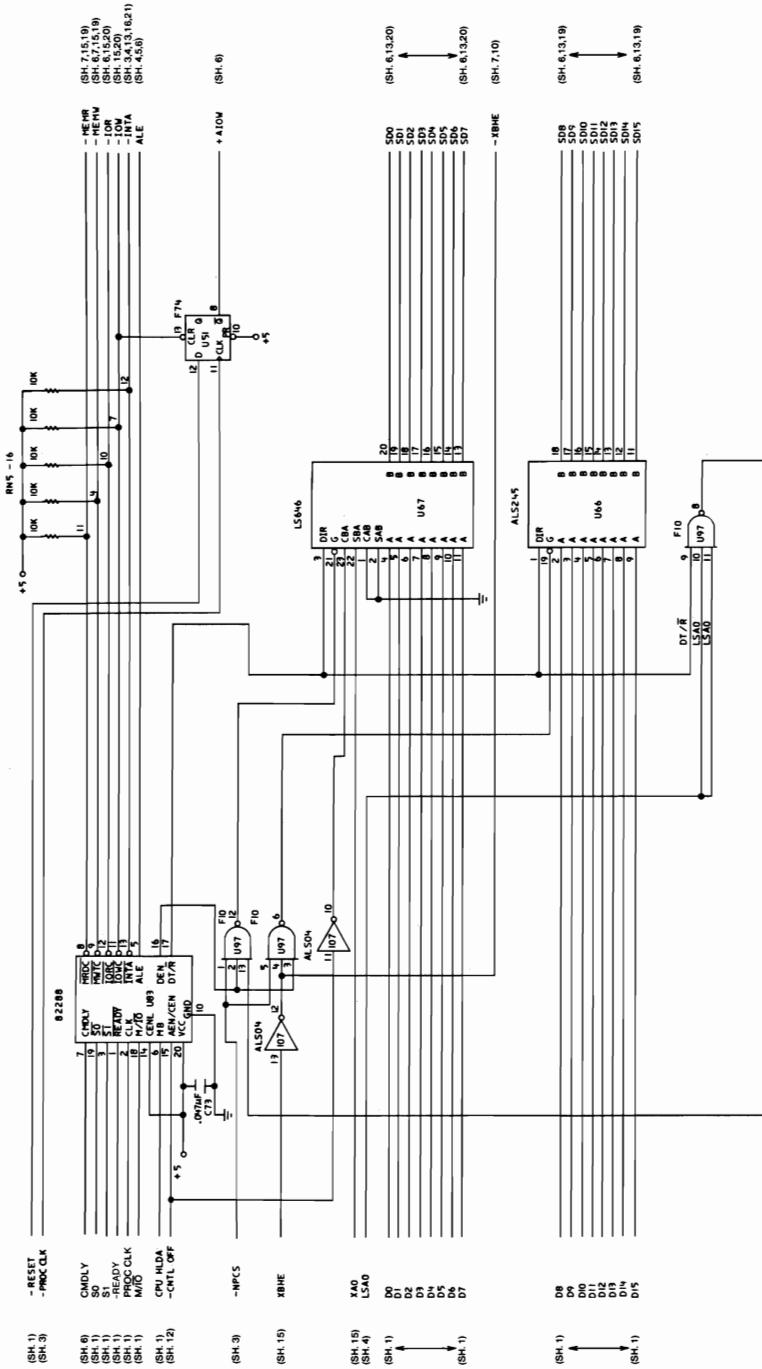
(

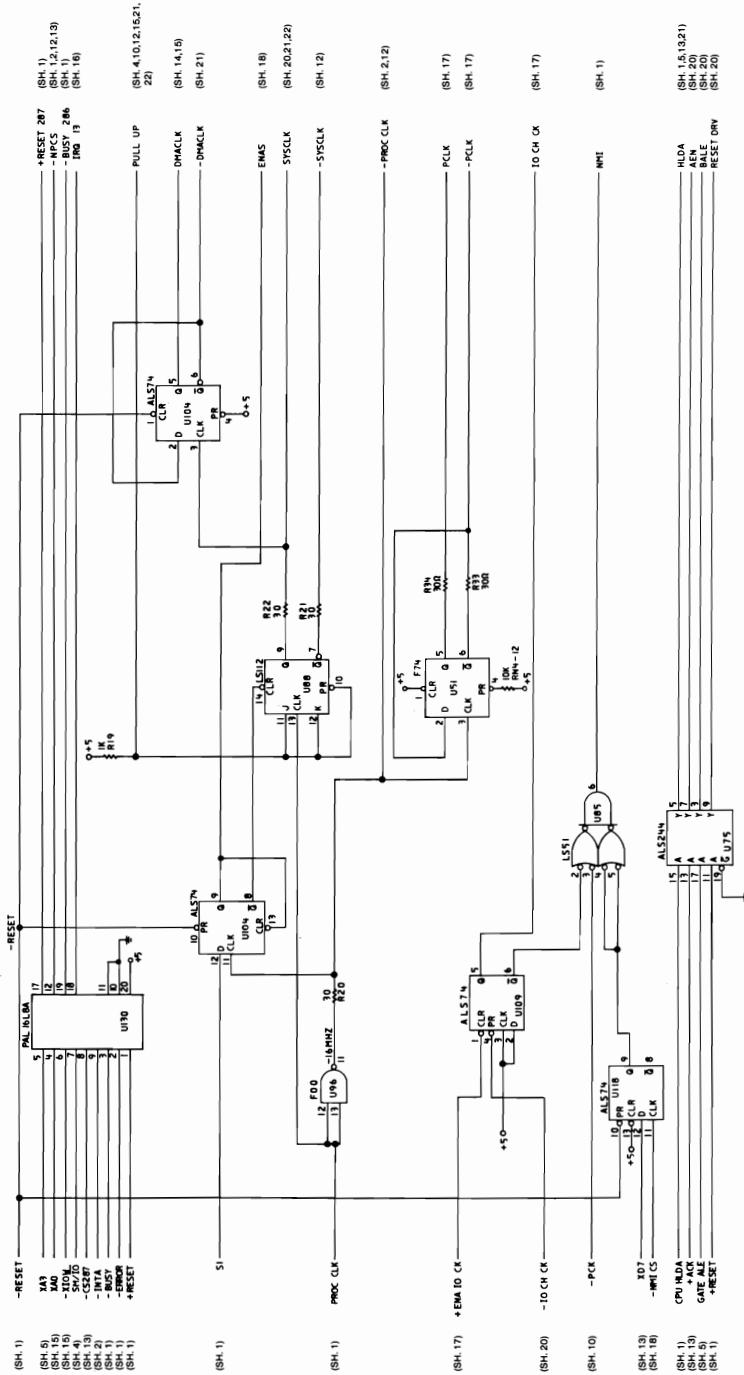
# Logic Diagrams - Type 1



Type 1 512KB Planar (Sheet 1 of 22)

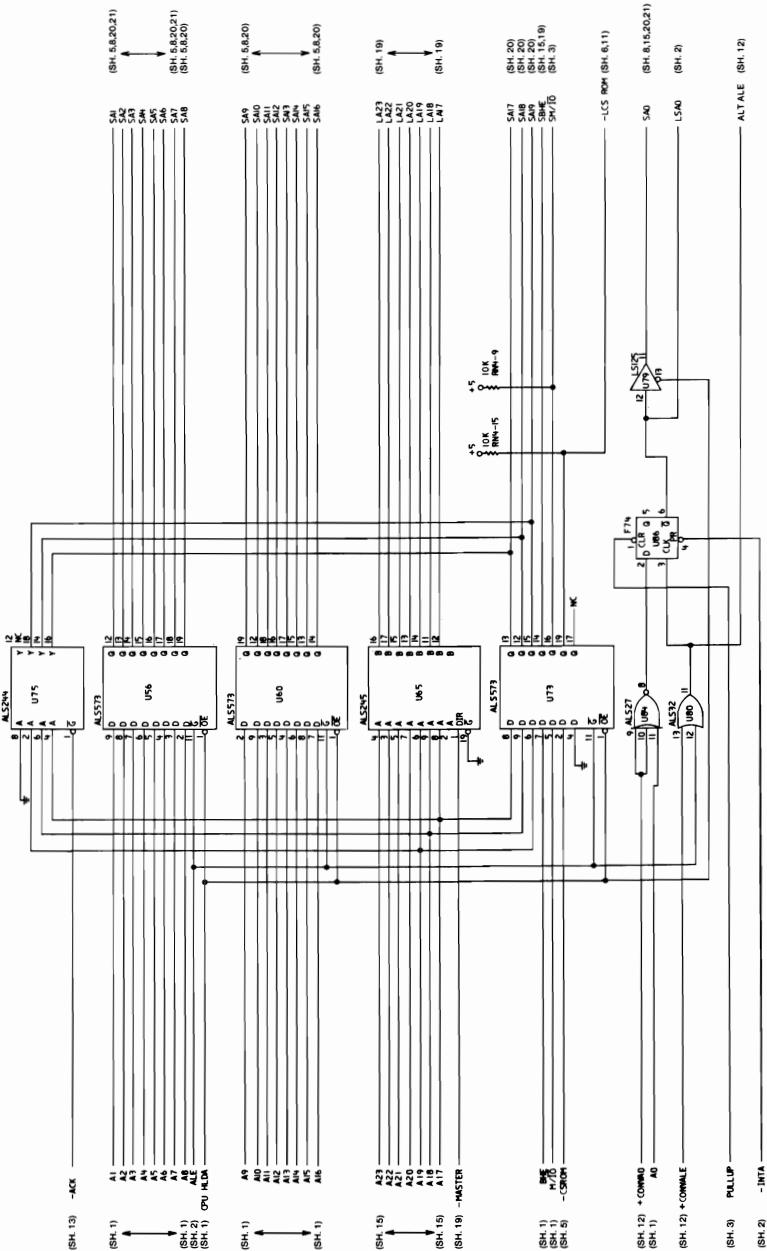
Type 1 512KB Planar (Sheet 2 of 22)

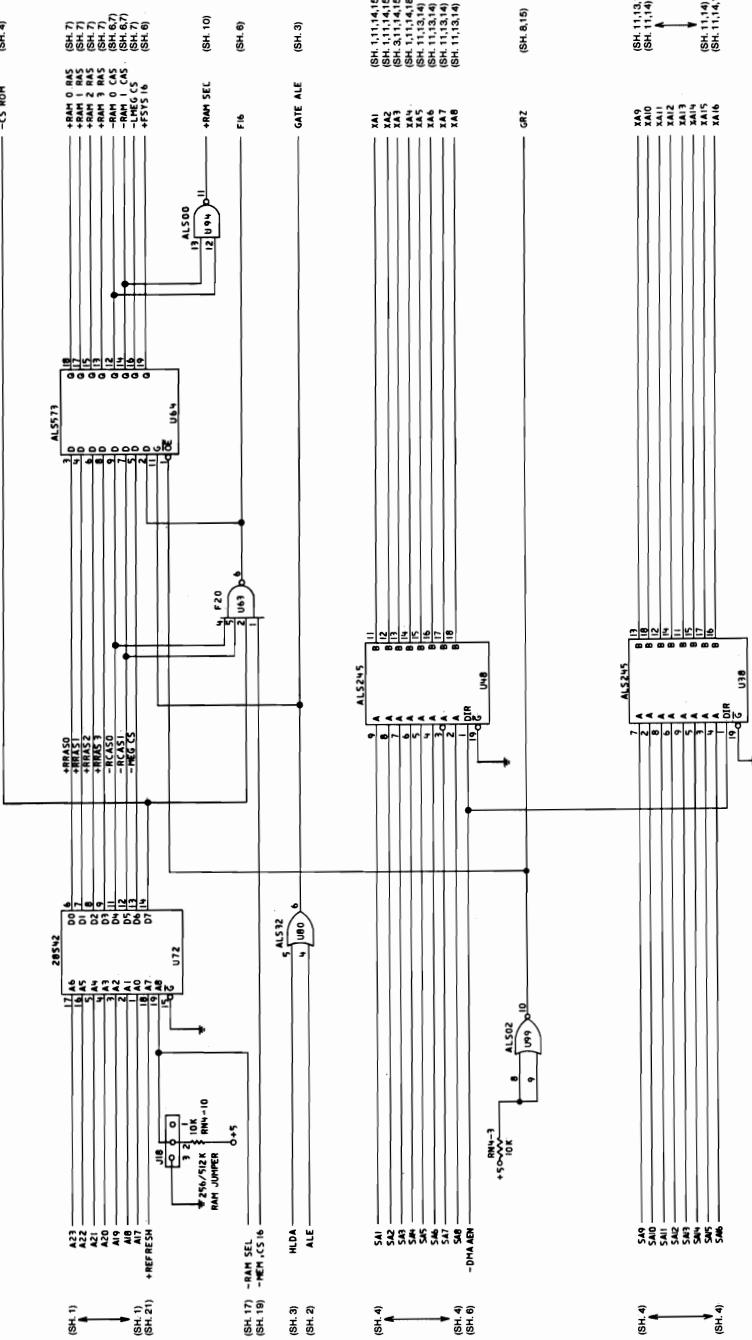




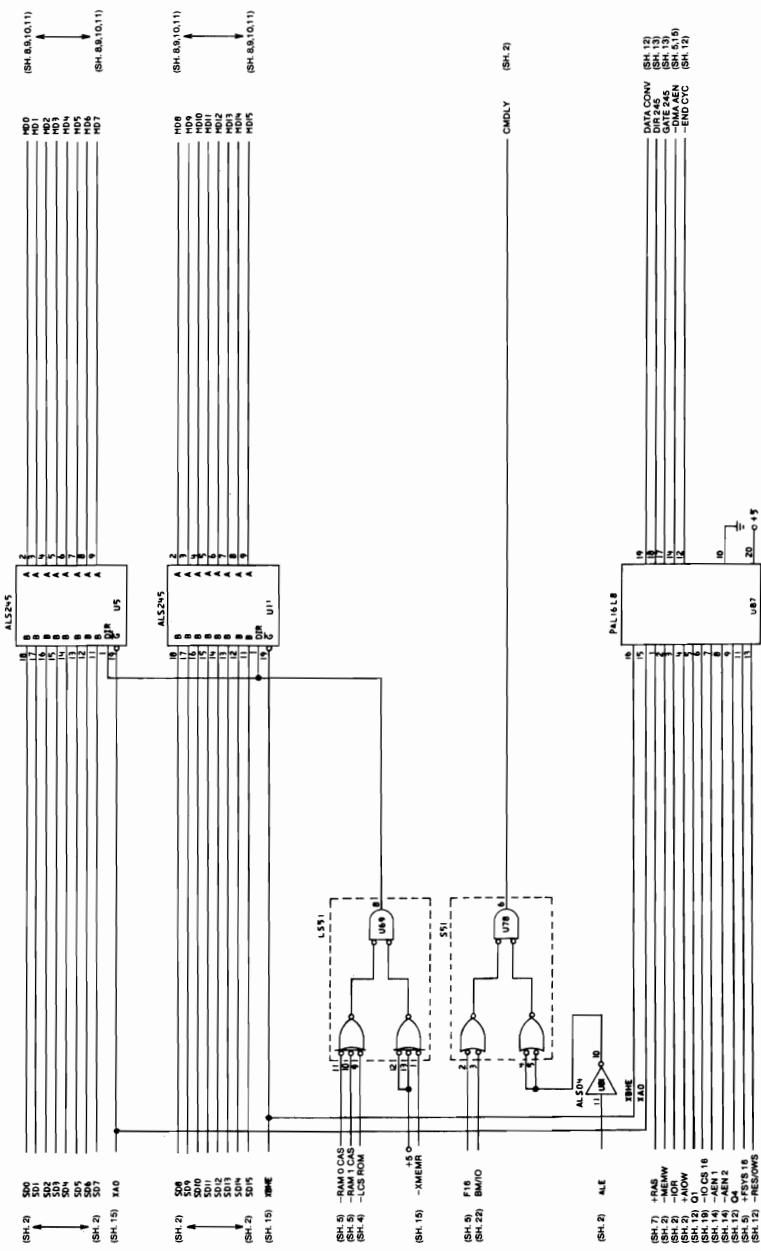
Type 1 512KB Planar (Sheet 3 of 22)

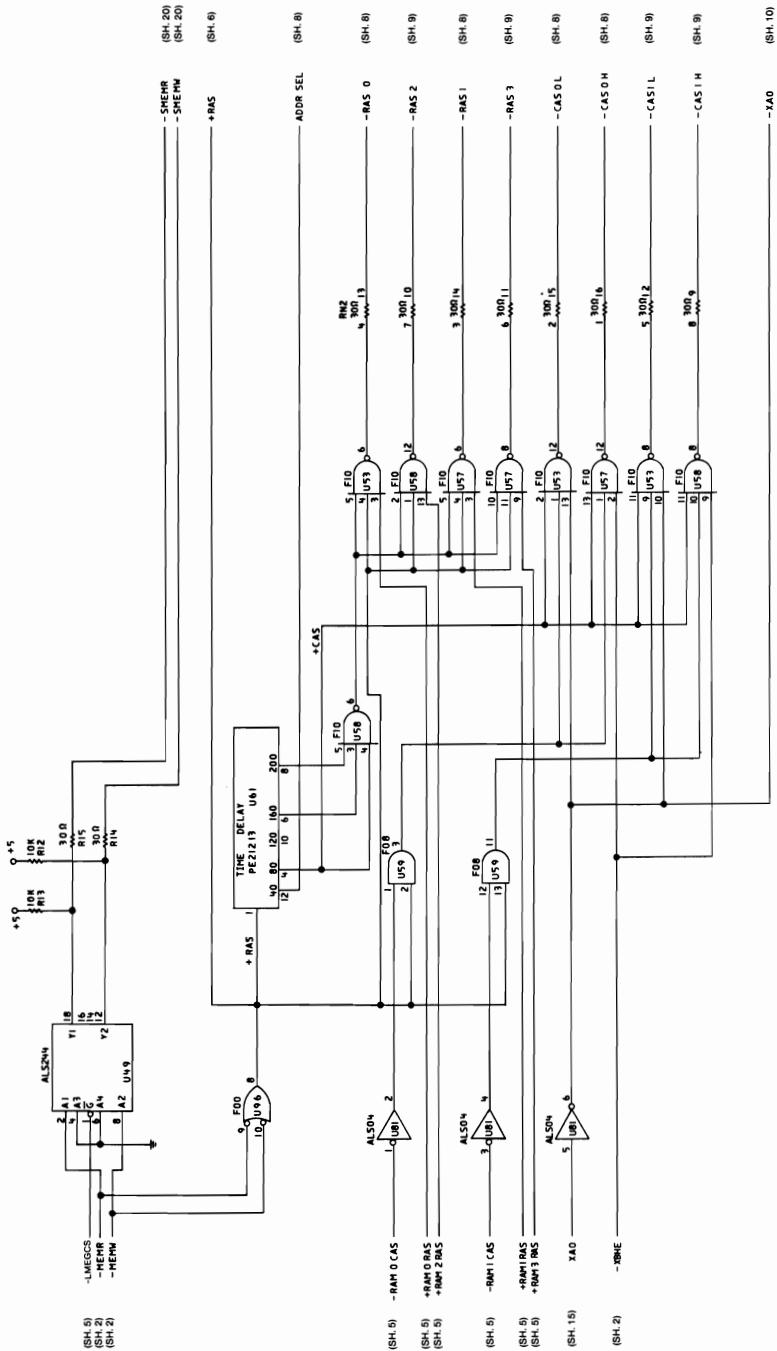
Type 1 512KB Planar (Sheet 4 of 22)



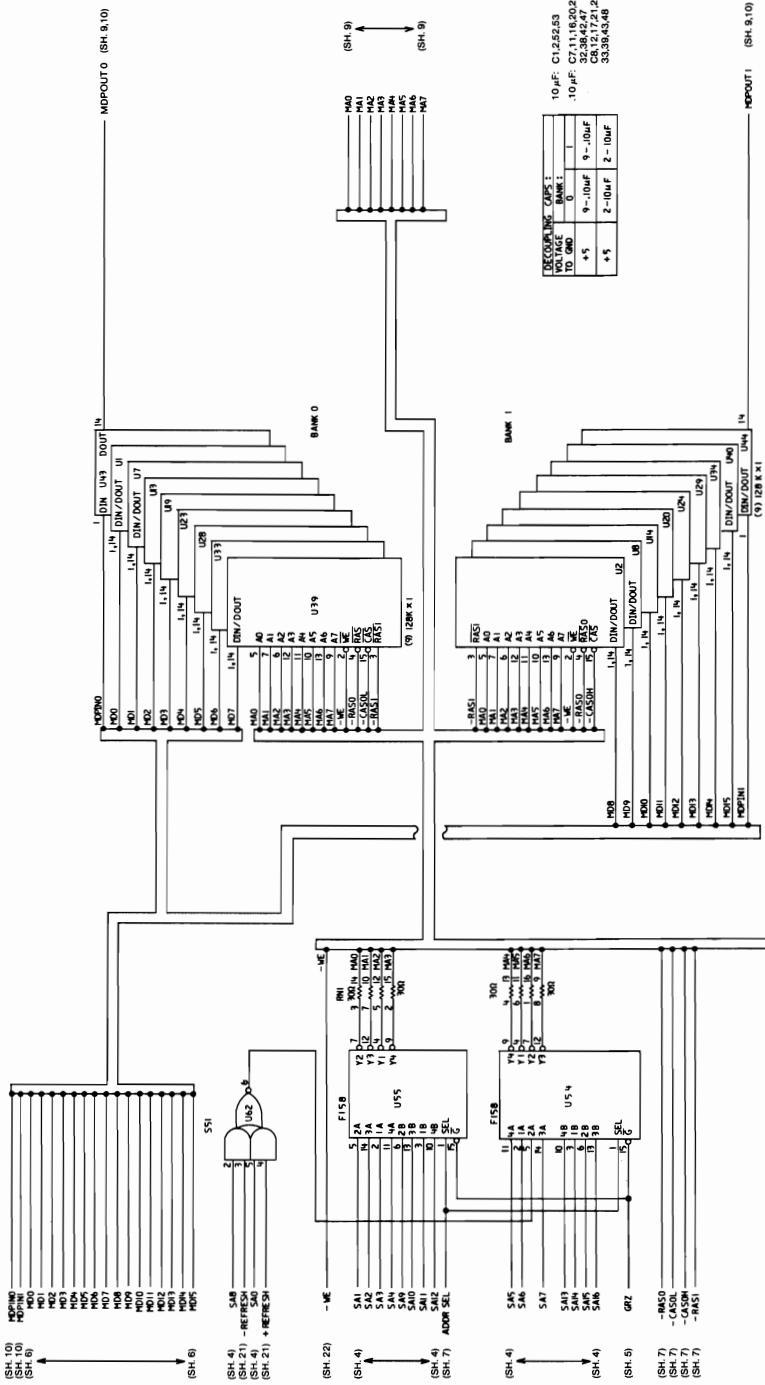


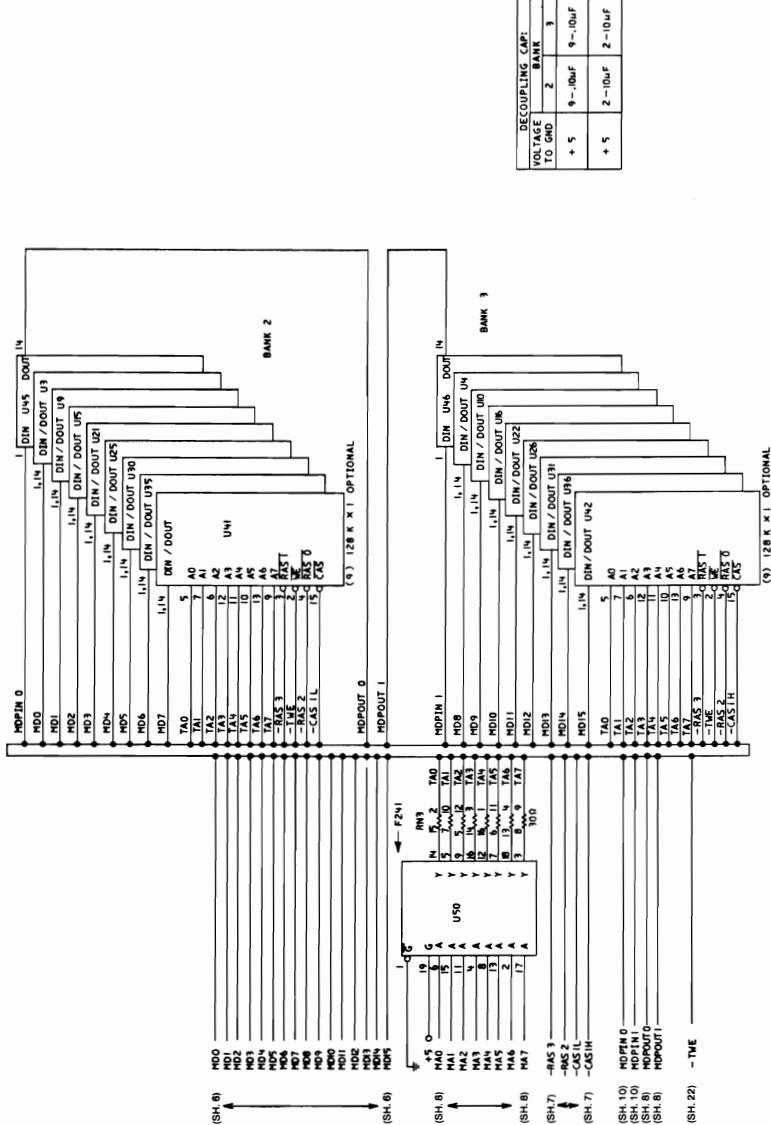
Type 1 512KB Planar (Sheet 5 of 22)





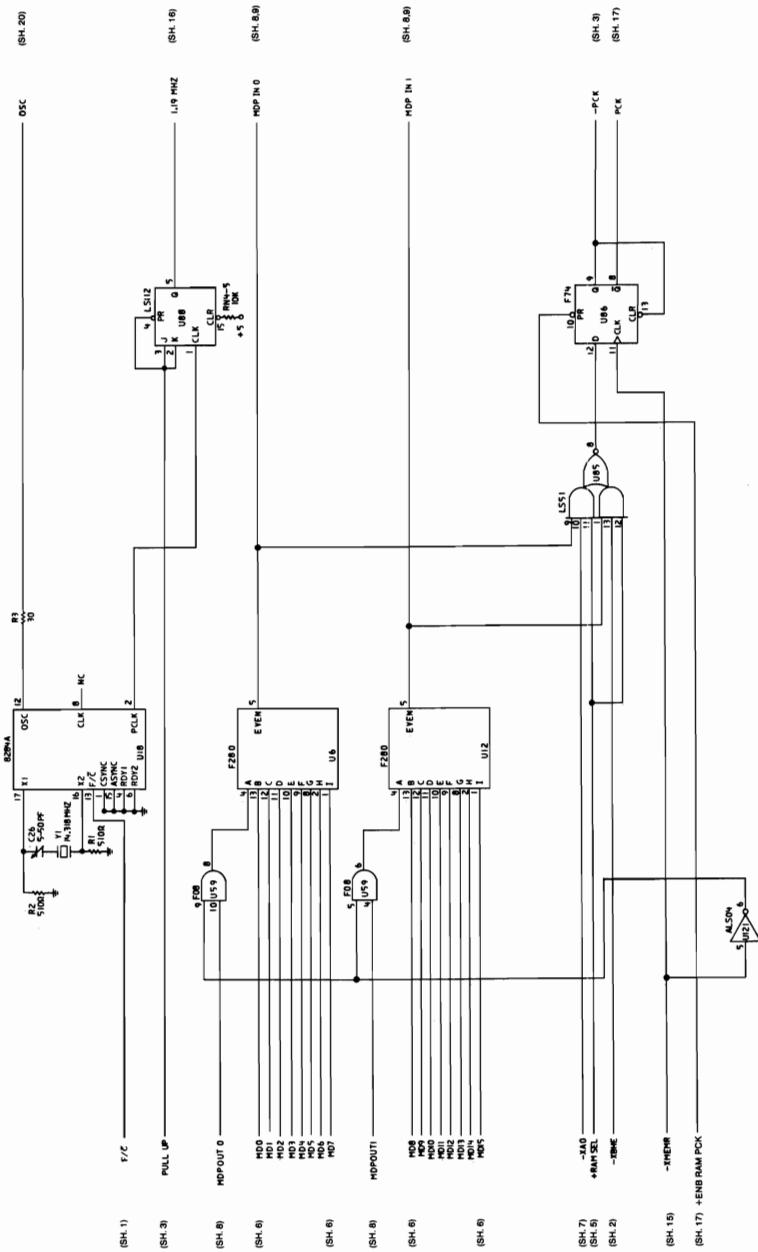
## Type 1 512KB Planar (Sheet 8 of 22)



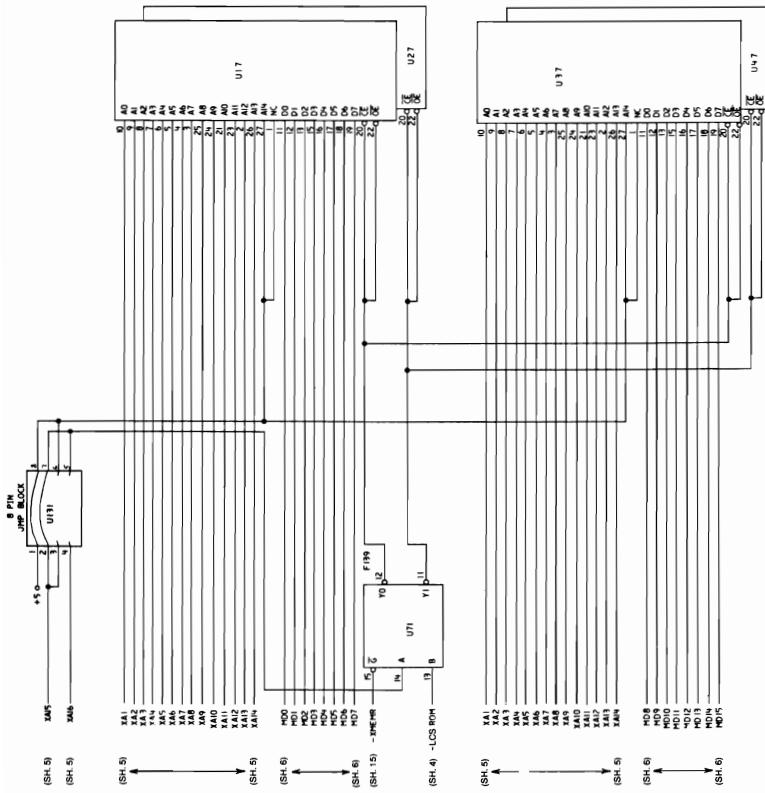


Type 1 512KB Planar (Sheet 9 of 22)

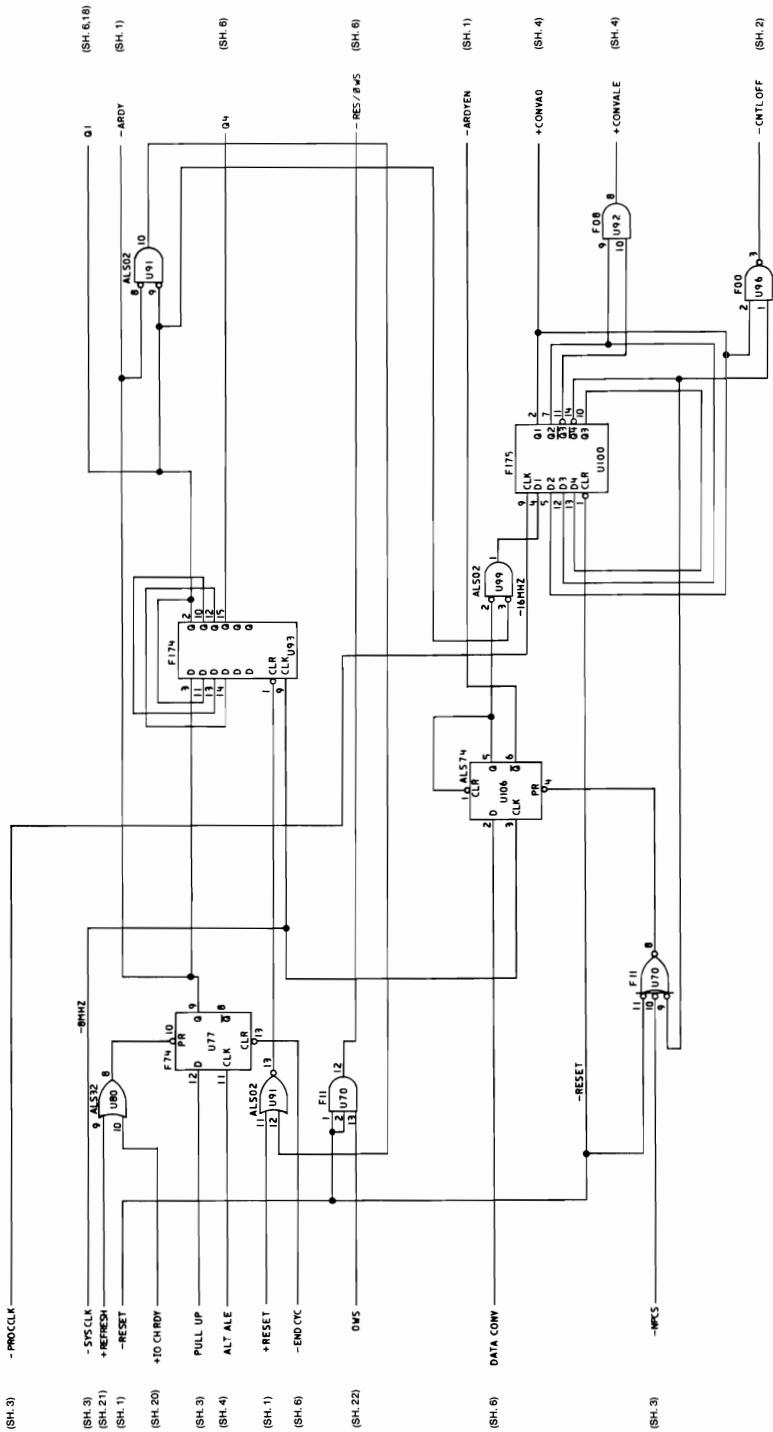
Type 1 512KB Planar (Sheet 10 of 22)

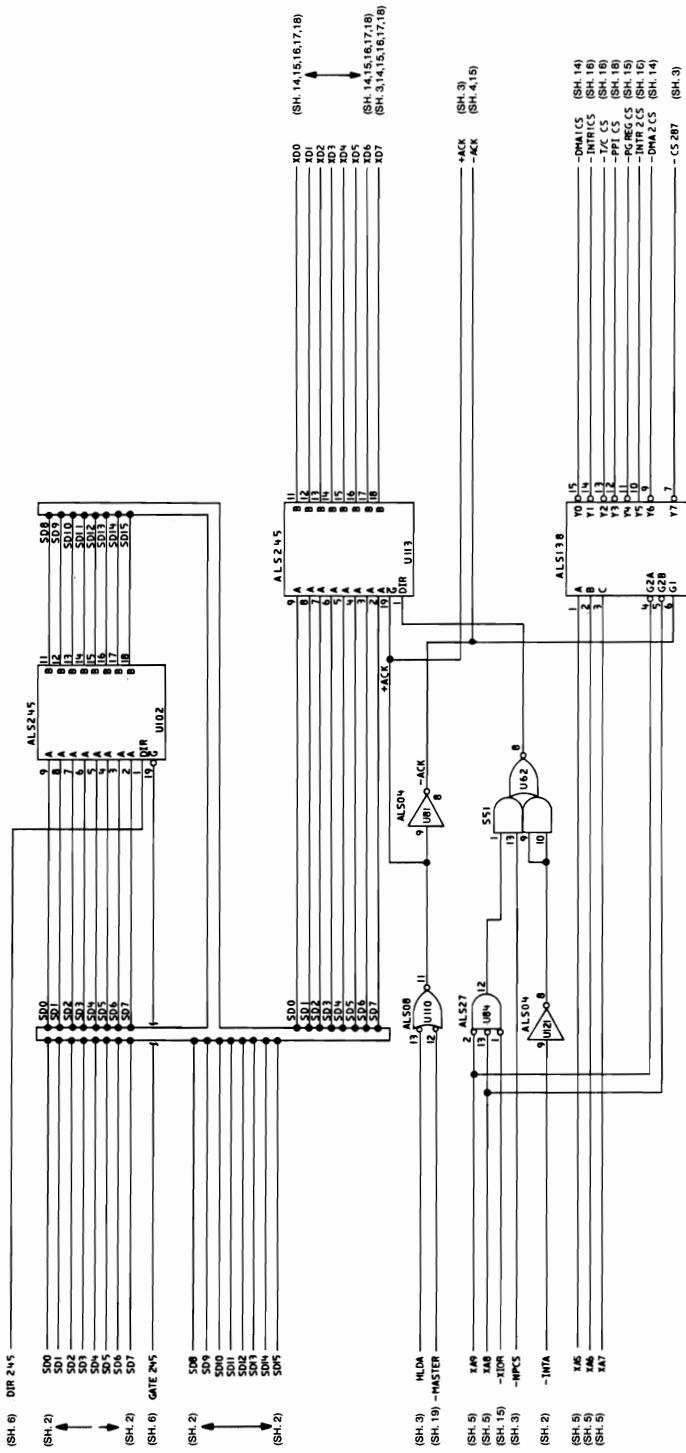


Type 1 512KB Planar (Sheet 11 of 22)

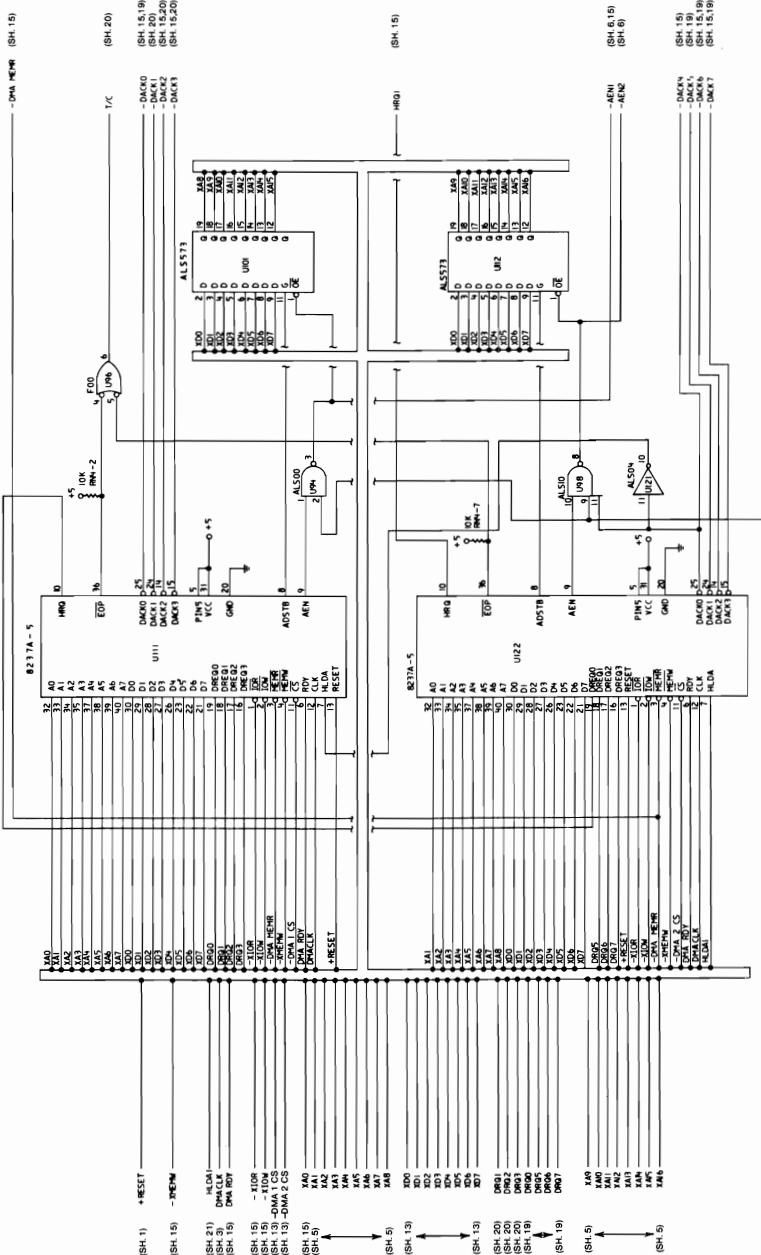


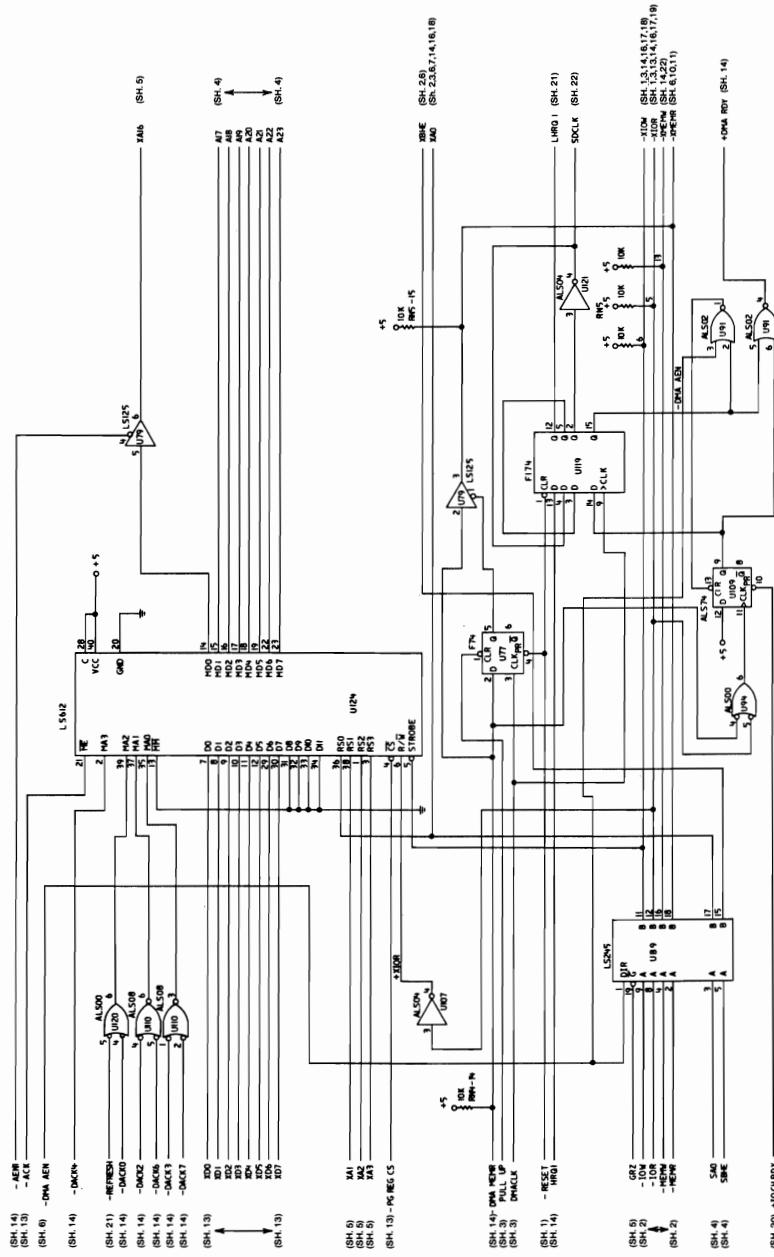
Type 1 512KB Planar (Sheet 12 of 22)





Type 1 512KB Planar (Sheet 14 of 22)

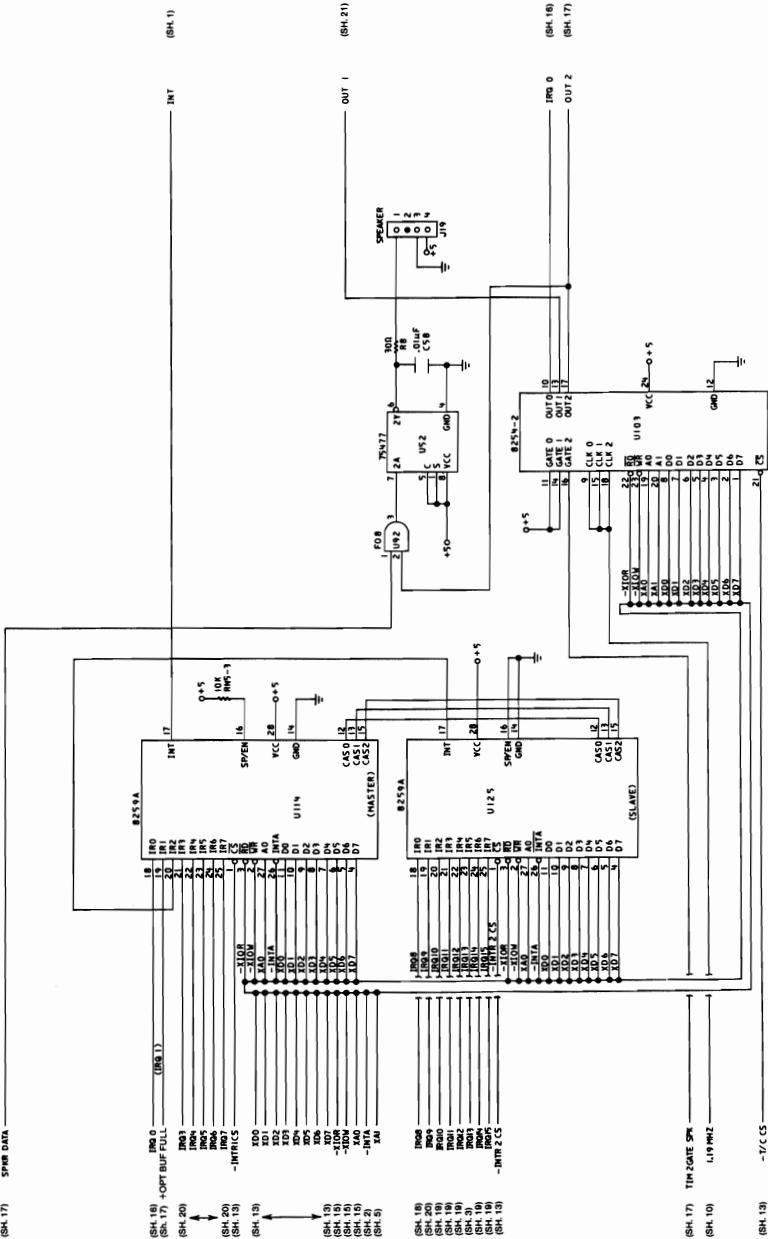


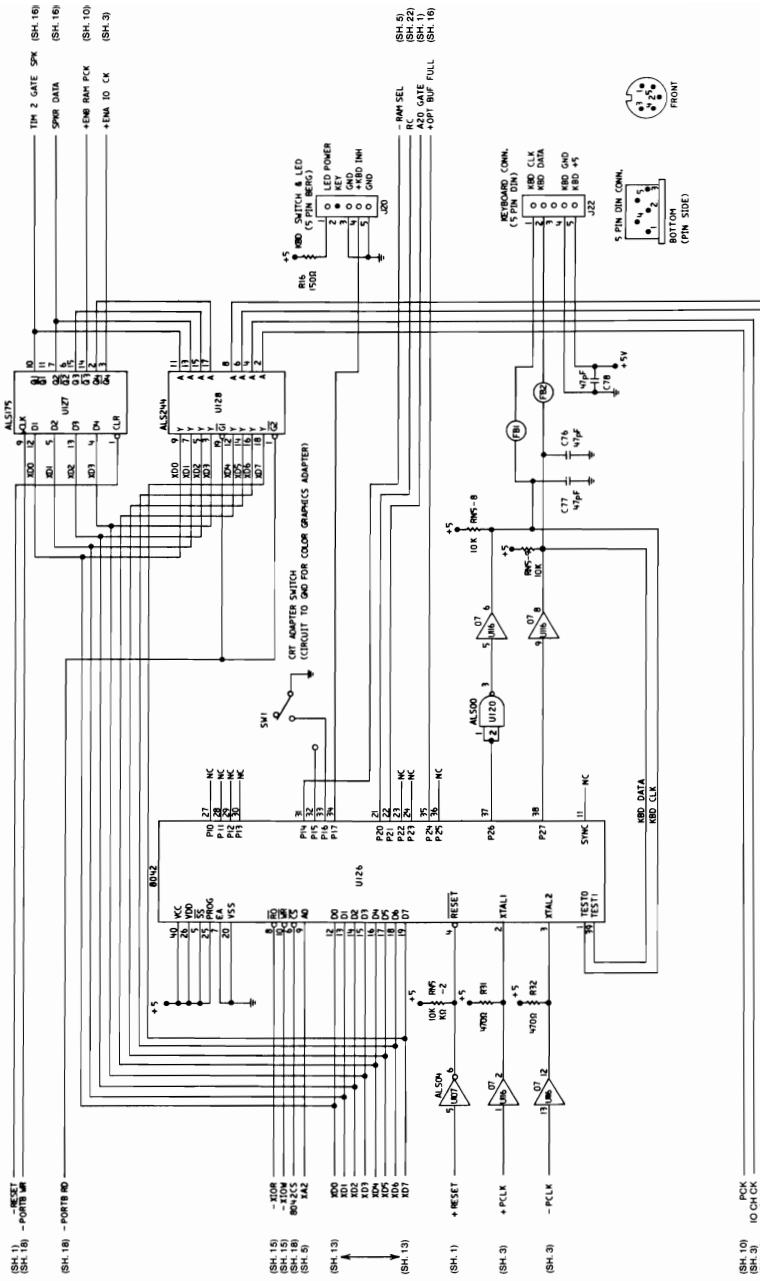


Type 1 51.2KB Planar (Sheet 15 of 22)

Type 1 512KB Planar (Sheet 16 of 22)

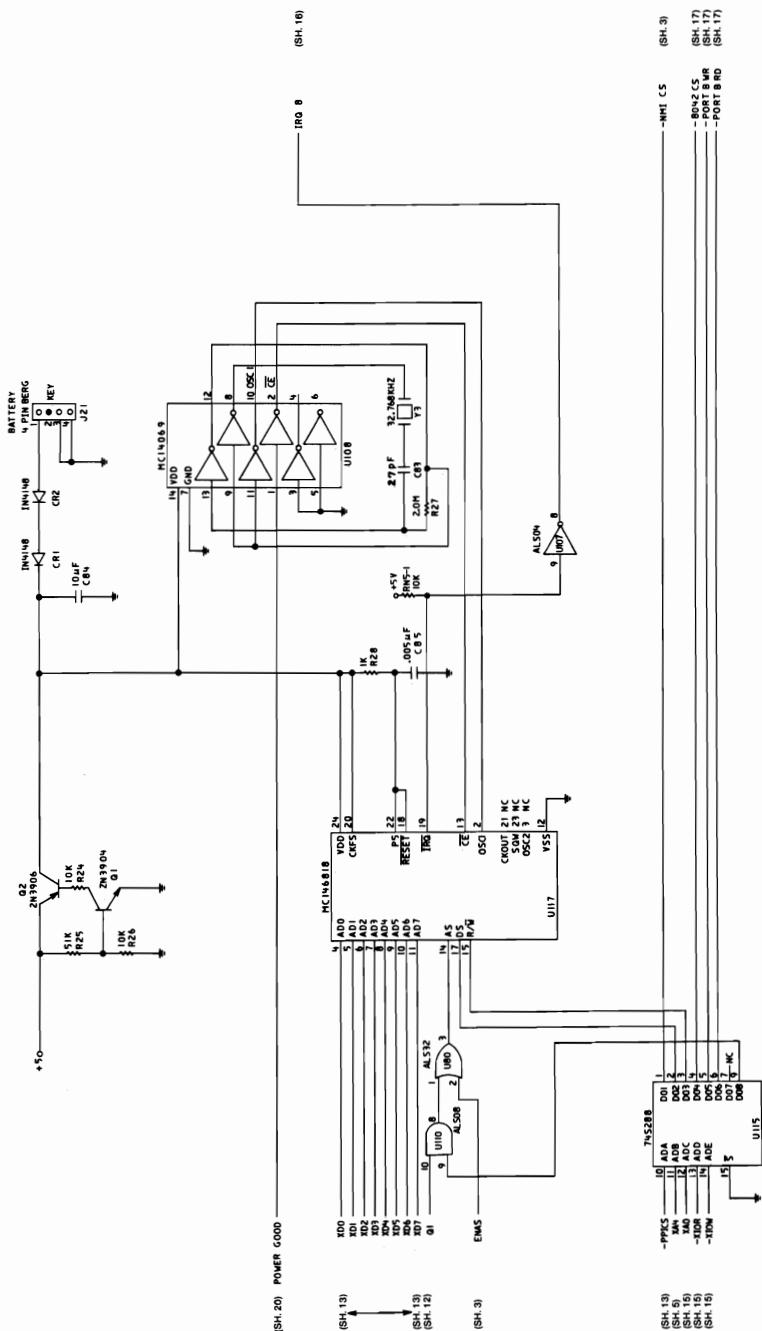
SH. 17) SPK& DATA

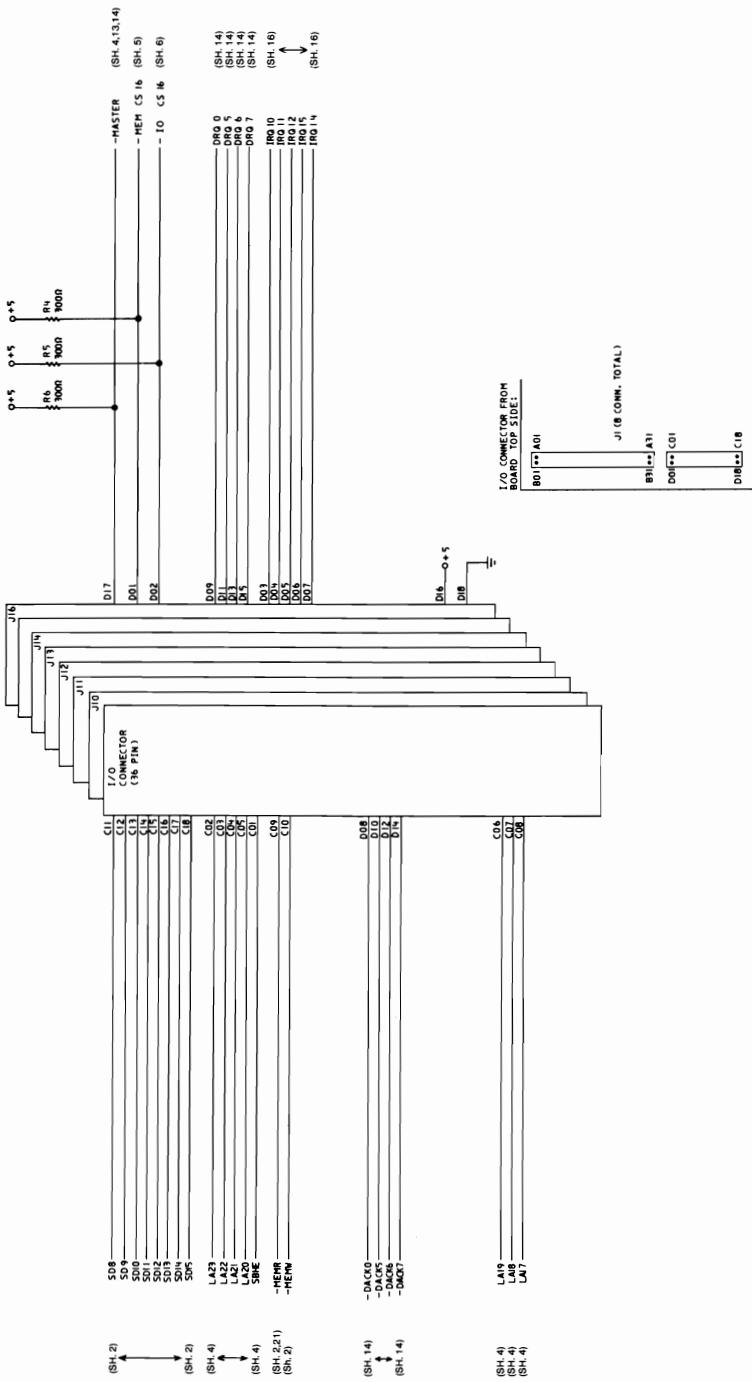




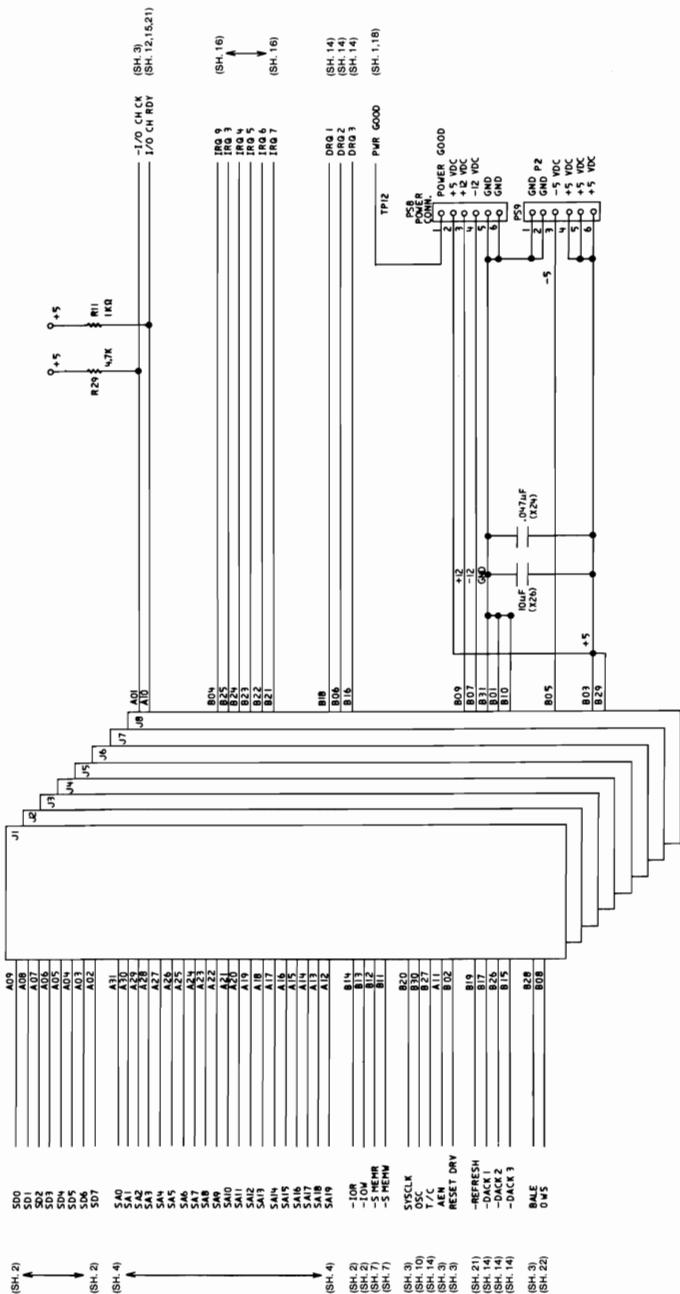
Type 1 512KB Planar (Sheet 17 of 22)

Type 1 512KB Planar (Sheet 18 of 22)

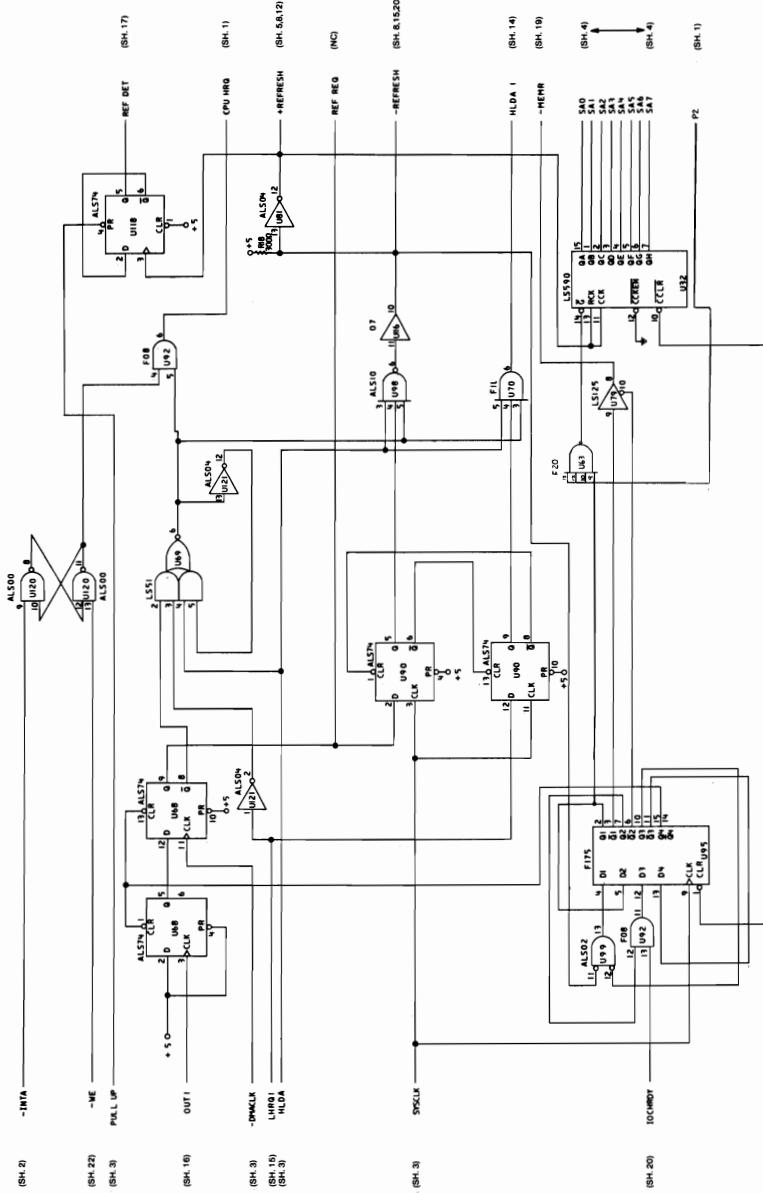




Type 1 512KB Planar (Sheet 20 of 22)

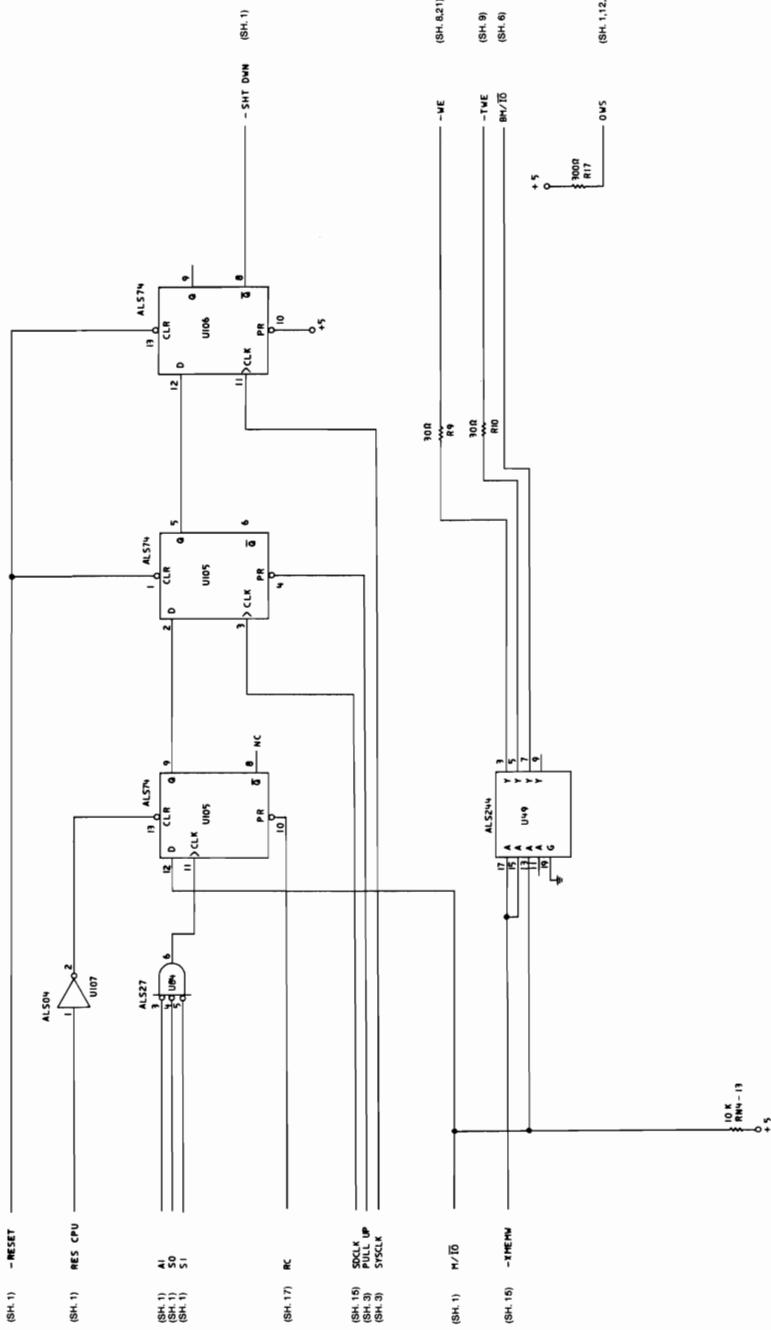


## 1-96 System Board

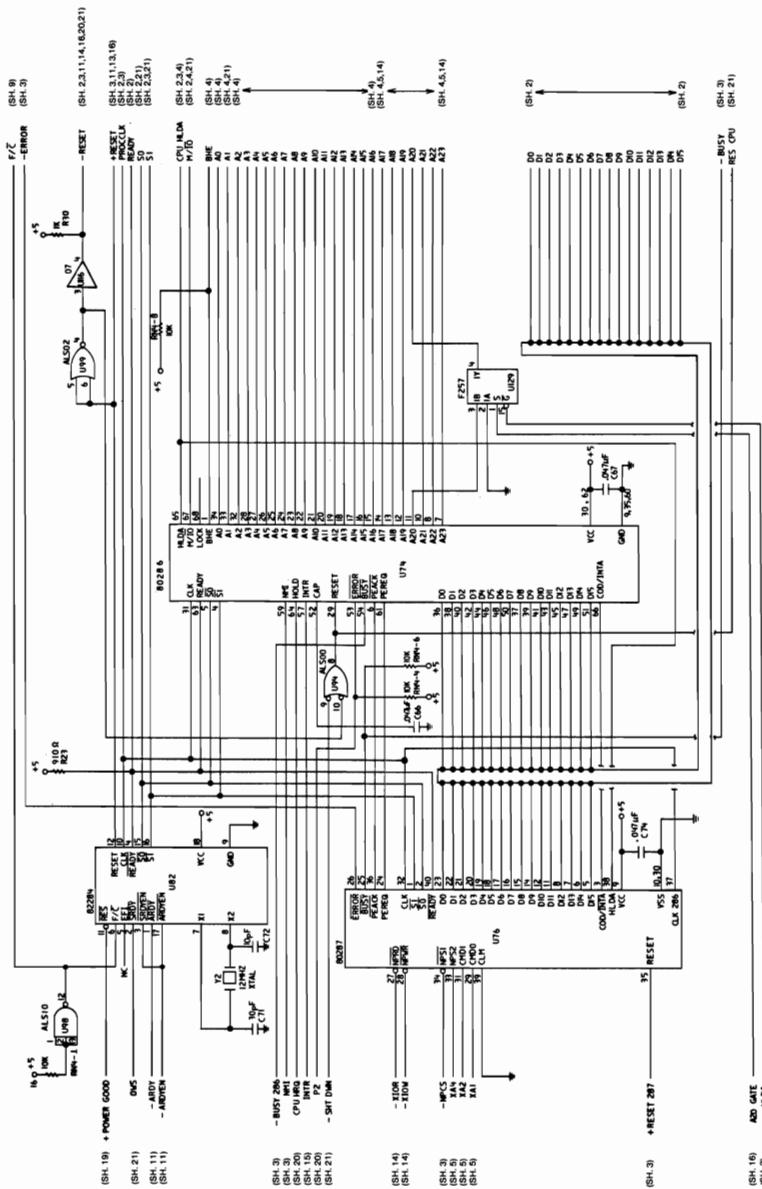


Type 1 512KB Planar (Sheet 21 of 22)

## Type 1 512KB Planar (Sheet 22 of 22)

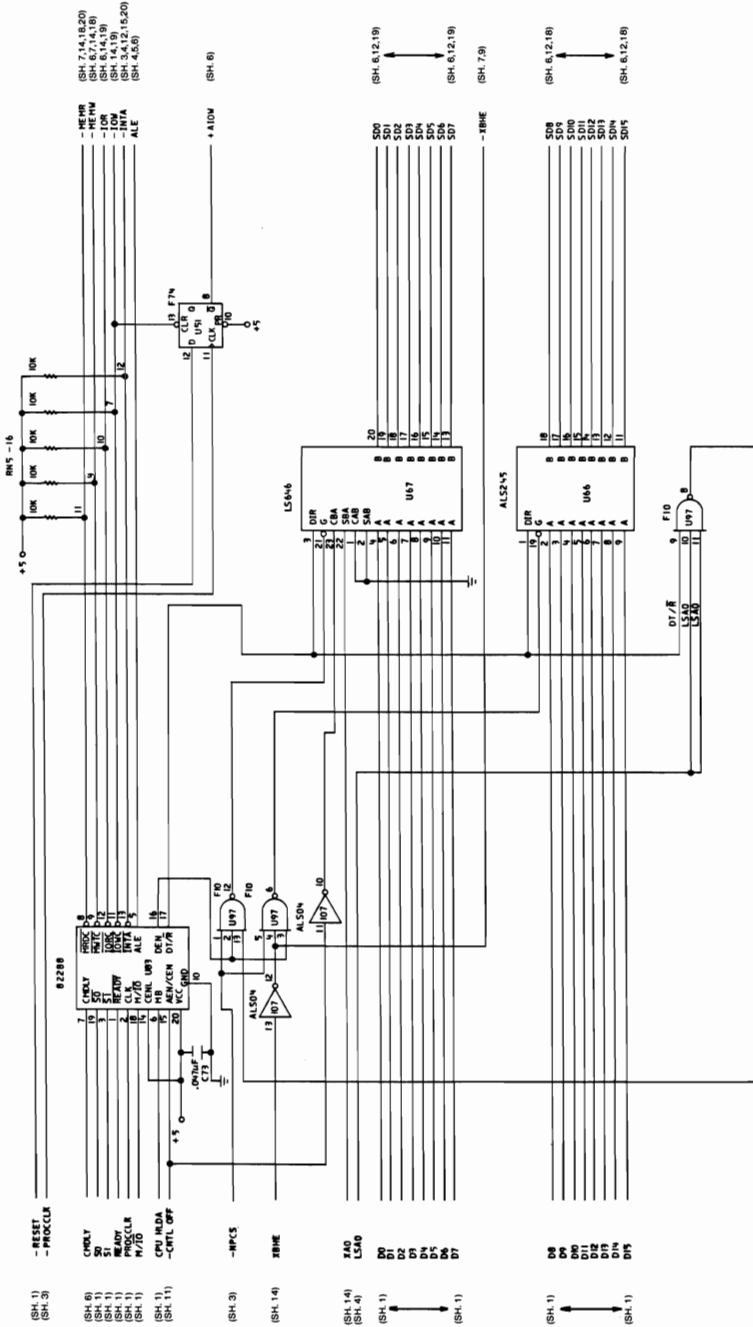


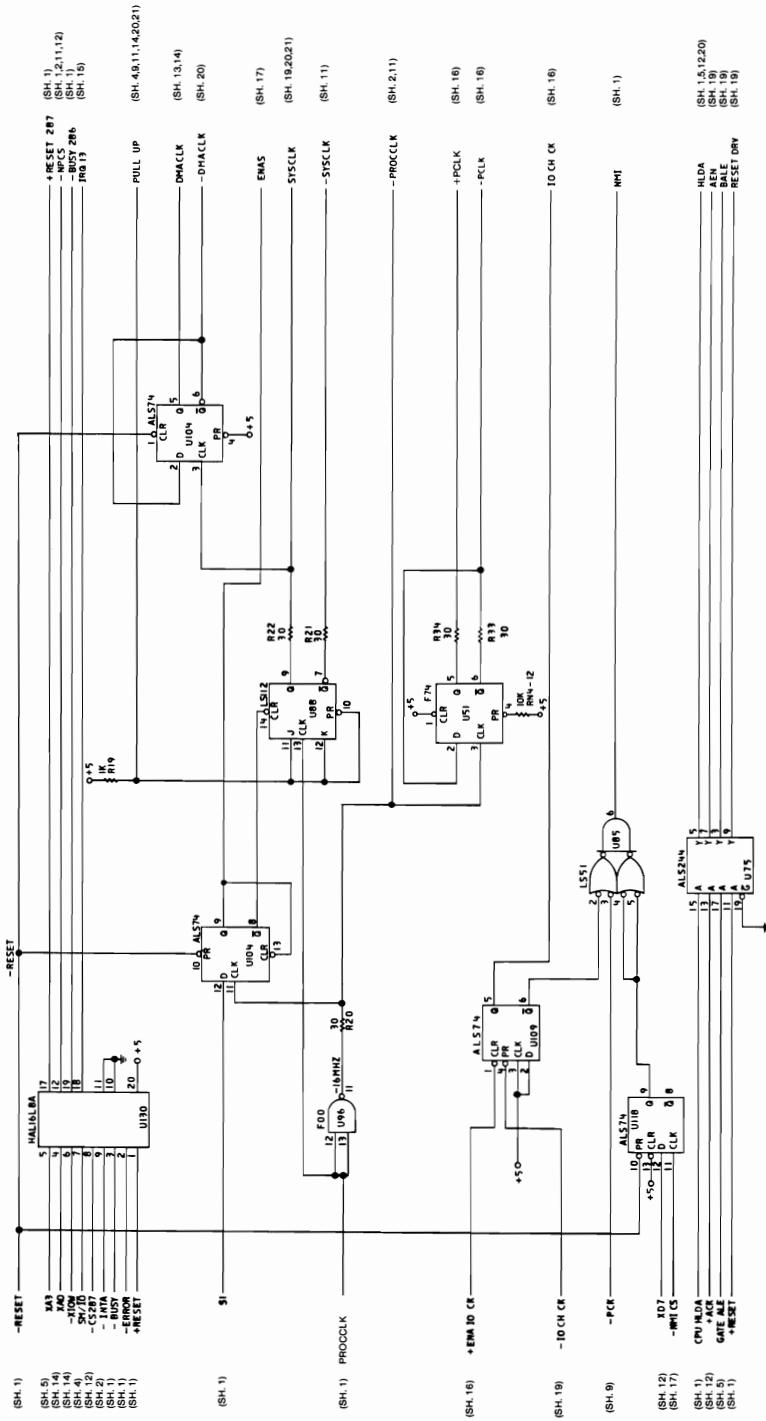
## Logic Diagrams - Type 2



Type 2 512KB Planar (Sheet 1 of 21)

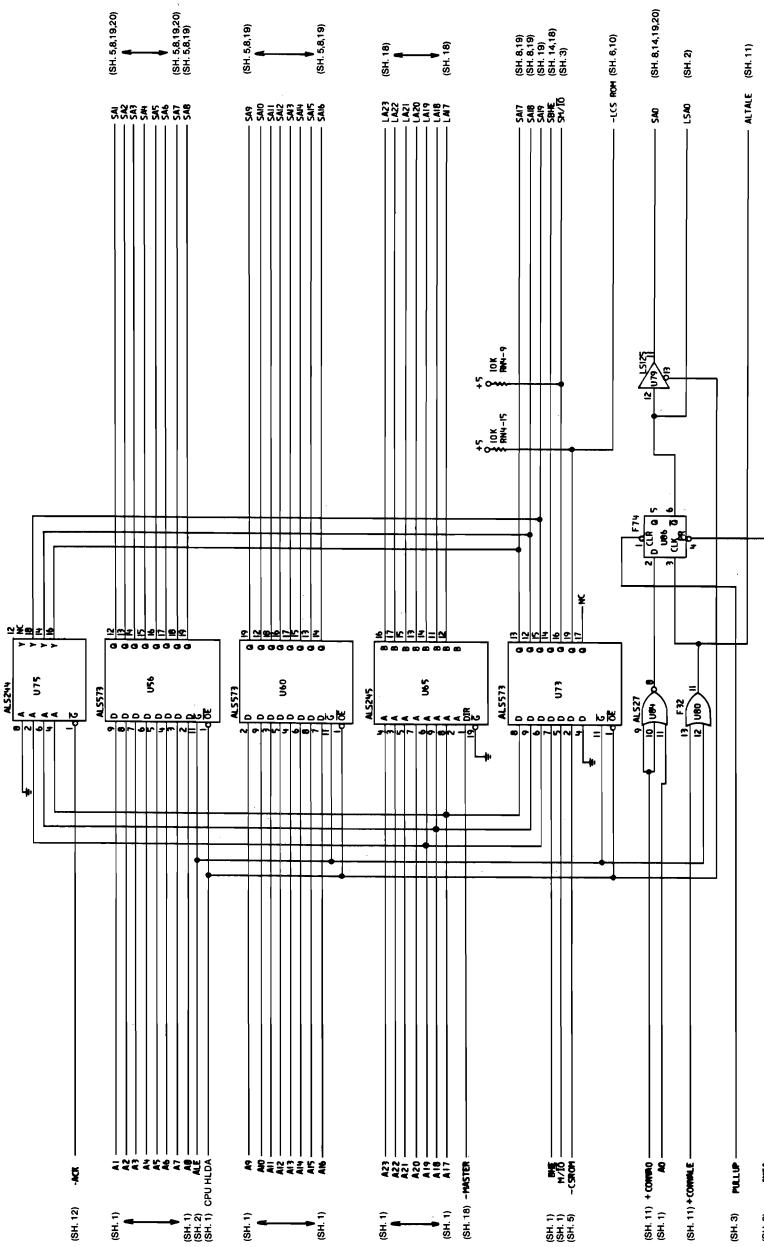
Type 2 512KB Planar (Sheet 2 of 21)

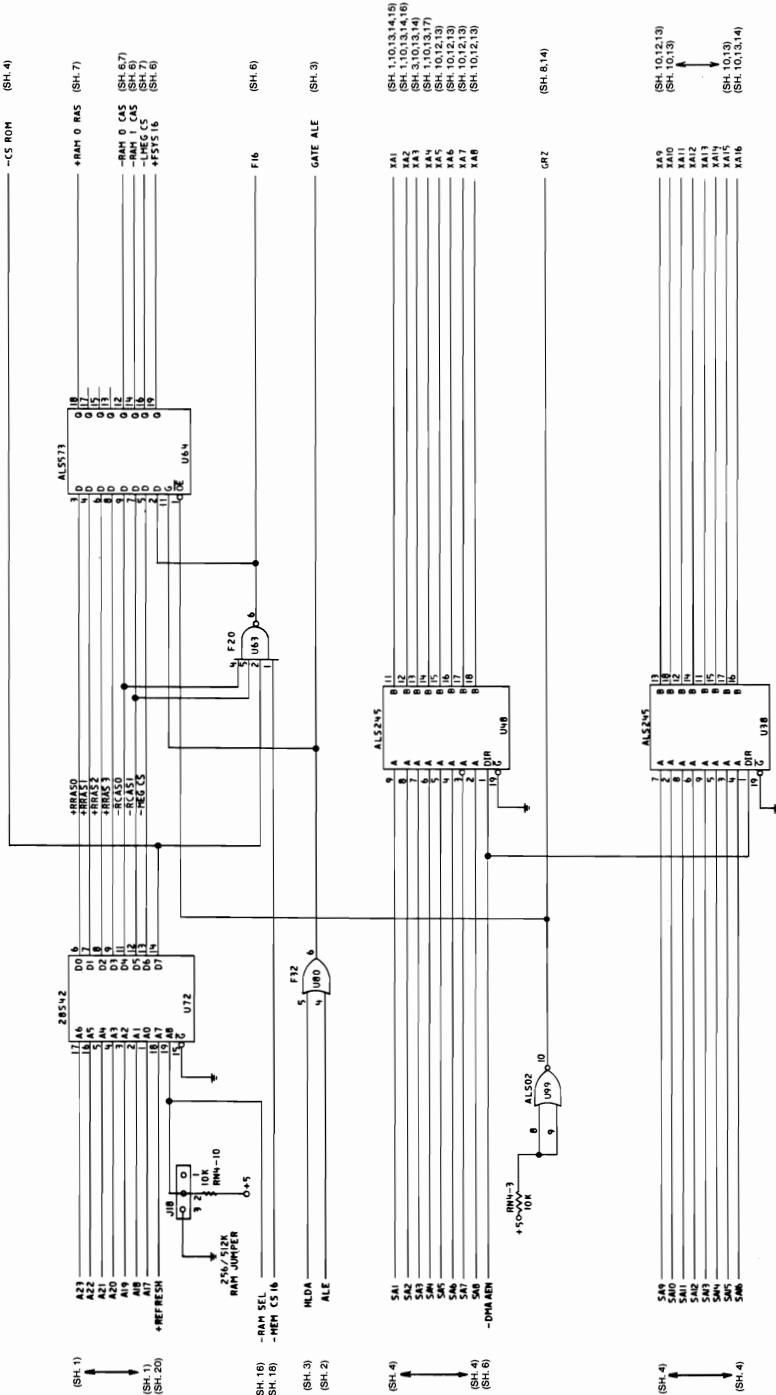


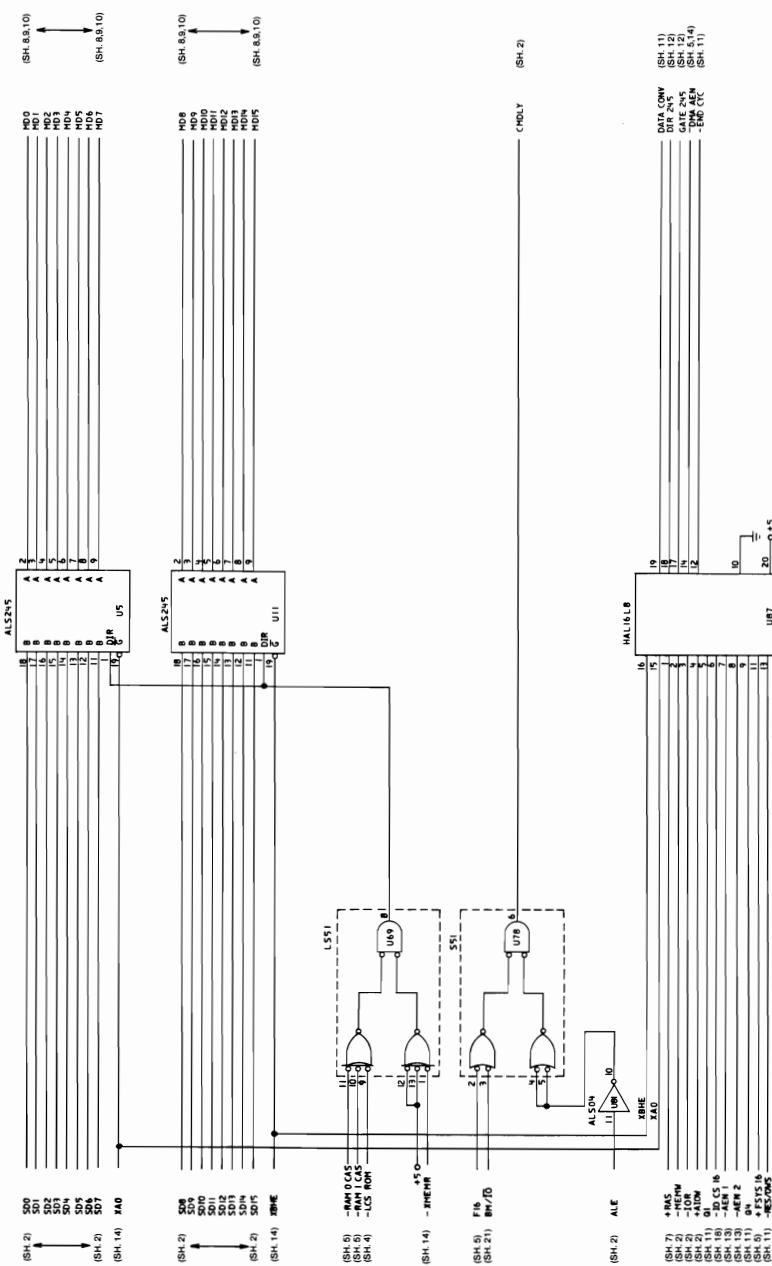


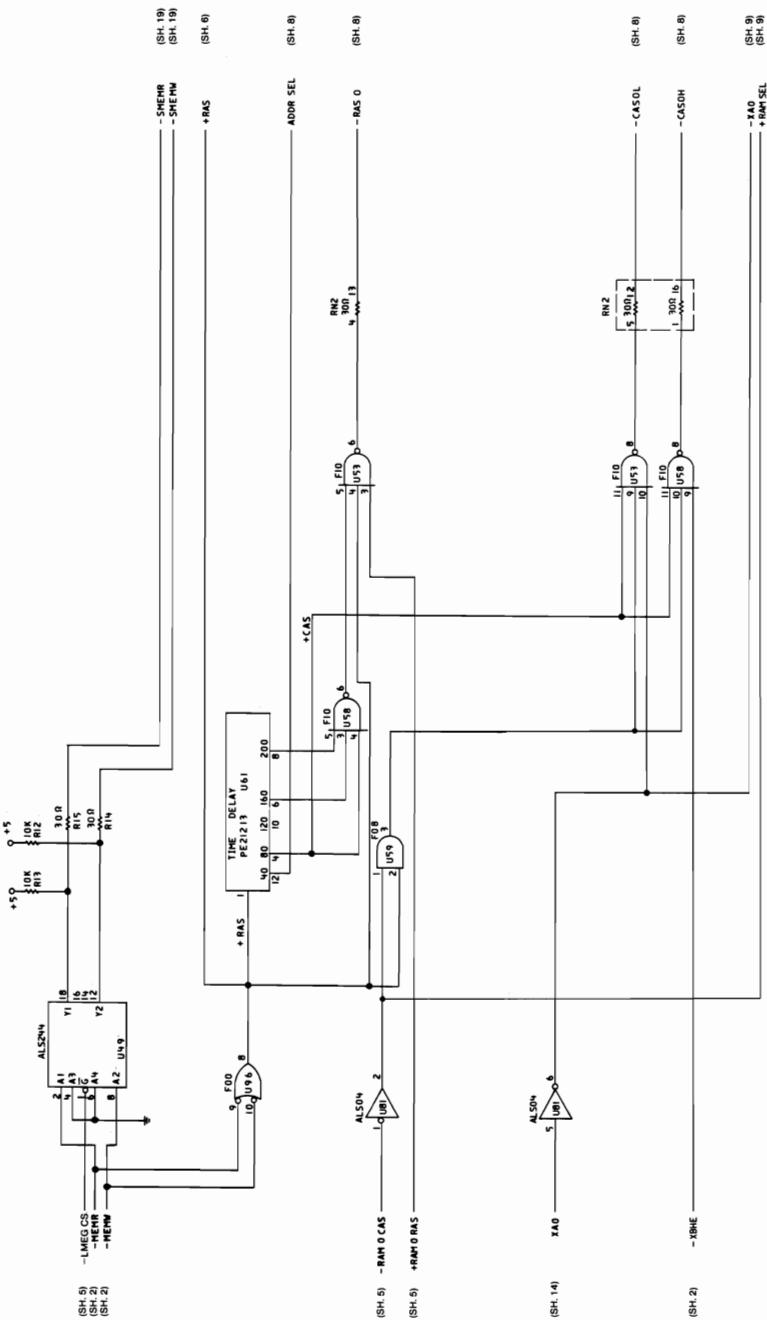
Type 2 512KB Planar (Sheet 3 of 21)

Type 2 512KB Planar (Sheet 4 of 21)



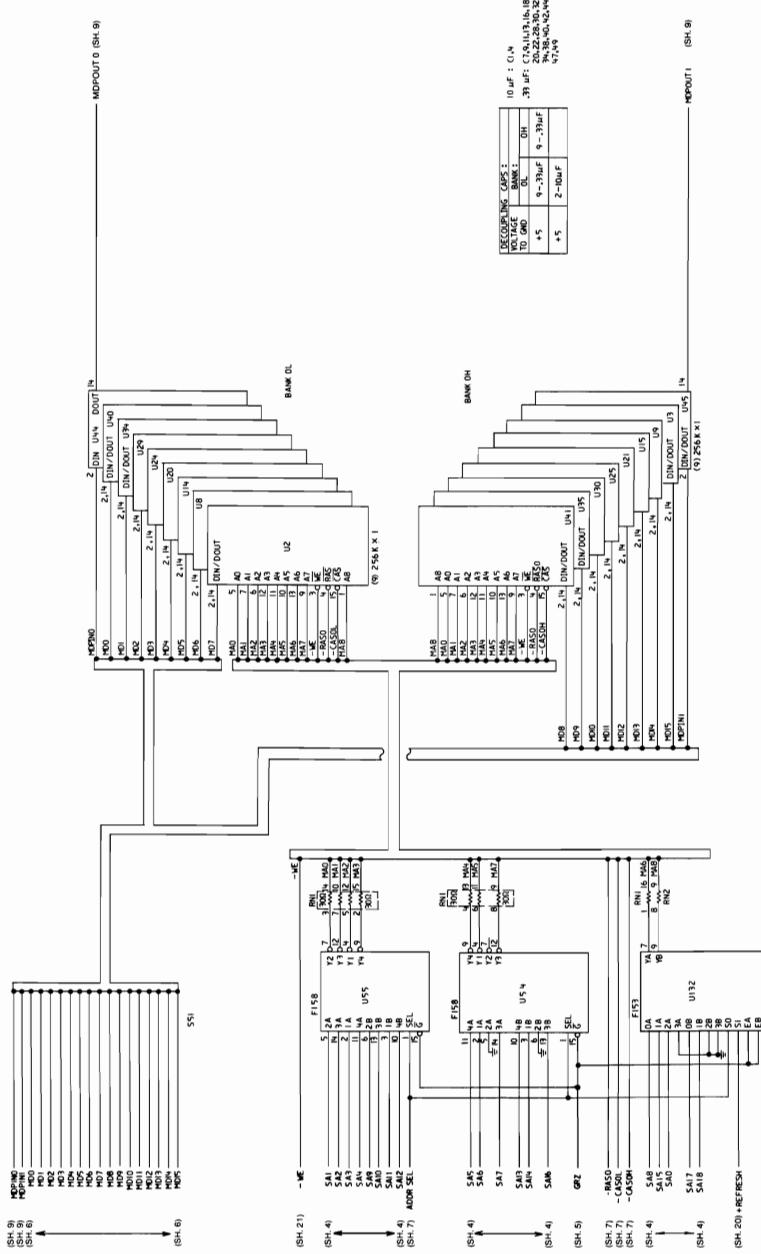


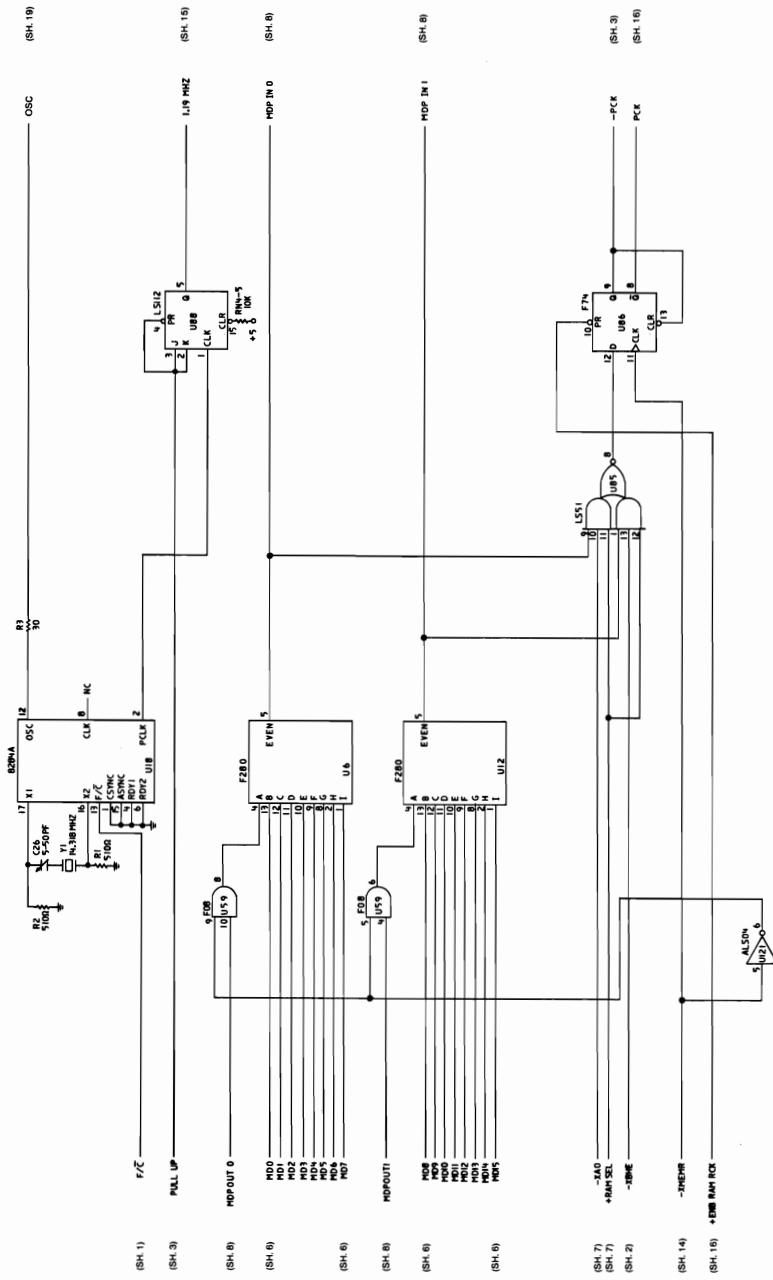




Type 2 512KB Planar (Sheet 7 of 21)

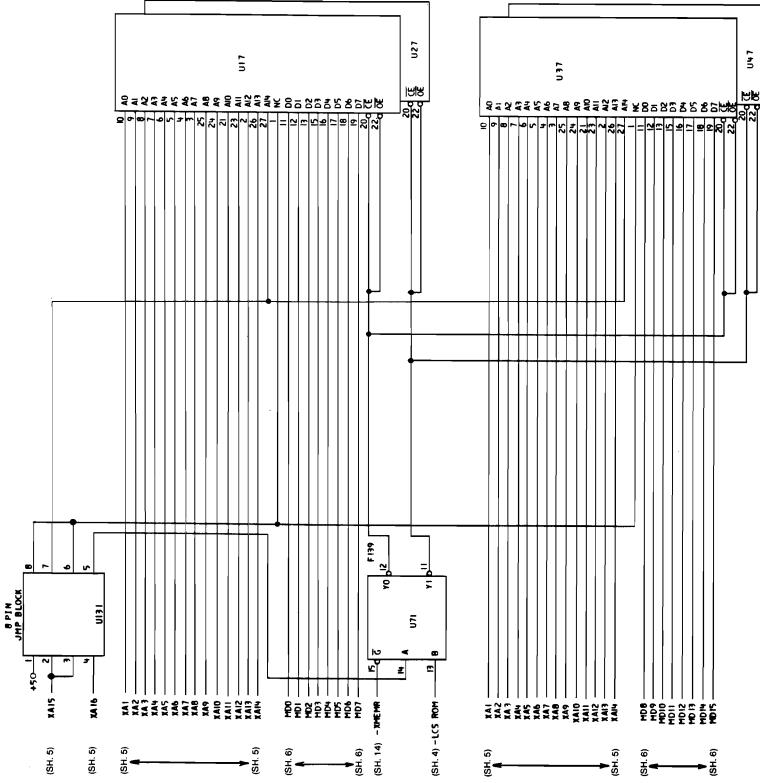
Type 2 512KB Planar (Sheet 8 of 21)

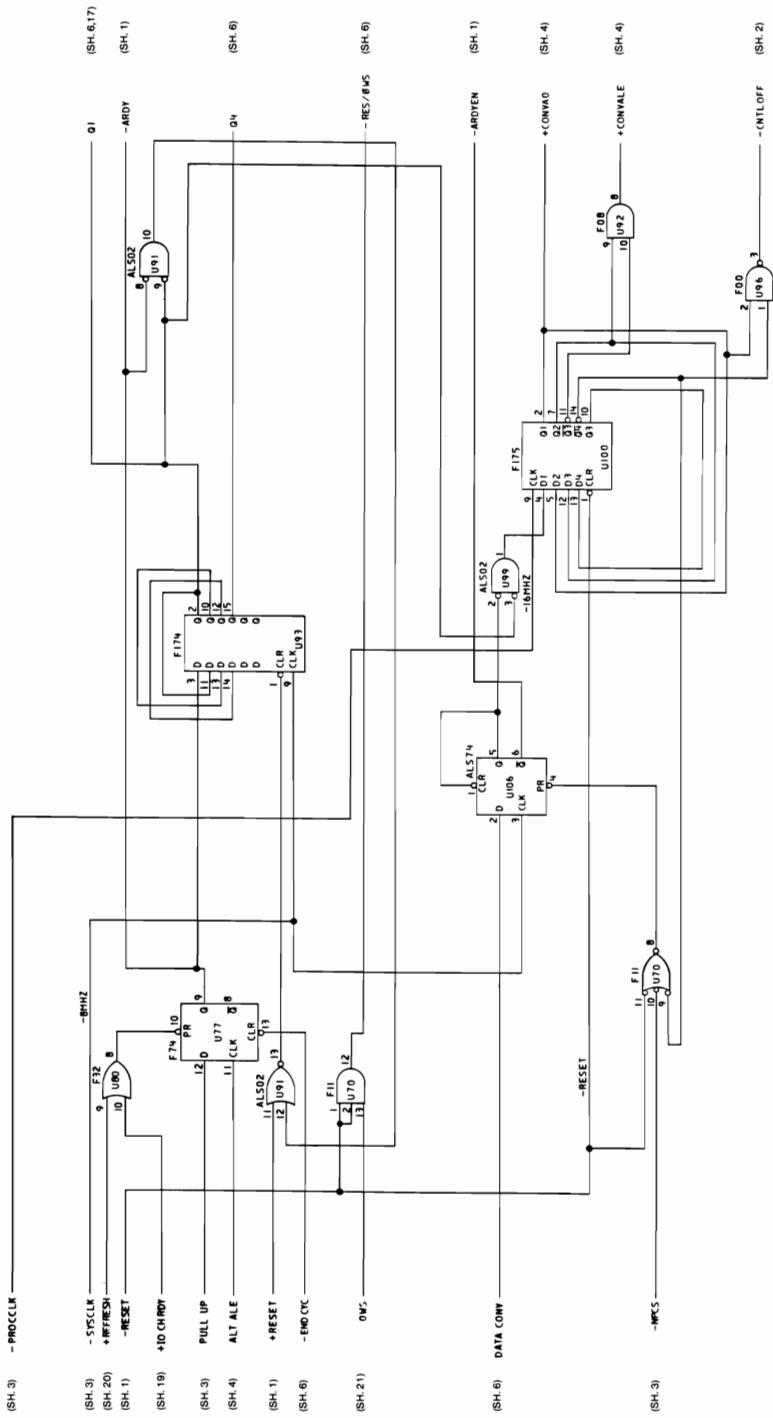




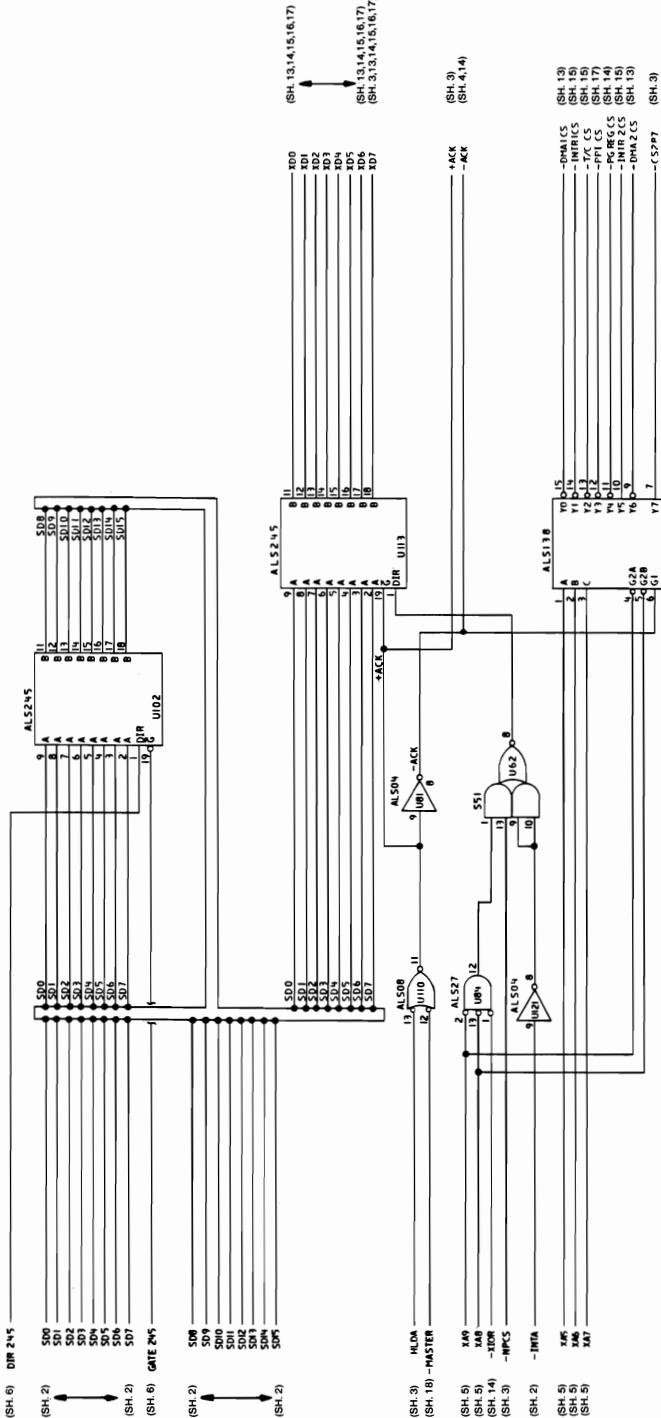
Type 2 512KB Planar (Sheet 9 of 21)

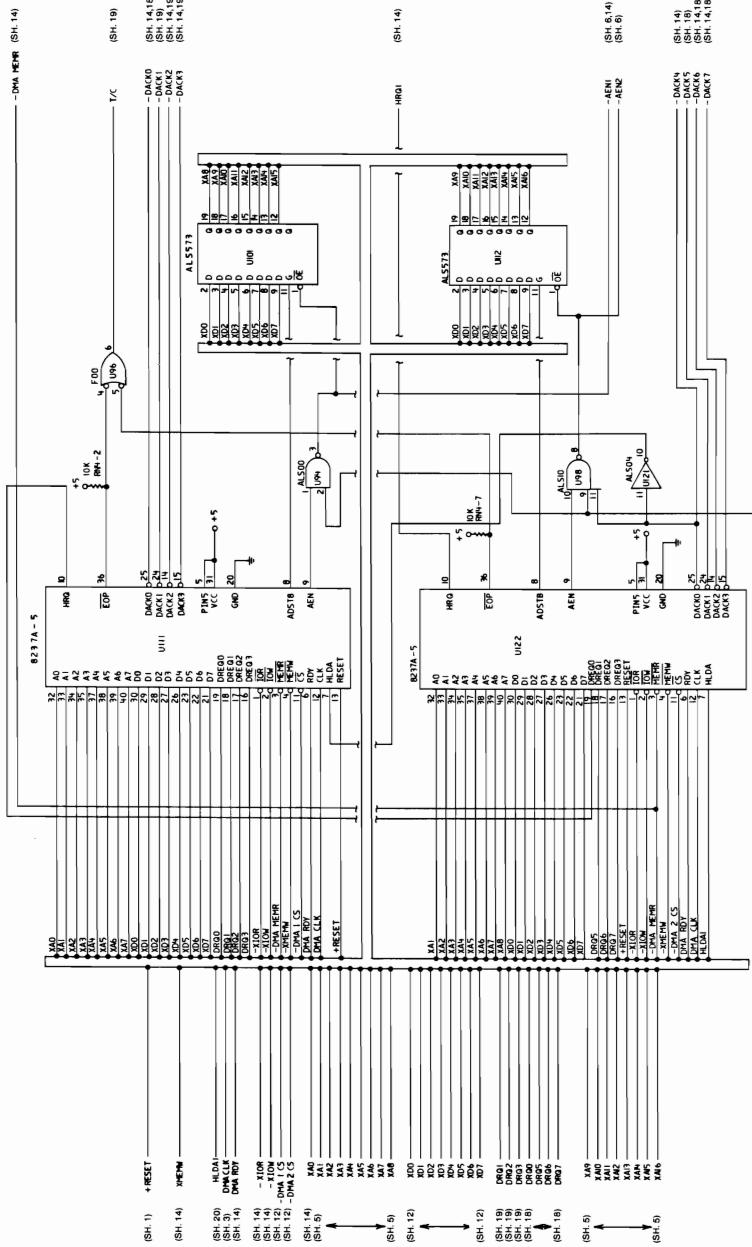
## Type 2 512KB Planar (Sheet 10 of 21)



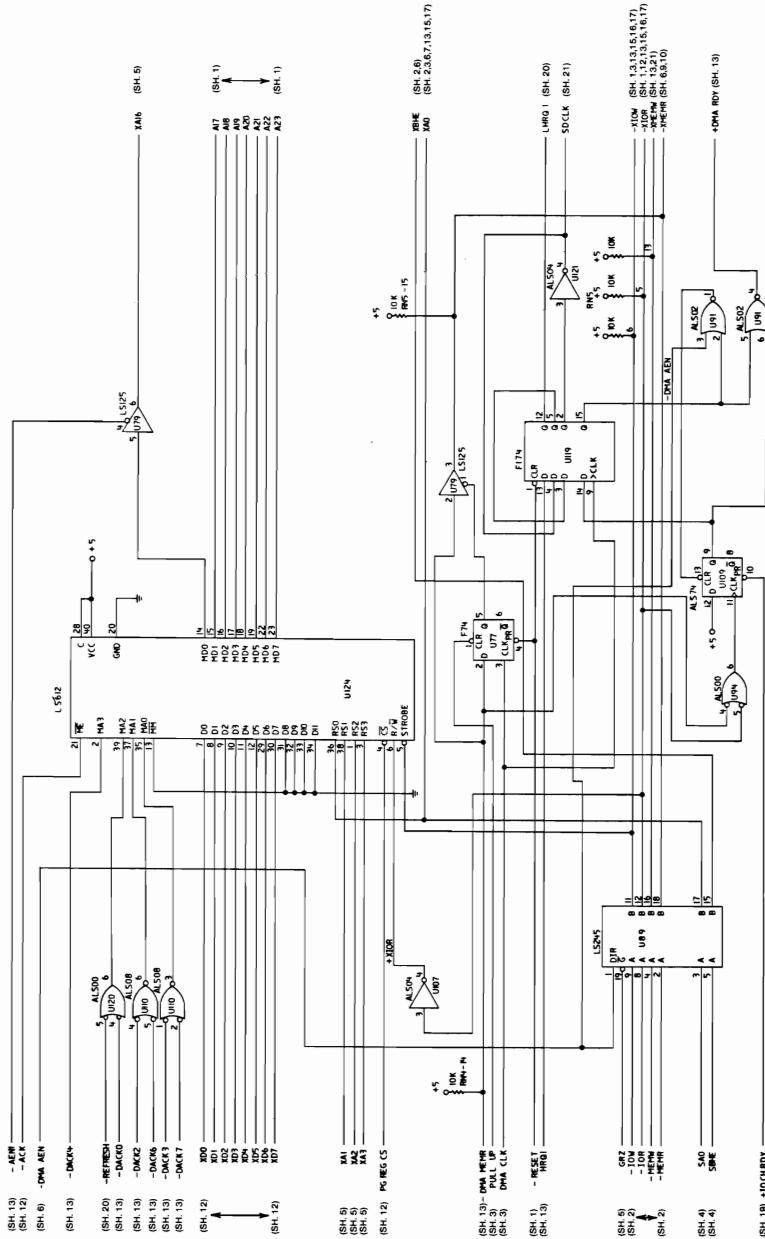


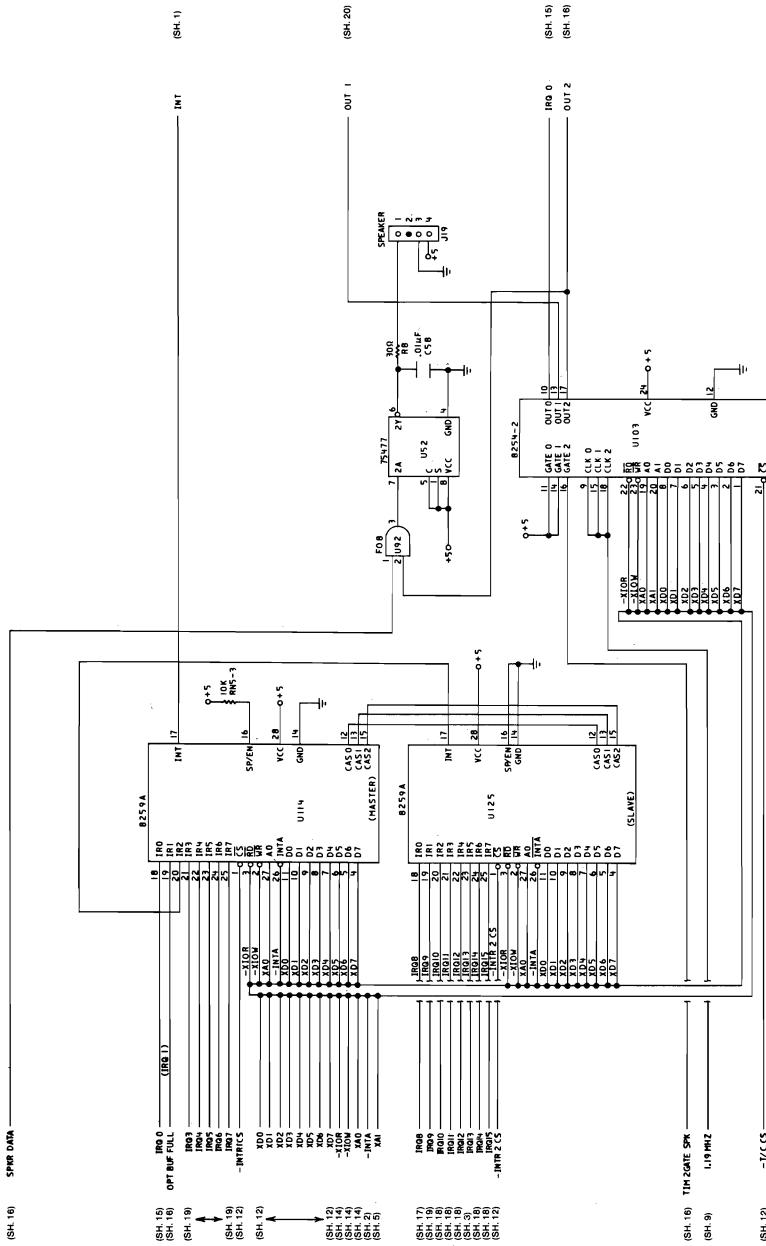
Type 2 512KB Planar (Sheet 11 of 21)





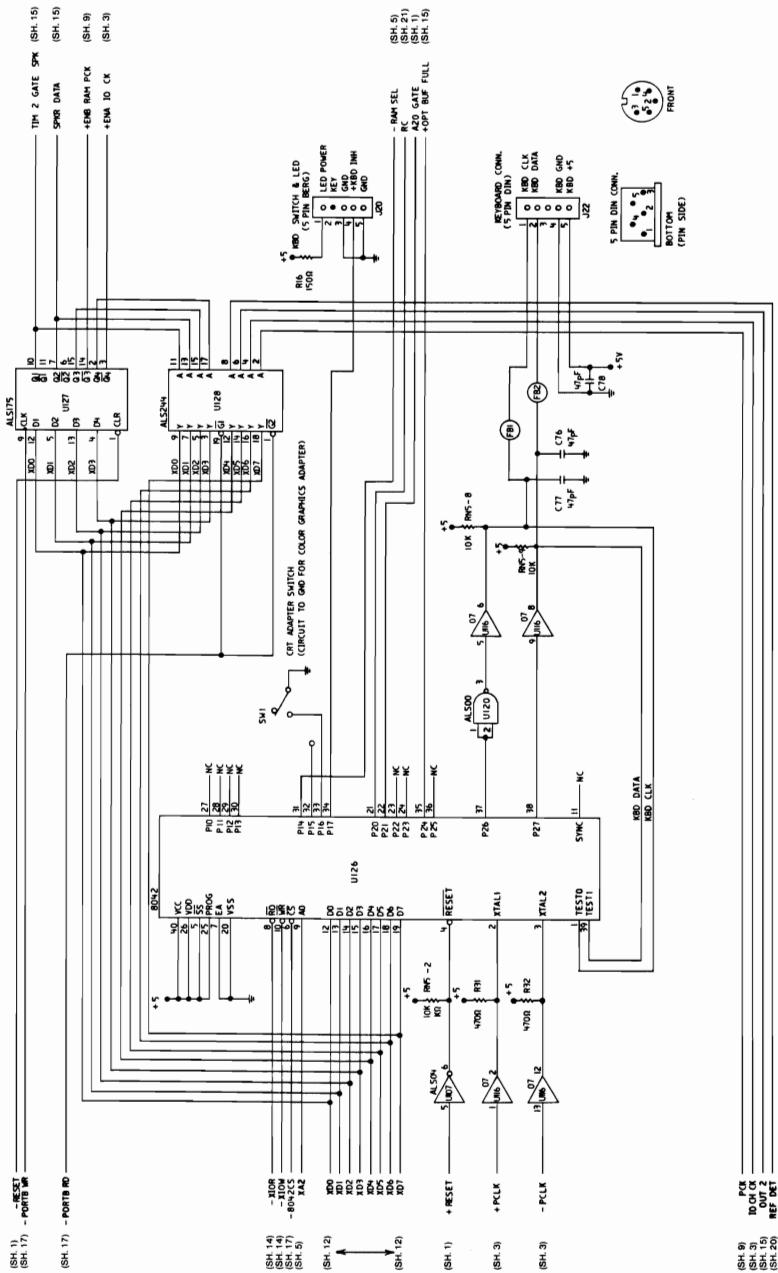
Type 2 512KB Planar (Sheet 14 of 21)

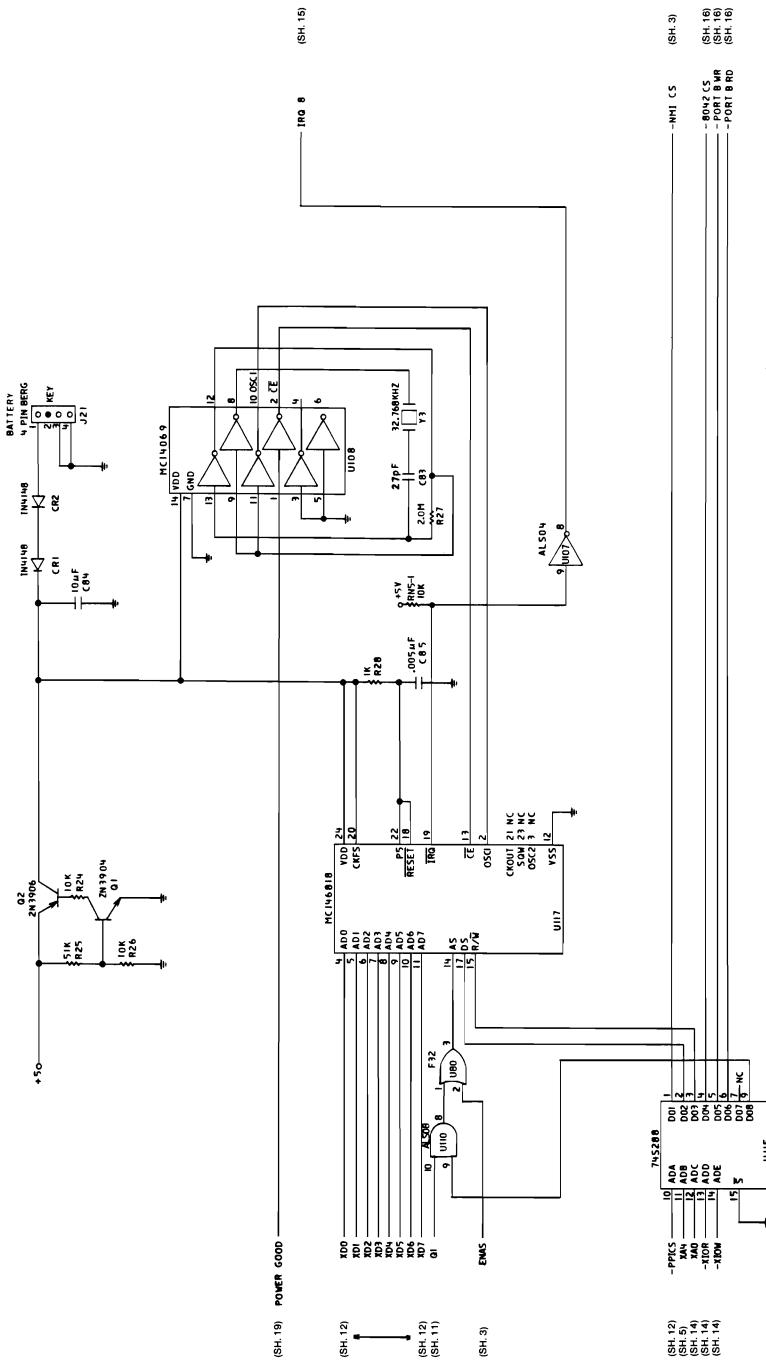




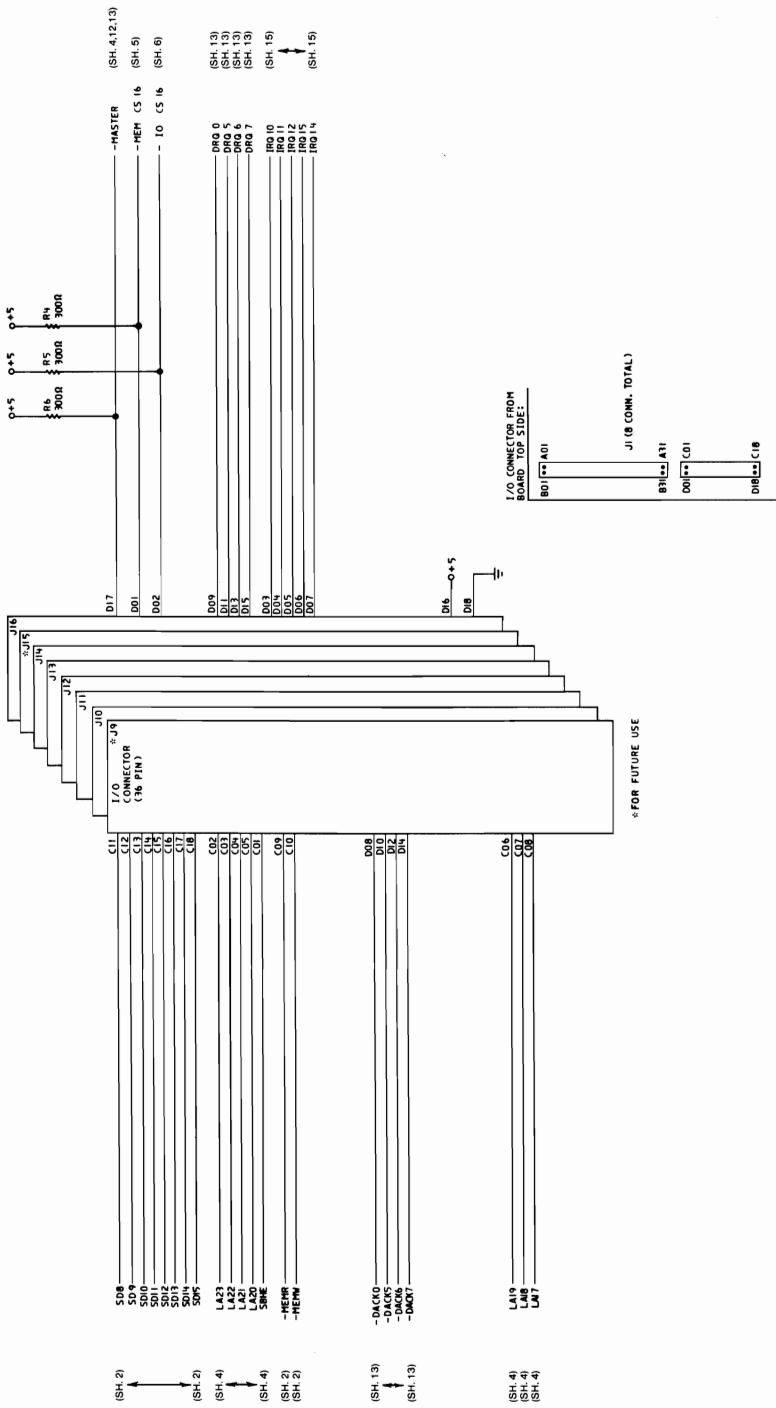
Type 2 512KB Planar (Sheet 15 of 21)

Type 2 512KB Planar (Sheet 16 of 21)

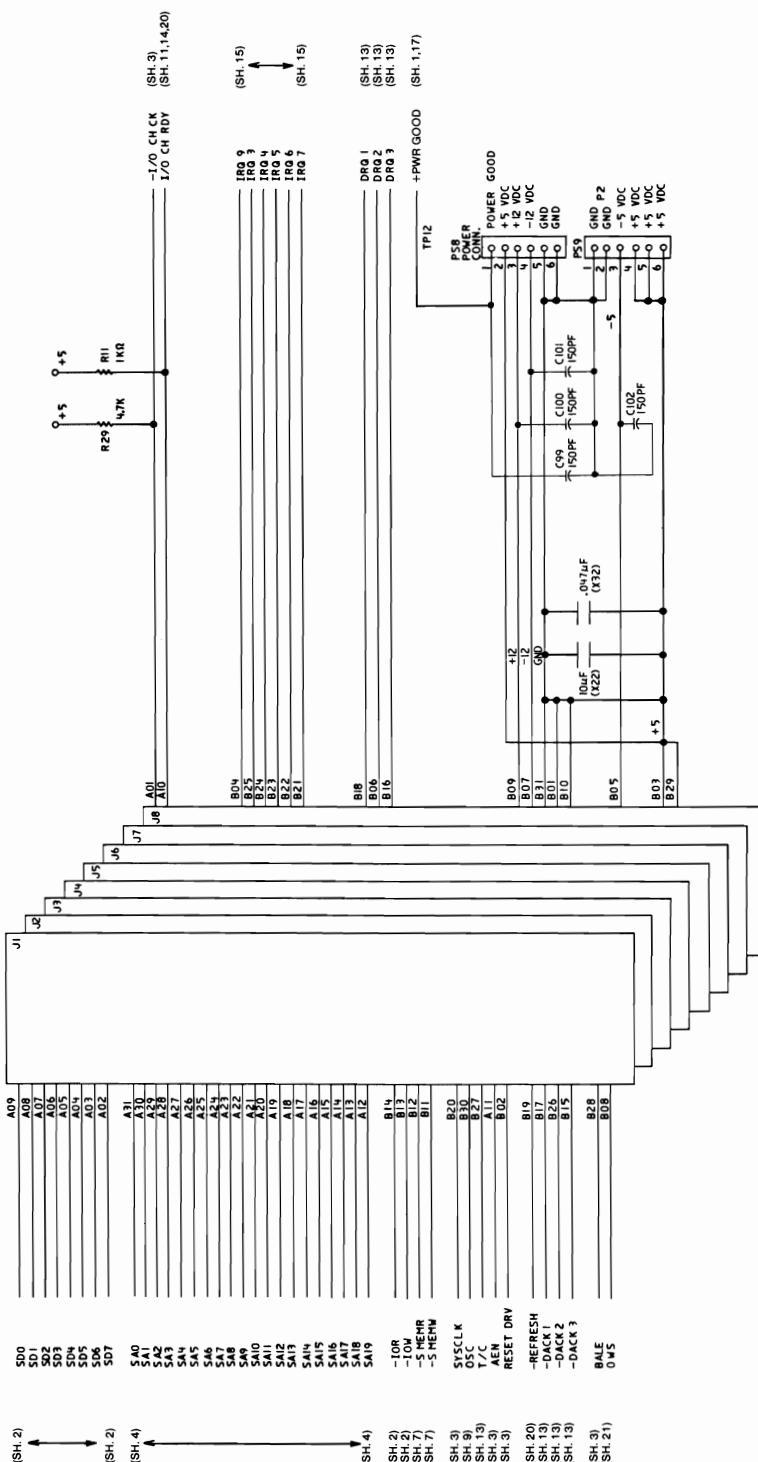




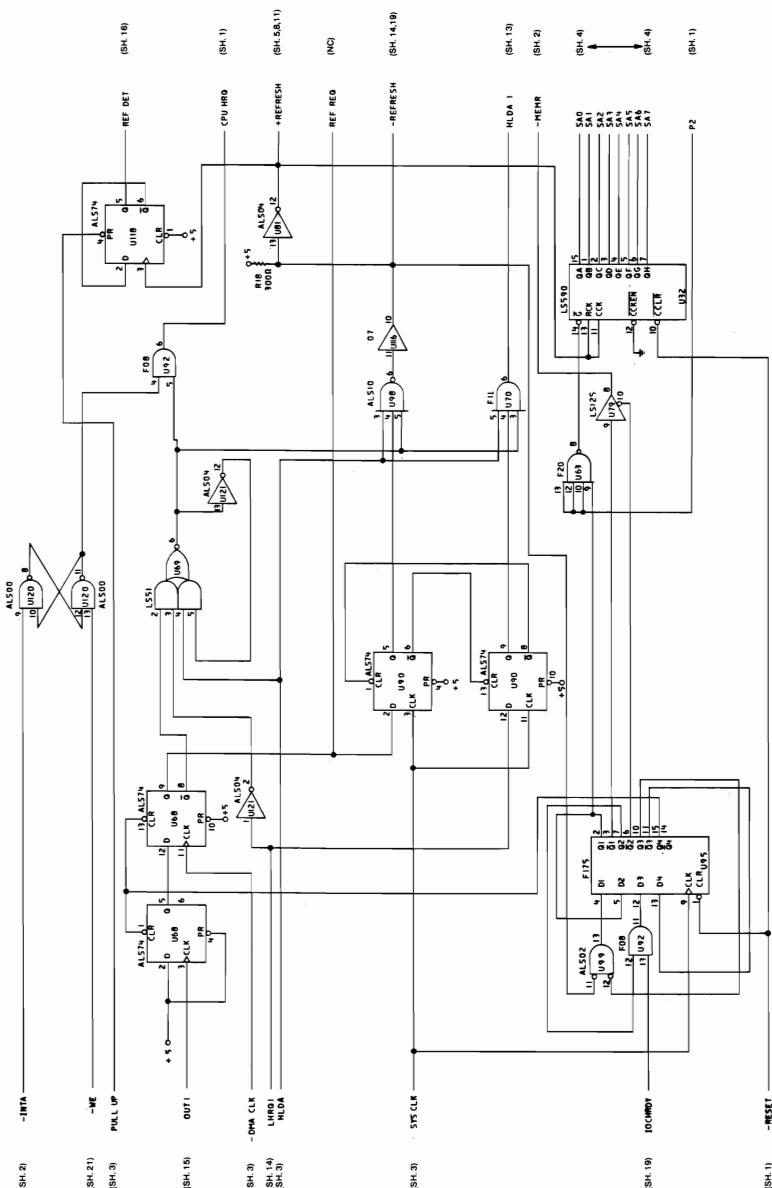
## **1-114 System Board**

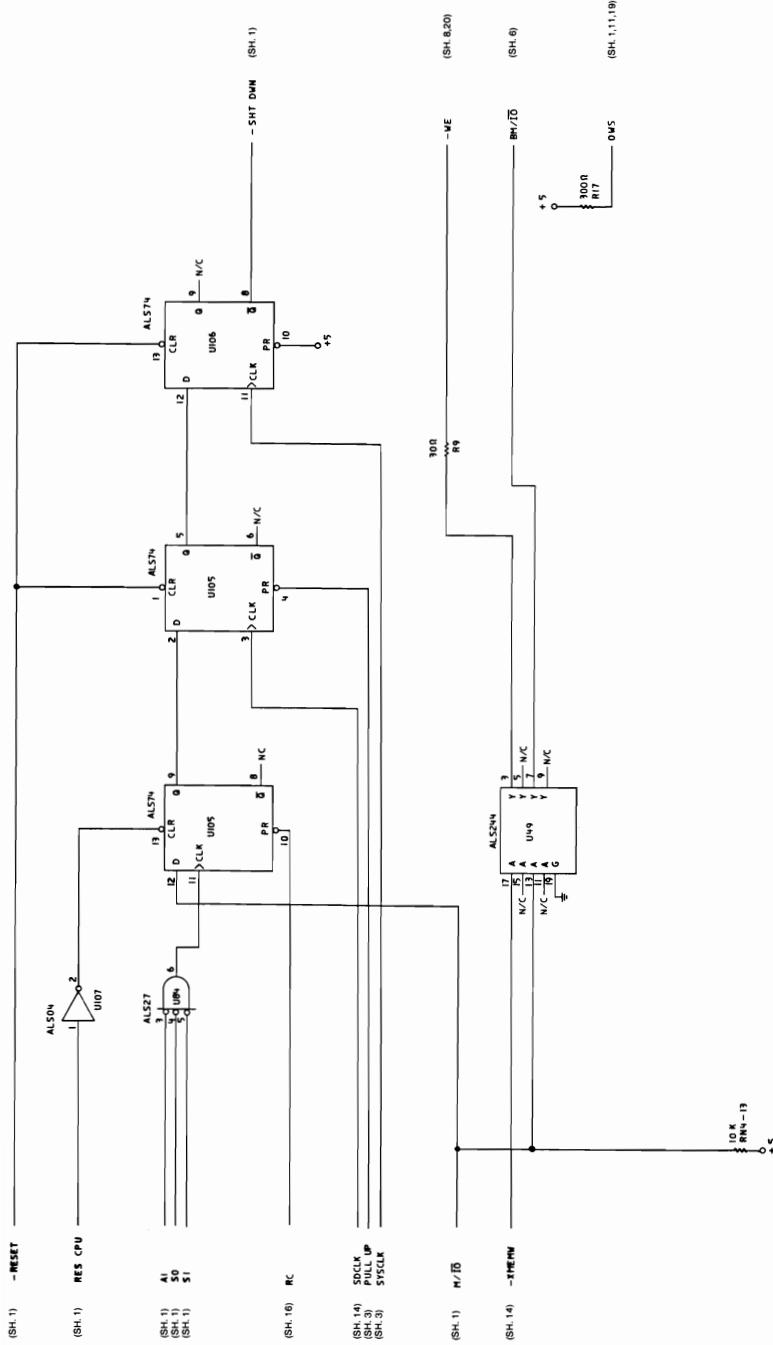


## 1-116 System Board



Type 2 512KB Planar (Sheet 19 of 21)





Type 2 512KB Planar (Sheet 21 of 21)

# SECTION 2. COPROCESSOR

## Contents

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

## Notes:

# Description

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

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

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

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

# Programming Interface

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

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

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

Data Type	Bits	Significant Digits (Decimal)	Approximate Range (Decimal)
Word Integer	16	4	$-32,768 \leq x \leq +32,767$
Short Integer	32	9	$-2 \times 10^9 \leq x \leq +2 \times 10^9$
Long Integer	64	19	$-9 \times 10^{18} \leq x \leq +9 \times 10^{18}$
Packed Decimal	80	18	$-99.99 \leq x \leq +99.99$ (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 the same clock generator as the microprocessor. It works at one-third the frequency of the system microprocessor (2.66 MHz). The coprocessor is wired so that it functions as an I/O device through I/O port addresses hex 00F8, 00FA, and 00FC. The microprocessor sends OP codes and operands through these I/O ports. The microprocessor also receives and stores results through the same I/O ports. The coprocessor's 'busy' signal informs the microprocessor that it is executing; the microprocessor's Wait instruction forces the microprocessor to wait until the coprocessor is finished executing.

The coprocessor detects six different exception conditions that can occur during instruction execution. If the appropriate exception mask within the coprocessor is not set, the coprocessor sets its error signal. This error signal generates a hardware interrupt (interrupt 13) and causes the 'busy' signal to the coprocessor to be held in the busy state. The 'busy' signal may

be cleared by an 8-bit I/O Write command to address hex F0 with D0 through D7 equal to 0.

The power-on self-test code in the system ROM enables IRQ 13 and sets up its vector to point to a routine in ROM. The ROM routine clears the 'busy' signal's latch and then transfers control to the address pointed to by the NMI interrupt vector. This allows code written for any IBM Personal Computer to work on an IBM Personal Computer AT. The NMI interrupt handler should read the coprocessor's status to determine if the NMI was caused by the coprocessor. If the interrupt was not generated by the coprocessor, control should be passed to the original NMI interrupt handler.

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

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

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

# Notes:

# SECTION 3. POWER SUPPLY

## Contents

<b>Inputs</b> .....	<b>3-3</b>
<b>Outputs</b> .....	<b>3-4</b>
<b>DC Output Protection</b> .....	<b>3-4</b>
<b>Output Voltage Sequencing</b> .....	<b>3-4</b>
<b>No-Load Operation</b> .....	<b>3-5</b>
<b>Power-Good Signal</b> .....	<b>3-5</b>
Load Resistor .....	3-5
<b>Connectors</b> .....	<b>3-7</b>

## Notes:

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

## Inputs

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

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

### Input Requirements

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

# Outputs

The power supply provides +5, -5, +12, and -12 Vdc. The following figure shows the load current and regulation tolerance for these voltages. The power supply also supplies either 115 Vac or 230 Vac for the IBM Monochrome Display.

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

## DC Load Requirements

# DC Output Protection

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

# Output Voltage Sequencing

Under normal conditions, the output voltage levels track within 300 milliseconds of each other when power is applied to, or removed from the power supply, provided at least minimum loading is present.

# No-Load Operation

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

## Power-Good Signal

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

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

## Load Resistor

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

The dc output-voltage sense signal holds the 'power-good' signal at a low level when power is switched on until all output voltages have reached their minimum sense levels. The 'power-good' signal has a turn-on delay of at least 100 milliseconds but not longer than 500 milliseconds and can drive six standard TTL loads.

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

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

**Sense Level**

# Connectors

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

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

## DC Load Distribution

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

## Notes:

# SECTION 4. KEYBOARD

## Contents

Description .....	4-3
Cabling .....	4-3
Sequencing Key Code Scanning .....	4-3
Keyboard Buffer .....	4-3
Keys .....	4-4
Power-On Routine .....	4-4
Power-On Reset .....	4-4
Basic Assurance Test .....	4-4
Commands from the System .....	4-5
Reset (Hex FF) .....	4-5
Resend (Hex FE) .....	4-6
No-Operation (NOP) (Hex FD through F7) .....	4-6
Set Default (Hex F6) .....	4-6
Default Disable (Hex F5) .....	4-6
Enable (Hex F4) .....	4-6
Set Typematic Rate/Delay (Hex F3) .....	4-7
No-Operation (NOP) (Hex F2 through EF) .....	4-8
Echo (Hex EE) .....	4-8
Set/Reset Mode Indicators (Hex ED) .....	4-8
Commands to the System .....	4-9
Resend (Hex FE) .....	4-9
ACK (Hex FA) .....	4-9
Overrun (Hex 00) .....	4-10
Diagnostic Failure (Hex FD) .....	4-10
Break Code Prefix (Hex F0) .....	4-10
BAT Completion Code (Hex AA) .....	4-10
ECHO Response (Hex EE) .....	4-10
Keyboard Scan-Code Outputs .....	4-11

<b>Clock and Data Signals</b> .....	<b>4-12</b>
Keyboard Data Output .....	4-13
Keyboard Data Input .....	4-13
 <b>Keyboard Layouts</b> .....	 <b>4-15</b>
French Keyboard .....	4-16
German Keyboard .....	4-17
Italian Keyboard .....	4-18
Spanish Keyboard .....	4-19
U.K. English Keyboard .....	4-20
U.S. English Keyboard .....	4-21
 <b>Specifications</b> .....	 <b>4-22</b>
Size .....	4-22
Weight .....	4-22
 <b>Logic Diagram</b> .....	 <b>4-23</b>

## Description

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

## Cabling

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

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

## Sequencing Key Code Scanning

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

## Keyboard Buffer

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

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

characters in the buffer will be sent as in normal operation, and new data entered will be detected and sent.

## Keys

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

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

## Power-On Routine

### Power-On Reset

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

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

## Basic Assurance Test

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

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

The BAT can also be started by a Reset command.

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

## Commands from the System

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

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

### Reset (Hex FF)

The system issues a Reset command to start a program reset and a keyboard internal self-test. The keyboard acknowledges the command with an 'acknowledge' signal (ACK) and ensures the

system accepts the ACK before executing the command. The system signals acceptance of the ACK by raising the clock and data for a minimum of 500 microseconds. The keyboard is disabled from the time it receives the Reset command until the ACK is accepted or until another command overrides the previous one. Following acceptance of the ACK, the keyboard begins the reset operation, which is similar to a power-on reset. The keyboard clears the output buffer and sets up default values for typematic and delay rates.

## **Resend (Hex FE)**

The system can send this command when it detects an error in any transmission from the keyboard. It can be sent only after a keyboard transmission and before the system enables the interface to allow the next keyboard output. Upon receipt of Resend, the keyboard sends the previous output again unless the previous output was Resend. In this case, the keyboard will resend the last byte before the Resend command.

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

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

## **Set Default (Hex F6)**

The Set Default command resets all conditions to the power-on default state. The keyboard responds with ACK, clears its output buffer, sets default conditions, and continues scanning (only if the keyboard was previously enabled).

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

This command is similar to Set Default, except the keyboard stops scanning and awaits further instructions.

## Enable (Hex F4)

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

## Set Typematic Rate/Delay (Hex F3)

The system issues this command, followed by a parameter, to change the typematic rate and delay. The typematic rate and delay parameters are determined by the value of the byte following the command. Bits 6 and 5 serve as the delay parameter and bits 4, 3, 2, 1, and 0 (the least-significant bit) are the rate parameter. Bit 7, the most-significant bit, is always 0. The delay is equal to 1 plus the binary value of bits 6 and 5 multiplied by 250 milliseconds  $\pm 20\%$ . The period (interval from one typematic output to the next) is determined by the following equation:

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

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

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

The default rate for the system keyboard is as follows:

The typematic rate = 10 characters per second  $\pm 20\%$  and the delay = 500 ms  $\pm 20\%$ .

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

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

## **Echo (Hex EE)**

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

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

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

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

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

11101101 – hex ED

The second byte is an option byte. It has a list of the indicators to be acted upon. The bit assignments for this option byte are as follows:

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

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

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

## Commands to the System

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

### 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.

## **ACK (Hex FA)**

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

## **Overrun (Hex 00)**

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

## **Diagnostic Failure (Hex FD)**

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

## **Break Code Prefix (Hex F0)**

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

## **BAT Completion Code (Hex AA)**

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

## **ECHO Response (Hex EE)**

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

# Keyboard Scan-Code Outputs

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

The typematic scan code for a key is the same as the key's make code. Refer to "Keyboard Layouts" beginning on page 4-15 to determine the character associated with each key number.

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

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

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

# Clock and Data Signals

The keyboard and system communicate over the 'clock' and 'data' lines. The source of each of these lines is an open-collector device on the keyboard that allows either the keyboard or the system to force a line to a negative level. When no communication is occurring, both the 'clock' and 'data' lines are at a positive level.

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

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

The parity bit is either 1 or 0, and the eight data bits plus the parity bit always equals an odd number.

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

When the keyboard sends data to, or receives data from the system, it generates the 'clock' signal to time the data. The system can prevent the keyboard from sending data by forcing the 'clock' line to a negative level; the 'data' line may go high or low during this time.

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

## Keyboard Data Output

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

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

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

## Keyboard Data Input

When the system is ready to send data to the keyboard, it first checks if the keyboard is sending data. If the keyboard is sending but has not reached the tenth clock, the system can override the keyboard output by forcing the 'clock' line to a negative level. If the keyboard transmission is beyond the tenth clock, the system must receive the transmission.

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

The keyboard checks the state of the 'clock' line at intervals of no less than 60 milliseconds. If a request-to-send is detected, the keyboard counts 11 bits. After the tenth bit, the keyboard forces the 'data' line low and counts one more (the stop bit). This action signals the system that the keyboard has received its data. Upon receipt of this signal, the system returns to a ready state, in which it can accept keyboard output, or goes to the inhibited state until it is ready.

Each system command or data transmission to the keyboard requires a response from the keyboard before the system can send its next output. The keyboard will respond within 20 milliseconds unless the system prevents keyboard output. If the keyboard response is invalid or has a parity error, the system sends the command or data again. A Resend command **should** not be sent in this case.

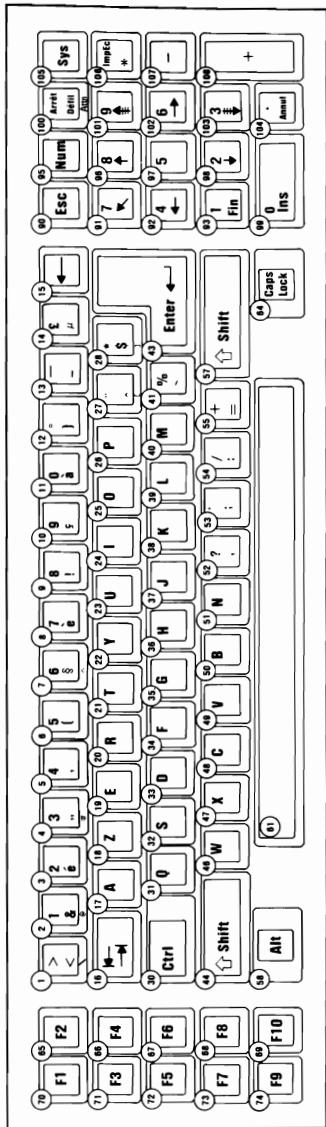
# Keyboard Layouts

The keyboard has six different layouts:

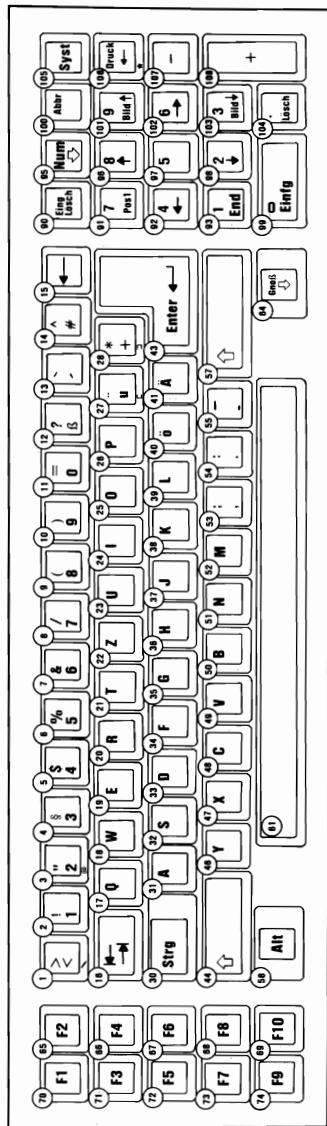
- French
- German
- Italian
- Spanish
- U.K. English
- U.S. English

The following pages show the six keyboard layouts.

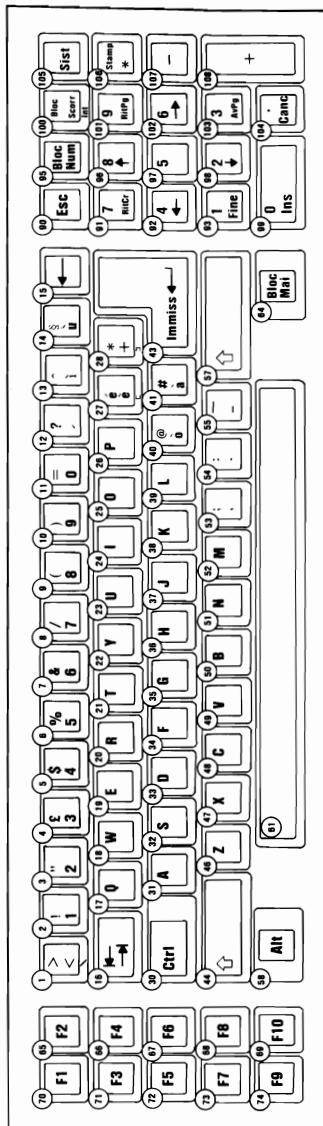
## French Keyboard



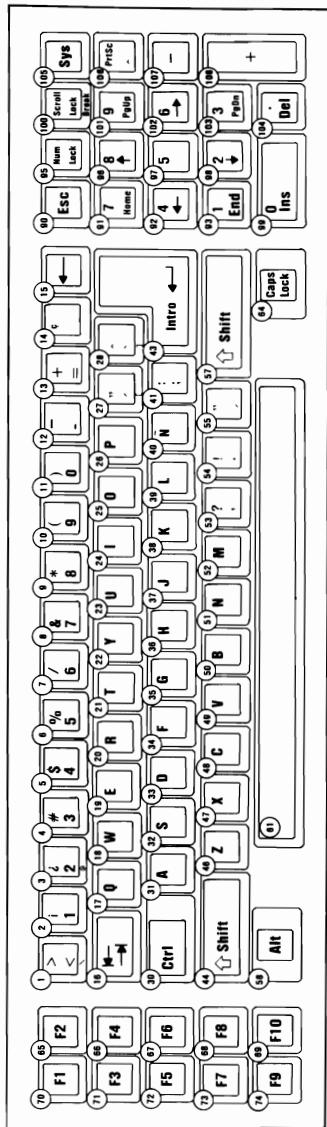
## German Keyboard



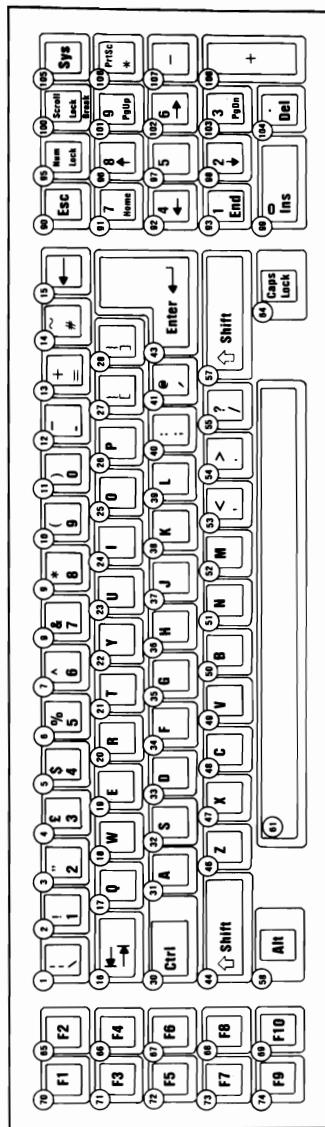
## Italian Keyboard



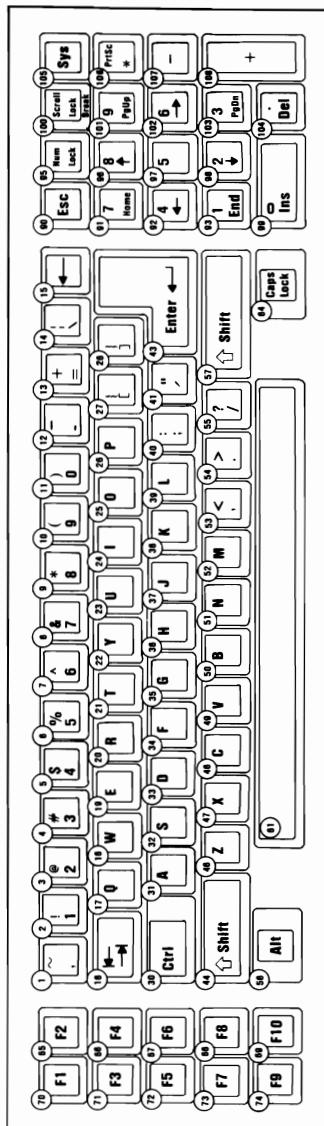
## Spanish Keyboard



## U.K. English Keyboard



## U.S. English Keyboard



# Specifications

## Size

- Length: 540 millimeters (21.6 inches)
- Depth: 100 millimeters (4 inches)
- Height: 225 millimeters (9 inches)

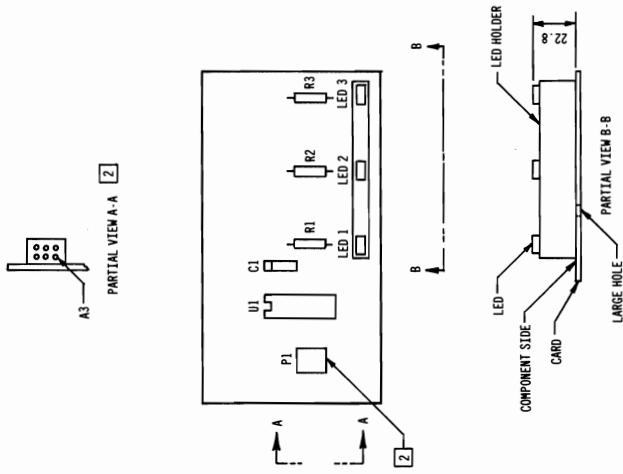
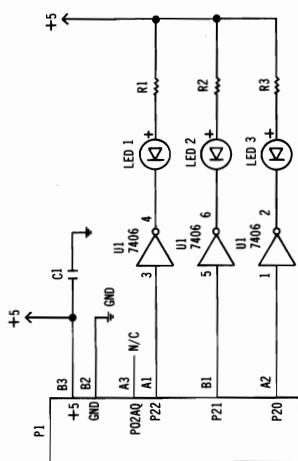
## Weight

- 2.8 kilograms (6.2 pounds)

# Logic Diagram

SECTION 4

## Enhancement Logic Card Assembly



## Notes:

# SECTION 5. SYSTEM BIOS

## Contents

<b>System BIOS Usage</b> .....	<b>5-3</b>
Parameter Passing .....	5-4
Vectors with Special Meanings .....	5-6
Other Read/Write Memory Usage .....	5-8
BIOS Programming Hints .....	5-10
Adapters with System-Accessible ROM Modules ..	5-12
Additional System Board ROM Modules .....	5-13
<b>Keyboard Encoding and Usage</b> .....	<b>5-13</b>
Character Codes .....	5-14
Extended Functions .....	5-18
Shift States .....	5-19
Special Handling .....	5-21
<b>Quick Reference</b> .....	<b>5-24</b>

## Notes:

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

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

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

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

## System BIOS Usage

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

## Parameter Passing

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

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

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

To read the time of day:

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

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

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

Int	Address	Name	BIOS Entry
0	0-3	Divide by Zero	D11
1	4-7	Single Step	D11
2	8-B	Nonmaskable	NMI INT
3	C-F	Breakpoint	D11
4	10-13	Overflow	D11
5	14-17	Print Screen	PRINT_SCREEN
6	18-1B	Reserved	D11
7	1C-1F	Reserved	D11
8	20-23	Time of Day	TIMER INT
9	24-27	Keyboard	KB_INT
A	28-2B	Reserved	D11
B	2C-2F	Communications	D11
C	30-33	Communications	D11
D	34-37	Alternate Printer	D11
E	38-3B	Diskette	DISK_INT
F	3C-3F	Printer	D11
10	40-43	Video	VIDEO IO
11	44-47	Equipment Check	EQUIPMENT
12	48-4B	Memory	MEMORY SIZE DETERMINE
13	4C-4F	Diskette/Disk	DISKETTE IO
14	50-53	Communications	RS232 IO
15	54-57	Cassette	CASSETTE IO/System Extensions
16	58-5B	Keyboard	KEYBOARD IO
17	5C-5F	Printer	PRINTER TO
18	60-63	Resident Basic	F600:0000
19	64-67	Bootstrap	BOOTSTRAP
1A	68-6B	Time of Day	TIME OF DAY
1B	6C-6F	Keyboard Break	DUMMY RETURN
1C	70-73	Timer Tick	DUMMY RETURN
1D	74-77	Video Initialization	VIDEO_PARMS
1E	78-7B	Diskette Parameters	DISK_BASE
1F	7C-7F	Video Graphics Chars	0

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

**Note:** For BIOS index, see the BIOS Quick Reference on page 5-24 .

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.

**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 the vector points to. Changing this parameter hook may be necessary to reflect the specifications of other fixed drives attached.

## Other Read/Write Memory Usage

The IBM BIOS routines use 256 bytes of memory from absolute hex 400 to hex 4FF. Locations hex 400 to 407 contain the base

addresses of any RS-232C adapters installed in the system. Locations hex 408 to 40F contain the base addresses of any printer adapters.

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

The following figure shows the reserved memory locations.

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

### Reserved Memory Locations

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*

### Basic Workspace Variables

\*Set to 1, 2, or 3 to get text in colors 1-3. Do not set to 0. The default is 3.

## Example

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

L	H
Hex 64	Hex 00

The following is a BIOS memory map.

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

## BIOS Memory Map

## BIOS Programming Hints

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

If an error is reported by the disk or diskette code, reset the drive adapter and retry the operation. A specified number of retries should be required for diskette reads to ensure the problem is not due to motor startup.

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

Additional information for BIOS programming can be found in Section 9 of this manual.

## Move Block BIOS

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

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

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

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

## Communication Interrupt Intervals

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

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

## Move Block BIOS Timing

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

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

## Additional System Board ROM Modules

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

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

**Byte 0** Hex 55

**Byte 1** Hex AA

**Byte 2** Not used

**Byte 3** Entry by a CALL FAR

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

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

## Keyboard Encoding and Usage

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

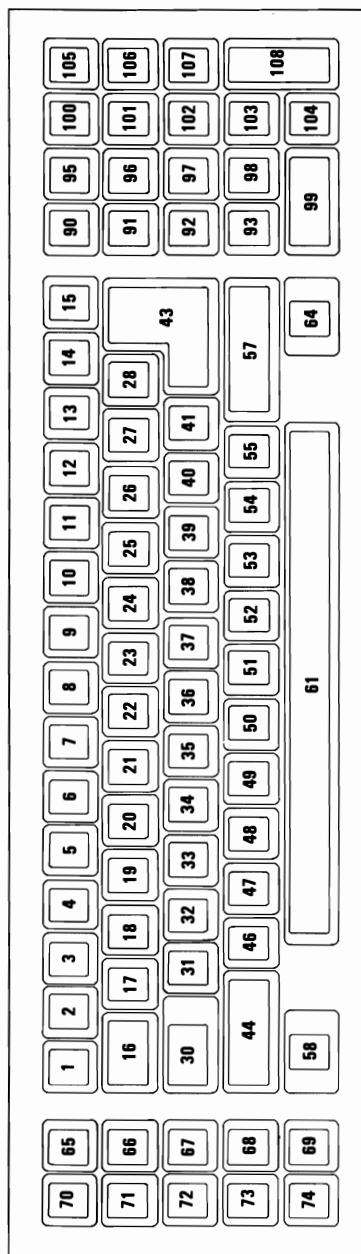
layout. Some operating systems may make provisions for alternate keyboard layouts by providing an interrupt replacer, which resides in the read/write memory. This section discusses only the ROM routine.

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

## Character Codes

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

The following figure shows the keyboard layout and key positions.



Key	Base Case	Uppercase	Ctrl	Alt
1	'	~	-1	-1
2	1	!	-1	(*)
3	2	@	Nu1(000) (*)	(*)
4	3	#	-1	(*)
5	4	\$	-1	(*)
6	5	%	-1	(*)
7	6	^	RS(030)	(*)
8	7	&	-1	(*)
9	8	¤	-1	(*)
10	9	(	-1	(*)
11	0	)	-1	(*)
12	-		US(031)	(*)
13	=		-1	(*)
14	\		FS(028)	-1
15	Backspace (008)	Backspace (008)	Del(127)	-1
16	→  (009)	← (*)	-1	-1
17	q	Q	DC1(017)	(*)
18	w	W	ETB(023)	(*)
19	e	E	ENQ(005)	(*)
20	r	R	DC2(018)	(*)
21	t	T	DC4(020)	(*)
22	y	Y	EM(025)	(*)
23	u	U	NAK(021)	(*)
24	i	I	HT(009)	(*)
25	o	O	SI(015)	(*)
26	p	P	DLE(016)	(*)
27	[	{	Esc(027)	(*)
28	]	}	GS(029)	-1
30 Ctrl	-1	-1	-1	-1
31	a	A	SOH(001)	(*)
32	s	S	DC3(019)	(*)
33	d	D	EOT(004)	(*)
34	f	F	ACK(006)	(*)
35	g	G	BEL(007)	(*)
36	h	H	BS(008)	(*)
37	j	J	LF(010)	(*)
38	k	K	VT(011)	(*)
39	l	L	FF(012)	(*)
40	;	:	-1	-1
41	,	;	-1	-1
43	CR	CR	LF(010)	-1
44 Shift (Left)	-1	-1	-1	-1
46	z	Z	SUB(026)	(*)
47	x	X	CAN(024)	(*)
48	c	C	ETX(003)	(*)

Notes:

(\*) Refer to "Extended Functions" in this section.

(\*\*) Refer to "Special Handling" in this section.

## Character Codes (Part 1 of 2)

Key	Base Case	Uppercase	Ctrl	Alt
49	v	V	SYN(022)	(*)
50	b	B	STX(002)	(*)
51	n	N	S0(014)	(*)
52	m	M	CR(013)	(*)
53	,	<	-1	-1
54	.	>	-1	-1
55	/	?	-1	-1
57 Shift (Right)	-1	-1	-1	-1
58 Alt	-1	-1	-1	-1
61	Space	Space	Space	Space
64 Caps Lock	-1	-1	-1	-1
90	Esc	Esc	Esc	-1
95 Num Lock	-1	-1 (*)	Pause (**)	-1
100 Scroll Lock	-1	-1	Break (**)	-1
107	-	-	(*)	(*)
108	Enter	Enter	-1	-1
112	Null (*)	Null (*)	Null (*)	Null(*)
113	Null (*)	Null (*)	Null (*)	Null(*)
114	Null (*)	Null (*)	Null (*)	Null(*)
115	Null (*)	Null (*)	Null (*)	Null(*)
116	Null (*)	Null (*)	Null (*)	Null(*)
117	Null (*)	Null (*)	Null (*)	Null(*)
118	Null (*)	Null (*)	Null (*)	Null(*)

Notes:

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

## Character Codes (Part 2 of 2)

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

Key	Num Lock	Base Case	Alt	Ctrl
91	7	Home (*) ← (*)	-1	Clear Screen
92	4		-1	Reverse Word (*)
93	1	End (*)	-1	Erase to EOL (*)
96	8	↑ (*)	-1	-1
97	5	-1	-1	-1
98	2	↓ (*)	-1	-1
99	0	Ins	-1	-1
101	9	Page Up (*)	-1	Top of Text and Home
102	6	→ (*)	-1	Advance Word (*)
103	3	Page Down (*)	-1	Erase to EOS (*)
104	.	Delete (*,**)	(**)	(**)
105	-	Sys Request	-1	-1
106	+	+ (*)	-1	-1

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

## Special Character Codes

## Extended Functions

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

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

Second Code	Function
3	Nul Character
15	← (Back-tab)
16-25	Alt Q, W, E, R, T, Y, U, I, O, P
30-38	Alt A, S, D, F, G, H, J, K, L
44-50	Alt Z, X, C, V, B, N, M
59-68	F1 to F10 Function Keys (Base Case)
71	Home
72	↑ (Cursor Up)
73	Page Up and Home Cursor
75	← (Cursor Left)
77	→ (Cursor Right)
79	End
80	↓ (Cursor Down)
81	Page Down and Home Cursor
82	Ins (Insert)
83	Del (Delete)
84-93	F11 to F20 (Shift-F1 through Shift-F10)
94-103	F21 to F30 (Ctrl-F1 through Ctrl-F10)
104-113	F31 to F40 (Alt-F1 through Alt-F10)
114	Ctrl PrtSc (Start/Stop Echo to Printer)
115	Ctrl ← (Reverse Word)
116	Ctrl → (Advance Word)
117	Ctrl End (Erase to End of Line-EOL)
118	Ctrl PgDn (Erase to End of Screen-EOS)
119	Ctrl Home (Clear Screen and Home)
120-131	Alt 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, -, = keys 2-13
132	Ctrl PgUp (Top 25 Lines of Text and Cursor Home)

### Keyboard Extended Functions

## Shift States

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

**Shift:** This key temporarily shifts keys 1 through 13, 15 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, 17 through 29, 31 through 39, 43, 46 through 52, 91 through 93, and 101 through 103 to the Ctrl state. The Ctrl key is also used with the Alt and Del keys to cause the system-reset function; with the Scroll Lock key to cause the break function; and with the Num Lock key to cause the pause function. The system-reset, break, and pause functions are described under "Special Handling" later in this section.

**Alt:** This key temporarily shifts keys 1 through 13, 17 through 26, 31 through 39, and 46 through 52 to the Alt state. The Alt key is also used with the Ctrl and Del keys to cause a system reset.

The Alt key also allows the user to enter any character code from 1 to 255.

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

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

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

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

changes the Scroll Lock Mode indicator. If the indicator was on, it will go off; and if it was off, it will go on.

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

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

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

## Special Handling

### System Reset

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

### Break

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

## Pause

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

## Print Screen

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

## System Request

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

Save the previous address.

Overlay interrupt vector hex 15.

Check AH for a value of hex 85:

If yes, process may begin.

If no, go to previous address.

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

## Other Characteristics

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

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

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

# Quick Reference

<b>Test1</b> .....	<b>5-28</b>
Data Area Description .....	5-30
Common POST and BIOS Equates .....	5-32
Test .01 Through Test .16 .....	5-36
POST and Manufacturing Test Routines .....	5-57
<b>Test2</b> .....	<b>5-58</b>
Test .17 Through Test .23 .....	5-58
<b>Test3. POST Exception Interrupt Tests</b> .....	<b>5-75</b>
<b>Test4. POST and BIOS Utility Routines</b> .....	<b>5-81</b>
CMOS_READ .....	5-81
CMOS_WRITE .....	5-81
E_MSG_P_MSG .....	5-82
ERR_BEEP .....	5-82
BEEP .....	5-83
WAITF .....	5-83
CONFIG_BAD .....	5-83
PRT_SEG .....	5-84
KBD_RESET .....	5-85
D11 - Dummy Interrupt Handler .....	5-87
Hardware Interrupt 9 Handler (Type 71) .....	5-87
<b>Test5. Exception Interrupt Tests</b> .....	<b>5-88</b>
SYSINIT1 - Build Protected Mode Descriptors .....	5-89
GDT_BLD - Build the GDT for POST .....	5-89
SIDT_BLD - Build the IDT for POST .....	5-91
<b>Test6</b> .....	<b>5-93</b>
STGTST_CNT .....	5-93
ROM_ERR .....	5-95
XMIT_8042 .....	5-95
BOOT_STRAP .....	5-95
<b>Diskette BIOS</b> .....	<b>5-97</b>
<b>Fixed Disk (Hard File) BIOS</b> .....	<b>5-116</b>

<b>Keyboard BIOS</b> .....	<b>5-129</b>
<b>Printer BIOS</b> .....	<b>5-138</b>
<b>RS232 BIOS</b> .....	<b>5-140</b>
<b>Video BIOS</b> .....	<b>5-143</b>
<b>BIOS</b> .....	<b>5-161</b>
Memory Size Determine .....	5-161
Equipment Determine .....	5-161
NMI .....	5-162
<b>BIOS1</b> .....	<b>5-163</b>
Event Wait .....	5-164
Joystick Support .....	5-165
Wait .....	5-166
Block Move .....	5-167
Extended Memory Size Determine .....	5-172
Processor to Virtual Mode .....	5-174
<b>BIOS2</b> .....	<b>5-176</b>
Time of Day .....	5-176
Alarm Interrupt Handler .....	5-179
Print Screen .....	5-180
Timer 1 Interrupt Handler .....	5-181
<b>ORGS - PC Compatibility and Tables</b> .....	<b>5-182</b>
POST Error Messages .....	5-182

Address	Publics by Name	Address	Publics by Value
F000:1E729	A1	F000:00000	POST1
F000:1E833	ACT_DISP_PAGE	F000:00100	Abs K6L
F000:1E600	BASIC	F000:00110	Abs NM
F000:1E9F0	BEEP	F000:00500	START_1
F000:1B1A	BLINK_INT	F000:0396	C8042_
F000:2022	BOOT_STRAP_I	F000:03A2	DBF_42
F000:1C596	C21	F000:0C96	POST2
F000:1E916	C3042	F000:1096	C21
F000:1E135	CASSETTE_IO_1	F000:1092	SHUT3
F000:1E941	CMOS_READ	F000:1086	SHUT2
F000:1E958	CMOS_WRITE	F000:1089	SHUT7
F000:1A45	CONFIG_BAD	F000:10DA	SHUT6
F000:1E6F5	CONF_TBL	F000:1613	SHUT4
F000:1E9E4	CRT_CHAR_GEN	F000:1671	POST3
F000:1E020	D	F000:1941	CMOS_READ
F000:1EBCA	D11	F000:1941	POST4
F000:1E030	D2	F000:195B	CMOS_WRITE
F000:1E040	D2A	F000:1975	DDS
F000:1E195	DDS	F000:197D	E_MSG
F000:120E3	DISKETTE_IO_1	F000:19A4	P_MSG
F000:1E917	DISK_PAGE	F000:19A8	ERR_BEEP
F000:1E117	DISK_INT_1	F000:19F0	BEEP
F000:1C2B8	DISK_IO	F000:1A36	WAITF
F000:1E282	DISK_SETUP	F000:1A45	CONFIG_BAD
F000:1E2A2	DISKETTE_SETUP	F000:1A59	XPC_BYT
F000:1F53	DUMMY_RETURN	F000:1A69	PRT_HEX
F000:1E918	DUMMY_RETURN_I	F000:1A70	PRT_SEC
F000:1E05E	E101	F000:1A85	PROT_PRT_HEX
F000:1E077	E102	F000:1A81	ROM_CHECKSUM
F000:1E090	E103	F000:1ABD	ROM_CHECK
F000:1E049	E104	F000:1AEF	KBD_RESET
F000:1E02C	E105	F000:1B1A	BLINK_INT
F000:1E109	E106	F000:1B28	SET_TOD
F000:1E0F4	E107	F000:1B4A	D
F000:1E10D	E108	F000:1C18	DUMMY_RETURN_I
F000:1E126	E109	F000:1C19	RE_DIRECT
F000:1E13F	E161	F000:1C22	INT_287
F000:1E168	E162	F000:1C31	PROC_SHUTDOWN
F000:1E141	E163	F000:1C38	POST
F000:1E187	E164	F000:1C4A	SYSINITI
F000:1E1D8	E201	F000:1EB5	POST6
F000:1E1EE	E202	F000:1EB5	STGTST_CNT
F000:1E209	E203	F000:1FB5	ROM_ERR
F000:1E224	E301	F000:1FE1	XMIT_8042
F000:1E239	E302	F000:1F22	BOOT_STRAP_I
F000:1E226	E303	F000:2093	DISKETTE_IO_1
F000:1E2A2	E304	F000:28C1	SEEK
F000:1E30E	E401	F000:2A17	DISK_INT_1
F000:1E31E	E501	F000:2A2E	DISKETTE_SETUP
F000:1E32E	E601	F000:2A82	DISK_SETUP
F000:1E343	E602	F000:2C2B	DISK_IO
F000:1E048	EQUIPMENT_I	F000:317F	HD_INT
F000:1E982	ERR_BEEP	F000:3172	KEYBOARD_IO_1
F000:1E17D	E_MSG	F000:31FE	KB_INT_1
F000:1E364	FT780	F000:3267	K16
F000:1E379	FT178	F000:366C	SND_DATA
F000:1E37E	FT178	F000:3716	PRINTER_IO_1
F000:1E34E	FT179	F000:3717	R5238_IO_1
F000:1E38F	FT1791	F000:3880	VIDEO_IO_1
F000:1E302	F3A	F000:38EF	SET_MODE
F000:1E25D	F3D	F000:39BF	SET_CTYPE
F000:1E30F	F3D1	F000:39E4	SET_CPOS
F000:1E301	FTD_BL	F000:3A0C	READ_CURSOR
F000:1E488	FILL	F000:3A23	ACQ_DISP_PAGE
F000:1F5E	FLOPPY	F000:3A47	SET_COLOR
F000:1E450	GATE_A20	F000:3A6D	VIDEO_STATE
F000:1314F	HD_INT	F000:3A90	SCROLL_UP
F000:1F55A	HRD	F000:3B2F	SCROLL_DOWN
F000:1E922	INT_287	F000:3B80	READ_E_CURRENT
F000:1E8E1	K10	F000:3B8D	WRITE_E_CURRENT
F000:1E91B	K11	F000:3C00	WRITE_C_CURRENT
F000:1E955	K12	F000:3C6D	READ_DOT
F000:1E95F	K13	F000:3CCE	WRITE_DOT
F000:1E969	K14	F000:3F72	WRITE_TTY
F000:1E976	K15	F000:40F9	READ_LPEN
F000:1E927	K16	F000:40F9	MEMORY_SIZE_DET_I
F000:1E87E	K6	F000:40A8	EQUIPMENT_I
F000:1E008	Abs K6L	F000:40B2	NMI_INT_I
F000:1E866	K7	F000:4135	CASSETTE_IO_1
F000:1E8E6	K8	F000:43BF	SHUT9
F000:1EBC8	K9	F000:4501	GATE_A20
F000:1E95F	KBD_RESET	F000:45D0	INT_DAY_I
F000:131FE	KB_TNT_I	F000:473F	RTC_INT
F000:3172	KEYBOARD_IO_1	F000:47A9	PRINT_SCREEN_I
F000:0010	Abs M4	F000:483F	TIMER_INT_I
F000:1F044	M5	F000:4888	FILL
F000:1F0EC	M6	F000:6000	BASIC
F000:1E914	M7	F000:6000	D
F000:1E90E	MEMORY_SIZE_DET_I	F000:6E30	D2
F000:1E2C3	NMI_INT	F000:6E40	D2A
F000:1E082	NMI_INT_I	F000:6E5E	E101
F000:1E3A2	OBF_42	F000:6E77	E102
F000:1E900	POST1	F000:6E90	E103
F000:1E906	POST2	F000:6E94	E104
F000:1E171	POST3	F000:6ECA	E105
F000:1E941	POST4	F000:6E0B	E106
F000:1C38	POST5	F000:6E04	E107
F000:1E85	POST6	F000:E10D	E108

F000:3716	PRINTER_IO_I	F000:E126	E109
F000:FF54	PRINT_SCREEN	F000:E13F	E161
F000:47A9	PRINT_SCREEN_I	F000:E168	E162
F000:1C31	PROC_SHUTDOWN	F000:E191	E163
F000:1A85	PROT_PRT_HEX	F000:E1B7	E164
F000:1A69	PRT_HEX	F000:E1DB	E201
F000:1A70	PRT_SEG	F000:E1E2	E202
F000:1946	P_WSD	F000:E209	E203
F000:FFF0	P_O_R	F000:E224	E301
F000:3B81	READ_AC_CURRENT	F000:E239	E302
F000:3AC0	READ_CURSOR	F000:E25D	F3D
F000:3C80	READ_DOT	F000:E2C3	NM1_INT
F000:3B79	READ_OPEN	F000:E2C6	E203
F000:1C19	REJECT	F000:E2E4	E204
F000:1ABD	ROM_CHECK	F000:E30E	E401
F000:1AB1	ROM_CHECKSUM	F000:E31E	E501
F000:1FB5	ROM_ERR	F000:E32E	E601
F000:37A0	RS232_IO_I	F000:E343	E602
F000:43F6	RTN	F000:E364	F1780
F000:3B2F	SCROLL_DOWN	F000:E379	F1781
F000:3A90	SCROLL_UP	F000:E38E	F1782
F000:28C1	SEEK	F000:E3AC	F1790
F000:FF62	SEEKS_I	F000:E3BF	F1791
F000:3A47	SET_COLOR	F000:E3D2	F3A
F000:39E4	SET_POS	F000:E3DF	F3D1
F000:39EF	SET_CTYPE	F000:E401	F4_TBL
F000:38EF	SET_MODE	F000:E4F5	CONF_TBL
F000:1B28	SET_TOD	F000:E729	A1
F000:10B6	SHUT2	F000:E87E	K6
F000:1052	SHUT3	F000:E88E	K7
F000:1A13	SHUT4	F000:E89E	K8
F000:10DA	SHUT6	F000:E8C9	K9
F000:10B9	SHUT7	F000:E8E1	K10
F000:43BF	SHUT9	F000:E91B	K11
F000:FF23	SLAVE_VECTOR_TABLE	F000:E955	K12
F000:366C	SND_DATA	F000:E95F	K13
F000:1050	START	F000:E969	K14
F000:1E95	STRTST_CNT	F000:E973	K15
F000:1D2A	SYSINITI	F000:EFCT	DISK_BASE
F000:483F	TIMER_INT_I	F000:F0A4	VIDEO_PARMS
F000:45BD	TIME_OF_DAY_I	F000:F0E4	M5
F000:FF66	TUTOR	F000:F0EC	M6
F000:FFC3	VECTOR_TABLE	F000:F0F4	M7
F000:10D0	VIDEO_TTY	F000:F1E6	CRT_CHAR_GEN
F000:F0A4	VIDEO_PARMS	F000:FEF3	VECTOR_TABLE
F000:3A6D	VIDEO_STATE	F000:FF23	SLAVE_VECTOR_TABLE
F000:1A36	WAITF	F000:FF53	DUMMY_RETURN
F000:3BDB	WRITE_AC_CURRENT	F000:FF54	PRINT_SCREEN
F000:3C0D	WRITE_C_CURRENT	F000:FF5A	HKEY
F000:3FCE	WRITE_I	F000:FFEE	FLOPPY
F000:3FT2	WRITE_TTY	F000:FF62	SEEKS_I
F000:1FE1	XMIT_8042	F000:FF66	TUTOR
F000:1A59	XPC_BYTE	F000:FFF0	P_O_R

THE BIOS ROUTINES ARE MEANT TO BE ACCESSED THROUGH SOFTWARE INTERRUPTS ONLY. ANY ADDRESSES PRESENT IN THESE LISTINGS ARE INCLUDED ONLY FOR COMPLETENESS, NOT FOR REFERENCE. APPLICATIONS WHICH REFERENCE ANY ADDRESS IN THESE LISTINGS WITHIN THE CODE SEGMENT OF BIOS VIOLATE THE STRUCTURE AND DESIGN OF BIOS. ADDRESSES WITHIN THE BIOS CODE SEGMENT ARE SUBJECT TO CHANGE AND ROUTINES SHOULD BE ACCESSED THROUGH POINTERS IN THE INTERRUPT VECTORS OR WHEN NECESSARY THROUGH THE POINTERS IN THE BIOS "DATA" SEGMENT.

PAGE 118,121  
TITLE TEST1 ---- 06/10/85 POWER ON SELF TEST (POST)  
.286C

## BIOS I/O INTERFACE

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

## MODULE REFERENCE

TEST1.ASM	--> POST AND MANUFACTURING TEST ROUTINES	
DSEG.INC	--> DATA SEGMENTS LOCATIONS	
POSTEQU.INC	--> COMMON EQUATES FOR POST AND BIOS	
SYSDATA.INC	--> POWER ON SELF TEST EQUATES FOR PROTECTED MODE	
	POST TEST_01 THROUGH TEST_16	
TEST2.ASM	--> POST TEST_17 THROUGH TEST_22	
TEST3.ASM	--> POST EXCEPTION INTERRUPT TESTS	
TEST4.ASM	--> POST AND BIOS UTILITY ROUTINES	
	CMSOS_READ - READ CMOS LOCATION ROUTINE	
	CMSOS_WRITE - WRITE CMOS LOCATION ROUTINE	
	DDOS - DATA (DD) 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	
	WAIT - F10H INTERRUPT 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	
	PROG_PHR_HEX - POST PROTECTED MODE DISPLAY	
	ROM_BLOCKSUM - ROM BLOCK ROM MODULES FOR CHECKSUM	
	ROM_CHECK - ROM SCAN AND INITIALIZATION	
	KBD_RESET - POST KEYBOARD RESET ROUTINE	
	BLINK_INT - MANUFACTURING TOGGLE BIT ROUTINE	
	SET_TOD - SET TIMER FROM CMOS RTC	
	D11 - DYNAMIC INTERRUPT HANDLER ->INT ??H	
	RE_DIRECT - HARDWARE INT 9 REDIRECT (L 2)	
	INT_287 - HARDWARE INT 13 REDIRECT (287)	
	PROC_SHUTDOWN - B0286 RESET ROUTINE	
TEST5.ASM	--> EXCEPTION INTERRUPT TEST HANDLERS FOR POST TESTS	
	SYNINITI - BUILD PROTECTED MODE POINTERS	
	GDT_BLD - BUILD THE GDT FOR POST	
	SGT_BLD - BUILD THE SGT FOR POST	
TEST6.ASM	--> POST TESTS AND SYSTEM BOOT STRAP	
	STGT_CNT - SEGMENT STORAGE TEST	
	ROM_ERR - ROM ERROR DISPLAY ROUTINE	
	XMIT_8042 - KEYBOARD DIAGNOSTIC OUTPUT	
	BOOT_STRAP - BOOT STRAP LOADER - INT 19H	
DSKETTE.ASM	--> DISKETTE BIOS	
	DISKETTE_10_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_10_0 - INT 13H BIOS ENTRY - INT 13H	
	HD_INT - HARDWARE INTERRUPT HANDLER - INT 7EH	
KYBD.ASM	--> KEYBOARD BIOS	
	KEYBOARD_10_0 - INT 16H BIOS ENTRY - INT 16H	
	KB_INT - HARDWARE INTERRUPT - INT 09H	
	SND_DATA - KEYBOARD TRANSMISSION	
PRT.ASM	--> PRINTER ADAPTER BIOS - INT 17H	
RS232.ASM	--> COMMUNICATIONS BIOS FOR RS232 - INT 14H	
VIDEO1.ASM	--> VIDEO BIOS - INT 10H	
BIOS.ASM	--> BIOS ROUTINES	
	MEMORY_SIZE_DET_1 - REAL MODE SIZE - INT 12H	
	EQUIMENT_1 - EQUIPMENT DETERMINATION - INT 12H	
	NMI_1 - NMI HANDLER - INT 02H	
BIOS1.ASM	--> INTERRUPT_15H BIOS ROUTINES - INT 15H	
	DEV_OPEN - NULL DEVICE OPEN HANDLER	
	DEV_CLOSE - NULL DEVICE CLOSE HANDLER	
	PROG_TERM - NULL PROGRAM TERMINATION	
	EVENT_WAIT - RTC EVENT AND INTERRUPT ROUTINE	
	JOYSTICK - 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_CODE - SET CPU PROGRAM VIRTUAL MODE	
	DEVICE_BUSY - NULL DEVICE BUSY HANDLER	
	INT_COMPLETE - NULL INTERRUPT COMPLETE HANDLER	
BIOS2.ASM	--> BIOS INTERRUPT ROUTINES	
	TIME_OF_DAY_1 - TIME OF DAY ROUTINES - INT 1AH	
	RTC_INT - REAL TIME CLOCK ALARM HANDLER - INT 70H	
	PRINT_SCREEN - PRINT SCREEN ROUTINE - INT 09H	
	TIMER_INT_1 - 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   : NON-MASKABLE INTERRUPT VECTOR
123 0008 ????????
124
125 0014      *INT5_PTR  ORG 4*005H   : PRINT SCREEN INTERRUPT VECTOR
126 0014 ????????
127
128 0020      *INT_PTR   ORG 4*008H   : HARDWARE INTERRUPT POINTER (8-F)
129 0020 ????????
130
131 0040      *VIDEO_INT ORG 4*010H   : VIDEO I/O INTERRUPT VECTOR
132 0040 ????????
133
134 004C      *ORG_VECTOR ORG 4*013H   : DISKETTE/DISK INTERRUPT VECTOR
135 004C ????????
136
137 0060      *BASIC_PTR ORG 4*018H   : POINTER TO CASSETTE BASIC
138 0060 ????????
139
140 0074      *PARM_PTR  ORG 4*01DH   : POINTER TO VIDEO PARAMETERS
141 0074 ????????
142
143 0078      *DISK_POINTER ORG 4*01EH   : POINTER TO DISKETTE PARAMETER TABLE
144 0078 ????????
145
146 007C      *EXT_PTR   ORG 4*01FH   : POINTER TO GRAPHIC CHARACTERS 128-255
147 007C ????????
148
149 0100      *DISK_VECTOR ORG 4*040H   : POINTER TO DISKETTE INTERRUPT CODE
150 0100 ????????
151
152 0104      *HF_TBL_VEC ORG 4*041H   : POINTER TO FIRST DISK PARAMETER TABLE
153 0104 ????????
154
155 0118      *HF1_TBL_VEC ORG 4*046H   : POINTER TO SECOND DISK PARAMETER TABLE
156 0118 ????????
157
158 01C0      *SLAVE_INT_PTR ORG 4*070H   : POINTER TO SLAVE INTERRUPT HANDLER
159 01C0 ????????
160
161 01D8      *HDISK_INT  ORG 4*076H   : POINTER TO FIXED DISK INTERRUPT CODE
162 01D8 ????????
163
164 0400      *TOS        ORG 0400H   : STACK -- USED DURING POST ONLY
165 0400 ??    *TOS        DW ?      : USE WILL OVERLAY INTERRUPTS VECTORS
166
167
168 0500      *MFG_TEST_RTN ORG 0500H   : LOAD LOCATION FOR MANUFACTURING TESTS
169 0500
170
171 7C00      *BOOT_LOCN  LABEL 7C00H   : BOOT STRAP CODE LOAD LOCATION
172 7C00
173
174 7C00      ABS0      ENDS

```

175  
176  
177  
178  
179  
180 0000  
181  
182 0000 ????  
183 0002 ????  
184 0004 ????  
185 0006 ????  
186 0008 ????  
187 000A ????  
188 000C ????  
189 000E ????  
190 0010 ????  
191 0012 ????  
192 0013 ????  
193 0015 ????  
194 0016 ????  
195  
196  
197  
198  
199  
200 0017 ????  
201 0018 ????  
202 0019 ????  
203 001A ????  
204 001C ????  
205  
206  
207  
208 001E 10 [ ???? ]  
209  
210  
211  
212  
213  
214  
215  
216 003E ????  
217  
218  
219 003F ????  
220  
221  
222 0040 ????  
223 0041 ????  
224  
225 0042 07 [ ???? ]  
226  
227  
228  
229  
230  
231  
232  
233  
234 0049 ????  
235 004A ????  
236 004C ????  
237 004E ????  
238 0050 08 [ ???? ]  
239  
240  
241  
242 0060 ????  
243 0062 ????  
244 0063 ????  
245 0065 ????  
246 0066 ????  
247  
248  
249  
250  
251  
252  
253 0067 ????  
254 0069 ????  
255 006B ????  
256  
257  
258  
259  
260  
261 006C ????  
262 006E ????  
263 0070 ????  
264  
265  
266  
267  
268  
269 0071 ????  
270 0072 ????  
271  
272  
273  
274  
275  
276 0074 ????  
277 0075 ????  
278 0076 ????  
279 0077 ????  
C PAGE  
C -----  
C | ROM BIOS DATA AREAS :  
C -----  
C DATA SEGMENT AT 40H : ADDRESS= 0040:0000  
C  
C @RS232\_BASE DW ? : BASE ADDRESSES OF RS232 ADAPTERS  
C @RS232\_BASE DW ? : SECOND LOGICAL RS232 ADAPTER  
C @RS232\_BASE DW ? : RESERVED  
C @RS232\_BASE DW ? : RESERVED  
C @PRINTER\_BASE DW ? : BASE ADDRESSES OF PRINTER ADAPTERS  
C @PRINTER\_BASE DW ? : SECOND LOGICAL PRINTER ADAPTER  
C @PRINTER\_BASE DW ? : THIRD LOGICAL PRINTER ADAPTER  
C @PRINTER\_BASE DW ? : RESERVED  
C @EQUIP\_FLAG DW ? : INSTALLED HARDWARE FLAGS  
C @MFG\_TST DB ? : INITIALIZATION FLAGS  
C @MEMORY\_SIZE DW ? : BASE MEMORY SIZE IN K BYTES (X 1024)  
C @MFG\_ERR\_FLAG DB ? : SCRATCHPAD FOR MANUFACTURING  
C @MFG\_ERR\_FLAG DB ? : ERROR CODES  
C  
C -----  
C | KEYBOARD DATA AREAS :  
C -----  
C  
C @KB\_FLAG DB ? : KEYBOARD SHIFT STATE AND STATUS FLAGS  
C @KB\_FLAG\_1 DB ? : SECOND BYTE OF KEYBOARD STATUS  
C @ALT\_INPUT DB ? : STORAGE FOR ALTERNATE KEY PAD ENTRY  
C @BUFFER\_HEAD DW ? : POINTER TO HEAD OF KEYBOARD BUFFER  
C @BUFFER\_TAIL DW ? : POINTER TO TAIL OF KEYBOARD BUFFER  
C  
C ----- HEAD = TAIL INDICATES THAT THE BUFFER IS EMPTY  
C @KB\_BUFFER DW 16 DUP(?) : ROOM FOR 15 SCAN CODE ENTRIES  
C  
C -----  
C | DISKETTE DATA AREAS :  
C -----  
C  
C @SEEK\_STATUS DB ? : DRIVE RECALIBRATION STATUS  
C @SEEK\_STATUS DB ? : BIT 0 = DRIVE 3-0 RECALIBRATION  
C @SEEK\_STATUS DB ? : BEFORE NEXT SEEK IF BIT IS = 0  
C @MOTOR\_STATUS DB ? : MOTOR STATUS  
C @MOTOR\_STATUS DB ? : BIT 3-0 = DRIVE 3-0 CURRENTLY RUNNING  
C @MOTOR\_STATUS DB ? : BIT 7 = CURRENT OPERATION IS A WRITE  
C @MOTOR\_COUNT DB ? : TIME OUT COUNTER FOR MOTOR(S) TURN OFF  
C @DISKETTE\_STATUS DB ? : RETURN CODE STATUS BYTE  
C @DISKETTE\_STATUS DB ? : CMD\_BLOCK IN STACK FOR DISK OPERATION  
C @NEC\_STATUS DB 7 DUP(?) : STATUS BYTES FROM DISKETTE OPERATION  
C  
C -----  
C | VIDEO DISPLAY DATA AREA :  
C -----  
C  
C @CRT\_MODE DB ? : CURRENT DISPLAY MODE (TYPE)  
C @CRT\_PAGES DW ? : NUMBER OF SCROLLS ON SCREEN  
C @CRT\_LEN DW ? : LENGTH OF REGEN BUFFER IN BYTES  
C @CRT\_START DW ? : STARTING ADDRESS IN REGEN BUFFER  
C @CURSOR\_POSN DW 8 DUP(?) : CURSOR FOR EACH OF UP TO 8 PAGES  
C  
C -----  
C @CURSOR\_MODE DW ? : CURRENT CURSOR MODE SETTING  
C @ACTIVE\_PAGE DB ? : CURRENT PAGE BEING DISPLAYED  
C @ADDR\_6845 DW ? : BASE ADDRESS FOR ACTIVE DISPLAY CARD  
C @CRT\_MODE\_SET DB ? : CURRENT SETTING OF THE 3x8 REGISTER  
C @CRT\_PALETTE DB ? : CURRENT PALETTE SETTING - COLOR CARD  
C  
C -----  
C | POST AND BIOS WORK DATA AREA :  
C -----  
C  
C @IO\_ROM\_INIT DW ? : STACK SAVE, etc.  
C @IO\_ROM\_INIT DW ? : POINTER TO ROM INITIALIZATION ROUTINE  
C @IO\_ROM\_SEG DW ? : POINTER TO I/O ROM SEGMENT  
C @INTR\_FLAG DB ? : FLAG INDICATING AN INTERRUPT HAPPENED  
C  
C -----  
C | TIMER DATA AREA :  
C -----  
C  
C @TIMER\_LOW DW ? : LOW WORD OF TIMER COUNT  
C @TIMER\_HIGH DW ? : HIGH WORD OF TIMER COUNT  
C @TIMER\_OFNL DB ? : TIMER HAS ROLLED OVER SINCE LAST READ  
C  
C -----  
C | SYSTEM DATA AREA :  
C -----  
C  
C @BIOS\_BREAK DB ? : BIT 7=1 IF BREAK KEY HAS BEEN PRESSED  
C @RESET\_FLAG DW ? : WORD=1234H IF KEYBOARD RESET UNDERWAY  
C  
C -----  
C | FIXED DISK DATA AREAS :  
C -----  
C  
C @DISK\_STATUS1 DB ? : FIXED DISK STATUS  
C @DISK\_STATUS1 DB ? : COUNT OF USED DISK DRIVES  
C @DISK\_STATUS1 DB ? : HEAD CONTROL BYTE  
C @PORT\_OFF DB ? : RESERVED (PORT OFFSET)

```

280 PAGE
281 ;-----[ TIME-OUT VARIABLES ]-----;
282
283
284 *PRINT_TIM_OUT DB ? ; TIME OUT COUNTERS FOR PRINTER RESPONSE
285 DB ? ; SECOND LOGICAL PRINTER ADAPTER
286 DB ? ; THIRD LOGICAL PRINTER ADAPTER
287 DB ? ; RESERVED
288 DB ? ; RESERVED
289 *RTC323_TIM_OUT DB ? ; TIME OUT COUNTERS FOR RS232 RESPONSE
290 DB ? ; SECOND LOGICAL RS232 ADAPTER
291 DB ? ; RESERVED
292 DB ? ; RESERVED
293
294 ;-----[ ADDITIONAL KEYBOARD DATA AREA ]-----;
295
296
297
298
299 *BUFFER_START DW ? ; BUFFER LOCATION WITHIN SEGMENT 40H
300 *BUFFER_END DW ? ; OFFSET OF KEYBOARD BUFFER START
301
302 ;-----[ EGA/PGA DISPLAY WORK AREA ]-----;
303
304
305
306 *ROWS DB ? ; ROWS ON THE ACTIVE SCREEN (LESS 1)
307 *POINTS DW ? ; BYTES PER CHARACTER
308 *INFO DB ? ; MODE OPTIONS
309 *INFO_3 DB ? ; FEATURE BIT SWITCHES
310 *INFO_4 DB ? ; RESERVED FOR DISPLAY ADAPTERS
311 *INFOA DB ? ; RESERVED FOR DISPLAY ADAPTERS
312
313
314 ;-----[ ADDITIONAL MEDIA DATA ]-----;
315
316
317 *LASTRATE DB ? ; LAST DISKETTE DATA RATE SELECTED
318 *HF_STATUS DB ? ; STATUS REGISTER
319 *HF_ERROR DB ? ; ERROR REGISTER
320 *HF_INT_FLAG DB ? ; FIXED DISK INTERRUPT FLAG
321 *HF_CNTL DB ? ; FIXED DISK/DISKETTE CARD BIT 0=1
322 *DSK_STATE DB ? ; DRIVE 0 MEDIA STATE
323 *DSK0 DB ? ; DRIVE 0 MEDIA STATE
324 *DSK1 DB ? ; DRIVE 0 OPERATION START STATE
325 *DSK2 DB ? ; DRIVE 0 OPERATION START STATE
326 *DSK3 DB ? ; DRIVE 0 PRESENT CYLINDER
327 *DSK4 DB ? ; DRIVE 1 PRESENT CYLINDER
328
329
330 ;-----[ ADDITIONAL KEYBOARD FLAGS ]-----;
331
332
333 *KB_FLAG_3 DB ? ; KEYBOARD MODE STATE AND TYPE FLAGS
334 *KB_FLAG_2 DB ? ; KEYBOARD LED FLAGS
335
336
337 ;-----[ REAL TIME CLOCK DATA AREA ]-----;
338
339
340 *USER_FLAG DW ? ; OFFSET ADDRESS OF USERS WAIT FLAG
341 *USER_FLAG_SEG DW ? ; SEGMENT ADDRESS OF USER WAIT FLAG
342 *RTC_LOW DW ? ; LOW WORD OF USER WAIT FLAG
343 *RTC_HIGH DW ? ; HIGH WORD OF USER WAIT FLAG
344 *RTC_WAIT_FLAG DB ? ; WAIT ACTIVE FLAG (01=BUSY, 80=POSTED)
345 ; (00=POST ACKNOWLEDGED)
346
347
348 ;-----[ AREA FOR NETWORK ADAPTER ]-----;
349
350 *NET DB ? DUP(?) ; RESERVED FOR NETWORK ADAPTERS
351
352
353
354
355 ;-----[ EGA/PGA PALETTE POINTER ]-----;
356
357
358 *SAVE_PTR DD ? ; POINTER TO EGA PARAMETER CONTROL BLOCK
359
360 ;-----[ DATA AREA - PRINT SCREEN ]-----;
361
362
363
364
365 0100 ORG 100H ; ADDRESS= 0040:0100 (REF 0050:0000)
366
367 *STATUS_BYTE DB ? ; PRINT SCREEN STATUS BYTE
368 ; 00=READY/OK, 01=BUSY, FF=ERROR
369
370 0101 DATA ENDS ; END OF BIOS DATA SEGMENT
371
372 .LIST

```

```

373
374
375
376
377
378
379 = 00FC
380 = 0000
381 = 0001
382 = F8A7
383 = F9FD
384
385
386 = 0060
387 = 0011
388 = 0073
389 = 000C
390 = 00C0
391 = 0001
392 = 0002
393 = 0010
394 = 0000
395 = 0040
396 = 0080
397 = 0064
398 = 0001
399 = 0002
400 = 0044
401 = 0008
402 = 0010
403 = 0020
404 = 0040
405 = 0080
406
407
408 = 0008
409 = 0010
410 = 0020
411 = 0040
412 = 0080
413
414
415 = 0060
416 = 00AA
417 = 00AB
418 = 00D0
419 = 00AE
420 = 00C0
421 = 00D0
422 = 00DF
423 = 00E0
424 = 00FE
425 = 0001
426
427
428 = 00ED
429 = 00F2
430 = 00F4
431
432
433 = 00AA
434 = 00FA
435 = 00FE
436 = 00FF
437
438
439 = 0001
440 = 0002
441 = 0004
442 = 0006
443 = 0010
444 = 0020
445 = 0040
446 = 0080
447
448
449 = 0004
450 = 0008
451 = 0010
452 = 0020
453 = 0040
454 = 0080
455
456
457 = 0007
458
459 = 0010
460 = 0020
461 = 0040
462 = 0080
463
464
465 = 0001
466 = 0002
467 = 0004
468
469 = 0020
470 = 0040
471 = 0080
472
473
474 = 00AB
475 = 0041
476 = 0038
477 = 001D
478 = 003A
479 = 0033
480 = 0052
481 = 002A
482 = 0045
483 = 0036
484 = 0046
485 = 0054

PAGE
C INCLUDE POSTEQU.INC
;----- EQUATES USED BY POST AND BIOS : ;-----


C MODEL_BYTE EQU 0FCH ; SYSTEM MODEL BYTE
C SUB_MODEL_BYTE EQU 000H ; SYSTEM SUB-MODEL TYPE
C BIOS_LEVEL EQU 001H ; BIOS REVISION LEVEL
C RATE_UPPER EQU 0F8A7H ; 0F952H +10%
C RATE_LOWER EQU 0F9FDH ; 0F952H -10%
C

C ;----- 8042 KEYBOARD INTERFACE AND DIAGNOSTIC CONTROL REGISTERS -----
C PORT_A EQU 060H ; 8042 KEYBOARD SCAN CODE/CONTROL PORT
C PORT_B EQU 061H ; PORT B READ/WRITE DIAGNOSTIC REGISTER
C RAM_PAR_ON EQU 10000011B ; AND MASK FOR PARITY CHECKING ENABLE ON
C RAM_PAR_OFF EQU 00000000B ; I/O MASK FOR PARITY CHECKING ENABLE OFF
C PARTTY_ERR EQU 11000000B ; R/W MEMORY - I/O CHANNEL PARITY ERROR
C GATE2 EQU 00000001B ; TIMER 2 INPUT GATE CLOCK BIT
C SPK2 EQU 00000010B ; SPEAKER OUTPUT DATA ENABLE BIT
C
C OUT_BUF EQU 00000000B ; REFRESH TEST BIT
C OUT_BUF_FULL EQU 064H ; SPEAKER TIMER OUT2 INPUT BIT
C INPT_BUF_FULL EQU 00000010B ; I/O MEMCHECK OCCURRED BIT MASK
C
C PARITY_CHECK EQU 10000000B ; MEMORY PARITY CHECK OCCURRED BIT MASK
C STATUS_PORT EQU 00000000B ; 8042 STATUS PORT
C
C OUT_BUF_FULL = +OUTPUT BUFFER FULL
C INPT_BUF_FULL = +INPUT BUFFER FULL
C PARITY_CHECK = -SYSTEM FLAG -POST-/SELF TEST
C STATUS_PORT = 3 = +OUTPUT BUFFER FULL
C OUT_BUF = 4 = +KEYBOARD INHIBITED
C INPT_BUF = 5 = +TRANSMIT TIMEOUT
C PARITY_CHECK = 6 = +RECEIVE TIME OUT
C STATUS_PORT = 7 = +PARITY IS EVEN

C ;----- 8042 INPUT PORT BIT DEFINITION SAVED IN #MF4.TST -----
C BASE_MEMB EQU 0000000B ; BASE PLANAR R/W MEMORY EXTENSION 640/X
C BASE_MEM EQU 0001000B ; BASE PLANAR R/W MEMORY SIZE 256/512
C MFG_LOOP EQU 0010000B ; LOOP POST JUMPER BIT FOR MANUFACTURING
C DSP_JMP EQU 01000000B ; DISPLAY TYPE SWITCH JUMPER BIT
C KEY_BO_INHIB EQU 10000000B ; KEYBOARD INHIBIT SWITCH BIT

C ;----- 8042 COMMANDS -----
C WRITE_8042_LOC EQU 060H ; WRITE 8042 COMMAND BYTE
C SELF_TEST EQU 0AAH ; 8042 SELF TEST
C INTR_FACE_CK EQU 0ABH ; CHECK 8042 INTERFACE COMMAND
C DIS_RKEY EQU 0ADH ; DISABLE KEYBOARD COMMAND
C ENR_RKEY EQU 0CH ; ENABLE KEYBOARD COMMAND
C READ_8042_INPUT EQU 0C0H ; READ 8042 INPUT PORT
C DISABLE_BIT20 EQU 0DDH ; DISABLE ADDRESS LINE BIT 20
C ENABLE_BIT20 EQU 0DFH ; ENABLE ADDRESS LINE BIT 20
C KYBD_CLK_DATA EQU 0E0H ; GET KEYBOARD CLOCK AND DATA COMMAND
C SHUT_CMD EQU 0FEH ; CAUSE A SHUTDOWN COMMAND
C KYBD_CLK EQU 011H ; KEYBOARD CLOCK BIT 0

C ;----- KEYBOARD/LED COMMANDS -----
C LED_CMD EQU 0EDH ; LED WRITE COMMAND
C KB_READ_ID EQU 0F2H ; READ KEYBOARD ID COMMAND
C KB_ENABLE EQU 0F4H ; KEYBOARD ENABLE

C ;----- 8042 KEYBOARD RESPONSE -----
C KB_OK EQU 0AAH ; RESPONSE FROM SELF DIAGNOSTIC
C KB_ACK EQU 0FAH ; ACKNOWLEDGE FROM TRANSMISSION
C KB_RESEND EQU 0FEH ; RESEND REQUEST
C KB_OVER_RUN EQU 0FFF ; OVER RUN SCAN CODE

C ;----- FLAG EQUATES WITHIN *KB_FLAG -----
C RIGHT_SHIFT EQU 00000001B ; RIGHT SHIFT KEY DEPRESSED
C LEFT_SHIFT EQU 00000010B ; LEFT SHIFT KEY DEPRESSED
C CTL_SHIFT EQU 00000100B ; CONTROL SHIFT KEY DEPRESSED
C ALT_SHIFT EQU 00001000B ; ALTERNATE SHIFT KEY DEPRESSED
C SCRLOCK_STATE EQU 00010000B ; SCRLOCK STATE HAS BEEN TOGGLED
C NUM_STATE EQU 00100000B ; NUM LOCK STATE HAS BEEN TOGGLED
C CAPS_STATE EQU 01000000B ; CAPS LOCK STATE HAS BEEN TOGGLED
C INS_STATE EQU 10000000B ; INSERT STATE IS ACTIVE

C ;----- FLAG EQUATES WITHIN *KB_FLAG_1 -----
C SYS_SHIFT EQU 00000000B ; SYSTEM KEY DEPRESSED AND HELD
C HOLD_STATE EQU 0000000B ; SUSPENDED KEY HAS BEEN TOGGLED
C SCROLL_SHIFT EQU 0000000B ; SCROLL LOCK KEY IS DEPRESSED
C NUM_SHIFT EQU 00100000B ; NUM LOCK KEY IS DEPRESSED
C CAPS_SHIFT EQU 01000000B ; CAPS LOCK KEY IS DEPRESSED
C INS_SHIFT EQU 10000000B ; INSERT KEY IS DEPRESSED

C ;----- FLAGS EQUATES WITHIN *KB_FLAG_2 -----
C KB_LEDS EQU 00000011B ; KEYBOARD LED STATE BITS
C ;----- RESERVED (MUST BE ZERO)
C KB_FA EQU 0000000B ; RESERVED (MUST BE ZERO)
C KB_FE EQU 0000000B ; ACKNOWLEDGMENT RECEIVED
C KB_PR_LED EQU 01000000B ; RESEND RECEIVED FLAG
C KB_ERR EQU 10000000B ; MODE INDICATOR UPDATE
C

C ;----- FLAGS EQUATES WITHIN *KB_FLAG_3 -----
C KBX EQU 00000001B ; KBX INSTALLED
C LC_HC EQU 00000010B ; LAST SCAN CODED WAS A HIDDEN CODE
C GRAPH_ON EQU 0000000B ; ALL GRAPHICS KEY DOWN (W.T. ONLY)
C ;----- RESERVED (MUST BE ZERO)
C SET_NUM_LK EQU 00100000B ; FORCE NUM LOCK IF READ ID AND KBX
C LC_AB EQU 01000000B ; LAST CHARACTER WAS FIRST ID CHARACTER
C RD_ID EQU 10000000B ; DOING A READ ID (MUST BE BIT0)

C ;----- KEYBOARD SCAN CODES -----
C ID_1 EQU 0ABH ; 1ST ID CHARACTER FOR KBX
C ID_2 EQU 041H ; 2ND ID CHARACTER FOR KBX
C ALT_KEY EQU 56 ; SCAN CODE FOR ALTERNATE SHIFT KEY
C CTL_KEY EQU 29 ; SCAN CODE FOR CONTROL KEY
C CAPS_KEY EQU 58 ; SCAN CODE FOR SHIFT LOCK KEY
C CAPS_DTC EQU 57 ; SCAN CODE FOR DELETED KEY
C INS_KEY EQU 82 ; SCAN CODE FOR INSERT KEY
C LEFT_KEY EQU 42 ; SCAN CODE FOR LEFT SHIFT
C NUM_KEY EQU 69 ; SCAN CODE FOR NUMBER LOCK KEY
C RIGHT_KEY EQU 54 ; SCAN CODE FOR RIGHT SHIFT
C SCROLL_KEY EQU 70 ; SCAN CODE FOR SCROLL LOCK KEY
C SYS_KEY EQU 84 ; SCAN CODE FOR SYSTEM KEY

```

```

486
487
488
489 = 0070
490 = 0071
492 = 0080
493
494
495
496 = 0000
497 = 0001
498 = 0002
499 = 0003
500 = 0004
501 = 0005
502 = 0006
503 = 0007
504 = 0008
505 = 0009
506 = 000A
507 = 000B
508 = 000C
509 = 000D
510 = 000E
511 = 000F
512 = 0010
513 = 0011
514 = 0012
515
516 = 0014
517 = 0015
518 = 0016
519 = 0017
520 = 0018
521 = 0019
522 = 001A
523
524 = 002E
525 = 002F
526 = 0030
527 = 0031
528 = 0032
529 = 0033
530
531
532
533 = 0004
534 = 0008
535 = 0010
536 = 0020
537 = 0040
538 = 0080
539
540
541 = 0080
542
543
544
545
546 = 0001
547 = 0080
548 = 0080
549 = 0010
550 = 0010
551 = 0004
552 = 0030
553 = 000A
554 = 0002
555 = 0000
556 = 0014
557 = 0025
558
559 = 0080
560 = 0040
561 = 0020
562 = 0010
563 = 0010
564 = 0009
565 = 0008
566 = 0006
567 = 0004
568 = 0002
569 = 0002
570 = 0001
571
572 = 0001
573 = 0002
574
575
576
577 = 0001
578 = 0002
579 = 0003
580 = 0010
581 = 0020
582 = 00C0
583 = 0000
584 = 0040
585 = 0080
586 = 000C
587 = 00C0
588
589 = 0000
590 = 0000
591 = 0002
592 = 0002
593 = 0007

C PAGE
C ; CMOS EQUATES FOR THIS SYSTEM :
C
C ; CMOS TABLE LOCATION ADDRESS'S ##
C
C CMOS_SECONDS EQU 070H ; SECONDS
C CMOS_SEC_ALARM EQU 001H ; SECONDS ALARM ## NOTE: ALL LOCATIONS
C CMOS_MINUTES EQU 002H ; MINUTES IN THE CMOS AREA
C CMOS_MIN_ALARM EQU 003H ; MINUTES ALARM ARE IBM USE ONLY
C CMOS_HOURS EQU 004H ; HOURS AND SUBJECT TO
C CMOS_HOUR_ALARM EQU 005H ; HOURS ALARM CHANGE ON THE
C CMOS_DAY_WEEK EQU 006H ; DAY OF THE WEEK POST & LOGIC
C CMOS_DAY_MONTH EQU 007H ; DAY OF THE MONTH SHOULD DIRECTLY
C CMOS_MONTH EQU 008H ; MONTH ACCESS LOCATIONS
C CMOS_YEAR EQU 009H ; YEAR (TWO DIGITS) IN CMOS STORAGE.
C CMOS_REG_A EQU 00AH ; STATUS REGISTER A
C CMOS_REG_B EQU 00BH ; STATUS REGISTER B ALARM
C CMOS_REG_C EQU 00CH ; STATUS REGISTER C FLAGS
C CMOS_REG_D EQU 00DH ; STATUS REGISTER D BATTERY
C CMOS_DIAG EQU 00EH ; POST DIAGNOSTIC STATUS RESULTS BYTE
C CMOS_SHUT_DOWN EQU 00FH ; SHUTDOWN STATUS COMMAND BYTE
C CMOS_DISKETTE EQU 010H ; DISKETTE DRIVE TYPE BYTE
C
C ; RESERVED
C CMOS_DISK EQU 011H ; RESERVED
C
C ; FIXED DISK TYPE BYTE
C CMOS_EQUIP EQU 014H ; EQUIPMENT WORD LOW BYTE
C CMOS_B_M_S_10 EQU 015H ; BASE MEMORY SIZE - LOW BYTE (X1024)
C CMOS_B_M_S_HI EQU 016H ; BASE MEMORY SIZE - HIGH BYTE
C CMOS_E_M_S_10 EQU 017H ; EXPANSION MEMORY SIZE - LOW BYTE
C CMOS_E_M_S_HI EQU 018H ; EXPANSION MEMORY SIZE - HIGH BYTE
C CMOS_DTSR_T EQU 019H ; FIXED DISK TYPE - DRIVE C EXTENSION
C CMOS_DISK_2 EQU 01AH ; FIXED DISK TYPE - DRIVE D EXTENSION
C
C ; 1B0 THROUGH 2D0 - RESERVED
C CMOS_CKSUM_HI EQU 02EH ; CMOS CHECKSUM - HIGH BYTE
C CMOS_CKSUM_LO EQU 02FH ; CMOS CHECKSUM - LOW BYTE
C CMOS_U_M_S_10 EQU 030H ; USABLE MEMORY ABOVE 1 MEG - LOW BYTE
C CMOS_U_M_S_HI EQU 031H ; USABLE MEMORY ABOVE 1 MEG - HIGH BYTE
C CMOS_CENTURY EQU 032H ; DATE CENTURY BYTE (BCD)
C CMOS_INFO128 EQU 033H ; 128K INFORMATION STATUS FLAG BYTE
C
C ; 340 THROUGH 3FH - RESERVED
C
C ;----- CMOS DIAGNOSTIC STATUS ERROR FLAGS WITHIN CMOS DIAG
C CMOS_CLK_FAIL EQU 00000100B ; CMOS CLOCK NOT UPDATING OR NOT VALID
C HF_FAIL EQU 00001000B ; FIXED DISK FAILURE ON INITIALIZATION
C W_MEM_SIZE EQU 00010000B ; MEMORY SIZE NOT EQUAL TO CONFIGURATION
C BAD_CONFIG EQU 00100000B ; MINIMUM CONFIG USED INSTEAD OF CMOS
C BAD_CKSUM EQU 01000000B ; CHECKSUM ERROR
C BAD_BAT EQU 10000000B ; DEAD BATTERY - CMOS LOST POWER
C
C ;----- CMOS INFORMATION FLAGS -----
C M640K EQU 10000000B ; 512K -> 640K OPTION INSTALLED (128K)
C
C ;----- DISKETTE EQUATES -----
C DUAL EQU 00000001B ; MASK FOR COMBO/DSP ADAPTER
C INT_FLAG EQU 10000000B ; INTERRUPT OCCURRENCE FLAG
C DSK_CHG EQU 10000000B ; DISKETTE CHANGE FLAG MASK BIT
C DETERMINED EQU 00000000B ; SET STATE DETERMINED IN STATE BITS
C DCOM EQU 00000000B ; TRACK MASK
C SENSE_DRY_ST EQU 00000100B ; SENSE DRIVE STATUS COMMAND
C TRK_SLAP EQU 030H ; CRASH STOP (48 TPI DRIVES)
C QUIET_SEEK EQU 00AH ; SEEK TO TRACK 10
C MAX_DRY EQU 2 ; MAX NUMBER OF DRIVES
C HD1024_SETTLE EQU 15 ; 1.2 M HEAD SETTLE TIME
C HD320_SETTLE EQU 20 ; 320 K HEAD SETTLE TIME
C MOTOR_WAIT EQU 37 ; 2 SECONDS OF COUNTS FOR MOTOR TURN OFF
C
C ;----- DISKETTE ERRORS -----
C TIME_OUT EQU 080H ; ATTACHMENT FAILED TO RESPOND
C BAD_SEEK EQU 040H ; SEEK OPERATION FAILED
C BAD_SETTLE EQU 020H ; HEAD SETTLE TIME OUT
C BAD_CRC EQU 010H ; BAD CRC ON DISKETTE READ
C DMA_BOUNDARY EQU 009H ; ATTEMPT TO DMA ACROSS 64K BOUNDARY
C BAD_DMA EQU 008H ; DMA OVERRUN ON OPERATION
C MEDIA_CHANGE EQU 006H ; MEDIA REMOVED ON DUAL ATTACH CARD
C RECORD_NOT_FND EQU 004H ; REQUESTED SECTOR NOT FOUND
C WRITE_PROTECT EQU 003H ; WRITE ATTEMPTED OR WRITE PROTECT DISK
C BAD_ADDR_MARK EQU 002H ; ADDRESS MARK NOT FOUND
C BAD_CMD EQU 001H ; BAD COMMAND PASSED TO DISKETTE I/O
C
C ;----- DISK CHANGE LINE EQUATES -----
C NOCHGLN EQU 001H ; NO DISK CHANGE LINE AVAILABLE
C CHGLN EQU 002H ; DISK CHANGE LINE AVAILABLE
C
C ;----- MEDIA/DRIVE STATE INDICATORS -----
C TRK_CAPA EQU 00000001B ; 80 TRACK CAPABILITY
C FMT_CAPA EQU 00000010B ; MULTIPLE FORMAT CAPABILITY (1.2M)
C DRV_DET EQU 00000100B ; DRIVE DETERMINED
C DED_DET EQU 01000000B ; MEDIA DETERMINED BIT
C DTSR_STEP EQU 00100000B ; DOUBLE STEP BIT
C RATE_MSK EQU 11000000B ; MASK FOR CLEARING ALL BUT RATE
C RATE_500 EQU 00000000B ; 500 KBS DATA RATE
C RATE_300 EQU 01000000B ; 300 KBS DATA RATE
C RATE_250 EQU 10000000B ; 250 KBS DATA RATE
C STRG_MSK EQU 00001100B ; OPERATION START RATE MASK
C SEND_MSK EQU 11000000B ; MASK FOR SEND RATE BITS
C
C ;----- MEDIA/DRIVE STATE INDICATORS COMPATIBILITY -----
C M3D3U EQU 00000000B ; 360 MEDIA/DRIVE NOT ESTABLISHED
C M3D1U EQU 00000001B ; 360 MEDIA, 1.2DRIVE NOT ESTABLISHED
C M3D1U EQU 00000010B ; 1.2 MEDIA/DRIVE NOT ESTABLISHED
C MED_UNK EQU 00000111B ; NONE OF THE ABOVE

```

594 C PAGE  
595 C ;----- INTERRUPT EQUATES -----  
596 E01 EQU 020H ; END OF INTERRUPT COMMAND TO 8259  
597 INTA00 EQU 020H ; 8259 PORT  
598 INTA01 EQU 021H ; 8259 PORT  
599 INTB00 EQU 0A0H ; 2ND 8259  
600 INTB01 EQU 0A1H ;  
601 INT\_TYPE EQU 070H ; START OF 8259 INTERRUPT TABLE LOCATION  
602 INT\_VIDEO EQU 010H ; VIDEO VECTOR  
603 ;-----  
604 DMA08 EQU 008H ; DMA STATUS REGISTER PORT ADDRESS  
605 DMA10 EQU 009H ; DMA CH.0 ADDRESS REGISTER PORT ADDRESS  
606 DMA12 EQU 00AH ; 2ND DMA STATUS PORT ADDRESS  
607 DMA14 EQU 00BH ; 2ND DMA CH.0 ADDRESS REGISTER ADDRESS  
608 ;-----  
609 TIMER EQU 040H ; 8254 TIMER - BASE ADDRESS  
610 ;-----  
611 MFG\_PORT EQU 80H ; MANUFACTURING AND POST CHECKPOINT PORT  
612 ; DMA CHANNEL 0 PAGE REGISTER ADDRESS  
614 ;-----  
615 ;----- MANUFACTURING BIT DEFINITION FOR OMFG\_ERR\_FLAG:1 -----  
616 MEM\_FAIL EQU 00000001B ; STORAGE TEST FAILED (ERROR 20X)  
617 PRO\_FAIL EQU 00000010B ; VIRTUAL MODE TEST FAILED (ERROR 104)  
618 LSI\_FAIL EQU 00000020B ; LOW SING CHIP SELECT FAILED (ERROR 09)  
619 KYCLR\_FAIL EQU 00001000B ; KEYBOARD CLEAR TEST FAILED (ERROR 309)  
620 KY\_SYS\_FAIL EQU 00010000B ; KEYBOARD OR SYSTEM FAILED (ERROR 303)  
621 KYBD\_FAIL EQU 00100000B ; KEYBOARD FAILED (ERROR 301)  
622 DSK\_FAIL EQU 01000000B ; DISKETTE TEST FAILED (ERROR 601)  
623 KEY\_FAIL EQU 10000000B ; KEYBOARD LOCKED (ERROR 302)  
624 ;-----  
625 ;-----  
626 DMA\_PAGE EQU 081H ; START OF DMA PAGE REGISTERS  
627 LAST\_DMA\_PAGE EQU 08FH ; LAST DMA PAGE REGISTER  
628 ;-----  
629 ;-----  
630 X287 EQU 0F0H ; MATH COPROCESSOR CONTROL PORT  
631 ;-----  
632 ;-----  
633 POST\_SS EQU 00000H ; POST STACK SEGMENT  
634 POST\_SP EQU 08000H ; POST STACK POINTER  
635 ;-----  
636 ;-----  
637 CR EQU 00DH ; CARRIAGE RETURN CHARACTER  
638 LF EQU 00AH ; LINE FEED CHARACTER  
639 RVRT EQU 00001000B ; VIDEO VERTICAL RETRACE BIT  
640 RHRZ EQU 00000001B ; VIDEO HORIZONTAL RETRACE BIT  
641 H EQU 256 ; HIGH BYTE FACTOR (X 100H)  
642 X EQU H+1 ; HIGH AND LOW BYTE FACTOR (X 101H)  
643 ;-----  
644 .LIST

```

645 PAGE
646 C INCLUDE SYSDATA.INC
647 C ;----- PROTECTED MODE EQUATES FOR POST TESTS AND BIOS ROUTINES :----- C
648 C ;----- LENGTH EQUATES FOR PROTECTED MODE TESTS :----- C
649 C SDA_LEN EQU 00300H ; SYSTEM DATA AREA LENGTH
650 C SYS_IDT_LEN EQU 256*8 ; 256 SYSTEM IDT ENTRIES, 8 BYTES EACH
651 C GDT_LEN EQU TYPE_GDT_DEF ; GDT STRUCTURE LENGTH
652 C DESC_LEN EQU TYPE_DATA_DESC ; LENGTH OF A DESCRIPTOR
653 C MCRTR_SIZE EQU 4*1024 ; MONOCHROME CRT SIZE
654 C CCRTR_SIZE EQU 4*1024 ; COLOR/COMPATIBLE CRT SIZE
655 C ECCRTR_SIZE EQU 0FFFFH ; SIZE OF EACH PORTION OF THE ENHANCED
656 C MAX_SEG_LEN EQU 0FFFFH ; MAXIMUM SEGMENT LENGTH = 64K
657 C NULL_SEG_LEN EQU 00000H ; NULL SEGMENT LENGTH = 0
658 C ;----- LOCATION EQUATES FOR PROTECTED MODE TESTS :----- C
659 C SYS_IDT_LOC EQU 000A0H ; THE SYSTEM IDT IS AT THE BOTTOM
660 C SDA_LOC EQU 00400H ; SAME AS REAL
661 C GDT_LOC EQU (SYS_IDT_LOC + SYS_IDT_LEN) ; (SYS_IDT_LOC + SYS_IDT_LEN)
662 C MCRTR_HI EQU 0000H ; MONOCHROME CRT ADDRESS
663 C CCRTR_HI EQU 0000H ; COMPATIBLE COLOR CRT ADDRESS
664 C ECCRTR_HI EQU 0000H ; (OB0000H)
665 C CSEGO_HI EQU 0000H ; (OB0000H) SEGMENT POST/BIOS
666 C NSEGO_HI EQU 0000H ; (OB0000H) FOR TESTS
667 C NSEGO_HI EQU 00H ; ABS0
668 C ;----- DEFINITIONS FOR ACCESS RIGHTS BYTES :----- C
669 C CPL3_DATA_ACCESS EQU 11110011B ; PRESENT
670 C ;----- THE GLOBAL DESCRIPTOR TABLE DEFINITION FOR POWER ON SELF TESTS :----- C
671 C ;----- GDT_DEF STRUCTURE :----- C
672 C GDT_DEF STRUCT
673 C     DQ ? ; UNUSED ENTRY
674 C     DQ ? ; THIS ENTRY POINTS TO THIS TABLE
675 C     DQ ? ; POST INTERRUPT DESCRIPTOR TABLE
676 C     DQ ? ; THE REAL SYSTEM DATA FOR POST
677 C     DQ ? ; COMPATIBLE BW CRT FOR POST
678 C     DQ ? ; COMPATIBLE COLOR CRT FOR POST
679 C     DQ ? ; ENHANCED COLOR GRAPHICS CRT (16 BYTES)
680 C     DQ ? ; UNRESIDENTED
681 C     DQ ? ; CS - POST IDT, ROM RESIDENT
682 C     DQ ? ; DS - POST IDT, ROM RESIDENT
683 C     DQ ? ; DS - TEMP, DYNAMIC
684 C     DQ ? ; DS - TEMP, DYNAMIC
685 C     DQ ? ; DS - TEMP, DYNAMIC
686 C     DQ ? ; DS - TEMP, DYNAMIC
687 C     DQ ? ; DS - TEMP, DYNAMIC
688 C     DQ ? ; DS - TEMP, DYNAMIC
689 C     DQ ? ; DS - TEMP, DYNAMIC
690 C     DQ ? ; DS - TEMP, DYNAMIC
691 C     DQ ? ; DS - TEMP, DYNAMIC
692 C     DQ ? ; DS - TEMP, DYNAMIC
693 C     DQ ? ; DS - TEMP, DYNAMIC
694 C     DQ ? ; DS - TEMP, DYNAMIC
695 C     DQ ? ; DS - TEMP, DYNAMIC
696 C     DQ ? ; DS - TEMP, DYNAMIC
697 C     DQ ? ; DS - TEMP, DYNAMIC
698 C     DQ ? ; DS - TEMP, DYNAMIC
699 C     DQ ? ; DS - TEMP, DYNAMIC
700 C     DQ ? ; DS - TEMP, DYNAMIC
701 C     DQ ? ; DS - TEMP, DYNAMIC
702 C     DQ ? ; DS - TEMP, DYNAMIC
703 C     DQ ? ; DS - TEMP, DYNAMIC
704 C     DQ ? ; DS - TEMP, DYNAMIC
705 C     DQ ? ; DS - TEMP, DYNAMIC
706 C     DQ ? ; DS - TEMP, DYNAMIC
707 C     DQ ? ; DS - TEMP, DYNAMIC
708 C     DQ ? ; DS - TEMP, DYNAMIC
709 C     DQ ? ; DS - TEMP, DYNAMIC
710 C     DQ ? ; DS - TEMP, DYNAMIC
711 C     DQ ? ; DS - TEMP, DYNAMIC
712 C     DQ ? ; DS - TEMP, DYNAMIC
713 C     DQ ? ; DS - TEMP, DYNAMIC
714 C     DQ ? ; DS - TEMP, DYNAMIC
715 C     DQ ? ; DS - TEMP, DYNAMIC
716 C     DQ ? ; DS - TEMP, DYNAMIC
717 C     DQ ? ; DS - TEMP, DYNAMIC
718 C     DQ ? ; DS - TEMP, DYNAMIC
719 C     DQ ? ; DS - TEMP, DYNAMIC
720 C     DQ ? ; DS - TEMP, DYNAMIC
721 C     DQ ? ; DS - TEMP, DYNAMIC
722 C     DQ ? ; DS - TEMP, DYNAMIC
723 C     DQ ? ; DS - TEMP, DYNAMIC
724 C ;----- SEGMENT DESCRIPTOR TABLE ENTRY STRUCTURE :----- C
725 C DATA_DESC STRUCT
726 C     SEG_LIMIT DW ? ; SEGMENT LIMIT (1 - 65535 BYTES)
727 C     BASE_LO_WORD DW ? ; 24 BIT SEGMENT PHYSICAL
728 C     BASE_HI_BYTE DB ? ; ADDRESS (0 - (16M-1))
729 C     DATA_ACC_RIGHTS DB ? ; ACCESS RIGHTS BYTE
730 C     DATA_RESERVED DW ? ; RESERVED - MUST BE 0000 FOR THE 80286
731 C     DATA_DESC ENDS
732 C ;----- GATE DESCRIPTOR TABLE ENTRY STRUCTURE :----- C
733 C GATE_DESC STRUCT
734 C     ENTRY_POINT DW ? ; DESTINATION ROUTINE ENTRY POINT
735 C     CS_SELECTOR DW ? ; SELECTOR FOR DESTINATION SEGMENT
736 C     GATE_TYPE DB ? ; NUMBER OF WORDS TO COPY FROM STACK
737 C     GATE_ACC_RIGHTS DB ? ; ACCESS RIGHTS BYTE
738 C     GATE_RESERVED DW ? ; RESERVED - MUST BE 0000 FOR THE 80286
739 C     GATE_DESC ENDS
740 C .LIST

```

```

743 PAGE
744 0000 CODE SEGMENT WORD PUBLIC
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 0000 POSTI PROC NEAR
780
781 = 0000
782 BEGIN EQU $ .6480090COPY. IBM CORP. 1981,1985 ;COPYRIGHT NOTICE
783 0000 36 34 26 30 30 39
784 30 43 4F 50 52 2E
785 20 49 42 40 20 43
786 4F 52 50 2E 20 31
787 39 38 31 2C 31 39
788 38 35 20 20
789 EVEN
790 ; 6 4 8 0 0 9 0 C O P R . I B M 1 9 8 5 ;EVEN BOUNDARY
791 ; 6 4 8 0 0 9 1 C O P R . I B M 1 9 8 5 ;EVEN MODULE
792 DB .66448800009901 CCOOPPR. IBBMM 11998855 ;ODD MODULE
793
794
795
796
797
798
799 004E 20 DB * * ;PAD
800
801
802 ;----- INITIAL RELIABILITY TESTS -- (POSTI) -----;
803
804
805 ;----- TEST.01 -----;
806 ;----- 80286 PROCESSOR TEST (REAL MODE) -----;
807 ;----- DESCRIPTION -----;
808 ;----- VERIFY FLAGS, REGISTERS -----;
809 ;----- AND CONDITIONAL JUMPS. -----;
810
811
812
813
814
815 ASSUME DS:DATA
816 0050 START_1:
817 0050 FA CLI ; DISABLE INTERRUPTS
818 0051 B8 D58D MOV AX,0D500H+CMOS_REG_D+NMI ; FLAG MASK IN (AH) AND NMI MASK IN (AL)
819 0054 E6 70 OUT CMOS_PORT,AL ; DISABLE NMI INTERRUPTS
820 0057 73 27 SAHF ; SET THE "AF" FLAG ON
821 0059 75 25 JNC ERR02 ; GO TO ERROR ROUTINE IF "CF" NOT SET
822 005B 73 23 JNZ ERR02 ; GO TO ERROR ROUTINE IF "ZF" NOT SET
823 005D 79 21 JNP ERR02 ; GO TO ERROR ROUTINE IF "PF" NOT SET
824 005F 97 LAHF ; LOAD FLAG IMAGE TO (AH)
825 0060 D1 05 MOV CL,5 ; LOAD COUNT IN (CL) WITH SHIFT COUNT
826 0062 D2 EC SHR AH,CL ; SHIFT "AF" INTO CARRY BIT POSITION
827 0064 73 1A JNC ERR02 ; GO TO ERROR ROUTINE IF "AF" NOT SET
828 0066 B0 40 MOV AL,40H ; SET THE "OF" FLAG ON
829 0068 D0 E0 SHL AL,1 ; SETUP FOR TESTING
830 006A D0 14 JNO ERR02 ; GO TO ERROR ROUTINE IF "OF" NOT SET
831 006C 32 E4 XOR AH,AH ; SET (AH) = 0
832 006E 9E SAHF ; CLEAR "SF", "CF", "ZF", AND "PF"
833 006F 76 0F JBE ERR02 ; GO TO ERROR ROUTINE IF "CF" ON
834
835 0071 78 0D JS ERR02 ; GO TO ERROR ROUTINE IF "ZF" ON
836 0073 7A 0B JP ERR02 ; GO TO ERROR ROUTINE IF "SF" ON
837 0075 99 00 LAHF ; LOAD IMAGE (AH)
838 0076 D2 EC SHR AH,CL ; SHIFT "AF" INTO CARRY BIT POSITION
839 0078 72 06 JC ERR02 ; GO TO ERROR ROUTINE IF ON
840 007A D0 E4 SHL AH,1 ; CHECK THAT "OF" IS CLEAR
841 007C 70 02 JD ERR02 ; GO TO ERROR ROUTINE IF ON
842 007E 74 03 JZ CTA ; CONTINUE CONFIDENCE TESTS IF "ZF" SET
843
844 0080 F4 HLT ; ERROR HALT
845 0081 EB FD JMP ERR02 ; ERROR LOOP TRAP
846
847 0083 C7A: MOV AX,DATA ; SET DATA SEGMENT
848 0083 B8 ---- R MOV DS,AX ; INTO THE (DS) SEGMENT REGISTER
849
850
851 ;----- CHECK FOR PROCESSOR SHUTDOWN -----;
852
853 0088 E4 64 IN AL,STATUS_PORT ; READ CURRENT KEYBOARD PROCESSOR STATUS
854 008A B8 04 TEST AL,SYN_FLAG ; CHECK FOR SHUTDOWN IN PROCESS FLAG
855 008C 75 03 JNZ C7B ; GO TO YES
856 008E E9 0123 R JMP SHUTO ; ELSE CONTINUE NORMAL POWER ON CODE

```

```

857      PAGE
858      ;----- CHECK FOR SHUTDOWN 09
859      0091
860      00F8 B0 8F
861      0093 E5 00
862      0095 E5 00
863      0097 E4 71
864      0099 3C 09
865      009B 86 C4
866      009D 74 41
867
868
869      ;----- CHECK FOR SHUTDOWN 0A
870      009F 80 FC 0A
871      00A2 74 3C
872
873      00A4 2A C0
874      00A6 E6 F1
875
876
877      ;----- RE-INITIALIZE THE 8259 INTERRUPT #1 CONTROLLER CHIP :
878
879      00A8 B0 11
880      00AA E6 20
881      00AC EB 00
882      00AE B0 08
883      00B0 E6 21
884      00B2 EB 00
885      00B4 B0 04
886      00B6 E6 21
887      00B8 EB 00
888      00BA B0 01
889      00BC E6 21
890      00BE EB 00
891      00C0 B0 FF
892      00C2 E6 21
893
894
895      ;----- RE-INITIALIZE THE 8259 INTERRUPT #2 CONTROLLER CHIP :
896      00C4 B0 11
897      00C6 EB A0
898      00C8 E6 00
899      00CA B0 70
900      00CC E6 A1
901      00CE B0 02
902      00D0 EB 00
903      00D2 EB 00
904      00D4 EB 00
905      00D6 B0 01
906      00D8 E6 A1
907      00DA EB 00
908      00DC B0 FF
909      00DE E6 A1
910
911      ;----- SHUTDOWN - RESTART
912      ;----- RETURN CONTROL AFTER A SHUTDOWN COMMAND IS ISSUED
913      ;----- DESCRIPTION
914      ;----- A TEST IS MADE FOR THE SYSTEM FLAG BEING SET. IF THE SYSTEM FLAG IS
915      ;----- SET, THE SHUTDOWN BYTE IN CMOS IS USED TO DETERMINE WHERE CONTROL IS
916      ;----- RETURNED.
917
918      ;----- CMOS = 0 SOFT RESET OR UNEXPECTED SHUTDOWN
919      ;----- CMOS = 1 SHUT DOWN AFTER MEMORY SIZE
920      ;----- CMOS = 2 SHUT DOWN AFTER MEMORY TEST
921      ;----- CMOS = 3 SHUT DOWN WITH MEMORY ERROR
922      ;----- CMOS = 4 SHUT DOWN WITH MEMORY REQUEST
923      ;----- CMOS = 5 JMP DWORD REQUEST - (INTERRUPT CHIPS & 287 ARE INITIALIZED)
924      ;----- CMOS = 6 PROTECTED MODE TEST3 PASSED
925      ;----- CMOS = 7 PROTECTED MODE TEST3 FAILED
926      ;----- CMOS = 8 PROTECTED MODE TEST1 FAILED
927      ;----- CMOS = 9 BLOCK MOVE SHUTDOWN REQUEST
928      ;----- CMOS = A JMP DWORD REQUEST - (W/O INTERRUPT CHIPS INITIALIZED)
929
930      ;----- NOTES: RETURNS ARE MADE WITH INTERRUPTS AND NMI DISABLED.
931      ;----- USER MUST RESTORE SS:SP (POST DEFAULT SET = 0000:0400),
932      ;----- ENABLE NON-MASKABLE INTERRUPTS (NMI) WITH AN OUT TO
933      ;----- PORT TOH WITH HIGH ORDER BIT OFF, AND THEN ISSUE A
934      ;----- STI TO ENABLE INTERRUPTS. FOR SHUTDOWN 5) THE USER
935      ;----- MUST ALSO RESTORE THE INTERRUPT MASK REGISTERS.
936
937
938      ;----- CHECK FROM WHERE
939      00E0
940      00E0 B0 8F
941      00E2 E6 70
942      00E4 90
943      00E5 2A C0
944      00E7 E6 71
945      00E9 86 E0
946      00EB 3C 9A
947      00ED 17 44
948      00EF BE 0103 R
949      00F2 03 F0
950      00F4 03 F0
951      00F6 2E1 BB 1C
952
953
954      ;----- SET TEMPORARY STACK FOR POST
955      00F9 B8 ---- R
956      00FC 8E D0
957      00FE BC 0400 R
958      0101 FF E3
959
960      0103 0123 R
961      0105 0990 R
962      0107 0000 E
963      0109 0000 E
964      010B 0000 E
965      010D 0119 R
966      010F 0000 E
967      0111 0000 E
968      0113 0793 R
969      0115 0000 E
970      0117 011F R

```

```

971 PAGE
972 I----- *IO_ROM_INIT MUST BE INITIALIZED BY THE USER FOR VECTORED REQUESTS
973
974 0119
975 0119 E4 60
976 011B B0 20
977 011D E6 20
978 011F
979 011F FF 2E 0067 R
980
981
982 I----- CHECKPOINT 01
983
984 0123
985 0123 B0 01
986 0125 E6 80
987
988 I----- READ/WRITE/TEST THE 80286 REGISTERS WITH ONE'S AND ZERO'S
989
990 0127 B8 FFFF
991 0128 F9
992 012B 73 21
993 012D
C8:  MOV DS,AX ; WRITE PATTERN TO ALL REGISTERS
      MOV BX,DS
      MOV ES,BX
      MOV CX,ES
      MOV SS,CX
      MOV DS,SS
      MOV SP,DX
      MOV BP,SP
      MOV SI,BP
      MOV DI,SI
      JNC C9
      XOR AX,DI ; PATTERN MAKE IT THROUGH ALL REGISTERS
      JNZ C9 ; NO - GO TO ERROR ROUTINE
      CLC ; CLEAR CARRY FLAG
      JMP C8
      OR AX,DI ; TSTIA
      JZ C10A ; ZERO PATTERN MAKE IT THROUGH ?
      JES C10A ; YES - GO TO NEXT TEST
      ERROR1: HLT ; HALT SYSTEM
      C9:  MOV AX,0FFFFH ; SETUP ONE'S PATTERN IN (AX)
            STC ; SET CARRY FLAG
            JNC ERROR1 ; GO IF NO CARRY
      C10:  MOV AL,01H ; ADDRESS TO BOTH (AH) AND (AL)
            OUT CMOS_PORT,AL ; ADDRESS CMOS ALARM BYTE WITH NMII=OFF
            NOP ; I/O DELAY
            IN AL,CMOS_DATA ; GET THE CURRENT CONTROL REGISTER
            AND AL,00000011B ; CLEAR SET,PIE,AIE, AND SQWE BITS
            XCHG AL,AH ; SAVE IT
            OUT CMOS_PORT,AL
            XCHG AL,AH
            OUT CMOS_DATA,AL
      C10A:  MOV AL,0 ; ADDRESS CMOS FLAGS BYTE WITH NMII=OFF
            MOV DX,03D8H ; GET COLOR MODE CONTROL PORT ADDRESS
            OUT DX,AL ; DISABLE COLOR VIDEO
            INC AL ; MONOCHROME MODE RESET MASK
            MOV DL,0B8H ; GET ADDRESS OF MONOCHROME MODE CONTROL
            OUT DL,AL ; DISABLE B/WHITE/GRAY MODES
            MOV DL,0BAH ; ADDRESS MONOCHROME STATUS REGISTER
            IN AL,DX ; READ STATUS TO DISABLE EGA VIDEO
            MOV DL,0DAH ; ADDRESS OF COLOR MODE STATUS REGISTER
            IN AL,DX ; READ STATUS TO DISABLE EGA VIDEO
            MOV AL,0 ; SELECT ATTRIBUTE PALETTE REGISTER 0
            MOV DL,0C0H ; WRITE 0 TO ATTRIBUTE ADDRESS REGISTER
            OUT DL,AL ; TO DISABLE EGA VIDEO
            MOV AL,1111110B ; DISABLE PARITY CHECKERS
            OUT PORT_B,AL

      I----- RESET VIDEO
      C10B:  MOV AL,0 ; CLEAR DATA BYTE TO DISABLE VIDEO
            MOV DX,03D8H ; GET COLOR MODE CONTROL PORT ADDRESS
            OUT DX,AL ; DISABLE COLOR VIDEO
            INC AL ; MONOCHROME MODE RESET MASK
            MOV DL,0B8H ; GET ADDRESS OF MONOCHROME MODE CONTROL
            OUT DL,AL ; DISABLE B/WHITE/GRAY MODES
            MOV DL,0BAH ; ADDRESS MONOCHROME STATUS REGISTER
            IN AL,DX ; READ STATUS TO DISABLE EGA VIDEO
            MOV DL,0DAH ; ADDRESS OF COLOR MODE STATUS REGISTER
            IN AL,DX ; READ STATUS TO DISABLE EGA VIDEO
            MOV AL,0 ; SELECT ATTRIBUTE PALETTE REGISTER 0
            MOV DL,0C0H ; WRITE 0 TO ATTRIBUTE ADDRESS REGISTER
            OUT DL,AL ; TO DISABLE EGA VIDEO
            MOV AL,1111110B ; DISABLE PARITY CHECKERS
            OUT PORT_B,AL

      I----- TEST.02
      I----- ROM CHECKSUM TEST I
      I----- DESCRIPTION
      I----- A CHECKSUM IS DONE FOR THE 32K
      I----- READ ONLY MEMORY MODULES (TWO)
      I----- CONTAINING POST, BASIC AND BIOS.:
      I----- CHECKPOINT 02
      C11:  ASSUME SS:CODE ; SETUP SS SEGMENT REGISTER
            MOV AX,CS
            MOV SS,AX ; SET UP DATA SEGMENT TO POINT TO
            MOV DS,AX ; ROM ADDRESS START
            XOR SI,SI ; CLEAR CHECK REGISTER
            MOV BX,BX ; COUNT FOR 32K WORDS
            MOV CH,0B0H
            LODSW ; MOVE TWO BYTES INTO AX -- SI=SI+2
            ADD BL,AH ; ADD ODD BYTE AT DS:SI+1 TO CHECKSUM
            ADD DS,AX ; ADD EVEN BYTE AT DS:SI TO CHECKSUM
            LOOP C11 ; LOOP COUNT FOR 64K WORDS (32K WORDS)
            JNC C11E ; EXIT IF LOWEST RESET THE CARRY FLAG
            INVOKE MODEL_BYTE MUST NOT = ZERO)
            CONTINUE IF CHECKSUM VALID (ZERO)
            C11E: JZ C11A ; ELSE HALT IF CHECKSUM PROBLEM
            HLT ; VERIFY CMOS SHUTDOWN BYTE
      C11F: I----- TEST.03
      C120: I----- VERIFY CMOS SHUTDOWN BYTE
  
```

```

085
086
087
088
089
090
091 019F
092 019F B0 03
093 01A1 E6 80
094
095 01A3 B9 0009
096 01A6 B4 01
097 01A8 B0 8F
098 01AA E6 70
099 01AC B4 C4
100 01AE E6 71
101 01B0 B0 8F
102 01B2 B0 8F
103 01B3 E6 70
105 01B5 90
106 01B6 E4 71
107 01B8 3A C4
108 01B9 75 92
109 01BC D0 D4
110 01BE E2 E8
111
112
113
114
115
116
117
118
119
120 01C0 B8 ---- R
121 01C3 BE D8
122 01C5 B0 04
123 01C7 E6 80
124
125
126
127 01C9 E6 08
128 01CB E6 D0
129
130
131
132 01C0 B8 16 0072 R
133 01D1 B0 54
134 01D2 E6 43
135 01D3 B0 00
136 01D7 B0 C1
137 01D9 E6 41
138 01D8 B7 05
139 01D0
140 01D0 B0 40
141 01D0 E6 00
142 01E1 E6 43
143 01E3 80 FB FF
144 01E4 T4 00
145 01E5 E4 41
146 01E6 0A D8
147 01EC E2 EF
148 01F0 75 CB
149 01F0 T5 CB
150 01F2 F4
151
152
153
154
155
156
157
158
159
160
161
162 01F3 B0 05
163 01F5 E6 80
164
165 01F7 B8 C3
166 01F8 2B C9
167 01FB E6 41
168 01FD B1 05
169 01F9
170 01F7 B0 40
171 0201 E6 43
172 0203 EB 00
173 0205 00 00
174 0207 E4 41
175 0209 22 D8
176 0209 T4 07
177 020B E2 F0
178 020F FE CF
179 0211 T5 EC
180 0213 F4
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195 0214
196 0214 B8 ---- R
197 0217 BE D8
198 0219 90 04
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
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
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
788
789
789
790
791
792
793
794
795
796
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
888
889
889
890
891
892
893
894
895
896
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
949
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
979
979
980
981
982
983
984
985
986
987
988
989
989
990
991
992
993
994
995
996
997
998
999
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
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
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
12010
12011
12012
12013
12014
12015
12016
12017
12018
12019
12020
12021
12022
12023
12024
12025
12026
12027
12028
12029
12030
12031
12032
12033
12034
12035
12036
12037
12038
12039
12040
12041
12042
12043
12044
12045
12046
12047
12048
12049
120410
120411
120412
120413
120414
120415
120416
120417
120418
120419
120420
120421
120422
120423
120424
120425
120426
120427
120428
120429
120430
120431
120432
120433
120434
120435
120436
120437
120438
120439
120440
120441
120442
120443
120444
120445
120446
120447
120448
120449
120450
120451
120452
120453
120454
120455
120456
120457
120458
120459
120460
120461
120462
120463
120464
120465
120466
120467
120468
120469
120470
120471
120472
120473
120474
120475
120476
120477
120478
120479
120480
120481
120482
120483
120484
120485
120486
120487
120488
120489
120490
120491
120492
120493
120494
120495
120496
120497
120498
120499
1204100
1204101
1204102
1204103
1204104
1204105
1204106
1204107
1204108
1204109
1204110
1204111
1204112
1204113
1204114
1204115
1204116
1204117
1204118
1204119
1204120
1204121
1204122
1204123
1204124
1204125
1204126
1204127
1204128
1204129
1204130
1204131
1204132
1204133
1204134
1204135
1204136
1204137
1204138
1204139
1204140
1204141
1204142
1204143
1204144
1204145
1204146
1204147
1204148
1204149
1204150
1204151
1204152
1204153
1204154
1204155
1204156
1204157
1204158
1204159
1204160
1204161
1204162
1204163
1204164
1204165
1204166
1204167
1204168
1204169
1204170
1204171
1204172
1204173
1204174
1204175
1204176
1204177
1204178
1204179
1204180
1204181
1204182
1204183
1204184
1204185
1204186
1204187
1204188
1204189
1204190
1204191
1204192
1204193
1204194
1204195
1204196
1204197
1204198
1204199
1204200
1204201
1204202
1204203
1204204
1204205
1204206
1204207
1204208
1204209
1204210
1204211
1204212
1204213
1204214
1204215
1204216
1204217
1204218
1204219
1204220
1204221
1204222
1204223
1204224
1204225
1204226
1204227
1204228
1204229
12042210
12042211
12042212
12042213
12042214
12042215
12042216
12042217
12042218
12042219
12042220
12042221
12042222
12042223
12042224
12042225
12042226
12042227
12042228
12042229
12042230
12042231
12042232
12042233
12042234
12042235
12042236
12042237
12042238
12042239
12042240
12042241
12042242
12042243
12042244
12042245
12042246
12042247
12042248
12042249
12042250
12042251
12042252
12042253
12042254
12042255
12042256
12042257
12042258
12042259
12042260
12042261
12042262
12042263
12042264
12042265
12042266
12042267
12042268
12042269
12042270
12042271
12042272
12042273
12042274
12042275
12042276
12042277
12042278
12042279
12042280
12042281
12042282
12042283
12042284
12042285
12042286
12042287
12042288
12042289
12042290
12042291
12042292
12042293
12042294
12042295
12042296
12042297
12042298
12042299
120422100
120422101
120422102
120422103
120422104
120422105
120422106
120422107
120422108
120422109
120422110
120422111
120422112
120422113
120422114
120422115
120422116
120422117
120422118
120422119
120422120
120422121
120422122
120422123
120422124
120422125
120422126
120422127
120422128
120422129
120422130
120422131
120422132
120422133
120422134
120422135
120422136
120422137
120422138
120422139
120422140
120422141
120422142
120422143
120422144
120422145
120422146
120422147
120422148
120422149
120422150
120422151
120422152
120422153
120422154
120422155
120422156
120422157
120422158
120422159
120422160
120422161
120422162
120422163
120422164
120422165
120422166
120422167
120422168
120422169
120422170
120422171
120422172
120422173
120422174
120422175
120422176
120422177
120422178
120422179
120422180
120422181
120422182
120422183
120422184
120422185
120422186
120422187
120422188
120422189
120422190
120422191
120422192
120422193
120422194
120422195
120422196
120422197
120422198
120422199
120422200
120422201
120422202
120422203
120422204
120422205
120422206
120422207
120422208
120422209
120422210
120422211
120422212
120422213
120422214
120422215
120422216
120422217
120422218
120422219
120422220
120422221
120422222
120422223
120422224
120422225
120422226
120422227
120422228
120422229
120422230
120422231
120422232
120422233
120422234
120422235
120422236
120422237
120422238
120422239
120422240
120422241
120422242
120422243
120422244
120422245
120422246
120422247
120422248
120422249
120422250
120422251
120422252
120422253
120422254
120422255
120422256
120422257
120422258
120422259
120422260
120422261
120422262
120422263
120422264
120422265
120422266
120422267
120422268
120422269
120422270
120422271
120422272
120422273
120422274
120422275
120422276
120422277
120422278
120422279
120422280
120422281
120422282
120422283
120422284
120422285
120422286
120422287
120422288
120422289
120422290
120422291
120422292
120422293
120422294
120422295
120422296
120422297
120422298
120422299
120422300
120422301
120422302
120422303
120422304
120422305
120422306
120422307
120422308
120422309
120422310
120422311
120422312
120422313
120422314
120422315
120422316
120422317
120422318
120422319
120422320
120422321
120422322
120422323
120422324
120422325
120422326
120422327
120422328
120422329
120422330
120422331
120422332
120422333
120422334
120422335
120422336
120422337
120422338
120422339
120422340
120422341
120422342
120422343
120422344
120422345
120422346
120422347
120422348
120422349
120422350
120422351
120422352
120422353
120422354
120422355
120422356
120422357
120422358
120422359
120422360
120422361
120422362
120422363
120422364
120422365
120422366
120422367
120422368
120422369
120422370
120422371
120422372
120422373
120422374
120422375
120422376
120422377
120422378
120422379
120422380
120422381
120422382
120422383
120422384
120422385
120422386
120422387
120422388
120422389
120422390
120422391
120422392
120422393
120422394
120422395
120422396
120422397
120422398
120422399
120422400
120422401
120422402
120422403
120422404
120422405
120422406
120422407
120422408
120422409
120422410
120422411
120422412
120422413
120422414
120422415
120422416
120422417
120422418
120422419
120422420
120422421
120422422
120422423
120422424
120422425
120422426
120422427
120422428
120422429
120422430
120422431
120422432
120422433
120422434
120422435
120422436
120422437
120422438
120422439
120422440
120422441
120422442
120422443
120422444
120422445
120422446
120422447
120422448
120422449
120422450
120422451
120422452
120422453
120422454
120422455
120422456
120422457
120422458
120422459
120422460
120422461
120422462
120422463
120422464
120422465
120422466
120422467
120422468
120422469
120422470
120422471
120422472
120422473
120422474
120422475
120422476
120422477
120422478
120422479
120422480
120422481
120422482
120422483
120422484
120422485
120422486
120422487
120422488
120422489
120422490
120422491
120422492
120422493
120422494
120422495
120422496
120422497
120422498
120422499
120422500
120422501
120422502
120422503
120422504
120422505
120422506
120422507
120422508
120422509
12042
```

```

I199 021B E6 80          OUT   MFG_PORT_AL      ; <><> CHECKPOINT 06 <><>
I200 021D B9 16 0072 R   OUT   @RESET_FLAG,DX   ; RESTORE SOFT RESET FLAG
I201 0221 E6 0D          OUT   DMA+ODR,AL      ; SEND MASTER CLEAR TO DMA
I202
I203          ;----- WRAP DMA 0 CHANNEL ADDRESS AND COUNT REGISTERS
I204
I205 0223 B0 FF          MOV   AL,0FFH      ; WRITE PATTERN "FF" TO ALL REGISTERS
I206 0225 B8 D8          MOV   BL,AL       ; SAVE PATTERN FOR COMPARE
I207 0227 8A F8
I208 0229 B9 0008          MOV   CX,8       ; SETUP LOOP COUNT
I209 022C BA 0000          MOV   DX,DXA      ; SETUP I/O PORT ADDRESS OF REGISTER
I210 022F EE              OUT  DX,AL       ; WRITE PATTERN TO REGISTER, LSB
I211 0231 EE 00          JMP  $+2       ; I/O DELAY
I212 0232 EE 00          OUT  DX,AL       ; MSB OF 16 BIT REGISTER
I213 0233 B0 01          MOV   AL,01H      ; AL TO ANOTHER PATTERN BEFORE READ
I214 0235 EB 00          JMP  $+2       ; I/O DELAY
I215 0237 EC              IN   AL,DX       ; READ 16-BIT DMA CH REG, LSB 2ST DMA
I216 0238 EB 00          JMP  $+2       ; I/O DELAY
I217 023A B8 E0          MOV   AH,AL      ; SAVE LSB OF 16-BIT REGISTER
I218 023D E7 00          IN   AL,DX       ; READ MSB OF DMA CHANNEL REGISTER
I219 023D B8 D8          CMP   BX,AX      ; PATTERN READ AS WRITTEN?
I220 023F 74 01          JE   C18       ; YES - CHECK NEXT REGISTER
I221 0241 F4              HLT
I222 0242
I223 0242 42          INC   DX       ; NO - HALT THE SYSTEM
I224 0242 EE EA          LOOP  C17      ; NXT DMA CH
I225 0245 EE C0          INC   AL       ; SET I/O PORT TO NEXT CHANNEL REGISTER
I226 0247 74 DC          JZ   C16       ; WRITE PATTERN TO NEXT REGISTER
I227
I228          ;----- WRITE DMA WITH 55 PATTERN
I229
I230 0249 B0 55          CMP   BL,055H      ; CHECK IF "55" PATTERN DONE
I231 024C B0 09          JZ   C19       ; GO IF YES
I232 024E 80 FB AA          CMP   BL,0AAH      ; CHECK IF "AA" PATTERN DONE
I233 0251 74 08          JZ   C20       ; GO IF YES
I234 0253 B0 55          MOV   AL,055H      ; SAVE LSB OF 16-BIT REGISTER
I235 0255 EB CE          JMP   C16       ; READ MSB OF 16-BIT REGISTER
I236
I237
I238
I239 0257 B0 AA          C19:  MOV   AL,0AAH      ; SET I/O PORT AS WRITTEN?
I240 0259 EB CA          JMP   C16       ; PATTERN READ AS WRITTEN?
I241
I242          ;----- TEST.07
I243
I244 025D B0 07          ;----- 8237 DMA 1 INITIALIZATION
I245 025D E6 80          ;----- CHANNEL REGISTER TEST
I246 025F E6 DA          ;----- DESCRIPTION
I247 025F E6 DA          ;----- DISABLE 8237 DMA CONTROLLER 1.
I248 025F E6 DA          ;----- WRITE/READ THE CURRENT DMA 1
I249 025F E6 DA          ;----- ADDRESS AND WORD COUNT
I250 025F E6 DA          ;----- REGISTERS FOR ALL CHANNELS.
I251
I252
I253
I254          ;----- CHECKPOINT 07 - DMA 1
I255
I256 0261 B0 FF          C20:  MOV   AL,07H      ; <><><><><><><><><><>
I257 0263 B8 D8          OUT   MFG_PORT_AL      ; <><> CHECKPOINT 07 <><>
I258 0265 B8 F8          OUT   DMA1+0DH*2,AL  ; SEND MASTER CLEAR TO 2ND DMA
I259
I260          ;----- WRAP DMA 1 CHANNEL ADDRESS AND COUNT REGISTERS
I261
I262 0261 B0 FF          C16A:  MOV   AL,0FFH      ; WRITE PATTERN FF TO ALL REGISTERS
I263 0263 B8 D8          MOV   BH,AL       ; SAVE PATTERN FOR COMPARE
I264 0265 B8 F8
I265 0267 B9 0008          MOV   CX,8       ; SETUP LOOP COUNT
I266 026A BA 00C0          MOV   DX,DXA      ; SETUP I/O PORT ADDRESS OF REGISTER
I267 026E EE 00          C17A:  OUT  DX,AL       ; WRITE PATTERN TO REGISTER, LSB
I268 026E EE 00          JMP  $+2       ; I/O DELAY
I269 0271 B0 01          OUT  DX,AL       ; MSB OF 16 BIT REGISTER
I270 0273 EB 00          MOV   AL,01H      ; AL TO ANOTHER PAT BEFORE RD
I271 0275 EC              JMP  $+2       ; I/O DELAY
I272 0276 EB 00          IN   AL,DX       ; READ 16-BIT DMA CH REG, LSB 2ST DMA
I273 0278 B8 A0          MOV   AH,AL      ; I/O DELAY
I274 0278 B8 A0          IN   AL,DX       ; SAVE LSB OF 16-BIT REGISTER
I275 027B B8 D8          CMP   BX,AX      ; READ MSB OF DMA CHANNEL REGISTER
I276 027D 74 01          JE   C18A      ; PATTERN READ AS WRITTEN?
I277 027F F4 01          HLT
I278 0280
I279 0280 B3 C2 02          C18A:  ADD  DX,2       ; YES - CHECK NEXT REGISTER
I280 0285 E2 00          LOOP  C17A      ; NO - HALT THE SYSTEM
I281 0285 FE C0          INC   AL       ; NXT DMA CH
I282 0287 74 DA          JZ   C16A      ; SET I/O PORT TO NEXT CHANNEL REGISTER
I283
I284          ;----- WRITE DMA WITH 55 PATTERN
I285
I286 0289 B0 FB 55          CMP   BL,55H      ; CHECK IF 55 PATTERN DONE
I287 028C 74 09          JZ   C20A      ; GO IF YES
I288 028E B0 FB AA          CMP   BL,0AAH      ; CHECK IF AA PATTERN DONE
I289 0291 74 08          JZ   C21       ; GO IF YES
I290 0293 B0 55          MOV   AL,55H      ; SAVE LSB OF 16-BIT REGISTER
I291 0295 EB CC          JMP   C16A      ; READ MSB OF 16-BIT REGISTER
I292
I293          ;----- WRITE DMA WITH AA PATTERN
I294
I295 0297 B0 AA          C20A:  MOV   AL,0AAH      ; GET THE RESET FLAG
I296 0299 EB C8          JMP   C16A      ; DO A DUMMY MEMORY WRITE BEFORE REFRESH
I297
I298          ;----- INITIALIZE AND START MEMORY REFRESH
I299
I300 029B
I301 029B B8 1E 0072 R   C21:  MOV   BX,@RESET_FLAG      ; GET THE RESET FLAG
I302 029F A3 0010 R   MOV   @EQUIP_FLAG,AX      ; DO A DUMMY MEMORY WRITE BEFORE REFRESH
I303 02A2 B0 12          MOV   AL,18       ; START REFRESH TIMER
I304 02A4 E6 41          OUT   TIMER+1,AL
I305
I306          ;----- SET DMA COMMAND
I307
I308 02A6 2A C0          SUB   AL,AL       ; DACK SENSE LOW,DREQ SENSE HIGH
I309 02A6 E6 08          OUT   DMA+8,AL      ; LATE WRITE, FIXED PRIORITY, NORMAL
I310
I311
I312 02AA E6 D0          OUT   DMA18,AL      ; HOLD DISABLING MEMORY TO MEMORY DISABLE
I313

```

```

1313
1314
1315
1316 02AC B0 40 ;----- MODE SET ALL DMA CHANNELS
1317 02AE B6 0B OUT AL,40H ; SET MODE FOR CHANNEL 0
1318 0280 B0 C0 MOV DMA+0BH,AL
1319 02B2 E6 D6 OUT AL,0COH ; SET CASCADE MODE ON CHANNEL 4
1320 0284 EB 00 JMP $+2 ; I/O DELAY
1321 0286 B0 41 MOV AL,41H ; SET MODE FOR CHANNEL 1
1322 02B8 E6 B5 OUT DMA16+06H,AL
1323 0280 B6 D6 JMP $+2 ; SET MODE FOR CHANNEL 5
1324 02BC EB 00 MOV AL,42H ; I/O DELAY
1325 028E B0 42 OUT DMA+0BH,AL ; SET MODE FOR CHANNEL 2
1326 02C0 E6 0B OUT DMA16+06H,AL ; SET MODE FOR CHANNEL 6
1327 02C2 E6 D6 JMP $+2 ; I/O DELAY
1328 02C4 EB 00 MOV AL,43H ; SET MODE FOR CHANNEL 3
1329 02C6 E6 0B OUT DMA+0BH,AL ; SET MODE FOR CHANNEL 7
1330 02CA E6 D6 OUT DMA16+06H,AL ; SET MODE FOR CHANNEL 8
1331
1332
1333 ;----- RESTORE RESET FLAG
1334
1335 02CC 89 1E 0072 R MOV *RESET_FLAG,BX
1336
1337
1338
1339 ;----- TEST.08
1340 ;----- DMA PAGE REGISTER TEST
1341 ;----- DESCRIPTION
1342 ;----- WRITE/READ ALL PAGE REGISTERS
1343
1344
1345 ;----- CHECKPOINT 08
1346 02D0 B0 0B MOV AL,0BH ; <><><><><><><><><><><>
1347 02D2 B6 0D OUT MFG_PORT,AL ; <><> CHECKPOINT 08 <><>
1348 02D4 B4 C0 SUB AL,AL
1349 02D6 BA 0081 MOV DX,DX_PAGE
1350 02D9 B9 00FF MOV CX,0FFF ; DO ALL DATA PATTERNS
1351 02DC EE OUT DX,AL
1352 02DD 42 INC DX
1353 02DE FE C0 INC AL
1354 02E0 80 FA 008F CMP DX,8FH ; TEST DMA PAGES 81 THROUGH 8EH
1355 02E4 75 F6 JNZ C22A
1356 02E6 85 E0 XCHG AH,AL ; SAVE CURRENT DATA PATTERN
1357 02E8 FE CC DEC AH ; CHECK LAST WRITTEN
1358 02EA 4A DEC DX
1359 02EB 2A C0 C22A: SUB AL,AL ; CHANGE DATA BEFORE READ
1360 02F0 75 F5 INC DX
1361 02EE 3A C4 CMP AL,AH ; DATA AS WRITTEN?
1362 02F0 75 30 JNZ C26 ; GO ERROR HALT IF NOT
1363 02F2 FE CC DEC AH
1364 02F4 4A DEC DX
1365 02F5 80 FA 0080 CMP DX,MFG_PORT ; CONTINUE TILL PORT 80
1366 02F6 75 F0 JNZ C22B
1367 02FB FE C4 INC AL
1368 02FD 8A C4 MOV AL,AH ; NEXT PATTERN TO RIPPLE
1369 02FF E2 DB LOOP C22A
1370
1371
1372
1373 0301 B0 CC ;----- TEST LAST DMA PAGE REGISTER (USED FOR ADDRESS LINES DURING REFRESH)
1374 0303 BA 008F MOV AL,0CCH ; WRITE AN CC TO PAGE REGISTERS
1375 0306 BA E0 MOV DX,LAST_DMA_PAGE
1376 0308 EE OUT AH,AL ; SAVE THE DATA PATTERN
1377
1378
1379
1380 0309 2A C0 ;----- VERIFY PAGE REGISTER 8F
1381 030B EC SUB AL,AL ; CHANGE DATA PATTERN BEFORE READ
1382 030C 3A C4 IN AL,DX ; GET THE DATA FROM PAGE REGISTER
1383 030E 75 12 CMP AL,AH ; GO IF ERROR
1384 0310 80 FC CC JNZ C26
1385 0319 75 34 CMP AH,0CCH ; GO IF ERROR
1386 0314 B0 33 JNZ C22
1387 0317 EB EA MOV AL,03BH ; SET UP DATA PATTERN OF 33
1388 0319 JMP C22 ; DO DATA 33
1389
1390 0319 80 FC 00 C25: CMP AH,0 ; CHECK DONE
1391 031C T4 05 JZ C27 ; GO IF YES
1392 031E 2A C0 SUB AL,AL ; SET UP FOR DATA PATTERN 00
1393 0320 EB E1 JMP C22 ; DO DATA 0
1394
1395 0322 ;----- ERROR HALT
1396 0322 F4 C26: HLT ; HALT SYSTEM
1397
1398
1399 ;----- TEST.09
1400 ;----- STORAGE REFRESH TEST
1401 ;----- DESCRIPTION
1402 ;----- VERIFY REFRESH IS OCCURRING
1403
1404
1405 ;----- CHECKPOINT 09 - TEST MEMORY REFRESH
1406 0323 B0 09 C27: MOV AL,09H ; <><><><><><><><><><>
1407 0325 E6 80 OUT MFG_PORT,AL ; <><> CHECKPOINT 09 <><>
1408 0327 2B C9 SUB CX,CX
1409 0329
1410 0329 C28: IN AL,PORT_B ; INSURE REFRESH BIT IS TOGGLED
1411 0329 E4 61 TEST AL,REFRESH_BIT
1412 032B A8 10 LOOPZ C28 ; INSURE REFRESH IS OFF
1413 032D E1 FA JCXZ C26 ; ERROR HALT IF TIMEOUT
1414 0330 E3 F1 C29: IN AL,PORT_B ; INSURE REFRESH IS ON
1415 0331 2B C9 TEST AL,REFRESH_BIT
1416 0331 E4 61 LOOPNZ C29 ; INSURE REFRESH IS ON
1417 0333 A8 10 JCXZ C26 ; ERROR HALT IF NO REFRESH BIT
1418 0335 E0 FA
1419 0337 E3 E9
1420
1421
1422 ;----- TEST.10
1423 ;----- 8042 INTERFACE TEST
1424 ;----- READ CONFIGURATION JUMPERS
1425 ;----- DESCRIPTION
1426 ;----- ISSUE A SELF TEST TO THE 8042. :

```



```

1541 03C6 2B F6          SUB    SI,SI
1542 03C7 00 C0          SUB    AX,AX
1543 03CA 8E D8          MOV    DX,AX
1544 03CC 8E C0          MOV    ES,AX
1545 03CE 81 FB 1234     CMP    BX,1234H ; WARM START?
1546 03D2 T5 03          JNZ    E30A_0 ; GO IF NOT
1547 03D4 E9 0582 R     JMP    CLR_STG
1548
1549 ;----- GET THE INPUT BUFFER (SWITCH SETTINGS)
1550
1551 03D7 B0 0F          E30A_0: MOV    AL,0FH ; <><><><><><><><><><>
1552 03D9 E6 80          OUT    MFG_PORT,AL ; <><> CHECKPOINT OF <><>
1553
1554 03D8 B0 80          MOV    AL,PARITY_CHECK ; SET BASE MEMORY PARITY
1555 03D9 E6 81          OUT    DMA_PARM+6,AL ; USE AS TEMPORARY SAVE
1556 03DF BC 03EC R     MOV    SP,OFFSET C2 ; SET RETURN ADDRESS
1557 03E2 E9 0000 E     JMP    STGTST_CNT
1558 03E5 BB D8          C30:   MOV    BX,AX ; SAVE FAILING BIT PATTERN
1559 03E7 T5 0F          JNZ    C31 ; STORAGE OK, CONTINUE
1560 03E9 E9 058D R     JMP    C33
1561
1562
1563 ;----- TEMPORARY STACK FOR POST ROUTINES
1564 03EC 03E5 R          C2:   DW    C30
1565 03EE 035D R          C8042A DW    TST4_B
1566 03F0 0368 R          DBF_42A DW    TST4_C
1567 03F1 0370 R          C8042B DW    TST4_D
1568 03F4 0379 R          C8042C DW    E30B
1569 03F6 037E R          DBF_42B DW    E30C
1570
1571
1572
1573 ;----- BASE 64K STORAGE FAILURE
1574 ;----- DISPLAY THE CHECKPOINT (MFG_CHECKPOINT)
1575 ;----- AND XOR'D DATA HIGH BYTE MFG_PORT+1
1576 ;----- DISPLAY CHECKPOINT IN MFG_PORT+3
1577 ;----- DISPLAY XOR'D DATA HIGH BYTE MFG_PORT+1
1578 ;----- LOW BYTE IN MFG_PORT+2
1579 ;----- A READ/WRITE SCOPE LOOP OF THE FIRST
1580 ;----- WORD FOR POSSIBLE ADDRESS LINE FAILURES
1581
1582 03F8
1583 03F8 8A C7          C31:   MOV    AL,BH ; SAVE HIGH BYTE
1584 03FA E6 81          OUT    MFG_PORT+1,AL ; SAVE LOW BYTE
1585 03FC 8A C3
1586 03FE E6 82
1587
1588 ;----- CHECK FOR VIDEO ROM
1589
1590 0400 B9 C000          M1:   MOV    CX,0C000H ; START OF I/O ROM
1591 0401 03 8E 00          MOV    DS,CX ; POINT TO SEGMENT
1592 0402 00 00 DB          SUB    BX,BX ; GET THE FIRST 2 LOCATIONS
1593 0407 BB 07          MOV    AX,[BX]
1594 0409 EB 00          JMP    $+2 ; BUS SETTLE
1595 040B 3D AA55          CMP    AX,0AA55H ; IS THE VIDEO ROM PRESENT?
1596 040E 6A 00          POP    AX
1597 0410 T4 0C
1598 0411 00 00 0080
1599 0416 81 F9 C800
1600 041A 7C E7
1601 041C 23 C9
1602 041E
1603 041E T5 03
1604 0420 E9 050F R
1605
1606
1607 ;----- SET VIDEO MODE TO DISPLAY MEMORY ERROR
1608 ;----- THIS ROUTINE INITIALIZES THE ATTACHMENT TO
1609 ;----- TO DISPLAY FIRST 64K STORAGE ERRORS.
1610 ;----- BOTH COLOR AND MONOCHROME ATTACHMENTS ARE INITIALIZED.
1611
1612
1613 ;----- INITIALIZE COLOR/MONOCHROME
1614
1615 0423 BA 03DB          C32:   MOV    DX,3D8H ; CONTROL REGISTER ADDRESS OF COLOR CARD
1616 0424 2A C0
1617 0428 EE
1618
1619 0429 BA 03B8          MOV    DX,03B8H ; CONTROL REGISTER ADDRESS OF B/W CARD
1620 042C B0 01
1621 042E EE
1622 042F B3 EA 04
1623
1624 = 0010
1625
1626 0432 BB 0030 E          M4:   EQU    10H
1627
1628 0435 B9 0010
1629
1630
1631
1632 0438 32 E4          XOR    AH,AH ; AH IS REGISTER NUMBER DURING LOOP
1633
1634
1635 ;----- LOOP THROUGH TABLE, OUTPUTTING REGISTER ADDRESS, THEN VALUE FROM TABLE
1636 043A 8A C4          M10:  MOV    AL,AH ; GET 6845 REGISTER NUMBER
1637 043C EE
1638 043D 42          OUT    DX,AL ; POINT TO DATA PORT
1639 043E FE C4          INC    DX ; NEXT REGISTER VALUE
1640 043F 2E 07: BA 07    INC    AH ; GET TABLE VALUE
1641 0443 2E 07          MOV    AH,CS:[BX]
1642 0444 43          OUT    DX,AL ; OUT TO CHIP
1643 0445 4A          INC    BX ; NEXT IN TABLE
1644 0446 E2 F2          DEC    DX ; BACK TO PINTER REGISTER
1645 0448 8A E2          LOOP   M10 ; DO THE WHOLE TABLE
1646 0449 00 00 F0          MOV    AH,DL ; CHECK IF COLOR CARD DONE
1647 044D 80 FC D0          AND    AH,0F0H ; STRIP COLOR BITS
1648 0450 T4 08          CMP    AH,0D0H ; IF IT THE COLOR CARD?
1649 0452 BB 0000 E          JZ    2 ; CONTINUE IF COLOR
1650 0455 BA 03D4          MOV    BX,OFFSET VIDEO_PARMS ; POINT TO VIDEO PARAMETERS
1651 0458 EB DB          MOV    DX,3D4H ; COLOR BASE
1652
1653 ;----- FILL REGEN AREA WITH BLANK
1654
1655

```

```

1655 045A 33 FF Z_31: XOR DI,DI ; SET UP POINTER FOR REGEN
1655 045C B8 0000 MOV AX,0B000H ; SET UP ES TO VIDEO REGEN
1657 045F BE C0 MOV ES,AX

1658 0461 B9 0800 MOV CX,2048 ; NUMBER OF WORDS IN MONOCHROME CARD
1660 0464 B8 0720 MOV AX,1000H+7*H ; FILL CHARACTER FOR ALPHA + ATTRIBUTE
1661 0467 F3/ AB REP STOSW ; FILL THE REGEN BUFFER WITH BLANKS

1662 0469 33 FF XOR DI,DI ; CLEAR COLOR VIDEO BUFFER MEMORY
1664 046B BB B800 MOV BX,0B800H ; SET UP ES TO COLOR VIDEO MEMORY
1665 046E BE C3 MOV ES,BX
1666 0470 B9 2000 MOV CX,8192 ; SET UP ES TO COLOR VIDEO MEMORY
1667 0473 F3/ AB REP STOSW ; FILL WITH BLANKS

1668 0470 33 FF ;----- ENABLE VIDEO AND CORRECT PORT SETTING
1670 0475 B8 03B8 MOV DX,3BBH
1672 0478 B8 29 MOV AL,29H
1673 047A EE OUT DX,AL ; SET VIDEO ENABLE PORT

1674 0475 33 FF ;----- SET UP OVERSCAN REGISTER
1677 047B 42 INC DX ; SET OVERSCAN PORT TO A DEFAULT
1679 047C B8 30 MOV AL,30H ; VALUE 30H FOR ALL MODES EXCEPT 640X200
1679 047E EE OUT DX,AL ; OUTPUT THE CORRECT VALUE TO 3D9 PORT

1680 047F 33 FF ;----- ENABLE COLOR VIDEO AND CORRECT PORT SETTING
1683 047F BA 03D8 MOV DX,3DBH
1684 0482 B8 28 MOV AL,28H
1685 0484 EE OUT DX,AL ; SET VIDEO ENABLE PORT

1686 047F 33 FF ;----- SET UP OVERSCAN REGISTER
1689 0485 42 INC DX ; SET OVERSCAN PORT TO A DEFAULT
1690 0486 B8 30 MOV AL,30H ; VALUE 30H FOR ALL MODES EXCEPT 640X200
1691 0488 EE OUT DX,AL ; OUTPUT THE CORRECT VALUE TO 3D9 PORT

1692 0489 33 FF ;----- DISPLAY FAILING CHECKPOINT AND
1695 0489 BC C8 MOV AX,CS ; SET STACK SEGMENT TO CODE SEGMENT
1696 048B BE D0 MOV SS,AX
1697 048C BE 0000
1698 048D BB B800 MOV BX,0B000H
1699 0490 BE DB MOV DS,BX ; SET DS TO B/W DISPLAY BUFFER
1700 0492 B8 30 Z_0: MOV AL,'0' ; DISPLAY BANK 000000
1702 0494 B9 0006 MOV CX,6 ; START AT 0
1703 0495 B8 05 SUB DI,DI ; WRITE TO DISPLAY REGEN BUFFER
1704 0499 B8 05 Z: MOV [DI],AL ; POINT TO NEXT POSITION
1705 049B 47 INC DI
1706 049C 47 INC DI
1707 049D E2 FA LOOP Z ; POINT TO START OF BUFFER

1708 049F 80 FF B8 CMP BH,0B8H ; CHECK THAT COLOR BUFFER WRITTEN
1710 04A2 74 0C JZ Z_0
1711 04A4 2B FF SUB DI,DI ; POINT TO START OF BUFFER

1713 04A6 B7 B0 MOV BH,0B0H ; ES = MONOCHROME
1714 04A8 BE C3 MOV ES,BX ; SET SEGMENT TO COLOR
1715 04AA B7 B8 MOV BH,0B8H ; DS = COLOR
1716 04AC BE DB MOV DS,BX
1717 04AE EB E2 JMP Z_0

1719 0470 33 FF ;----- PRINT FAILING BIT PATTERN
1720 04B0 B8 20 Z_1: MOV AL,'.' ; DISPLAY A BLANK
1722 04B2 B8 05 MOV ES:[DI],AL ; WRITE TO COLOR BUFFER
1723 04B4 26: 88 05 MOV ES:[DI],AL ; WRITE TO MONOCHROME REGEN BUFFER
1724 04B7 47 INC DI ; POINT TO NEXT POSITION
1725 04B8 47
1726 04B9 E4 81 IN AL,MFG_PORT+1 ; GET THE HIGH BYTE OF FAILING PATTERN
1727 04B8 B8 04 MOV CL,4 ; SHIFT COUNT
1728 04B9 B8 02 SHR AL,CL ; NIBBLE SWAP
1729 04B9 EB 05TA R MOV SP,OFFSET Z1_0
1730 04C2 EB 1B JMP SHORT PR

1731 04C4 E4 81 Z1: IN AL,MFG_PORT+1 ; ISOLATE TO LOW NIBBLE
1733 04C6 24 0F AND AL,0FH
1734 04C8 B8 05TC R MOV SP,OFFSET Z2_0 ; RETURN TO Z4:
1735 04C9 EB 12 JMP SHORT PR
1736 04CD E4 82 Z2: IN AL,MFG_PORT+2 ; GET THE HIGH BYTE OF FAILING PATTERN
1737 04CF B1 04 MOV CL,4 ; SHIFT COUNT
1738 04D1 D2 E8 SHR AL,CL ; NIBBLE SWAP
1739 04D3 BC 05TE R MOV SP,OFFSET Z3_0 ; ISOLATE TO LOW NIBBLE
1740 04D5 B8 05TC R JMP SHORT PR
1741 04D9 E4 82 Z3: IN AL,MFG_PORT+2 ; RETURN TO Z4:
1742 04DA 24 0F AND AL,0FH
1743 04DC BC 0580 R MOV SP,OFFSET Z4_0 ; ISOLATE TO LOW NIBBLE
1744 04E5 33 FF ;----- CONVERT AND PRINT
1745 04E6 33 FF PR: ADD AL,090H ; CONVERT 00-0F TO ASCII CHARACTER
1747 04DF 04 90 DAA ; ADD FIRST CONVERSION FACTOR
1748 04E1 27 ADC AL,040H ; ADJUST FOR NUMERIC AND ALPHA RANGE
1749 04E2 14 40 DAA ; ADD CONVERSION AND ADJUST LOW NIBBLE
1750 04E4 27 RET ; ADJUST HIGH NIBBLE TO ASCII RANGE

1751 04E5 B8 05 Z4: MOV [DI],AL ; WRITE TO COLOR BUFFER
1753 04E7 26: 88 05 MOV ES:[DI],AL ; WRITE TO MONOCHROME BUFFER
1754 04E8 47 INC DI ; POINT TO NEXT POSITION
1755 04E9 B7 47
1756 04EC C3 RET

1757 04E5 33 FF ;----- DISPLAY 201 ERROR
1758 04E6 33 FF Z4: MOV AL,'.' ; DISPLAY A BLANK
1760 04ED B8 20 MOV [DI],AL ; WRITE TO DISPLAY REGEN BUFFER
1761 04EF B8 05 MOV ES:[DI],AL ; WRITE TO MONOCHROME BUFFER
1762 04F1 26: 88 05 INC DI ; POINT TO NEXT POSITION
1763 04F4 47
1764 04F5 B8 05 MOV AL,'2' ; DISPLAY 201 ERROR
1766 04F8 B8 05 MOV [DI],AL ; WRITE TO DISPLAY REGEN BUFFER
1767 04FA 26: 88 05 MOV ES:[DI],AL ; WRITE TO MONOCHROME BUFFER
1768 04FD 47 INC DI ; POINT TO NEXT POSITION

```

```

1769 04FE 47           INC    DI
1770 04FF B0 30         MOV    AL, '0'
1771 0501 88 05         MOV    [DI], AL      ; WRITE TO DISPLAY REGEN BUFFER
1772 0502 26: 88 05     MOV    ES:[DI], AL   ; WRITE TO MONOCHROME BUFFER
1773 0503 47           INC    DI      ; POINT TO NEXT POSITION
1774 0504 00             INC    DI
1775 0505 80 31         MOV    AL, '1'
1776 050A 88 05         MOV    [DI], AL      ; WRITE TO DISPLAY REGEN BUFFER
1777 050C 26: 88 05     MOV    ES:[DI], AL   ; WRITE TO MONOCHROME BUFFER
1778
1779           ;----- ROLL ERROR CODE IN MFG_PORT --> FIRST THE CHECKPOINT
1780
1781 050F 80 00         C31_0: MOV    AL, 0DDH      ; <><><><><><><><>
1782 0511 E6 80         OUT   MFG_PORT, AL    ; <><> CHECKPOINT DD <><>
1783 0513 E6 83         OUT   MFG_PORT+3, AL  ; ALSO DISPLAY CHECK POINT IN PORT B3
1784 0515 2B C9
1785 0517
1786
1787 0517 88 00         C31_A: SUB   AX, AX      ; SETUP SEGMENT
1788 0519 8E D8         MOV    DS, AX
1789 051B B8 AA55       MOV    AX, 0AA55H    ; WRITE AN AA55
1790 051E 2B FF         SUB   D1, D1
1791 0522 89 05         MOV    [D1], AX
1792 0522 88 05         MOV    AX, [D1]
1793 0522 E2 F1         LOOP  C31_A      ; READ THE FIRST WORD
1794 0526 00 00
1795 0526 89 05         C31_B: MOV    [D1], AX
1796 0526 BB 05         MOV    AX, [D1]
1797 052A E2 FA         LOOP  C31_B      ; DISPLAY CHECKPOINT LONGER
1798 052C
1799 052C 89 05         C31_C: MOV    [D1], AX
1800 052C BB 05         MOV    AX, [D1]
1801 0530 E2 FA         LOOP  C31_C
1802 0532
1803 0532 89 05         C31_D: MOV    [D1], AX
1804 0534 BB 05         MOV    AX, [D1]
1805 0534 E2 FA         LOOP  C31_D
1806
1807 0538 89 05         C31_E: MOV    [D1], AX
1808 053A BB 05         MOV    AX, [D1]
1809 053C E2 FA         LOOP  C31_E
1810
1811           ;----- ROLL ERROR CODE IN MFG_PORT --> NEXT THE HIGH BYTE
1812
1813 053E E4 81         IN    AL, MFG_PORT+1    ; XOR OF FAILING BIT PATTERN
1814 0540 E6 80         OUT  MFG_PORT, AL    ; HIGH BYTE
1815 0542
1816 0542 BB AA55       C31_G: MOV    AX, 0AA55H    ; WRITE AN AA55
1817 0545 89 05         MOV    [D1], AX
1818 0545 BB 05         MOV    AX, [D1]
1819 0549 E2 F7         LOOP  C31_G      ; READ THE FIRST WORD
1820 054B
1821 054B 89 05         C31_H: MOV    [D1], AX
1822 054D BB 05         MOV    AX, [D1]
1823 054F E2 FA         LOOP  C31_H
1824 0551
1825 0551 89 05         C31_I: MOV    [D1], AX
1826 0553 BB 05         MOV    AX, [D1]
1827 0555 E2 FA         LOOP  C31_I
1828
1829           ;----- ROLL ERROR CODE IN MFG_PORT --> THEN THE LOW BYTE
1830
1831 0557 E4 82         IN    AL, MFG_PORT+2    ; LOW BYTE
1832 0559 E6 80         OUT  MFG_PORT, AL
1833 055B B8 AA55       C31_K: MOV    AX, 0AA55H    ; WRITE AN AA55
1834 055E 2B FF         SUB   D1, D1
1835 0561 89 05         MOV    [D1], AX
1836 0562 BB 05         MOV    AX, [D1]
1837 0564 E2 F8         LOOP  C31_K      ; READ THE FIRST WORD
1838 0566
1839 0566 89 05         C31_L: MOV    [D1], AX
1840 0568 BB 05         MOV    AX, [D1]
1841 056A E2 FA         LOOP  C31_L
1842 056C
1843 056C 89 05         C31_M: MOV    [D1], AX
1844 056E BB 05         MOV    AX, [D1]
1845 0570 E2 FA         LOOP  C31_M
1846 0572
1847 0572 89 05         C31_N: MOV    [D1], AX
1848 0574 BB 05         MOV    AX, [D1]
1849 0574 E2 FA         LOOP  C31_N
1850 0576 EB 95         JMP   C31_O      ; DO AGAIN
1851
1852 057A 04CA R        Z1_0: DW   Z1      ; TEMPORARY STACK
1853 057C 04CD R        Z2_0: DW   Z2      ; TEMPORARY STACK
1854 057E 04DB R        Z3_0: DW   Z3      ; TEMPORARY STACK
1855 0580 04ED R        Z4_0: DW   Z4      ; TEMPORARY STACK
1856
1857
1858           ;----- CLEAR STORAGE ENTRY
1859
1860
1861 0582           CLR_STG: ASSUME DS:DATA
1862             PUSH DS
1863             MOV  AX, DATA      ; STORE 32K WORDS OF 0000
1864             MOV  DS, AX
1865             MOV  DS, AX      ; RESTORE DATA SEGMENT
1866             MOV  DS, AX
1867             MOV  DS, AX      ; RESTORE RESET FLAG
1868
1869           ;----- SETUP STACK SEGMENT AND SP
1870 058D B8 ---- R      C331: MOV   AX, DATA      ; SET DATA SEGMENT
1871 058D B8 ---- R      MOV   DS, AX
1872 0590 8E D8         MOV   SP, POST_SS    ; GET STACK VALUE
1873 0592 BC 0000       MOV   SS, SP      ; SET THE STACK UP
1874 0595 BB D4         MOV   SP, POST_SP    ; STACK IS READY TO GO
1875
1876
1877           ;----- INITIALIZE DISPLAY ROW COUNT
1878
1879 059A C6 06 0084 R 18 MOV  @ROWS, 25-1    ; SET ROWS FOR PRINT SCREEN DEFAULT
1880
1881 059F B0 11         MOV  AL, 11H
1882 05A1 E6 80         OUT  MFG_PORT, AL    ; <><><><><><><><>

```



```

1997
1998      ;----- READ AND VERIFY 286 REGISTERS
1999
2000 062A BD D8A0      MOV    BP,_GDT_LOC      ; STORE THE REGISTERS HERE
2001      SEG0V  ES
2002 062D 26      +    DB    026H
2003      S1DT  [BP]
2004 062E 0F      +    DB    00FH
2005 062F      +    ??0007  LABEL  BYTE
2006 062F BB 4E 00      +    MOV    CX,[BP]
2007      ??0008  LABEL  BYTE
2008      +    DB    00H
2009 062F 01      +    DB    001H
2010 0632      +    ORG   OFFSET CS:??0007
2011 0632 BD D8A5      MOV    BP,_GDT_LOC+5
2012      SEG0V  ES
2013 0635 26      +    DB    026H
2014      SGD7  [BP]
2015 0636 0F      +    DB    00FH
2016 0637      +    ??000A  LABEL  BYTE
2017 0637 03 46 00      +    ADD   AX,[BP]
2018 063A      +    ??000B  LABEL  BYTE
2019 063A      +    ORG   OFFSET CS:??000A
2020 0637 01      +    DB    001H
2021 063A      +    ORG   OFFSET CS:??000B
2022 063A BF D0A0      MOV    DI,_SYS_IDT_LOC
2023 063B B9 0005      MOV    AX,[DI]
2024 063F B9 0005      MOV    CX,5
2025 0642 BE D8A0      C37B:  MOV    SI,_GDT_LOC
2026 0643 B9 3B 04      CMP   AX,ES:[SI]
2027 0648 75 C8      JNZ   ERR_PRNT
2028 064A 46      INC   SI
2029 064B 46      INC   SI
2030 064C E2 F7      LOOP  C37B
2031 064E C3      RET
2032
2033
2034
2035
2036      ;----- INITIALIZE THE 8259 INTERRUPT #1 CONTROLLER CHIP
2037
2038 064F      C37A:  SUB   AL,AL      ; RESET MATH PROCESSOR
2039 0651 E6 F1      OUT   X287+1,AL
2040 0653 B0 11      MOV   AL,11H
2041 0655 E6 20      OUT   INTA00,AL
2042 0657 EB 00      JMP   $+2
2043 0659 B0 08      MOV   AL,8
2044 065B E6 21      OUT   INTA01,AL
2045 065D EB 00      JMP   $+2
2046
2047 065F B0 04      MOV   AL,04H
2048 0661 E6 21      OUT   INTA01,AL
2049 0663 EB 00      JMP   $+2
2050 0665 B0 01      MOV   AL,01H
2051 0667 75 21      OUT   INTA01,AL
2052 0669 EB 00      JMP   $+2
2053 066B B0 FF      MOV   AL,0FFH
2054 066D E6 21      OUT   INTA01,AL
2055
2056
2057
2058
2059      ;----- INITIALIZE THE 8259 INTERRUPT #2 CONTROLLER CHIP
2060
2061 066F B0 13      MOV   AL,13H      ; <><><><><><><><><><><>
2062 0671 E6 80      OUT   MFG_PORT,AL      ; <><> CHECKPOINT 13 <><>
2063
2064 0673 B0 11      MOV   AL,11H
2065 0675 E6 A0      OUT   INTB00,AL
2066 0677 EB 00      JMP   $+2
2067 067B E6 A1      MOV   AL,INT_TYPE
2068 067D B0 02      OUT   INTB01,AL
2069 067F 75 00      MOV   AL,02H
2070 0681 E6 A1      OUT   INTB01,AL
2071 0683 EB 00      JMP   $+2
2072 0685 B0 01      MOV   AL,01H
2073 0687 E6 A1      OUT   INTB01,AL
2074 0689 EB 00      JMP   $+2
2075 068B B0 FF      MOV   AL,0FFH
2076 068D E6 A1      OUT   INTB01,AL
2077
2078
2079
2080 068F B0 14      MOV   AL,14H      ; <><><><><><><><><><><>
2081 0691 E6 80      OUT   MFG_PORT,AL      ; <><> CHECKPOINT 14 <><>
2082
2083 0693 B9 0078      MOV   CX,78H
2084 0696 2B FF      SUB   DI,DI
2085 0698 B6 C7      MOV   ES,DI
2086 069A B8 0000 E  D3:   MOV   AX,OFFSET D11
2087 069B B6 AB      STOSW
2088 069E C6 C8      MOV   AX,CS
2089 06A0 AB      STOSW
2090 06A1 E2 F7      LOOP  D3
2091
2092      ;----- SET UP THE INTERRUPT VECTORS TO TEMPORARY INTERRUPT
2093
2094 06A3 B0 15      MOV   AL,15H      ; <><><><><><><><><><><>
2095 06A5 E6 80      OUT   MFG_PORT,AL      ; <><> CHECKPOINT 15 <><>
2096
2097
2098 06A7 BF 0040 R  D3A:  MOVSW
2099 06AA B6 DE      MOV   DI,OFFSET *VIDEO_INT
2100 06AB 1F      POP   CS
2101 06AC B8 D8      MOV   DS,DI
2102 06AE BE 0010 E  MOV   AX,DS
2103 06B1 B9 0010      MOV   SI,OFFSET VECTOR_TABLE+16
2104
2105 06B4 A5      MOV   CX,16
2106 06B5 47      INC   DI
2107 06B6 47      INC   DI
2108 06B7 E2 FB      LOOP  D3A
2109
2110      ;----- ESTABLISH BIOS SUBROUTINE CALL INTERRUPT VECTORS

```



```

2225 0762 BC 8000      MOV    SP,POST_SP
2226 0765 E8 0000 E     CALL   SYSINIT      ; CALL THE DESCRIPTOR TABLE BUILDER
2227                                     ; AND REAL-TO-PROTECTED MODE SWITCHER
2228
2229 0768 B0 1A      MOV    AL,IAH
2230 076A E6 80      OUT   MFG_PORT,AL   ; 0000000000000000
2231
2232      ;---- SET TEMPORARY STACK
2233
2234 076C 6A 08      PUSH  BYTE PTR GDT_PTR  ; SET (DS:) SELECTOR TO GDT SEGMENT
2235 076E F          POP   DS
2236 076F C7 06 005A 0000  MOV   DS:SS_TEMP,base_lo_word,0
2237 0775 C6 06 005C 00  MOV   BYTE PTR DS:(SS_TEMP,base_hi_byte),0
2238 077A BE 0058      MOV   SI,SS_TEMP
2239 077D BE D6      MOV   SS,SI-
2240 077F BC FFFF      MOV   SP,MAX_SEG_LEN-2
2241
2242
2243      ;---- TEST_13
2244      ; PROTECTED MODE TEST AND MEMORY SIZE DETERMINE ( 0 --> 640K )
2245
2246      ;---- DESCRIPTION:
2247      ; THIS ROUTINE RUNS IN PROTECTED MODE IN ORDER TO ADDRESS ALL OF STORAGE.
2248      ; IT CHECKS THE MACHINE STATUS WORD (MSW) FOR PROTECTED MODE AND THE BASE
2249      ; MEMORY SIZE IS DETERMINED AND SAVED. BIT 4 OF THE CMOS DIAGNOSTIC
2250      ; STATUS BYTE IS SET IF 512K --> 640K MEMORY IS INSTALLED.
2251      ; DURING A POWER UP SEQUENCE THE MEMORY SIZE DETERMINE IS DONE WITH
2252      ; PLANAR AND I/O PARITY CHECKS DISABLED. DURING A SOFT RESET THE MEMORY
2253      ; SIZE DETERMINE WILL CHECK FOR PARITY ERRORS.
2254
2255
2256      ;---- INSURE PROTECTED MODE
2257
2258 0782 0F 01 E0      +  SMSW  AX          ; GET THE MACHINE STATUS WORD
2259 0785 A9 0001      DB    00FH,001H,0E0H
2260 0788 75 0C      TEST  AX,VIRTUAL_ENABLE ; ARE WE IN PROTECTED MODE
2261 JNZ   VIR_OK
2262
2263 078A B8 088F      SHUT_8: MOV   AX,8H+(CMOS_SHUT_DOWN+NNI) ; SET THE RETURN ADDRESS
2264 078D E8 0000 E     CALL   CMOS_WRITE      ; AND SET SHUTDOWN 8
2265 0790 E9 0000 E     JMP   PROC_SHUTDOWN ; CAUSE A SHUTDOWN
2266
2267      ;---- VIRTUAL MODE ERROR HALT
2268
2269 0793 F4      SHUT8: HLT
2270 0794 EB FD      JMP   SHUT8      ; ERROR HALT
2271
2272      ;---- 64K SEGMENT LIMIT
2273
2274 0796 C7 06 0048 FFFF  VIR_OK: MOV   DS:ES_TEMP,SEG_LIMIT,MAX_SEG_LEN
2275
2276      ;---- CPL0, DATA ACCESS RIGHTS
2277
2278 079C C6 06 004D 93  MOV   BYTE PTR DS:(ES_TEMP,DATA_ACC_RIGHTS),CPL0_DATA_ACCESS
2279
2280      ;---- START WITH SEGMENT ADDRESS 01-0000 (SECOND 64K)
2281
2282 07A1 C6 06 004C 01  MOV   BYTE_PTR DS:(ES_TEMP,BASE_HI_BYTE),01H
2283 07A6 D6 06 004A 0000 DS:ES_TEMP,BASE_LO_WORD,0H
2284
2285 07AC B0 1B      MOV   AL,1BH
2286 07AE E6 80      OUT   MFG_PORT,AL   ; 0000000000000000
2287
2288 07B0 BB 0040      MOV   BX,16*4  ; SET THE FIRST 64K DONE
2289
2290
2291      ;---- START STORAGE SIZE/CLEAR
2292 07B3
2293 07B3 6A 48      NOT_DONE: PUSH  BYTE PTR ES_TEMP ; POINT ES TO DATA
2294 07B5 07      POP   ES
2295 07B6 E8 07D2 R   CALL   HOW_BIG      ; POINT TO SEGMENT TO TEST
2296 07B9 74 03      JZ    NOT_FIN      ; DO THE FIRST 64K
2297 07B9 E9 0870 R   JMP   DONE
2298
2299 07BE
2300 07BE 83 C3 40      NOT_FIN: ADD   BX,16*4  ; BUMP MEMORY COUNT BY 64K
2301
2302      ;---- DO NEXT 64K (0X0000) BLOCK
2303
2304 07C1 FE 06 004C 04  INC   BYTE PTR DS:(ES_TEMP,BASE_HI_BYTE)
2305
2306      ;---- CHECK FOR END OF FIRST 640K (END OF BASE MEMORY)
2307
2308 07C5 80 3E 004C 0A  CMP   BYTE_PTR DS:(ES_TEMP,BASE_HI_BYTE),0AH
2309 07CA 75 E7      JNZ   NOT_DONE      ; GO IF NOT
2310 07CC E8 084D R   CALL   HOW_BIG_END ; GO SET MEMORY SIZE
2311 07CF E9 0870 R   JMP   DONE
2312
2313      ;---- FILL/CHECK LOOP
2314
2315 07D2
2316 07D2 2B FF      HOW_BIG: SUB   DI,DI
2317 07D4 B8 AA55      MOV   AX,0AA55H      ; TEST PATTERN
2318 07D7 B8 C8      MOV   CX,AX      ; SAVE PATTERN
2319 07D9 261,89 05    MOV   ES:[DI],AX ; WRITE PATTERN TO MEMORY
2320 07D9 261,89 05    MOV   AL,CFH      ; PUT PATTERN IN AL
2321 07D9 261,89 05    MOV   AX,ES:[DI] ; TEST PATTERN
2322 07E1 261,89 05    MOV   ES:[DI],AX ; INSURE NO PARITY I/O CHECK
2323 07E4 33 C1      XOR   AX,CX      ; COMPARE PATTERNS
2324 07E6 75 65      JNZ   HOW_BIG_END ; GO END IF NO COMPARE
2325
2326 07E8 1E
2327 07E9 6A 18      PUSH  DS          ; POINT TO SYSTEM DATA AREA
2328 07EB 1F      PUSH  BYTE PTR RSDA_PTR
2329 POP   DS          ; GET (DS)
2330      ;---- IS THIS A SOFT RESET
2331
2332 07EC 81 3E 0072 R 1234  CMP   @RESET_FLAG,1234H ; SOFT RESET
2333 07F2 1F      POP   DS          ; RESTORE DS
2334 07F3 75 36      JNZ   HOW_BIG_2 ; GO IF NOT SOFT RESET
2335
2336      ;---- INSURE NO PARITY WITH PARITY BITS OFF
2337
2338 07F5 26: C7 05 0101  MOV   WORD PTR ES:[DI],0101H ; TURN OFF BOTH PARITY BITS

```



```

2453
2454 0892 FE 06 004C           INC    BYTE PTR DS:[ES_TEMP.BASE_HI_BYTE]
2455
2456           ;----- CHECK FOR TOP OF MEMORY (FE0000)
2457
2458 0896 80 3E 004C FE         CMP    BYTE PTR DS:[ES_TEMP.BASE_HI_BYTE],0FEH ; LAST OF MEMORY?
2459 0898 75 E7               JNZ    NOT_DONE1 ; GO IF NOT
2460 089D E8 0917 R            CALL   HOW_BIG_END1 ; GO SET MEMORY SIZE
2461 08A0 E9 092A R            JMP    DONE1
2462
2463           ;----- FILL/CHECK LOOP
2464
2465 00A3 HOW_BIG1:
2466 00A3 2B FF               SUB   D1,D1
2467 00A5 B8 AA55             MOV    AX,0AA55H ; TEST PATTERN
2468 00A8 B8 C8               MOV    CX,0AX
2469 00AA 26: 89 05            MOV    ES:[D1],AX ; SAVE PATTERN
2470 00AD B0 0F               MOV    AL,0FH
2471 00AF 26: 88 05            MOV    AX,ES:[D1] ; SEND PATTERN TO MEMORY
2472 00B2 26: 89 05            MOV    ES:[D1],AX ; PUT SOMETHING IN AL
2473 00B5 33 C1               XOR   AX,CX
2474 00B7 75 5E               JNZ    HOW_BIG_END1 ; GET PATTERN
2475
2476           ;----- IS THIS A SOFT RESET
2477
2478 00B9 1E
2479 00BA 6A 18
2480 00BC 1F
2481 00BD 81 3E 0072 R 1234
2482 00C3 1F
2483 00C4 75 2F
2484
2485           ;----- CHECK PARITY WITH PARITY BITS OFF
2486
2487 00C6 26: C7 05 0101
2488 00C6 6A FF               PUSH  DS
2489 00C6 58                 PUSH  BYTE PTR RSDA_PTR ; POINT TO SYSTEM DATA AREA
2490 00C6 26: 8B 05            POP   DS
2491 00C6 58                 CMP   PRESET_FLAG,1234H ; SOFT RESET
2492 00C6 26: 8B 05            POP   DS
2493 00C6 58                 JNZ   HOW_BIG_2A ; RESTORE DS
2494
2495           ;----- CHECK PARITY WITH PARITY BITS ON
2496
2497 00D1 E4 61               IN    AL,PORT_B ; TURN OFF BOTH PARITY BITS
2498 00D3 24 C0               AND   AL,PARITY_ERR ; PLACE 0FFFH IN STACK (BUS BITS ON)
2499 00D5 26: 89 05            MOV   ES:[D1],AX ; DELAY - CAUSING BUS BITS ON
2500 00D5 75 3D               JNZ   HOW_BIG_END1 ; CHECK PARITY
2501
2502 00C6 26: C7 05 FFFF
2503 00E1 58
2504 00E2 26: 8B 05
2505 00E6 61
2506 00E8 24 C0
2507 00E8 26: 89 05
2508 00E9 58
2509 00E9 75 27
2510 00F3 0FFF
2511 00F3 75 22
2512
2513           ;----- CLEAR 64K BLOCK OF MEMORY
2514 00F5 HOW_BIG_2A:
2515 00F5 2B C0
2516 00F7 B9 8000
2517 00FA F3/ AB
2518
2519           ;----- CHECK 64K BLOCK FOR PARITY CHECK (VALID TEST DURING SOFT RESET ONLY)
2520
2521 00FC 1E
2522 00FD 06
2523 00FE 06
2524 00FF 1F
2525 0090 B9 8000
2526 0093 2B F6
2527 0093 F3 AD
2528 0097 00 FF
2529 0090 E4 61
2530 0090 24 C0
2531 0090 26: C7 05 0000
2532 0092 07
2533 0093 1F
2534 0094 75 01
2535
2536 0016 C3
2537
2538 0017
2539 0017 B0 1E
2540 0019 E6 60
2541
2542           ;----- SET EXPANSION MEMORY SIZE DETERMINED IN CMOS
2543
2544 001B B0 B0               MOV   AL,CMOS_U_M_S_LO+NMI ; ADDRESS LOW BYTE
2545 001D B0 E3               CALL  CMOS_WRITE ; GET LOW MEMORY SIZE
2546 001F B0 0000 E            MOV   AL,CMOS_U_M_S_HI+NMI ; SET LOW MEMORY
2547 0022 B0 B1               CALL  CMOS_WRITE ; ADDRESS HI BYTE
2548 0024 B0 E7               MOV   AH,BH ; GET THE HIGH MEMORY SIZE
2549 0026 E8 0000 E            CALL  CMOS_WRITE ; PLACE IN CMOS
2550 0029 C3
2551
2552           ;----- TEST ADDRESS LINES 19 - 23
2553
2554 002A B0 1F
2555 002C E6 80
2556 002E C6 06 004C 00
2557 0033 2B D0
2558 0036 00 FFFF
2559 0036 E8 0967 R
2560 0038 2B D2
2561
2562 003D C6 06 004C 08
2563 0042 E8 0967 R
2564 0045 C6 06 004C 10
2565 004A E8 0967 R
2566 004D C6 06 004C 20

```

```

2561 0952 E8 0967 R      CALL    SD0
2568 0955 C6 06 004C 40    MOV    BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE),40H
2569 095A E8 0967 R      CALL    SD0
2570 0960 C6 06 004C 80    MOV    BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE),80H
2571 0962 E8 0967 R      CALL    SD0
2572
2573 0965 EB 18      JMP    SHORT SD2      ; TEST PASSED CONTINUE
2574
2575 0967 SD0:           SD0
2576 0967 6A 48      PUSH   BYTE PTR ES_TEMP      ; POINT ES TO DATA
2577 0969 07          POP    ES
2578 096A 26: 89 15      MOV    ES:[DI],DX      ; POINT TO SEGMENT TO TEST
2579
2580 096D C6 06 004C 00    MOV    BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE),00H
2581
2582 0972 6A 48      PUSH   BYTE PTR ES_TEMP      ; POINT ES TO DATA
2583 0973 07          POP    ES
2584 0975 26: 83 3D FF    CMP    WORD PTR ES:[DI],0FFFFH ; POINT LOCATION TO CHANGE?
2585 0979 74 03          JZ    SD1
2586 097B E9 078A R      JMP    SHUT_8      ; CONTINUE IF NOT
2587 097E SD1:           SD1
2588 097E C3          RET
2589
2590
2591
2592 097F B0 20      SD2:           SD2
2593 0981 E6 80          MOV    AL,20H      ; <><><><><><><><><><><>
2594 0983 E4 61          OUT   MFG_PORT,AL      ; <><> CHECKPOINT 20 <><>
2595 0985 00 0C          IN    AL,PORT_B
2596 0986 00 01          OR    AL,PORT_B
2597 0989 24 F3          OUT   AL,PORT_B
2598 098B E6 61          AND   AL,PORT_B
2599 098D E9 0000 E      OUT   AL,PORT_B
2600
2601
2602
2603
2604
2605 0990 B0 21      SHUT1:          MOV    AL,21H      ; <><><><><><><><><><><>
2606 0992 E6 80          OUT   MFG_PORT,AL      ; <><> CHECKPOINT 21 <><>
2607 0994 BC --- R      MOV    SP,AB50      ; SET REAL MODE STACK
2608 0997 8E D4          MOV    SS,SP
2609 0999 BC 0400 R      MOV    SP,OFFSET @TOS
2610
2611
2612
2613 099C 2B FF      ;----- SET DIVIDE 0 VECTOR OFFSET
2614 099D 8E C7          SUB   DI,DI      ; POINT TO FIRST INTERRUPT LOCATION
2615 09A0 B8 0000 E      MOV    ES,DI
2616 09A3 AB          MOV    AX,OFFSET DI1      ; SET ES TO ABS0 SEGMENT
2617
2618 09A4 E8 0000 E      STOSW
2619
2620
2621
2622 09A7 B8 8E8E      ;----- GET THE CONFIGURATION FROM CMOS
2623 09AA E8 0000 E      MOV    AX,X'1CMOS_DIAG+NMI' ; CHECK CMOS GOOD
2624 09AD A8 C0          CALL   CMOS_READ      ; GET THE STATUS
2625 09AF 74 03          TEST  AL,BAD_BAT+BAD_CKSUM ; VALID CMOS ?
2626 09B0 E9 003A R      JZ    M_OK
2627 09B4
2628 09B4 24 DF          JMP    BAD_MOS      ; GO IF YES
2629 09B6 86 C4          M_OK:           AND   AL,0DFH      ; CLEAR THE MINIMUM CONFIG BIT
2630 09B8 E8 0000 E      XCHG  AL,AH      ; SAVE THE STATUS BYTE
2631
2632
2633
2634 09B8 81 3E 0072 R 1234    CALL   CMOS_WRITE      ; BACK INTO CMOS
2635 09C1 74 10          CMP    @RESET_FLAG,1234H ; CHECK FOR SOFT RESET
2636 09C1 64
2637 09C3 B0 96          JE    M_OK_64      ; BYPASS IF SOFT RESET
2638 09C5 E9 0000 E      MOV    AL,CMOS_B_M_S_HI+NMI ; GET THE BASE MEMORY SIZE HIGH BYTE
2639 09C6 80 24          CALL   CMOS_READ      ; MASK FOR MANUFACTURING TEST BITS
2640 09CA 3C C0          AND   AL,0E0H
2641 09CC 75 05          CMP    AL,0C0H      ; CHECK FOR MANUFACTURING TEST MODE SET
2642
2643 09CE C6 06 0072 R 64    JNE    M_OK_64      ; SKIP IF NOT MANUFACTURING LINE TEST
2644
2645
2646
2647 09D3
2648 09D3 B0 94          M_OK_64:          MOV    AL,CMOS_EQUIP+NMI ; GET THE EQUIPMENT BYTE
2649 09D5 E8 0000 E      CALL   CMOS_READ      ; SAVE VIDEO TYPE
2650 09D8 80 E0          MOV    AH,AL
2651 09D9 80 00          TEST  AL,0B0H ; ANY VIDEO?
2652 09DC 75 31          JNZ    MOS_OK_1      ; CONTINUE
2653 09DE EB 09EC R      CALL   CHK_VIDEO      ; INSURE VIDEO ROM PRESENT
2654 09E1 74 4C          JZ    MOS_OK_1
2655
2656 09E3 F6 06 0012 R 20    MOS_OK_1:          MOV    MOS_OK
2657 09E8 74 6F          TEST  @MFG_TST,MFG_LOOP ; EXCEPT IF MFG JUMPER IS INSTALLED
2658
2659 09EA EB 4E          JZ    NORMAL_CONFIG ; GO IF INSTALLED
2660
2661
2662
2663 09EC
2664 09EC B9 C000      ;----- ROUTINE CHECK FOR VIDEO FEATURE ROM PRESENT
2665 09EF
2666 09EF 50          CHK_VIDEO:          MOV    CX,0C000H ; START OF FEATURE I/O ROM
2667 09F0 1E          CHK_VIDEO01:        PUSH  AX      ; SAVE THE CONFIGURATION
2668 09F1 57          PUSH  DS      ; SAVE THE DATA SEGMENT
2669 09F2 00          PUSH  DI      ; SAVE COMPARE REGISTER
2670 09F4 00 D9          MOV    DS,CX      ; GET ROM SEGMENT
2671 09F4 BF A455      MOV    DX,104A455H ; GET THE PRESENCE SIGNATURE
2672 09F7 2B D8          SUB   BX,BX      ; CLEAR INDEX POINTER
2673 09F9 8B 07          MOV    AX,[BX]      ; GET THE FIRST 2 LOCATIONS
2674 09FB 3B C7          CMP    AX,DI      ; IS THE VIDEO FEATURE ROM PRESENT?
2675 09FD 5F          POP    DI      ; RESTORE WORD REGISTER
2676 09FF 1F          POP    DS      ; RESTORE DATA REGISTER
2677 09FF 58          POP    AX      ; GET THE CONFIGURATION
2678 0A00 74 0C          JZ    CHK_VIDEO02 ; GO IF VIDEO ROM INSTALLED
2679 0A02 81 C1 0080      ADD    CX,080H      ; POINT TO NEXT 2K BLOCK
2680 0A06 81 F7 C800      CMP    CX,0C800H ; TOP OF VIDEO ROM AREA YET?

```

```

2681 0A0A 0C E3      JL     CHK_VIDEO1      : TRY AGAIN
2682 0A0C 23 C9      AND    CX,CX          : SET NON ZERO FLAG
2683 0A0E              CHK_VIDEO02      : RETURN TO CALLER
2684 0A0E C3
2685
2686 ;----- CMOS VIDEO BITS NON ZERO (CHECK FOR PRIMARY DISPLAY AND NO VIDEO ROM)
2687
2688 0A0F              MOS_OK_1:      CALL   CHK_VIDEO      : IS THE VIDEO ROM INSTALLED?
2689 0A0F E8 09EC R    JZ    BAD_MOS        : ; WRONG CONFIGURATION IN CONFIG BYTE
2690 0A12 T4 26
2691
2692 0A14 8A C4      MOV    AL,AH          : RESTORE CONFIGURATION
2693 0A16 F6 06 0012 R 40  TEST   @MFG_TST,DSP_JMP : ; CHECK FOR DISPLAY JUMPER
2694 0A1B T4 0A      JZ    MOS_OK_2        : ; GO IF COLOR CARD IS PRIMARY DISPLAY
2695
2696 ;----- MONOCHROME CARD IS PRIMARY DISPLAY (NO JUMPER INSTALLED)
2697
2698 0A1D 24 30      AND    AL,30H          : INSURE MONOCHROME IS PRIMARY
2699 0A1F 3C 30      CMP    AL,30H          : CONFIGURATION OK?
2700 0A21 75 17      JNZ    BAD_MOS        : ; GO IF NOT
2701 0A23 8A C4      MOV    AL,AH          : RESTORE CONFIGURATION
2702 0A25 EB 08      JMP    SHORT MOS_OK  : ; USE THE CONFIGURATION BYTE FOR DISPLAY
2703
2704
2705 ;----- COLOR CARD
2706 0A27              MOS_OK_2:      AND    AL,30H          : STRIP UNWANTED BITS
2707 0A27 24 30      CMP    AL,30H          : MUST NOT BE MONO WITH JUMPER INSTALLED
2708 0A29 3C 30      MOV    AL,AH          : RESTORE CONFIGURATION
2709 0A2B 8A C4      JZ    BAD_MOS        : ; GO IF YES
2710 0A2D T4 0B
2711
2712 ;----- CONFIGURATION MUST HAVE AT LEAST ONE DISKETTE
2713
2714 0A2F A8 01      MOS_OK:      TEST   AL,01H          : MUST HAVE AT LEAST ONE DISKETTE
2715 0A31 75 26      JNZ    NORMAL_CONFIG  : ; GO SET CONFIGURATION IF OK
2716 0A33 F6 06 0012 R 20  TEST   @MFG_TST,MFG_LOOP : ; EXCEPT IF MFG JUMPER IS INSTALLED
2717 0A38 74 1F      JZ    NORMAL_CONFIG  : ; GO IF INSTALLED
2718
2719 ;----- MINIMUM CONFIGURATION WITH BAD CMOS OR NON VALID VIDEO
2720
2721 0A3A              BAD_MOS:      MOV    AX,CMOS_DIAG+NMI : GET THE DIAGNOSTIC STATUS
2722 0A3A E8 008E      CALL   CMOS_READ      : ; WAS BATTERY DEFECTIVE OR BAD CHECKSUM
2723 0A3D E8 0000 E    TEST   AL,BAD_BAT+BAD_CKSUM : ; GO IF YES
2724 0A40 A8 C0      JNZ    BAD_MOS        : ; SET THE MINIMUM CONFIGURATION FLAG
2725 0A42 75 03
2726
2727 0A44 E8 0000 E    CALL   CONFIG_BAD      : ; CHECK FOR VIDEO ROM
2728 0A44 80 01      MOV    AL,01H          : ; DISKETTE ONLY
2729 0A47 E8 09EC R    JZ    NORMAL_CONFIG  : ; ; GO IF VIDEO ROM PRESENT
2730 0A4A B0 01
2731 0A4C T4 0B
2732
2733 0A4E F6 06 0012 R 40  TEST   @MFG_TST,DSP_JMP : ; CHECK FOR DISPLAY JUMPER
2734 0A52 B0 11      MOV    AL,11H          : ; DEFAULT TO 40X25 COLOR
2735 0A55 74 02      JZ    NORMAL_CONFIG  : ; ; GO IF JUMPER IS INSTALLED
2736
2737 0A57 B0 31      MOV    AL,31H          : ; DISKETTE / B/W DISPLAY 80X25
2738
2739
2740 ;----- CONFIGURATION AND MFG MODE
2741
2742
2743 0A59              NORMAL_CONFIG: TEST   @MFG_TST,MFG_LOOP : ; IS THE MANUFACTURING JUMPER INSTALLED
2744 0A59 F6 06 0012 R 20  JNZ    NORMT          : ; GO IF NOT
2745 0A5E 75 02      AND    AL,03EH          : ; STRIP DISKETTE FOR MFG TEST
2746 0A60 24 3E
2747
2748 0A62 2A E4      NORMT:      SUB   AH,AH          : ; SAVE SWITCH INFORMATION
2749 0A64 A3 0010 R    MOV    @EQUIP_FLAG,AX : ; ; BYPASS IF SOFT RESET
2750 0A67 B1 3E 0072 R 1234  CMP    @RESET_FLAG,1234H
2751 0A6D T4 2C      JZ    E6
2752
2753 ;----- GET THE FIRST SELF TEST RESULTS FROM KEYBOARD
2754
2755 0A6F B0 60      MOV    AL,WRITE_8042_LOC : ; ; ENABLE KEYBOARD
2756 0A71 E8 0396 R    CALL   C8042          : ; ; ISSUE WRITE BYTE COMMAND
2757 0A74 B0 4D      MOV    AL,4DH          : ; ; ENABLE OUTPUT BUFFER FULL INTERRUPT,
2758 0A76 E6 60      OUT   PORT_A,AL      : ; ; SET SYSTEM FLAG, PC I COMPATIBILITY,
2759 0A76 E6 60      SUB   CX,CX          : ; ; INHIBIT OVERRIDE, ENABLE KEYBOARD
2760
2761 0A78 2B C9      CALL   C42_1          : ; ; WAIT FOR COMMAND ACCEPTED
2762 0A7A E8 039B R    MOV    CX,07FFH         : ; ; SET LOOP COUNT FOR APPROXIMATELY 100MS
2763
2764 0A7D B9 7FFF      TST6: IN   AL,STATUS_PORT : ; ; RD RESPOND
2765 0A7E 00 00          TEST   AL,_OUT_BUF_FULL : ; ; WAIT FOR OUTPUT BUFFER FULL
2766 0A7F 00 00          LOOPZ  TST6          : ; ; TRY AGAIN IF NOT
2767
2768 0A80 E4 64      PUSHF
2769 0A82 A8 01      MOV    AL,DIS_KBD      : ; ; SAVE FLAGS
2770 0A84 E1 FA      CALL   C8042          : ; ; DISABLE KEYBOARD
2771 0A85 9D          POPF
2772 0A89 E8 0396 R    JZ    E6
2773 0A8C 9D          CALL   C8042          : ; ; ISSUE THE COMMAND
2774 0A8D T4 0C      POPF
2775
2776 0A8F E4 60      IN    AL,PORT_A      : ; ; RESTORE FLAGS
2777 0A91 A2 0072 R    MOV    BYTE PTR @RESET_FLAG,AL : ; ; CONTINUE WITHOUT RESULTS
2778
2779 ;----- CHECK FOR MFG REQUEST
2780
2781 0A94 3C 65      CMP    AL,065H          : ; ; GET INPUT FROM KEYBOARD
2782 0A96 75 03      JNE    E6
2783 0A98 E9 0C27 R    JMP    MFG_BOOT        : ; ; TEMPORARY SAVE FOR AA RECEIVED
2784
2785 ;----- TEST.14
2786 ;----- INITIALIZE AND START CRT CONTROLLER (6845)
2787 ;----- TEST VIDEO READ/WRITE STORAGE.
2788 ;----- DESCRIPTION
2789 ;----- RESET THE VIDEO ENABLE SIGNAL.
2790 ;----- SELECT ALPHANUMERIC MODE, 40 * 25, B & W.
2791 ;----- READ/WRITE DATA PATTERNS TO MEMORY. CHECK
2792 ;----- STORAGE ADDRESSABILITY.
2793 ;----- ERROR = 1 LONG AND 2 SHORT BEEPS
2794

```



```

2909
2910      ;----- CHECK HORIZONTAL LINE
2911
2912 0B44 B1 03      E16: MOV CL,3          ; GET NEXT BIT TO CHECK
2913 0B46 D2 EC      SHR AH,CL
2914 0B48 75 E4      JNZ E12          ; CONTINUE
2915 0B4A
2916 0B4A 58
2917 0B4B B4 00      POP AX
2918 0B4D CD 10      MOV AH,0
2919
2920      ;----- CHECK FOR THE ADVANCED VIDEO CARD
2921
2922 0B4F BA C000      E18_I: MOV DX,0C000H ; SET THE LOW SEGMENT VALUE
2923 0B52
2924 0B53 B0 23      E18_A: MOV AL,23H
2925 0B54 E6 80      OUT MFG_PORT,AL
2926 0B56 8E DA      MOV DS,DX
2927 0B58 57      PUSH DI
2928 0B59 BF AA55      MOV DI,0AA55H
2929 0B5C 2B DB      SUB BX,BX
2930 0B5D 00 07      MOV AX,[BX]
2931 0B60 2B C7      CMP AX,01
2932 0B62 5F      POP DI
2933 0B63 75 05      JNZ E18B
2934
2935 0B65 E8 0000 E      CALL ROM_CHECK
2936 0B66 EB 04      JMP SHORT E18C ; GO SCAN MODULE
2937 0B6A
2938 0B6A B1 C2 0080      E18B: ADD DX,0080H ; POINT TO NEXT 2K BLOCK
2939 0B6E
2940 0B6E B1 FA C000      E18C: CMP DX,0C000H ; TOP OF VIDEO ROM AREA YET?
2941 0B72 TC DE      JL E18A ; GO SCAN FOR ANOTHER MODULE
2942
2943 0B74 B0 24      MOV AL,24H
2944 0B76 E6 80      OUT MFG_PORT,AL
2945
2946 0B78 E9 0000 E      JMP POST2 ; GO TO NEXT TEST
2947
2948
2949      ;----- CRT ERROR SET MFG CHECKPOINT AND ERROR BEEP
2950
2951 0B7B E8 0000 E      E17: CALL DDS ; POINT TO DATA
2952
2953      ;----- CHECKPOINT 0C = MONOCHROME FAILED
2954
2955 0B7E C6 06 0015 R 0C      MOV OMFG_ERR_FLAG,0CH ; <><> CRT ERROR CHECKPOINT 0C <><>
2956 0B83 B0 3E 0072 R 64      CMP BYTE PTR _RESET_FLAG,064H ; IS THIS A MFG REQUEST?
2957 0B88 74 0D      JZ E19          ; BY PASS ERROR BEEP IF YES
2958 0B8A F6 06 0012 R 20      TEST OMFG_TST,MFG_LOOP ; IS THE MFG LOOP JUMPER INSTALLED?
2959 0B8F T4 06      JZ E19          ; BY PASS ERROR BEEP IF YES
2960 0B90 B0 0102      MOV DX,102H
2961 0B94 EB 0000 E      CALL ERR_BEEP ; GO BEEP SPEAKER
2962 0B97
2963 0B97 1E      E19: PUSH DS ; GET THE CURRENT VIDEO
2964 0B98 A1 0010 R      MOV AX,0EQUIP_FLAG ; STRIP OTHER BITS
2965 0B98 24 30      AND AL,30H ; IS IT MONOCHROME ?
2966 0B9D 3C 30      CMP AL,30H ; GO IF YES
2967 0B9F T4 30      TRY_COLOR
2968
2969      ;----- COLOR FAILED TRY MONOCHROME - CHECKPOINT 0D = COLOR FAILED
2970
2971 0B91 C6 06 0015 R 0D      MOV OMFG_ERR_FLAG,0DH ; <><> CRT ERROR CHECKPOINT 0D <><>
2972
2973 0B96 BA 03B8      MOV DX,3B8H ; DISABLE B/W
2974 0B99 B0 01      MOV AL,1
2975 0BAB EE      OUT DX,AL ; OUTPUT THE DISABLE
2976 0BAC BB B000      MOV BX,0B000H ; CHECK FOR MONOCHROME VIDEO MEMORY
2977 0BAC 8E DB      MOV DS,BX
2978 0BAC 7F A555      MOV AX,0AA55H ; WRITE AN AA55
2979 0B94 B9 DB      SUB BX,[BX] ; TO THE FIRST LOCATION
2980 0B96 B9 07      MOV [BX],AX ; ALLOW BUS TO SETTLE
2981 0B98 EB 00      JMP $+2 ; READ THE FIRST LOCATION
2982 0B98 BB 07      MOV AX,[BX] ; IS THE MONOCHROME VIDEO CARD THERE?
2983 0B9C 3D AA55      CMP AX,0AA55H ; RESTORE THE DATA SEGMENT
2984 0B9F 1F      POP DS ; GO IF NOT
2985 0B99 74 55      INT 3 ; TURN ON MONOCHROME BITS IN EQUIP FLAG
2986 0B92 81 0E 0010 R 0030      OR AX,0EQUIP_FLAG,30H ; ENABLE VIDEO
2987 0B98 A1 0010 R      MOV AX,0EQUIP_FLAG
2988 0B9B 2A E4      SUB AH,AH
2989 0BCD CD 10      INT VIDEO
2990 0BCF EB 34      JMP SHORT E17_! ; CONTINUE
2991
2992
2993      ;----- MONOCHROME FAILED TRY COLOR
2994 0BD1
2995 0BD1 B0 01      TRY_COLOR: MOV AL,01H ; SET MODE COLOR 40X25
2996 0BD3 2A E4      INT VIDEO
2997 0BD5 00 10      MOV DX,3D0H ; DISABLE COLOR
2998 0BD7 BA 03D8      MOV AL,0
2999 0BDA B0 00      OUT DX,AL ; OUTPUT THE DISABLE
3000 0BDC EE      MOV BX,0B000H ; CHECK FOR COLOR VIDEO MEMORY
3001 0BD9 BB B800      MOV DS,BX
3002 0BDE 8E DB      MOV AX,0AA55H ; WRITE AN AA55
3003 0B95 B9 DB      SUB BX,[BX] ; TO THE FIRST LOCATION
3004 0B9E B9 07      MOV [BX],AX ; ALLOW BUS TO SETTLE
3005 0B99 EB 00      JMP $+2 ; READ THE FIRST LOCATION
3006 0B9B BB 07      MOV AX,[BX] ; IS THE COLOR VIDEO CARD THERE?
3007 0B9C 3D AA55      CMP AX,0AA55H ; RESTORE THE DATA SEGMENT
3008 0B9F 1F      POP DS ; GO IF NOT
3009 0B99 75 24      INT 3 ; TURN OFF VIDEO BITS
3010 0B9F 15 24      OR 0EQUIP_FLAG,0FFCFH ; SET COLOR 40X24
3011 0B93 B1 26 0010 R FFFC
3012 0B99 B1 0E 0010 R 0010      OR 0EQUIP_FLAG,10H
3013 0BFF B0 01      MOV AL,01H
3014 0B99 2A E4      SUB AH,AH
3015 0B93 CD 10      INT VIDEO
3016 0C05
3017 0C05 58      E17_1: POP AX ; SET NEW VIDEO TYPE ON STACK
3018 0C06 A1 0010 R      MOV AX,0EQUIP_FLAG
3019 0C09 24 30      AND AL,30H ; IS IT THE B/W?
3020 0C0B 3C 30      CMP AL,30H
3021 0C0D 2A C0      SUB AL,AL ; GO IF YES
3022 0C0F 74 02      JZ E17_2

```

```
3023 0C11 FE C0           INC    AL          ; INITIALIZE FOR 40X25
3024 0C13                 E17_2: PUSH   AX
3025 0C13 50              E17_4: JMP    E18
3026 0C14
3027 0C14 E9 0B4A R
3028
3029
3030
3031 0C17                 E17_3: PUSH   DS
3032 0C17 1E
3033 0C1A 28 C0
3034 0C1A 8E D8
3035 0C1C BF 0040 R
3036 0C1F C7 05 0000 E
3037 0C23 1F
3038 0C24 E9 0B4F R

          ;----- BOTH VIDEO CARDS FAILED SET DUMMY RETURN IF RETRACE FAILURE
          ;----- SET DS SEGMENT TO 0
          ;----- SET INTERRUPT 10H TO DUMMY
          ;----- RETURN IF NO VIDEO CARD
          ;----- BYPASS REST OF VIDEO TEST

          ;-----
```

3039  
3040  
3041  
3042  
3043  
3044  
3045  
3046  
3047  
3048  
3049  
3050  
3051  
3052  
3053  
3054 OC27  
3055 OC27 B4 DD  
3056 OC29 E8 0000 E  
3057  
3058  
3059  
3060 OC2C 68 ---- R  
3061 OC2F 01  
3062 OC30 B9 0018  
3063 OC30 8C C8  
3064 OC35 8E D8  
3065 OC31 BE 0000 E  
3066 OC3A BF 0020 R  
3067 OC3D  
3068 OC3D A5  
3069 OC3E AB  
3070 OC3F E2 FC  
3071  
3072  
3073  
3074 OC41 B9 0008  
3075 OC44 BE 0000 E  
3076 OC47 BF 01C0 R  
3077 OC4A  
3078 OC4A A5  
3079 OC4B AB  
3080 OC4C E2 FC  
3081  
3082  
3083  
3084  
3085 OC4E 06  
3086 OC4F 1F  
3087 OC50 CT 06 0008 R 0000 E  
3088 OC56 CT 06 0014 R 0000 E  
3089 OC5C CT 06 0062 R F600  
3090  
3091  
3092  
3093 OC62 B0 60  
3094 OC64 E8 0396 R  
3095 OC67 B0 09  
3096 OC68 E6 60  
3097  
3098 OC68 E8 OCBD R  
3099 OC68 B8 F8  
3100 OC70 E8 OCBD R  
3101 OC73 EA E8  
3102 OC74 EA CF  
3103 OC77 EA F8  
3104 OC78 BF 0500 R  
3105 OC7B  
3106 OC7B E4 64  
3107 OC7D A8 01  
3108 OC7E 00 FA  
3109 OC81 E4 60  
3110 OC83 AA  
3111 OC84 E6 80  
3112 OC84 E2 F3  
3113  
3114 OC88 EA 0500 ---- R  
3115 OC8D  
3116 OC8D  
3117 OC8D E4 64  
3118 OC8E A8 01  
3119 OC91 E1 FA  
3120  
3121 OC93 E4 60  
3122 OC95 C3  
3123  
3124 OC96  
3125 OC96  
3126

PAGE  
-----  
: MANUFACTURING BOOT TEST CODE ROUTINE  
: LOAD A BLOCK OF TEST CODE THROUGH THE KEYBOARD PORT FOR MANUFACTURING  
: TESTS.  
: THIS ROUTINE WILL LOAD A TEST (MAX LENGTH=FFFF) THROUGH THE KEYBOARD  
: PORT. CODE WILL BE LOADED AT LOCATION 0000:0500. AFTER LOADING,  
: CONTROL WILL BE TRANSFERRED TO LOCATION 0000:0500. THE STACK WILL  
: BE LOCATED AT 0000:0400. THIS ROUTINE ASSUMES THAT THE FIRST 2 BYTES  
: TRANSFERRED CONTAIN THE COUNT OF BYTES TO BE LOADED  
: (BYTE I=COUNT LOW, BYTE 2=COUNT HI.)  
-----  
:----- DEGATE ADDRESS LINE 20  
MFG\_BOOT:  
    MOV AH,DISABLE\_BIT20 ; DEGATE COMMAND FOR ADDRESS LINE 20  
    CALL GATE\_A20 ; ISSUE TO KEYBOARD ADAPTER AND CLI  
-----  
:----- SETUP HARDWARE INTERRUPT VECTOR TABLE LEVEL 0-7 AND SOFTWARE INTERRUPTS  
MFG\_B1:  
    PUSH ABS0 ; SET ES SEGMENT REGISTER TO ABS0  
    POP ES  
    MOV CX,24 ; GET VECTOR COUNT  
    MOV AX,CS ; GET THE CURRENT CODE SEGMENT VALUE  
    MOV DS,AX ; SETUP DS SEGMENT REGISTER TO  
    MOV SI,OFFSET VECTOR\_TABLE ; POINT TO THE ROUTINE ADDRESS TABLE  
    MOV DI,OFFSET @INT\_PTR ; SET DESTINATION TO FIRST USED VECTOR  
MFG\_B2:  
    MOVSW ; MOVE ONE ROUTINE OFFSET ADDRESS  
    STOSW ; INSERT CODE SEGMENT VALUE  
    LOOP MFG\_B1 ; MOVE THE NUMBER OF ENTRIES REQUIRED  
-----  
:----- SETUP HARDWARE INTERRUPT VECTORS LEVEL 8-15 (VECTORS START AT INT 70 H)  
MFG\_B2:  
    MOV CX,08 ; GET VECTOR COUNT  
    MOV SI,OFFSET SLAVE\_VECTOR\_TABLE ; GET THE CURRENT CODE SEGMENT VALUE  
    MOV DI,OFFSET @SLAVE\_INT\_PTR ; SET DESTINATION TO FIRST USED VECTOR  
MFG\_B3:  
    MOVSW ; MOVE ONE ROUTINE OFFSET ADDRESS  
    STOSW ; INSERT CODE SEGMENT VALUE  
    LOOP MFG\_B2 ; MOVE THE NUMBER OF ENTRIES REQUIRED  
-----  
:----- SET UP OTHER INTERRUPTS AS NECESSARY  
ASSUME DS:ABS0,ES:ABS0  
MFG\_B4:  
    PUSH ES ; ES= ABS0  
    POP DS ; SET DS TO ABS0  
    MOV WORD PTR @NMI\_PTR,OFFSET NMI\_INT ; NMI INTERRUPT  
    MOV WORD PTR @INT5\_PTR,OFFSET PRINT\_SCREEN ; PRINT SCREEN  
    MOV WORD PTR @BASIC\_PTR+2,0F600H ; CASSETTE BASIC SEGMENT  
-----  
:----- ENABLE KEYBOARD PORT  
MFG\_B5:  
    MOV AL,60H ; WRITE 8042 MEMORY LOCATION 0  
    CALL C8042 ; ISSUE THE COMMAND  
    MOV AL,0000001B ; SET INHIBIT OVERRIDE/ENABLE OBF  
    OUT PORT\_A,AL ; INTERRUPT AND NOT PC COMPATIBLE  
MFG\_B6:  
    CALL MFG\_B4 ; GET COUNT LOW  
    MOV BH,AL ; SAVE IT  
    CALL MFG\_B4 ; GET COUNT HI  
    MOV CH,AL ; CX NOW HAS COUNT  
    MOV CL,BH ; SET DIRECTION FLAG TO INCREMENT  
    CLD ; SET TARGET OFFSET (DS=0000)  
    DI,OFFSET @MFG\_TEST\_RTN ; SET TARGET  
MFG\_B7:  
    IN AL,STATUS\_PORT ; GET 8042 STATUS PORT  
    TEST AL,OUT\_BUF\_FULL ; KEYBOARD REQUEST PENDING?  
    JZ MFG\_B3 ; LOAD ALL DATA PRESENT  
    IN AL,PORT\_A ; GET DATA  
    STOSB ; STORE IT  
    OUT MFG\_PORT,AL ; DISPLAY CHARACTER AT MFG PORT  
    LOOP MFG\_B3 ; LOOP TILL ALL BYTES READ  
    JMP @MFG\_TEST\_RTN ; FAR JUMP TO CODE THAT WAS JUST LOADED  
MFG\_B8:  
    IN AL,STATUS\_PORT ; CHECK FOR OUTPUT BUFFER FULL  
    TEST AL,OUT\_BUF\_FULL ; HANG HERE IF NO DATA AVAILABLE  
    MFG\_B4 ;  
    LOOPZ ;  
    IN AL,PORT\_A ; GET THE COUNT  
    RET  
POST1 ENDP  
CODE ENDS  
END

```

PAGE 118,121
TITLE TEST2 ---- 06/10/85 POST TESTS AND INITIALIZATION ROUTINES
3
4
5
6 0000      .LIST
7 0000      CODE SEGMENT BYTE PUBLIC
8 0000      PUBLIC IC C21
9 0000      PUBLIC IC POST2
10 0000     PUBLIC IC SHUT2
11 0000     PUBLIC IC SHUT3
12 0000     PUBLIC IC SHUT4
13 0000     PUBLIC IC SHUT6
14 0000     PUBLIC IC SHUT7
15
16      EXTRN BLINK_INT:NEAR
17      EXTRN C8042:NEAR
18      EXTRN CMOS_READ:NEAR
19      EXTRN CMOS_WRITE:NEAR
20      EXTRN COM1_T_BAD:NEAR
21      EXTRN D1:NEAR
22      EXTRN D2:NEAR
23      EXTRN DDS:NEAR
24      EXTRN D1SK_SETUP:NEAR
25      EXTRN DSKETTE_SETUP:NEAR
26      EXTRN ERASE:NEAR
27      EXTRN E_MSC:NEAR
28      EXTRN F3D:NEAR
29      EXTRN F3D1:NEAR
30      EXTRN GATE_A20:NEAR
31      EXTRN HD_INT:NEAR
32      EXTRN KBD_RESET:NEAR
33      EXTRN NM1:NEAR
34      EXTRN OBF_42:NEAR
35      EXTRN POST3:NEAR
36      EXTRN PRINT_SCREEN:NEAR
37      EXTRN PROC_SHUTDOWN:NEAR
38      EXTRN PROT_PRT_HEX:NEAR
39      EXTRN PROT_PRT:NEAR
40      EXTRN P_MSG:NEAR
41      EXTRN ROM_CHECK:NEAR
42      EXTRN ROM_CHECKSUM:NEAR
43      EXTRN SEEK:NEAR
44      EXTRN SET_TOD:NEAR
45      EXTRN SLAVE_INTERRUPT_TABLE:NEAR
46      EXTRN SMDATA:NEAR
47      EXTRN START_1:NEAR
48      EXTRN STGTST_CNT:NEAR
49      EXTRN SYSINIT1:NEAR
50      EXTRN VECTOR_TABLE:NEAR
51      EXTRN WAITF:NEAR
52      EXTRN XPC_BYT:NEAR
53
54      EXTRN E101:NEAR      ; 101 ERROR CODE - INTERRUPT FAILURE
55      EXTRN E102:NEAR      ; 102 ERROR CODE - TIMER FAILURE
56      EXTRN E103:NEAR      ; 103 ERROR CODE - TIMER INTERRUPT
57      EXTRN E104:NEAR      ; 104 ERROR CODE - PROTECTED MODE ERROR
58      EXTRN E105:NEAR      ; 105 ERROR CODE - 040 COMMAND FAILURE
59      EXTRN E106:NEAR      ; 106 ERROR CODE - CONVERTING LOGIC
60      EXTRN E107:NEAR      ; 107 ERROR CODE - NMI ERROR
61      EXTRN E108:NEAR      ; 108 ERROR CODE - TIMER BUS ERROR
62      EXTRN E109:NEAR      ; 109 ERROR CODE - MEMORY SELECT ERROR
63      EXTRN E110:NEAR      ; 110 ERROR CODE - GATE BY TERTIARY
64      EXTRN E111:NEAR      ; 111 ERROR CODE - GATE_CHECKSUM/CONFIG
65      EXTRN E163:NEAR      ; 163 ERROR CODE - BAD REAL TIME CLOCK
66      EXTRN E164:NEAR      ; 164 ERROR CODE - MEMORY SIZE WRONG
67      EXTRN E201:NEAR      ; 201 ERROR CODE - MEMORY DATA ERROR
68      EXTRN E202:NEAR      ; 202 ERROR CODE - MEMORY ADDRESS ERROR
69      EXTRN E203:NEAR      ; 203 ERROR CODE - SEGMENT ADDRESS ERROR
70      EXTRN E301:NEAR      ; 301 ERROR CODE - KEYBOARD/PLANNER ERROR
71      EXTRN E302:NEAR      ; 302 ERROR CODE - DCX IS ON
72      EXTRN E303:NEAR      ; 303 ERROR CODE - KEYBOARD/PLANNER ERROR
73      EXTRN E401:NEAR      ; 401 ERROR CODE - MONOCHROME ADAPTER
74      EXTRN E501:NEAR      ; 501 ERROR CODE - COLOR ADAPTER
75      EXTRN E601:NEAR      ; 601 ERROR CODE - DISKETTE ADAPTER
76
77
78      ; TEST.17
79      ; 8259 INTERRUPT CONTROLLER TEST
80      ; DESCRIPTION
81      ; READ/WRITE THE INTERRUPT MASK REGISTER (IMR)
82      ; WITH ALL ONES AND ZEROS, ENABLE SYSTEM
83      ; INTERRUPTS, MASK DEVICE INTERRUPTS OFF, CHECK
84      ; FOR HOT INTERRUPTS (UNEXPECTED).
85
86
87      ASSUME CS:CODE,DS:DATA
88
89 0000      POST2 PROC NEAR
90
91 0000 B0 0A      C21: MOV AL,10      ; LINE FEED ON DISPLAY
92 0002 E8 0000 E    CALL PRT_HEX
93 0005 E8 0000 E    CALL DDS
94
95      ;----- CLEAR ERROR FLAG REGISTER (BP) <> 0 FLAGS ERROR
96
97 0008 2B ED      SUB BP,BP      ; CLEAR (BP) REGISTER AS ERROR FLAG REG
98
99      ;----- TEST THE INTERRUPT MASK REGISTER REGISTERS
100
101 000A FA      C21A: CL1      ; TURN OFF INTERRUPTS
102 000B B0 00      MOV AL,0      ; SET INTERRUPT MASK REGISTER TO ZERO
103 000D E6 21      OUT INTA01,AL
104 000F E6 A1      OUT INTB01,AL      ; SEND TO 2ND INTERRUPT CONTROLLER ALSO
105 0011 E9 D0      JMP $+2
106 0013 E9 21      IN AL,INTA01      ; READ INTERRUPT MASK REGISTER
107 0015 E9 D0      MOV AH,AL      ; SAVE RESULTS
108 0017 E4 A1      IN AL,INTB01      ; READ 2ND INTERRUPT MASK REGISTER
109
110 0019 0A E0      OR AH,AL      ; BOTH IMR = 0?
111 001B 75 2C      JNZ D6
112
113 001D B0 25      MOV AL,25H      ; <><><><><><><><>
114 001E F6 80      OUT NEC_PORT_AI      ; <><> CHECKPOINT_25 <><>

```

```

115 0021 B0 FF      MOV    AL,0FFH      ; 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 0028 E6 21      IN    AL,INTA01   ; READ INTERRUPT MASK REGISTER
121 002B E6 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,I      ; ALL IMR BITS ON?
125 0032 75 15      JNZ   D6          ; NO - GO TO ERR ROUTINE
126
127 ;----- CHECK FOR HOT INTERRUPTS
128
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 ; <><> CHECKPOINT 26 <><>
135
136 003B FB          STI
137 003C B9 19E4    MOV   CX,6628    ; ENABLE EXTERNAL INTERRUPTS
138 003F EB 0000 E  CALL  WAITF     ; WAIT 100 MILLISECONDS FOR ANY
139 0042 80 3E 006B R 00  CMP   *INTR_FLAG,00H ; INTERRUPTS THAT OCCUR
140 0047 74 0D      JZ    D7          ; DID ANY INTERRUPTS OCCUR?
141
142 0049 C6 06 0015 R 05 D6: MOV   *MFG_ERR_FLAG,05H ; NO - GO TO NEXT TEST
143
144 004E BE 0000 E  MOV   SI,OFFSET E101 ; <><><><><><><><><><>
145 0051 E8 0000 E  D6A: CALL  E_MSG   ; <><> CHECKPOINT 5 <><>
146 0054 FA          CALL  CLR_I
147 0055 F4          HLT
148
149 ;----- CHECK THE CONVERTING LOGIC
150
151 0056 B0 27      D7: MOV   AL,27H      ; <><><><><><><><><><>
152 0058 E6 80      OUT  MFG_PORT,AL ; <><> CHECKPOINT 27 <><>
153
154 005A B8 AA55    MOV   AX,0AA55H    ; WRITE A WORD
155 005D E7 82      OUT  MFG_PORT+2,AX ; GET THE FIRST BYTE
156 005E E4 82      IN   AL,MFG_PORT+2 ; SAVE IT
157 0061 E4 04      XCHG AL,AL      ; GET THE SECOND BYTE
158 0063 E4 83      IN   AL,MFG_PORT+3 ; IS IT OK?
159 0065 3D 55AA    CMP   AX,55AAH    ; GO IF YES
160 0068 74 05      JZ    D7_A
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
168 0070 B0 00      D7_A: MOV   AL,CMOS_REG_D ; TURN ON NMI
169 0071 E6 70      OUT  CMOS_PORT,AL ; ADDRESS DEFAULT READ ONLY REGISTER
170 0073 B9 0007    MOV   CX,1          ; DELAY COUNT FOR 100 MICROSECONDS
171 0076 E8 0000 E  CALL  WAITF     ; WAIT FOR HOT NMI TO PROCESS
172 0079 B0 8D      MOV   AL,CMOS_REG_D+NMI ; TURN NMI ENABLE BACK OFF
173 007B E6 70      OUT  CMOS_PORT,AL
174 007D 80 3E 006B R 00  CMP   *INTR_FLAG,00H ; DID ANY INTERRUPTS OCCUR?
175 0082 74 09      JZ    D7_C          ; CONTINUE IF NOT
176
177 0084 B0 28      MOV   AL,28H      ; <><><><><><><><><><>
178 0086 E6 80      OUT  MFG_PORT,AL ; <><> CHECKPOINT 28 <><>
179
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      D7_C: MOV   AL,29H      ; <><><><><><><><><><>
186 008E E6 80      OUT  MFG_PORT,AL ; <><> CHECKPOINT 29 <><>
187 0091 E4 11      IN   AL,PORT_B   ; GET CURRENT SETTING OF PORT
188 0093 80 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,1010000B ; SELECT TIM 2,LSB,MSB,BINARY,MODE 0
193 009A E6 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 80 C4      MOV   AL,AH      ; WRITE TIMER 2 COUNT - MSB
199 00A8 E4 42      OUT  TIMER+2,AL ; I/O DELAY
200 00AA EB 00      JMP   $+2          ; GET THE LSB
201 00AC E4 42      IN   AL,TIMER+2 ; SAVE IT
202 00AE 86 E0      XCHG AH,AL
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,055AAH    ; BUS OK?
206 00B7 74 05      JZ    D7_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 ;----- TEST.18
212 ;----- 8254 TIMER CHECKOUT
213 ;----- DESCRIPTION
214 ;----- VERIFY THAT THE SYSTEM TIMER (0) DOESN'T COUNT
215 ;----- TOO FAST OR TOO SLOW.
216
217
218
219 00BE B0 2A      D7_D: MOV   AL,2AH      ; <><><><><><>
220 00C0 E6 80      OUT  MFG_PORT,AL ; <><> CHECKPOINT 2A <><>
221 00C2 FA          CLC
222 00C3 EB FE      MOV   AL,0FEH
223 00C4 E6 21      OUT  INT01,AL   ; MASK ALL INTERRUPTS EXCEPT LEVEL 0
224 00C7 B0 10      MOV   AL,0001000B ; WRITE THE 8259 IMR
225 00C9 E6 43      OUT  TIMER+3,AL ; SELECT TIM 0,LSB, MODE 0, BINARY
226 00CB B9 002C    MOV   CX,2CH    ; WRITE TIMER CONTROL MODE REGISTER
227
228 00CE EB 00      JMP   $+2          ; SET PROGRAM LOOP COUNT
229

```



```

343 0170 26: C7 06 005A 0000 POP ES
344 0171 26: C6 06 005C 00 MOV ES:SS TEMP.BASE_LO_WORD,0
345 0172 26: C6 06 005C 00 MOV BYTE PTR ES:(SS TEMP.BASE_HI_BYTE),0
346 017E BE 0058 MOV SI,SS TEMP
347 0181 8E D6 MOV SS,SI
348 0183 BC FFFD MOV SP,MAX_SEG_LEN-2
349
350 ;----- DATA SEGMENT TO SYSTEM DATA AREA
351
352 0186 6A 18 PUSH BYTE PTR RSDA_PTR ; POINT TO DATA AREA
353 0188 1F POP DS
354
355 0189 B0 80 MOV AL,PARITY_CHECK ; SET CHECK PARITY
356 018B E6 87 OUT DMA_PAGE+6,AL ; SAVE WHICH CHECK TO USE
357
358 ;----- PRINT 64 K BYTES OK
359
360 018D B8 0040 MOV AX,64 ; STARTING AMOUNT OF MEMORY OK
361 0190 E8 099F R CALL PRT_OK ; POST 65K OK MESSAGE
362
363
364 ;----- GET THE MEMORY SIZE DETERMINED (PREPARE BX AND DX FOR BAD CMOS)
365 0193 B8 B0B1 MOV AX,(CMOS_U_M_S_LO+NMI)*4+CMOS_U_M_S_HI+NMI
366 0196 E8 0000 E CALL CMOS_READ ; HIGH BYTE
367 0199 86 E0 XCHG AH,AL ; SAVE HIGH BYTE
368 019B E8 0000 E CALL CMOS_READ ; LOW BYTE
369 019C E8 0013 R MOV BX,BX ; LOW BYTES OF BASE MEMORY SIZE
370 01A2 B8 D3 MOV DX,BX ; SAVE BASE MEMORY SIZE
371 01A4 03 D8 ADD BX,AX ; SET TOTAL MEMORY SIZE
372
373 ;----- IS CMOS GOOD?
374
375 01A6 B0 8E MOV AL,CMOS_DIAG+NMI ; DETERMINE THE CONDITION OF CMOS
376 01A8 E8 0000 E CALL CMOS_READ ; GET THE CMOS STATUS
377
378 01AB A8 C0 TEST AL,BAD_BAT+BAD_CKSUM ; CMOS OK?
379 01AD 74 02 JZ E2080 ; GO IF YES
380 01AF EB 5B JMP SHORT E20C ; DEFAULT IF NOT
381
382 ;----- GET THE BASE 0->640K MEMORY SIZE FROM CONFIGURATION IN CMOS
383 01B1 E20B0: MOV AX,(CMOS_B_M_S_LO+NMI)*4+CMOS_B_M_S_HI+NMI
384 01B4 E8 9596 CALL CMOS_READ ; HIGH BYTE
385 01B7 24 3F AND AL,03FH ; MASK OFF THE MANUFACTURING TEST BITS
386 01B9 E8 E0 XCHG AH,AL ; SAVE HIGH BYTE
387 01B8 E8 0000 E CALL CMOS_READ ; LOW BYTE OF BASE MEMORY SIZE
388 01B9 E8 3B D0 CMP DX,AX ; IS MEMORY SIZE GREATER THAN CONFIG?
389 01C0 74 13 JZ E20B1 ; GO IF EQUAL
391
392
393 ;----- SET MEMORY SIZE DETERMINE NOT EQUAL TO CONFIGURATION
394 01C2 50 PUSH AX ; SAVE AX
395 01C3 B8 8E8E MOV AX,X*(CMOS_DIAG+NMI) ; ADDRESS THE STATUS BYTE
396 01C6 E8 0000 E CALL CMOS_READ ; GET THE STATUS
397 01C9 0C 10 OR AL,W_MEMORY_SIZE ; SET CMOS FLAG
398 01CB E8 C4 XCHG AH,AL ; SAVE AL AND GET ADDRESS
399 01C9 E8 0000 E CALL CMOS_WRITE ; WRITE UPDATED STATUS
400 01D0 88 08 POP AX ; RESTORE AX
401 01D1 3B D0 CMP DX,AX ; IS MEMORY SIZE GREATER THAN CONFIG ?
402 01D3 77 37 JA E20C ; DEFAULT TO MEMORY SIZE DETERMINED ?
403 01D5
404 01D5 B8 D8 E20B1: MOV BX,AX ; SET BASE MEMORY SIZE IN TOTAL REGISTER
405 01D7 B8 D0 MOV DX,AX ; SAVE IN BASE SIZE REGISTER
406
407 ;----- CHECK MEMORY SIZE ABOVE 640K FROM CONFIGURATION
408
409 01D9 B8 9798 MOV AX,(CMOS_E_M_S_LO+NMI)*4+(CMOS_E_M_S_HI+NMI)
410 01DC E8 0000 E CALL CMOS_READ ; HIGH BYTE
411 01E0 86 E0 XCHG AH,AL ; SAVE HIGH BYTE
412 01E1 E8 0000 E CALL CMOS_READ ; LOW BYTE
413 01E4 E8 88 C8 MOV CX,AX ; SAVE CX ABOVE 640K MEMORY SIZE
414
415 ;----- ABOVE 640K SIZE FROM MEMORY SIZE DETERMINE
416 01E6 B8 B0B1 MOV AX,(CMOS_U_M_S_LO+NMI)*4+(CMOS_U_M_S_HI+NMI)
417 01E7 E8 0000 E CALL CMOS_READ ; HIGH BYTE
418 01E9 86 E0 XCHG AH,AL ; SAVE HIGH BYTE
419 01EE E8 0000 E CALL CMOS_READ ; LOW BYTE
420 ;----- WHICH IS GREATER - AX = MEMORY SIZE DETERMINE
421 ;----- CX = CONFIGURATION (ABOVE 640) BX = SIZE (BELOW 640)
422
423 01F1 3B C8 CMP CX,AX ; IS CONFIGURATION EQUAL TO DETERMINED?
424 01F3 74 0F JZ SET_MEM1 ; GO IF EQUAL
425
426 ;----- SET MEMORY SIZE DETERMINE NOT EQUAL TO CONFIGURATION
427
428 01F5 50 PUSH AX ; SAVE AX
429 01F6 86 8E8E MOV AX,X*(CMOS_DIAG+NMI) ; ADDRESS THE STATUS BYTE
430 01F9 E8 0000 E CALL CMOS_READ ; GET THE STATUS
431 01FC 0C 10 OR AL,W_MEMORY_SIZE ; SET CMOS FLAG
432 01FE 86 C4 XCHG AH,AL ; SAVE AL
433 0200 E8 0000 E CALL CMOS_WRITE ; UPDATE STATUS BYTE
434 0203 58 POP AX ; RESTORE AX
435
436 0204 SET_MEM1: CMP CX,AX ; IS CONFIG GREATER THAN DETERMINED?
437 0204 3B C8 JA SET_MEM ; GO IF YES
438 0206 77 02 MOV CX,AX ; USE MEMORY SIZE DETERMINE IF NOT
439 0208 88 C8
440 020A
441 020B 03 D9 SET_MEM:
442 020C ADD BX,CX ; SET TOTAL MEMORY SIZE
443 020C 81 FA 0201 E20C: CMP DX,513 ; CHECK IF BASE MEMORY LESS 512K
444 0210 72 D0 JB NO_640 ; GO IF YES
445
446 0212 B8 B3B3 MOV AX,X*(CMOS_INFO128+NMI) ; SET 640K BASE MEMORY BIT
447 0215 E8 0000 E CALL CMOS_READ ; GET THE CURRENT STATUS
448 0218 C8 00 OR AL,4040K ; 4040K = 640K IF NOT ALREADY ON
449 021A 86 C4 XCHG AH,AL ; SAVE THE CURRENT DIAGNOSTIC STATUS
450 021C E8 0000 E CALL CMOS_WRITE ; RESTORE THE STATUS
451 021F NO_640: MOV WORD PTR KB_FLAG,BX ; SAVE TOTAL SIZE FOR LATER TESTING
452 0223 C1 EB 06 SHR BX,6 ; DIVIDE BY 64
453 0226 4B DEC BX ; 1ST 64K ALREADY DONE
454 0227 C1 EA 06 SHR DX,6 ; DIVIDE BY 64 FOR BASE
455
456

```

```

457           ;----- SAVE COUNTS IN STACK FOR BOTH MEMORY AND ADDRESSING TESTS
458
459 022A 52           PUSH  DX          ; SAVE BASE MEMORY SIZE COUNT
460 022B 6A 40           PUSH  BYTE PTR 64   ; SAVE STARTING AMOUNT OF MEMORY OK
461 022D 53           PUSH  BX          ; SAVE COUNT OF 64K BLOCKS TO BE TESTED
462
463 022E 52           PUSH  DX          ; SAVE BASE MEMORY SIZE COUNT
464 022F 6A 40           PUSH  BYTE PTR 64   ; SAVE STARTING AMOUNT OF MEMORY OK
465 0231 53           PUSH  BX          ; SAVE COUNT OF 64K BLOCKS TO BE TESTED
466
467           ;----- MODIFY DESCRIPTOR TABLES
468
469 0232 6A 08           PUSH  BYTE PTR GDT_PTR ; MODIFY THE DESCRIPTOR TABLE
470 0234 07           POP   ES
471
472           ;----- SET TEMPORARY ES DESCRIPTOR 64K SEGMENT LIMIT STARTING AT 000000
473
474 0235 26; CT 06 0048 FFFF           MOV   ES:ES TEMP SEG LIMIT,MAX SEG LEN
475 023C 24; CT 06 0044 0000           MOV   ES:DS TEMP BASE LO WORD,0
476 0243 26; C6 06 004C 00           MOV   BYTE PTR ES:(ES TEMP BASE HI BYTE),0
477 0249 26; C6 06 004D 93           MOV   BYTE PTR ES:(ES TEMP DATA ACC RIGHTS),CPL0 DATA ACCESS
478
479           ;----- SET TEMPORARY DS DESCRIPTOR 64K SEGMENT LIMIT AT FIRST 65K BLOCK
480
481 024F 24; CT 06 0060 FFFF           MOV   ES:DS TEMP SEG LIMIT,MAX SEG LEN
482 0256 24; CT 06 0062 0000           MOV   ES:DS TEMP BASE LO WORD,0
483 025D 26; C6 06 0064 00           MOV   BYTE PTR ES:(DS TEMP BASE HI BYTE),0
484 0263 26; C6 06 0065 93           MOV   BYTE PTR ES:(DS TEMP DATA ACC RIGHTS),CPL0 DATA ACCESS
485
486           ;----- TEMPORARY SEGMENT SAVE IN DMA PAGE REGISTER FOR SECOND 65K BLOCK
487
488 0269 2A C0           SUB   AL,AL          ; INITIALIZE VALUES TO 010000
489 026B E6 85           OUT   DMA PAGE+4,AL   ; HIGH BYTE OF LOW WORD OF SEGMENT
490 026D E6 86           OUT   DMA PAGE+5,AL   ; LOW BYTE OF LOW WORD OF SEGMENT
491 026F FE C0           INC   AL             ; SET HIGH BYTE OF SEGMENT WORD
492 0271 E6 84           OUT   DMA PAGE+3,AL   ; HIGH BYTE OF SEGMENT
493
494           ;----- MEMORY TEST LOOP - POINT TO NEXT BLOCK OF 32K WORDS (64K)
495
496 0273 6A 08           E21:  PUSH  BYTE PTR GDT_PTR ; MEMORY TEST LOOP
497 0275 1F           POP   DS             ; POINT TO START OF DESCRIPTOR TABLE
498 0276 FE 06 0064           INC   BYTE PTR DS:(DS TEMP BASE HI BYTE) ; POINT TO NEXT BLOCK
499 027A FE 06 004C           INC   BYTE PTR DS:(ES TEMP BASE HI BYTE)
500
501           ;----- CHECK FOR END OF 256K PLANAR MEMORY
502
503 027E 80 3E 0064 04           CMP   BYTE PTR DS:(DS TEMP BASE HI BYTE),04H
504 0283 72 04           JB    E21_0          ; GO IF STILL FIRST 256K OF BASE MEMORY
505
506 0285 80 C0           MOV   AL,PARITY_CHECK+10_CHECK; CHECK FOR ANY TYPE OF PARITY ERROR
507 0287 E6 87           OUT   DMA PAGE+6,AL   ; AFTER FIRST 256K
508
509           ;----- CHECK END OF FIRST 640K OR ABOVE (END OF MAXIMUM BASE MEMORY)
510
511 0289 6A 08           E21_0:  CMP   BYTE PTR DS:(DS TEMP BASE HI BYTE),0AH
512 0289 80 3E 0064 0A           JA    NEXT           ; CONTINUE IF ABOVE 1 MEG
513 028E 77 16
514
515           ;----- CHECK FOR END OF BASE MEMORY TO BE TESTED
516
517 0290 59           POP   CX             ; GET COUNT
518 0291 5B           POP   BX             ; GET COUNT TESTED
519 0292 58           POP   AX             ; RECOVER COUNT OF BASE MEMORY BLOCKS
520 0293 50           PUSH  AX             ; SAVE BASE COUNT
521 0294 53           PUSH  BX             ; SAVE TESTED COUNT
522 0295 51           PUSH  CX             ; SAVE TOTAL COUNT
523 0296 38 06 0064           CMP   BYTE PTR DS:(DS TEMP BASE HI BYTE),AL ; MAX BASE COUNT
524 029A 72 0A           JB    NEXT           ; CONTINUE IF NOT DONE WITH BASE MEMORY
525
526
527           ;----- DO ADDITIONAL STORAGE ABOVE 1 MEG
528
529 029C C6 06 0064 10           MOV   BYTE PTR DS:(DS TEMP BASE HI BYTE),10H
530 02A1 C6 06 004C 10           MOV   BYTE PTR DS:(ES TEMP BASE HI BYTE),10H
531
532           ;----- SAVE BASE_HI_BYTE IN DMA PAGE REGISTERS 3
533
534 02A6 A0 0064           NEXT:  MOV   AL,BYTE PTR DS:(DS TEMP BASE HI BYTE)
535 02A9 E6 84           OUT   DMA PAGE+3,AL   ; SAVE THE HIGH BYTE OF SEGMENT
536
537           ;----- CHECK FOR TOP OF MEMORY (FE0000) 16 MEG
538
539 02AB 80 3E 004C FE           CMP   BYTE PTR DS:(ES TEMP BASE HI BYTE),0FEH ; TOP OF MEMORY?
540 02B0 74 29           JE    KB_LOOP3        ; EXIT NEXT TEST IF DONE
541
542
543           ;----- SET ES AND DS REGISTERS TO MEMORY BLOCK
544
545 02B2 6A 60           PUSH  BYTE PTR DS TEMP
546 02B4 1F           POP   DS
547 02B5 6A 48           PUSH  BYTE PTR ES TEMP
548 02B7 07           POP   ES
549
550 02B8 B9 8000           MOV   AL,31H          ; <><><><><><><><><><>
551 02B8 E6 80           OUT   MFG_PORT,AL   ; <><> CHECKPOINT 31 <>
552
553 02BC B9 8000           MOV   CX,8000H ; SET COUNT FOR 32K WORDS
554 02BF E8 0000 E          CALL  STGTST_CNT
555 02C2 74 03           JZ    N1
556 02C4 E9 0367 R          JMP   E21A          ; SKIP IF OK
557 02C7
558 02C7 59           N1:   POP   CX             ; POP CX TO GET AX
559 02C8 58           POP   AX             ; RECOVER TESTED MEMORY
560
561           ;----- WRITE THE CURRENT SIZE FOR (ADDRESS LINE 23-17 TEST) USED LATER
562
563 02C9 2B FF           SUB   DI,DI          ; POINT TO BEGINNING OR A BLOCK
564 02CB AB           STOSW
565
566 02CC 05 0040           ADD   AX,64          ; WRITE THE CURRENT SIZE
567 02CF 50           PUSH  AX          ; AT START OF ADDRESS
568 02D0 51           PUSH  CX          ; ADVANCE COUNT TO NEXT BLOCK
569
570 02D1 E8 099F R          CALL  PRT_OK        ; SAVE TESTED MEMORY
571
572           ;----- DISPLAY "0XXXX OK" MESSAGE

```

```

571 02D4 59          POP    CX          ; RECOVER 64K BLOCK COUNT
572 02D5 49          DEC    CX          ; DECREMENT BLOCK COUNT FOR LOOP
573 02D6 E3 03        JCXZ   KB_LOOP3   ; CONTINUE TO NEXT TEST IF DONE
574
575 02D8 51          PUSH   CX          ; SAVE LOOP COUNT
576 02D9 EB 98        JMP    E21        ; LOOP TILL ALL MEMORY CHECKED
577
578 02DB             KB_LOOP3:      ; END MAIN TEST LOOP
579 02DB 58          POP    AX          ; CLEAR MAXIMUM BLOCK COUNT
580 02DC 58          POP    AX          ; CLEAR BASE SIZE COUNT FROM STACK
581
582 ;----- ADDRESS LINE 16-23 TEST
583
584 02DD B9 40BB      MOV    CX,16571   ; LET FIRST PASS BE SEEN
585 02E0 E8 0000 E     CALL   WAITF    ; COUNT FOR 250 MS FIXED TIME DELAY
586
587 ;----- INITIALIZE DS DESCRIPTOR
588
589 02E3 6A 08        PUSH   BYTE PTR GDT_PTR
590 02E5 07          POP    ES          ; CLEAR TEST VALUES ARE IN STACK
591 02E6 26: C6 06 0064 00  MOV    BYTE PTR ES:(DS_TEMP.BASE_HI_BYTE),0
592 02EC 26: C7 06 0062 00  MOV    ES:DS_TEMP.BASE_LO_WORD,0
593
594 ;----- TEMPORARY SEGMENT SAVE IN DMA PAGE REGISTER
595
596 02F3 2A C0        SUB    AL,AL
597 02F5 E6 85        OUT   DMA_PAGE+4,AL ; HIGH BYTE OF LOW WORD OF SEGMENT
598 02F7 E6 86        OUT   DMA_PAGE+5,AL ; LOW BYTE OF LOW WORD OF SEGMENT
599 02F9 B0 01        MOV    AL,01H
600 02FB E6 84        OUT   DMA_PAGE+3,AL ; SET HIGH BYTE OF SEGMENT WORD
601
602 ;----- POINT TO NEXT BLOCK OF 64K
603
604 02FD             E21_A:      ; <><><><><><><><><><>
605 02FD B0 33        MOV    AL,33H
606 02FF E6 80        OUT   MFG_PORT,AL ; <><> CHECKPOINT 33 <><>
607 0301 26: B0 06 0064 01  ADD   BYTE PTR ES:(DS_TEMP.BASE_HI_BYTE),01
608
609 ;----- CHECK FOR END OF BASE MEMORY TO BE TESTED
610
611 0307 26: B0 3E 0064 0A  CMP   BYTE PTR ES:(DS_TEMP.BASE_HI_BYTE),0AH
612 030D 77 13        JA    NEXT_A    ; CONTINUE IF ABOVE 1 MEG
613
614 030F 59          POP    CX          ; GET COUNT
615 0310 58          POP    BX          ; GET COUNT TESTED
616 0311 58          POP    AX          ; RECOVER COUNT OF BASE MEMORY BLOCKS
617 0312 50          PUSH   AX          ; SAVE BASE COUNT
618 0313 53          PUSH   BX          ; SAVE TESTED COUNT
619 0314 51          PUSH   CX          ; SAVE TOTAL COUNT
620 0315 26: 38 06 0064  CMP   BYTE PTR ES:(DS_TEMP.BASE_HI_BYTE),AL ; MAX BASE COUNT
621 031A 72 06        JB    NEXT_A    ; CONTINUE IF NOT DONE WITH BASE MEMORY
622
623 ;----- DO ADDITIONAL STORAGE ABOVE 1 MEG
624
625 031C             NEXT_A2:    ; <><><><><><><><><><>
626 031C 26: C6 06 0064 10  MOV   BYTE PTR ES:(DS_TEMP.BASE_HI_BYTE),10H
627 0322             NEXT_A:    ; <><><><><><><><><><>
628 0322 26: A0 0064 0     MOV   AL,BYTE PTR ES:(DS_TEMP.BASE_HI_BYTE)
629
630 ;----- DMA PAGE REGISTERS 3
631
632 0326 E6 84        OUT   DMA_PAGE+3,AL ; SAVE THE HIGH BYTE OF SEGMENT
633
634 ;----- CHECK FOR TOP OF MEMORY (FE0000) 16 MEG
635
636 0328 3C FE        CMP   AL,0FEH
637 032A 74 34        JZ    KB_LOOP_3 ; TOP OF MEMORY?
638
639 ;----- SET DS REGISTER
640
641 032C 6A 60        PUSH   BYTE PTR DS_TEMP
642 032E 1F          POP    DS          ; POINT TO START OF BLOCK
643 0330 26: FF        SUB   DI,DI
644 0331 8B 15        MOV   DS:[DI]
645 0333 8B F7        MOV   SI,DI
646 0335 2B C0        SUB   AX,AX
647 0337 89 05        MOV   [DI],AX ; SET SI FOR POSSIBLE ERROR
648
649 ;----- ALLOW DISPLAY TIME TO DISPLAY MESSAGE AND REFRESH TO RUN
650
651 0339 B9 1A69      MOV   CX,6761   ; COUNT FOR 102 MS FIXED TIME DELAY
652 033C E8 0000 E     CALL   WAITF    ; ALLOW FIVE DISPLAY REFRESH CYCLES
653 033F 59          POP    CX          ; GET THE LOOP COUNT
654 0340 58          POP    AX          ; RECOVER TESTED MEMORY
655 0341 50          PUSH   AX          ; SAVE TESTED MEMORY
656 0342 51          PUSH   CX          ; SAVE LOOP COUNT
657 0343 3B C2        CMP   AX,DX
658 0345 8B C2        MOV   AX,DX
659 0347 75 1E        JNZ   E21A      ; DOES THE BLOCK ID MATCH
660
661 ;----- CHECK FOR CHECK PARITY
662
663 0349 E4 61        IN    AL,PORT_B
664 034B 24 C0        AND   AL,PARITY_ERR
665 034D 75 18        JNZ   E21A      ; STRIP UNWANTED BITS
666
667 034F 59          POP    CX          ; EXIT IF PARITY ERROR
668 0350 58          POP    AX          ; POP CX TO GET AX
669 0351 05 0040      ADD   AX,64
670 0354 50          PUSH   AX          ; RECOVER TESTED MEMORY
671 0355 51          PUSH   CX          ; 64K INCREMENTS
672 0356 E8 099F R     CALL   PRT_OK
673 0357 59          POP    CX          ; SAVE TESTED MEMORY
674 0358 49          DEC    CX          ; SAVE LOOP COUNT
675 035B E3 03        JCXZ KB_LOOP_3 ; DISPLAY ON MESSAGE
676
677 035D 51          PUSH   CX          ; RECOVER 64K BLOCK COUNT
678 035E EB 9D        JMP    E21_A    ; POP TILL ALL MEMORY CHECKED
679
680 ;----- BACK TO REAL MODE - MEMORY TESTS DONE
681
682 KB_LOOP_3:          ; <><><><><><><><><><>
683 0360             MOV   AL,34H
684 0362 E6 80        OUT   MFG_PORT,AL ; <><> CHECKPOINT 34 <><>

```

```
685 0364 E9 0000 E      JMP    PROC_SHUTDOWN      ; BACK TO REAL MODE
686                                         ; NEXT TEST VIA JUMP TABLE (I$HUT2)
687
688
689
690      ;----- PRINT FAILING ADDRESS AND XOR'ED PATTERN IF DATA COMPARE ERROR
691      ;----- USE DMA PAGE REGISTERS AS TEMPORARY SAVE AREA FOR ERROR
692      ;----- SET SHUTDOWN 3
693 0367 E6 82      E21A: OUT   DMA_PAGE+1,AL      ; SAVE FAILING BIT PATTERN (LOW BYTE)
694      MOV   AX,AX
695      OUT   DMA_PAGE+2,AL      ; SAVE HIGH BYTE
696 036D E8 C4      MOV   AX,51
697 036F E6 86      OUT   DMA_PAGE+5,AL      ; GET THE FAILING OFFSET
698 0371 E6 E0      XCHG  AH,AL
699 0373 E6 85      OUT   DMA_PAGE+4,AL
700
701      ;----- CLEAR I/O CHANNEL CHECK OR R/W PARITY CHECK
702
703 0375 2B F6      SUB   $1,$1      ; WRITE TO FAILING BLOCK
704 0377 AB          STOSB
705 0378 E6 61      IN    AL,PORT_B      ; GET PARITY CHECK LATCHES
706 0379 E6 88      OUT   DMA_PAGE+7,AL      ; SAVE FOR ERROR HANDLER
707 037C DC 0C      OR    AL,RAM_PAR_OFF
708 037E E6 61      OUT   PORT_B,AL      ; TOGGLE I/O-PARITY CHECK ENABLE
709 0380 24 F3      AND   AL,RAM_PAR_ON
710 0382 E6 61      OUT   PORT_B,AL
711
712
713
714 0384 58      POP   AX      ; CLEAR_BLOCK_COUNT
715 0385 58      POP   AX      ; GET THE LAST OF GOOD MEMORY
716 0386 5B      POP   BX      ; GET BASE MEMORY COUNTER
717 0387 C1 E3 06      SHL   BX,6      ; CONVERT TO MEMORY SIZE COUNTS
718 038A 2B C3      SUB   AX,BX
719 038C T3 17      JAE   E211      ; COMPARE LAST GOOD MEMORY WITH BASE
720
721      ;----- GET THE LAST OF GOOD MEMORY
722
723 038E 6A 18      POP   DS      ; IF ABOVE OR EQUAL, USE REMAINDER IN
724 0390 1F          PUSH  BYTE PTR RSDA_PTR      ; CMOS_U_M_S_(H/L)
725
726 0391 03 C3      ADD   AX,BX      ; SET THE DATA SEGMENT
727 0393 A3 0013 R     MOV   @MEMORY_SIZE,AX      ; IN PROTECTED MODE
728
729      ;----- ELSE SET BASE MEMORY SIZE
730
731 0396 B8 B3B3      MOV   AX,X*(CMOS_INFO128+NMI) ; ADDRESS OPTIONS INFORMATION BYTE
732 0399 E8 0000 E     CALL  CMOS_READ      ; READ THE MEMORY INFORMATION FLAG
733 039C 24 7F      AND   AL,NOT M640K
734 039E 86 C4      XCHG  AL,AH      ; SET 640K OPTION OFF
735 03A0 E8 0000 E     CALL  CMOS_WRITE     ; MOVE TO WORK REGISTER
736 03A2 33 C0      XOR   AX,AX      ; UPDATE STATUS IF IT WAS ON
737 03A5             E211: MOV   CX,AX      ; CLEAR VALUE FOR EXTENSION MEMORY
738 03A5 B8 C8      MOV   AL,CMOS_U_M_S_HI+NMI
739 03A7 B0 B1      CALL  CMOS_WRITE     ; SAVE THE HIGH BYTE MEMORY SIZE
740 03A9 E8 0000 E     MOV   AH,CL      ; GET THE LOW BYTE
741 03AC 8A E1      MOV   AL,CMOS_U_M_S_LO+NMI
742 03AE B0 B0      CALL  CMOS_WRITE     ; DO THE LOW BYTE
743 03B0 E8 0000 E
744
745
746
747 03B3 B8 038F      MOV   AX,3*H+CMOS_SHUT_DOWN+NMI      ; ADDRESS FOR SHUTDOWN RETURN
748 03B6 E8 0000 E     CALL  CMOS_WRITE     ; SET RETURN 3
749
750
751
752 03B9 E9 0000 E      JMP    PROC_SHUTDOWN
```

```

PAGE
MEMORY ERROR REPORTING (R/W/ MEMORY OR PARITY ERRORS)
DESCRIPTION FOR ERRORS 201 (CMP ERROR OR PARITY)
OR 202 (ADDRESS LINE 0-15 ERROR)
"AABBCC DDEE 201" (OR 202)
AA=HIGH BYTE OF 24 BIT ADDRESS
BB=MIDDLE BYTE OF 24 BIT ADDRESS
CC=LOW BYTE OF 24 BIT ADDRESS
DD=HIGH BYTE OF XOR FAILING BIT PATTERN
EE=LOW BYTE OF XOR FAILING BIT PATTERN

DESCRIPTION FOR ERROR 202 (ADDRESS LINE 00-15)
A WORD OF FFFF IS WRITTEN AT THE FIRST WORD AND LAST WORD
OF EACH 64K BLOCK WITH ZEROS AT ALL OTHER LOCATIONS OF THE
BLOCK. A SCAN OF THE BLOCK IS MADE TO INSURE ADDRESS LINE
0-15 ARE FUNCTIONING.

DESCRIPTION FOR ERROR 203 (ADDRESS LINE 16-23)
AT THE LAST PASS OF THE STORAGE TEST, FOR EACH BLOCK OF
64K, THE CURRENT STORAGE SIZE (ID) IS WRITTEN AT THE FIRST
WORD OF EACH BLOCK. IT IS USED TO FIND ADDRESSING FAILURES.

"AABBCC DDEE 203"
SAME AS ABOVE EXCEPT FOR DDEE

GENERAL DESCRIPTION FOR BLOCK ID (DDEE WILL NOW CONTAINED THE ID)
DD=HIGH BYTE OF BLOCK ID
EE=LOW BYTE OF BLOCK ID

BLOCK ID ADDRESS RANGE
0000 000000 --> 00FFFF
0040 010000 --> 01FFFF
//
0200 090000 --> 09FFFF (512->576K) IF 640K BASE
100000 100000 --> 10FFFF (1024->1088K) IF 512K BASE

EXAMPLE (640K BASE MEMORY + 512K I/O MEMORY = 1152K TOTAL)
NOTE: THE CORRECT BLOCK ID FOR THIS FAILURE IS 0200 HEX.
DUE TO AN ADDRESS FAILURE THE BLOCK ID=128K OVERLAYED
THE CORRECT BLOCK ID.

00640K OK <-- LAST OK MEMORY
10000 0300 202 <-- ERROR DUE TO ADDRESS FAILURE

IF A PARITY LATCH WAS SET THE CORRESPONDING MESSAGE WILL DISPLAY.

"PARITY CHECK 1" (OR 2)

DMA PAGE REGISTERS ARE USED AS TEMPORARY SAVE AREAS FOR SEGMENT
DESCRIPTOR VALUES.

SHUT3: : ENTRY FROM PROCESSOR SHUTDOWN 3
CALL DDS : SET REAL MODE DATA SEGMENT

012 03BF C6 06 0016 R 01
MOV #MF6_ERR_FLAG+1, MEM_FAIL : <=> MEMORY FAILED <=>
03CA 00 00 00 E
MOV AL, CR : CLEAR AND SET MANUFACTURING ERROR FLAG
03C6 E8 0000 E
CALL PRT_HEX : CARRIAGE RETURN
03C9 B0 00 E
MOV AL, LF : LINE FEED
03CB E8 0000 E
CALL PRT_HEX
03CE E4 84
IN AL, DMA_PAGE+3 : GET THE HIGH BYTE OF 24 BIT ADDRESS
03D0 E8 0000 E
CALL XPC_BYT : CONVERT AND PRINT CODE
03D9 E8 85
IN AL, DMA_PAGE+4 : GET THE MIDDLE BYTE OF 24 BIT ADDRESS
03D6 E8 0000 E
CALL XPC_BYT
03D8 E8 0000 E
IN AL, DMA_PAGE+5 : GET THE LOW BYTE OF 24 BIT ADDRESS
03D0 E8 0000 E
CALL XPC_BYT
03D2 B0 20
MOV AL, T : SPACE TO MESSAGE
03D3 E8 0000 E
CALL PRT_HEX
03D4 E8 0000 E
IN AL, DMA_PAGE+2 : GET HIGH BYTE FAILING BIT PATTERN
03D5 E8 0000 E
CALL XPC_BYT : CONVERT AND PRINT CODE
03E1 E4 82
IN AL, DMA_PAGE+1 : GET LOW BYTE FAILING BIT PATTERN
03E9 E8 0000 E
CALL XPC_BYT : CONVERT AND PRINT CODE

030 :----- CHECK FOR ADDRESS ERROR
03E E4 80
IN AL, MFC_PORT : GET THE CHECKPOINT
03E3 0003 33 : IS IT AN ADDRESS FAILURE?
03F0 BE 0000 E
MOV S1,OFFSET E203 : LOAD ADDRESS ERROR 16->23
03F3 T4 0A
JZ ERR2 : GO IF YES

03F5 BE 0000 E
MOV S1,OFFSET E202 : LOAD ADDRESS ERROR 00->15
03F8 3C 32
CMP AL, 32H : GO IF YES
03FA T4 03
JZ ERR2

03FCE BE 0000 E
MOV S1,OFFSET E201 : SETUP ADDRESS OF ERROR MESSAGE
03FF
ERR2: CALL E_MSG : PRINT ERROR MESSAGE
IN AL, DMA_PAGE+7 : GET THE PORT_B VALUE

0404 E8 0000 E
0402 E4 88
0404 :----- DISPLAY "PARITY CHECK ?" ERROR MESSAGES
0404 A0 80
TEST AL, PARITY_CHECK : CHECK FOR PLANAR ERROR
0406 74 0B
JZ NMI_M1 : SKIP IF NOT

0408 50
PUSH AX : SAVE STATUS
0409 E8 098F R
CALL PADING : INSERT BLANKS
040C BE 0000 E
MOV S1,OFFSET D1 : PLANAR ERROR, ADDRESS "PARITY CHECK 1"
040F E8 0000 E
CALL P_MSG : DISPLAY "PARITY CHECK 1" MESSAGE
0412 58
POP AX : AND RECOVER STATUS

0413
NMI_M1: TEST AL, 10_CHECK : I/O PARITY CHECK ?
0415 A8 40
JZ NMI_M2 : SKIP IF CORRECT ERROR DISPLAYED
0415 74 09

0417 E8 098F R
CALL PADING : INSERT BLANKS
041A BE 0000 E
MOV S1,OFFSET D2 : ADDRESS OF "PARITY CHECK 2" MESSAGE
041D E8 0000 E
CALL P_MSG : DISPLAY "PARITY CHECK 2" ERROR
0420
NMI_M2: : CONTINUE TESTING SYSTEM

```



```

979 048B E8 0000 E  LOOP1: CALL 0BF_42 ; CHECK FOR OUTPUT BUFFER FULL
980 048E 75 04 JNZ G10 ; GO IF BUFFER FULL
981 0490 FE CF DEC
982 0491 00 07 JNZ LOOP1
983 0494 B0 AD G10: MOV AL,_DIS_KBD ; DISABLE KEYBOARD
984 0496 E8 0000 E CALL C8042 ; FLUSH
985 0499 E4 60 IN AL,_PORT_A ; GET THE CLOCK AND DATA LINES
986 049B E8 00 E MOV AL,_KYBD_CLK_DATA
987 049D E8 0000 E CALL C8042 ; WAIT FOR OUTPUT BUFFER FULL
988 049E E8 0000 E IN AL,_PORT_A ; GET THE RESULTS
989 04A3 E4 60 TEST AL,_KYBD_CLK ; KEYBOARD CLOCK MUST BE LOW
990 04A5 A8 01 JZ G11
991 04A7 74 0A
992
993 04A9 80 0E 0016 R 08 OR 0MFG_ERR_FLAG+i,KYCLK_FAIL ; <><><><><><><><><><><>
994
995 04AE BE 0000 E MOV SI,OFFSET E304 ; DISPLAY 304K ERROR
996 04B1 EB 60 JMP SHORT F6D ; REPORT ERROR
997 04B3 E8 0000 E G11: CALL KBD_RESET ; ISSUE RESET TO KEYBOARD
998 04B6 E3 29 JCXZ F6 ; PRINT ERROR MESSAGE IF NO INTERRUPT
999 04B8 B0 37 MOV AL,37H ; <><><><><><><><><>
1000 04B8 E6 80 OUT MFG_PORT,AL ; <><> CHECKPOINT 37 <><>
1001 04C4 8C 80 CMP BL,_KB_OK ; SCAN CODE AS EXPECTED?
1002 04B7 75 20 JNE F6 ; NO - DISPLAY ERROR MESSAGE
1003
1004 ;----- CHECK FOR STUCK KEYS
1005
1006 04C1 B0 38 MOV AL,38H ; <><><><><><><><><>
1007 04C3 E6 80 OUT MFG_PORT,AL ; <><> CHECKPOINT 38 <><>
1008
1009 04C5 B0 AE MOV AL,_ENA_KBD ; ASSURE KEYBOARD ENABLED
1010 04C7 E8 0000 E CALL C8042 ; ISSUE THE COMMAND
1011 04CA B9 19E4 MOV CX,6628 ; COUNT FOR 100 MILLISECONDS
1012 04C9 E8 0000 E CALL WAIT ; DELAY FOR A WHILE
1013 04D0 E4 44 IN AL,_STATUS_PORT ; CHECK FOR STUCK KEYS
1014 04D2 88 01 TEST AL,_OUT_BUF_FULL ; OUR BUFFER FULL?
1015 04D4 74 40 JE F7 ; YES - CONTINUE TESTING
1016
1017 04D6 B0 39 MOV AL,39H ; <><><><><><><><><>
1018 04D8 E6 80 OUT MFG_PORT,AL ; <><> CHECKPOINT 39 <><>
1019
1020 04DA E4 60 IN AL,_PORT_A ; GET THE SCAN CODE
1021 04DC E8 0000 E CALL XPC_BYT ; CONVERT AND PRINT
1022 04DF EB 2A JMP SHORT F6C ; CONTINUE
1023
1024 ;----- KEYBOARD ERROR TRY TO DETERMINE IF 8042 INTERFACE IS WORKING
1025
1026 04E1 FA F6: CLI ; COMMAND TO 8042
1027 04E2 B0 AB MOV AL,_INTR_FACE_CK ; STATUS_PORT,AL
1028 04E4 E6 64 OUT SUB CX,CX ; WAIT FOR OUTPUT BUFFER FULL
1029 04E6 2B 00 MOV BH,05 ; 8042 FINISHED TEST?
1030 04E8 B7 05 IN AL,_STATUS_PORT ; GO CHECK RESULTS
1031 04E9 E4 44 TEST AL,_OUT_BUF_FULL ; TRY AGAIN
1032 04EC A8 01 JNZ F6B ; INDICATE PLANAR FAILURE
1033 04E4 E1 FA MOV SHORT F6D ; REMOVE KEYBOARD TRY AGAIN
1034 04F0 75 09 IN AL,_PORT_A ; GET THE RESULTS OF INTERFACE TEST
1035 04F2 FE CF CMP AL,0 ; IS THE INTERFACE OK?
1036 04F4 7E F4 JZ F6C ; OR 0MFG_ERR_FLAG+i,KY_SYS_FAIL ; <><><><><><><><><>
1037 04F5 E8 0000 E MOV SI,OFFSET E303 ; PLANAR FAILURE
1038 04F9 EB 18 JMP SHORT F6D ; GO IF YES
1039 04F8 E4 60 F6B: IN AL,_PORT_A ; GET MESSAGE ADDRESS
1040 04FD 3C 00 CMP F6C ; IS THE INTERFACE OK?
1041 04F4 74 0A JZ F6C ; OR 0MFG_ERR_FLAG+i,KYBD_FAIL; <><><><><><><><><>
1042 0501 80 0E 0016 R 10
1043
1044 0506 BE 0000 E MOV SI,OFFSET E303 ; <><> KEYBOARD/SYSTEM <><>
1045 0509 EB 08 JMP SHORT F6D ; <><> KEYBOARD/SYSTEM <><>
1046 050B BE 0000 E F6C: MOV SI,OFFSET E301 ; <><> KEYBOARD FAILED <><>
1047
1048 050E 80 0E 0016 R 20 OR 0MFG_ERR_FLAG+i,KYBD_FAIL; <><><><><><><><><>
1049
1050
1051 0513 E8 0000 E F6D: CALL E_MSG ; PRINT MESSAGE ON SCREEN
1052
1053 ;----- INITIALIZE 8042 TO HONOR KEY LOCK
1054
1055 0516 B0 3A F7: MOV AL,3AH ; <><><><><><><><><>
1056 0518 E6 80 OUT MFG_PORT,AL ; <><> CHECKPOINT 3A <><>
1057
1058 051A B0 FF MOV AL,0FFH ; DISABLE INTERRUPTS
1059 051C E6 21 OUT INTA01,AL
1060 051E F0 CLI ; WRITE 8042 MEMORY COMMAND
1061 051F B0 60 MOV AL,_WRITE_8042_LOC ; ISSUE THE COMMAND
1062 0521 E8 0000 E CALL C8042 ; SET SYSTEM FLAG - OUTBUF INTERRUPT -
1063 0524 B0 45 MOV AL,45H ; PC 1 COMPATIBILITY
1064 0526 E6 60 OUT PORT_A,AL ; RESET INHIBIT OVER RIDE
1065
1066
1067 ;----- DEGATE ADDRESS LINE 20
1068 0528 B4 DD MOV AH,_DISABLE_BIT20 ; SET COMMAND IN AH
1069 052A E8 0000 E CALL GATE_A20 ; ISSUE THE COMMAND
1070
1071 ;----- SETUP HARDWARE INTERRUPT VECTOR TABLE LEVEL 0-7
1072
1073 052D 2B C0 SUB AX,AX ; GET VECTOR COUNT
1074 053F BE C0 MOV DS,AX ; SETUP DS SEGMENT REGISTER
1075 0531 B9 0008 PUSH CS
1076 0534 0E POP DS ; GET VECTOR COUNT
1077 0535 1F MOV SI,OFFSET VECTOR_TABLE ; SETUP DS SEGMENT REGISTER
1078 0536 BE 0000 E MOV DI,OFFSET *INT_PTR
1079 0537 80 0020 R MOV DS,SI ; SKIP OVER SEGMENT
1080 053C A5 F7A: MOV SWI INC DI ; GET VECTOR COUNT
1081 053D 47 INC INC DI ; SETUP DS SEGMENT REGISTER
1082 053E 47 LOOP F7A ; GET VECTOR COUNT
1083 053F E2 FB POP DS ; SETUP DS SEGMENT REGISTER
1084
1085
1086
1087 ;----- SETUP HARDWARE INTERRUPT VECTORS LEVEL 8-15 (VECTORS START AT INT 70H)
1088 0541 2B C0 ASSUME ES:AB50
1089 0543 BE C0 SUB AX,AX
1090 0545 B9 0008 MOV ES,AX
1091 0548 0E PUSH CS ; GET VECTOR COUNT
1092 0549 1F POP DS ; SETUP DS SEGMENT REGISTER

```

```

1093 054A BE 0000 E      MOV    S1,OFFSET SLAVE_VECTOR_TABLE
1094 054D BF 01C0 R      MOV    D1,OFFSET *SLAVE_INT_PTR
1095 0550 A5              FTA1: MOVSW
1096 0551 47              INC    DI
1097 0552 47              INC    DI
1098 0553 E2 FB          LOOP   FTA1
1099
1100          ;----- SET UP OTHER INTERRUPTS AS NECESSARY
1101
1102          ASSUME DS:AB50
1103 0555 2B C0          SUB    AX,AX      ; DS=0
1104 0557 BE D8          MOV    AX,AX
1105 0559 CT 06 0008 R 0000 E MOV    WORD PTR *NMI_PTR,OFFSET NMI_INT ; NMI INTERRUPT
1106 055F CT 06 0014 R 0000 E MOV    WORD PTR *INT5_PTR,OFFSET PRNT_SCREEN ; PRINT SCREEN
1107 0565 CT 06 0062 R F600 MOV    WORD PTR *BASIC_PTR+2,OF600H ; SEGMENT FOR CASSETTE BASIC
1108
1109          ;----- ZERO RESERVED VECTORS
1110
1111 0568 BF 0180          MOV    DI,60H*4      ; FILL INTERRUPT 60 THRU 67 WITH ZERO
1112 056E B9 0010          MOV    CX,16      ; CLEAR 16 WORDS
1113 0571 CT 05 0000          FTA2: MOVSW WORD PTR DS:[DI],0
1114 0575 B3 CT 02          ADD    DI,2      ; POINT TO NEXT LOCATION
1115 0578 E2 F7          LOOP   FTA2
1116
1117          ;----- SETUP TIMER 0 TO BLINK LED IF MANUFACTURING TEST MODE
1118
1119          ASSUME DS:DATA
1120 057A E8 0000 E          CALL   DDS      ; ESTABLISH DATA SEGMENT
1121
1122 057D F6 06 0012 R 20          TEST   #MFG_TST,MFG_LOOP ; MFG. TEST MODE?
1123 0582 75 0B              JNZ   F9
1124 0584 26; CT 06 0020 R 0000 E          MOV    WORD PTR ES:INT_PTR,OFFSET BLINK_INT ; SETUP TIMER TO BLINK LED
1125 0588 BA FE              MOV    AL,0FEH      ; ENABLE TIMER INTERRUPT
1126 058D B0 04              OUT   INTA01,AL
1127 058F FB              F9:   STI
1128
1129          ;----- ISSUE A RESET TO THE HARD FILE IF SOFT RESET
1130
1131 0590 B1 3E 0072 R 1234          CMP    #RESET_FLAG,1234H ; SOFT RESET?
1132 0592 75 0E              JNZ   F9A      ; CONTINUE IF NOT
1133 0594 B9 00FF
1134 0598 00 03FF
1135 059E B0 04
1136 05A0 EE
1137 05A1 E2 FE          F9_A: LOOP   F9A      ; HOLD RESET
1138 05A3 E2 C0              SUB   AL,AL
1139 05A5 EE              OUT   DX,AL      ; REMOVE RESET
1140
1141
1142          ;----- TEST.23
1143          ;----- DISKETTE ATTACHMENT TEST
1144          ;----- DESCRIPTION
1145          ;----- CHECK IF IPL DISKETTE DRIVE IS ATTACHED TO SYSTEM. IF
1146          ;----- ATTACHED, SET DISKETTE STATUS IN NEC FDC AFTER A RESET. ISSUE
1147          ;----- A RECALIBRATE AND SEEK COMMAND TO FDC AND CHECK STATUS.
1148          ;----- COMPLETE SYSTEM INITIALIZATION THEN PASS CONTROL TO THE
1149          ;----- BOOT LOADER PROGRAM.
1150
1151
1152 05A6 B0 3C          F9A:  MOV    AL,3CH      ; <><><><><><><><><><>
1153 05A8 E6 80          OUT   MFG_PORT,AL ; <><> CHECKPOINT 3C <><>
1154
1155 05A9 BA 02          MOV    AL,02H      ; SET DATA RATE TO 250 K BITS PER SECOND
1156 05AC BA 03F7
1157 05AC EE
1158 05B0 00 0010 R 01          TEST   BYTE PTR *EQUIP_FLAG,1H ; DISKETTE PRESENT?
1159 05B5 74 55
1160 05B7 F6 06 0012 R 20          JZ    F10      ; MFG JUMPER INSTALLED?
1161 05BC T4 4E              JZ    F15      ; GO IF YES
1162 05BE
1163 05BE E4 21          F10:  IN    AL,INTA01 ; DISKETTE TEST
1164 05C0 00 00
1165 05C2 24 BF              JPP   F12      ; I/O DELAY
1166 05C4 E6 21              AND   AL,0BFH      ; ENABLE DISKETTE INTERRUPTS
1167 05C5 B4 00              OUT   INTA01,AL
1168 05C8 8A D4              MOV    AH,0
1169 05CA CD 13              MOV    DL,AH      ; RESET NEC FDC
1170 05CC F6 C4 FF              INT   13H      ; SET FOR DRIVE 0
1171 05CF T5 25              TEST   AH,0FFH      ; VERIFY STATUS AFTER RESET
1172
1173          ;----- TURN DRIVE 0 MOTOR ON
1174
1175 05D1 BA 03F2          MOV    DX,03F2H      ; GET ADDRESS OF FDC CARD
1176 05D4 B0 1C              MOV    AL,1CH      ; TURN MOTOR ON, ENABLE DMA, INTERRUPTS
1177 05D6 EE
1178 05D7 2B C9              OUT   AL,1H      ; WRITE FDC CONTROL REGISTER
1179 05D9 E8 0000 E          SUB   CX,CX      ; WAIT COUNT FOR 0.988 SECONDS
1180              CALL   WAITF
1181 05DC 33 FF              XOR   DI,DI      ; WAIT 1 SECOND FOR MOTOR
1182 05E0 C6 01
1183 05E0 C6 06 003E R 00          MOV    ORTC_SEEK_STATUS,0 ; SELECT DRIVE 0
1184 05E5 80 0E 00A0 R 01          OR    #RTC_WAIT_FLAG,01 ; SELECT TRACK
1185 05E5 E8 0000 E          CALL   SEEK
1186 05E5 T2 07              JC    F13      ; NO REAL TIME CLOCK, USE WAIT LOOP
1187 05E5 B5 22              MOV    CH,34      ; RECALIBRATE DISKETTE
1188 05F4 E8 0000 E          CALL   SEEK
1189 05F4 T3 0B              JNC   F14      ; GO TO ERR SUBROUTINE IF ERR
1190 05F6
1191 05F6 80 0E 0016 R 40          F13:  XOR   DI,DI      ; SELECT TRACK 34
1192              OR    #MFG_ERR_FLAG+1,DSK_FAIL ; SEEK TO TRACK 34
1193 05F6 BE 0000 E          MOV    S1,OFFSET E601 ; OK, TURN MOTOR OFF
1194 05F6 E8 0000 E          CALL   E_MSG      ; DSK_ERR!
1195
1196          ;----- TURN DRIVE 0 MOTOR OFF
1197
1198 0601
1199 0601 B0 26 00A0 R FE          F14:  AND   #RTC_WAIT_FLAG,0FEH ; <><><><><><><><><><><>
1200 0606 B0 0C              MOV    AL,0CH      ; ALLOW FOR RTC WAIT
1201 0608 BA 03F2
1202 0608 EE
1203
1204          ;----- SETUP KEYBOARD PARAMETERS
1205
1206 060C C6 06 006B R 00          F15:  MOV    #INTR_FLAG,00H ; SET STRAY INTERRUPT FLAG = 00

```

```

1207 0611 BE 001E R           MOV    $1,OFFSET_WB_BUFFER ; SETUP KEYBOARD PARAMETERS
1208 0614 B9 30 001A R           MOV    @OFFSET_HEAD,S1
1209 0618 B9 36 001C R           MOV    @BUFFER_TAIL,S1
1210 061C B9 36 0080 R           MOV    @BUFFER_START,S1
1211 0620 83 C6 20             ADD    S1,32          ; DEFAULT BUFFER OF 32 BYTES
1212 0623 89 36 0082 R           MOV    @BUFFER_END,S1
1213
1214
1215 ;----- SET PRINTER TIMEOUT DEFAULT
1216 0627 BF 0078 R           MOV    D1,OFFSET @PRINT_TIM_OUT ; SET DEFAULT PRINTER TIMEOUT
1217 062A 1E
1218 062B 07             PUSH   DS
1219 062C B8 1414             POP    ES
1220 062F AB             MOV    AX,1414H ; DEFAULT=20
1221 0630 AB             STOSW
1222 0631 AB             STOSW
1223 ;----- SET RS232 DEFAULT
1224 0631 B8 0101             MOV    AX,0101H ; RS232 DEFAULT=01
1225 0634 AB             STOSW
1226 0635 AB             STOSW
1227
1228 ;----- ENABLE TIMER INTERRUPTS
1229 0636 E4 21             IN     AL,INTA01
1230 0637 E4 24 FE             AND    AL,0FEH ; ENABLE TIMER INTERRUPTS
1231 063A EB 00             JMP    $+2          ; I/O DELAY
1232 063C EB 21             OUT   INTA01,AL
1233
1234 ;----- CHECK CMOS BATTERY AND CHECKSUM
1235 063E F6 06 0012 R 20             TEST   @MFG_TST,MFG_LOOP ; MFG JUMPER?
1236 0643 T9 03             JNZ    BL_OR ; GO IF NOT
1237 0645 E9 072E R             JMP    F15C ; BYPASS IF YES
1238
1239 B1_OK: MOV    AL,CMOS_DIAG+NMI ; ADDRESS DIAGNOSTIC STATUS BYTE
1240 0648 CALL   CMOS_READ ; READ IT FROM CMOS
1241
1242 0648 B0 8E             MOV    $1,OFFSET E161 ; LOAD BAD BATTERY MESSAGE 161
1243 064A E8 0000 E            TEST   AL,BAD_BAT ; BATTERY BAD?
1244 064B 80 8E             JNZ    BI_ER ; DISPLAY ERROR IF BAD
1245 064D BE 0000 E            MOV    $1,OFFSET E162 ; LOAD CHECKSUM BAD MESSAGE 162
1246 0657 AB 60             TEST   AL,BAD_CKSUM+BAD_CONFIG ; CHECK FOR CHECKSUM OR NO DISKETTE
1247 0659 74 09             JZ     C_0K ; SKIP AND CONTINUE TESTING CMOS CLOCK
1248
1249 0654 BE 0000 E            CALL   E_MSG ; ELSE DISPLAY ERROR MESSAGE
1250 0657 AB 60             OR    $F0,0000H ; FLAG "SET SYSTEM OPTIONS" DISPLAYED
1251 0659 74 09             JMP    SHORT H_0K1 ; SKIP CLOCK TESTING IF ERROR
1252 065B
1253 065B E8 0000 E            CALL   E_MSG ; TEST CLOCK UPDATING
1254 065E 81 CD 8000             OR    $F0,0000H
1255 0662 EB 45             JMP    SHORT H_0K1
1256
1257
1258 ;----- TEST CLOCK UPDATING
1259 0664 B3 04             C_0K: MOV    BL,04H ; OUTER LOOP COUNT
1260 0666 2B C9             D_0K: SUB    CX,CX ; INNER LOOP COUNT
1261 0668 B0 8A             E_0K: MOV    AL,CMOS_REG_A+NMI ; GET THE CLOCK UPDATE BYTE
1262 066E E9 0000 E            CALL   CMOS_READ
1263 0670 80 80             TEST   AL,80H ; CHECK FOR UPDATE IN PROGRESS
1264 067F 15 1B             JNZ    G_0K ; GO IF YES
1265 0671 E2 F5             LOOP   E_0K ; TRY AGAIN
1266 0673 FE CB             DEC    BL ; DEC OUTER LOOP
1267 0675 75 EF             JNZ    D_0K ; TRY AGAIN
1268 0677 BE 0000 E            F_0K: MOV    ST,OFFSET E163 ; PRINT MESSAGE
1269 067A E8 0000 E            CALL   E_MSG
1270
1271 ;----- SET CMOS DIAGNOSTIC STATUS TO 04 (CLOCK ERROR)
1272 067D BB 008E             MOV    AX,X*CMOS_DIAG+NMI ; SET CLOCK ERROR
1273 0680 E9 0000 E            CALL   CMOS_READ ; GET THE CURRENT STATUS
1274 0683 C0 04             D_0K: OR    AL,CMOS_CLK_FAIL ; GET STATUS
1275 0685 B6 C4             E_0K: TEST   AL,80H ; GET STATUS ADDRESS AND SAVE NEW STATUS
1276 0686 E0 F7             XCHG   AL,AH ; MOVE NEW DIAGNOSTIC STATUS TO CMOS
1277 0687 EB 0000 E            CALL   CMOS_WRITE ; CONTINUE
1278 068A EB 0E             JMP    SHORT H_0K
1279
1280 ;----- CHECK CLOCK UPDATE
1281 068C B9 0320             G_0K: MOV    CX,800 ; LOOP COUNT
1282 068F B0 8A             I_0K: MOV    AL,CMOS_REG_A+NMI ; CHECK FOR OPPOSITE STATE
1283 0693 EB 0000 E            CALL   CMOS_READ
1284 0694 A9 80             TEST   AL,80H ; CHECK FOR OPPOSITE STATE
1285 0696 E0 F7             LOOPNZ I_0K ; TRY AGAIN
1286 0698 E3 DD             JCXZ F_0K ; PRINT ERROR IF TIMEOUT
1287
1288 ;----- CHECK MEMORY SIZE DETERMINED = CONFIGURATION
1289 069A
1290 H_0K: MOV    AL,CMOS_DIAG+NMI ; GET THE STATUS BYTE
1291 069A B0 8E             I_0K: CALL   CMOS_READ ; CHECK FOR OPPOSITE STATE
1292 069B E8 0000 E            TEST   AL,W*MEM_SIZE ; WAS THE CONFIG= MEM_SIZE_DETERMINED?
1293 069C A8 10             JZ     H_0KTA ; GO IF YES
1294 069F A8 10             H_0KTA: MOV    AL,W*MEM_SIZE ; WAS THE CONFIG= MEM_SIZE_DETERMINED?
1295 06A1 74 06             H_0KTA: TEST   AL,80H ; GO IF YES
1296
1297 ;----- MEMORY SIZE ERROR
1298 06A3 BE 0000 E            MOV    $1,OFFSET E164 ; PRINT SIZE ERROR
1299 06A6 E8 0000 E            E_0K: CALL   E_MSG ; DISPLAY ERROR
1300
1301 ;----- CHECK FOR CRT ADAPTER ERROR
1302 06A9 80 3E 0015 R 0C             H_0KIA: CMP    @MFG_ERR_FLAG,0CH ; CHECK FOR MONOCHROME CRT ERROR
1303 06AE BE 0000 E            MOV    S1,OFFSET E401 ; LOAD MONOCHROME CRT ERROR
1304 06B1 74 0A             JZ     H_0KIB ; GO IF YES
1305
1306 06B3 80 3E 0015 R 0D             H_0KIB: CMP    @MFG_ERR_FLAG,0DH ; CHECK FOR COLOR CRT ADAPTER ERROR
1307 06B8 75 06             JNZ    J_0K ; CONTINUE IF NOT
1308 06B8 BE 0000 E            MOV    ST,OFFSET E501 ; CRT ADAPTER ERROR MESSAGE
1309 06B9 80 3E 0015 R 0D
1310 06B9 80 3E 0015 R 0D
1311 06B9
1312 06B0 EB 0000 E            H_0KIB: CALL   E_MSG
1313
1314 ;----- CHECK FOR MULTIPLE DATA RATE CAPABILITY
1315 06C0
1316 06C0 BA 03F1             J_0K: MOV    DX,03F1H ; D/S/P DIAGNOSTIC REGISTER
1317 06C3 EC             IN     AL,DX ; READ D/S/P TYPE CODE
1318 06C4 24 F8             AND    AL,11111000B ; KEEP ONLY UNIQUE CODE FOR D/S/P
1319 06C6 3C 50             CMP    AL,01010000B ; D/S/P CARD - MULTIPLE DATA RATE ?
1320

```



```

1435 0773 57          PUSH    DI          : SAVE WORK REGISTER
1436 0774 BF AA55      MOV     DI,0AA55H   : GET TEST PATTERN
1437 0777 2B DB        SUB     BX,BX    : SET BX=0000
1438 0778 BB 07        MOV     AX,[BX]   : GET 1ST WORD FROM MODULE
1439 077B 3B C7        CMP     AX,DI    : RECOVER THE REGISTER
1440 077D 00 00          POP    DI          : PROCEED TO NEXT ROM IF NOT
1441 077E 75 05        JNZ    NEXT_ROM  : GO CHECK OUT MODULE
1442 0780 E0 0000 E     CALL    ROM_CHECK : CHECK FOR END OF ROM SPACE
1443 0783 EB 04        JMP    SHORT_ARE_WEDONE : CHECK FOR END OF ROM SPACE

NEXT_ROM:          ADD    DX,0080H   : POINT TO NEXT 2K ADDRESS
1445 0785 B1 C2 0080      ARE_WEDONE: CMP    DX,0E000H   : AT E0000 YET?
1446 0785 B1 FA E000      JL     ROM_SCAN2 : GO CHECK ANOTHER ADD. IF NOT
1448 078D 7C E2
1449
1450 :---- TEST FOR KEYBOARD LOCKED
1451
1452 078F E0 0000 E     CALL    DDS        : SET DATA SEGMENT
1453 0792 E4 64        IN     AL,STATUS_PORT : IS KEYBOARD UNLOCKED?
1454 0794 24 10        AND    AL,KEYBD_INH
1455 0796 74 02        JZ    KEY1      : NO - SET ERROR FLAGS AND PRINT MESSAGE
1456 0798 EB 0B        JMP    SHORT_KEY10 : GO IF OFF
1457 079A
1458 079A 80 0E 0016 R 80      KEY1: OR     #MFG_ERR_FLAG+1,KEY_FAIL: <><><><><><><><><><><>
1459
1460 ASSUME DS:DATA
1461 079F BE 0000 E     MOV     SI,OFFSET E302 : PRINT LOCKED MESSAGE (302)
1462 07A2 E8 0000 E     CALL    E_MSG
1463 07A5 KEY10:        ;=====
1464
1465 ;=====
1466 ;=====
1467
1468 07A5 BF 09D5 R     MOV     DI,OFFSET F4 : OFFSET OF PRINTER ADDRESS TABLE
1469 07A8 BE 0000
1470 07AB
1471 07AB 2E: 8B 15      F16: MOV     DX,CS:[DI] : GET PRINTER BASE ADDRESS
1472 07B0 00 AA        MOV     AL,0AAH   : WRITE DATA TO PORT A
1473 07B0 EE
1474 07B1 EB 00
1475 07B3 1E
1476 07B4 EC
1477 07B5 1F
1478 07B6 00 AA        CMP     AL,0AAH   : DATA PATTERN SAME
1479 07B8 75 06        JNE    F17      : NO - CHECK NEXT PRINTER CARD
1480 07B8 89 94 0008 R     MOV     #PRINTER_BASE[SI],DX : YES - STORE PRINTER BASE ADDRESS
1481 07B8 46        INC    SI          : INCREMENT TO NEXT WORD
1482 07B8 46
1483 07C0
1484 07C0 47      F17: INC    DI          : POINT TO NEXT BASE ADDRESS
1485 07C1 47
1486 07C2 B1 FF 09DB R     CMP     DI,OFFSET F4E : ALL POSSIBLE ADDRESSES CHECKED?
1487 07C3 75 E3        JNE    F16      : PRT_BASE
1488
1489 ;=====
1490 ;=====
1491 07C8 BB 0000      ;=====
1492 07CB BA 03FA      ;=====
1493 07CE EC
1494 07CF A8 F8        TEST   AL,0F8H   : READ INTERRUPT ID REGISTER
1495 07D1 75 08        JNZ    F18      : BASE_END
1496 07D3 87 0000 R 03F8      MOV     #RS232_BASE[BX],3F8H : SETUP RS232 CARD #1 ADDRESS
1497 07D9 43
1498 07DA 43
1499 07DB BA 02FA      F18: MOV     DX,2F4H   : CHECK IF RS232 CARD 2 ATTACHED
1500 07DE EC        IN     AL,DX    : READ INTERRUPT ID REGISTER
1501 07DE A8 F8        TEST   AL,0F8H
1502 07E7 75 08        JNZ    F19      : BASE_END
1503 07E3 87 0000 R 02F8      MOV     #RS232_BASE[BX],2F8H : SETUP RS232 CARD #2
1504 07E9 43
1505 07EA 43
1506
1507 ;=====
1508 ;=====
1509 07EB BB 0000      ;=====
1510 07EB 8B C6        F19: MOV     AX,51 : BASE_END
1511 07ED B1 03        MOV     CL,3  : SHIFT COUNT
1512 07EF D2 C8        ROR    AL,CL   : ROTATE RIGHT 3 POSITIONS
1513 07F1 0A C3        OR     AL,BL   : OR IN THE PRINTER COUNT
1514 07F3 A2 0011 R     MOV     BYTE PTR #EQUIP_FLAG+1,AL : STORE AS SECOND BYTE
1515
1516
1517
1518 07F6 EB 0000 E     ;=====
1519 ;=====
1520 ;=====
1521 ;=====
1522 07F9 B0 40      ;=====
1523 07FB E6 80        ;=====
1524
1525 07FB BF 0067 R     MOV     AL,40H   : ADDRESS WORK STORAGE LOCATION
1526 0800 33 C0        OUT    MFG_PORT,AL : CLEAR WORK REGISTER (AH=0 IN 287)
1527 0800 89 05        XOR    AX,AX   : CLEAR THE WORK LOCATION
1528 0804 00 00        MOV     WORD PTR [DI],AX : INITIALIZE THE 2827 WITH NO WAIT
1529 0806 EB 00        FNINIT
1530 0806 D9 3D        JMP    $-2      : DELAY
1531 080A 60        FNSTSW WORD PTR [DI] : WRITE THE CURRENT 2827 CONTROL WORD
1532 0808 61        POPA
1533 0808 B1 25 1F3F      AND    WORD PTR [DI],01F3FH : CLEAR UNUSED 2827 BITS
1534 0810 B1 3D 033F      CMP    WORD PTR [DI],0033FH : IS THE 2827 INSTALLED?
1535 0814 75 13        JNE    NO_287 : GO IF MATH PROCESSOR IS NOT INSTALLED
1536
1537 0816 9B DD 3D      FSTSW WORD PTR [DI] : STORE THE STATUS WORD (WITH WAIT)
1538 0819 60        PUSHA
1539 081A 61        POPA
1540 081A F7 05 B8BF      TEST   WORD PTR [DI],0B8BFH : ALL BITS SHOULD BE OFF (OR ERROR)
1541 081A 75 08        JNZ    NO_287 : GO IF NOT INSTALLED
1542
1543 0821 E4 A1        IN     AL,INTB01 : GET THE SLAVE INTERRUPT MASK
1544 0823 24 DF        AND    AL,0DFH : ENABLE 2827 INTERRUPTS
1545 0825 B4 E2        MOV     AH,002H : SET WORK REGISTER FOR 2827 FOUND
1546 0826 E6 A1        OUT    INTB01,AL
1547 0829
1548 0829 A0 0010 R     NO_287: MOV     AL,RYTE PTR #EQUIP_FLAG : GET LOW EQUIPMENT FLAG

```

```

1549 082C 24 02          AND    AL,002H          ; STRIP OFF OTHER BITS
1550 082E 3A C4          CMP    AL,AH          ; DOES CMOS MATCH HARDWARE ?
1551 0830 74 08          JE     OK_287          ; SKIP IF EQUIPMENT FLAG CORRECT
1552
1553 0832 80 36 0010 R 02          XOR    BYTE PTR @EQUIP_FLAG,2H ; ELSE SET 80287 BIT TO CORRECT VALUE
1554 0837 E8 0000 E          CALL   CONFIG_BAD          ; AND SET THE CONFIGURATION ERROR FLAG
1555 083A
1556          OK_287: ;----- SET KEYBOARD STATE FLAGS
1557
1558 083A C7 06 0017 R 0000          MOV    WORD PTR @KB_FLAG,0 ; RESET ALL KEYBOARD STATUS FLAGS
1559
1560          ;----- ENABLE KEYBOARD/TIMER INTERRUPTS
1561
1562 0840 E4 21          IN     AL,INTA01          ; IN AL,INTA01
1563 0842 24 FC          AND    AL,10FCH          ; AND AL,10FCH
1564 0844 EB 00          JMP    $+2             ; JMP $+2
1565 0846 E6 21          OUT   INTA01,AL          ; OUT INTA01,AL
1566 0848 C6 06 0015 R 00          MOV    @MFG_ERR_FLAG,0 ; CLEAR MFG ERROR FLAG
1567
1568          ;----- READ KEYBOARD ID TO INITIALIZE KEYBOARD TYPE AND NUM LOCK STATE
1569
1570 084D C6 06 0096 R A0          MOV    @KB_FLAG,3,RD_ID+SET_NUM_LK ; SET READ ID COMMAND FOR KBX
1571 0852 B0 F2          AL,KB_READ_ID          ; AL,KB_READ_ID
1572 0854 E8 0000 E          CALL   SND_DATA          ; CALL SND_DATA
1573 0857 B9 067A          MOV    CX,T658          ; MOV CX,T658
1574 0858 E8 0000 E          CALL   WAITF          ; CALL WAITF
1575 085D 80 26 0096 R 1F          AND    @KB_FLAG,3,NOT RD_ID+LC_ABSET_NUM_LK ; RESET READ ID COMMAND
1576
1577          ;----- CHECK FOR SECOND FIXED DISK PRESENT BUT NOT DEFINED
1578
1579 0862 80 3E 0075 R 02          CMP    @HFN_NUM,2          ; CHECK FOR TWO DRIVES DEFINED BY CMOS
1580 0867 74 13          JE     F15G             ; SKIP TEST IF TWO DRIVES DEFINED
1581
1582 0869 B4 10          MOV    AH,010H          ; GET TEST DRIVE READY COMMAND
1583 086B B2 81          MOV    DL,001H          ; POINT TO SECOND FIXED DISK
1584 086D FE 06 0075 R          INC    @HFN_NUM          ; INC @HFN_NUM
1585 0871 CD 13          INT    13H             ; TELL BIOS IT HAS TWO DRIVES
1586 0873 FE 0E 0075 R          DEC    @HFN_NUM          ; DEC @HFN_NUM
1587 0877 T2 03          JC    F15G             ; JC F15G
1588
1589 0879 E8 0000 E          CALL   CONFIG_BAD          ; SET CONFIGURATION BAD
1590 087C
1591          ;----- TEST FOR ANY ERRORS (BP NOT ZERO)
1592
1593
1594
1595 087C 0B ED          OR    BP,BP          ; CHECK (BP)= NON-ZERO (ERROR HAPPENED)
1596 087E 74 55          JE     F15A_0          ; SKIP PAUSE IF NO ERROR
1597
1598 0880 80 3E 0072 R 64          CMP    BYTE PTR @RESET_FLAG,64H ; MFG RUN IN MODE?
1599 0885 BA 0002          MOV    DX,2             ; 2 SHORT BEEP COUNT FOR ERROR(S)
1600 0888 75 0E          JNZ   ERR_WAIT          ; GO IF NOT
1601
1602          ;----- MFG RUN IN MODE -> SET ERROR FLAG
1603
1604 088A C6 06 0015 R AA          MOV    @MFG_ERR_FLAG,0AAH ; INDICATE ERROR
1605 088F E4 64          IN     AL,STATUS_PORT          ; CHECK KEY LOCK STATUS
1606 0891 24 10          AND    AL,KYBD_INH          ; IS THE KEYBOARD LOCKED
1607 0893 75 40          JNZ   F15A_0          ; CONTINUE MFG MODE IF NOT LOCKED
1608
1609 0895 BA 0005          MOV    DX,5             ; ELSE
1610 0897 E8 0000 E          ERW_WAIT1          ; 5 SHORT BEEPS FOR MFG SETUP ERROR
1611 0898 E8 0000 E          CALL   ERR_BEEP          ; BEEPS FOR ERROR(S)
1612 0899 B0 0E          MOV    AL,CMOS_DIAG          ; ADDRESS CMOS
1613 089D E8 0000 E          CALL   CMOS_READ          ; GET THE DIAGNOSTIC STATUS BYTE
1614 08A0 A8 20          TEST   AL,BAD_CONFIG          ; CHECK FOR BAD HARDWARE CONFIGURATION
1615 08A2 74 0C          JZ    ERR_KKEY          ; SKIP IF NOT SET
1616
1617 08A4 F7 C5 8000          TEST   BP,08000H          ; ELSE CHECK FOR E161/E162 POSTED
1618 08A8 75 06          JNZ   ERR_KKEY          ; SKIP IF DISPLAYED BEFORE NOW
1619
1620 08A8 BE 0000 E          MOV    SI,OFFSET E162          ; ELSE DISPLAY "OPTIONS NOT SET"
1621 08AD E8 0000 E          CALL   P_MSG             ; WITH NON HALTING ROUTINE
1622
1623          ;----- CHECK FOR "UNLOCK SYSTEM UNIT KEYLOCK" MESSAGE REQUIRED
1624
1625 08B0
1626 08B0 E4 64          ERW_KKEY: ;----- IN AL,STATUS_PORT          ; CHECK IF RESUME MESSAGE NEEDED
1627 08B2 24 10          AND    AL,KYBD_INH          ; IS THE KEYBOARD LOCKED
1628 08B4 75 06          JNZ   ERR_WAIT2          ; SKIP LOCK MESSAGE IF NOT
1629
1630 08B6 BE 0000 E          MOV    SI,OFFSET F3D1          ; ERROR MESSAGE FOR KEYBOARD LOCKED
1631 08B9 E8 0000 E          CALL   P_MSG             ; P_MSG
1632
1633          ;----- DISPLAY '(RESUME = "F1" KEY)' FOR ERRORS
1634
1635 08BC
1636 08BC BE 0000 E          ERW_WAIT2: ;----- MOV    SI,OFFSET F3D          ; RESUME ERROR MESSAGE
1637 08BF E8 0000 E          CALL   P_MSG             ; P_MSG
1638
1639          ;----- INITIALIZE PRINTER (ALTERNATE DISPLAY DEVICE)
1640
1641 08C2 B4 01          MOV    AH,1             ; FIRST PRINTER
1642 08C4 2B D2          SUB    DX,DX
1643 08C6 CD 17          INT    17H
1644 08C8
1645 08C8 B0 3F          ERW_WAIT1: ;----- MOV    AL,3FH          ; <><><><><><><><><><><><>
1646 08CA E8 80          OUT   MFG_PORT,AL          ; <><> CHECKPOINT 3F <><>
1647 08C8 C4 00          MOV    AL,00
1648 08CE CD 16          INT    16H          ; WAIT FOR 'F1' KEY
1649 08D0 80 FC 3B          CMP    AH,3BH
1650 08D3 75 F3          JNE   ERR_WAIT1
1651 08D5
1652 08D5 F6 06 0012 R 20          F15A_0: ;----- TEST   @MFG_TST,MFG_LOOP          ; MFG BURN IN MODE
1653 08D9 75 04          JNZ   F15A_0          ; JNZ F15A_0
1654 08DC E9 0000 E          JMP    START_1          ; JMP START_1
1655 08DF 80 3E 0072 R 64          F15A: ;----- CMP    BYTE PTR @RESET_FLAG,64H ; MFG RUN IN?
1656 08E4 74 06          JZ    F15B             ; JZ F15B
1657
1658 08E6 BA 0001          MOV    DX,1             ; 1 SHORT BEEP (NO ERRORS)
1659 08E9 E8 0000 E          CALL   ERR_BEEP          ; ERR_BEEP
1660
1661          ;----- SET TIME OF DAY
1662

```

```

1663
1664 08EC E0 0000 E
1665
1666 ;----- CLEAR DISPLAY SCREEN
1667
1668 08EF 2A E4
1669 08F1 A0 0049 R
1670 08F4 CD 10
1671
1672 ;----- CLEAR DESCRIPTOR TABLES
1673
1674 08F6 B9 01F4
1675 08F9 B0 D0A0
1676 08FC 2B C0
1677 08FE 8E C0
1678 0900 26: 89 05
1679 0903 85 C7 02
1680 0906 E2 F8
1681
1682
1683 ;----- SET POST SYSTEM STACK
1684 0908 B8 ---- R
1685 090B B8 D0
1686 090D BC 0400 R
1687
1688 ;----- ENSURE THAT MASTER LEVEL 2 ENABLED
1689
1690 0910 E4 21
1691 0912 24 FB
1692 0914 E0 00
1693 0916 E6 21
1694
1695 ;----- TEST FOR MFG RUN-IN TEST
1696
1697 0918 80 3E 0072 R 64
1698 091D 75 02
1699 091F E5 5C
1700
1701 ;----- UNMASK SLAVE HARDWARE INTERRUPT 9 (LEVEL 7)
1702 0921 E4 A1
1703 0923 24 FD
1704 0925 EB 00
1705 0927 E6 A1
1706
1707
1708
1709 ;----- TEST FOR SYSTEM CODE AT SEGMENT E000:00
1710 ; FIRST WORD = AA55H
1711 ; LAST BYTES = CHECKSUM
1712 ; ENTRY POINT = ENTRY POINT + BYTE + 3
1713 ; IF TEST IS SUCCESSFUL A CALL FAR TO THE ENTRY POINT IS EXECUTED
1714
1715 0929 B0 41
1716 092B E6 80
1717
1718 092D B0 8D
1719 092F E6 70
1720
1721
1722 ENDF
1723 0931 C6 06 0072 R 00
1724 0936 B8 E000
1725 0939 8E C0
1726 093B 2B FF
1727 093D 26: B8 05
1728 0940 53
1729 0941 5B
1730 0942 5C AA55
1731 0943 9C
1732 0946 26: 89 05
1733 0949 E4 61
1734 094B 00 0C
1735 094D E0 61
1736 0950 26: F3
1737 0951 56 61
1738 0953 9D
1739 0954 75 27
1740
1741
1742
1743 0956 1E
1744 0957 06
1745 0958 1F
1746 0959 2B DB
1747 095B E8 0000 E
1748 095E 1F
1749 095F 75 1C
1750
1751 ;----- ENABLE NMI AND I/O-MEMORY PARITY CHECKS
1752
1753 0961 B0 0D
1754 0963 E0 70
1755
1756 0965 E4 61
1757 0967 24 F3
1758 0969 E6 61
1759
1760 096B C7 06 0067 R 0003
1761 0971 8C 06 0069 R
1762
1763 0975 B0 42
1764 0977 E6 80
1765
1766 ;----- EXIT TO SYSTEM CODE
1767
1768 0979 FF 1E 0067 R
1769
1770
1771 ;----- ENABLE NMI INTERRUPTS + ENTRY FROM SHUTDOWN WITH BOOT REQUEST
1772
1773
1774 097D B0 0D
1774 097F E6 70
1775 0981 E4 61
1776 0983 24 F3
1777
1778 ;----- SHUT4: CALL DWORD PTR _IO_ROM_INIT ; GO TO SYSTEM CODE
1779 ; VIA CALL THROUGH DATA AREA LOCATION
1780
1781 ;----- SHUT4: MOV AL,CMOS_REG_D ; ENABLE NMI AND SET DEFAULT ADDRESS
1782 ; OUT CMOS_PORT_AL
1783 ; IN AL,PORT_B ; ENABLE PARITY
1784 ; AND AL,RAM_PAR_ON ; ENABLE MEMORY PARITY CHECK / I/O CHECK

```

```

1777 0985 E0 61      OUT    PORT_B,AL
1778
1779 0987 B0 43      MOV    AL,43H
1780 0989 E6 80      OUT    MFG_PORT,AL
1781 0988 FB          STI
1782
1783 098C CD 19      INT    19H
1784
1785 098E F4          HLT
1786
1787
1788 098F             PADING PROC  NEAR
1789 098F B9 000F      MOV    CX,15
1790 0992             PADI1:  MOV    AL,' '
1791 0992 B0 20      CALL   PRT_HEX
1792 0992 E8 0000 E
1793 0997 E2 F9      LOOP   PADT
1794 0999 B0 2D      MOV    AL,'-'
1795 0998 E0 0000 E
1796 099E C3          CALL   PRT_HEX
1797 099F             RET
1798             ENDP
1799
1800 099F             PRT_OK PROC  NEAR
1801 099F 50          PUSH   AX
1802 09A0 BB 000A      MOV    BX,10
1803
1804             ----- CONVERT AND SAVE
1805
1806 09A3 B9 0005      MOV    CX,5
1807 0946 2B FF      SUB    DI,DI
1808 0948             PRT_DIV:  XOR    DX,DX
1809 0948 33 D2      DIV    DX
1810 0948 F7 F3      OR     DX,30H
1811 09AC 50 CA 30
1812 09AF 52          PUSH   DX
1813 09B0 E2 F6      LOOP   PRT_DIV
1814
1815             ----- DISPLAY LAST OK MEMORY
1816
1817 09B2 B9 0005      MOV    CX,5
1818 09B5             PRT_DEC:  POP    AX
1819 09B5 58          CALL   PROT_PRT_HEX
1820 09B6 E0 0000 E
1821 09B9 47          INC    DI
1822 09B9 F2 F9      LOOP   DEC
1823 09B8 B9 0007      MOV    CX,0FFSET F3B_PAD-OFFSET F3B
1824 09B8 BE 09CE R
1825 09C2             PRT_LOOP: MOV    SI,0FFSET F3B
1826 09C2 2E: 8A 04      MOV    AL,CS:[SI]
1827 09C5 46          INC    SI
1828 09C5 E0 0000 E
1829 09C9 47          CALL   PROT_PRT_HEX
1830 09CA E2 F6      INC    SI
1831 09CC 52          LOOP   PRT_LOOP
1832 09CD C3          POP    AX
1833             RET
1834 09CE 20 4B 42 20 4F 4B  F3B   DB    ' KB OK'
1835 09D4 20          F3B_OK DB    '$'
1836  = 09D5          F3B_PAD EQU   $15T
1837
1838 09D5             PRT_OK ENDP
1839
1840             ----- PRINTER TABLE : -----
1841
1842
1843
1844 09D5 03BC          F4    DW    03BCH
1845 09D7 0378          DW    0378H
1846 09D9 0278          DW    0278H
1847 09DB              F4E   LABEL  WORD
1848
1849 09DB              POST2 ENDP
1850 09DB              CODE  ENDS
1851              END

```







```

343 ;----- ERROR EXIT
344
345 0188 ERROR_EXIT:    JMP    ERROR_EXIT
346
347
348
349 ;----- VERIFY ACCESS RIGHTS FUNCTION CORRECTLY :
350 ; DESCRIPTION : SET ACCESS RIGHTS OF DESCRIPTOR TO
351 ; READ ONLY.  VERIFY THE VERW/VERW INSTR :
352 ; ACCESS A READ ONLY WITH A WRITE AND
353 ; VERIFY AN EXCEPTION INTERRUPT 13
354
355
356
357 018B B0 F6    TT_10: MOV    AL,0F6H      ; <><><><><><><><><>
358 018D E6 80    OUT   MFG_PORT_AL    ; <><> CHECKPOINT F6 <><>
359 018F C7 00 004D FFFF    MOV    DS:ES_TEMP,SEG LIMIT,MAX SEC LEN ; SET SEGMENT TO 0FFFFH
360 0195 C6 00 004C 00    MOV    BYTE PTR DS:(ES_TEMP-BASE_HI-BYTE),0 ; SET THE ADDRESS
361 0197 C6 00 004A F000    MOV    DS:ES_TEMP,SEG LIMIT,MAX SEC LEN ; SET SEGMENT TO 0FFFFH
362 01A0 B8 0048    MOV    AX,ES_TEMP      ; LOAD ES REGISTER
363 01A3 BE C0    MOV    ES,AX      ; THIS SEGMENT SHOULD BE WRITEABLE
364
365 ;----- INSURE ACCESS RIGHTS MAY BE WRITTEN
366
367 01A5 3E    SEGOV DS      ; SET SEGMENT OVERRIDE TO START OF TABLE
368
369 01A6 0F    VERW AX      ; CHECK THE ACCESS RIGHTS OF ES_TEMP
370 01A7 0F    DB    0F0H      ; SET SEGMENT OVERRIDE TO START OF TABLE
371 01A7 0F    ?0009 LABEL  BYTE
372 01A7 0B E8    MOV    BYTE BP,AX
373 01A7 0F    ?000A LABEL  BYTE
374 01A7 0F    ORG    OFFSET CS:??0009
375 01A7 00    DB    000H      ; SET SEGMENT OVERRIDE TO START OF TABLE
376 01A9 0A    ORG    OFFSET CS:??000A
377 01A9 75 DD    JNZ   ERROR_EXITI   ; ERROR IF SEGMENT CAN NOT WRITE
378
379 ;----- SET ACCESS RIGHTS TO READ ONLY
380
381 01AB C6 06 004D 91    MOV    BYTE PTR DS:(ES_TEMP.DATA_ACC_RIGHTS),91H
382 01B0 B8 0048    MOV    AX,ES_TEMP      ; LOAD ES REGISTER
383 01B3 BE C0    MOV    ES,AX      ; SET SEGMENT OVERRIDE TO START OF TABLE
384
385 01B5 3E    SEGOV DS      ; SET SEGMENT OVERRIDE TO START OF TABLE
386
387 01B6 0F    VERW AX      ; CHECK THE ACCESS RIGHTS OF ES_TEMP
388 01B7 0F    DB    0F0H      ; SET SEGMENT OVERRIDE TO START OF TABLE
389 01B7 0B E8    ?000C LABEL  BYTE
390 01B7 0F    ?000D LABEL  BYTE
391 01B7 0F    ORG    OFFSET CS:??000C
392 01B7 00    DB    000H      ; SET SEGMENT OVERRIDE TO START OF TABLE
393 01B7 0F    ORG    OFFSET CS:??000D
394 01B9 74 CD    JZ    ERROR_EXITI   ; ERROR IF SEGMENT IS WRITEABLE
395
396 01BB B8 0048    MOV    AX,ES_TEMP      ; INSURE THAT SEGMENT IS READABLE
397
398 01BE 3E    SEGOV DS      ; SET SEGMENT OVERRIDE TO START OF TABLE
399
400 01BF 0F    VERW AX      ; CHECK THE ACCESS RIGHTS OF ES_TEMP
401 01C0 0F    ?2000F LABEL  BYTE
402 01C0 BB E0    MOV    SP,AX
403 01C2 0F    ?20010 LABEL  BYTE
404 01C2 0F    ORG    OFFSET CS:??2000F
405 01C2 00    DB    000H      ; SET SEGMENT OVERRIDE TO START OF TABLE
406 01C2 0F    ORG    OFFSET CS:??20010
407 01C2 75 C4    JNZ   ERROR_EXITI   ; GO IF SEGMENT NOT READABLE
408
409 ;----- CAUSE AN EXCEPTION 13 INTERRUPT
410
411 01C4 B0 9D    MOV    AL,09DH      ; SET EXCEPTION FLAG
412 01C4 E4 8B    OUT   MFG_PORT_AL    ; FOR INTERRUPT 13
413 01C5 2B F6    SUB   SI,SI      ; SET THE RPL FIELD OF A SELECTOR
414 01CA 26: C6 04 00    MOV    BYTE PTR ES:[SI],00 ; WRITE A BYTE THAT SHOULD
415 01CE 2B C9    SUB   CX,CX      ; WAIT FOR INTERRUPT
416 01D0 E4 8B    LOOPD IN,AL,DMA_PAGE+0AH ; DID THE INTERRUPT OCCUR?
417 01D2 22 C0    AND   AL,AL
418 01D4 FA 00    LOOPNZ IN,AL,DMA_PAGE+0AH
419 01D6 75 B0    JNZ   ERROR_EXITI   ; MISSING INTERRUPT
420
421 ;----- RESTORE THE ACCESS RIGHTS BYTE
422
423 01D8 C6 06 004D 93    MOV    BYTE PTR DS:(ES_TEMP.DATA_ACC_RIGHTS),CPL0_DATA_ACCESS
424
425 ;----- VERIFY ADJUST RPL FIELD OF SELECTOR
426 ; INSTRUCTION (ARPL) FUNCTIONS
427 ; DESCRIPTION : SET THE RPL FIELD OF A SELECTOR
428 ; AND VERIFY THAT THE ZERO FLAG IS SET
429 ; CORRECTLY AND THAT THE SELECTOR RPL
430 ; FIELD IS SET CORRECTLY
431
432
433
434
435 01DD B0 F7    MOV    AL,0F7H      ; <><><><><><><><><>
436 01E7 E6 80    OUT   MFG_PORT_AL    ; <><> CHECKPOINT F7 <><>
437 01E7 00 0048    MOV    AX,ES_TEMP      ; PUT A SELECTOR IN AX
438 01E4 BB 0040    MOV    BX,DS:TEMP      ; PUT A SELECTOR IN BX
439 01E7 0D 0003    OR    AX,03H      ; MAKE ACCESS OF AX < BX
440
441 ;----- NOTE BX = FIRST OPERAND AX = SECOND OPERAND
442
443
444 01E8 00 0001    ARPL AX,BX      ; ISSUE THE RPL COMMAND
445 01E8 BB C3    ?20011 LABEL  BYTE
446 01E8 00 0002    ?20012 LABEL  BYTE
447 01E8 00 0001    ?00011 LABEL  BYTE
448 01E8 63    ?00012 LABEL  BYTE
449 01E8 00 0001    ?00011 LABEL  BYTE
450 01E8 75 9A    ?00012 LABEL  BYTE
451 01E8 00 0001    ?00011 LABEL  BYTE
452 01F1 80 E3 03    AND   BL,03H      ; STRIP UNWANTED BITS
453 01F1 80 F0 03    CMP   BL,03H      ; AS EXPECTED?
454 01F4 75 92    JNZ   ERROR_EXITI   ; GO IF NOT
455
456 ;----- CHECK THAT ACCESS RIGHTS DO NOT CHANGE
457
458

```

```

457 01F6 BB 0060      MOV    BX,DS TEMP          ; PUT A SELECTOR IN BX
458 01F9 BB 0048      MOV    AX,ES TEMP          ; PUT A SELECTOR IN AX
459 01FC 80 CB 03      OR     BL,03H             ; MAKE ACCESS OF BX < AX
460
461      ;----- NOTE BX = FIRST OPERAND AX = SECOND OPERAND
462
463 01FF               ARPL   AX,BX             ; ISSUE THE RPL COMMAND
464 01FF 8B C3          + ?70013  LABEL  BYTC
465 0201               + ?70014  LABEL  BYTE
466 01FF               + ?70014  ORG    OFFSET CS:70013
467 01FF 63             + ?70014  ORG    OFFSET CS:70014
468 01FF               JZ    ERROR_EXIT1
469 0201 74 85          AND    BL,03H             ; GO IF RPL WAS NOT CHANGED
470 0203 80 E3 03      CMP    BL,03H             ; STRIP UNWANTED BITS
471 0206 80 FB 03      CMP    BL,03H             ; AS EXPECTED?
472 0209 75 2F          JNZ    ERROR_EXIT2
473
474      ;----- VERIFY LOAD SEGMENT LIMIT (LSL)
475      ;----- AND LOAD ACCESS RIGHTS (LAR) INSTRUCTION
476
477 0201               ;----- CHECK THE LAR INSTRUCTION
478
479 020B B0 F8          MOV    AL,0FBH             ; <><><><><><><><><>
480 020D E6 80          OUT   MFG_PORT,AL          ; <><> CHECKPOINT F8 <><>
481
482 0214               ;----- SET THE DESCRIPTOR TO LEVEL 3
483
484 020F C6 06 004D F3  MOV    BYTE PTR DS:(ES_TEMP.DATA_ACC_RIGHTS),CPL3_DATA_ACCESS
485 0214 BB 0048          MOV    BX,ES TEMP          ; CLEAR AX
486 0217 2B C0          SUB    AX,AX
487
488 0219               ;----- GET THE CURRENT DESCRIPTORS ACCESS RIGHTS
489
490 0219 0F             LAR    AX,BX             ; ISSUE THE LAR COMMAND
491 021A               + ?70015  LABEL  BYTC
492 021A 8B C3          + ?70016  LABEL  BYTE
493 021C               + ?70016  ORG    OFFSET CS:70015
494 021A               + ?70016  ORG    OFFSET CS:70016
495 021C
496 0219               ;----- INSURE THE DESCRIPTOR WAS VISIBLE
497 021A 02             JNZ    ERROR_EXIT2
498 021C               ; GO IF LAR WAS NOT CHANGED
499
500 021C 75 1C          ;----- THE DESCRIPTORS ACCESS RIGHTS MUST BE 3
501
502 021E 80 FC F3      CMP    AH,CPL3_DATA_ACCESS
503 0221 75 17             JNZ    ERROR_EXIT2
504
505 0221               ;----- CHECK THE LSL (LOAD SEGMENT LIMITS)
506
507 0223 B0 F9          MOV    AL,0F9H             ; <><><><><><><><>
508 0225 E0 60          OUT   MFG_PORT,AL          ; <><> CHECKPOINT F9 <><>
509 0227 C7 06 0048 AAAA DS:ES_TEMP.SEG_LIMIT,0AAAH
510
511 022D C6 06 004D 93  MOV    BYTE PTR DS:(ES_TEMP.DATA_ACC_RIGHTS),CPL0_DATA_ACCESS
512 0230 BB 0048          MOV    BX,ES TEMP          ; LOAD ES REGISTER
513 0236 C7 06 0048      DB    00H
514 0235 0F               + ?70017  LABEL  BYTC
515 0236               + ?70017  LABEL  BYTE
516 0236 BB D8          + ?70018  LABEL  BYTC
517 0236               + ?70018  LABEL  BYTE
518 0236 03             + ?70018  ORG    OFFSET CS:70017
519 0236               + ?70018  ORG    OFFSET CS:70018
520 0236 74 03          JZ    ROT
521
522 023A               ;----- GET IF OK
523 023A E9 02CD R      ERROR_EXIT2:           ; GO IF NOT SUCCESSFUL
524
525 023D 81 FB AAAA      JMP   ERROR_EXIT          ; GO IF NOT
526
527 0241 C7 06 0048 5555 R07:  CMP    BX,0AAAH             ; INSURE CORRECT SEGMENT LIMIT
528 0241               MOV    DS:ES TEMP.SEG_LIMIT,0555H
529 0241               ; SET THE SEGMENT LIMIT TO 0555H
530 0241 BB 0048          MOV    AX,ES TEMP          ; GET THE DESCRIPTOR SEGMENT LIMIT
531 0242 8B D8          LSL    BX,AX
532 0242               ;----- GET THE DESCRIPTOR SEGMENT LIMIT
533
534 024A 0F             + ?70019  LABEL  BYTC
535 024B               + ?70019  LABEL  BYTE
536 024B BB D8          + ?7001A  LABEL  BYTC
537 024D               + ?7001A  LABEL  BYTE
538 024B               + ?70019  ORG    OFFSET CS:70019
539 024B 03             + ?70019  DB    03H
540 024D               + ?7001A  ORG    OFFSET CS:7001A
541 024D 75 EB          JNZ    ERROR_EXIT2
542
543 024F 81 FB 5555      CMP    BX,0555H             ; GO IF NOT SUCCESSFUL
544 0253 75 E5          JNZ    ERROR_EXIT2
545
546 0255 B0 FA          ;----- LOW MEG CHIP SELECT TEST
547 0257 E6 80          ; TEST THAT A WRITE TO ADDRESS 1B0000 DOES NOT WRITE
548 0259 6A 08          ; TO B00010, OR 1B8000 DOES NOT WRITE TO B800:0
549
550
551 0255 BB 0048          MOV    AL,0FAH             ; <><><><><><><><>
552 0256 6A 08          OUT   MFG_PORT,AL          ; <><> CHECKPOINT FA <><>
553 0258 1F               PUSH  BYTE PTR GDT_PTR
554 0258               POP    DS
555
556 0259               ;----- SET TEMPORARY ES DESCRIPTOR 64K SEGMENT LIMIT/CPL0 DATA ACCESS
557
558 025C C7 06 0048 FFFF  MOV    DS:ES TEMP.SEG_LIMIT,MAX_SEG_LEN
559 0262 C6 06 004D 93  BYTE PTR DS:(ES_TEMP.DATA_ACC_RIGHTS),CPL0_DATA_ACCESS
560
561 0271               ;----- START WITH SEGMENT 1B0000
562
563 0267 C6 06 004C 1B  MOV    BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE),1BH
564 026C C6 06 004A 0000  MOV    DS:ES TEMP.BASE_LO_WORD,0
565 0272 6A 48          PUSH  BYTE PTR ES TEMP
566 0274 07          POP    ES
567 0275 2B FF          SUB    DI,DI
568 0277 26: C7 05 AA55  MOV    WORD PTR ES:[DI],0AA55H
569
570

```

```
571      ;---- DO FOR SEGMENT 1B8000
572      027C C7 06 004A 8000      MOV DS:ES TEMP.BASE_LO_WORD,8000H
573      0282 6A 48      PUSH DS:ES TEMP ; LOAD ES REGISTER
574      0284 01      POP ES
575      0285 26: C7 05 AA55      MOV WORD PTR ES:[DI],0AA55H ; WRITE A TEST PATTERN
576
577      ;---- DO FOR SEGMENT 1A0000
578
579      028A C6 06 004C 1A      MOV BYTE PTR DS:(ES TEMP.BASE_HI_BYTE),1AH
580      028F C7 06 004A 0000      MOV DS:ES TEMP.BASE_LO_WORD,0
581      0295 6A 48      PUSH DS:ES TEMP ; LOAD ES REGISTER
582      0297 07      POP ES
583      0298 26: C7 05 AA55      MOV WORD PTR ES:[DI],0AA55H ; WRITE A TEST PATTERN
584
585      ;---- B/W VIDEO CARD
586
587      029D 6A 20      PUSH BYTE PTR C_BWCRT_PTR
588      029F 1F      POP DS
589      02A0 8B 05      MOV AX,DS:[DI] ; SET DS TO B/W DISPLAY REGEN BUFFER
590
591      ;---- COMPATIBLE COLOR
592
593
594      02A2 6A 28      PUSH BYTE PTR C_CCRT_PTR ; SET DS TO COMPATIBLE COLOR MEMORY
595      02A4 1F      POP DS
596      02A5 8B 1D      MOV BX,DS:[DI] ; GET THE WORD FROM COLOR MEMORY
597
598      ;---- EGA COLOR
599
600      02A7 6A 30      PUSH BYTE PTR E_CCRT_PTR ; EGA COLOR CRT POINTER LOW 64K
601      02A9 1F      POP DS
602      02AA 8B 0D      MOV CX,DS:[DI]
603
604      ;---- TEST FOR ERROR
605
606      02AC 50      PUSH AX ; SAVE RESULTS
607      02AD B0 35      MOV AL,35H ; <><><><><><><><><>
608      02AF E6 80      OUT MFG_PORT,AL ; <><> CHECKPOINT 35 <><>
609      02B1 58      POP AX
610      02B2 3D AA55      CMP AX,0AA55H
611      02B3 74 10      JZ ERROR_EXIT
612      02B7 B1 FB AA55      CMP BX,0AA55H
613      02B8 74 10      JZ ERROR_EXIT
614      02BD B1 F9 AA55      CMP CX,0AA55H
615      02C1 74 0A      JZ ERROR_EXIT
616      02C3 B0 34      MOV AL,34H ; RESTORE CHECKPOINT
617      02C5 E6 80      OUT MFG_PORT,AL ; <><> CHECKPOINT 34 <><>
618
619      ;---- SHUTDOWN
620
621      02C7      NORMAL_EXIT:
622      02C7 B8 068F      MOV AX,6*M+CMOS_SHUT_DOWN+NMI ; ADDRESS FOR SHUTDOWN BYTE
623      02C8 E8 0000 E      CALL CMOS_WRITE ; SET GOOD ENDING
624      02CD      ERROR_EXIT:
625      02CD E9 0000 E      JMP PROC_SHUTDOWN
626
627      02D0      POST3 ENDP
628
629      02D0      CODE ENDS
630      END
```

```

1 PAGE 118,121
2 TITLE TEST4 ---- 06/10/85 POST AND BIOS UTILITY ROUTINES
3 .286C
4 .LIST
5 0000  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 DIS
13 PUBLIC DDS
14 PUBLIC DUMMY_RETURN_1
15 PUBLIC ERR_BEEP
16 PUBLIC E_MSG
17 PUBLIC INT_287
18 PUBLIC KBD_RESET
19 PUBLIC LST
20 PUBLIC PROT_PRT_HEX
21 PUBLIC PROC_SHUTDOWN
22 PUBLIC PRT_HEX
23 PUBLIC PRT_SEG
24 PUBLIC PMS
25 PUBLIC RE_DIRECT
26 PUBLIC ROM_CHECK
27 PUBLIC ROM_CHECKSUM
28 PUBLIC SET_TOD
29 PUBLIC WAIT
30 PUBLIC XPC_BYTE
31
32 EXTRN E163:NEAR
33 EXTRN OBF_42:NEAR
34 EXTRN ROM_ERR:NEAR
35 EXTRN XMIT_8042:NEAR
36
37 ASSUME CS:CODE,DS:DATA
38 0000
39 POST4: ----- CMOS_READ ----- READ BYTE FROM CMOS SYSTEM CLOCK CONFIGURATION TABLE
40
41 INPUT: (AL)= CMOS TABLE ADDRESS TO BE READ
42 BIT 7 = 0 FOR NMI ENABLED AND 1 FOR NMI DISABLED ON EXIT
43 BITS 6-0 = ADDRESS OF TABLE LOCATION TO READ
44
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
55 0000 9C ; READ LOCATION (AL) INTO (AL)
56 0001 D0 C0 ; SAVE INTERRUPT_ENABLE STATUS AND FLAGS
57 0003 F9 ; MOVE NMI BIT TO LOW POSITION
58 0004 D0 D8 ; FORCE NMI BIT ON IN CARRY FLAG
59 0006 FA ; HIGH BIT ON TO DISABLE NMI - OLD IN CY
60 0007 E6 70 ; DISABLE INTERRUPTS
61 0008 0E ; ADDRESS LOCATION AND DISABLE NMI
62 000A E4 71 ; LD AL,DEAY
63 000C 50 ; READ THE REQUESTED CMOS LOCATION
64 000D B0 1A ; SAVE (AH) REGISTER VALUE AND CMOS BYTE
65 000F D0 D8 ; GET ADDRESS OF DEFAULT LOCATION
66 0011 E6 70 ; PUT ORIGINAL NMI MASK BIT INTO ADDRESS
67 0013 58 ; SET DEFAULT TO READ ONLY REGISTER
68 0014 0E ; RESTORE (AH) AND (AL)= CMOS_BYTE
69 0015 E8 0019 R ; LD AL,CMOS_REG_D*2
70 0018 C3 ; *HANDLE POFF FOR B- LEVEL 80286
71
72 0019 CMOS_READ ENDP
73
74 0019 CMOS_POFF IRET ; POPF FOR LEVEL B- PARTS
75 0019 CF ; RETURN FAR AND RESTORE FLAGS
76
77 001A CMOS_POFF ENDP
78
79 ----- CMOS_WRITE ----- WRITE BYTE TO CMOS SYSTEM CLOCK CONFIGURATION TABLE
80
81 INPUT: (AL)= CMOS TABLE ADDRESS TO BE WRITTEN TO
82 BIT 7 = 0 FOR NMI ENABLED AND 1 FOR NMI DISABLED ON EXIT
83 BITS 6-0 = ADDRESS OF TABLE LOCATION TO WRITE
84
85 (AH)= NEW VALUE TO BE PLACED IN THE ADDRESSED TABLE LOCATION
86
87 OUTPUT: VALUE IN (AH) PLACED IN LOCATION (AL) WITH NMI LEFT DISABLED
88 IF BIT 7 OF (AL) IS ON, DURING THE CMOS UPDATE BOTH NMI AND
89 NORMAL_INTERRUPTS ARE DISABLED TO PROTECT CMOS DATA INTEGRITY.
90 THE CMOS ADDRESS REGISTER IS POINTED TO A DEFAULT VALUE AND
91 THE INTERRUPT FLAG RESTORED TO THE ENTRY STATE ON RETURN.
92 ONLY THE CMOS LOCATION AND THE NMI STATE IS CHANGED.
93
94
95 001A CMOS_WRITE PROC NEAR
96 001A 9C ; WRITE (AH) TO LOCATION (AL)
97 001B 50 ; SAVE INTERRUPT_ENABLE STATUS AND FLAGS
98 001C D0 C0 ; SAVE WORK REGISTER VALUES
99 001E F9 ; MOVE NMI BIT TO LOW POSITION
100 001F D0 D8 ; FORCE NMI BIT ON IN CARRY FLAG
101 0021 FA ; HIGH BIT ON TO DISABLE NMI - OLD IN CY
102 0022 E6 70 ; DISABLE INTERRUPTS
103 0024 8A C4 ; ADDRESS LOCATION AND DISABLE NMI
104 0025 0E 11 ; LD AL,AR
105 0028 B0 1A ; GET THE DATA BYTE TO WRITE
106 002A D0 D8 ; PLACEMENT OF ADDRESS CMOS LOCATION
107 002C E6 70 ; GET ADDRESS OF DEFAULT LOCATION
108 002E 58 ; PUT ORIGINAL NMI MASK BIT INTO ADDRESS
109 002F 0E ; SET DEFAULT TO READ ONLY REGISTER
110 0030 E8 0019 R ; PUSH CS
111 0033 C3 ; RESTORE WORK REGISTERS
112
113 0034 CMOS_WRITE ENDP

```

```

114          PAGE
115          DDS  PROC  NEAR
116          0034 2E: 8E IE 003A R  MOV  DS,C5:DDSDATA
117          0039 C3  RET
118
119          003A ---- R  DDSDATA DW  DATA
120
121          003C  DDS  ENDP
122
123          ---- E_MSG -- P_MSG
124          THIS SUBROUTINE WILL PRINT A MESSAGE ON THE DISPLAY
125
126          ENTRY REQUIREMENTS:
127          SI = OFFSET(ADDRESS) OF MESSAGE BUFFER
128          CX = MESSAGE BYTE COUNT
129          MAXIMUM MESSAGE LENGTH IS 36 CHARACTERS
130          BP = BIT 0=E161/E162, BIT 1=CONFIG_BAD, 2-15= FIRST MSG OFFSET
131
132          003C  E_MSG  PROC  NEAR
133          003C F7 C5 3FFF  TEST  BP,03FFFH
134          0040 75 08  JNZ  E_MSGI
135
136          0042 56  PUSH  SI
137          0043 81 E6 3FFF  AND   SI,03FFFH
138          0047 00 EE  OR    BP,SI
139          0049 5E  POP   SI
140
141          004A  E_MSGI: CALL  P_MSG
142          004B E5 0063 R  PUSH  DS
143          0050 1E  CALL  DS
144          0054 E8 0034 R  CALL  DS
145          0051 F6 06 0010 R 01  TEST  BYTE PTR EQUIP_FLAG,01H
146          0056 74 02  JZ   MFG_HALT
147
148          0058 1F  POP   DS
149          0059 C3  RET
150
151          005A  MFG_HALT: CALL  DS
152          005A FA  CLI
153          005B A0 0015 R  MOV   AL,0MFG_ERR_FLAG
154          005E E6 80  OUT   MFG_PORT,AL
155          0060 F4  HLT
156          0061 EB F7  JMP   MFG_HALT
157
158          0063  E_MSG  ENDP
159
160
161          0063 2E: 8A 04  P_MSG  PROC  NEAR
162          0066 46  MOV   AL,CS:[SI]
163          0067 50  INC   SI
164          0068 E8 0128 R  PUSH  AX
165          0069 58  CALL  PRT_HEX
166          006A E5 0040  POP   AX
167          006C 5C 0A  CMP   AL,LF
168          006E 75 F3  JNE   P_MSG
169          0070 C3  RET
170
171          0071  P_MSG  ENDP
172
173          ---- ERR_BEEP
174          THIS PROCEDURE WILL ISSUE LONG TONES (1-3/4 SECONDS) AND ONE OR MORE SHORT TONES (9/32 SECOND) TO INDICATE A FAILURE ON THE PLANAR BOARD, A BAD MEMORY MODULE, OR A PROBLEM WITH THE CRT.
175
176          ENTRY PARAMETERS:
177          DH = NUMBER OF LONG TONES TO BEEP.
178          DL = NUMBER OF SHORT TONES TO BEEP.
179
180
181          0071  ERR_BEEP  PROC  NEAR
182          0071 9C  PUSHF
183          0072 FA  CLI
184          0073 0A F6  OR   DH,DH
185          0075 74 1E  JZ   G3
186          0077  G1: CALL  DS
187          0077 B3 70  MOV   BL,112
188          0079 B9 0500  MOV   CX,1280
189          007C 00AF R  CALL  BEEP
190          007D E8 C233  MOV   CX,49715
191          0082 E8 0075 R  CALL  WAITF
192          0085 FE CE  DEC   DH
193          0087 75 EE  JNZ   G1
194
195          0089 1E  PUSH  DS
196          008A E5 0034 R  CALL  DS
197          008D 80 3E 0012 R 01  CMP   #MFG_TST,01H
198          0092 1F  POP   DS
199          0093 74 C5  JE   MFG_HALT
200
201          G3: CALL  DS
202          0095 12  MOV   BL,18
203          0097 B9 04B8  MOV   CX,1208
204          009A E8 00AF R  CALL  BEEP
205          009D B9 8178  MOV   CX,33144
206          00A0 E8 00F5 R  CALL  WAITF
207          00A3 FE CA  DEC   DL
208          00A5 75 EE  JNZ   G3
209
210          00A7 B9 8178  MOV   CX,33144
211          00A8 E8 00F5 R  CALL  WAITF
212          00AD 9D  POPF
213          00AE C3  RET
214
215          00AF  ERR_BEEP  ENDP

```

```

217
218
219
220
221
222
223
224
225
226
227 00AF      BEEP    PROC  NEAR
228 0000 9C      PUSHF
229 0000 00      CL
230 00B1 B0 B6      MOV    AL,10101010B
231 00B3 E6 43      OUT   TIMER+3,AL
232 00B5 EB 00      JMP   $+2
233 00B7 8A C1      MOV    AL,CL
234 00B9 E6 42      OUT   TIMER+2,AL
235 00B8 EB 00      JMP   $+2
236 00B9 E6 55      MOV    AL,CH
237 00BF E6 42      OUT   TIMER+2,AL
238 00C1 E4 61      IN    AL,PORT_B
239 00C3 8A E0      MOV    AH,AL
240 00C5 0C 03      OR    AL,GATE2+SPK2
241 00C9 E6 61      OUT   PORT_B,AL
242 00C9 9D      POPF
243 00CA      G7:
244 00CA B9 040B      MOV    CX,1035
245 00CE E8 00F5 R    CALL  WAITF
246 00D0 FE CB      DEC   BL
247 00D2 75 F6      JNZ   G7
248
249 00D4 9C      PUSHF
250 00D5 FA      CL
251 00D6 E4 61      IN    AL,PORT_B
252 00D6 0C FC      OR    AL,NOT GATE2+SPK2
253 00D8 22 E0      AND   AH,AL
254 00D9 E4 44      MOV   AL,AH
255 00D8 24 FC      AND   AL,NOT (GATE2+SPK2)
256 00E0 E6 61      OUT   PORT_B,AL
257 00E2 9D      POPF
258 00E3 B9 040B      MOV    CX,1035
259 00E6 E8 00F5 R    CALL  WAITF
260 00E8 00 00      PUSHF
261 00EA FA      CL
262 00EB E4 61      IN    AL,PORT_B
263 00ED 24 03      AND   AL,GATE2+SPK2
264 00EF 0A C4      OR    AL,AH
265 00F1 E6 61      OUT   PORT_B,AL
266 00F3 9D      POPF
267 00F4 C3      RET
268
269 00F5      BEEP    ENDP
270
271
272
273
274
275
276
277
278
279
280
281
282 00F5      WAITF   PROC  NEAR
283 00F5 50      PUSH  AX
284
285 00F6      WAITFI  IN    AL,PORT_B
286 00F6 E4 61      AND   AL,REFRESH_BIT
287 00F8 24 10      CMP   AL,AH
288 00FA 3A C4      JE    WAITFI
289 00FC T4 F8      LOOP
290
291 00FE 8A E0      MOV   AH,AL
292 0100 E2 F4      LOOP
293
294 0102 58      POP   AX
295 0103 C3      RET
296
297 0104      WAITF   ENDP
298
299
300
301
302
303
304 0104      CONFIG_BAD PROC  NEAR
305 0104 50      PUSHF
306 0105 B8 8E8E      MOV   AX,X*(CMOS_DIAG+NMI)
307 0108 E8 0000 R    CALL  CMOS_READ
308 0108 0C 20      OR    AL,BAD_CONFIG
309 010D E8 00 20      XCHG AH,AL
310 010F E8 001A R    CALL  CMOS_WRITE
311 0112 58      POP   AX
312 0113 81 CD 4000      OR    BP,04000H
313 0117 C3      RET
314
315 0118      CONFIG_BAD ENDP

```

```

316          PAGE
317          ;--- XPC_BYTE -- XLATE_PR -- PRT_HEX -----
318          ;----- CONVENT AND PRINT ASCII CODE CHARACTERS -----
319          ;----- AL CONTAINS NUMBER TO BE CONVERTED.
320          ;----- AX AND BX DESTROYED.
321          ;----- -----
322
323          XPC_BYTE PROC NEAR
324          ;----- DISPLAY TWO HEX DIGITS
325          0118 00 0118 50
326          PUSH AX          ; SAVE FOR LOW NIBBLE DISPLAY
327          0119 C0 E8 04
328          SHR AL,4          ; NIBBLE SWAP
329          CALL XLAT_PR        ; DO THE HIGH NIBBLE DISPLAY
330          011F 58
331          POP AX          ; RECOVER THE NIBBLE
332          AND AL,0FH        ; ISOLATE TO LOW NIBBLE
333          ;----- FALL INTO LOW NIBBLE CONVERSION
334          0122 00 0122 90
335          ADD AL,090H        ; CONVERT 00-0F TO ASCII CHARACTER
336          0124 27
337          DAA
338          0125 14 40
339          ADC AL,040H        ; ADD FIRST CONVERSION FACTOR
340          0127 27
341          DAA
342          012C CD 10
343          ;----- ADD JUST FOR NUMERIC AND ALPHA RANGE
344          012E C3
345          RET
346
347          PRT_HEX PROC NEAR
348          012B 00 012B 0E
349          MOV AH,0EH        ; DISPLAY CHARACTER IN (AL) COMMAND
350          012A 00 012A 00
351          MOV BH,0
352          012C CD 10
353          INT 10H
354          ;----- CALL VIDEO_IO
355          RET
356
357          XLAT_PR ENDP
358          XPC_BYTE ENDP
359
360          ;--- PRT SEG -----
361          ;----- PRINT A SEGMENT VALUE TO LOOK LIKE A 21 BIT ADDRESS
362          ;----- DX MUST CONTAIN SEGMENT VALUE TO BE PRINTED
363
364          PRT_SEG PROC NEAR
365          013F 00 013F 8A C6
366          MOV AL,DX          ; GET MSB
367          CALL XPC_BYTE        ; DISPLAY SEGMENT HIGH BYTE
368          0131 E6 0118 R
369          MOV AL,DI          ; LSB
370          CALL XPC_BYTE        ; DISPLAY SEGMENT LOW BYTE
371          0134 8A C2
372          MOV AL,TO          ; PRINT A '0'
373          CALL PRT_HEX        ; TO MAKE LOOK LIKE ADDRESS
374          0136 E8 0118 R
375          MOV AL,00
376          CALL PRT_HEX        ; ADD ENDING SPACE
377          0139 B0 30
378          RET
379
380          PRT_SEG ENDP
381
382          ;--- PROT_PRT_HEX -----
383          ;----- PUT A CHARACTER TO THE DISPLAY BUFFERS WHEN IN PROTECTED MODE
384          ;----- (AL)= ASCII CHARACTER
385          ;----- (DI)= DISPLAY REGEN BUFFER POSITION
386
387          PROT_PRT_HEX PROC NEAR
388          0144 00 0144 06
389          PUSH ES          ; SAVE CURRENT SEGMENT REGISTERS
390          0145 57
391          PUSH DI          ; MULTIPLY OFFSET BY TWO
392          0146 DI E7
393
394          ;----- MONOCHROME VIDEO CARD
395          0144 07 0144 06
396          PUSH BYTE PTR C_BWCRT_PTR ; GET MONOCHROME BUFFER SEGMENT SELECTOR
397          0145 57
398          POP ES          ; SET (ES) TO B/W DISPLAY BUFFER
399          0146 DI E7
400          STOSB
401          014C 4F
402          DEC DI
403
404          ;----- ENHANCED GRAPHICS ADAPTER
405          0144 00 0144 07
406          PUSH BYTE PTR E_CCRT_PTR ; ENHANCED COLOR DISPLAY POINTER LOW 64K
407          0145 57
408          POP ES          ; LOAD SEGMENT SELECTOR
409          0146 DI 38
410          STOSB
411          DEC DI
412          PUSH ES          ; PLACE CHARACTER IN BUFFER
413          0145 57
414          STOSB
415          DEC DI
416          ;----- COMPATIBLE COLOR
417          0144 00 0144 07
418          PUSH BYTE PTR C_CCRT_PTR ; SET (DS) TO COMPATIBLE COLOR MEMORY
419          0145 57
420          POP ES          ; SAVE WORK REGISTERS
421          0146 DI 38
422          STOSB
423
424          PROT_S: IN AL,DX          ; GET COLOR CARD STATUS
425          TEST AL,1FVRT+RHRZ ; CHECK FOR VERTICAL RETRACE (OR HZRZ)
426          JZ PROT_S
427          LOOPZ PROT_S        ; TIMEOUT LOOP TILL FOUND
428          XCHG AX,BX          ; RECOVER CHARACTERS
429          STOSB
430
431          ;----- -----
432          POP CX
433          POP DX
434          POP BX
435          POP DI
436          POP ES
437          RET
438
439          PROT_PRT_HEX ENDP

```

```

423
424
425
426
427
428 0170 ROM_CHECKSUM PROC NEAR
429 0170 2B C9 SUB CX,CX ; NUMBER OF BYTES TO ADD IS 64K
430
431 0172 ROM_CHECKSUM_CNT: XOR AL,AL ; ENTRY FOR OPTIONAL ROM TEST
432 0172 32 C0
433 0174
434 0174 02 07 ADD AL,[BX] ; GET (DS:BX)
435 0176 43 INC BX ; POINT TO NEXT BYTE
436 0177 E2 FB LOOP ROM_L ; ADD ALL BYTES IN ROM MODULE
437
438 0179 0A C0 OR AL,AL ; SUM = 0?
439 017B C3 RET
440
441 017C ROM_CHECKSUM ENDP
442
443
444
445
446
447
448 017C
449 017C B8 ---- R ROM_CHECK PROC NEAR
450 017F BE C0 MOV AX,DATA ; POINT ES TO DATA AREA
451 0181 2A E4 MOV ES,AX
452 0183 BA 47 02 SUB AH,AH ; ZERO OUT AH
453 0186 00 09 MOV AL,[BX+2] ; GET LENGTH INDICATOR
454 0189 BB E8 SHL AX,9 ; MULTIPLY BY 512
455 018B C1 E8 04 MOV CX,AX ; SET COUNT
456 018E 03 D0 ADD DX,AX ; SET_POINTER TO NEXT MODULE
457 0190 E8 0172 R CALL ROM_CHECKSUM_CNT ; DO CHECKSUM
458 0193 T4 05 JZ ROM_CHECK_I
459
460 0195 E8 0000 E CALL ROM_ERR ; POST_CHECKSUM_ERROR
461 0198 EB 13 JMP SHORT ROM_CHECK_END ; AND EXIT
462
463 019A ROM_CHECK_I: PUSH DX ; SAVE POINTER
464 019A 52 PUSH ES:IO_ROM_INIT,0003H ; LOAD OFFSET
465 019B 26: C7 06 0067 R 0003 MOV DX,ES:IO_ROM_INIT ; LOAD SEGMENT
466 019C 26: 8C IE 0069 R CALL WORD PTR ES:IO_ROM_INIT ; CALL INITIALIZE/TEST ROUTINE
467 01A1 5A POP DX
468 01AC 5A
469
470 01AD ROM_CHECK_END: RET ; RETURN TO CALLER
471
472 01AD C3
473 01AE ROM_CHECK ENDP
474
475
476
477
478
479
480
481 01AE KBD_RESET PROC NEAR
482 01AE B0 FF MOV AL,0FDH ; SET KEYBOARD RESET COMMAND
483 01B0 E8 0000 E CALL XMIT_8042 ; GO ISSUE THE COMMAND
484 01B3 E3 23 JCXZ G13 ; EXIT IF ERROR
485
486 01B5 3C FA CMP AL,KB_ACK
487 01B7 75 1F JNZ G13
488
489 01B9 B0 FD MOV AL,0FDH ; ENABLE KEYBOARD INTERRUPTS
490 01C0 E4 21 OUT AL,INTA01,AL ; WRITE 8040 INTERRUPT MASK REGISTER
491 01B0 C6 06 006B R 00 MOV @INTR_FLAG,0 ; RESET INTERRUPT INDICATOR
492 01C2 FB STI ; ENABLE INTERRUPTS
493 01C3 B3 0A MOV BL,10 ; TRY FOR 400 MILLISECONDS
494 01C5 2B C9 SUB CX,CX ; SETUP INTERRUPT TIMEOUT COUNT
495 01C7
496 01C8 F6 06 006B R 02 G11: TEST @INTR_FLAG,02H ; DID A KEYBOARD INTERRUPT OCCUR ?
497 01C9 75 06 JNZ G12 ; YES - READ SCAN CODE RETURNED
498 01CE E2 F7 LOOP G11 ; NO - LOOP TILL TIMEOUT
499
500 01D0 FE CB DEC BL ; TRY AGAIN
501 01D2 75 F3 JNZ G11
502 01D4 E4 60 G12: IN AL,PORT_A ; READ KEYBOARD SCAN CODE
503 01D4 E4 60 MOV BL,AL ; SAVE SCAN CODE JUST READ
504 01D6 8A D8
505 01D8
506 01D8 C3 RET ; RETURN TO CALLER
507
508 01D9 KBD_RESET ENDP
509
510
511
512
513
514
515 01D9 BLINK_INT PROC NEAR
516 01D9 FB STI ; SAVE AX REGISTER CONTENTS
517 01DA 50 PUSH AX ; READ CURRENT VALUE OF MFG_PORT
518 01DB E4 80 IN AL,MFG_PORT ; FLIP CONTROL BIT
519 01DD 34 40 XOR AL,01000000B
520 01E0 00 00 OUT MFG_PORT,AL
521 01E1 B0 20 MOV AL,E2H
522 01E3 E6 20 OUT INTA00,AL
523 01E5 58 POP AX ; RESTORE AX REGISTER
524 01E6 CF IRET
525
526 01E7 BLINK_INT ENDP

```

```

527
528
529
530
531
532
533
534
535
536
537
538
539
540
541 = 0012
542 = 0444
543 = 0007
544 = 0080
545
546
547 01E7
548 01E7 60
549 01E8 1E
550 01E9 E8 0034 R
551 01EC 2B C0
552 01EE A2 0070 R
553 01F1 A3 006C R
554 01F4 A3 006E R
555 01F5 E8 0000 R
556 01F9 E8 0000 R
557 01FC 24 C4
558 01FE B2 C9
559 0200 75 64
560
561 0202 B0 8A
562 0204 E8 0000 R
563 0207 A8 80
564 0209 E1 F7
565
566 0208 E3 59
567 0200
568 0200 B0 8A
569 0200 E8 0000 R
570 0212 A8 80
571 0214 E0 F7
572
573 0216 E3 4E
574
575 0218 B0 80
576 021A E8 0000 R
577 021D 3C 59
578 021F 77 48
579
580 0221 E8 027F R
581 0224 BB C8
582 0226 C1 E9 02
583 0229 B3 12
584 022B F6 E8
585 022C F6 68
586 022F B0 82
587 0231 E8 0000 R
588 0234 3C 59
589 0236 77 31
590 0238 E8 027F R
591 0239 E8 0000 R
592 023C D1 E8
593 023E 03 C8
594 0240 58
595 0241 BB 0444
596 0244 F0 E3
597 0245 03 C8
598 0248 B0 84
599 024A E8 0000 R
600 024D 3C 23
601 024F 77 18
602
603 0251 E8 027F R
604 0254 BB D0
605 0256 B3 07
606 0258 F6 E3
607 025A 03 C1
608 025C B3 D2 00
609 025E 69 16 006E R
610 0263 A3 006C R
611 0266
612 0266 1F
613 0267 61
614 0268 C3
615
616 0269
617 0269 1F
618 026A 61
619 026B BE 0000 E
620 026E E8 003C R
621 0271 BB BE80
622 0274 03 C0 R
623 0277 0C 04
624 0279 B6 C4
625 027B E8 001A R
626 027E C3
627
628 027F
629
630 027F
631 027F 8A E0
632 0281 C0 EC 04
633 0284 24 0F
634 0286 D5 0A
635 0288 C3
636
637 0289

```

PAGE

```

;----- ROUTINE INITIALIZES THE TIMER DATA AREA IN THE ROM BIOS
;----- DATA AREA. IT IS CALLED BY THE POWER ON ROUTINES. IT CONVERTS
;----- HR:MIN:SEC FROM CMOS TO TIMER TICS. IF CMOS IS INVALID, TIMER
;----- IS SET TO ZERO.

;----- INPUT  NONE PASSED TO ROUTINE BY CALLER
;----- CMOS LOCATIONS USED FOR TIME

;----- OUTPUT  #TIMER_LOW
;----- #TIMER_HIGH
;----- #TIMER_OFN
;----- ALL REGISTERS UNCHANGED

;----- COUNTS_SEC EQU 16 ; TIMER DATA CONVERSION EQUATES
;----- COUNTS_MIN EQU 1092 ; COUNTS = 65536 - 55536
;----- COUNTS_HOUR EQU 7 ; RTC UPDATE IN PROCESS BIT MASK
;----- UPDATE_TIMER EQU 10000000B ; RTC UPDATE IN PROCESS BIT MASK

SET_TOD PROC NEAR
    PUSH DS
    CALL DDS
    SUB AX,AX
    MOV #TIMER_OFN,AL ; ESTABLISH SEGMENT
    MOV #TIMER_LOW,AX ; RESET TIMER ROLL OVER INDICATOR
    MOV #TIMER_HIGH,AX ; AND TIMER COUNT
    MOV AL,CMOS_REG_A+NMI ; CHECK CMOS VALIDITY
    CMOS_READ ; READ DIAGNOSTIC LOCATION IN CMOS
    AND AL,BAD_BAT-BAD_CKSUM+CMOS_CLK_FAIL ; BAD BATTERY, CKSUM ERROR, CLOCK ERROR
    SUB CX,CX ; CMOS NOT VALID -- TIMER SET TO ZERO
    JNZ POD_DONE

    UIP:
    MOV AL,CMOS_REG_A+NMI ; ACCESS REGISTER A
    CMOS_READ ; READ CMOS CLOCK REGISTER A
    TEST AL,UPDATE_TIMER ; WAIT TILL UPDATE BIT IS ON
    LOOPZ UIP

    UIPOFF:
    JCXZ POD_DONE ; CMOS CLOCK STUCK IF TIMEOUT
    MOV AL,CMOS_REG_A+NMI ; ACCESS REGISTER A
    CMOS_READ ; READ CMOS CLOCK REGISTER A
    TEST AL,UPDATE_TIMER ; NEXT WAIT TILL END OF UPDATE
    LOOPZ UIPOFF

    JCXZ POD_DONE ; CMOS CLOCK STUCK IF TIMEOUT
    MOV AL,CMOS_SECONDS+NMI ; TIME JUST UPDATED
    CMOS_READ ; ACCESS SECONDS VALUE IN CMOS
    CMP AL,59H ; ARE THE SECONDS WITHIN LIMITS?
    JA TOD_ERROR ; GO IF NOT

    CALL CVT_BINARY ; CONVERT IT TO BINARY
    MOV CX,AX ; MOVE COUNT TO ACCUMULATION REGISTER
    SHR CX,2 ; ADJUST FOR SYSTEMATIC SECONDS ERROR
    MOV BL,COUNTS_SEC ; COUNT FOR SECONDS
    MUL BL,CX ; ADD CX,AX
    ADD BL,CX ; ADD CX,AX
    MOV AL,CMOS_MINUTES+NMI ; ACCESS MINUTES VALUE IN CMOS
    CMOS_READ ; ARE THE MINUTES WITHIN LIMITS?
    CMP AL,59H ; GO IF NOT
    JA TOD_ERROR ; CONVERT IT TO BINARY
    PUSH AX ; SAVE SECONDS COUNT
    CALL CVT_BINARY ; ADJUST FOR SYSTEMATIC MINUTES ERROR
    ADD AX,1 ; ADD ADJUSTMENT TO COUNT
    ADD AX,AX ; RECOVER BCD MINUTES VALUE
    POP AX ; COUNT FOR MINUTES
    MUL BX,COUNTS_MIN ; ADD TO ACCUMULATED VALUE
    ADD CX,AX ; COUNT FOR HOURS
    MUL BL,COUNTS_HOUR ; COUNT FOR HOURS
    ADD AX,CX ; COUNT FOR HOURS
    ADC DX,0000H ; ACCESS HOURS VALUE IN CMOS
    CMOS_READ ; ARE THE HOURS WITHIN LIMITS?
    CMP AL,23H ; GO IF NOT
    JA TOD_ERROR ; SET_CLOCK_ERROR IN STATUS
    CALL E_MSG ; SET_CLOCK_ERROR IN STATUS
    MOV AX,X1(CMOS_DIAG+NMI) ; SET_CLOCK_ERROR IN STATUS
    CMOS_READ ; SET_CLOCK_ERROR IN STATUS
    OR AL,CMOS_CLK_FAIL ; SET_CLOCK_ERROR IN STATUS
    MOV AL,AH ; SET_CLOCK_ERROR IN STATUS
    XCHG AL,AH ; SET_CLOCK_ERROR IN STATUS
    CALL CMOS_WRITE ; MOVE NEW STATUS TO WORK REGISTER
    RET ; UPDATE STATUS LOCATION

    POD_DONE:
    POP DS
    POPA
    RET

    TOD_ERROR:
    POP DS ; RESTORE SEGMENT
    POPA ; RESTORE REGISTERS
    SI,OFFSET E163 ; DISPLAY CLOCK ERROR
    CALL E_MSG ; DISPLAY CLOCK ERROR

    SET_CLOCK_ERROR IN STATUS
    CMOS_READ ; SET_CLOCK_ERROR IN STATUS
    OR AL,CMOS_CLK_FAIL ; SET_CLOCK_ERROR IN STATUS
    MOV AL,AH ; SET_CLOCK_ERROR IN STATUS
    XCHG AL,AH ; SET_CLOCK_ERROR IN STATUS
    CALL CMOS_WRITE ; MOVE NEW STATUS TO WORK REGISTER
    RET ; UPDATE STATUS LOCATION

    SET_TOD ENDP

    CVT_BINARY PROC NEAR
        MOV AH,AL ; UNPACK 2 BCD DIGITS IN AL
        SHR AH,4 ; RESULT IS IN AX
        AND AL,0FH ; CONVERT UNPACKED BCD TO BINARY
        RET
    ENDP

```

```

638
639
640
641
642
643
644
645
646
647
648
649 0289
650 0289 50
651 028A 53
652 028B B0 0B
653 028D E6 20
654 028F EB 00
655 0291 E4 20
656 0293 8A E0
657 0295 0A C4
658 0297 75 04
659
660 0299 B4 FF
661 029B EB 2F
662 029D 00 00
663 02A0 B0 0B
664 029F E6 A0
665 02A1 EB 00
666 02A3 E4 A0
667 02A5 8A F8
668 02A7 0A FF
669 02A9 74 10
670
671 02AB E4 A1
672 02AD 0A C7
673 02AF EB 00
674 02B1 E6 A1
675 02B3 80 20
676 02B5 EB 00
677 02B7 E6 A0
678 02B9 EB 00
679 02BB
680 02BB E4 21
681 02CB EB 00
682 02BF 80 E4 FB
683 02C2 0A C4
684 02C4 E6 21
685 02C6 EB 00
686 02C8
687 02CA B0 20
688 02CA E6 20
689 02CC
690 02CC 5B
691 02CD 1E
692 02CE E8 0034 R
693 02D0 00 26 006B R
694 02D5 1F
695 02D6 58
696 02D7
697 02D7 CF
698
699 02D8
700
701
702
703
704
705
706
707 02D8
708 02D8 50
709 02D9 B0 20
710 02D8 E6 A0
711 02DD 58
712 02DE CD 0A
713
714 02E0 CF
715
716 02E1
717
718
719
720
721
722
723
724
725 02E1
726 02E1 50
727 02E2 32 C0
728 02E4 E6 F0
729
730 02E6 B0 20
731 02E8 E6 A0
732 02E8 E6 20
733 02EC 58
734 02ED CD 02
735
736 02EF CF
737
738 02F0
739
740 02F0
741
742 02F0 B0 FE
743 02F2 E6 64
744 02F4
745 02F4 F4
746 02F5 EB FD
747
748 02F7
749 02F7
750

PAGE
; D11 -- INT ?? H -- ( IRQ LEVEL ?? ) -----
; TEMPORARY INTERRUPT SERVICE ROUTINE FOR POST
;
; THIS ROUTINE IS ALSO LEFT IN PLACE AFTER THE POWER ON DIAGNOSTICS
; TO SERVICE UNUSED INTERRUPT VECTORS. LOCATION "0INTR_FLAG" WILL
; CONTAIN EITHER:
; 1) "FF" FOR A HARDWARE INTERRUPT THAT CAUSED CODE TO BE EXECUTED, OR
; 2) "00" FOR A NON-HARDWARE INTERRUPT THAT WAS EXECUTED ACCIDENTALLY.
;

D11  PROC  NEAR
    PUSH  AX
    PUSH  BX
    MOV   AL,0BH
    OUT   INTA00,AL
    JMP   $+2
    IN    AL,INTA00
    MOV   AH,AL
    OR    AL,AH
    JNZ   HW_INT

    MOV   AH,0FFH
    PUSH  BH
    SHORT_SET_INTR_FLAG

    HW_INT:
    MOV   AL,0BH
    OUT   INTB00,AL
    JMP   $+2
    IN    AL,INTB00
    MOV   BH,AL
    OR    BH,BH
    JZ    NOT_SEC

    NOT_SEC:
    IN    AL,INTB01
    OR   AL,BH
    JMP  $+2
    OUT  INTB01,AL
    MOV  AL,EO1
    OUT  INTA01,AL
    JMP  $+2
    OUT  INTB00,AL
    JMP  SHORT_IS_SEC

    IS_SEC:
    IN    AL,INTA01
    AND  AH,0FBH
    JMP  $+2
    OUT  INTA01,AL
    JMP  $+2

    SET_INTR_FLAG:
    POP   BX
    PUSH  DS
    CALL  DDS
    MOV   $0INTR_FLAG,AH
    POP   DS
    POP   AX
    RET

    DUMMY_RETURN_IRET:
    IRET

D11  ENDP

;---- HARDWARE INT 71 H -- ( IRQ LEVEL 9 ) --> TO INT 0A H -----
;REDIRECT SLAVE INTERRUPT 9 TO INTERRUPT LEVEL 2
;THIS ROUTINE FIELDS LEVEL 9 INTERRUPTS AND
;CONTROL IS PASSED TO MASTER INTERRUPT LEVEL 2
;

RE_DIRECT PROC  NEAR
    PUSH  AX
    MOV   AL,EO1
    OUT   INTB00,AL
    POP   AX
    INT  0AH
    IRET

RE_DIRECT ENDP

;---- HARDWARE INT 75 H -- ( IRQ LEVEL 13 ) -----
;SERVICE X287 INTERRUPTS
;THIS ROUTINE FIELDS X287 INTERRUPTS AND CONTROL
;IS PASSED TO THE NMI INTERRUPT HANDLER FOR
;COMPATIBILITY.
;

INT_287 PROC  NEAR
    PUSH  AX
    XOR   AL,AL
    OUT   X287,AL
    IRET

INT_287 ENDP

;---- HARDWARE INT 75 H -- ( IRQ LEVEL 13 ) -----
;ENABLE THE INTERRUPT
;THE SLAVE
;THE MASTER
;RESTORE (AX)
;GIVE CONTROL TO NMI
;RETURN
;

PROC_SHUTDOWN PROC
    MOV   AL,SHUT_CMD
    OUT   STATUS_PORT,AL
    HLT
    JMP   PROC_S

PROC_S:
    HLT
    PROC_S

PROC_SHUTDOWN ENDP
CODE_SHUTDOWN ENDS
END

```

```
1 PAGE 118,121
2 TITLE TESTS ---- 06/10/85 EXCEPTION INTERRUPT TEST HANDLERS
3 .286C
4 .LIST
5 0000 CODE SEGMENT BYTE PUBLIC
6
7 PUBLIC POSTS
8 PUBLIC SYSINIT1
9
10
11 ;-----+
12 ;-----+ EXCEPTION INTERRUPT ROUTINE :-----+
13 ;-----+
14
15 0000 ASSUME CS:CODE,DS:AB50
16 0000
17 0000 B0 90 MOV AL,90H ; <>> SET CHECKPOINT <><>
18 0002 E9 00B2 R JMP TEST_EXC ; GO TEST IF EXCEPTION WAS EXPECTED
19 0005
20 0005 B0 91 MOV AL,91H ; <>> SET CHECKPOINT <><>
21 0007 E9 00B2 R JMP TEST_EXC ; GO TEST IF EXCEPTION WAS EXPECTED
22 000A B0 92 MOV AL,92H ; <>> SET CHECKPOINT <><>
23 000C E9 00B2 R JMP TEST_EXC ; GO TEST IF EXCEPTION WAS EXPECTED
24 000F B0 93 MOV AL,93H ; <>> SET CHECKPOINT <><>
25 0011 E9 00B2 R JMP TEST_EXC ; GO TEST IF EXCEPTION WAS EXPECTED
26 0014 B0 94 MOV AL,94H ; <>> SET CHECKPOINT <><>
27 0016 E9 00B2 R JMP TEST_EXC ; GO TEST IF EXCEPTION WAS EXPECTED
28 0019
29 0014 B0 94 PUSH ES ; LOAD ES REGISTER WITH SELECTOR
30 0016 E9 00B2 R PUSH BYTE PTR ES_TEMP
31 0019 POP ES
32 001A 6A 48
33 001C 07
34
35
36 ;-----+ FIX BOUND PARAMETERS
37
38 001D 2B FF SUB DI,DI ; POINT BEGINNING OF THE BLOCK
39 001F 26 1C CT 05 0000 MOV WORD PTR ES:[DI],0 ; SET FIRST WORD TO ZERO
40 0020 26 1C CT 05 02 7FFF MOV WORD PTR ES:[DI+2],07FFH ; SET SECOND TO 07FFH
41 0024
42 0028 B0 95 MOV AL,95H ; <>> SET CHECKPOINT <><>
43 002D E9 00B2 R JMP TEST_EXC ; GO TEST IF EXCEPTION WAS EXPECTED
44
45 0030
46 0032 B0 96 MOV AL,96H ; <>> SET CHECKPOINT <><>
47 0032 EB 7E JMP SHORT TEST_EXC ; GO TEST IF EXCEPTION WAS EXPECTED
48 0034
49 0034 B0 97 MOV AL,97H ; <>> SET CHECKPOINT <><>
50 0036 EB 7A JMP SHORT TEST_EXC ; GO TEST IF EXCEPTION WAS EXPECTED
51 0038
52 0038 B0 98 MOV AL,98H ; <>> SET CHECKPOINT <><>
53 003A EB 76 JMP SHORT TEST_EXC ; GO TEST IF EXCEPTION WAS EXPECTED
54 003C
55 003C B0 99 MOV AL,99H ; <>> SET CHECKPOINT <><>
56 003E EB 72 JMP SHORT TEST_EXC ; GO TEST IF EXCEPTION WAS EXPECTED
57 0040
58 0042 B0 9A MOV AL,9AH ; <>> SET CHECKPOINT <><>
59 0042 EB 6E JMP SHORT TEST_EXC ; GO TEST IF EXCEPTION WAS EXPECTED
60 0044
61 0044 B0 9B MOV AL,9BH ; <>> SET CHECKPOINT <><>
62 0046 EB 6A JMP SHORT TEST_EXC ; GO TEST IF EXCEPTION WAS EXPECTED
63 0048
64 0048 B0 9C MOV AL,9CH ; <>> SET CHECKPOINT <><>
65 004A EB 66 JMP SHORT TEST_EXC ; GO TEST IF EXCEPTION WAS EXPECTED
66 004C
67 004C B0 9D MOV AL,9DH ; <>> SET CHECKPOINT <><>
68 004E EB 62 JMP SHORT TEST_EXC ; GO TEST IF EXCEPTION WAS EXPECTED
69 0050
70 0052 B0 9E MOV AL,9EH ; <>> SET CHECKPOINT <><>
71 0052 EB 5E JMP SHORT TEST_EXC ; GO TEST IF EXCEPTION WAS EXPECTED
72 0054
73 0054 B0 9F MOV AL,9FH ; <>> SET CHECKPOINT <><>
74 0056 EB 5A JMP SHORT TEST_EXC ; GO TEST IF EXCEPTION WAS EXPECTED
75 0058
76 0058 B0 A0 MOV AL,0A0H ; <>> SET CHECKPOINT <><>
77 005C EB 56 JMP SHORT TEST_EXC ; GO TEST IF EXCEPTION WAS EXPECTED
78 005C
79 005C B0 A1 MOV AL,0A1H ; <>> SET CHECKPOINT <><>
80 005E EB 52 JMP SHORT TEST_EXC ; GO TEST IF EXCEPTION WAS EXPECTED
81 0060
82 0060 B0 A2 MOV AL,0A2H ; <>> SET CHECKPOINT <><>
83 0062 EB 4E JMP SHORT TEST_EXC ; GO TEST IF EXCEPTION WAS EXPECTED
84 0064
85 0064 B0 A3 MOV AL,0A3H ; <>> SET CHECKPOINT <><>
86 0066 EB 4A JMP SHORT TEST_EXC ; GO TEST IF EXCEPTION WAS EXPECTED
87 0068
88 0068 B0 A4 MOV AL,0A4H ; <>> SET CHECKPOINT <><>
89 0068 EB 46 JMP SHORT TEST_EXC ; GO TEST IF EXCEPTION WAS EXPECTED
90 006C
91 006C B0 A5 MOV AL,0A5H ; <>> SET CHECKPOINT <><>
92 006E EB 42 JMP SHORT TEST_EXC ; GO TEST IF EXCEPTION WAS EXPECTED
93 0070
94 0070 B0 A6 MOV AL,0A6H ; <>> SET CHECKPOINT <><>
95 0072 EB 3E JMP SHORT TEST_EXC ; GO TEST IF EXCEPTION WAS EXPECTED
96 0074
97 0074 B0 A7 MOV AL,0A7H ; <>> SET CHECKPOINT <><>
98 0076 EB 3A JMP SHORT TEST_EXC ; GO TEST IF EXCEPTION WAS EXPECTED
99 0078
100 0078 B0 A8 MOV AL,0A8H ; <>> SET CHECKPOINT <><>
101 007A EB 36 JMP SHORT TEST_EXC ; GO TEST IF EXCEPTION WAS EXPECTED
102 007C
103 007C B0 A9 MOV AL,0A9H ; <>> SET CHECKPOINT <><>
104 007E EB 32 JMP SHORT TEST_EXC ; GO TEST IF EXCEPTION WAS EXPECTED
105 0080
106 0080 B0 AA MOV AL,0AAH ; <>> SET CHECKPOINT <><>
107 0080 EB 2E JMP SHORT TEST_EXC ; GO TEST IF EXCEPTION WAS EXPECTED
108 0084
109 0084 B0 AB MOV AL,0ABH ; <>> SET CHECKPOINT <><>
110 0086 EB 2A JMP SHORT TEST_EXC ; GO TEST IF EXCEPTION WAS EXPECTED
111 0088
112 0088 B0 AC MOV AL,0ACH ; <>> SET CHECKPOINT <><>
113 008A EB 26 JMP SHORT TEST_EXC ; GO TEST IF EXCEPTION WAS EXPECTED
114 008C
```

```

115 008C B0 AD      MOV    AL,0ADH      ; <><> SET CHECKPOINT <><>
116 008E EB 22      JMP    SHORT TEST_EXC  ; GO TEST IF EXCEPTION WAS EXPECTED
117 0090             MOV    AL,0AEH      ; <><> SET CHECKPOINT <><>
118 0090 B0 AE      JMP    SHORT TEST_EXC  ; GO TEST IF EXCEPTION WAS EXPECTED
119 0092 EB 1E      MOV    AL,0AFH      ; <><> SET CHECKPOINT <><>
120 0094             JMP    SHORT TEST_EXC  ; GO TEST IF EXCEPTION WAS EXPECTED
121 0094 B0 AF      MOV    AL,0B0H      ; <><> SET CHECKPOINT <><>
122 0096 EB 1A      JMP    SHORT TEST_EXC  ; GO TEST IF INTERRUPT WAS EXPECTED
123 0098             SYS_32:      MOV    AL,0B0H      ; <><> SET CHECKPOINT <><>
124 0098 B0 B0      JMP    SHORT TEST_EXC  ; GO TEST IF INTERRUPT WAS EXPECTED
125 009A EB 16      MOV    AL,0B1H      ; <><> SET CHECKPOINT <><>
126 009C             SYS_33:      MOV    AL,0B1H      ; GO TEST IF INTERRUPT WAS EXPECTED
127 009C B0 B1      JMP    SHORT TEST_EXC  ; <><> SET CHECKPOINT <><>
128 009E EB 12      MOV    AL,0B2H      ; GO TEST IF INTERRUPT WAS EXPECTED
129 00A0             SYS_34:      MOV    AL,0B2H      ; <><> SET CHECKPOINT <><>
130 00A0 B0 B2      JMP    SHORT TEST_EXC  ; GO TEST IF INTERRUPT WAS EXPECTED
131 00A2 EB 0E      MOV    AL,0B2H      ; <><> SET CHECKPOINT <><>
132 00A4             SYS_35:      MOV    AL,0B2H      ; GO TEST IF INTERRUPT WAS EXPECTED
133 00A4 B0 B3      MOV    AL,0B3H      ; <><> SET CHECKPOINT <><>
134 00A6 EB 0A      JMP    SHORT TEST_EXC  ; GO TEST IF INTERRUPT WAS EXPECTED
135 00A8             SYS_36:      MOV    AL,0B4H      ; <><> SET CHECKPOINT <><>
136 00A8 B0 B4      JMP    SHORT TEST_EXC  ; GO TEST IF INTERRUPT WAS EXPECTED
137 00AA EB 06      MOV    AL,0B4H      ; <><> SET CHECKPOINT <><>
138 00AC             SYS_37:      MOV    AL,0B5H      ; GO TEST IF INTERRUPT WAS EXPECTED
139 00AC B0 B5      JMP    SHORT TEST_EXC  ; <><> SET CHECKPOINT <><>
140 00AE EB 02      MOV    AL,0B5H      ; GO TEST IF INTERRUPT WAS EXPECTED
141 00B0             SYS_38:      MOV    AL,0B6H      ; <><> SET CHECKPOINT <><>
142 00B0 B0 B6      JMP    SHORT TEST_EXC  ; GO TEST IF INTERRUPT WAS EXPECTED
143
144
145 00B2             TEST_EXC:   OUT   MFG_PORT,AL  ; OUTPUT THE CHECKPOINT
146 00B2 E6 80      CMP   AL,0AFH      ; CHECK FOR EXCEPTION
147 00B4 3C AF      JA    TEST_EXC0  ; GO IF A SYSTEM INTERRUPT
148
149
150 00B8 1E          PUSH  DS          ; SAVE THE CURRENT DATA SEGMENT
151 00B9 6A 08          PUSH  BYTE PTR GDT_PTR
152 00B8 1F          POP   DS          ; POP DS
153 00BC C7 06 0048 FFFF
154 00BC C6 06 004D 93
155 00C7 00 48
156 00C9 07
157 00CA 1F          PUSH  DS          ; RESTORE REGISTERS
158 00CB 5A          POP   DS          ; CHECK IF CODE SEGMENT SECOND ON STACK
159 00CC 59
160 00CD 31
161 00CE 83 F9 40
162 00D1 75 01
163
164 00D3 52          PUSH  DX          ; PUT SEGMENT BACK ON STACK
165 00D4
166 00E4 B6 E0          TEST_EXC0: XCHG  AH,AL  ; SAVE THE CHECKPOINT
167 00E4 E4 B8          IN    AL,DMA_PAGE+0AH
168 00D8 3A C4          CMP   AL,AH      ; WAS THE EXCEPTION EXPECTED?
169 00D4 74 0E          JZ    TEST_EXC3 ; GO IF YES
170 00DC
171 00DC E4 80          TEST_EXC1: IN    AL,MFG_PORT ; CHECK THE CURRENT CHECKPOINT
172 00DE 3C 3B          CMP   AL,03BH      ; HALT IF CHECKPOINT BELOW 3BH
173 00E0 72 01          JB    TEST_EXC2 ; TEST_EXC2
174 00E2 CF          IRET
175
176 00E3
177 00E3 B6 E0          TEST_EXC2: XCHG  AH,AL  ; OUTPUT THE CURRENT CHECKPOINT
178 00E5 E6 80          OUT   MFG_PORT,AL ; <><> CHECKPOINT 90 THRU B5 <><>
179 00E6 74 00          HLT
180 00E8 EB F9          JMP   TEST_EXC2 ; INSURE SYSTEM HALT
181
182 00EA
183 00EA 2A C0          TEST_EXC3: SUB   AL,AL  ; CLEAR DMA PAGE
184 00EC E6 BB          OUT   DMA_PAGE+0AH,AL ; FOR BOUND INSTRUCTION EXPECTED (INT 5)
185 00EB BB 0100
186 00F1 CF          IRET
187
188
189
190
191
192
193
194 00F2             SYSINITI PROC  NEAR
195 00F2 FA          CLI
196 00F3 55          PUSH  BP          ; NO INTERRUPTS ALLOWED
197 00F3 80 81          MOV   AL,01H      ; SAVE BP
198 00F4 E6 80          OUT   MFG_PORT,AL ; <><> <><> <><> <><> <><>
199 00F8 E8 0149 R          CALL  SIDT_BLD ; <><> CHECKPOINT 81 <><>
200 00FB BB EF          MOV   BP,DT      ; SAVE THE POINTER TO JUST PAST THE IDT
201
202
203
204 00FD BB 0800          MOV   AX,SYSLDT_LEN ; AS WE HAVE NO SDA, USE THE SIX BYTES
205 0100 AB          STOSW ; HERE TO LOAD THE IDTR. WE WILL SIDT
206 0101 BB D0A0          MOV   AX,SYSLDT_LOC ; WHEN WE GET TO SDA INITIALIZATION.
207 0104 AB          STOSW ; SET IDT LIMIT = LENGTH OF IDT
208 0105 BB 0000          MOV   AX,0
209 0108 AB          STOSW ; IDT ADDRESS
210
211 0109 26          +    MOV   ES
212
213 010A 0F          +    DB   026H      ; LOAD THE IDT
214 010B             +    LDIDT [BP] ; REGISTER FROM THIS AREA
215 010B BB 5E 00          ??0001 LABEL  BYTE
216 010E             +    ??0002 LABEL  BYTE
217 010B             +    ??0002 LABEL  BX,WORD PTR [BP]
218 010B             +    ORG   001H
219 010E             +    ORG   OFFSET CS:??0001
220 010E BB FD          MOV   DI,BP      ; ES:DI NOW --> END OF IDT AGAIN
221
222
223
224 0110 BF D8A0
225 0113 EB 0140 R          MOV   DI,GDT_LOC ; BUILD THE GDT.
226 0116 BB EF          CALL  GDT_BLD ; SAVE THE ES:DI POINTER
227 0118 BB 0088          MOV   BP,DI ; AX = LENGTH OF THE GDT
228 011B AB          STOSW ; PUT THAT IN THE LIMIT FIELD

```

```

229 011C B8 D8A0 MOV AX,GDT_LOC ; AX = LOW WORD OF GDT ADDRESS
230 011F AB STOSW ; PUT THAT IN BASE FIELD - LOW
231 0120 B8 0000 MOV AX,0 ; AX = HIGH BYTE OF ADDRESS, AND
232 0123 AB STOSW ; ACCESS RIGHTS BYTE IS UNDEFINED
233 0124 00 SEGOV ; LOAD THE GDTR
234 0124 26 + DD 026H ; FROM THIS AREA
235 0125 00 LGDT [BP]
236 0125 0F + DB 00FH
237 0126 + ??0004 LABEL BYTE
238 0126 B8 56 00 + MOV DX,WORD PTR [BP]
239 0126 00 ??0005 LABEL BYTE
240 0126 + ??0005 LABEL BYTE
241 0126 01 + DB 001FH
242 0129 + DB 001FH
243 0129 B8 FD MOV DI,BP ; RESTORE THE ES:DI POINTER
244 012B AB STOSW
245 012C AB STOSW
246 012D AB FD MOV DI,BP
247
248 ;----- SWITCH TO VIRTUAL MODE
249
250 012F 5D POP BP ; RESTORE BP
251 0130 B8 0001 MOV AX,VIRTUAL_ENABLE ; MACHINE STATUS WORD NEEDED TO
252 0131 00 LMSW AX ; SWITCH TO VIRTUAL MODE
253 0133 0F 01 F0 + DB 00FH,001H,0FOH
254
255 0136 EA DB 0EAH ; JUMP FAR TO PURGE PRE-FETCH QUEUE
256 0137 013B R DW OFFSET DONE ; TO OFFSET
257 0139 0040 DS,SYN_RS ; IN SEGMENT
258 013A 0000
259 013B B8 85 MOV AL,85H ; <><><><><><><><>
260 013D E6 80 OUT MFQ_PORT,AL ; <><> CHECKPOINT B2 <><>
261 013F C3 RET ; SYSTEM INITIALIZATION
262
263 0140 SYSINITI ENDP
264
265
266 0140 GDT_BLD PROC NEAR
267 0140 BE 01AF R MOV SI,OFFSET GDT_DATA_START ; DS:SI --> GDT
268 0143 B9 0044 MOV CX,(OFFSET GDT_DATA_END-OFFSET GDT_DATA_START)/2 ; WORD COUNT
269 0146 F3/ A5 REP MOVSW ; COPY GDT INTO MEMORY
270 0148 C3 RET
271 0149 GDT_BLD ENDP
272
273
274 0149 SIDT_BLD PROC NEAR
275
276 ;----- BUILD THE IDT. THE IDT WILL CONTAIN VECTORS FOR EXCEPTION HANDLERS
277
278 0149 BE 0237 R MOV SI,OFFSET SYS_IDT_OFFSETS ; MAKE DS:SI POINT TO
279 014C 8C C8 MOV AX,CS ; INTERRUPT ENTRY POINTS
280 014E BE D8 MOV DS,AX
281 0150 B0 D0A0 MOV DI,SYN_IDT_LOC ; POINT TO SYS_IDT_LOC
282 0152 2B C0 SUB AX,AX
283 0153 00 00 MOV ES,AX ; WHERE THE IDT WILL BE
284 0155 B0 0040 MOV BX,DS,ROM_CS ; CS IS THE SAME FOR ALL INTERRUPTS
285 0156 00 0040 MOV DH,TRAP_GATE ; ACCESS RIGHTS BYTE FOR THE GATE
286 0158 B0 0000 MOV DL,0 ; THE WORD COUNT FIELD IS UNUSED
287 015E B9 0020 MOV CX,32 ; THERE ARE 32 RESERVED INTERRUPTS
288 0161 LOW_IDT: ; THIS LOOP BUILDS 32 DESCRIPTORS IN THE
289
290 0161 A5 MOVSW ; IDT FOR THE RESERVED INTERRUPTS
291
292 0162 B8 C3 MOV AX,BX ; GET ROUTINE ENTRY POINT
293 0164 AB STOSW ; AND PUT IT IN THE OFFSET FIELD
294 0165 B8 C2 MOV AX,DX ; GET THE SYSTEM CODE SEGMENT SELECTOR
295 0166 00 00 STOSW ; AND PUT IT IN THE SELECTOR FIELD
296 0168 B0 0000 MOV AX,0 ; GET THE INTERRUPT GATE BYTE
297 016B AB STOSW ; AND PUT IN THE ACCESS RIGHTS FIELD
298 016C E2 F3 LOOP LOW_IDT ; ZERO OUT THE RESERVED POSITIONS
299 016E B9 00E0 MOV CX,256-32 ; AND REPEAT AS DIRECTED
300 0171 B0 2777 R MOV BP,OFFSET FREE_INTS ; 256 TOTAL - 32 DONE = WHATEVER IS LEFT
301
302 0174 HIGH_IDT: ; THERE IS A COPY OF AN UN-INITIALIZED
303 0174 B8 F5 MOV SI,BP ; INTERRUPT DESCRIPTOR AT FREE_INTS
304
305 0176 A5 MOVSW ; DS:SI --> FREE DESCRIPTOR
306 0177 A5 MOVSW ; (ES:DI) LEFT OFF AT INT 32!
307 0177 A5 MOVSW ; MOVE OFFSET OF THE IRET INSTRUCTION
308 0179 AB STOSW ; MOVE THE CS SELECTOR
309 017A E2 F8 LOOP HIGH_IDT ; MOVE THE ACCESS RIGHTS BYTE
310
311 ;----- INITIALIZE THE ENTRY POINTS FOR POST TEST
312
313 017C 26: C7 06 D10 0098 R MOV ES:(SYS_IDT_LOC+(032*DESC_LEN).ENTRY_POINT),OFFSET SYS_32
314 017D 26: C7 06 D10 0099 R MOV ES:(SYS_IDT_LOC+(033*DESC_LEN).ENTRY_POINT),OFFSET SYS_33
315 018A 26: C7 06 D10 009A R MOV ES:(SYS_IDT_LOC+(034*DESC_LEN).ENTRY_POINT),OFFSET SYS_34
316 0191 26: C7 06 D10 009B R MOV ES:(SYS_IDT_LOC+(035*DESC_LEN).ENTRY_POINT),OFFSET SYS_35
317 0198 26: C7 06 D10C 0044 R MOV ES:(SYS_IDT_LOC+(036*DESC_LEN).ENTRY_POINT),OFFSET SYS_36
318 019F 26: C7 06 D10C 004C R MOV ES:(SYS_IDT_LOC+(037*DESC_LEN).ENTRY_POINT),OFFSET SYS_37
319 01A6 26: C7 06 D10D 0080 R MOV ES:(SYS_IDT_LOC+(038*DESC_LEN).ENTRY_POINT),OFFSET SYS_38
320 01AD C3 RET
321
322 01AE IRET_ADDR IRET LABEL WORD ; FOR UN-INITIALIZED INTERRUPTS
323 01AE CF IRET ; NULL_RETURN

```

```

324
325
326
327
328
329 = 01AF
330
331
332
333 01AF 0000
334 01B1 0000
335 01B3 00
336 01B4 00
337 01B5 0000
338
339
340
341 01B7 0088
342 01B9 D8A0
343 01B8 00
344 01BC 93
345 01BD 0000
346
347
348
349 01BF 0800
350 01C1 D0A0
351 01C3 00
352 01C4 93
353 01C5 0000
354
355
356
357 01C7 0300
358 01C9 0400
359 01CB 00
360 01CC 93
361 01CD 0000
362
363
364
365 01CF 1000
366 01D1 0000
367 01D3 0B
368 01D4 93
369 01D5 0000
370
371
372
373 01D7 4000
374 01D9 8000
375 01D9 9B
376 01DC 93
377 01DD 0000
378
379
380
381 01DF FFFF
382 01E1 0000
383 01E3 0A
384 01E4 93
385 01E5 0000
386
387
388
389 01E7 FFFF
390 01E9 0000
391 01EB 0B
392 01EC 93
393 01ED 0000
394
395
396
397 01EF FFFF
398 01F1 0000
399 01F1 F
400 01F4 9B
401 01F5 0000
402
403
404
405 01F7 FFFF
406 01F9 0000
407 01FB 00
408 01FC 93
409 01FD 0000
410
411
412
413 01FF FFFF
414 0201 0000
415 0203 00
416 0204 93
417 0205 0000
418
419
420
421 0207 FFFF
422 0209 0000
423 020A 00
424 020C 93
425 020D 0000
426
427
428
429 020F FFFF
430 0211 0000
431 0213 00
432 0214 93
433 0215 0000

PAGE
; THE FOLLOWING DATA DEFINES THE PRE-INITIALIZED GDT FOR POST TESTS.
; THESE MUST BE INITIALIZED IN THE ORDER IN WHICH THEY APPEAR IN THE
; GDT_DEF STRUCTURE DEFINITION AS IT IS IN "SYSDATA.INC".
GDT_DATA_START EQU $

;---- FIRST ENTRY UNUSABLE - (UNUSED_ENTRY)

DW 0 ; SEGMENT LIMIT
DW 0 ; SEGMENT BASE ADDRESS - LOW WORD
DB 0 ; SEGMENT BASE ADDRESS - HIGH BYTE
DB 0 ; ACCESS RIGHTS BYTE
DW 0 ; RESERVED - MUST BE ZERO

;---- THE GDT ITSELF - (GDT_PTR)

DW GDT_LEN ; SEGMENT LIMIT
DW GDT_LOC ; SEGMENT BASE ADDRESS - LOW WORD
DB 0 ; SEGMENT BASE ADDRESS - HIGH BYTE
DB CPL0_DATA_ACCESS ; ACCESS RIGHTS BYTE
DW 0 ; RESERVED - MUST BE ZERO

;---- THE SYSTEM IDT DESCRIPTOR - (SYS_IDT_PTR)

DW SYS_IDT_LEN ; SEGMENT LIMIT
DW SYS_IDT_LOC ; SEGMENT BASE ADDRESS - LOW WORD
DB 0 ; SEGMENT BASE ADDRESS - HIGH BYTE
DB CPL0_DATA_ACCESS ; ACCESS RIGHTS BYTE
DW 0 ; RESERVED - MUST BE ZERO

;---- THE SYSTEM DATA AREA DESCRIPTOR - (RSDA_PTR)

DW SDA_LEN ; SEGMENT LIMIT
DW SDA_LOC ; SEGMENT BASE ADDRESS - LOW WORD
DB 0 ; SEGMENT BASE ADDRESS - HIGH BYTE
DB CPL0_DATA_ACCESS ; ACCESS RIGHTS BYTE
DW 0 ; RESERVED - MUST BE ZERO

;---- COMPATIBLE MONOCHROME DISPLAY REGEN BUFFER - (C_BWCRT_PTR)

DW MCRT_SIZE ; SEGMENT LIMIT
DW MCRT0_LO ; SEGMENT BASE ADDRESS - LOW WORD
DB MCRT0_HI ; SEGMENT BASE ADDRESS - HIGH BYTE
DB CPL0_DATA_ACCESS ; ACCESS RIGHTS BYTE
DW 0 ; RESERVED - MUST BE ZERO

;---- COMPATIBLE COLOR DISPLAY REGEN BUFFER - (C_CCRT_PTR)

DW CCRT_SIZE ; SEGMENT LIMIT
DW CCRT0_LO ; SEGMENT BASE ADDRESS - LOW WORD
DB CCRT0_HI ; SEGMENT BASE ADDRESS - HIGH BYTE
DB CPL0_DATA_ACCESS ; ACCESS RIGHTS BYTE
DW 0 ; RESERVED - MUST BE ZERO

;---- ENHANCED GRAPHIC ADAPTER REGEN BUFFER - (E_CCRT_PTR)

DW ECCR_SIZE ; SEGMENT LIMIT
DW ECCR0_LO ; SEGMENT BASE ADDRESS - LOW WORD
DB ECCR0_HI ; SEGMENT BASE ADDRESS - HIGH BYTE
DB CPL0_DATA_ACCESS ; ACCESS RIGHTS BYTE
DW 0 ; RESERVED - MUST BE ZERO

;---- SECOND PART OF EGA - (E_CCRT_PTR2)

DW ECCRT_SIZE ; SEGMENT LIMIT
DW ECCRT0_HI_LO ; SEGMENT BASE ADDRESS - LOW WORD
DB ECCRT0_HI_HI ; SEGMENT BASE ADDRESS - HIGH BYTE
DB CPL0_DATA_ACCESS ; ACCESS RIGHTS BYTE
DW 0 ; RESERVED - MUST BE ZERO

;---- CODE SEGMENT FOR POST CODE, SYSTEM IDT - (SYS_ROM_CS)

DW MAX_SEG_LEN ; SEGMENT LIMIT
DW CSSEG0_LO ; SEGMENT BASE ADDRESS - LOW WORD
DB CSSEG0_HI ; SEGMENT BASE ADDRESS - HIGH BYTE
DB CPL0_CODE_ACCESS ; ACCESS RIGHTS BYTE
DW 0 ; RESERVED - MUST BE ZERO

;---- TEMPORARY DESCRIPTOR FOR ES - (ES_TEMP)

DW MAX_SEG_LEN ; SEGMENT LIMIT
DW NSEGO_LO ; SEGMENT BASE ADDRESS - LOW WORD
DB NSEGO_HI ; SEGMENT BASE ADDRESS - HIGH BYTE
DB CPL0_DATA_ACCESS ; ACCESS RIGHTS BYTE
DW 0 ; RESERVED - MUST BE ZERO

;---- TEMPORARY DESCRIPTOR FOR CS AS A DATA SEGMENT - (CS_TEMP)

DW MAX_SEG_LEN ; SEGMENT LIMIT
DW NSEGO_LO ; SEGMENT BASE ADDRESS - LOW WORD
DB NSEGO_HI ; SEGMENT BASE ADDRESS - HIGH BYTE
DB CPL0_DATA_ACCESS ; ACCESS RIGHTS BYTE
DW 0 ; RESERVED - MUST BE ZERO

;---- TEMPORARY DESCRIPTOR FOR SS - (SS_TEMP)

DW MAX_SEG_LEN ; SEGMENT LIMIT
DW NSEGO_LO ; SEGMENT BASE ADDRESS - LOW WORD
DB NSEGO_HI ; SEGMENT BASE ADDRESS - HIGH BYTE
DB CPL0_DATA_ACCESS ; ACCESS RIGHTS BYTE
DW 0 ; RESERVED - MUST BE ZERO

;---- TEMPORARY DESCRIPTOR FOR DS - (DS_TEMP)

DW MAX_SEG_LEN ; SEGMENT LIMIT
DW NSEGO_LO ; SEGMENT BASE ADDRESS - LOW WORD
DB NSEGO_HI ; SEGMENT BASE ADDRESS - HIGH BYTE
DB CPL0_DATA_ACCESS ; ACCESS RIGHTS BYTE
DW 0 ; RESERVED - MUST BE ZERO

```

```

434 PAGE
435
436 0217 TR_Loc: DW 00800H ; SEGMENT LIMIT
437 0217 0800 DW 00000H ; SEGMENT BASE ADDRESS - LOW WORD
438 0219 C000 DB 0 ; SEGMENT BASE ADDRESS - HIGH BYTE
439 021B 00 DB FREE_TSS ; ACCESS RIGHTS BYTE
440 021C 61 DW 0 ; RESERVED - MUST BE ZERO
441 021D 0000
442
443 ;----- (POST_TSS_PTR)
444
445 021F 0800 DW 00800H ; SEGMENT LIMIT
446 0221 0217 R TR_Loc: DW 00000H ; SEGMENT BASE ADDRESS - LOW WORD
447 0222 00 DB 0 ; SEGMENT BASE ADDRESS - HIGH BYTE
448 0224 93 DB CPL0_DATA_ACCESS ; ACCESS RIGHTS BYTE
449 0225 0000 DW 0 ; RESERVED - MUST BE ZERO
450
451 ;----- (POST_LDT)
452 0227 LDT_Loc: DW GDT_LEN ; SEGMENT LIMIT
453 0227 0088 DW 00000H ; SEGMENT BASE ADDRESS - LOW WORD
454 0229 0000 DB 0 ; SEGMENT BASE ADDRESS - HIGH BYTE
455 0228 00 DB LDT_DESC ; ACCESS RIGHTS BYTE
456 0220 E2 DW 0 ; RESERVED - MUST BE ZERO
457 022D 0000
458
459 ;----- (POST_LDT_PTR)
460
461 022F 0088 DW GDT_LEN ; SEGMENT LIMIT
462 0231 0227 R DW LDT_LOC ; SEGMENT BASE ADDRESS - LOW WORD
463 0233 00 DB 0 ; SEGMENT BASE ADDRESS - HIGH BYTE
464 0234 93 DB CPL0_DATA_ACCESS ; ACCESS RIGHTS BYTE
465 0235 0000 DW 0 ; RESERVED - MUST BE ZERO
466
467 = 0237 GDT_DATA_END EQU $
468
469 ;----- END OF PRE-ALLOCATED GDT
470
471
472 ;----- ENTRY POINTS FOR THE FIRST 32 SYSTEM INTERRUPTS
473
474 0237 SYS_IDT_OFFSETS LABEL WORD
475
476 0237 0000 R DW OFFSET EXC_00 ; INTERRUPTS AS DEFINED
477 0239 0005 R DW OFFSET EXC_01 ; EXCPT 00 - DIVIDE ERROR
478 023B 0004 R DW OFFSET EXC_02 ; EXCPT 01 - SINGLE STEP
479 023D 000F R DW OFFSET EXC_03 ; EXCPT 02 - INT 3 SYSTEM REQUEST FOR DI
480 023F 0014 R DW OFFSET EXC_04 ; EXCPT 03 - BREAKPOINT
481 0241 0019 R DW OFFSET EXC_04 ; EXCPT 04 - INTO DETECT
482 0243 0030 R DW OFFSET EXC_05 ; EXCPT 05 - BOUND
483 0245 0031 R DW OFFSET EXC_06 ; EXCPT 06 - INVALID OPCODE
484 0247 0038 R DW OFFSET EXC_07 ; EXCPT 07 - Processor Ext Not Avail
485 0249 003C R DW OFFSET EXC_08 ; EXCPT 08 - Double Exception
486 024B 0040 R DW OFFSET EXC_09 ; EXCPT 09 - Processor Ext Segment Err
487 024D 0044 R DW OFFSET EXC_10 ; EXCPT 10 - TSS Bad In Gate Transfer
488 024F 0048 R DW OFFSET EXC_11 ; EXCPT 11 - Segment Not Present
489 0251 0049 R DW OFFSET EXC_12 ; EXCPT 12 - Stack Segment Not Present
490 0253 0050 R DW OFFSET EXC_13 ; EXCPT 13 - General Protection
491 0255 0054 R DW OFFSET EXC_14 ; EXCPT 14 - Processor Extension Error
492 0257 0058 R DW OFFSET EXC_15 ; EXCPT 15 - Processor Extension Error
493 0259 005C R DW OFFSET EXC_16 ; EXCPT 16 - Processor Extension Error
494 025B 0060 R DW OFFSET EXC_17 ; EXCPT 17 - Processor Extension Error
495 025D 0061 R DW OFFSET EXC_18 ; EXCPT 18 - Processor Extension Error
496 025F 0068 R DW OFFSET EXC_19 ; EXCPT 19 - Processor Extension Error
497 0261 006C R DW OFFSET EXC_20 ; EXCPT 20 - Processor Extension Error
498 0263 0070 R DW OFFSET EXC_21 ; EXCPT 21 - Processor Extension Error
499 0265 0074 R DW OFFSET EXC_22 ; EXCPT 22 - Processor Extension Error
500 0267 0078 R DW OFFSET EXC_23 ; EXCPT 23 - Processor Extension Error
501 0269 0079 R DW OFFSET EXC_24 ; EXCPT 24 - Processor Extension Error
502 026B 0080 R DW OFFSET EXC_25 ; EXCPT 25 - Processor Extension Error
503 026D 0084 R DW OFFSET EXC_26 ; EXCPT 26 - Processor Extension Error
504 026F 0088 R DW OFFSET EXC_27 ; EXCPT 27 - Processor Extension Error
505 0271 008C R DW OFFSET EXC_28 ; EXCPT 28 - Processor Extension Error
506 0273 0090 R DW OFFSET EXC_29 ; EXCPT 29 - Processor Extension Error
507 0275 0094 R DW OFFSET EXC_30 ; EXCPT 30 - Processor Extension Error
508
509 ;----- FORMAT INTERRUPT DESCRIPTORS (GATES) 32 - 255
510
511 0277 01AE R FREE_INTS DW OFFSET IRET_ADDR ; DESTINATION OFFSET
512 0279 0040 DW SYS_ROM_CS ; DESTINATION SEGMENT
513 027B 00 86 DB 0,INT_GATE ; UNUSED AND ACCESS RIGHTS BYTE
514 027D ENDP
515
516 027D CODE ENDS
517 END

```

```

PAGE 118,121
TITLE TEST6 ---- 06/10/85 POST TESTS AND SYSTEM BOOT STRAP
.286C
.LIST
0000      CODE  SEGMENT BYTE PUBLIC
0000          PUBLIC  BOOT_STRAP_
0000          PUBLIC  POSTS
0000          PUBLIC  STGTST_CNT
0000          PUBLIC  ROM_ERR
0000          PUBLIC  XMIT_8042
0000          EXTRN  CMOS_READ:NEAR
0000          EXTRN  DDS_READ:NEAR
0000          EXTRN  DISK_BASE:NEAR
0000          EXTRN  E602INEAR
0000          EXTRN  ERR_BEEP:NEAR
0000          EXTRN  E_MSG:NEAR
0000          EXTRN  F3A:NEAR
0000          EXTRN  PRT_SEG:NEAR
0000          ASSUME CS:CODE,DS:DATA
0000
POST6  PROC  NEAR
; THIS SUBROUTINE PERFORMS A READ/WRITE STORAGE TEST ON A BLOCK :
; DS = ADDRESS OF STORAGE SEGMENT BEING TESTED
; DS = ADDRESS OF STORAGE SEGMENT BEING TESTED
; CX = WORD COUNT OF STORAGE BLOCK TO BE TESTED
; EXIT PARAMETERS:
; ZERO FLAG = 0 IF STORAGE ERROR (DATA COMPARE OR PARITY
; CHECK), AL=0 DENOTES A PARITY CHECK. ELSE AL=XOR'D
; BIT PATTERN OF THE EXPECTED DATA PATTERN VS THE ACTUAL
; DATA READ.
; AX,BX,CX,DX,DI, AND SI ARE ALL DESTROYED.
;-----;
STGTST_CNT  PROC  NEAR
    MOV    BX,CX          ; SAVE WORD COUNT OF BLOCK TO TEST
    IN     AL,PORT_B
    OR     AL,RAM_PAR_OFF
    OUT   PORT_B,AL
    AND   AL,RAM_PAR_ON
    OUT   PORT_B,AL
;-----; ROLL A BIT THROUGH THE FIRST WORD
;-----;
    XOR   DX,DX          ; CLEAR THE INITIAL DATA PATTERN
    MOV   CX,16
    SUB  D1,D1
    SUB  SI,SI
    STC
    C1:
    RCL   DX,1            ; MOVE BIT OVER LEFT TO NEXT POSITION
    MOV   [D1],DX          ; STORE DATA PATTERN
    MOV   AX,[D1]
    XOR   AX,DX
    LOOPZ C1
    JNZ   C13             ; EXIT IF ERROR
;-----; CHECK CAS LINES FOR HIGH BYTE LOW BYTE
;-----;
    MOV   DX,0FF00H
    MOV   [D1],AX          ; TEST DATA - AX= 0000H
    MOV   [D1+1],DH
    MOV   AX,[D1]
    XOR   AX,DX
    JNZ   C13             ; ERROR EXIT IF NOT ZERO
    MOV   [D1],AX          ; STORE DATA PATTERN OF 0000H
    MOV   [D1],DH
    XCHG  DH,DL
    MOV   AX,[D1]
    XOR   AX,DX
    JNZ   C13             ; CHECK THE FIRST WRITTEN
;-----; CHECK FOR I/O OR BASE MEMORY ERROR
;-----;
    IN    AL,PORT_B
    XCHG  AL,AH
    IN    AL,DMA_PAGE+6
    AND  AH,AL
;-----; PARITY ERROR EXIT
;-----;
    MOV   AX,0
    JNZ   C13             ; RESTORE AX TO 0000
;-----; PARITY ERROR EXIT
;-----;
    MOV   DX,0AA55H
    C3:
    SUB  D1,D1
    SUB  SI,SI
    MOV   CX,BX
    MOV   AX,DX
    REP   STOSW
    MOV   CX,BX
    SUB  SI,SI
;-----; C6:
    LODSW
    XOR   AX,DX
    LOOPZ C6
    JNZ   C13             ; EXIT IF NOT EXPECTED (ERROR BITS ON)
;-----; CHECK FOR I/O OR BASE MEMORY ERROR
;-----;
    IN    AL,PORT_B
    XCHG  AL,AH
    IN    AL,DMA_PAGE+6
    AND  AH,AL

```

```

115          ;----- PARITY ERROR EXIT
116          MOV    AX,0
117          JNZ    C13      ; RESTORE AX TO 0000
118          ; GO IF YES
119
120          ;----- CHECK FOR END OF 64K BLOCK
121          AND   DX,DX
122          JZ    C13      ; ENDING ZERO PATTERN WRITTEN TO MEMORY?
123          ; YES - RETURN TO CALLER WITH AL=0
124
125          ;----- SETUP NEXT PATTERN
126          CMP   DX,055AA
127          JZ    C9       ; CHECK IF LAST PATTERN =55AA
128          ; GO IF NOT
129          CMP   DX,0101H
130          JZ    C10      ; LAST PATTERN 0101?
131          ; GO IF YES
132          MOV   DX,055AA
133          JMP   C3       ; WRITE 55AA TO STORAGE
134
135          ;----- INSURE PARITY BITS ARE NOT STUCK ON
136          CMP   DX,0101H
137          JMP   C3       ; WRITE 0101 TO STORAGE
138
139          ;----- EXIT STORAGE TEST
140          0088
141          0088 C3
142
143          ;----- CHECKER BOARD TEST
144          0089 2B FF
145          C10: SUB   DI,DI
146          MOV   CX,BX
147          SHR   CX,1
148          MOV   AX,101010101010101010B
149          MOV   SI,0101010101010101010B
150          ; FIRST CHECKER PATTERN
151          0095 96
152          XCHG  AX,SI
153          STOSW
154          XCHG  AX,SI
155          STOSW
156          LOOPZ C11
157          ; DO IT FOR CX COUNT
158          0098 2B F6
159          SUB   SI,SI
160          MOV   CX,BX
161          SHR   CX,1
162          MOV   AX,0101010101010101010B
163          ; CHECK CORRECT
164          00A7 AD
165          LODSW
166          XOR   AX,DI
167          JNZ   C13      ; GET THE DATA
168          ; CHECK CORRECT
169          ; EXIT IF NOT
170
171          00A8 33 C7
172          XOR   AX,DX
173          LOOPZ C12
174          ; CONTINUE TILL DONE
175          00A9 75 DC
176          JNZ   C13      ; EXIT IF NOT
177
178          ;----- CHECK FOR I/O OR BASE MEMORY PARITY CHECK
179          00B3 E4 61
180          IN    AL,PORT_B
181          XCHG  AL,AH
182          00B8 86 C4
183          IN    AL,DMA_PAGE+6
184          AND   AH,AL
185          00B9 22 E0
186
187          ;----- CHECKPOINT 32 FOR ADDRESS LINE 0->15 FAILURE
188          00BB B0 32
189          MOV   AL,32H
190          OUT   MFG_PORT,AL
191          ; <><><><><><><><><><>
192          00BD E6 80
193          MOV   AX,0
194          ; RESTORE AX (SET AX TO ZERO)
195          00BF B0 0000
196          JNZ   C13      ; EXIT IF PARITY ERROR
197
198          ;----- 64K ADDRESS TEST AND FILL WITH ZERO
199          00C4 48
200          DEC   AX
201          ; WRITE FIRST AND LAST LOCATION=FFFF
202          SUB   DI,DI
203          MOV   CX,BX
204          SUB   CX,2
205          STOSW
206          ; POINT TO START OF BLOCK
207          ; GET THE BLOCK COUNT
208          ; DO ALL LOCATIONS BUT LAST
209          ; WRITE FIRST LOCATION AS FFFFH
210          INC   AX
211          REP   STOSW
212          DEC   AX
213          STOSW
214          ; LAST WORD IS FFFF
215          ;----- CLEAR WORD 0 AND FFFF
216          SUB   SI,SI
217          MOV   CX,BX
218          SUB   CX,2
219          LODSW
220          XOR   AX,0FFFFH
221          JNZ   C13      ; GET THE DATA
222          ; CHECK CORRECT
223          ; EXIT IF NOT
224          00E2 E1 FB
225          LODSW
226          XOR   AX,0FFFFH
227          JNZ   C13      ; CONTINUE TILL LAST WORD
228          ; GO IF NOT CORRECT
229          ; GET LAST WORD
230          00E4 85 FFFF
231          LODSW
232          XOR   AX,0FFFFH
233          JNZ   C13      ; SB FFFF
234          ; EXIT IF NOT
235
236          ;----- CLEAR WORD 0 AND FFFE
237          SUB   DI,DI
238          STOSW
239          MOV   DI,0FFEFEH
240          STOSW
241
242          ;----- CLEAR WORD 0 AND FFFE
243          SUB   DI,DI
244          STOSW
245          MOV   DI,0FFEFEH
246          STOSW
247
248          ;----- CHECK FOR I/O OR BASE MEMORY
249          00F3 E4 61
250          IN    AL,PORT_B
251          XCHG  AL,AH
252          00F5 86 C4
253          IN    AL,DMA_PAGE+6
254          AND   AH,AL
255          00F9 22 E0
256          MOV   AX,0
257          ; SET AX EQUAL ZERO
258          00FB B0 0000
259          JMP   C13      ; ERROR EXIT IF ZF NOT SET
260          0100
261          STGTST_CNT
262          ENDP

```



```

343      ;----- CLEAR $BOOT_LOCN
344
345      01E 33 C0      XOR  AX,AX
346      0180 00 0100    MOV  CX,256
347      0183 BF TC00 R  MOV  D1,OFFSET $BOOT_LOCN ; CLEAR 256 WORDS
348      0186 F3 / AB   REP  STOSW
349
350      ;----- LOAD SYSTEM FROM DISKETTE -- CX HAS RETRY COUNT
351
352      0188 FB 0004    STI
353      0189 B9 0004    MOV  CX,4
354      018C 51          HI:  PUSH  CX
355      018D B4 00      MOV  AH,0
356      018F CD 13      INT  13H
357      0191 72 0F      JC   H2
358
359      0193 BB 0201    MOV  AX,201H
360      0196 2B D2      SUB  DX,DX
361      0198 BE C2      MOV  ES,DX
362      019A BB TC00 R  MOV  D1,OFFSET $BOOT_LOCN ; DRIVE 0, HEAD 0
363      019D B9 0001    MOV  CX,1
364      01A0 CD 13      INT  13H
365      01A1 72 0F      H2:  POP  CX
366      01A3 73 09      JNC  H4
367      01A5 80 FC 80  CMP  AH,80H
368      01A8 74 22      JZ   H5
369      01AA E2 E0      LOOP H1
370      01AC EB 1E      JMP  SHORT H5
371
372      ;----- BOOT RECORD READ SUCCESSFUL
373      ;----- INSURE FIRST BYTE OF LOADED BOOT RECORD IS VALID (NOT ZERO)
374
375      01AE 80 3E TC00 R 06  H4:  CMP  BYTE PTR $BOOT_LOCN,06H ; CHECK FOR FIRST INSTRUCTION INVALID
376      01B3 72 71          JB   H10 ; IF BOOT NOT VALID PRINT MESSAGE HALT
377
378      ;----- INSURE DATA PATTERN FIRST 8 WORDS NOT ALL EQUAL
379
380      01B5 BF TC00 R  MOV  D1,OFFSET $BOOT_LOCN ; CHECK DATA PATTERN
381      01B8 B9 0008    MOV  CX,8
382      01B8 A1 TC00 R  MOV  AX,WORD PTR $BOOT_LOCN ; CHECK THE NEXT 8 WORDS
383
384      01B8 83 C7 02    H4A:  ADD  D1,2
385      01C1 3B D5      CMP  AX,[D1]
386      01C3 E1 F9      LOOPZ H4A
387      01C5 74 5F      JZ   H10
388
389      01C7 EA TC00 ---- R H4_A: JMP  $BOOT_LOCN
390
391      ;----- ATTEMPT BOOTSTRAP FROM FIXED DISK
392
393      01CC B0 44      H5:  MOV  AL,044H
394      01CE E6 80      OUT  MFG_PORT,AL ; <><><><><><><><><><>
395
396      01D0 8E 0000 E  ASSUME DS:DATA
397      01D3 F6 06 008B R 01 TEST  DL,ASTRATE,DUAL ; FLOPPY/FIXED DISK CARD INSTALLED
398
399      01D8 B8 ---- R ASSUME DS:AB50
400      01D8 BE D8      MOV  AX,AB50
401      01D9 74 3D      MOV  DS,AX
402
403      ;----- CHECK FOR FIXED DISK INITIALIZATION ERROR
404
405      01DF B0 0E      MOV  AL,CMOS_DIAG ; GET POST POWER ON STATUS (NMI ENABLED)
406      01E1 E8 0000 E  CALL  CMOS_READ
407      01E4 AB 08      TEST  AL,HF_FAIL ; FROM DIAGNOSTIC STATUS BYTE
408      01E6 75 34      JNZ  H9 ; DID WE HAVE A FIXED DISK FAILURE?
409
410      01E8 2B C0      SUB  AX,AX
411      01EA 2B D2      SUB  DX,DX
412      01EC CD 13      INT  13H
413      01F0 B9 0003    MOV  CX,3
414
415      01F1 51          H6:  PUSH  CX
416      01F2 BA 0080    MOV  DX,0080H ; SAVE RETRY COUNT
417      01F5 BB 0201    MOV  CX,0201H ; FIXED DISK ZERO
418      01F6 2B 0B      SUB  BX,BX
419      01F7 00 00      MOV  ES,BX
420      01FC BB TC00 R  MOV  D1,OFFSET $BOOT_LOCN ; READ IN A SINGLE SECTOR
421      01FF B9 0001    MOV  CX,1
422      0202 CD 13      INT  13H
423      0204 59          POP  CX
424      0205 72 08      JC   H8
425      0207 B1 3E 7DFF R AA55  CMP  WORD PTR $BOOT_LOCN+510D,0AA55H ; TEST FOR GENERIC BOOT BLOCK
426      020D 74 B8      JZ   H4_A
427
428      020F 51          H8:  PUSH  CX
429      0210 BA 0080    MOV  CX,0080H ; TO THE BOOT LOCATION
430      0213 2B C0      SUB  AX,AX
431      0216 CD 13      INT  13H
432      0217 59          POP  CX
433      0218 72 08      JC   H10
434      021A E2 D5      LOOP H6
435
436      ;----- UNABLE TO IPL FROM THE DISKETTE OR FIXED DISK
437
438      021C B0 45      H9:  MOV  AL,045H
439      021E E6 80      OUT  MFG_PORT,AL ; <><><><><><><><><>
440
441      0220 CD 18      INT  18H
442
443      ;----- HARD FILE RESET FAILURE
444
445      0222 E2 EB      H10A: LOOP H8
446      0224 EB F6      JMP  H9
447
448      ;----- IF DISKETTE READ OK BUT BOOT RECORD IS NOT STOP SYSTEM ALLOW SOFT RESET
449
450      0226 BE 0000 E  H10: MOV  SI,OFFSET E602 ; PRINT DISKETTE BOOT
451      0229 EB 0000 E  CALL  E_MSG
452      022C EB FE      H11: JMP  HT1 ; PRINT MESSAGE
453      022E
454      022E POST6 ENDP
455      022E CODE ENDS
456      022E END

```

```

1 PAGE 118,121
2 TITLE DSKETTE -- 06/10/85 DISKETTE BIOS
3 .286C
4 .LIST
5 0000      CODE SEGMENT BYTE PUBLIC
6
7         PUBLIC  DSK_INT_1
8         PUBLIC  SEEK
9         PUBLIC  DSKETTE_SETUP
10        PUBLIC  DSKETTE_10_1
11
12        EXTRN  CMOS_READ:NEAR           ; READ CMOS LOCATION ROUTINE
13        EXTRN  DDS:NEAR                ; LOAD DS WITH DATA SEGMENT SELECTOR
14        EXTRN  DISK_BASE:NEAR          ; DISKETTE PARAMETER TABLE LOCATION
15        EXTRN  WAIT:NEAR               ; FIXED WAIT ROUTINE - (CX)=15,086 US
16
17 ;-- INT 13H -----
18 ; DISKETTE I/O
19 ; THIS INTERFACE PROVIDES ACCESS TO THE 5 1/4" DISKETTE DRIVES
20 ; 320/360K DISKETTE DRIVES AND 1.2M DISKETTE DRIVES SUPPORTED
21 ; INPUT
22 ; (AH)= 00H RESET DISKETTE SYSTEM
23 ; HARD RESET TO NEC, PREPARE COMMAND, RECALIBRATE REQUIRED
24 ; ON ALL DRIVES
25 ; (AH)= 01H READ THE STATUS OF THE SYSTEM INTO (AH)
26 ; #DSKETTE STATUS FROM LAST OPERATION IS USED
27
28 ; REGISTERS FOR READ/WRITE/VERIFY/FORMAT
29 ; (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)
30 ; (DH) - HEAD NUMBER (0-1 ALLOWED, NOT VALUE CHECKED)
31 ; (CH) - TRACK NUMBER (NOT VALUE CHECKED)
32 ; (CL) - SECTOR NUMBER (NOT VALUE CHECKED)
33 ;          MEDIA   DRIVE   TRACK NUMBER
34 ;          320/360 320/360 0-39
35 ;          320/360 1.2M    0-39
36 ;          1.2M    1.2M    0-79
37 ;          720K   720K    0-79
38 ; (CL) - SECTOR NUMBER (NOT VALUE CHECKED, NOT USED FOR FORMAT)
39 ;          MEDIA   DRIVE   SECTOR NUMBER
40 ;          320/360 320/360 1-8/9
41 ;          320/360 1.2M    1-8/9
42 ;          1.2M    1.2M    1-15
43 ;          720K   720K    1-9
44 ; (AL) - NUMBER OF SECTORS (NOT VALUE CHECKED)
45 ;          MEDIA   DRIVE   MAX NUMBER OF SECTORS
46 ;          320/360 320/360 8/9
47 ;          320/360 1.2M    8/9
48 ;          1.2M    1.2M    15
49 ;          720K   720K    9
50
51 ; (ES:BX) - ADDRESS OF BUFFER ( REQUIRED FOR VERIFY)
52
53 ; (AH)= 02H READ THE DESIRED SECTORS INTO MEMORY
54
55 ; (AH)= 03H WRITE THE DESIRED SECTORS FROM MEMORY
56
57 ; (AH)= 04H VERIFY THE DESIRED SECTORS
58
59 ; (AH)= 05H FORMAT THE DESIRED TRACK
60 ; FOR THE FIRST OPERATION, THE BUFFER POINTER (ES,BX) MUST
61 ; POINT TO THE CORRECTION OF DESIRED ADDRESS FIELD FOR THE
62 ; TRACK. EACH FIELD IS COMPOSED OF 4 BYTES (C,H,R,N) WHERE
63 ; C = TRACK NUMBER, H=HEAD NUMBER, R = SECTOR NUMBER, N = NUMBER
64 ; OF BYTES PER SECTOR (00=128, 01=256, 02=512, 03=1024).
65 ; THERE MUST BE ONE ENTRY FOR EVERY SECTOR ON THE TRACK.
66 ; THE INFORMATION IS USED TO FIND THE REQUESTED SECTOR DURING
67 ; READ/WRITE ACCESS.
68 ; PRIOR TO FORMATTING A DISKETTE, IF THERE EXISTS MORE THAN
69 ; ONE SUPPORTED MEDIA FORMAT TYPE WITHIN THE DRIVE IN QUESTION,
70 ; THEN "SET DASD TYPE" (INT 13H, AH = 17H) MUST BE CALLED TO
71 ; SET THE DISKETTE TYPE THAT IS TO BE FORMATTED. IF "SET DASD
72 ; TYPE" IS NOT CALLED, THE FORMAT ROUTINE WILL ASSUME THE
73 ; MEDIA FORMAT TO BE THE MAXIMUM CAPACITY OF THE DRIVE.
74 ; IN ORDER TO FORMAT 320/360K MEDIA IN EITHER A 320/360K OR
75 ; 1.2M DISKETTE DRIVE THE GAP LENGTH FOR FORMAT PARAMETER
76 ; DISK BASE MUST BE CHANGE TO 050H. ALSO THE EOT
77 ; PARAMETER (LAST SECTOR ON TRACK) MUST BE SET TO THE
78 ; DESIRED NUMBER OF SECTORS/TRACK - 8 FOR 320K, 9 FOR 360K.
79 ; DISK BASE MUST BE POINTED TO BY DISK POINTER LOCATED AT
80 ; ABSOLUTE ADDRESS 0178H.
81 ; WHEN 320/360K FORMAT OPERATIONS ARE COMPLETE, THE PARAMETERS
82 ; SHOULD BE RESTORED TO THEIR RESPECTIVE INITIAL VALUES.
83
84 ; (AH)= 08H READ DRIVE PARAMETERS
85 ; REGISTERS
86 ; INPUT
87 ; (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)
88 ; OUTPUT
89 ; (ES:DI) POINTS TO DISK BASE
90 ; (CH) - LOW ORDER 8 OF 10 BITS MAXIMUM NUMBER OF TRACKS
91 ; (CL) - BITS 5 & 6 HIGH ORDER TWO BITS OF MAXIMUM TRACKS
92 ; (DH) - MAXIMUM HEAD NUMBER
93 ; (BL) - NUMBER OF DISKETTE DRIVES INSTALLED
94 ; (BH) - 0
95 ; (BL) - BITS 7 THRU 4 - 0
96 ; (BL) - BITS 3 THRU 0 - VALID DRIVE TYPE VALUE IN CMOS
97 ; (AX) - 0
98 ; UNDER THE FOLLOWING CIRCUMSTANCES:
99 ; (1) THE DRIVE NUMBER IS INVALID,
100 ; (2) THE DRIVE TYPE IS UNKNOWN AND CMOS IS NOT PRESENT,
101 ; (3) THE DRIVE TYPE IS UNKNOWN AND CMOS IS BAD,
102 ; (4) ONE OR MORE DRIVE TYPES IS UNKNOWN AND THE CMOS DRIVE TYPE IS INVALID
103 ; THEN ES,AX,CX,DH,DI=0 AND NUMBER OF DRIVES,
104 ; #DSKETTE_STATUS = 0 AND CY IS RESET.
105 ; IF NO DRIVES ARE PRESENT THEN; ES,AX,BX,CX,DH,DI=0.
106 ; #DSKETTE_STATUS = 0 AND CY IS RESET.
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
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
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
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
```

PAGE

```
1 (AH)= 15H READ DASD TYPE
REGISTERS
(AH) - ON RETURN IF CARRY FLAG NOT SET, OTHERWISE ERROR
00 - DRIVE NOT PRESENT
01 - DISKETTE, NO CHANGE LINE AVAILABLE
02 - DISKETTE, CHANGE LINE AVAILABLE
03 - FIXED DISK
(DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)
```

```
1 (AH)= 16H DISK CHANGE LINE STATUS
REGISTERS
(AH)=00 - DISK CHANGE LINE NOT ACTIVE
06 - DISK CHANGE LINE ACTIVE & CARRY BIT ON
(DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)
```

```
1 (AH)= 17H SET DASD TYPE FOR FORMAT
REGISTERS
(AL) - 00 - NOT USED
01 - DISKETTE 320/360K IN 360K DRIVE
02 - DISKETTE 360K IN 1.2M DRIVE
03 - DISKETTE 1.2M IN 1.2M DRIVE
04 - DISKETTE 720K IN 720K DRIVE
(DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)
DO NOT USE WHEN DISKETTE ATTACH CARD USED!
```

DISK CHANGE STATUS IS ONLY CHECKED WHEN A 1.2M BYTE DISKETTE DRIVE IS SPECIFIED. IF THE DISK CHANGE LINE IS FOUND TO BE ACTIVE THE FOLLOWING ACTIONS TAKE PLACE:

- IF ATTEMPT TO READ DISK CHANGE LINE TO INACTIVE STATE.
- IF ATTEMPT SUCCESS SET DASD TYPE FOR FORMAT AND RETURN DISK CHANGE ERROR CODE
- IF ATTEMPT FAILS RETURN TIMEOUT ERROR CODE AND SET DASD TYPE TO A PREDETERMINED STATE INDICATING MEDIA TYPE UNKNOWN.
- IF THE DISK CHANGE LINE IN INACTIVE PERFORM SET DASD TYPE FOR FORMAT.

DATA VARIABLE -- **DISK\_POINTER**  
DOUBLE WORD POINTER TO THE CURRENT SET OF DISKETTE PARAMETERS

OUTPUT FOR ALL FUNCTIONS

AH = STATUS OF OPERATION

DS,ES,CS,DX PRESERVED

NOTE: IF AN ERROR IS REPORTED BY THE DISKETTE CODE, THE APPROPRIATE ACTION IS TO RESET THE DISKETTE, THEN RETRY THE OPERATION. ON READ ACCESSSES, NO MOTOR START DELAY IS TAKEN, SO THAT THREE RETRIES ARE REQUIRED ON READS TO ENSURE THAT THE PROGRAM IS NOT DUE TO MOTOR START-UP.

```
1 LIST
1 DISKETTE STATE MACHINE - ABSOLUTE ADDRESS 40:90 (DRIVE A) & 91 (DRIVE B)
1 LIST
```

7	6	5	4	3	2	1	0
---	---	---	---	---	---	---	---

```
1 RESERVED
1 PRESENT STATE
1 000: 360K IN 360K DRIVE UNESTABLISHED
1 001: 360K IN 1.2M DRIVE UNESTABLISHED
1 010: 1.2M IN 1.2M DRIVE UNESTABLISHED
1 011: 360K IN 360K DRIVE ESTABLISHED
1 100: 360K IN 1.2M DRIVE ESTABLISHED
1 101: 1.2M IN 1.2M DRIVE ESTABLISHED
1 110: RESERVED
1 111: NONE OF THE ABOVE
```

-----> MEDIA/DRIVE ESTABLISHED

-----> DOUBLE STEPPING REQUIRED (360K IN 1.2M DRIVE)

-----> DATA TRANSFER RATE FOR THIS DRIVE:

```
00: 500 KBS
01: 300 KBS
01: 250 KBS
11: RESERVED
```

```
1 LIST
1 STATE OPERATION STARTED - ABSOLUTE ADDRESS 40:92 (DRIVE A) & 93 (DRIVE B)
1 PRESENT CYLINDER NUMBER - ABSOLUTE ADDRESS 40:94 (DRIVE A) & 95 (DRIVE B)
```

```

205 PAGE ASSUME CS:CODE,DS:DATA,ES:DATA
206
207 .LIST
208 0000 F0
209 0000 FB
210 0001 55
211 0002 57
212 0003 52
213 0004 53
214 0005 51
215 0006 88 EC
216
217
218
219
220
221
222
223
224
225
226
227
228
229 0008 1E
230 0009 56
231 000A E8 0000 E
232 000B 00 18
233 0010 72 02
234
235 0012 B4 14
236 0014
237 0014 80 FC 01
238 0014 76 CC
239 0019 80 FC 08
240 001C 74 07
241 001E 80 FA 01
242 0021 76 02
243 0023 B4 14
244 0025
245 0026 8A CC
246 0027 32 ED
247 0029 D0 E1
248 002B BB 004E R
249 002E 03 D9
250 002F 00 E6
251 0032 32 6
252 0034 BB F0
253 0036 8B FA
254 0038 BA 26 0041 R
255 003C C6 06 0041 R 00
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274 0041 2E1 FF 17
275 0044 8E
276 0045 1F
277 0046 59
278 0047 5B
279 0048 5A
280 0049 5F
281 004A 5D
282 004B CA 0002
283
284
285 004E 007E R
286 0050 0068 R
287 0051 0074 R
288 0054 0100 R
289 0056 010C R
290 0058 0118 R
291 005A 016A R
292 005C 016A R
293 005D 016A R
294 0060 016A R
295 0062 016A R
296 0064 016A R
297 0066 016A R
298 0068 016A R
299 0069 016A R
300 006C 016A R
301 006E 016A R
302 0070 016A R
303 0072 016A R
304 0074 016A R
305 0076 016A R
306 0078 021A R
307 007A 023C R
308 007C 0267 R
309 = 007E
310
311 007E

PAGE ASSUME CS:CODE,DS:DATA,ES:DATA
DISKETTE_10_I PROC FAR
    ST1:
    PUSH BP
    PUSH DI
    PUSH DX
    PUSH BX
    PUSH CX
    MOV BP, SP

    ;>>> ENTRY POINT FOR ORG 00C59H
    ;INTERRUPTS BACK ON
    ;USER REGISTER
    ;USER REGISTER
    ;HEAD #, DRIVE # OR USER REGISTER
    ;BUFFER OFFSET PARAMETER OR REGISTER
    ;TRACK # OR USER REGISTER
    ;BP => PARAMETER LIST DEP. ON AH
    ;[BP] = SECTOR #
    ;[BP+1] = TRACK #
    ;[BP+2] = BUFFER OFFSET
    ;FOR RETURN OF DRIVE PARAMETERS:
    ;CL/[BP] = BITS 16 HIGH BITS OF MAX CYL
    ;BITS 0-7 MAX SECTORS/TRACK
    ;CH/[BP+1] = LOW 8 BITS OF MAX CYL.
    ;BL/[BP+2] = BITS 7-4 = 0
    ;BITS 3-0 = VALID CMOS TYPE
    ;BH/[BP+3] = 0
    ;DL/[BP+4] = # DRIVES INSTALLED
    ;DH/[BP+5] = MAX HEAD #
    ;D1/[BP+6] = OFFSET TO DISK BASE
    ;BUFFER SEGMENT PARA OR USER REGISTER
    ;USER REGISTERS
    ;SEGMENT OF BIOS DATA AREA TO DS
    ;CH FOR > LARGEST FUNCTION
    ;FUNCTION OK

    PUSH DS
    PUSH SI
    CALL DDS
    CMP AH, [FNC_TAE-FNC_TAB]/2
    JBE OK_FUNC
    MOV AH, _FUNC

    MOV AH, 14H
    ;REPLACE WITH KNOWN INVALID FUNCTION

    OK_FUNC:
    CMP AH, 1
    JBE OK_DRV
    CMP AH, 8
    JZ OK_DRV
    CMP AH, 9
    JZ OK_DRV
    CMP AH, 10
    JZ OK_DRV
    CMP AH, 11
    JZ OK_DRV
    CMP AH, 12
    JZ OK_DRV
    CMP AH, 13
    JZ OK_DRV
    CMP AH, 14
    JZ OK_DRV
    ;REPLACE WITH KNOWN INVALID FUNCTION

    OK_DRV:
    MOV CL, AH
    XOR CH, CH
    SHL CL, 1
    ;FUNCTION TIMES 2
    MOV BX, [OFFSET FNC_TAB]
    ADD BX, CX
    MOV AH, AH
    XOR DH, DH
    MOV SI, AX
    MOV DL, DX
    MOV AH, [DSKETTE_STATUS]
    MOV AH, [DSKETTE_STATUS, 0]
    ;CL = FUNCTION
    ;CX = FUNCTION
    ;FUNCTION TIMES 2
    ;LOAD START OF FUNCTION TABLE
    ;ADD OFFSET INTO TABLE => ROUTINE
    ;AH = FUNCTION #, # OF SECTORS OR DASD TYPE
    ;DX = DRIVE #
    ;SI = HEAD #, # OF SECTORS OR DASD TYPE
    ;DI = DRIVE #
    ;DI, DX = # OF SECTORS OR DASD TYPE
    ;INITIALIZE FOR ALL OTHERS

    ;THROUGHOUT THE DISKETTE BIOS, THE FOLLOWING INFORMATION IS CONTAINED IN
    ;THE FOLLOWING MEMORY LOCATIONS AND REGISTERS. NOT ALL DISKETTE BIOS
    ;FUNCTIONS REQUIRE ALL OF THESE PARAMETERS.

    ;DI : DRIVE #
    ;SI-HI : HEAD #
    ;SI-LOW : # OF SECTORS OR DASD TYPE FOR FORMAT
    ;ES : BUFFER SEGMENT
    ;[BP] : SECTOR #
    ;[BP+1] : TRACK #
    ;[BP+2] : BUFFER OFFSET

    ;ACROSS CALLS TO SUBROUTINES THE CARRY FLAG (CY=1), WHERE INDICATED IN
    ;SUBROUTINE PROLOGUES, REPRESENTS AN EXCEPTION RETURN (NORMALLY AN ERROR
    ;CONDITION). IN MOST CASES, WHEN CY = 1, _DSKETTE_STATUS CONTAINS THE
    ;SPECIFIC ERROR CODE.

    CALL WORD PTR CS:[BX]
    ;(AH) = _DSKETTE_STATUS
    ;CALL THE REQUESTED FUNCTION
    POP SI
    ;RESTORE ALL REGISTERS
    POP DS
    POP CX
    POP BX
    POP DX
    POP DI
    POP BP
    RET 2
    ;THROW AWAY SAVED FLAGS

    FNC_TAB DW DISK_RESET ;AH = 00: RESET
    DW DISK_STATUS ;AH = 01: STATUS
    DW DISK_READ ;AH = 02: READ
    DW DISK_WRITE ;AH = 03: WRITE
    DW DISK_VERIFY ;AH = 04: VERIFY
    DW DISK_FORMAT ;AH = 05: FORMAT
    DW FNC_ERR ;AH = 06: INVALID
    DW FNC_ERR ;AH = 07: INVALID
    DW DISK_PARAMS ;AH = 08: LOAD/SAVE PARAMETERS
    FNC_ERR ;AH = 09: INVALID
    FNC_ERR ;AH = 0A: INVALID
    FNC_ERR ;AH = 0B: INVALID
    FNC_ERR ;AH = 0C: INVALID
    FNC_ERR ;AH = 0D: INVALID
    FNC_ERR ;AH = 0E: INVALID
    FNC_ERR ;AH = 0F: INVALID
    FNC_ERR ;AH = 10: INVALID
    FNC_ERR ;AH = 11: INVALID
    FNC_ERR ;AH = 12: INVALID
    FNC_ERR ;AH = 13: INVALID
    FNC_ERR ;AH = 14: INVALID
    DW DISK_TYPE ;AH = 15: READ DASD TYPE
    DW DISK_CHANGE ;AH = 16: CHANGE STATUS
    FORMAT_SET ;AH = 17: SET DASD TYPE
    FNC_TAE EQU $
    ENDP
DISKETTE_10_I ENDP

```

```

312
313
314 ;-----: DSK_RESET: RESET THE DISKETTE SYSTEM.
315 ;-----: ON EXIT:  @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
316
317 DSK_RESET PROC NEAR
318 007E
319 007E BA 03F2
320 0081 FA
321 0082 A0 003F R
322 0085 24 3F
323 0087 CO CO 04
324
325 008A OC 08
326 008C EE
327 008D C6 06 003E R 00
328 0092 EB 00
329 0094 OC 04
330 0095 00 00
331 0097 FB
332 0098 E8 087C R
333 0098 72 44
334 0090 B9 00C0
335
336 00A0
337 00A0 51
338 00A1 B8 00E0 R
339 00A4 50
340 00A5 B4 08
341 00A7 E8 07BD R
342 00A8 00
343 00A9 E8 08A4 R
344 00AE 59
345 00AF 72 30
346 00B1 3A 06 0042 R
347 00B5 75 24
348 00B6 FC C1
349 00B9 B0 9 C3
350 00BC 76 E2
351
352 ;-----: SEND SPECIFY COMMAND TO NEC
353
354 00BE B8 00D8 R
355 00C1 50
356 00C2 B4 03
357 00C4 E8 07BD R
358 00C7 2A D2
359 00C9 E8 06CC R
360 00C9 E8 07BD R
361 00CF B2 00
362 00D1 E8 06CC R
363 00D4 E8 07BD R
364 00D7 58
365
366 00D8
367 00D8 E8 0620 R
368 00D9 BB DE
369 00D0 8A C3
370
371 00E0
372 00E0 59
373 00E1
374 00E1 80 0E 0041 R 20
375 00E6 EB F0
376 00E8
377
378
379 ;-----: DSK_STATUS: DISKETTE STATUS.
380
381 ;-----: ON ENTRY: AH : STATUS OF PREVIOUS OPERATION
382
383 ;-----: ON EXIT: AH, @DSKETTE_STATUS, CY REFLECT STATUS OF PREVIOUS OPERATION.
384
385 DSK_STATUS PROC NEAR
386 00E8
387 00EC E8 0620 R
388 00EF BB DE
389 00F1 8A C3
390 00F3 C3
391 00F4
392
393 ;-----: DSK_READ: DISKETTE READ.
394
395 ;-----: ON ENTRY: DI : DRIVE #
396 ;-----: SI-HI : HEAD #
397 ;-----: SI-LOW : SECTOR #
398 ;-----: ES : BUFFER SEGMENT
399 ;-----: [BP] : SECTOR #
400 ;-----: [BP+1] : TRACK #
401 ;-----: [BP+2] : BUFFER OFFSET
402
403 ;-----: ON EXIT: @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
404
405 00F4
406 00F4 B8 26 0041 R
407 00F9 BB E646
408 00FC E8 035C R
409 00FF C3
410 0100
411
412 ;-----: DSK_WRITE: DISKETTE WRITE.
413
414 ;-----: ON ENTRY: DI : DRIVE #
415 ;-----: SI-HI : HEAD #
416 ;-----: SI-LOW : SECTOR #
417 ;-----: ES : BUFFER SEGMENT
418 ;-----: [BP] : SECTOR #
419 ;-----: [BP+1] : TRACK #
420 ;-----: [BP+2] : BUFFER OFFSET
421
422 ;-----: ON EXIT: @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
423
424 0100
425 0100 B8 C54A

```

```

426 0103 80 DE 003F R 80      OR      @MOTOR_STATUS,1000000B ; INDICATE WRITE OPERATION
427 0108 E8 035C R           CALL    RD_W_VF
428 010B C3                 RET
429 010C
430
431 ; DISK_VERIFY: DISKETTE VERIFY.
432
433 ; ON ENTRY: DI : DRIVE #
434 ;           SI-HI : HEAD #
435 ;           SI-LOW : # OF SECTORS
436 ;           ES : BUFFER SEGMENT
437 ;           [BP] : SECTOR #
438 ;           [BP+1] : TRACK #
439 ;           [BP+2] : BUFFER OFFSET
440
441 ; ON EXIT:  @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
442
443 010C
444 010C 80 26 003F R 7F      DISK_VERIFY PROC NEAR
445 0111 B8 E642 AND          @MOTOR_STATUS,0111111B ; INDICATE A READ OPERATION
446 0114 E8 035C R           MOV    AX,0E642H ; AX = NEC COMMAND, DMA COMMAND
447 0117 C3                 CALL    RD_W_VF
448 0118
449
450 ; DISK_FORMAT: DISKETTE FORMAT.
451
452 ; ON ENTRY: DI : DRIVE #
453 ;           SI-HI : HEAD #
454 ;           SI-LOW : # OF SECTORS
455 ;           ES : BUFFER SEGMENT
456 ;           [BP] : SECTOR #
457 ;           [BP+1] : TRACK #
458 ;           [BP+2] : BUFFER OFFSET
459
460 ; ON EXIT:  @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
461
462 0118
463 0118 E8 02C8 R           DISK_FORMAT PROC NEAR
464 011B E8 03CE R           CALL   XLAT_NEW
465 011E 80 0E 003F R 80      CALL   FMT_TINIT
466 0120 E8 035C R           OR    @MOTOR_STATUS,1000000B ; INDICATE WRITE OPERATION
467 0123 E8 035C R           CALL   MED_CHANGE
468 0128 E8 0451 R           JC    F1_DON
469 012B E8 04 A             CALL   SERD_RATE
470 012D E8 0471 R           MOV   AL,04AH ; WILL WRITE TO THE DISKETTE
471 0130 72 2D               CALL   DMA_SETUP
472 0132 84 4D               JC    FM_DON
473 0134 E8 04C7 R           MOV   AH,04DH ; ESTABLISH THE FORMAT COMMAND
474 0137 E8 015F R           CALL   NEC_INIT
475 013A 50                 MOV   AX,0FFH ; LOAD ERROR ADDRESS
476 013B B2 03               PUSH  AX ; PUSH NEC OUT ERROR RETURN
477 013D E8 06CC R           MOV   AX,0
478 0140 E8 07BD R           CALL   GET_PARM
479 0143 E8 02 04             CALL   NEC_OUTPUT
480 0145 E8 06CC R           MOV   DL,4 ; SECTORS/TRACK VALUE TO NEC
481 0148 E8 07BD R           CALL   GET_PARM
482 014B B2 07               CALL   NEC_OUTPUT
483 014D E8 06CC R           MOV   DL,7 ; GAP LENGTH VALUE TO NEC
484 0150 E8 07BD R           CALL   GET_PARM
485 0152 E8 02 05             CALL   NEC_OUTPUT
486 0155 E8 06CC R           MOV   DL,8 ; FILLER BYTE TO NEC
487 0158 E8 07BD R           CALL   GET_PARM
488 015B 58                 POP   AX ; THROW AWAY ERROR
489 015C E8 0530 R           CALL   NEC_TERM ; TERMINATE, RECEIVE STATUS, ETC.
490 015F
491 0161 E8 02EE R           CALL   XLAT_OLD
492 0162 E8 0620 R           CALL   SETUP_END ; TRANSLATE STATE TO COMPATIBLE MODE
493 0165 B8 DE               MOV   BX,SI- ; VARIOUS CLEANUPS
494 0167 8A C3               MOV   AL,BL ; GET SAVED AL TO BL
495 0169 C3                 RET
496 016A
497
498 ; FNC_ERR : INVALID FUNCTION REQUESTED OR INVALID DRIVE; SET BAD COMMAND IN
499 ; STATUS.
500
501 ; ON EXIT:  @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
502
503 016A
504 016A B8 C6               FNC_ERR PROC NEAR
505 016C B4 01               MOV   AH,SI ; RESTORE AL
506 016E 88 26 0041 R       MOV   AH,BAD_CMD ; SET BAD COMMAND ERROR
507 0172 F9                 MOV   @DSKETTE_STATUS,AH ; STORE IN DATA AREA
508 0173 C3                 STC
509 0174
510
511 ; DISK_PARAMS: READ DRIVE PARAMETERS.
512
513 ; ON ENTRY: DI : DRIVE #
514
515 ; ON EXIT:  CL/[BP] = BITS 7 & 6 HIGH 2 BITS OF MAX CYLINDER
516 ;           BITS 0-5 MAX SECTORS/TRACK
517 ;           CH/[BP+1] = LOW 8 BITS OF MAX CYLINDER
518 ;           BL/[BP+2] = 0
519 ;           BITS 3-0 = VALID CMOS DRIVE TYPE
520
521 ;           BH/[BP+3] = 0
522 ;           DL/[BP+4] = MAX DRIVES INSTALLED (VALUE CHECKED)
523 ;           DH/[BP+5] = MAX HEAD #
524 ;           DI/[BP+6] = OFFSET OF DISK_BASE
525 ;           ES = SEGMENT OF DISK_BASE
526 ;           AX = 0
527
528 ; NOTE : THE ABOVE INFORMATION IS STORED IN THE USERS STACK AT
529 ; THE LOCATIONS WHERE THE MAIN ROUTINE WILL POKE THEM
530 ; INTO THE APPROPRIATE REGISTERS BEFORE RETURNING TO THE
531 ; CALLER.
532 0174
533 0174 81 FF 0080          DISK_PARAMS PROC NEAR
534 0178 72 06               CMP   DI,80H ; CHECK FOR FIXED MEDIA TYPE REQUEST
535
536 ;----- FIXED DISK REQUEST FALL THROUGH ERROR
537
538 017A B8 C6               MOV   AX,SI ; RESTORE AL WITH CALLERS VALUE
539 017C B4 01               MOV   AH,BAD_CMD ; SET BAD COMMAND ERROR IN (AH)

```

```

540 017E F9          STC          ; SET ERROR RETURN CODE
541 017F C3          RET

543 0180             ; DISK_P2:
544 0180 E8 02C8 R   CALL  XLAT_NEW    ; TRANSLATE STATE TO PRESENT ARCH.
545 0180 WORD PTR [BP+2],0
546 0180 AX,FEQUP FLAG
547 0180 01 0010 R   MOV  AX,FEQUP FLAG
548 0180 AND AL,1000001B
549 0180 24 C1       AND AL,1000001B
550 0180 01BD B2 02   MOV  DL,2
551 0180 01BF 3C 41   CMP  AL,01000001B
552 0191 74 06       JZ   DISK_P3    ; KEEP DISKETTE DRIVE BITS
553 0199             ; DISK_P3:
554 0199 FE CA       DEC  DL
555 0199 3C 01       CMP  AL,00000001B
556 0197 75 6A       JNZ  DISK_P8    ; 2 DRIVES INSTALLED?
557 0199             ; IF YES JUMP
558 0199 88 56 04   MOV  [BP+4],DL
559 0199 8C F0 01   CMP  DI,1
560 0199 71 66       MOV  JA,DISK_P9
561 0199 8E 05 01   CALL  WORD PTR [BP+5],1
562 0199 80 72 18   CMOS_TYPE
563 0199 8A C0       OR   AL,AL
564 0199 74 14       JZ   DISK_P4
565 0199 8C 03       CMP  AL,(DR_PTE-DR_PT)/2
566 0199 77 10       JA   DISK_P4
567 01B2 88 46 02   MOV  [BP+2],AL
568 01B5 FE C0       DEC  AL
569 01B7 D0 E0       SHL  AL,1
570 01B9 8A D8       MOV  BH,AL
571 01B9 3C FF       XOR  BH,BH
572 01B0 2E: BB 8F 0214 R MOV  CX,CS:WORD PTR DR_PT[BX]; GET MAX TRACK AND SECTOR
573             ; DISK_P4:
574 01C2             ; DISK_P4:
575 01C2 8A 85 0090 R MOV  AL,0DSK_STATE[DI]    ; LOAD STATE FOR THIS DRIVE
576 01C6 8A 10         TEST AL,0_MED_DET
577 01C6 80 00         JNZ  DISK_P6
578 01CA 80 7E 02 00   CMP  BYTE PTR [BP+2],0
579 01CE 74 37         JZ   DISK_P9
580 01D0 EB 1C         JMP  SHORT DISK_P6
581             ; DISK_P5:
582 01D2             ; DISK_P5:
583 01D2 24 C0       AND  AL,RATE_MSK
584 01D2 2E1 BB 0E 0216 R MOV  CX,WORD PTR CS:DR_PT+2
585 01D9 3C 80       CMP  AL,RATE_250
586 01D9 75 11       JNE  DISK_P6
587             ; DISK_P6:
588 01DD 2E1 BB 0E 0214 R MOV  CX,WORD PTR CS:DR_PT
589 01E2 F6 85 0090 R TEST  0DSK_STATE[DI],TRK_CAPA
590 01E7 74 03         JZ   DISK_P6
591             ; DISK_P7:
592 01E9 2E1 BB 0E 0218 R MOV  CX,WORD PTR CS:DR_PT+4
593 01EE             ; DISK_P6:
594 01EE 89 4E 00     MOV  [BP],CX
595 01EE 80 06 0000 E  LSA  AL,DISK_BASE
596 01F8 89 46 06     MOV  [BP+4],AX
597 01F8 8C C8       MOV  AX,CS
598             ; DISK_P7:
599 01FA 8E C0       MOV  ES,AX
600 01FC E8 02EE R   CALL  XLAT_OLD
601 01FA 33 C0       XOR  AX,AX
602 0201 F8          CLC
603 0202 C3          RET

604             ;----- NO DRIVE PRESENT HANDLER
605             ;----- DISK_P8:
606 0203             ;----- DISK_P9:
607 0203 C6 46 04 00  MOV  BYTE PTR [BP+4],0
608 0207             ;----- DISK_P9:
609 0207             ;----- DR_PT:
610 0207 33 C0       XOR  AX,AX
611 0209 89 46 00     MOV  [BP],AX
612 020C 88 66 05     MOV  [BP+5],AH
613 0209 89 46 06     MOV  [BP+6],AX
614 0212 EB E6       JMP  DISK_PT
615 0214             ;----- DISK_PARMS:
616             ;----- DISK_TYPE:
617             ;----- DRIVE_PARAMETER_TABLE
618             ;----- DR_PT:
619 0214 09 27       DR_PT DB 09H,027H
620 0216 0F 4F       DB 09H,04FH
621 0218 09 4F       DB 09H,04FH
622 = 021A             DR_PTE EQU $
623             ;----- DISK_TYPE:
624             ;----- THIS ROUTINE RETURNS THE TYPE OF MEDIA INSTALLED.
625             ;----- ON ENTRY: DI : DRIVE #
626             ;----- ON EXIT: AH : DRIVE_TYPE, CY=0
627             ;----- DISK_TYPE:
628             ;----- THIS ROUTINE RETURNS THE TYPE OF MEDIA INSTALLED.
629             ;----- ON ENTRY: DI : DRIVE #
630             ;----- ON EXIT: AH : DRIVE_TYPE, CY=0
631 021A             ;----- DISK_TYPE:
632 021A E8 02C8 R   CALL  XLAT_NEW    ; TRANSLATE STATE TO PRESENT ARCH.
633 021A 85 0090 R   MOV  AL,0DSK_STATE[DI]    ; GET PRESENT STATE INFORMATION
634 0221 0A C0       OR   AL,AL
635 0223 74 13       JZ   NO_DRV
636 0225 B4 01       MOV  AH,NOCHGLN
637 0227 A8 01       TEST AL,TRK_CAPA
638 0229 74 02       JZ   DT_BACK
639 0228 BB 02       MOV  AH,CHGLN
640             ;----- DT_BACK:
641 022D 50          PUSH AX
642 022E E8 02EE R   CALL  XLAT_OLD
643 022E 80 00         POP  AX
644 0232 F8          CLC
645 0233 BB DE       MOV  BX,SI
646 0235 8A C3       MOV  AL,BL
647 0237 C3          RET

648             ;----- NO_DRV:
649 0236 32 E4       XOR  AH,AH
650 023A EB F1       JMP  SHORT DT_BACK
651 023C             ;----- DISK_TYPE:
652             ;----- DISK_CHANGE:
653             ;----- THIS ROUTINE RETURNS THE STATE OF THE DISK CHANGE LINE.

```

```

654
655      ; ON ENTRY:  DI : DRIVE #
656
657      ; ON EXIT:   AH : #DSKETTE_STATUS
658      ;          00 - DISK CHANGE LINE INACTIVE, CY = 0
659
660      ;          04 - DISK CHANGE LINE ACTIVE, CY = 1
661
662      023C E8 02C8 R
663      023F 8A 85 0090 R
664      0243 0A C0
665      0245 74 19
666      0247 A8 01
667      0249 74 05
668
669      024B E8 08E3 R
670      024E T4 05
671
672      0250 C6 06 0041 R 06
673
674      0255 E8 02EE R
675      0258 E8 0620 R
676      025B 8B DE
677      025D 8A C3
678      025F C3
679
680      0260
681      0260 80 0E 0041 R 80
682      0265 E8 EE
683      0267
684
685      ; FORMAT_SET : THIS ROUTINE IS USED TO ESTABLISH THE TYPE OF MEDIA TO BE USED
686      ; FOR THE FOLLOWING FORMAT OPERATION.
687
688      ; ON ENTRY:   SI LOW : DASD TYPE FOR FORMAT
689      ;          DI : DRIVE #
690
691      ; ON EXIT:    #DSKETTE_STATUS REFLECTS STATUS
692      ;          AH : #DSKETTE_STATUS
693      ;          CY = 1 IF ERROR
694
695      0267
696      0267 E8 02C8 R
697      026A 56
698      026B 8B C6
699      026D 32 E4
700      026F 8B F0
701      0271 80 A5 0090 R 0F
702      0271 80 A5 0090 R 0F
703      0271 15 07
704      0279 80 8D 0090 R 90
705      027E EB 37
706
707      0280
708      0280 E8 0416 R
709      0283 80 3E 0041 R 80
710      0288 74 2D
711
712      028A 4E
713      028B 75 07
714      028D 80 8D 0090 R 70
715      0292 EB 23
716
717      0294
718      0294 4E
719      0295 75 01
720      0297 80 8D 0090 R 10
721      029C EB 19
722
723      029E
724      029E 4E
725      029F 75 20
726
727      02A1 F6 85 0090 R 04
728      02A6 74 09
729      02AB B0 50
730      02AA F6 85 0090 R 02
731      02AF 75 02
732
733      02B1
734      02B1 B0 90
735
736      02B3
737      02B3 08 85 0090 R
738
739      02B7
740      02B7 E8 02EE R
741      02B8 E8 0620 R
742      02BD 5B
743      02BE 8A C3
744      02C0 C3
745
746      02C1
747      02C1 C6 06 0041 R 01
748      02C6 EB EF
749
750      02C8
751
752      ; XLAT_NEW: TRANSLATES DISKETTE STATE LOCATIONS FROM COMPATIBLE MODE TO
753      ;          NEW ARCHITECTURE.
754
755      ; ON ENTRY:   DI : DRIVE
756
757      ; XLAT_NEW: PROC  NEAR
758      ;          CMP  DI,_I          ; VALID DRIVE ?
759      ;          JA   XN_OUT        ; IF INVALID BACK
760      ;          CMP  #DSK_STATE[DI],0 ; NO DRIVE ?
761      ;          JZ   DO_DET        ; IF NO DRIVE ATTEMPT DETERMINE
762      ;          MOV  CX,_1          ; CX = DRIVE NUMBER
763      ;          SHL  CL,_2          ; CL = SHIFT COUNT, A=0, B=4
764      ;          MOV  AL,OHF_CNTRL   ; DRIVE INFORMATION
765      ;          ROR  AL,CL           ; TO LOW NIBBLE
766      ;          AND  AL,DRV_DET+FMT_CAPA+TRK_CAPA ; KEEP DRIVE BITS
767      ;          AND  #DSK_STATE[DI],NOT DRV_DET+FMT_CAPA+TRK_CAPA

```

```

768 02E5 08 65 0090 R      OR      #DSK_STATE[DI],AL      ; UPDATE DRIVE STATE
769 02E9                          XN_OUT:  RET
770 02E9 C3
771
772 02E4                          DO_DET:  CALL    DRIVE_DET      ; TRY TO DETERMINE
773 02E4 E8 08ED R      RET
774 02ED C3
775
776 02EE                          XLAT_NEW: ENDP
777
778 ;----- ; XLAT_OLD : TRANSLATES DISKETTE STATE LOCATIONS FROM NEW ARCHITECTURE TO
779 ;----- ; COMPATIBLE MODE.
780
781 ;----- ; ON ENTRY: DI : DRIVE
782
783 02EE                          XLAT_OLD: PROC  NEAR
784 02EE 83 FF 01      CMP    DI,1      ; VALID DRIVE ?
785 02F1 77 68      JA     XO_OUT      ; IF INVALID BACK
786 02F3 80 BD 0090 R 00      CMP    #DSK_STATE[DI].0      ; NO DRIVE ?
787 02F8 74 61      JZ     XO_OUT      ; IF NO DRIVE TRANSLATE DONE
788
789 ;----- ; TEST FOR SAVED DRIVE INFORMATION ALREADY SET
790
791 02FA 8B CF      MOV    CX,DI      ; CX = DRIVE NUMBER
792 02FC C0 E1 02      SHL    CL,2      ; CL = SHIFT COUNT, A=0, B=4
793 02FF B4 02      MOV    AH,FMT_CAPA      ; LOAD MULTIPLE DATA RATE BIT MASK
794 0301 D2 C2      ROR    AH,CL      ; ROTATE BY MASK
795 0303 84 26 008F R      TEST   #HF_CNTL,AH      ; MULTIPLE-DATA RATE DETERMINED ?
796 0307 75 16      JNZ    SAVE_SET      ; IF SO, NO NEED TO RE-SAVE
797
798 ;----- ; ERASE DRIVE BITS IN #HF_CNTL FOR THIS DRIVE
799
800 0309 B4 07      MOV    AH,DRV_DET+FMT_CAPA+TRK_CAPA      ; MASK TO KEEP
801 030B D2 C2      ROR    AH,CL      ; FIX MASK TO KEEP
802 030D F6 D4      NOT    AH      ; TRANSLATE MASK
803 030F 20 26 008F R      AND    #HF_CNTL,AH      ; KEEP BITS FROM OTHER DRIVE INTACT
804
805 ;----- ; ACCESS CURRENT DRIVE BITS AND STORE IN #HF_CNTL
806
807 0313 8A 85 0090 R      MOV    AL,#DSK_STATE[DI]      ; ACCESS STATE
808 0317 24 07      AND    AL,DRV_DET+FMT_CAPA+TRK_CAPA      ; KEEP DRIVE BITS
809 0319 D2 C8      ROR    AL,CL      ; FIX FOR THIS DRIVE
810 031B 08 06 008F R      OR     #HF_CNTL,AL      ; UPDATE SAVED DRIVE STATE
811
812
813 031F 8A A5 0090 R      SAVE_SET: PROC  NEAR
814 031F 8A FC      MOV    AH,#DSK_STATE[DI]      ; ACCESS STATE
815 0323 8A FC      MOV    BH,AH      ; TO BH FOR LATER
816 0325 80 E4 C0      AND    AH,RATE_MSK      ; KEEP ONLY RATE
817 0328 B0 02 00      MOV    AL,MIDIU      ; AL = 1.2 IN 1.2 UNESTABLISHED
818 032A 80 FC 00      CMP    AH,RATE_500      ; RATE 500
819 032C 80 FC 00      TEST   BH,TRK_STEP      ; JUMP 360 IN 1.2
820 032E 74 01      JZ     TST_DET      ; JUMP 360 IN 1.2
821 032F B0 01      MOV    AL,MIDIU      ; AL = 360 IN 1.2 UNESTABLISHED
822 0331 80 FC 40      CMP    AH,RATE_300      ; RATE 300 ?
823 0334 75 09      JNZ    CHK_250      ; IF SO FALL THRU
824 0336 F6 C7 20      TEST   BH,DBL_STEP      ; CHECK FOR DOUBLE STEP
825 0339 75 10      JNZ    TST_DET      ; MUST BE 360 IN 1.2
826
827 033B
828 033B B0 07      UNKNO:  MOV    AL,MED_UNK      ; NONE OF THE ABOVE
829 033D EB 13      JMP    SHORT_AL_SET      ; PROCESS COMPLETE
830
831 033F
832 033F B0 00      CHK_250:  MOV    AL,MIDIU      ; AL = 360 IN 360 UNESTABLISHED
833 0341 80 FC 80      CMP    AH,RATE_250      ; RATE 250 ?
834 0344 75 F5      JNZ    UNKNO      ; IF SO FALL THRU
835 0346 F6 C7 01      TEST   BH,TRK_CAPA      ; 80 TRACK CAPABILITY ?
836 0349 75 F0      JNZ    UNKNO      ; IF SO JUMP, FALL THRU TEST DET
837
838 034B
839 034B F6 C7 10      TST_DET:  TEST   BH,MED_DET      ; DETERMINED ?
840 034E 74 02      JZ     AL_SET      ; IF NOT THEN SET
841 0350 04 03      ADD    AL,3      ; MAKE DETERMINED/ESTABLISHED
842
843 0352
844 0352 80 A5 0090 R F8      AL_SET:  AND    #DSK_STATE[DI],NOT DRV_DET+FMT_CAPA+TRK_CAPA      ; CLEAR DRIVE
845 0357 08 85 0090 R      OR     #DSK_STATE[DI],AL      ; REPLACE WITH COMPATIBLE MODE
846 035B
847 035B C3
848 035C
849
850
851 ;----- ; RD_WV_VF : COMMON READ, WRITE AND VERIFY; MAIN LOOP FOR STATE RETRIES.
852
853 ;----- ; ON ENTRY: AH : READ/WRITE/VERIFY DMA PARAMETER
854 ;----- ; AL : READ/WRITE/VERIFY NEC PARAMETER
855
856 ;----- ; ON EXIT: #DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
857 035C
858 035C 50      RD_WV_VF: PROC  NEAR
859 035D E8 02C8 R      PUSH   AX      ; SAVE DMA, NEC PARAMETERS
860 0360 E8 039B R      CALL    XLAT_NEW      ; TRANSLATE STATE TO PRESENT ARCH.
861 0363 58      CALL    SETUP_STATE      ; STATE INITIALIZATIONS
862
863
864 0364
865 0365 E8 0416 R      POP    AX      ; RESTORE DMA,NEC PARAMETERS
866 0368 58
867 0369 72 21      DO AGAIN: JC     RWV_END      ; MEDIA CHANGE AND RESET IF CHANGED
868 036C 58
869 036C E8 0451 R      PUSH   AX      ; RESTORE READ/WRITE/VERIFY
870 036F E8 063A R      CALL    SEND_RATE      ; MEDIA CHANGE ERROR OR TIME-OUT
871 0372 72 12      CALL    SETUP_DBL      ; SAVE READ/WRITE/VERIFY PARAMETER
872 0374 58      JC     CHK_RET      ; SETUP_DBL RATE = NEC
873 0375 50      POP    AX      ; CHECK FOR DOUBLE STEP
874 0379 E8 0471 R      PUSH   AX      ; ERROR FROM READ ID, POSSIBLE RETRY
875 0379 58      CALL    DMA_SETUP      ; RESTORE DMA
876 037A 52 16      JC     RWV_BAC      ; SET THE DMA
877 037C 50      PUSH   AX      ; RESTORE NEC COMMAND
878 037D E8 04C7 R      CALL    NEC_INIT      ; CHECK FOR DMA BOUNDARY ERROR
879 0380 E8 04EC R      CALL    RWV_COM      ; SAVE NEC COMMAND
880 0383 E8 0530 R      CALL    NEC_TERM      ; INITIALIZE NEC
881

```

```

882 0366
883 0366 EA 05B1 R
884 0369 55
885 038A 72 D8
886
887 038C
888 038C E8 05B2 R
889 039F E8 05F3 R
890
891 0392
892 0392 50
893 0392 80 02EE R
894 0396 58
895 0397 E8 0620 R
896 039A C3
897 039B RD_WF_VF ENDP
898
899
900
901 039B
902 039B F6 85 0090 R 10
903 03A0 75 2B
904 03A2 BB 4080
905 03A5 F6 85 0090 R 04
906
907 03A0 B0 00
908 03AE F6 85 0090 R 02
909 03B3 75 03
910 03B8 BB 8080
911
912 03B8
913 03B8 80 A5 0090 R IF
914 03B0 80 A5 0090 R
915 03C1 80 26 008B R F3
916 03C3 C0 CB 04
917 03C9 08 06 008B R
918 03CD
919 03C3 C3
920 03CE
921
922
923
924 03CE
925 03CE F6 85 0090 R 10
926 03D3 75 3C
927 03D5 E8 06B3 R
928 03D6 72 38
929 03DA FE C8
930 03DC 78 34
931 03E0 80 00 0090 R
932 03E2 80 E4 0F
933 03E5 04 C0
934 03E7 75 05
935 03E9 80 CC 90
936 03E9 EB 1F
937
938 03EE
939 03EE FE C8
940 03F0 75 05
941 03F2 80 CC 10
942 03F5 EB 16
943
944 03F7
945 03F7 FE C8
946 03F9 75 17
947 03FE F6 C4 04
948 03FE T4 0A
949 0400 F6 C4 02
950 0400 80 CC 05
951 0405 80 CC 50
952 0408 EB 03
953
954 040A
955 040A 80 CC 90
956
957 040D
958 040D 88 A5 0090 R
959
960 0411
961 0411 C3
962
963 0412
964 0412 32 E4
965 0414 EB F7
966 0416
967
968
969
970
971
972
973
974 0416
975 0416 E8 08E3 R
976 0419 74 34
977 041B 80 A5 0090 R EF
978
979
980
981
982
983 0420 8B CF
984 0422 B0 01
985 0424 D2 E0
986 0426 80 D0
987 0428 FA
988 0429 20 06 003F R
989 042D FB
990 042E 80 06E1 R
991
992
993
994 0431 E8 007E R
995 0434 B5 01

```

```

996 0436 E8 0TDE R CALL SEEK : ISSUE SEEK
997 0439 32 ED XOR CH,CH : MOVE TO CYLINDER 0
998 043B E8 0TDE R CALL SEEK : ISSUE SEEK
999 043E C6 06 0041 R 06 MOV #DSKETTE_STATUS,MEDIA_CHANGE ; STORE IN STATUS
1000
1001 0443 E8 08E3 R CALL READ_DSKCHNG ; CHECK MEDIA CHANGED AGAIN
1002 0446 T4 05 JZ MED_C8 ; IF ACTIVE, NO DISKETTE, TIMEOUT
1003
1004 0448 C6 06 0041 R 80 MOV #DSKETTE_STATUS,TIME_OUT; TIMEOUT IF DRIVE EMPTY
1005 044D MED_C8: MED_C9: MED_CHANGE: MED_CHANGE: ENDP
1006 044D F9 STC : MEDIA CHANGED, SET CY
1007 044E C3 RET
1008 044F C3 CMP CLC : NO MEDIA CHANGED, CLEAR CY
1009 044F F8 RET
1010 0450 C3
1011 0451
1012
1013
1014
1015 0451 SEND_RATE: PROC NEAR
1016 0451 8A 26 008B R MOV AH, #PLASTRATE
1017 0455 8A 85 0090 R MOV AL, #DSK_STATE[D1] ; GET LAST DATA RATE SELECTED
1018 0459 25 C0C0 AND AX, SEND_MSK*X ; GET RATE STATE OF THIS DRIVE
1019 0459 3A C4 CMP AL, AH ; KEEP ONLY RATE BITS OF BOTH
1020 045E 74 10 JE C_5_OUT ; COMPARE TO PREVIOUSLY TRIED
1021
1022 0460 80 26 008B R 3F AND #LASTRATE,NOT SEND_MSK ; IF SAME, NO NEW TRANSFER RATE
1023 0465 08 06 008B R OR #LASTRATE,AL ; ELSE CLEAR LAST RATE ATTEMPTED
1024 0469 C0 C0 02 ROL AL, 2 ; SAVE NEW RATE FOR NEXT CHECK
1025 046C BA 03F7 MOV DX, 03F7H ; MOVE TO BIT OUTPUT POSITIONS
1026 046F EE OUT DX, AL ; OUTPUT NEW DATA RATE
1027
1028 0470 C_5_OUT: RET
1029 0470 C3 SEND_RATE: ENDP
1030 0471
1031
1032
1033
1034
1035
1036
1037
1038 0471 DMA_SETUP: PROC NEAR
1039 0471 FA CL1: OUT DMA+12,AL ; DISABLE INTERRUPTS DURING DMA SET-UP
1040 0472 E6 0C JMP $+2 ; SET THE FIRST/LAST F/F
1041 0474 EB 00 OUT DMA+11,AL ; WAIT FOR I/O
1042 0476 E6 0B OUT DMA+1,AL ; OUTPUT THE MODE BYTE
1043 0478 BC 00 MOV AX,ES ; GET THE ES VALUE
1044 0479 00 C0 04 ROL AX,4 ; ROTATE LEFT
1045 047A 8A 08 MOV AL, CH ; GET HIGH NIBBLE OF ES TO CH
1046 047F 24 F0 AND AL, 1111000B ; ZERO THE LOW NIBBLE FROM SEGMENT
1047 0481 03 46 02 ADD AX, [BP+2] ; TEST FOR CARRY FROM ADDITION
1048 0484 73 02 JNC J33 ; INC CH
1049 0486 FE C5 INC CH ; CARRY MEANS HIGH 4 BITS MUST BE INC
1050
1051 0488 50 J33: PUSH AX ; SAVE START ADDRESS
1052 0489 E6 04 OUT DMA+4,AL ; OUTPUT LOW ADDRESS
1053 048B EB 00 JMP $+2 ; WAIT FOR I/O
1054 048D 8A C4 MOV AL, AH ; OUTPUT HIGH ADDRESS
1055 048E E6 04 OUT DMA+4,AL ; GET HIGH 4 BITS
1056 0491 00 05 MOV AX, CH ; I/O WAIT STATE
1057 0493 EB 00 JMP $+2 ; OUTPUT HIGH 4 BITS TO PAGE REGISTER
1058 0495 24 0F AND AL, 00000111B
1059 0497 E6 81 OUT 081H,AL
1060
1061
1062
1063 0499 B8 C6 MOV AX, $1 ; DETERMINE COUNT
1064 049B 86 C4 XCHG AL, AH ; AH = # OF SECTORS
1065 049D 2A C0 SUB AL, AL ; AL = 0, AX = # OF SECTORS * 256
1066 049F D1 E8 SHR AX, 1 ; AX = # SECTORS * 128
1067 04A1 50 PUSH AX ; SAVE # OF SECTORS * 128
1068 04A2 00 03 MOV DL, 3 ; GET BYTES/SECTOR PARAMETER
1069 04A4 E8 06CC R CALL CL, PARM ; * 128 = 16384
1070 04A7 8A CC MOV CL, AH ; SHIFT COUNT (0x128 = 1256 ETC)
1071 04A9 58 POP AX ; AX = # OF SECTORS * 128
1072 04AA D3 E0 SHL AX, CL ; SHIFT BY PARAMETER VALUE
1073 04AC 48 DEC AX ; -1 FOR DMA VALUE
1074 04B0 00 PUSH AX ; SAVE COUNT VALUE
1075 04AE E6 05 OUT DMA+5,AL ; LOW BYTE OF COUNT
1076 04B0 EB 00 JMP $+2 ; WAIT FOR I/O
1077 04B2 8A C4 MOV AL, AH ; HIGH BYTE OF COUNT
1078 04B4 E6 05 OUT DMA+5,AL ; RE-ENABLE INTERRUPTS
1079 04B6 FB ST1: POP CX ; RECOVER COUNT VALUE
1080 04B8 59 POP AX ; ADD COUNT TO ADDRESS VALUE
1081 04B8 00 ADD AX, CX ; ADD TEST FOR 64K OVERFLOW
1082 04B9 03 C1 MOV AL, 2 ; MODE FOR 8237
1083 04BB B0 02 MOV AL, 2 ; INITIALIZE THE DISKETTE CHANNEL
1084 04BD E6 0A OUT DMA+10,AL ; MODE FOR 8237
1085 04BF T3 05 JNC NO_BAD ; CHECK FOR ERROR
1086 04C0 C6 06 0041 R 09 MOV #DSKETTE_STATUS,DMA_BOUNDARY ; SET ERROR
1087 04C4 NO_BAD: RET ; CY SET BY ABOVE IF ERROR
1088 04C6 C3 DMA_SETUP: ENDP
1089 04C7
1090
1091
1092
1093
1094
1095
1096
1097
1098 04C7 NEC_INIT: NEC_INIT: ENDP
1099 04C7 50 PUSH AX ; NEC INIT
1100 04C8 E8 06E1 R CALL MOTOR_ON ; SAVE NEC COMMAND
1101
1102
1103
1104 04CB 8A 6E 01 MOV CH, [BP+1] ; TURN MOTOR ON FOR SPECIFIC DRIVE
1105 04CE E8 07DE R CALL SEEK ; CH = TRACK #
1106 04D1 58 POP AX ; MOVE TO CORRECT TRACK
1107 04D2 72 17 JC ER_1 ; RECOVER COMMAND
1108 04D4 BB 04EB R MOV BX, OFFSET ER_1 ; ERROR ON SEEK
1109 04D7 53 PUSH BX ; LOAD ERROR ADDRESS
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
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
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
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
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
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
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
18010
18011
18012
18013
18014
18015
18016
18017
18018
18019
18020
18021
18022
18023
18024
18025
18026
18027
18028
18029
18030
18031
18032
18033
18034
18035
18036
18037
18038
18039
18040
18041
18042
18043
18044
18045
18046
18047
18048
18049
18050
18051
18052
18053
18054
18055
18056
18057
18058
18059
18060
18061
18062
18063
18064
18065
18066
18067
18068
18069
18070
18071
18072
18073
18074
18075
18076
18077
18078
18079
18080
18081
18082
18083
18084
18085
18086
18087
18088
18089
18090
18091
18092
18093
18094
18095
18096
18097
18098
18099
180100
180101
180102
180103
180104
180105
180106
180107
180108
180109
180110
180111
180112
180113
180114
180115
180116
180117
180118
180119
180120
180121
180122
180123
180124
180125
180126
180127
180128
180129
180130
180131
180132
180133
180134
180135
180136
180137
180138
180139
180140
180141
180142
180143
180144
180145
180146
180147
180148
180149
180150
180151
180152
180153
180154
180155
180156
180157
180158
180159
180160
180161
180162
180163
180164
180165
180166
180167
180168
180169
180170
180171
180172
180173
180174
180175
180176
180177
180178
180179
180180
180181
180182
180183
180184
180185
180186
180187
180188
180189
180190
180191
180192
180193
180194
180195
180196
180197
180198
180199
180200
180201
180202
180203
180204
180205
180206
180207
180208
180209
180210
180211
180212
180213
180214
180215
180216
180217
180218
180219
180220
180221
180222
180223
180224
180225
180226
180227
180228
180229
180230
180231
180232
180233
180234
180235
180236
180237
180238
180239
180240
180241
180242
180243
180244
180245
180246
180247
180248
180249
180250
180251
180252
180253
180254
180255
180256
180257
180258
180259
180260
180261
180262
180263
180264
180265
180266
180267
180268
180269
180270
180271
180272
180273
180274
180275
180276
180277
180278
180279
180280
180281
180282
180283
180284
180285
180286
180287
180288
180289
180290
180291
180292
180293
180294
180295
180296
180297
180298
180299
180300
180301
180302
180303
180304
180305
180306
180307
180308
180309
180310
180311
180312
180313
180314
180315
180316
180317
180318
180319
180320
180321
180322
180323
180324
180325
180326
180327
180328
180329
180330
180331
180332
180333
180334
180335
180336
180337
180338
180339
180340
180341
180342
180343
180344
180345
180346
180347
180348
180349
180350
180351
180352
180353
180354
180355
180356
180357
180358
180359
180360
180361
180362
180363
180364
180365
180366
180367
180368
180369
180370
180371
180372
180373
180374
180375
180376
180377
180378
180379
180380
180381
180382
180383
180384
180385
180386
180387
180388
180389
180390
180391
180392
180393
180394
180395
180396
180397
180398
180399
180400
180401
180402
180403
180404
180405
180406
180407
180408
180409
180410
180411
180412
180413
180414
180415
180416
180417
180418
180419
180420
180421
180422
180423
180424
180425
180426
180427
180428
180429
180430
180431
180432
180433
180434
180435
180436
180437
180438
180439
180440
180441
180442
180443
180444
180445
180446
180447
180448
180449
180450
180451
180452
180453
180454
180455
180456
180457
180458
180459
180460
180461
180462
180463
180464
180465
180466
180467
180468
180469
180470
180471
180472
180473
180474
180475
180476
180477
180478
180479
180480
180481
180482
180483
180484
180485
180486
180487
180488
180489
180490
180491
180492
180493
180494
180495
180496
180497
180498
180499
180500
180501
180502
180503
180504
180505
180506
180507
180508
180509
180510
180511
180512
180513
180514
180515
180516
180517
180518
180519
180520
180521
180522
180523
180524
180525
180526
180527
180528
180529
180530
180531
180532
180533
180534
180535
180536
180537
180538
180539
180540
180541
180542
180543
180544
180545
180546
180547
180548
180549
180550
180551
180552
180553
180554
180555
180556
180557
180558
180559
180560
180561
180562
180563
180564
180565
180566
180567
180568
180569
180570
180571
180572
180573
180574
180575
180576
180577
180578
180579
180580
180581
180582
180583
180584
180585
180586
180587
180588
180589
180590
180591
180592
180593
180594
180595
180596
180597
180598
180599
180600
180601
180602
180603
180604
180605
180606
180607
180608
180609
180610
180611
180612
180613
180614
180615
180616
180617
180618
180619
180620
180621
180622
180623
180624
180625
180626
180627
180628
180629
180630
180631
180632
180633
180634
180635
180636
180637
180638
180639
180640
180641
180642
180643
180644
180645
180646
180647
180648
180649
180650
180651
180652
180653
180654
180655
180656
180657
180658
180659
180660
180661
180662
180663
180664
180665
180666
180667
180668
180669
180670
180671
180672
180673
180674
180675
180676
180677
180678
180679
180680
180681
180682
180683
180684
180685
180686
180687
180688
180689
180690
180691
180692
180693
180694
180695
180696
180697
180698
180699
180700
180701
180702
180703
180704
180705
180706
180707
180708
180709
180710
180711
180712
180713
180714
180715
180716
180717
180718
180719
180720
180721
180722
180723
180724
180725
180726
180727
180728
180729
180730
180731
180732
180733
180734
180735
180736
180737
180738
180739
180740
180741
180742
180743
180744
180745
180746
180747
180748
180749
180750
180751
180752
180753
180754
180755
180756
180757
180758
180759
180760
180761
180762
180763
180764
180765
180766
180767
180768
180769
180770
180771
180772
180773
180774
180775
180776
180777
180778
180779
180780
180781
180782
180783
180784
180785
180786
180787
180788
180789
180790
180791
180792
180793
180794
180795
180796
180797
180798
180799
180800
180801
180802
180803
180804
180805
180806
180807
180808
180809
180810
180811
180812
180813
180814
180815
180816
180817
180818
180819
180820
180821
180822
180823
180824
180825
180826
180827
180828
180829
180830
180831
180832
180833
180834
180835
180836
180837
180838
180839
180840
180841
180842
180843
180844
180845
180846
180847
180848
180849
180850
180851
180852
180853
180854
180855
180856
180857
180858
180859
180860
180861
180862
180863
180864
180865
180866
180867
180868
180869
180870
180871
180872
180873
180874
180875
180876
180877
180878
180879
180880
180881
180882
180883
180884
180885
180886
180887
180888
180889
180890
180891
180892
180893
180894
180895
180896
180897
180898
180899
180900
180901
180902
180903
180904
180905
180906
180907
180908
180909
180910
180911
180912
180913
180914
180915
180916
180917
180918
180919
180920
180921
18092
```

```

1110
1111
1112 ;----- SEND OUT THE PARAMETERS TO THE CONTROLLER
1113 04D8 E8 07BD R CALL NEC_OUTPUT ; OUTPUT THE OPERATION COMMAND
1114 04D9 E8 C6 MOV AX,51 ; AH = HEAD #
1115 04D0 E8 FF MOV BX,DI ; BX = DRIV #
1116 04D9 CO E4 02 SAL AH,2 ; MOVE AH TO BIT 2
1117 04E2 80 E4 04 AND AH,00000100B ; ISOLATE THAT BIT
1118 04E5 04 E3 OR AH,BL ; OR IN THE DRIVE NUMBER
1119 04E7 E8 07BD R CALL NEC_OUTPUT ; FALL THRU CY SET IF ERROR
1120 04E4 5B POP BX ; THROW AWAY ERROR RETURN
1121 04EB C3 ER_1: RET
1122 04EC
1123
1124 ;----- NEC_INIT: ENDP
1125 ;----- RWY_COM: THIS ROUTINE SENDS PARAMETERS TO THE NEC SPECIFIC TO THE
1126 ;----- READ/WRITE/VERIFY OPERATIONS.
1127 ;----- ON EXIT: @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
1128 ;----- NEAR
1129 RWY_COM PROC NEAR
1130 04EC BB 052F R MOV AX,OFFSET ER_2 ; LOAD ERROR ADDRESS
1131 04E4 50 PUSH AX ; PUSH NEC OUT ERROR RETURN
1132 04F0 8A 66 01 MOV AH,[BP+1] ; OUTPUT TRACK #
1133 04F0 8A 07BD R CALL NEC_OUTPUT
1134 04F5 BB 05 CALL NEC_OUTPUT ; OUTPUT HEAD #
1135 04F6 BB 05 MOV AX,51 ; BYTES/SECTOR PARAMETER FROM BLOCK
1136 04F8 E8 07BD R CALL GET_PARM ; TO THE NEC
1137 04F4 8A 66 00 NEC_OUTPUT ; OUTPUT TO CONTROLLER
1138 04F6 E8 07BD R CALL ED_PARM ; ED_PARM FROM BLOCK
1139 0501 B2 03 MOV AX,4 ; TO THE NEC
1140 0501 E8 06CC R CALL GET_PARM ; OUTPUT TO CONTROLLER
1141 0506 80 E0 00 CALL NEC_OUTPUT ; GET DRIVE STATE VALUE
1142 0509 B2 04 MOV AL,_DSK_STATE[DI] ; 1.2/1.2 DRIVE GAP LENGTH
1143 0509 E8 06CC R CALL GET_PARM ; STRIP OFF HIGH BITS
1144 050E E8 07BD R CALL NEC_OUTPUT ; 320,360/1.2 DRIVE GAP LENGTH
1145 0511 8A 85 0090 R MOV AL,_01BH ; CHECK FOR 320 MEDIA IN 1.2 DRIVE
1146 0511 B4 1B AND AL,_RATE_MSK ; IF SO JUMP
1147 0511 80 E0 00 JZ R15 ; STRIP OFF HIGH BITS
1148 0519 74 28 MOV AL,023H ; 320,360/1.2 DRIVE GAP LENGTH
1149 051B B4 23 DEC AL ; CHECK FOR 320 MEDIA IN 1.2 DRIVE
1150 051D FE C8 DEC AL ; IF SO JUMP
1151 051F 74 02 JZ R15 ; STRIP OFF HIGH BITS
1152
1153 0521 B4 2A MOV AH,02AH ; 360/360 DRIVE GAP LENGTH
1154 0523
1155 0523 E8 07BD R R15: CALL NEC_OUTPUT ; DTL PARAMETER FROM BLOCK
1156 0524 B2 06 MOV DL,6 ; TO THE NEC
1157 0528 E8 06CC R CALL GET_PARM ; OUTPUT TO CONTROLLER
1158 0528 E8 07BD R CALL NEC_OUTPUT ; THROW AWAY ERROR EXIT
1159 052E 58 POP AX
1160 052F C3 ER_2: RET
1161 0530
1162 ;----- RWY_COM ENDP
1163 ;----- NEC_TERM: THIS ROUTINE WAITS FOR THE OPERATION THEN ACCEPTS THE STATUS
1164 ;----- FROM THE NEC FOR THE READ/WRITE/VERIFY/FORMAT OPERATION.
1165 ;----- ON EXIT: @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
1166 ;----- NEAR
1167 NEC_TERM PROC NEAR
1168 ;----- LET THE OPERATION HAPPEN
1169 0530 05030
1170 ;----- PUSH SI ; SAVE HEAD #, # OF SECTORS
1171 ;----- CALL WAIT_INT ; WAIT FOR THE INTERRUPT
1172 ;----- PUSHF
1173 0530 56 LODS @NEC_STATUS ; POINT TO STATUS FIELD
1174 0531 E8 087C R CALL RESULTS ; GET STO
1175 0534 9C JC SET_END_POP ; GET THE NEC STATUS
1176 0535 E8 0844 R CALL SET_END ; TEST FOR NORMAL TERMINATION
1177 0538 72 45 JC POPF ; NOT ABNORMAL, BAD NEC
1178 053A 9D JC SET_END ; LOOK FOR ERROR
1179 053B 72 3A
1180
1181 ;----- CHECK THE RESULTS RETURNED BY THE CONTROLLER
1182 ;----- CLD ; SET THE CORRECT DIRECTION
1183 053D FC MOV SI,OFFSET @NEC_STATUS ; POINT TO STATUS FIELD
1184 053E BE 0042 R LODS @NEC_STATUS ; GET STO
1185 053F D0 E0 JC SET_END_POP ; TEST FOR EOT FOUND
1186 0542 C0 AND AL,_1000000B ; TEST FOR NORMAL TERMINATION
1187 0544 72 31 JC SET_END ; TEST FOR ABNORMAL TERMINATION
1188 0546 3C 40 CMP AL,_0100000B ; NOT ABNORMAL, BAD NEC
1189 0546 75 27 JNZ J18 ; TEST FOR DMA OVERRUN
1190
1191 ;----- ABNORMAL TERMINATION, FIND OUT WHY
1192 ;----- LODS @NEC_STATUS ; GET ST1
1193 0544 AC JC SET_END ; TEST FOR EOT FOUND
1194 0548 D0 E0 LODS @NEC_STATUS ; TEST FOR EOT FOUND
1195 054F D0 E0 JC SET_END ; TEST FOR EOT FOUND
1196 054F T2 22 JC SET_END ; TEST FOR EOT FOUND
1197 0551 CO E0 02 JC SET_END ; TEST FOR EOT FOUND
1198 0551 B4 10 JC SET_END ; TEST FOR EOT FOUND
1199 0556 D0 E0 JC SET_END ; TEST FOR EOT FOUND
1200 0558 D0 E0 JC SET_END ; TEST FOR EOT FOUND
1201 055A B4 08 JC SET_END ; TEST FOR EOT FOUND
1202 055C T2 15 JC SET_END ; TEST FOR EOT FOUND
1203 055E CO E0 02 JC SET_END ; TEST FOR EOT FOUND
1204 0561 B4 04 JC SET_END ; TEST FOR EOT FOUND
1205 0562 0E 00 JC SET_END ; TEST FOR EOT FOUND
1206 0565 D0 E0 JC SET_END ; TEST FOR EOT FOUND
1207 0567 B4 03 JC SET_END ; TEST FOR EOT FOUND
1208 0569 72 08 JC SET_END ; TEST FOR EOT FOUND
1209 056B D0 E0 JC SET_END ; TEST FOR EOT FOUND
1210 056D B4 02 JC SET_END ; TEST FOR EOT FOUND
1211 056F T2 02 JC SET_END ; TEST FOR EOT FOUND
1212
1213 ;----- NEC MUST HAVE FAILED
1214 0571 J18: MOV AH,BAD_NEC
1215 0571 B4 20 J19: OR @DSKETTE_STATUS,AH
1216 0573 SET_END: CMP @DSKETTE_STATUS,! ; SET ERROR CONDITION
1217 0577 08 26 0041 R CMC
1218 0577 80 3E 0041 R 01 POP SI ; RESTORE HEAD #, # OF SECTORS
1219 0577 80 3E 0041 R 01
1220 057C F5
1221 057D 5E
1222 057E C3 RET

```

```

1224 057F
1225 057F 9D
1226 0580 EB F5
1227 0582
1228
1229 ; DSTATE: ESTABLISH STATE UPON SUCCESSFUL OPERATION.
1230
1231 0582
1232 0582 80 3E 0041 R 00
1233 0582 75 20 0041 R 00
1234 0589 80 8D 0090 R 10
1235 058E F6 85 0090 R 04
1236 0593 75 1B
1237 0595 8A 85 0090 R
1238 0599 2A C0
1239 059C 2C 00
1240 059D 75 0C
1241 059F 80 A5 0090 R FD
1242 05A4 80 8D 0090 R 04
1243 05A9 EB 05
1244
1245 05AB
1246 05AB 80 8D 0090 R 06
1247
1248 05B0
1249 05B0 C3
1250 05B1
1251
1252 ; RETRY: DETERMINES WHETHER A RETRY IS NECESSARY. IF RETRY IS REQUIRED
1253 ; THEN STATE INFORMATION IS UPDATED FOR RETRY.
1254
1255 ; ON EXIT: CY = 1 FOR RETRY, CY = 0 FOR NO RETRY
1256
1257 05B1
1258 05B1 80 3E 0041 R 00
1259 05B6 74 39
1260 05B8 80 3E 0041 R 80
1261 05BD 74 32
1262 05BF 8A A5 0090 R
1263 05C3 2F 10
1264 05C5 15 29
1265 05CA 80 E4 C0
1266 05CB 8A 2E 0008B R
1267 05CF 0C C0 04
1268 05D2 80 E5 C0
1269 05D5 3A EC
1270 05D7 74 18
1271
1272 ; SETUP STATE INDICATOR FOR RETRY ATTEMPT TO NEXT RATE
1273 0000000B (500) -> 1000000B (250)
1274 1000000B (250) -> 0100000B (300)
1275 0100000B (300) -> 0000000B (500)
1276
1277 05D9 80 FC 01
1278 05DC D0 DC
1279 05DE 80 E4 C0
1280 05E1 80 A5 0090 R IF
1281 05E6 0B A5 0090 R
1282 05E6 0C 06 0041 R 00
1283 05EF F9
1284 05F0 C3
1285
1286 05F1
1287 05F1 F8
1288 05F2 C3
1289 05F3
1290 ; NUM_TRANS: THIS ROUTINE CALCULATES THE NUMBER OF SECTORS THAT WERE
1291 ; ACTUALLY TRANSFERRED TO/FROM THE DISKETTE.
1292
1293 ; ON ENTRY: [BP+1] = TRACK
1294 ; SI-HI = HEAD
1295 ; [BP] = START SECTOR
1296
1297 ; ON EXIT: AL = NUMBER ACTUALLY TRANSFERRED
1298
1299 05F3
1301 05F3 32 C0
1302 05F5 80 3E 0041 R 00
1303 05FA 75 23
1304 05FB 2C 04
1305 05FE E0 06CC R
1306 0600 8A 2E 0047 R
1307 0605 3B CE
1308 0607 3A 2E 0046 R
1309 060B 75 0B
1310
1311 060D 8A 2E 0045 R
1312 0611 3A 6E 01
1313 0614 74 04
1314
1315 0616 02 DC
1316 0618
1317 0618 02 DC
1318 061A
1319 061A 2A 5E 00
1320 061D 8A C3
1321
1322 061F
1323 061F C3
1324 0620
1325
1326 ; SETUP_END: RESTORES •MOTOR_COUNT TO PARAMETER PROVIDED IN TABLE AND LOADS
1327 ; •DSKETTE_STATUS TO AH, AND SETS CY.
1328
1329 ; ON EXIT: AH, •DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
1330
1331 0620
1332 0620 B2 02
1333 0622 50
1334 0623 E8 06CC R
1335 0626 88 26 0040 R
1336 062A 58
1337 062B 8A 26 0041 R

```

```

1339 0627 04 E4          OR    AH, AH          ; CHECK FOR ERROR
1339 0631 74 E2          JZ    NUN_ERR        ; NO ERROR
1340 0633 32 C0          XOR    AL, AL          ; CLEAR NUMBER RETURNED
1341

1342 0635               NUN_ERR:
1343 0635 80 FC 01        CMP   AH, 1           ; SET THE CARRY FLAG TO INDICATE
1344 0638 F5              RET
1345 0639 C3              RET
1346 063A
1347               SETUP_END: ENDP
1348               ;----- ; SETUP_DBL: CHECK DOUBLE STEP.
1349               ;----- ; ON ENTRY : AH = RATE; DI = DRIVE
1350               ;----- ; ON EXIT : CY = 1 MEANS ERROR
1351
1352
1353
1354 063A
1355 063A 8A A5 0090 R   SETUP_DBL PROC NEAR
1356 063E F6 C0 10        AH, @DSK_STATE[DI]
1357 0641 75 59          TEST  AH, MED_DET      ; ACCESS STATE
1358          JNZ   NO_DBL        ; ESTABLISHED STATE ?
1359          ;----- ; IF ESTABLISHED THEN DOUBLE DONE
1360
1361 0643 C6 06 003E R 00  MOV   @SEEK_STATUS,0      ; SET RECALIBRATE REQUIRED ON ALL DRIVES
1362 0644 E8 0E51 R        CALL  MOTOR_ON        ; ENSURE MOTOR STAY ON
1363 0649 00 00             MOV   CH, 1           ; LOAD TRACK
1364 064D E8 07DE R        CALL  SEEK            ; SEEK TO TRACK 0
1365 0650 E8 069E R        CALL  READ_ID        ; READ ID FUNCTION
1366 0653 72 32          JC    SD_ERR        ; IF ERROR NO TRACK 0
1367
1368
1369               ;----- ; INITIALIZE START AND MAX TRACKS (TIMES 2 FOR BOTH HEADS)
1370 0655 B9 0450          MOV   CX, 0450H        ; START, MAX TRACKS
1371 0658 F6 85 0090 R 01  TEST  @DSK_STATE[DI], TRK_CAPA ; TEST FOR 80 TRACK CAPABILITY
1372 065D 74 02          JZ    CNT_0K        ; IF NOT COUNT IS SETUP
1373 065F B1 A0          MOV   CL, @0A0H        ; MAXIMUM TRACK 1.2 MB
1374
1375               ;----- ; ATTEMPT READ ID OF ALL TRACKS, ALL HEADS UNTIL SUCCESS; UPON SUCCESS,
1376               ;----- ; MUST SEE IF ASKED FOR TRACK IN SINGLE STEP MODE = TRACK ID READ; IF NOT
1377               ;----- ; THEN SET DOUBLE STEP ON.
1378
1379 0661
1380 0661 51
1381 0662 06 06 0041 R 00  CNT_0K: PUSH CX          ; SAVE TRACK, COUNT
1382 0667 33 C0          XOR   AX, AX          ; CLEAR AX
1383 0669 D0 ED          SHR   CH, 1           ; HALVE TRACK, CY = HEAD
1384 066B C0 D0 03        RCL   AL, 3           ; AX = HEAD IN CORRECT BIT
1385 066E 50
1386 0671 E8 07DE R        PUSH  AX           ; SAVE HEAD
1387 0672 00 00             CALL  SEEK            ; SEEK TO TRACK
1388 0673 08 F8             POP   AX           ; RESTORE HEAD 0
1389 0675 E8 069E R        OR    DI, AX          ; 1 = NEW ORDERED DRIVE
1390 0678 9C
1391 0679 81 E7 00FB        CALL  READ_ID        ; READ ID HEAD 0
1392 067D 9D
1393 0680 59
1394 067F F3 08          AND   DI, 1111011B ; SAVE RETURN FROM READ_ID
1395 0681 FE C5          POPF  CX           ; TURN OFF HEAD 1 BIT
1396 0683 3A E9          INC   CH             ; RESTORE COUNT
1397 0685 75 DA          CMP   CH, CL          ; IF CY = 1 = RETURNED TRACK ?
1398          JNZ   CNT_0K        ; INC FOR NEXT TRACK
1399
1400               ;----- ; FALL THRU, READ ID FAILED FOR ALL TRACKS
1401 0687
1402 0687 F9
1403 0688 C3
1404
1405 0687
1406 0689 8A 0E 0045 R   SD_ERR: STC          ; SET CARRY FOR ERROR
1407 068D 88 AD 0094 R   RET          ; SETUP_DBL ERROR EXIT
1408 0691 D0 ED
1409 0693 3A E9
1410 0695 74 05
1411 0697 80 8D 0090 R 20  DO_CHK: MOV   CL, @NEC_STATUS+3      ; LOAD RETURNED TRACK
1412
1413 069C
1414 069C F8
1415 069D C3
1416 069E
1417
1418
1419
1420
1421
1422
1423
1424
1425 069E
1426 069E BB 06B2 R       NO_DBL: CLC          ; CLEAR ERROR FLAG
1427 06A1 50
1428 06A2 B4 4A
1429 06A3 E8 07BD R
1430 06A7 BB C1
1431 06A9 8A E0
1432 06AB E8 07BD R
1433 06AE E8 0530 R
1434 06B1 58
1435 06B2
1436 06B2 C3
1437 06B3
1438
1439
1440
1441
1442
1443
1444
1445 06B3
1446 06B3 B0 0E
1447 06B8 00 0000 E
1448 06B8 A8 C0
1449 06B8 F9
1450 06B8 75 0E
1451               ;----- ; READ_ID : READ ID FUNCTION.
1452               ;----- ; ON ENTRY: DI : BIT 2 = HEAD; BITS 1,0 = DRIVE
1453               ;----- ; ON EXIT: DI : BIT 2 IS RESET, BITS 1,0 = DRIVE
1454               ;----- ; @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
1455
1456 069E READ_ID PROC NEAR
1457          MOV   AX, OFFSET ER_3      ; MOVE NEC OUTPUT ERROR ADDRESS
1458          PUSH  AX
1459          MOV   AH, 4AH          ; READ ID COMMAND
1460          CALL  NEC_OUTPUT      ; TO CONTROLLER
1461          MOV   AX, D             ; DRIVE # TO AH, HEAD 0
1462          MOV   AL, AL
1463          CALL  NEC_OUTPUT      ; TO CONTROLLER
1464          POP   AX             ; WAIT FOR OPERATION, GET STATUS
1465          ;----- ; THROW AWAY ERROR ADDRESS
1466 069E ER_3: RET
1467 069E READ_ID ENDP
1468
1469               ;----- ; CMOS_TYPE: RETURNS DISKETTE TYPE FROM CMOS
1470               ;----- ; ON ENTRY: DI : DRIVE #
1471               ;----- ; ON EXIT: AL = TYPE (IF VALID) ; CY REFLECTS STATUS
1472
1473 069E CMOS_TYPE PROC NEAR
1474          MOV   AL, @CMOS_DIAG      ; CMOS DIAGNOSTIC STATUS BYTE ADDRESS
1475          CALL  CMOS_READ        ; GET CMOS STATUS
1476          TEST  AL, @BAD_BAT+BAD_CKSUM ; BATTERY GOOD AND CHECKSUM VALID ?
1477          STC
1478          JNZ   CMOS_T9        ; SET CY = 1 INDICATING ERROR FOR RETURN
1479          ;----- ; ERROR EXIT IF EITHER ERROR BIT WAS ON
1480

```

```

1452 06BD B0 10          MOV    AL,CMOS_DISKETTE : ADDRESS OF DISKETTE BYTE IN CMOS
1453 06BF E8 0000 E        CALL   CMOS_READ
1454 06C2 DB FF          OR    CMOS_T5
1455 06C4 T5 03          JNZ   CMOS_T5
1456
1457
1458 06C6 C0 C8 04        ROR   AL,4      : EXCHANGE NIBBLES IF SECOND DRIVE
1459 06C9 CMOS_T5          AND   AL,00FH
1460 06CB CMOS_T9          RET
1461 06CB C3              RET
1462 06CC CMOS_TYPE
1463
1464 :-----: GET_PARM: THIS ROUTINE FETCHES THE INDEXED POINTER FROM THE DISK BASE
1465 :-----:          BLOCK POINTED TO BY THE DATA VARIABLE @DISK_POINTER. A BYTE FROM
1466 :-----:          THAT TABLE IS THEN MOVED INTO AH, THE INDEX OF THAT BYTE BEING
1467 :-----:          THE PARAMETER IN DL.
1468
1469 :-----: ON ENTRY: DL = INDEX OF BYTE TO BE FETCHED
1470 :-----: ON EXIT: AH = THAT BYTE FROM BLOCK
1471 :-----:          AH,DL DESTROYED
1472
1473 :-----: ON EXIT: AH = THAT BYTE FROM BLOCK
1474 :-----:          AH,DL DESTROYED
1475
1476 06CC GET_PARM PROC NEAR
1477 06CC IE              PUSH  DS
1478 06CC 56 00            PUSH  SI
1479 06CC 56 00            SUB   AX,AX
1480 06CC 56 00            MOV   DS,AX
1481 06D0 BE D8            MOV   AX,BX
1482 06D2 87 D3            XCHG  DX,BX
1483 06D4 2A FF            SUB   BH,BH
1484 06D6 C5 36 0078 R      ASSUME DS:AB50
1485 06D6 C5 36 0078 R      LDS   SI, @DISK_POINTER : POINT TO BLOCK
1486 06D4 8A 20            MOV   AH,[SI+BX] : GET THE WORD
1487 06D6 87 D3            XCHG  DX,BX
1488 06D6 5E              POP   SI
1489 06D6 1F              POP   DS
1490 06E0 C3              RET
1491
1492 :-----: ASSUME DS:DATA
1493 :-----: ENDP
1494 :-----: MOTOR_ON : TURN MOTOR ON AND WAIT FOR MOTOR START UP TIME. THE @MOTOR_COUNT
1495 :-----:          IS REPLACED WITH A SUFFICIENTLY HIGH NUMBER (0FFH) TO ENSURE
1496 :-----:          THAT THE MOTOR DOES NOT GO OFF DURING THE OPERATION. IF THE
1497 :-----:          MOTOR NEEDED TO BE TURNED ON, THE MULTI-TASKING HOOK FUNCTION
1498 :-----:          (@X90FDH) IS CALLED TELLING THE OPERATING SYSTEM
1499 :-----:          THAT THE BLOCK IS ABOUT TO BE WRITTEN. ONCE START UP TIME IS
1500 :-----:          RETURNED WITH CY = 1, IT MEANS THAT THE MINIMUM WAIT
1501 :-----:          HAS BEEN COMPLETED. AT THIS POINT A CHECK IS MADE TO ENSURE
1502 :-----:          THAT THE MOTOR WASN'T TURNED OFF BY THE TIMER. IF THE HOOK DID
1503 :-----:          NOT WAIT, THE @WAIT FUNCTION (AH=06H) IS CALLED TO WAIT THE
1504 :-----:          PRESCRIBED AMOUNT OF TIME. IF THE CARRY FLAG IS SET ON RETURN,
1505 :-----:          IT MEANS THAT THE FUNCTION IS IN USE AND DID NOT PERFORM THE
1506 :-----:          WAIT. A TIMER 1 WAIT LOOP WILL THEN DO THE WAIT.
1507
1508 :-----: ON ENTRY: DI = DRIVE #
1509 :-----: ON EXIT: AX,BX,CX,DX DESTROYED
1510 :-----: MOTOR_ON PROC NEAR
1511 06E1 E8 072A R        CALL   TURN_ON_MOTOR : TURN ON MOTOR
1512 06E4 T2 43            JC    MOT_TS_ON : IF CY=1 NO WAIT
1513 06E6 E8 02EE R        CALL   XLAT_OLD : TRANSLATE STATE TO COMPATIBLE MODE
1514 06E9 BB 90FD           MOV   AX,090FDH : LOAD WAIT CODE & TYPE
1515 06E9 00 15            INT   15H : TURN ON WAITING SYSTEM ABOUT TO DO WAIT
1516 06EE 9C              PUSHF : SAVE CY FOR TEST
1517 06EF E8 02C8 R        CALL   XLAT_NEW : TRANSLATE STATE TO PRESENT ARCH.
1518 06F2 9D              POPF  : RESTORE CY FOR TEST
1519 06F3 T3 05            JNC   M_WAIT : BYPASS LOOP IF OS SYSTEM HANDLED WAIT
1520 06F5 E8 072A R        CALL   TURN_ON : CHECK AGAIN IF MOTOR ON
1521 06F6 T2 2F            JC    MOT_TS_ON : IF NO WAIT MEANS IT IS ON
1522
1523 06FA B2 0A            MOV   DL,10 : GET THE MOTOR WAIT PARAMETER
1524 06FB E8 06CC R        CALL   GET_PARM
1525 06FF BA C4            MOV   AL,AH : AL = MOTOR_WAIT PARAMETER
1526 0700 00 00            XOR   AH,AH : AH = MOTOR_WAIT PARAMETER
1527 0701 32 E4            CMP   AL,8 : SEE IF AT LEAST ONE SECOND IS SPECIFIED
1528 0705 T3 02            JAE   GP2 : IF YES, CONTINUE
1529 0707 B0 08            MOV   AL,8 : ONE SECOND WAIT FOR MOTOR START UP
1530
1531 :-----: AX CONTAINS NUMBER OF 1/8 SECONDS (12500 MICROSECONDS) TO WAIT
1532
1533 0709 50              GP2: PUSH  AX : SAVE WAIT PARAMETER
1534 070A B4 F424           MOV   DX,62500 : LOAD LARGEST POSSIBLE MULTIPLIER
1535 070D F7 E2            MUL   DX : MULTIPLY BY HALF OF WHAT'S NECESSARY
1536 070F BB CA            MOV   CX,DX : CX = HIGH WORD
1537 0711 BB D0            MOV   DX,AX : CX,DX = 1/2 * (# OF MICROSECONDS)
1538 0713 F8              CLC   : CLEAR CARRY FOR ROTATE
1539 0714 D1 D2            RCL   DX,I : DOUBLE LOW WORD, CY CONTAINS OVERFLOW
1540 0716 D1 D1            RCL   CX,I : DOUBLE HI, INCLUDING LOW WORD OVERFLOW
1541 0718 B4 86            MOV   AH,86H : LOAD WAIT CODE
1542 071A CD 15            INT   15H : PERFORM WAIT
1543 071C 58              POP   AX : RESTORE WAIT PARAMETER
1544 071D T3 0A            JNC   MOT_IS_ON : CY MEANS WAIT COULD NOT BE DONE
1545
1546 :-----: FOLLOWING LOOPS REQUIRED WHEN RTC WAIT FUNCTION IS ALREADY IN USE
1547
1548 071F B9 205E           J13: MOV   CX,8286 : WAIT FOR 1/8 SECOND PER (AL)
1549 0720 E8 0000 E          CALL   WAITF : COUNT FOR 1/8 SECOND AT 15.085737 US
1550 0722 FE C8            DEC   AL : GO TO FIXED WAIT ROUTINE
1551 0725 T2 76            JNZ   J13 : DECREMENT TIME VALUE
1552 0727 T5 F6            : ARE WE DONE YET
1553
1554 0729 MOT_IS_ON:
1555 0729 C3              RET
1556 072A MOTOR_ON PROC NEAR
1557
1558 :-----: TURN_ON : TURN MOTOR ON AND RETURN WAIT STATE.
1559 :-----: ON_ENTRY: DI = DRIVE #
1560 :-----: ON_EXIT: CY = 0 MEANS WAIT REQUIRED
1561 :-----:          CY = 1 MEANS NO WAIT REQUIRED
1562 :-----:          AX,BX,CX,DX DESTROYED
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
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
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2298
2299
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2398
2399
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2498
2499
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2598
2599
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2698
2699
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2798
2799
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2898
2899
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2998
2999
2999
3000
3001
3002
3003
3004
3005
3006
3007
3008
3009
3009
3010
3011
3012
3013
3014
3015
3016
3017
3018
3019
3019
3020
3021
3022
3023
3024
3025
3026
3027
3028
3029
3029
3030
3031
3032
3033
3034
3035
3036
3037
3038
3039
3039
3040
3041
3042
3043
3044
3045
3046
3047
3048
3049
3049
3050
3051
3052
3053
3054
3055
3056
3057
3058
3059
3059
3060
3061
3062
3063
3064
3065
3066
3067
3068
3069
3069
3070
3071
3072
3073
3074
3075
3076
3077
3078
3079
3079
3080
3081
3082
3083
3084
3085
3086
3087
3088
3089
3089
3090
3091
3092
3093
3094
3095
3096
3097
3098
3098
3099
3099
3100
3101
3102
3103
3104
3105
3106
3107
3108
3109
3109
3110
3111
3112
3113
3114
3115
3116
3117
3118
3119
3119
3120
3121
3122
3123
3124
3125
3126
3127
3128
3129
3129
3130
3131
3132
3133
3134
3135
3136
3137
3138
3139
3139
3140
3141
3142
3143
3144
3145
3146
3147
3148
3149
3149
3150
3151
3152
3153
3154
3155
3156
3157
3158
3159
3159
3160
3161
3162
3163
3164
3165
3166
3167
3168
3169
3169
3170
3171
3172
3173
3174
3175
3176
3177
3178
3179
3179
3180
3181
3182
3183
3184
3185
3186
3187
3188
3189
3189
3190
3191
3192
3193
3194
3195
3196
3197
3198
3198
3199
3199
3200
3201
3202
3203
3204
3205
3206
3207
3208
3209
3209
3210
3211
3212
3213
3214
3215
3216
3217
3218
3219
3219
3220
3221
3222
3223
3224
3225
3226
3227
3228
3229
3229
3230
3231
3232
3233
3234
3235
3236
3237
3238
3239
3239
3240
3241
3242
3243
3244
3245
3246
3247
3248
3249
3249
3250
3251
3252
3253
3254
3255
3256
3257
3258
3259
3259
3260
3261
3262
3263
3264
3265
3266
3267
3268
3269
3269
3270
3271
3272
3273
3274
3275
3276
3277
3278
3279
3279
3280
3281
3282
3283
3284
3285
3286
3287
3288
3289
3289
3290
3291
3292
3293
3294
3295
3296
3297
3298
3298
3299
3299
3300
3301
3302
3303
3304
3305
3306
3307
3308
3309
3309
3310
3311
3312
3313
3314
3315
3316
3317
3318
3319
3319
3320
3321
3322
3323
3324
3325
3326
3327
3328
3329
3329
3330
3331
3332
3333
3334
3335
3336
3337
3338
3339
3339
3340
3341
3342
3343
3344
3345
3346
3347
3348
3349
3349
3350
3351
3352
3353
3354
3355
3356
3357
3358
3359
3359
3360
3361
3362
3363
3364
3365
3366
3367
3368
3369
3369
3370
3371
3372
3373
3374
3375
3376
3377
3378
3379
3379
3380
3381
3382
3383
3384
3385
3386
3387
3388
3389
3389
3390
3391
3392
3393
3394
3395
3396
3397
3398
3398
3399
3399
3400
3401
3402
3403
3404
3405
3406
3407
3408
3409
34
```

```

1566 072A          TURN_ON PROC  NEAR
1567 072C 8B 0F    MOV    BX,DI
1568 072C 8A CB    MOV    CL,BL
1569 072E C0 C3 04  PRL    BL,4
1570 0731 FA        CLI
1571 0732 C6 06 0040 R FF  MOV    *MOTOR_COUNT,0FFH
1572 0737 A0 003F R  MOV    AL,*MOTOR_STATUS
1573 0739 24 30    AND    AL,0011000B
1574 073C 00 01    MOV    AH,CL
1575 073E D2 E4    SHL    AH,CL
1576
1577 ; AL = DRIVE SELECT FROM *MOTOR_STATUS
1578 ; BL = DRIVE SELECT DESIRED
1579 ; AH = MOTOR ON MASK DESIRED
1580
1581 0740 3A C3    CMP    AL,BL
1582 0742 75 06    JNZ    TURN_IT_ON
1583 0744 84 26 003F R  TEST   AH,*MOTOR_STATUS
1584 0748 75 2C    JNZ    NO_MOT_WAIT
1585
1586 074A          TURN_IT_ON:
1587 074A 0A E3    CMP    AH,BL
1588 074C 8A 3E 003F R  MOV    BH,*MOTOR_STATUS
1589 0750 80 E7 0F    AND    BH,0000111B
1590 0753 80 26 003F R  AND    BH,*MOTOR_STATUS,1100111B
1591 0756 80 26 003F R  OR     BH,*MOTOR_STATUS,AH
1592 075C 80 E7 0F    MOV    AH,*MOTOR_STATUS
1593 075F 8A 08    MOV    BH,BL
1594 0761 80 E3 0F    AND    BL,0000111B
1595 0764 FB        STI
1596 0765 24 3F    AND    AL,0011111B
1597 0767 C0 C0 04    ROL    AL,4
1598 076C 00 0C    OR     AL,00001100B
1599 076C BA 03F2    MOV    DX,03F2H
1600 076F EE        OUT    DX,AL
1601 0770 3A 0F    CMP    BL,BH
1602 0772 74 02    JZ    NO_MOT_WAIT
1603 0774 F8        CLC
1604 0775 C3        RET
1605
1606 0776          NO_MOT_WAIT:
1607 0776 F9        STC
1608 0777 FB        STI
1609 0778 C3        RET
1610 0779
1611
1612 ; HD_WAIT : WAIT FOR HEAD SETTLE TIME.
1613
1614 ; ON ENTRY: DI : DRIVE #
1615
1616 ; ON EXIT: AX,BX,CX,DX DESTROYED
1617
1618 0779          HD_WAIT PROC  NEAR
1619 0779 B2 09    MOV    DL,9
1620 077B E8 06CC R  CALL   GET_PARM
1621 077E F6 06 003F R 80  TEST   *MOTOR_STATUS,1000000B
1622 0783 74 14    JZ    ISNT_WRITE
1623 0783 74 E4    OR     AH,AH
1624 0787 75 14    JNZ    DO_WAIT
1625 0789 BA 04    MOV    AH,HD12_SETTLE
1626 078B 8A 85 0090 R  MOV    AL,*DSK_STATE[DI]
1627 078F 24 C0    AND    AL,RATE_MSK
1628 0791 3C 80    CMP    AL,RATE_250
1629 0793 75 08    JNZ    DO_WAIT
1630
1631 0795 B4 14    MOV    AH,HD320_SETTLE
1632 0797 EB 04    JMP    SHORT DO_WAIT
1633
1634 0799          ISNT_WRITE:
1635 0799 0A E4    OR     AH,AH
1636 079B 74 1F    JZ    HW_DONE
1637
1638 ;----- AH CONTAINS NUMBER OF MILLISECONDS TO WAIT
1639
1640 079D          DO_WAIT:
1641 079D 8A C4    MOV    AL,AH
1642 079F 32 E4    XOR    AH,AH
1643 07A1 50    PUSH   AX
1644 07A2 BA 03E8    MOV    DX,1000
1645 07A5 F7 E2    MUL    DX
1646 07A7 88 CA    MOV    CX,DX
1647 07A9 00 D0    MOV    DX,AX
1648 07AB BA 86    MOV    AH,86H
1649 07AD CD 15    INT    15H
1650 07AF 58    POP    AX
1651 07B0 T3 0A    JNC    HW_DONE
1652
1653 07B2          J29:
1654 07B2 B9 0042    MOV    CX,66
1655 07B5 E8 0000 E  CALL   WAITF
1656 07B8 FE C8    DEC    AL
1657 07BA 75 F6    JNZ    J29
1658 07BC          HW_DONE:
1659 07BD          RET
1660 07BD          HD_WAIT ENDP
1661
1662 ;----- NEC_OUTPUT: THIS ROUTINE SENDS A BYTE TO THE NEC CONTROLLER AFTER TESTING
1663 ;----- FOR CORRECT DIRECTION AND CONTROLLER READY THIS ROUTINE WILL
1664 ;----- TIME OUT IF THE BYTE IS NOT ACCEPTED WITHIN A REASONABLE AMOUNT
1665 ;----- OF TIME, SETTING THE DISKETTE STATUS ON COMPLETION.
1666
1667 ; ON ENTRY: AH = BYTE TO BE OUTPUT
1668 ; ON EXIT: CY = 0 SUCCESS
1669 ;           CY = 1 FAILURE -- DISKETTE STATUS UPDATED
1670 ;           CY = 2 FAILURE HAS OCCURRED, THE RETURN IS MADE ONE LEVEL
1671 ;           HIGHER THAN THE CALLER OF NEC_OUTPUT. THIS REMOVES THE
1672 ;           REQUIREMENT OF TESTING AFTER EVERY CALL OF NEC_OUTPUT.
1673
1674 ; AX,BX,CX,DX DESTROYED
1675
1676 07BD          NEC_OUTPUT PROC  NEAR
1677 07BD BA 03F4    MOV    DX,03F4H
1678 07C0 B3 02    MOV    BL,2
1679 07C2 33 C9    XOR    CX,CX
1680
1681 ; BX = DRIVE #
1682 ; DL = DRIVE #
1683 ; BL = DRIVE SELECT
1684 ; INTERRUPTS WHILE DETERMINING STATUS
1685 ; ENSURE MOTOR STAYS ON FOR OPERATION
1686 ; GET DIGITAL OUTPUT REGISTER REFLECTION
1687 ; KEEP ONLY DRIVE SELECTED
1688 ; MASK FOR DETERMINING MOTOR BIT
1689 ; AH = MOTOR ON, A=00000001, B=00000010
1690
1691 ; REQUESTED DRIVE ALREADY SELECTED ?
1692 ; IF NOT SELECTED JUMP
1693 ; TEST MOTOR ON BIT
1694 ; JUMP IF MOTOR ON AND SELECTED
1695
1696 ; AH = DRIVE SELECT AND MOTOR ON
1697 ; SAVE COPY OF *MOTOR_STATUS BEFORE
1698 ; KEEP ONLY MOTOR BITS
1699 ; CLEAR OUT DRIVE SELECT
1700 ; OR IN DRIVE SELECTED AND MOTOR ON
1701 ; OR IN DRIVE SELECTED AND MOTOR ON
1702 ; BL=*MOTOR_STATUS AFTER, BH-BEFORE
1703 ; KEEP ONLY MOTOR BITS
1704 ; ENABLE INTERRUPTS AGAIN
1705 ; STRIP AWAY UNWANTED BITS
1706 ; PUT BITS IN DESIRED POSITION
1707 ; NO RESET, ENABLE DMA/INTERRUPT
1708 ; SELECT DRIVE AND TURN ON MOTOR
1709
1710 ; NEW MOTOR TURNED ON ?
1711 ; NO WAIT REQUIRED IF JUST SELECT
1712 ; SET CARRY MEANING WAIT
1713
1714 ; SET NO_WAIT REQUIRED
1715 ; INTERRUPTS BACK ON
1716
1717 ;----- HD_WAIT : WAIT FOR HEAD SETTLE TIME.
1718
1719 ; ON ENTRY: DI : DRIVE #
1720
1721 ; ON EXIT: AX,BX,CX,DX DESTROYED
1722
1723 ;----- AH CONTAINS NUMBER OF MILLISECONDS TO WAIT
1724
1725 ;----- DO_WAIT:
1726 ;----- AH = # MILLISECONDS
1727 ;----- AX = # MILLISECONDS
1728 ;----- SAVE HEAD SETTLE PARAMETER
1729 ;----- SET UP FOR MULTIPLY TO MICROSECONDS
1730 ;----- DX,AX = # MICROSECONDS
1731 ;----- CX,DX = # MICROSECONDS
1732 ;----- DL,DX = # MICROSECONDS
1733 ;----- LOAD_WAIT CODE
1734 ;----- LOAD_WAIT CODE
1735 ;----- PERFORM_WAIT
1736 ;----- RESTORE HEAD SETTLE PARAMETER
1737 ;----- CHECK FOR EVENT_WAIT ACTIVE
1738
1739 ;----- I_MILLISECOND_LOOP:
1740 ;----- COUNT AT 15.085737 US PER COUNT
1741 ;----- DELAY FOR 1 MILLISECOND
1742 ;----- DECREMENT THE COUNT
1743 ;----- DO AL MILLISECOND # OF TIMES
1744
1745 ;----- NEC_OUTPUT: THIS ROUTINE SENDS A BYTE TO THE NEC CONTROLLER AFTER TESTING
1746 ;----- FOR CORRECT DIRECTION AND CONTROLLER READY THIS ROUTINE WILL
1747 ;----- TIME OUT IF THE BYTE IS NOT ACCEPTED WITHIN A REASONABLE AMOUNT
1748 ;----- OF TIME, SETTING THE DISKETTE STATUS ON COMPLETION.
1749
1750 ;----- ON ENTRY: AH = BYTE TO BE OUTPUT
1751 ;----- ON EXIT: CY = 0 SUCCESS
1752 ;           CY = 1 FAILURE -- DISKETTE STATUS UPDATED
1753 ;           CY = 2 FAILURE HAS OCCURRED, THE RETURN IS MADE ONE LEVEL
1754 ;           HIGHER THAN THE CALLER OF NEC_OUTPUT. THIS REMOVES THE
1755 ;           REQUIREMENT OF TESTING AFTER EVERY CALL OF NEC_OUTPUT.
1756
1757 ; AX,BX,CX,DX DESTROYED
1758
1759 ;----- NEC_OUTPUT PROC  NEAR
1760 ;----- DX,03F4H
1761 ;----- STATUS PORT
1762 ;----- HIGH ORDER COUNTER
1763 ;----- COUNT FOR TIME OUT

```

```

1680
1681 07C4 EC      J23:  IN    AL,DX      ; GET STATUS
1682 07C5 24 C0    AND   AL,1000000B ; KEEP STATUS AND DIRECTION 0
1683 07C7 3C 80    CMP   AL,1000000B ; STATUS 1 AND DIRECTION 0 ?
1684 07C9 74 0E    JZ    J27      ; STATUS AND DIRECTION OK
1685 07CB E2 F7    LOOP  J23      ; CONTINUE TILL CX EXHAUSTED
1686
1687 07CD FE CB    DEC   BL       ; DECREMENT COUNTER
1688 07CF 75 F3    JNZ   J23      ; REPEAT TILL DELAY FINISHED, CX = 0
1689
1690
1691 ;----- FALL THRU TO ERROR RETURN
1692 07D1 80 0E 0041 R 80
1693 07D6 58      OR    @DSKETTE_STATUS,TIME_OUT
1694 07D7 F9      POP   AX       ; DISCARD THE RETURN ADDRESS
1695 07D8 C3      STC   AX       ; INDICATE ERROR TO CALLER
1696
1697
1698 ;----- DIRECTION AND STATUS OK; OUTPUT BYTE
1699 07D9 8A C4      J27:  MOV   AL,AH      ; GET BYTE TO OUTPUT
1700 07DB 42      INC   DX       ; DATA PORT = STATUS PORT + 1
1701 07DC EE      OUT   DX,AL    ; OUTPUT THE BYTE
1702 07DD C3      RET
1703 07DE      NEC_OUTPUT ENDP
1704
1705 ;----- SEEK:
1706 ;----- THIS ROUTINE WILL MOVE THE HEAD ON THE NAMED DRIVE TO THE NAMED
1707 ;----- TRACK. IF THE DRIVE HAS NOT BEEN ACCESSED SINCE THE DRIVE
1708 ;----- RESET COMMAND WAS ISSUED, THE DRIVE WILL BE RECALIBRATED.
1709
1710 ;----- ON ENTRY: DI : DRIVE #
1711 ;----- CH : TRACK #
1712
1713 ;----- ON EXIT: @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION.
1714 ;----- AX,BX,CX,DX DESTROYED
1715 07DE      SEEK  PROC NEAR
1716 07DE 8B DF      MOV   BX,DI      ; BX = DRIVE #
1717 07E0 BA 083D R    MOV   AX,0FFH    ; LOAD RETURN ADDRESS
1718 07E3 52      PUSH  DX      ; ON STACK FOR NEC_OUTPUT ERROR
1719 07E4 B0 01      MOV   AL,I      ; ESTABLISH MASK FOR RECALIBRATE TEST
1720 07E6 86 CB      XCHG  CL,BL    ; GET DRIVE VALUE INTO CL
1721 07E8 D2 C0      ROL   AL,CL    ; SHIFT MASK BY THE DRIVE VALUE
1722 07E9 86 CB      XCHG  CL,BL    ; RECOVER DRIVE VALUE
1723 07EC 84 06 003E R    TEST  AL,?SEEK_STATUS
1724 07F0 75 1C      JNZ   J28A    ; TEST FOR RECALIBRATE REQUIRED
1725
1726 07F2 08 06 003E R    OR    @SEEK_STATUS,AL ; TURN ON THE NO RECALIBRATE BIT IN FLAG
1727 07F6 E8 083E R    CALL  RECAL    ; RECALIBRATE DRIVE
1728 07F9 T3 0A      JNC   AFT_RECAL ; RECALIBRATE DONE
1729
1730 ;----- ISSUE RECALIBRATE FOR 80 TRACK DISKETTES
1731
1732 07FB C6 06 0041 R 00
1733 0800 E8 083E R    MOV   @DSKETTE_STATUS,0 ; CLEAR OUT INVALID STATUS
1734 0803 T2 37      CALL  RECAL    ; RECALIBRATE DRIVE
1735
1736 0805      AFT_RECAL:
1737 0805 C6 85 0094 R 00
1738 0804 0A ED      MOV   CH,CH      ; SAVE NEW CYLINDER AS PRESENT POSITION
1739 080C 74 29      OR    CH,CH      ; CHECK FOR SEEK TO TRACK 0
1740
1741 ;----- DRIVE IS IN SYNCHRONIZATION WITH CONTROLLER, SEEK TO TRACK
1742
1743 080E F6 85 0090 R 20
1744 0813 74 02      J28A: TEST  @DSK_STATE[DI],DBL_STEP ; CHECK FOR DOUBLE STEP REQUIRED
1745 0815 D0 E5      JZ    R7      ; SINGLE STEP REQUIRED BYPASS DOUBLE
1746
1747 0817 3A AD 0094 R    SHL   CH,1      ; DOUBLE NUMBER OF STEP TO TAKE
1748 081B 74 1F      R7:  CMP   CH,@DSK_TRK[DI] ; SEE IF ALREADY AT THE DESIRED TRACK
1749
1750 081D 88 AD 0094 R    JE    RB      ; IF YES, DO NOT NEED TO SEEK
1751 0821 80 0F      MOV   @DSK_TRK[DI],CH ; SAVE NEW CYLINDER AS PRESENT POSITION
1752 0823 28 07BD R    MOV   AH,0FH    ; SEEK COMMAND TO NEC
1753 0826 88 DF      CALL  NEC_OUTPUT ; NO NEC_OUTPUT
1754 0828 8A E3      MOV   AH,BL    ; BX = DRIVE #
1755 082A E8 07BD R    CALL  NEC_OUTPUT ; OUTPUT DRIVE NUMBER
1756 082D 8A A5 0094 R    MOV   AH,@DSK_TRK[DI] ; GET CYLINDER NUMBER
1757 0831 E8 07BD R    CALL  NEC_OUTPUT ; ENDING INTERRUPT AND SENSE STATUS
1758 0834 E8 0855 R    CALL  CHK_STAT_2
1759
1760 ;----- WAIT FOR HEAD SETTLE
1761
1762 0837      DO_WAIT:
1763 0838 9C      PUSHF
1764 083E E8 0779 R    CALL  HD_WAIT ; SAVE STATUS
1765 083B 9D      POPF   AX       ; WAIT FOR HEAD SETTLE TIME
1766 083C
1767 083C 58      RB:  POP   AX       ; RESTORE STATUS
1768 083D
1769 083D C3      NEC_ERR: RET
1770 083E      SEEK  ENDP
1771
1772 ;----- RECAL : RECALIBRATE DRIVE
1773
1774 ;----- ON ENTRY: DI = DRIVE #
1775
1776 ;----- ON EXIT: CY REFLECTS STATUS OF OPERATION.
1777
1778 083E      RECAL  PROC NEAR
1779 083E 51      PUSH  CX      ; LOAD NEC_OUTPUT ERROR
1780 083F B8 0853 R    MOV   AX,0FFH    ; RECALIBRATE COMMAND
1781 0845 50      PUSH  AX
1782 0845 07      MOV   AH,07H    ; RECALIBRATE
1783 0845 E8 07BD R    CALL  NEC_OUTPUT ; NO NEC_OUTPUT
1784 0848 88 DF      MOV   BX,DI    ; BX = DRIVE #
1785 084A 8A E3      MOV   AH,BL    ; OUTPUT THE DRIVE NUMBER
1786 084C E8 07BD R    CALL  NEC_OUTPUT ; GET THE INTERRUPT AND SENSE INT STATUS
1787 084E E8 0855 R    CALL  CHK_STAT_2
1788 0852 58      POP   AX       ; THROW AWAY ERROR
1789 0853      RC_BACK: POP   CX
1790 0853 59      RET
1791 0854 C3      RECAL ENDP
1792 0855
1793

```

```

1794 ;----- CHK_STAT_2: THIS ROUTINE HANDLES THE INTERRUPT RECEIVED AFTER RECALIBRATE,
1795 ;----- SEEK, OR RESET TO THE ADAPTER. THE INTERRUPT IS WAITED FOR THE
1796 ;----- INTERRUPT STATUS SENSED, AND THE RESULT RETURNED TO THE CALLER.
1797 ;----- ON EXIT: #DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION.
1798 ;----- CHK_STAT_2 PROC NEAR
1799 ;----- MOV AL,[OFFSET CS_BACK
1800 ;----- PUSH AX
1801 ;----- CALL WAIT_INT
1802 ;----- JC J34
1803 ;----- MOV AH,0BH
1804 ;----- CALL NEC_OUTPUT
1805 ;----- CALL RESULTS
1806 ;----- JC J34
1807 ;----- MOV AL,NEC_STATUS
1808 ;----- AND AL,0110000B
1809 ;----- CMP AL,0110000B
1810 ;----- JZ J35
1811 ;----- CLC
1812 ;----- OR AL,NEC_STATUS
1813 ;----- AND AL,0110000B
1814 ;----- CMP AL,0110000B
1815 ;----- JZ J35
1816 ;----- CLC
1817 ;----- POP AX
1818 ;----- CS_BACK:
1819 ;----- RET
1820 ;----- J34:
1821 ;----- OR STC
1822 ;----- CS_BACK:
1823 ;----- RET
1824 ;----- CHK_STAT_2 ENDP
1825 ;----- ;----- WAIT_INT: THIS ROUTINE WAITS FOR AN INTERRUPT TO OCCUR A TIME OUT ROUTINE
1826 ;----- TAKES PLACE DURING THE WAIT, SO THAT AN ERROR MAY BE RETURNED
1827 ;----- IF THE DRIVE IS NOT READY.
1828 ;----- ON EXIT: #DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION.
1829 ;----- WAIT_INT PROC NEAR
1830 ;----- STI
1831 ;----- CLC
1832 ;----- MOV AX,09001H
1833 ;----- INT 15H
1834 ;----- JC J36
1835 ;----- INT 15H
1836 ;----- JC J36
1837 ;----- MOV BL,4
1838 ;----- XOR CX,CX
1839 ;----- TEST #SEEK_STATUS, INT_FLAG
1840 ;----- JNZ J37
1841 ;----- LOOP J36
1842 ;----- DEC BL
1843 ;----- JC J36
1844 ;----- DEC BL
1845 ;----- JC J36
1846 ;----- DEC BL
1847 ;----- JC J36A
1848 ;----- OR STC
1849 ;----- JC J37
1850 ;----- PUSHF #SEEK_STATUS, NOT INT_FLAG
1851 ;----- AND POPF
1852 ;----- JC J37
1853 ;----- RET
1854 ;----- #SEEK_STATUS, TIME_OUT
1855 ;----- OR STC
1856 ;----- JC J37
1857 ;----- #SEEK_STATUS, NOT INT_FLAG
1858 ;----- AND POPF
1859 ;----- JC J37
1860 ;----- RET
1861 ;----- #SEEK_STATUS, CY REFLECT STATUS OF OPERATION.
1862 ;----- AX,BX,CX,DX DESTROYED
1863 ;----- RESULTS PROC NEAR
1864 ;----- PUSH DI
1865 ;----- MOV DI,[OFFSET NEC_STATUS
1866 ;----- MOV BL,7
1867 ;----- MOV DX,03F4H
1868 ;----- ;----- WAIT FOR REQUEST FOR MASTER
1869 ;----- R10: MOV BH,2
1870 ;----- XOR CX,CX
1871 ;----- JC J39
1872 ;----- IN AL,DX
1873 ;----- AND AL,0100000B
1874 ;----- CMP AL,0100000B
1875 ;----- JZ J42
1876 ;----- LOOP J39
1877 ;----- DEC BH
1878 ;----- JC J39
1879 ;----- #DECREMENT HIGH ORDER COUNTER
1880 ;----- #REPEAT TILL DELAY DONE
1881 ;----- #----- READ IN THE STATUS
1882 ;----- R08: OR #DSKETTE_STATUS, TIME_OUT
1883 ;----- STC
1884 ;----- JMP SHORT POPRES
1885 ;----- SET ERROR RETURN
1886 ;----- POPRES
1887 ;----- #----- READ IN THE STATUS
1888 ;----- R09: J42: INC DX
1889 ;----- IN AL,DX
1890 ;----- MOV [DI],AL
1891 ;----- INC DI
1892 ;----- #----- POINT AT DATA PORT
1893 ;----- MOV CX,2
1894 ;----- CALL WAITF
1895 ;----- DEC DX
1896 ;----- IN AL,DX
1897 ;----- TEST AL,0001000B
1898 ;----- JZ POPRES
1899 ;----- #POINT AT STATUS PORT
1900 ;----- #GET STATUS
1901 ;----- #TEST FOR NEC STILL BUSY
1902 ;----- #RESULTS DONE ?
1903 ;----- DEC BL
1904 ;----- JC R10
1905 ;----- #DECREMENT THE STATUS COUNTER
1906 ;----- #GO BACK FOR MORE
1907 ;----- #TOO MANY STATUS BYTES
1908 ;----- #SET ERROR FLAG
1909 ;----- #----- RESULT OPERATION IS DONE

```

```
1908 08E1          POPRES:      POP    DI
1909 08E1 5F        RET             ; RETURN WITH CARRY SET
1910 08E2 C3        RESULTS ENDP
1911 08E3
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922 08E3          READ_DSKCHNG PROC  NEAR
1923 08E4 E8 06E1 R CALL  MOTOR_ON      ; TURN ON THE MOTOR IF OFF
1924 08E5 E8 03F7 R MOV   DX,03F7H    ; ADDRESS DIGITAL INPUT REGISTER
1925 08E9 EC        IN    AL,DX        ; INPUT DIGITAL INPUT REGISTER
1926 08EA A8 80      TEST  AL,DSK_CHG  ; CHECK FOR DISK CHANGE LINE ACTIVE
1927 08EC C3        RET             ; RETURN TO CALLER WITH ZERO FLAG SET
1928 08ED
1929
1930
1931
1932
1933
1934
1935 08E5          READ_DSKCHNG ENDP
1936 08E6          I-  DRIVE_DET: DETERMINES WHETHER DRIVE IS 80 OR 40 TRACKS AND UPDATES STATE
1937 08E7 E8 083E R CALL  MOTOR_ON      ; TURN ON MOTOR IF NOT ALREADY ON
1938 08F3 12 3C      CALL  RECALIBRATE  ; RECALIBRATE DRIVE
1939 08F5 B5 30      JC   DD_BAC      ; ASSUME NO DRIVE PRESENT
1940 08F7 E8 07DE R CALL  SEEK        ; SEEK TO TRACK #8
1941 08FA T2 35      JC   DD_BAC      ; ERROR NO DRIVE
1942 08F8 B5 0B      MOV   CH,QUIET_SEEK+1 ; SEEK TO TRACK 10
1943 08FE
1944 08FE FF CD      SK_GIN:      DEC   CH      ; DECREMENT TO NEXT TRACK
1945 0900 51          PUSH  CX      ; SAVE TRACK
1946 0901 E8 07DE R CALL  SEEK        ; *
1947 0904 T2 2C      JC   POP_BAC    ; POP AND RETURN
1948 0905 00 931 R   MOV   AX,OFFSET DD_BAC ; LOAD NEC OUTPUT ERROR ADDRESS
1949 0909 B0          PUSH  AX      ; SENSE DRIVE STATUS COMMAND BYTE
1950 090A B4 04      MOV   AH,SENSE_DRY_ST ; NEC_OUTPUT
1951 090C E8 07BD R CALL  NEC_OUTPUT  ; OUTPUT TO NEC
1952 090F B8 C7      MOV   AX,DI      ; AL = DRIVE
1953 0911 8A E0      MOV   AH,AL      ; AH = DRIVE
1954 0913 80 07BD R CALL  NEC_OUTPUT  ; OUTPUT TO NEC
1955 0916 E8 0844 R CALL  NEC_RESULTS ; GO TILL TRACK 0
1956 0919 58          POP   AX      ; THROW AWAY ERROR ADDRESS
1957 091A 59          POP   CX      ; RESTORE TRACK
1958 091B F6 06 0042 R 10     TEST  ONEC_STATUS,HOME ; TRACK 0 ?
1959 0920 T4 DC      JZ   SK_GTN    ; GO TILL TRACK 0
1960 0922 0A ED      OR    CH,CH      ; IS HOME AT TRACK 0 ?
1961 0924 T4 06      JZ   IS_80      ; MUST BE 80 TRACK DRIVE
1962
1963
1964
1965
1966 0926 80 8D 0090 R 94     IS_80:      OR    #DSK_STATE[DI],DRV_DET+MED_DET+RATE_250
1967 092B C3          RET             ; ALL INFORMATION SET
1968
1969 092C
1970 092C 80 8D 0090 R 01     DD_BAC:    OR    #DSK_STATE[DI],TRK_CAPA ; SETUP 80 TRACK CAPABILITY
1971 0931 C3          RET
1972
1973
1974 0932
1975 0932 59
1976 0933 C3
1977
1978 0934          DRIVE_DET:    ENDP
```

```

1979
1980
1981
1982
1983
1984
1985
1986
1987
1988 0934 PAGE
1989 0934 FB ;--- HARDWARE INT 08 H -- ( IRQ LEVEL 6 ) -----
1990 0935 50 ; DISK_INT THIS ROUTINE HANDLES THE DISKETTE INTERRUPT.
1991 0936 1E
1992 0937 E8 0000 E ; ON EXIT: THE INTERRUPT FLAG IS SET IN @SEEK_STATUS.
1993 0937 0E 003E R 80
1994 093F 1F
1995 0940 B0 20
1996 0942 E6 20
1997 0944 B8 9101
1998 0947 CD 15
1999 0949 58
2000 094A CF
2001
2002 094B DISK_INT_I PROC FAR
2003
2004
2005
2006
2007
2008
2009
2010 094B DISK_INT_I ENDP
2011
2012 0949 50
2013 094D 51
2014 094E 52
2015 094F 57
2016 0950 1E
2017 0951 0000 E
2018 0954 B0 0E 00A0 R 01
2019 0959 33 FF
2020 095B C7 04 0090 R 0000
2021 0961 80 24 008B R 33
2022 0966 80 0E 008B R CO
2023 0967 80 0E 008B R CO
2024 0970 C6 06 0040 R 00
2025 0975 C6 06 003F R 00
2026 097A C6 06 0041 R 00
2027
2028 097F
2029 0982 E8 08ED R
2030 0982 E8 02EE R
2031 0985 47
2032 0986 83 FF 02
2033 0989 75 F4
2034 098B C6 06 003E R 00
2035 098C 80 26 00A0 R FE
2036 0995 E8 0620 R
2037 0998 1F
2038 0999 5F
2039 099A 5A
2040 099B 59
2041 099C 5B
2042 099D 5B
2043 099E C3
2044
2045 099F
2046
2047 099F
2048

PAGE
DISK_INT_I PROC FAR
    PUSH AX
    PUSH DS
    CALL DDS
    ORF @SEEK_STATUS, INT_FLAG
    POP DS
    MOV AL, EO1
    OUT INTA00, AL
    MOV AX, 09101H
    INT 15H
    POP AX
    IRET

    ;----- ENTRY POINT FOR ORG 00F5TH
    ;----- RE-ENABLE INTERRUPTS
    ;----- SAVE WORK REGISTER
    ;----- SAVE REGISTERS
    ;----- SETUP DATA ADDRESSING
    ;----- TURN ON INTERRUPT OCCURRED
    ;----- RESTORE USER (DS)
    ;----- END OF INTERRUPT MARKER
    ;----- INTERRUPT CONTROL PORT
    ;----- INTERRUPT POST CODE AND TYPE
    ;----- GO PERFORM OTHER TASK
    ;----- RECOVER REGISTER
    ;----- RETURN FROM INTERRUPT

DISK_INT_I ENDP

;----- DISKETTE_SETUP: THIS ROUTINE DOES A PRELIMINARY CHECK TO SEE WHAT TYPE OF
;----- DISKETTE DRIVES ARE ATTACH TO THE SYSTEM.
;----- DISKETTE_SETUP PROC NEAR
;-----     PUSH AX ; SAVE REGISTERS
;-----     PUSH BX
;-----     PUSH CX
;-----     PUSH DX
;-----     PUSH DI
;-----     PUSH DS
;-----     ORF @RTC_WAIT_FLAG, 01 ; POINT DATA SEGMENT TO BIOS DATA AREA
;-----     XOR DI, DT ; NO RTC WAIT, FORCE USE OF LOOP
;-----     MOV WORD PTR @DSK_STATUS, 0 ; INITIALIZE DRIVE POINTER
;-----     AND @LASTRATE, NOT @STRT_MSK + SEND_MSK ; CLEAR START & SEND
;-----     ORF @LASTRATE, SEND_MSK ; INITIALIZE SENT TO IMPOSSIBLE
;-----     MOV @SEEK_STATUS, 0 ; INDICATE RECALIBRATE NEEDED
;-----     MOV @DR_STATUS, 0 ; INITIALIZE DRIVES COUNT
;-----     MOV @MOTOR_STATUS, 0 ; INITIALIZE DRIVES TO OFF STATE
;-----     MOV @DSKETTE_STATUS, 0 ; NO ERRORS

SUP0: ;----- DETERMINE DRIVE
    CALL DRIVE_DET ; TRANSLATE STATE TO COMPATIBLE MODE
    INC DI ; POINT TO NEXT DRIVE
    CMP DI, MAX_DRV ; SEE IF DONE
    JNZ SUP0 ; REPEAT FOR EACH DRIVE
    CALL SETUP_END ; FORCE RECALIBRATE
    ORF @RTC_WAIT_FLAG, 0FEH ; ALLOW FOR RTC WAIT
    CALL SETUP_END ; VARIOUS CLEANUPS
    RET ; RESTORE CALLERS RESISTERS

DSKETTE_SETUP ENDP

CODE ENDS
ENDS

```

```

1 PAGE 118,121
2 TITLE DISK ----- 06/10/85 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 DDSI:NEAR
14 EXTRN E_MSGI:NEAR
15 EXTRN FT780I:NEAR
16 EXTRN F1781I:NEAR
17 EXTRN F1782I:NEAR
18 EXTRN F1790I:NEAR
19 EXTRN F1791I: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

PAGE 118,121
TITLE DISK ----- 06/10/85 FIXED DISK BIOS
.D286C
.LIST
CODE SEGMENT BYTE PUBLIC
PUBLIC DISK IO
PUBLIC DISK SETUP
PUBLIC HD_INT
EXTRN CMOS_READ:NEAR
EXTRN CMOS_WRITE:NEAR
EXTRN DDSI:NEAR
EXTRN E_MSGI:NEAR
EXTRN FT780I:NEAR
EXTRN F1781I:NEAR
EXTRN F1782I:NEAR
EXTRN F1790I:NEAR
EXTRN F1791I:NEAR
EXTRN FD_TBL:NEAR

--- INT 13H ---
;-----+
; FIXED DISK I/O INTERFACE
;-----+
; THIS INTERFACE PROVIDES ACCESS TO 5 1/4" FIXED DISKS THROUGH
; THE IBM FIXED DISK CONTROLLER.
;-----+
; THE BIOS ROUTINES ARE MEANT TO BE ACCESSED THROUGH
; SOFTWARE INTERRUPTS ONLY. ANY ADDRESSES PRESENT IN
; THESE LISTINGS ARE INCLUDED ONLY FOR COMPLETENESS,
; NOT FOR REFERENCE. APPLICATIONS WHICH REFERENCE ANY
; ABSOLUTE ADDRESSES WITHIN THE CODE SEGMENTS OF BIOS
; VIOLATE THE STRUCTURE AND DESIGN OF BIOS.
;-----+
;-----+
; INPUT (AH)= HEX COMMAND VALUE
;-----+
; (AH)= 00H RESET DISK (DL = 80H,81H) / DISKETTE
; (AH)= 01H READ THE STATUS OF THE LAST DISK OPERATION INTO (AL)
;       NOTE: DL = 80H - DISKETTE
;             DL > 80H - DISK
; (AH)= 02H READ THE DESIRED SECTORS INTO MEMORY
; (AH)= 03H WRITE THE DESIRED SECTORS FROM MEMORY
; (AH)= 04H VERIFY THE DESIRED SECTORS
; (AH)= 05H POSITION THE DESIRED TRACK
; (AH)= 06H UNUSED
; (AH)= 07H UNUSED
; (AH)= 08H RETURN THE CURRENT DRIVE PARAMETERS
; (AH)= 09H INITIALIZE DRIVE PARTIAL 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)= 11H TEST DRIVE READY
; (AH)= 12H UNUSED
; (AH)= 13H UNUSED
; (AH)= 14H CONTROLLER INTERNAL DIAGNOSTIC
; (AH)= 15H READ DASD TYPE
;-----+
;-----+
; REGISTERS USED FOR FIXED DISK OPERATIONS
;-----+
; (DL) - DRIVE NUMBER (80H-81H FOR DISK, VALUE CHECKED)
; (DH) - HEAD NUMBER (0-15 ALLOWED, NOT VALUE CHECKED)
; (CH) - CYLINDER NUMBER (0-1023, NOT VALUE CHECKED) (SEE CL)
; (CL) - SECTOR NUMBER (1-17, NOT VALUE CHECKED)
;-----+
; NOTE: HIGH 2 BITS OF CYLINDER NUMBER ARE PLACED
;       IN THE HIGH 2 BITS OF THE CL REGISTER
;       (10 BITS TOTAL)
;-----+
; (AL) - NUMBER OF SECTORS (MAXIMUM POSSIBLE RANGE 1-80H,
;       FOR READ/WRITE LONG 1-79H)
;-----+
; (ES:BX) - ADDRESS OF BUFFER FOR READS AND WRITES,
;       (NOT REQUIRED FOR VERIFY)
;-----+
; FORMAT (AH=5) ES:BX POINTS TO A 512 BYTE BUFFER. THE FIRST
; 2* (SECTORS/TRACK) BYTES CONTAIN F,N FOR EACH SECTOR.
; F = 00H FOR A GOOD SECTOR
;     80H FOR A BAD SECTOR
; N = SECTOR NUMBER
; FOR AN INTERLEAVE OF 2 AND 17 SECTORS/TRACK
; THE TABLE SHOULD BE:
;-----+
; DB 00H,01H,00H,04H,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
;-----+

```



```
188 PAGE
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216 = 01F0 HF_PORT EQU 01F0H ; DISK PORT
217 = 03F6 HF_REG_PORT EQU 03F6H
218
219 ;---- STATUS REGISTER
220
221 = 0001 ST_ERROR EQU 00000001B
222 = 0002 ST_INDEX EQU 00000010B
223 = 0004 ST_ECC_CTD EQU 00000010B ; ECC CORRECTION SUCCESSFUL
224 = 0008 ST_DRO EQU 00000000B
225 = 0010 ST_SEEK_COMPL EQU 00000000B ; SEEK COMPLETE
226 = 0020 ST_WRT_FLT EQU 00100000B ; WRITE FAULT
227 = 0040 ST_READY EQU 01000000B
228 = 0080 ST_BUSY EQU 10000000B
229
230 ;---- ERROR REGISTER
231
232 = 0001 ERR_DAM EQU 00000001B ; DATA ADDRESS MARK NOT FOUND
233 = 0002 ERR_TRK_0 EQU 00000010B ; TRACK 0 NOT FOUND ON RECAL
234 = 0004 ERR_ABORT EQU 00000000B ; ABORTED COMMAND
235 ; 1
236 = 0010 ERR_ID EQU 00010000B ; ID NOT FOUND
237 ; 1
238 = 0040 ERR_DATA_ECC EQU 01000000B ; NOT USED
239 = 0080 ERR_BAD_BLOCK EQU 10000000B
240
241
242 = 0010 RECAL_CMD EQU 00010000B ; DRIVE RECAL (10H)
243 = 0020 READ_CMD EQU 00100000B ; READ (20H)
244 = 0030 WRITE_CMD EQU 00110000B ; WRITE (30H)
245 = 0040 VERIFY_CMD EQU 01000000B ; VERIFY (40H)
246 = 0050 FMTTRK_CMD EQU 01010000B ; FORMAT TRACK (50H)
247 = 0060 INIT_CMD EQU 01100000B ; INITIALIZATION (60H)
248 = 0070 SEEK_CMD EQU 01110000B ; SEEK (70H)
249 = 0090 DIAG_CMD EQU 10010000B ; DIAGNOSTIC (90H)
250 = 0091 SET_PARM_CMD EQU 10010001B ; DRIVE PARMS (91H)
251 = 0001 NO_RETRIES EQU 00000001B ; CMD MODIFIER (01H)
252 = 0002 ECC_MODE EQU 00000010B ; CMD MODIFIER (02H)
253 = 0008 BUFFER_MODE EQU 00000100B ; CMD MODIFIER (08H)
254
255 = 0002 MAX_FILE EQU 2
256 = 0002 S_MAX_FILE EQU 2
257
258 = 0025 DELAY_1 EQU 25H ; DELAY FOR OPERATION COMPLETE
259 = 0600 DELAY_2 EQU 0600H ; DELAY FOR READ
260 = 0100 DELAY_3 EQU 0100H ; DELAY FOR DATA REQUEST
261
262
263 = 0008 HF_FAIL EQU 08H ; CMOS FLAG IN BYTE 0EH
264
265 ;---- COMMAND BLOCK REFERENCE
266
267 = 0CMD_BLOCK EQU BYTE PTR [BP]-8 ; *CMD_BLOCK REFERENCES BLOCK HEAD IN SS
268 ; (BPT POINTS TO COMMAND BLOCK TAIL
269 ; AS DEFINED BY THE "ENTER" PARMS
```

```

270
271
272
273
274
275
276
277
278
279
280
281 0000 ASSUME CS:CODE,DS:ABS0 ; WORK OFF DS REGISTER
282 0000 FA
283 0001 BB ---- R
284 0004 BE DB
285 0006 A1 004C R
286 0009 A3 0100 R
287 000C A1 004E R
288 0012 C7 0000 R
289 0018 8C 0E 004E R
290 001C C7 06 01DB R 06CD R
291 0020 0004 R
292 0022 8E 001D R
293 0026 C7 06 0104 R 0000 E
294 002C 8C 0E 0106 R
295 0030 0004 R 0018 R 0000 E
296 0036 8C 0E 011A R
297 003A E4 A1
298 003C 24 BF
299 003E EB 00
300 0040 E6 A1
301 0042 0000 R
302 0044 24 FB
303 0046 EB 00
304 0048 E6 21
305
306 004A FB
307
308 004B 1E
309 004C 07
310 004D E8 0000 E
311 0050 C6 06 0074 R 00
312 0053 C6 06 0075 R 00
313 005A C6 06 0076 R 00
314 005D C6 06 0077 R 00
315 0061 E8 0000 E
316 0064 BA E0
317 0066 24 C0
318 0068 74 03
319 006A E9 00F6 R
320 0070 0000 R
321 007D 80 E4 F7
322 0070 80 E8
323 0072 E8 0000 E
324 0075 B8 92
325 0077 E8 0000 E
326 007A B8 90 0077 R 00
327 007F B8 D8
328 0081 25 00F0
329 0084 74 72
330
331 0086 3C F0
332 0088 75 10
333
334 008A B0 99
335 008C E8 0000 E
336 008F B8 00
337 0091 74 65
338 0093 3C 2F
339 0095 77 61
340 0097 C1 E0 04
341 009A
342 009A 05 FFFF E
343 009D 26: A3 0104 R
344 00A1 C6 06 0075 R 01
345 00A3 80 C5 00
346 00A5 80 00 04
347 00A8 74 2A
348 00AD B4 00
349
350 00AF 3C F0
351 00B1 75 10
352
353 00B3 B8 9A
354 00B5 E8 0000 E
355 00B8 38 00
356 00BA 74 1B
357 00BC 3C 2F
358 00BD 77 17
359 00C0 C1 E0 04
360 00C3
361 00C3 05 FFFF E
362 00C6 8B D8
363 00C8 E8: 83 3F 00
364 00CC 4C 00
365 00D5 26: A3 0118 R
366 00D2 C6 06 0075 R 02
367 00D7
368 00E7 B8 80
369 00E9 B4 14
370 00EB C8 3B
371 00F2 D2 12
372 00F6 A1 006C R
373 00E2 8B D8
374 00E4 05 044
375 00E7 8B C8
376 00E9 B8 004 R
377 00EC 20 06 0075 R 01
378 00F1 76 05
379 00F3 B2 81
380 00F5 E8 0104 R
381 00F8
382 00F8 C3
PAGE
: FIXED DISK I/O SETUP
: - ESTABLISH TRANSFER VECTORS FOR THE FIXED DISK
: - PERFORM POWER ON DIAGNOSTICS
: SHOULD AN ERROR OCCUR A "1701" MESSAGE IS DISPLAYED
:
DISK_SETUP PROC NEAR
    MOV CL,0
    MOV AX,AB50 ; GET ABSOLUTE SEGMENT
    MOV DS,AX ; SET SEGMENT REGISTER
    MOV AX,WORD PTR ORG_VECTOR ; GET DISKETTE VECTOR
    MOV WORD PTR ODISK_VECTOR,AX ; INTO INT 40H
    MOV AX,WORD PTR ORG_VECTOR+2
    MOV WORD PTR ODISK_VECTOR+2,AX ; GET FIXED DISK VECTOR
    MOV WORD PTR ORG_VECTOR+4,WORD PTR DISK_ID ; GET FIXED DISK HANDLER
    MOV WORD PTR ORG_VECTOR+2,CS ; GET FIXED DISK INTERRUPT
    MOV WORD PTR HDISK1_INT,OFFSET HD_INT ; FIXED DISK INTERRUPT
    MOV WORD PTR HDISK1_INT+2,CS ; PARM TABLE DRIVE 80
    MOV WORD PTR HF_TBL_VEC,OFFSET FD_TBL ; PARM TABLE DRIVE 81
    MOV WORD PTR HF_TBL_VEC+2,CS ; PARM TABLE DRIVE 81
    MOV WORD PTR HF_TBL_VEC,OFFSET FD_TBL ; PARM TABLE DRIVE 81
    MOV WORD PTR HF_TBL_VEC+2,CS ; PARM TABLE DRIVE 81
    IN AL,INTB01 ; TURN ON SECOND INTERRUPT CHIP
    AND AL,0BFH
    JMP $+2
    OUT INTB01,AL
    IN AL,INTA01 ; LET INTERRUPTS PASS THRU TO
    AND AL,0BFH ; SECOND CHIP
    JMP $+2
    OUT INTA01,AL
306
307 004A FB
308 004B 1E
309 004C 07
310 004D E8 0000 E
311 0050 C6 06 0074 R 00
312 0053 C6 06 0075 R 00
313 005A C6 06 0076 R 00
314 005D C6 06 0077 R 00
315 0061 E8 0000 E
316 0064 BA E0
317 0066 24 C0
318 0068 74 03
319 006A E9 00F6 R
320 0070 0000 R
321 007D 80 E4 F7
322 0070 80 E8
323 0072 E8 0000 E
324 0075 B8 92
325 0077 E8 0000 E
326 007A B8 90 0077 R 00
327 007F B8 D8
328 0081 25 00F0
329 0084 74 72
330
331 0086 3C F0
332 0088 75 10
333
334 008A B0 99
335 008C E8 0000 E
336 008F B8 00
337 0091 74 65
338 0093 3C 2F
339 0095 77 61
340 0097 C1 E0 04
341 009A
342 009A 05 FFFF E
343 009D 26: A3 0104 R
344 00A1 C6 06 0075 R 01
345 00A3 80 C5 00
346 00A5 80 00 04
347 00A8 74 2A
348 00AD B4 00
349
350 00AF 3C F0
351 00B1 75 10
352
353 00B3 B8 9A
354 00B5 E8 0000 E
355 00B8 38 00
356 00BA 74 1B
357 00BC 3C 2F
358 00BD 77 17
359 00C0 C1 E0 04
360 00C3
361 00C3 05 FFFF E
362 00C6 8B D8
363 00C8 E8: 83 3F 00
364 00CC 4C 00
365 00D5 26: A3 0118 R
366 00D2 C6 06 0075 R 02
367 00D7
368 00E7 B8 80
369 00E9 B4 14
370 00EB C8 3B
371 00F2 D2 12
372 00F6 A1 006C R
373 00E2 8B D8
374 00E4 05 044
375 00E7 8B C8
376 00E9 B8 004 R
377 00EC 20 06 0075 R 01
378 00F1 76 05
379 00F3 B2 81
380 00F5 E8 0104 R
381 00F8
382 00F8 C3
ASSUME DS:DATA,ES:ABS0
309 004C 07
310 004D E8 0000 E
311 0050 C6 06 0074 R 00
312 0053 C6 06 0075 R 00
313 005A C6 06 0076 R 00
314 005D C6 06 0077 R 00
315 0061 E8 0000 E
316 0064 BA E0
317 0066 24 C0
318 0068 74 03
319 006A E9 00F6 R
320 0070 0000 R
321 007D 80 E4 F7
322 0070 80 E8
323 0072 E8 0000 E
324 0075 B8 92
325 0077 E8 0000 E
326 007A B8 90 0077 R 00
327 007F B8 D8
328 0081 25 00F0
329 0084 74 72
330
331 0086 3C F0
332 0088 75 10
333
334 008A B0 99
335 008C E8 0000 E
336 008F B8 00
337 0091 74 65
338 0093 3C 2F
339 0095 77 61
340 0097 C1 E0 04
341 009A
342 009A 05 FFFF E
343 009D 26: A3 0104 R
344 00A1 C6 06 0075 R 01
345 00A3 80 C5 00
346 00A5 80 00 04
347 00A8 74 2A
348 00AD B4 00
349
350 00AF 3C F0
351 00B1 75 10
352
353 00B3 B8 9A
354 00B5 E8 0000 E
355 00B8 38 00
356 00BA 74 1B
357 00BC 3C 2F
358 00BD 77 17
359 00C0 C1 E0 04
360 00C3
361 00C3 05 FFFF E
362 00C6 8B D8
363 00C8 E8: 83 3F 00
364 00CC 4C 00
365 00D5 26: A3 0118 R
366 00D2 C6 06 0075 R 02
367 00D7
368 00E7 B8 80
369 00E9 B4 14
370 00EB C8 3B
371 00F2 D2 12
372 00F6 A1 006C R
373 00E2 8B D8
374 00E4 05 044
375 00E7 8B C8
376 00E9 B8 004 R
377 00EC 20 06 0075 R 01
378 00F1 76 05
379 00F3 B2 81
380 00F5 E8 0104 R
381 00F8
382 00F8 C3
POD_DONE PROC NEAR
    ASSUME DS:DATA,ES:ABS0
    PUSH DS
    POP ES
    CALL DDS ; ESTABLISH DATA SEGMENT
    MOV WORD PTR DISK_STATUS1,0 ; RESET THE STATUS INDICATOR
    MOV WORD PTR NUM_0,0 ; ZERO NUMBER OF FIXED DISKS
    MOV WORD PTR CMOS_DIAG+NMI ; CHECK CMOS VALIDITY
    CMP AX,0 ; SAVE CMOS FLAG
    JE POD_DONE ; CHECK FOR VALID CMOS
    CALL CMOS_READ ; ALLOW FIXED DISK IPL
    MOV AH,40 ; WRITE IT BACK
    CALL CMOS_WRITE ; ZERO CARD OFFSET
    MOV BX,0 ; SAVE CARD OFFSET
    AND AX,00F0H ; GET FIRST DRIVE TYPE AS OFFSET
    JZ POD_DONE ; NO FIXED DISKS
    CMP AL,0F0H ; CHECK FOR EXTENDED DRIVE TYPE BYTE USE
    JNE L1 ; USE DRIVE TYPE 1 --> 14 IF NOT IN USE
    CMP AL,0 ; ALLOW FIXED DISK
    JNE L1 ; WRITE IT BACK
    CMP AL,NOT HF FAIL ; GET EXTENDED TYPE FOR DRIVE C:
    JNE L1 ; FROM CMOS
    CMP AL,0 ; IS TYPE SET TO ZERO
    JNE L1 ; EXIT IF NOT VALID AND NO FIXED DISKS
    CMP AL,47 ; IS TYPE WITHIN VALID RANGE
    JNE L1 ; EXIT WITH NO FIXED DISKS IF NOT VALID
    SHL AX,4 ; ADJUST TYPE TO HIGH NIBBLE
    CMP AL,0F0H ; CHECK FOR EXTENDED DRIVE TYPE BYTE USE
    JNE L2 ; USE DRIVE TYPE 1 --> 14 IF NOT IN USE
    CMP AL,0 ; ALLOW FIXED DISK
    JNE L2 ; WRITE IT BACK
    CMP AL,NOT HF FAIL ; GET EXTENDED TYPE FOR DRIVE D:
    JNE L2 ; FROM CMOS
    CMP AL,0 ; IS TYPE SET TO ZERO
    JNE L2 ; SKIP IF SECOND FIXED DISK NOT VALID
    CMP AL,47 ; IS TYPE WITHIN VALID RANGE
    JNE L2 ; SKIP IF NOT VALID
    SHL AX,4 ; ADJUST TYPE TO HIGH NIBBLE
    CMP AL,0F0H ; CHECK FOR EXTENDED DRIVE TYPE BYTE USE
    JNE L3 ; USE DRIVE TYPE 1 --> 14 IF NOT IN USE
    CMP AL,0 ; ALLOW FIXED DISK
    JNE L3 ; WRITE IT BACK
    CMP AL,NOT HF FAIL ; GET EXTENDED TYPE FOR DRIVE E:
    JNE L3 ; FROM CMOS
    CMP AL,0 ; IS TYPE SET TO ZERO
    JNE L3 ; SKIP IF SECOND FIXED DISK NOT VALID
    CMP AL,47 ; IS TYPE WITHIN VALID RANGE
    JNE L3 ; SKIP IF NOT VALID
    SHL AX,4 ; ADJUST TYPE TO HIGH NIBBLE
    CMP AL,0F0H ; CHECK FOR EXTENDED DRIVE TYPE BYTE USE
    JNE L4 ; USE DRIVE TYPE 1 --> 14 IF NOT IN USE
    CMP AL,0 ; ALLOW FIXED DISK
    JNE L4 ; WRITE IT BACK
    CMP AL,NOT HF FAIL ; GET EXTENDED TYPE FOR SECOND FIXED DISK
    JNE L4 ; FROM CMOS
    CMP AL,0 ; IS TYPE SET TO ZERO
    JNE L4 ; SKIP DRIVE IF NOT A VALID TABLE ENTRY
    CMP AL,47 ; IS TYPE WITHIN VALID RANGE
    JNE L4 ; GET START TIMER COUNTS
    SHL AX,4 ; ADJUST TYPE TO HIGH NIBBLE
    CMP AL,0F0H ; CHECK THE CONTROLLER
    JNE L4 ; USE CONTROLLER DIAGNOSTIC COMMAND
    INT 13H ; CALL BIOS WITH DIAGNOSTIC COMMAND
    CMP AX,0 ; DISPLAY ERROR MESSAGE IF BAD RETURN
    JNE L4 ; GET START TIMER COUNTS
    MOV DL,80H ; SET UP DRIVE 0
    INT 13H ; WERE THERE TWO DRIVES?
    CMP AX,0 ; NO-ALL DONE
    JNE L4 ; SET UP DRIVE 1
    MOV DL,81H ; SET UP OFF DS REGISTER
    CALL HD_RESET_1 ; WORK OFF DS REGISTER
    RET

```

```

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_I:
394 0104 53         PUSH   BX,NEAR           ; SAVE TIMER LIMITS
395 0105 51         PUSH   CX
396 0106 B4 09       MOV    AH,09H           ; SET DRIVE PARAMETERS
397 0108 CD 13       INT    13H
398 010A B4 00       JC    RES_1
399 010C B4 11       MOV    AH,T1H           ; RECALIBRATE DRIVE
400 010E CD 13       INT    13H
401 0110 73 39     JNC    RES_2
402 0112 EB 018A R  CALL   POD_TCHK           ; CHECK TIME OUT
403 0113 73 00     CALL   RES_FL
404 0117 00 0000 E  MOV    SI,OFFSET FIT81      ; INDICATE DISK I FAILURE
405 011A F6 C2 01  TEST   DL,0H
406 011D 75 57       JNZ    RES_EI
407 011F BE 0000 E  MOV    SI,OFFSET FIT80      ; INDICATE DISK 0 FAILURE
408 0122 EB 017C R  CALL   SET_FAIL           ; DO NOT TRY TO IPL DISK 0
409 0125 EB 0000 E  JMP    SHORT RES_E1
410 0127 00 0000 E  RES_RS:
411 0129 CD 13       RES_RS: MOV    AH,00H           ; RESET THE DRIVE
412 012B B4 08       RES_RS: INT    13H
413 012D 8A DA       RES_RS: MOV    AH,08H           ; GET MAX CYLINDER,HEAD,SECTOR
414 012F CD 13       RES_RS: BL,DL           ; SAVE DRIVE CODE
415 0131 72 38       RES_RS: INT    13H
416 0133 00 0000 E  RES_RS: JC    RES_ER
417 0137 8A D3       RES_RS: MOV    WORD PTR @NEC_STATUS,CX ; SAVE MAX CYLINDER, SECTOR
418 0139 BB 0401     RES_RS: MOV    DL,BL           ; RESTORE DRIVE CODE
419 013C CD 13       RES_RS: INT    13H
420 013E T3 39       RES_RS: MOV    AX,0401H          ; VERIFY THE LAST SECTOR
421 0140 80 FC 0A     RES_RS: INT    13H
422 0141 80 FC 11     RES_RS: CMP   AH,BAD_SECTOR
423 0144 80 FC 12     RES_RS: JE    RES_OK
424 0148 80 FC 10     RES_RS: CMP   AH,BAD_ECC
425 014D T4 2A       RES_RS: JE    RES_OK
426 0152 00 0000 E  RES_RS: CALL   POD_TCHK           ; CHECK FOR TIME OUT
427 0154 BB 0000 E  RES_RS: JC    RES_ES
428 0156 8B 00 0042 R RES_RS: MOV    CX,WORD PTR @NEC_STATUS,CX ; FAILED
429 0158 BB 00 0042 R RES_RS: MOV    AL,CL           ; GET SECTOR ADDRESS, AND CYLINDER
430 015A 8A C1       RES_RS: AND   AL,3FH           ; SEPARATE OUT SECTOR NUMBER
431 015A 24 3F       RES_RS: DEC   AL
432 015C FE C8       RES_RS: INT    13H
433 015E T4 CT       RES_RS: AND   AL,0DH           ; TRY PREVIOUS ONE
434 0160 80 0000 E  RES_RS: OR    AL,0DH           ; WE'VE TRIED ALL SECTORS ON TRACK
435 0163 00 0000 E  RES_RS: CLC
436 0165 89 BE 0042 R RES_RS: MOV    WORD PTR @NEC_STATUS,CX ; KEEP CYLINDER BITS
437 0169 EB CE       RES_RS: JMP    RES_3
438 016B BE 0000 E  RES_RS: MOV    SI,OFFSET F1791      ; SAVE CYLINDER, NEW SECTOR NUMBER
439 016E FE C2 01     RES_RS: INT    13H
440 0170 75 03       RES_RS: JC    RES_ER
441 0173 BE 0000 E  RES_RS: MOV    DL,1
442 0176 E8 0000 E  RES_RS: INT    13H
443 0176 E8 0000 E  RES_RS: MOV    SI,OFFSET FIT90      ; INDICATE DISK 0 ERROR
444 0179             RES_RS: CALL   E_MSG              ; DISPLAY ERROR AND SET (BP) ERROR FLAG
445 0179 59         RES_RS: POP    CX
446 017A 5B         RES_RS: POP    BX
447 017B C3         RES_RS: RET
448 017C             HD_RESET_I:
449
450 017C             ENDP
451 017C             SET_FAIL:
452 017C BE BE BE    MOV    AX,X1(CMOS_DIAG+NMI) ; GET CMOS ERROR BYTE
453 017C 00 0000 E  CALL   CMOS_READ           ; SET DO NOT IPL FROM DISK FLAG
454 0182 0C 00       OR    AL,WF_FAIL
455 0184 86 E0       XCHG   AL,WF_FAIL
456 0186 E8 0000 E  CALL   CMOS_WRITE          ; SAVE IT
457 0189 C3         CALL   CMOS_WRITE          ; PUT IT OUT
458 018A             RET
459
460 018A             SET_FAIL:
461 018A 58         POP    AX
462 018B 58         POP    CX
463 018C 5B         POP    BX
464 018D 53         PUSH   BX
465 018E 51         PUSH   CX
466 018F 50         PUSH   AX
467 0190 A1 006C R  MOV    AX,PTIMER_LOW
468
469 0193 3B D9       CMP   BX,CX           ; RESTORE RETURN
470 0194 72 06       CMP   JB,TCHK2          ; AX = CURRENT TIME
471 0197 72 06       CMP   JC,TCHK2          ; BX = START TIME
472 0199 72 0C       CMP   JB,TCHKNG         ; CX = END TIME
473 019B EB 04       JMP    SHORT TCHK2
474 019D 3B C3       TCHK1:  CMP   AX,BX           ; START < END
475 019F 72 04       TCHK1:  JMP    TCHKNG         ; END < START < CURRENT
476 01A1 3B C1       TCHK2:  CMP   AX,CX           ; CURRENT < START < END
477 01A3 72 02       TCHK2:  JMP    TCHKNG         ; OR CURRENT < END < START
478
479 01A5 F9         TCHKNG: STC
480 01A6 C3         RET
481 01A7 F8         TCHKNG: CLC
482 01A8 C3         TCHKNG: RET
483 01A9             POD_TCHK
484
485 01A9             DISK_SETUP

```

```

486          PAGE
487          -----
488          ;----- FIXED DISK BIOS ENTRY POINT -----:
489          ;
490          ;
491 01A9          DISK_10 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:          STI              ; ENABLE INTERRUPTS
500 01B3 FB 00        OR AH, AH
501 01B4 04 E4 00        JNZ A2              ; RESET NEC WHEN AH=0
502 01B5 00 09 00        INT 40H             ; RESET
503 01B8 CD 40          SUB AH, AH
504 01B8 2A E4          CMP DL, (80H + S_MAX_FILE - 1)
505 01B8 80 FA 81  JA RET_2
506 01BF 77 EF          ;
507 01C1 00 00          A2:          CMP AH, 08H          ; GET PARAMETERS IS A SPECIAL CASE
508 01C1 80 FC 08  JNZ A3              ; IN THE STACK, THE COMMAND BLOCK IS:
509 01C4 75 03          A3:          JMP GET_PARM_N
510 01C6 E9 0393 R     CMP AH, T5H          ; READ DASD TYPE IS ALSO
511 01C9 80 FC 15  JNZ A4              ; IN THE STACK, THE COMMAND BLOCK IS:
512 01CC 75 03          A4:          JMP READ_DASD_TYPE
513 01CE E9 0353 R
514          ;
515 01D1          A4:          ENTER B, 0          ; SAVE REGISTERS DURING OPERATION
516 01D1 C8 0008 00  PUSH BX             ; SAVE (BP) AND MAKE ROOM FOR CMD_BLOCK
517 01D5 53          PUSH CX             ; IN THE STACK, THE COMMAND BLOCK IS:
518 01D6 51          PUSH DX             ; *CMD_BLOCK == BYTE PTR [BP]-8
519 01D7 52          PUSH DS             ;
520 01D8 50          PUSH ES             ;
521 01D9 66          PUSH SI             ;
522 01DA 56          PUSH DI             ;
523 01DB 57          OR AH, AH
524 01DC 04 E4 00        A5:          CALL DISK_10_CONT          ; FORCE DRIVE 80 FOR RESET
525 01DE 75 02          MOV DL, 80H          ; PERFORM THE OPERATION
526 01E0 E2 80          CALL DDS             ; ESTABLISH SEGMENT
527 01E1 00 00 225 R    MOV AH, *DISK_STATUS1
528 01E5 E8 0000 E     CMP AH, 1          ; GET STATUS FROM OPERATION
529 01E8 80 26 0074 R    SET CARRY FLAG TO INDICATE
530 01EC 80 FC 01  CMP CMC             ; SUCCESS OR FAILURE
531 01EF F5          POP DI             ; RESTORE REGISTERS
532 01F0 5F          POP SI             ;
533 01F1 5E          POP ES             ;
534 01F2 07          POP DS             ;
535 01F3 1F          POP DX             ;
536 01F4 5A          POP CX             ;
537 01F5 59          POP BX             ;
538 01F6 5B          POP AX             ;
539 01F7 C9          LEAVE AX          ; ADJUST (SP) AND RESTORE (BP)
540 01F8 CA 0002  RET 2          ; THROW AWAY SAVED FLAGS
541 01FB          DISK_10 ENDP
542          ;
543 01FB          M1  LABEL WORD
544 01FB 0000 R     DW DISK_RESET          ; FUNCTION TRANSFER TABLE
545 01FD 0315 R     DW RETURN_STATUS
546 01FF 031E R     DW DISK_READ
547 0201 0325 R     DW DISK_WRITE
548 0203 032C R     DW DISK_VRF
549 0205 033E R     DW FMT_TRK
550 0207 0289 R     DW BAD_COMMAND          ; 006H FORMAT BAD SECTORS
551 0209 0289 R     DW BAD_COMMAND          ; 007H FORMAT DRIVE
552 020B 0289 R     DW BAD_COMMAND          ; 008H RETURN PARAMETERS
553 020D 03F1 R     DW INIT_DRY
554 020F 0423 R     DW RD_LONG
555 0211 0424 R     DW WR_LONG
556 0213 0431 R     DW DISK_SEEK
557 0215 0431 R     DW DISK_SET
558 0217 0289 R     DW BAD_COMMAND          ; 00EH READ BUFFER
559 0219 0289 R     DW BAD_COMMAND          ; 00FH WRITE BUFFER
560 021B 044F R     DW TST_RDY
561 021D 0466 R     DW HD15_RECAL
562 021F 0289 R     DW BAD_COMMAND          ; 012H MEMORY DIAGNOSTIC
563 0221 0289 R     DW BAD_COMMAND          ; 013H DRIVE DIAGNOSTIC
564 0223 046E R     DW CTL_DIAGNOSTIC      ; 014H CONTROLLER DIAGNOSTIC
565 002A          M1L  EQU $-M1
566          ;
567 0225          DISK_10_CONT PROC NEAR
568 0225 E8 0000 E    CALL DDS             ; ESTABLISH SEGMENT
569 0225 80 F0 01  CMP AH, 01H          ; RETURN STATUS
570 022B 75 03  JNZ SUH              ;
571 022D E9 0315 R    JMP RETURN_STATUS
572 0230          SUH:          ;
573 0230 C6 06 0074 R 00  MOV *DISK_STATUS1, 0          ; RESET THE STATUS INDICATOR
574 0235 53          PUSH BX             ; SAVE DATA ADDRESS
575 0235 00 00 1E 0075 R  MOV BX, *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          PUSH ES             ;
581 0243 E8 0687 R    CALL GET_VEC          ; GET DISK PARAMETERS
582 0243 26 41 8A 47 05  MOV AX, WORD PTR ES:[BX][5] ; GET WRITE PRE-COMPENSATION CYLINDER
583 0244 C1 E8 02  SHR AX, 2
584 0240 88 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 82          PUSH DX             ;
587 0254 00 00 03F6  MOV DX, *HF_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_BYT
592 025B 80 E4 C0  AND AH, 00CH          ; SET EXTRA HEAD OPTION IN
593 0262 8A E0 00        OR AH, AL
594 0264 88 26 0076 R  MOV *CONTROL_BYT, AH
595 0268 58          POP AX             ;
596 0269 88 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,CF
603 0279 C0 E0 06      SHR    AL,6
604 027C 88 4C FC      MOV    @CMD_BLOCK+,AL      ; CYLINDER HIGH ORDER 2 BITS
605 027F 80 C0          MOV    AL,DL      ; DRIVE NUMBER
606 0280 80 00 04      SHL    AL,4
607 0284 80 E6 0F      AND    DH,0FH      ; HEAD NUMBER
608 0287 80 C6 0F      OR     AL,80H OR 20H      ; ECC AND 512 BYTE SECTORS
609 0289 80 A0          OR     AL,00H      ; ECC/SIZE/DRIVE/HEAD
610 028B 88 4C FD      MOV    @CMD_BLOCK+5,AL
611 028E 80 00          POP    AX
612 0290 80 00          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 8B F0          MOV    SI,AX      ; PUT INTO SI FOR BRANCH
617 0298 3D 002A      CMP    AX,MIL      ; TEST WITHIN RANGE
618 0299 73 1A          JNB    BAD_COMMAND_POP
619 029D 5A 00          POP    AX      ; RESTORE AX
620 029E 5B 00          POP    BX      ; AND DATA ADDRESS
621 029F 51 00          PUSH   CX
622 02A0 50 00          PUSH   AX      ; ADJUST ES:BX
623 02A1 88 CB          MOV    CX,BX      ; GET 3 HIGH ORDER NIBBLES OF BX
624 02A2 80 C9 04      SHR    CX,4
625 02A6 8C C0          MOV    AL,ES5
626 02A8 03 C1          ADD    AX,CX
627 02AA 8E C0          MOV    ES,AX
628 02AC 81 E0 000F      AND    BX,000FH      ; ES:BX CHANGED TO ES:000X
629 02B0 58 00          POP    AX
630 02B1 59 00          POP    CX
631 02B2 2E 1F A4 01FB R  JMP    WORD PTR CS:[SI + OFFSET MI]
632 02B7 BAD_COMMAND_POP:
633 02B7 58 00          POP    AX
634 02B8 5B 00          POP    BX
635 02B9 BAD_COMMAND:
636 02B9 80 C6 06 0074 R 01  MOV    @DISK_STATUS1,BAD_CMD      ; COMMAND ERROR
637 02B8 B0 00          MOV    AL,0
638 02C0 C3 00          RET
639 02C1 DISK_IO_CONT      ENDP

640
641
642
643
644
645 02C1 DISK_RESET      PROC   NEAR
646 02C1 FA 00          CLD
647 02C2 E4 A1          IN     AL,INTB01      ; GET THE MASK REGISTER
648 02C3 EB 00          JMP    $+2
649 02C4 80 BF          AND    AL,0BFH      ; ENABLE FIXED DISK INTERRUPT
650 02C6 E6 A1          OUT    INTB01,AL
651 02CA FB 00          STI
652 02CB B0 04          MOV    AL,04H
653 02CD BA 03F6          MOV    DX,HF_REG_PORT
654 02D0 80 00          OUT    DX,AL      ; RESET
655 02D1 80 B9 000A      MOV    CX,10      ; DELAY COUNT
656 02D4 49 00          DEC    CX
657 02D5 75 FD          DRD
658 02D7 A0 0076 R      MOV    AL,0CONTROL_BYTE
659 02D8 24 0F          AND    AL,0FH      ; SET HEAD OPTION
660 02D9 80 00          OUT    AL,1      ; TURN RESET OFF
661 02D9 E8 05E6 R      CALL   NOT_BUSY
662 02E0 75 2D          JNZ    DRERR      ; TIME OUT ON RESET
663 02E2 BA 01F1          MOV    DX,HF_PORT+1
664 02E5 EC 00          IN     AL,DX      ; GET RESET STATUS
665 02E6 3C 01          CMP    AL,1
666 02E8 75 05          JNZ    DRERR      ; BAD RESET STATUS
667 02E9 80 66 FD EF    AND    @CMD_BLOCK+5,0EFH
668 02EE 2A 02          SUB    DL,DL      ; SET TO DRIVE 0
669 02F0 E8 03F1 R      CALL   INIT_DRV      ; SET MAX HEADS
670 02F3 E8 0466 R      CALL   HDISK_RECAL      ; RECAL TO RESET SEEK SPEED
671 02F6 80 3E 0075 R 01  CMP    @HFI_NUM,1
672 02F8 80 00 00          JBE    DRE,DRERR
673 02F9 80 66 FD 10      OR     @CMD_BLOCK+5,010H
674 0301 B2 01          MOV    DL,1      ; SET TO DRIVE 1
675 0303 E8 03F1 R      CALL   INIT_DRV      ; SET MAX HEADS
676 0306 E8 0466 R      CALL   HDISK_RECAL      ; RECAL TO RESET SEEK SPEED
677 0309 C6 06 0074 R 00  DRE:  MOV    @DISK_STATUS1,0      ; IGNORE ANY SET UP ERRORS
678 030E C3 00          RET
679 030F C6 06 0074 R 05  DRERR: MOV    @DISK_STATUS1,BAD_RESET      ; CARD FAILED
680 0314 C9 00          RET
681 0315 00 00          DISK_RESET      ENDP

682
683
684
685
686
687 0315 RETURN_STATUS      PROC   NEAR
688 0315 A0 0074 R      MOV    @DISK_STATUS1      ; OBTAIN PREVIOUS STATUS
689 0318 C6 06 0074 R 00  MOV    @DISK_STATUS1,0      ; RESET STATUS
690 031D C3 00          RET
691 031E RETURN_STATUS      ENDP

```

```

692          PAGE
693          ;----- DISK READ ROUTINE (AH = 02H) :-----;
694
695
696
697 031E      DISK_READ    PROC    NEAR
698 031E C6 46 FE 20  MOV    0CMD_BLOCK+6,READ_CMD
699 0322 E9 04C6 R   JMP    COMMANDO
700 0325      DISK_READ    ENDP
701
702
703
704
705
706 0325      DISK_WRITE   PROC    NEAR
707 0325 C6 46 FE 30  MOV    0CMD_BLOCK+6,WRITE_CMD
708 0329 E9 0505 R   JMP    COMMANDO
709 032C      DISK_WRITE   ENDP
710
711
712
713
714
715 032C      DISK_VERIFY  PROC    NEAR
716 032C C6 46 FE 40  MOV    0CMD_BLOCK+6,VERIFY_CMD
717 0330 E8 054F R   CALL   COMMAND
718 0331 E9 054F R   JNZ    VERF_EXIT      ; CONTROLLER STILL BUSY
719 0335 E8 0585 R   CALL   WAIT
720 0338 75 03      JNZ    VERF_EXIT      ; TIME OUT
721 033A E8 0623 R   CALL   CHECK_STATUS
722 033D
723 033D C3
724 033E
725
726
727
728
729
730 033E      FMT_TRK    PROC    NEAR      ; FORMAT TRACK (AH = 005H)
731 033E C6 46 FE 50  MOV    0CMD_BLOCK+6,FMTTRK_CMD
732 0342 06
733 0343 53
734 0344 E8 06B7 R   PUSH   ES
735 0347 26: 8A 47 0E  CALL   GET_VEC      ; GET DISK PARAMETERS ADDRESS
736 0348 00 46 F9    MOV    AL,ES:[BX][14]  ; GET SECTORS/TRACK
737 034E E8 05B9      MOV    0CMD_BLOCK+1,AL  ; SET SECTOR COUNT IN COMMAND
738 034F 07
739 0350 E9 050A R   POP    BX
740 0353      FMT_TRK    ENDP      ; GO EXECUTE THE COMMAND
741
742
743
744
745
746
747 0353      READ_DASD_TYPE  LABEL   NEAR
748 0353      READ_D_    PROC    FAR      ; GET DRIVE PARAMETERS
749 0353 1E      PUSH   DS      ; SAVE REGISTERS
750 0354 06
751 0355 53
752
753 0356 E8 0000 E   ASSUME DS:DATA
754 0357 C6 06 0074 R 00  CALL   DD1      ; ESTABLISH ADDRESSING
755 035E 8A 0E 0075 R   MOV    DL,DSK_STATUS1,0  ; GET NUMBER OF DRIVES
756 0362 80 E2 7F      MOV    BL,0HF_NUM    ; GET DRIVE NUMBER
757 0365 3A DA
758 0367 T6 22
759 0369 E8 06B7 R   AND    DL,TFN
760 0370 26: 8A 47 02  CMP    DL,DL
761 0370 26: 8A 4F 0E  JBE    RDT_NOT_PRESENT  ; RETURN DRIVE NOT PRESENT
762 0374 F6 E9
763 0376 26: 8B 0F
764 0379 49
765 037A F6 E9
766 037B BB AA
767 037E BB D0
768 0380 BB C0
769 0382 B4 03
770 0384 5B
771 0385 07
772 0386 1F
773 0387 F6
774 0388 CA 0002
775 0388      RDT2:      SUB    AX,AX      ; INDICATE FIXED DISK
776 038B BB C0      POP    BX      ; RESTORE REGISTERS
777 038D BB C8
778 038F BB D0
779 0391 EB F1      POP    ES
780 0393      READ_D_T    ENDP

```

```

781          PAGE
782          ;-----[AH = 08H]-----
783          ;-- GET PARAMETERS (AH = 08H) :
784          ;-----[AH = 08H]-----
785
786 0393          GET_PARM_N  LABEL  NEAR
787 0393          GET_PARM_  PROC   FAR
788 0393  IE      PUSH   DS
789 0394 06      PUSH   ES
790 0395 53      PUSH   BX
791          ASSUME DS:ABSO
792 0396 88  ---- R  MOV    DS,ABSO
793 0399 8E D8  MOV    DS,AX
794 039B F6 C2 01 TEST   DL,1
795 039E 74 06  JZ    G0
796 03A0 C4 1E 0118 R  LES    BX,0HF1_TBL_VEC
797 03A4 EB 04  JMP   BX,0HF1_TBL_VEC
798 03A6 C4 1E 0104 R  G01   LES    BX,0HF2_TBL_VEC
799          ASSUME DS:DATA
800 03AA          G1:   ASSUME DS:DATA
801 03AA 08 0000 E  CALL   DDS
802 03AD 80 EA 80  SUB    DL,80H
803 03B0 80 00 02  CMP    DL,MAX_FILE
804 03B3 13 0C  JAE   G1
805 03B5 C6 06 0074 R 00  MOV    DS,DISK_STATUS1,0
806 03B8 26 0B 07  MOV    AX,EST[BX]
807 03BD 2D 0002  SUB    AX,2
808 03C0 8A EA  MOV    CH,AL
809 03C2 25 0300  AND   AX,0300H
810 03C5 00 0000 E  SHR    AX,2
811 03C7 D1 E8  SHR    AX,1
812 03C9 26 0A 47 0E OR    AL,ES:[BX][14]
813 03CD 8A C0  MOV    CL,AL
814 03CF 26 0A 77 02 MOV    DH,ES:[BX][2]
815 03D0 F0 CE  DEC   DH
816 03D5 8A 16 0075 R  MOV    DH,0-N RANGE
817 03D9 28 C0  MOV    DH,0-N RANGE
818 03DB          G5:   SUB    AX,AX
819 03DB 5B  POP   BX
820 03DC 07  POP   ES
821 03D9 1F  POP   DS
822 03D9 CA 0002  RET   2
823 03E1          G4:   ASSUME DS:DATA
824 03E1 C6 06 0074 R 07  MOV    DS,DISK_STATUS1,INIT_FAIL ; OPERATION FAILED
825 03E6 B4 07  MOV    AH,INIT_FAIL
826 03E8 2A C0  SUB    AL,AL
827 03EA 2B D0  SUB    DL,DX
828 03EB 2A C9  SUB    CX,CX
829 03EE F9  STC
830 03EF EB EA  JMP   G5 ; SET ERROR FLAG
831 03F1          GET_PARM ENDP
832
833
834
835          ;-----[AH = 09H]-----
836 03F1          INIT_DRV  PROC   NEAR
837 03F1 C6 46 FE 91  MOV    ECMD_BLOCK+6,SET_PARM_CMD
838 03F5 E8 06B7 R  CALL   GET_VEC
839 03F8 26 0A 47 02  ; ES:BX -> PARAMETER BLOCK
840 03F9 00 0000 E  MOV    AL,ES:[BX][2]
841 03F9 00 0000 E  DEC   AL
842 03F9 00 0000 E  MOV    AH,ECMD_BLOCK+5
843 0400 80 EA F0  AND   AH,0F0H
844 0406 88 66 FD  OR    AH,AL
845 0409 26 0A 47 0E  MOV    ECMD_BLOCK+5,AH
846 0409 00 0000 E  MOV    AL,ES:[BX][14]
847 0410 28 C0  SUB    AX,AL
848 0412 88 46 FB  MOV    ECMD_BLOCK+3,AL
849 0415 E8 054F R  CALL   COMMAND
850 0418 75 08  JNZ   INIT_EXIT
851 041B E8 05E6 R  CALL   NOT_BUSY
852 041D 75 03  JNZ   INIT_EXIT
853 041F E8 0623 R  CALL   CHECK_STATUS
854 0422 C3  INIT_EXIT: RET
855 0423          INIT_DRV ENDP
856
857
858
859          ;-----[AH = 0AH]-----
860
861
862 0423          RD_LONG  PROC   NEAR
863 0423 C6 46 FE 22  MOV    ECMD_BLOCK+6,READ_CMD OR ECC_MODE
864 0427 E9 04C6 R  JMP   COMMAND1
865 042A          RD_LONG ENDP
866
867
868
869          ;-----[AH = 0BH]-----
870
871 042A          WR_LONG  PROC   NEAR
872 042A C6 46 FE 32  MOV    ECMD_BLOCK+6,WRITE_CMD OR ECC_MODE
873 042E E8 0505 R  JMP   COMMAND0
874 0431          WR_LONG ENDP
875
876
877
878          ;-----[AH = 0CH]-----
879
880 0431          DISK_SEEK PROC   NEAR
881 0431 C6 46 FE 70  MOV    ECMD_BLOCK+6,SEEK_CMD
882 0435 28 054F R  CALL   COMMAND
883 0438 75 14  JNZ   DS_EXIT
884 043A E8 05B5 R  CALL   WATT
885 043D 75 0F  JNZ   DS_EXIT
886 043F E8 0623 R  CALL   DS_EXIT
887 0442 80 3E 0074 R 40  CMP   DS_EXIT
888 0442 80 3E 0074 R 40  JNE   DS_EXIT
889 0449 C6 06 0074 R 00  MOV    DS_EXIT,0
890 044E C3  DS_EXIT: RET
891 044E C3
892
893 044F          DISK_SEEK ENDP

```

```

894 PAGE
895 ;----- TEST DISK READY (AH = 10H) :-----;
896
897
898
899 044F TST_RDY PROC NEAR
900 044E EB 05E6 R CALL NOT_BUSY ; WAIT FOR CONTROLLER
901 0452 75 11 JNZ TR_EX
902 0454 BA 46 FD MOV AL,00H ; SELECT DRIVE
903 0457 BA 01F6 MOV DX,HF_PORT+6
904 045A EE OUT DX,AL
905 045B EB 0635 R CALL CHECK_ST ; CHECK STATUS ONLY
906 045E 75 05 JNZ TR_EX
907 0460 C6 06 0074 R 00 MOV DX,AL
908 0465 C3 MOV @DTSK_STATUS1,0 ; WIPE OUT DATA CORRECTED ERROR
909 0466
910 TST_RDY ENDP
911
912 ;----- RECALIBRATE (AH = 11H) :-----;
913
914
915 0466 HDISK_RECAL PROC NEAR
916 0466 C6 46 FE 10 MOV @CMD_BLOCK+6,RECAL_CMD ; START THE OPERATION
917 046A EB 054F R CALL COMMAND ; ERROR?
918 046D 75 19 JNZ RECAL_EXIT ; FOR COMPLETION
919 046F EB 05B5 R CALL WAIT ; TIME OUT ONE OK ?
920 0472 75 05 JZ RECAL_X ; WAIT FOR ONE LONGER
921 0474 EB 05B5 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 0623 R RECAL_X:
925 047C 80 3E 0074 R 40 CALL CHECK_STATUS ; SEEK NOT COMPLETE
926 0480 75 05 JNE RECAL_EXIT ; IS OK
927 0483 C6 06 0074 R 00 MOV @DISK_STATUS1,0
928 0488 80 3E 0074 R 00 RECAL_EXIT:
929 048B 80 3E 0074 R 00 CMP @DISK_STATUS1,0
930 048D C3 RET
931 048E HDISK_RECAL ENDP
932
933
934
935
936
937
938 048E CTR_DIAGNOSTIC PROC NEAR
939 048F FA CLI ; DISABLE INTERRUPTS WHILE CHANGING MASK
940 0491 E4 A1 IN AL,INTB01 ; TURN ON SECOND INTERRUPT CHIP
941 0493 EB 00 AND AL,0BFH
942 0495 E6 A1 JMP $+2
943 0497 E4 21 OUT INTB01,AL
944 0499 E4 FB IN AL,INTA01 ; LET INTERRUPTS PASS THRU TO
945 049B EB 00 AND AL,0FBH ; SECOND CHIP
946 049D E6 21 JMP $+2
947 049F FB OUT INTA01,AL
948 04A0 EB 05E6 R STI
949 04A3 75 1A CALL NOT_BUSY ; WAIT FOR CARD
950 04A5 EB 01F7 CD_ERR ; BAD CARD
951 04A8 BA 90 IN DX,HF_PORT+7
952 04AA EB EE MOV AL,DIG_CMD
953 04AB EB 05E6 R OUT DX,AL
954 04AE BA 80 CALL NOT_BUSY ; START DIAGNOSE
955 04B1 75 0F MOV AH,TIME_OUT
956 04B2 B9 01F1 JNZ CD_EXIT ; TIME OUT ON DIAGNOSTIC
957 04B5 EC IN DX,PORT+I ; GET ERROR REGISTER
958 04B6 A2 008D R MOV @HF_ERROR,AL ; SAVE IT
959 04B9 B4 00 MOV AH,0
960 04BB 3C 01 CMP AL,1 ; CHECK FOR ALL OK
961 04C0 74 02 JE SHORT CD_EXIT
962 04BF B4 20 CD_EXIT:
963 04C1 B8 26 0074 R MOV AH,BAD_CNTL
964 04C5 C3 RET
965 04C6
966
967 CTR_DIAGNOSTIC ENDP
968
969 ;----- COMMAND1 :-----;
970 ;----- REPEATEDLY INPUTS DATA TILL :-----;
971 ;----- NSECTOR RETURNS ZERO :-----;
972
973 04C6 COMMAND1
974 04C6 EB 0694 R CALL CHECK_DMA ; CHECK 64K BOUNDARY ERROR
975 04C9 72 39 JC CMD_ABORT
976 04CB BB FB MOV DI,BX
977 04CD EB 054F R CALL COMMAND ; OUTPUT COMMAND
978 04D0 75 32 JNZ CMD_ABORT
979 04D2
980 04D3 EB 05B5 R CALL WAIT ; WAIT FOR DATA REQUEST INTERRUPT
981 04D5 75 20 JNZ TM_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 04E0 75 6D CMP INSW ; GET THE SECTOR
987 04E1 FB TEST @CMD_BLOCK+6,ECC_MODE ; CHECK FOR NORMAL INPUT
988 04E2 F6 46 FE 02 JZ CMD_T3
989 04E6 74 12 CALL WAIT_DRQ ; WAIT FOR DATA REQUEST
990 04EB E8 060D R JC TM_OUT
991 04EB B9 017 MOV CX,0
992 04F0 B9 01F0 MOV DX,HF_PORT
993 04F0 B9 0004 MOV CX,0
994 04F3 EC CALL IN ; GET ECC BYTES
995 04F4 26: 88 05 MOV ES:BYTE PTR [DI],AL ; GO SLOW FOR BOARD
996 04F7 47 IN DI
997 04F8 E8 F9 CALL CMD_12 ; SLOW DOWN
998 04F9 75 0232 R CMP_DG_STATUS ; 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: RET
1004 0504 C3

```



```

1109          PAGE
1110          -----
1111          -----
1112          -----
1113 05B5          WAIT PROC NEAR
1114 05B5 FB        STI          ; MAKE SURE INTERRUPTS ARE ON
1115 05B6 2B C9      SUB CX,CX   ; SET INITIAL DELAY BEFORE TEST
1116 05B8 F8        CLC
1117 05B9 BB 9000    MOV AX,9000H ; DEVICE WAIT INTERRUPT
1118 05B9 CD 15      INT 15H
1119 05B8 72 0F      JC  WT2    ; DEVICE TIMED OUT
1120          -----
1121 05C0 B3 25      MOV BL,DELAY_1 ; SET DELAY COUNT
1122          -----
1123          -----
1124          -----
1125 05C2 F6 06 008E R 80  WT1: TEST 0HF_INT_FLAG,80H ; TEST FOR INTERRUPT
1126 05D4 EB 0A      LOOPZ WT1
1127 05C9 75 0B      JNZ WT3   ; INTERRUPT--LETS GO
1128 05CB FE CB      DEC BL
1129 05C0 75 F3      JNZ WT1   ; KEEP TRYING FOR A WHILE
1130          -----
1131 05C5 C6 06 0074 R 80  WT2: MOV 0DISK_STATUS1,TIME_OUT ; REPORT TIME OUT ERROR
1132 05D4 EB 0A      SHORT_W4
1133 05D6 C6 06 0074 R 00  WT3: MOV 0DISK_STATUS1,0
1134 05D8 C6 06 008E R 00
1135 05E0 80 3E 0074 R 00  WT4: CMP 0HF_INT_FLAG,0
1136 05E1 C3        RET
1137 05E4          ENDP
1138          -----
1139          -----
1140          -----
1141          -----
1142 05E6          NOT_BUSY PROC NEAR
1143 05E6 FB        STI          ; MAKE SURE INTERRUPTS ARE ON
1144 05E7 53        PUSH BX
1145 05E8 B3 25      MOV BL,DELAY_1 ; SET INITIAL DELAY BEFORE TEST
1146 05E8 2B C9      SUB CX,CX
1147 05E9 01F7      IN AL,DX
1148 05E9 00        AL,DX_PORT+7
1149 05F0 A8 80      NB1: TEST AL,DX
1150 05F2 E0 FB      LOOPNZ NB1 ; CHECK STATUS
1151 05F4 74 0B      JZ NB2   ; NOT BUSY--LETS GO
1152 05F6 FE CB      DEC BL
1153 05F8 75 F5      JNZ NB1   ; KEEP TRYING FOR A WHILE
1154          -----
1155 05FA C6 06 0074 R 80  MOV 0DISK_STATUS1,TIME_OUT ; REPORT TIME OUT ERROR
1156 05FF EB 05      JMP SHORT_NB3
1157 0601 C6 06 0074 R 00  NB2: MOV 0DISK_STATUS1,0
1158 0606 5B        POP BX
1159 0607 80 3E 0074 R 00  CMP 0DISK_STATUS1,0
1160 060C C3        RET
1161 060D          ENDP
1162          -----
1163          -----
1164          -----
1165          -----
1166 060D          WAIT_DRQ PROC NEAR
1167 060D B9 0100    MOV CX,DELAY_3
1168 0610 BA 01F7    MOV DX,DX_PORT+7
1169 0613 EC        WQ_I: IN AL,DX
1170 0614 A8 08      TEST AL,ST_DRQ ; GET STATUS
1171 0616 75 09      JNZ WQ_OK ; WAIT FOR DRQ
1172 0617 E0 F9      LOOP WQ_I
1173 061A 66 0074 R 80  MOV 0DISK_STATUS1,TIME_OUT ; KEEP TRYING FOR A SHORT WHILE
1174 061F F9        STC
1175 0620 C3        RET
1176 0621 F8        WQ_OK: CLC
1177 0622 C3        RET
1178 0623          ENDP
1179          -----
1180          -----
1181          -----
1182 0623          CHECK_STATUS PROC NEAR
1183 0623 E8 0635 R  CALL CHECK_ST ; CHECK THE STATUS BYTE
1184 0624 75 07      JNZ CHECK_ST ; AN ERROR WAS FOUND
1185 0628 A8 01      TEST AL,ST_ERROR ; WHERE THERE ANY OTHER ERRORS
1186 062A 74 03      JZ CHECK_SI ; NO ERROR REPORTED
1187 062C E8 0669 R  CALL CHECK_ER ; ERROR REPORTED
1188 062F          CMP 0DISK_STATUS1,0 ; SET STATUS FOR CALLER
1189 062F 80 3E 0074 R 00
1190 0634 C3        RET
1191 0635          ENDP
1192          -----
1193          -----
1194          -----
1195 0635          CHECK_ST PROC NEAR
1196 0635 BA 01F7    MOV DX,DX_PORT+7 ; GET THE STATUS
1197 0638 EC        IN AL,DX
1198 0639 A2 008C R  MOV AH,0
1199 063C B4 00      MOV 0HF_STATUS,AL
1200 063E A8 80      TEST AL,ST_BUSY ; IF STILL BUSY
1201 0640 75 1A      JNZ CKST_EXIT ; REPORT OK
1202 0642 04 0C      MOV AH,WRTE_FAULT
1203 0644 A8 20      TEST AL,ST_WRTFLT ; CHECK FOR WRITE FAULT
1204 0646 75 14      JNZ CKST_EXIT
1205 0648 B4 AA      MOV AH,NOT_RDY
1206 064A A8 40      TEST AL,ST_READY ; CHECK FOR NOT READY
1207 064C 74 0E      JZ CKST_EXIT
1208 064E 00 00      MOV AH,BAD_SEEK
1209 0650 A8 10      TEST AL,ST_SEEK_COMPL ; CHECK FOR SEEK NOT COMPLETE
1210 0652 74 08      JZ CKST_EXIT
1211 0654 B4 11      MOV AH,DATA_CORRECTED
1212 0656 A8 04      TEST AL,ST_CORRCTD ; CHECK FOR CORRECTED ECC
1213 0658 75 02      JNZ CKST_EXIT
1214 065A B4 00      MOV AH,0
1215 065C          CKST_EXIT: RET
1216 065C 88 26 0074 R  MOV 0DISK_STATUS1,AH ; SET ERROR FLAG
1217 0660 80 FC 11    CMP AH,DATA_CORRECTED
1218 0663 74 03      JZ CKST_EXIT ; KEEP GOING WITH DATA CORRECTED
1219 0665 80 FC 00    CMP AH,0
1220          -----
1221 0668 C3        CKST_EXIT: RET
1222 0669          ENDP

```

```

1223
1224
1225
1226
1227 0669
1228 0669 BA 01F1
1229 066C EC
1230 066D A2 008D R
1231 0670 53
1232 0671 BB 0008
1233 0674 00 E0
1234 0676 72 02
1235 0678 E2 FA
1236 067A BB 0688 R
1237 067D 03 D9
1238 0681 2E 8A 27
1239 0682 00 26 0074 R
1240 0686 5B
1241 0687 80 FC 00
1242 068A C3
1243 068B E0
1244 068C 02 40 01 BB
1245 0690 04 BB 10 0A
1246 0694
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257 0694
1258 0694 50
1259 0695 B8 8000
1260 0698 F6 46 FE 02
1261 069C 74 03
1262 069E BB 7F04
1263 06A0 77 66 F9
1264 06A4 77 66 F9
1265 06A6 72 07
1266 06A8 3A C3
1267 06AA 72 03
1268 06AC F8
1269 06AD 68
1270 06AE C3
1271 06AF F9
1272 06B0 C6 06 0074 R 09
1273 06B5 58
1274 06B6 C3
1275 06B7
1276
1277
1278
1279
1280 06B7
1281 06B7 2B C0
1282 06B9 BE C0
1283
1284 06BB F6 C2 01
1285 06BE T4 07
1286 06C0 26: C4 IE 0118 R
1287 06C1 EB 05
1288 06C1
1289 06C7 26: C4 IE 0104 R
1290 06CC
1291 06CC C3
1292 06CD
1293
1294
1295
1296
1297
1298
1299
1300 06CD
1301 06CD 50
1302 06CE 1E
1303 06CF E8 0000 E
1304 06D2 C6 06 008E R FF
1305 06D3 00 20
1306 06D9 E6 00
1307 06DB EB 00
1308 06DD E6 20
1309 06DF 1F
1310 06E0 FB
1311 06E1 BB 9100
1312 06E4 CD 15
1313 06E6 58
1314 06E7 CF
1315
1316 06E8
1317
1318 06E8 30 36 2F 31 30 2F
1319 38 35
1320 06F0
1321

PAGE
;---- CHECK FIXED DISK ERROR REGISTER :----;
;---- PROC NEAR :----;
;---- GET THE ERROR REGISTER :----;
;---- IN AL,DX :----;
;---- MOV @HF_ERROR,AL :----;
;---- PUSH BX :----;
;---- MOV CX,8 :----;
;---- TEST ALL 8 BITS :----;
;---- MOV AL,1 :----;
;---- JC CK2 :----;
;---- CK1 :----;
;---- LOOP CK1 :----;
;---- KEEP TRYING :----;
;---- ADD BX,CX :----;
;---- COMPUTE ADDRESS OF :----;
;---- MOV AH,BYTE PTR CS:[BX] :----;
;---- ERROR CODE :----;
;---- GET ERROR CODE :----;
;---- MOV @SK_STATUS1,AH :----;
;---- SAVE ERROR CODE :----;
;---- POP BX :----;
;---- CMP AH,0 :----;
;---- RET :----;
;---- NO_ERR :----;
;---- BAD_ADDR_MARK,BAD_SEEK,BAD_CMD,UNDEF_ERR :----;
;---- RECORD_NOT_FOUND,UNDEF_ERR,BAD_ECC,BAD_SECTOR :----;
;---- ENDP :----;
;---- PROC NEAR :----;
;---- TEST @CMD_BLOCK+6,ECC_MODE :----;
;---- JZ CKD1 :----;
;---- CKD1 :----;
;---- CMP AX,7F04H :----;
;---- ECC IS 4 MORE BYTES :----;
;---- TEST AX,@CMD_BLOCK+1 :----;
;---- AH = MAX # SECTORS :----;
;---- NUMBER OF SECTORS :----;
;---- JZ CKDOK :----;
;---- IT WILL FIT :----;
;---- JB CKDERR :----;
;---- TOO MANY :----;
;---- CMP AL,BL :----;
;---- CHECK OFFSET ON MAX SECTORS :----;
;---- JB CKDERR :----;
;---- ERROR :----;
;---- JB CKDERR :----;
;---- CLEAR CARRY :----;
;---- POP AX :----;
;---- RET :----;
;---- NORMAL RETURN :----;
;---- CKDERR :----;
;---- STC :----;
;---- INDICATE ERROR :----;
;---- MOV AX @DISK_STATUS1,DMA_BOUNDARY :----;
;---- POP AX :----;
;---- RET :----;
;---- ENDP :----;
;---- SET UP ES:BX-> DISK PARMs :----;
;---- PROC NEAR :----;
;---- SUB AX,AX :----;
;---- MOV BX,AX :----;
;---- GET DISK PARAMETER ADDRESS :----;
;---- ASSUME ES:AB50 :----;
;---- TEST DL,1 :----;
;---- JZ GV_0 :----;
;---- LES BX,@HF_I_TBL_VEC :----;
;---- ES:BX -> DRIVE PARAMETERS :----;
;---- JMP SHORT GV_EXIT :----;
;---- GV_0 :----;
;---- LES BX,@HF_TBL_VEC :----;
;---- ES:BX -> DRIVE PARAMETERS :----;
;---- GV_EXIT :----;
;---- RET :----;
;---- GET_VEC ENDP :----;
;--------- HARDWARE INT 76H -- ( IRQ LEVEL 14 ) -----;
;--------- FIXED DISK INTERRUPT ROUTINE -----;
;--------- PROC NEAR :----;
;--------- CALL DDS :----;
;--------- MOV @HF_INT_FLAG,0FFH :----;
;--------- ALL DONE :----;
;--------- MOV AX,EO1 :----;
;--------- NON-SPECIFIC END OF INTERRUPT :----;
;--------- OUT INTA00,AL :----;
;--------- FOR CONTROLLER #2 :----;
;--------- JMP $+2 :----;
;--------- WAIT :----;
;--------- OUT INTA00,AL :----;
;--------- FOR CONTROLLER #1 :----;
;--------- POP DS :----;
;--------- ST1 :----;
;--------- MOV AX,9100H :----;
;--------- RE-ENABLE INTERRUPTS :----;
;--------- INT 15H :----;
;--------- DEVICE POST :----;
;--------- POP AX :----;
;--------- INTERRUPT :----;
;--------- IRET :----;
;--------- RETURN FROM INTERRUPT :----;
;--------- HD_INT ENDP :----;
;--------- DB '06/10/85' :----;
;--------- RELEASE MARKER :----;
;--------- CODE ENDS :----;
;--------- END :----;

```

```

1 PAGE 116,121
2 TITLE KYBD ----- 06/10/85 KEYBOARD BIOS
3 .LIST
4 0000 CODE SEGMENT BYTE PUBLIC
5
6 PUBLIC K16
7 PUBLIC KEYBOARD_IO_1
8 PUBLIC KB_INT
9 PUBLIC SND_DATA
10
11 EXTRN BEEP:NEAR
12 EXTRN DDS:NEAR
13 EXTRN START_1:NEAR
14 EXTRN K10:BYTE
15 EXTRN K11:BYTE
16 EXTRN K12:BYTE
17 EXTRN K13:BYTE
18 EXTRN K14:BYTE
19 EXTRN K15:BYTE
20 EXTRN K6:BYTE
21 EXTRN K7:BYTE
22 EXTRN K8:BYTE
23 EXTRN K9:BYTE
24
25
26
27
28 ; KEYBOARD I/O
29 ; THESE ROUTINES PROVIDE READ KEYBOARD SUPPORT
30 ; INPUT
31 ; (AH)= 00H READ THE NEXT ASCII CHARACTER ENTERED FROM THE KEYBOARD,
32 ; RETURN THE RESULT IN (AL), SCAN CODE IN (AH).
33 ; (AH)= 01H SET THE ZERO FLAG TO INDICATE IF AN ASCII CHARACTER IS
34 ; AVAILABLE TO BE READ FROM THE KEYBOARD BUFFER.
35 ; (ZF)= 0 -- NO CODE AVAILABLE.
36 ; (ZF)= 1 -- 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 ; (AH)= 02H RETURN THE CURRENT SHIFT STATUS IN (AL) REGISTER
40 ; THE BIT SETTINGS FOR THIS CODE ARE INDICATED IN THE
41 ; THE EQUATES FOR KB_FLAG
42 ; OUTPUT
43 ; AS NOTED ABOVE, ONLY (AX) AND FLAGS CHANGED
44 ; ALL REGISTERS RETAINED
45
46 ASSUME CS:CODE,DS:DATA
47
48 0000 KEYBOARD_IO_1 PROC FAR
49 0001 IE STI ;>>> ENTRY POINT FOR ORG 0E82EH
50 0002 83 PUSH DS ; INTERRUPTS BACK ON
51 0003 B8 0000 E CALL DDS ; SAVE CURRENT DS
52 0004 E4 PUSH BX ; SAVE BX TEMPORARILY
53 0005 83 OR AH,AH ; SET BX POINTER TO DATA REGION
54 0006 74 0B JZ K1B ; CHECK FOR (AH)= 00H
55 0007 FE CC DEC AH ; ASCII READ
56 0008 74 45 JZ K2 ; ASCII_STATUS
57 0009 FE CC DEC AH ; CHECK FOR (AH)= 01H
58 000A 66 67 JZ K3 ; SHIFT_STATUS
59 0012 8B POP BX ; RECOVER REGISTER
60 0013 1F POP DS ; INVALID COMMAND EXIT
61 0014 CF IRET
62
63 ;----- READ THE KEY TO FIGURE OUT WHAT TO DO
64 0015 BB IE 001A R KIB: MOV BX, *BUFFER_HEAD ; GET POINTER TO HEAD OF BUFFER
65 0016 9B IE 001C R CMP BX, *BUFFER_TAIL ; TEST END OF BUFFER
66 001D 75 07 JNE K1C ; IF ANYTHING IN BUFFER SKIP INTERRUPT
67
68 001F BB 9002 MOV AX, 090802H ; MOVE IN WAIT CODE & TYPE
69 0022 CD 15 INT 15H ; PENDING OTHER FUNCTION
70
71 0024 FB STI ; SCII READ
72 0025 90 NOP ; INTERRUPTS BACK ON DURING LOOP
73 0026 FA KIC: CLI ; ALLOW AN INTERRUPT TO OCCUR
74 0027 BB IE 001A R MOV BX, *BUFFER_HEAD ; INTERRUPTS BACK OFF
75 0028 9B IE 001C R CMP BX, *BUFFER_TAIL ; GET POINTER TO HEAD OF BUFFER
76 002F 83 PUSHF ; TEST END OF BUFFER
77 0030 9C PUSH BX ; SAVE ADDRESS
78 0031 E8 0587 R CALL MAKE_LED ; SAVE FLAG
79 0034 8A IE 0097 R MOV BL, *KB_FLAG_2 ; GO GET MODE INDICATOR DATA BYTE
80 0038 32 D8 XOR BL, AL ; GET PREVIOUS BITS
81 003A 80 E3 07 AND BL, KB_LEDS ; SEE IF ANY DIFFERENT
82 003D 74 04 JZ K1A ; ISOLATE INDICATOR BITS
83
84 003F E8 0549 R CALL SND_LED1 ; IF NO CHANGE BYPASS UPDATE
85 0042 FA KIA: CLI ; GO TURN ON MODE INDICATORS
86 0043 9D POPF ; DISABLE INTERRUPTS
87 0044 5B POP BX ; RESTORE FLAGS
88 0045 74 DD JZ K1 ; RESTORE ADDRESS
89
90 0047 BB 07 MOV AX, [BX] ; LOOP UNTIL SOMETHING IN BUFFER
91 0049 E8 007F R CALL K4 ; GET SCAN CODE AND ASCII CODE
92 004C 89 IE 001A R MOV *BUFFER_HEAD, BX ; MOVE POINTER TO NEXT POSITION
93
94 0050 5B POP BX ; STORE VARIABLE IN BX
95 0051 1F POP DS ; RECOVER REGISTER
96 0052 CF IRET ; RECOVER SEGMENT
97
98 ;----- ASCII STATUS
99
100 0053 FA K2: CLI ; RETURN TO CALLER
101 0054 BB IE 001A R MOV BX, *BUFFER_HEAD ; INTERRUPTS OFF
102 0055 3B IE 001C R CMP BX, *BUFFER_TAIL ; GET HEAD POINTER
103 0056 80 07 MOV AX, [BX] ; IF EQUAL (Z=1) THEN NOTHING THERE
104 0057 E8 60 PUSHF ; SAVE FLAGS
105 0058 80 POP AX ; HAVE CODE
106 0059 E8 0587 R CALL MAKE_LED ; GO GET MODE INDICATOR DATA BYTE
107 0060 8A IE 0097 R MOV BL, *KB_FLAG_2 ; GET PREVIOUS BITS
108 0063 8A IE 0097 R XOR BL, AL ; SEE IF ANY DIFFERENT
109 0067 32 D8 AND BL, KB_LEDS ; ISOLATE INDICATOR BITS
110 0069 80 E3 07 JZ SK2 ; IF NO CHANGE BYPASS UPDATE
111 006C 74 03
112
113 006E E8 0549 R CALL SND_LED1 ; GO TURN ON MODE INDICATORS
114 0071 58 POP AX ; RESTORE CODE

```

```

115 0072 9D          POPF          ; RESTORE FLAGS
116 0073 FB          STI           ; INTERRUPT BACK ON
117 0074 5B          POP  BX        ; RECOVER REGISTER
118 0075 F           POP  DS        ; RECOVER SEGMENT
119 0076 CA 0002     RET  2        ; THROW AWAY FLAGS
120
121          ;----- SHIFT STATUS
122
123 0079             K3:          ;----- SHIFT STATUS
124 0079 A0 0017 R    MOV  AL,KB_FLAG  ; GET THE SHIFT STATUS FLAGS
125 007C 5B          POP  BX        ; RECOVER REGISTER
126 007D 1F          POP  DS        ; RECOVER REGISTERS
127 007E CF          RETF          ; RETURN TO CALLER
128 007F             KEYBOARD_10_1  ENDP
129
130          ;----- INCREMENT A BUFFER POINTER
131
132 007F             K4:          ;----- INCREMENT A BUFFER POINTER
133 007F 43          PROC NEAR    ; MOVE TO NEXT WORD IN LIST
134 0080 43          INC  BX        ; MOVE TO NEXT WORD IN LIST
135 0080 3E 0082 R    INC  BX        ; MOVE TO NEXT WORD IN LIST
136 0085 15 04          CMP  BX,OBUFFER_END  ; AT END OF BUFFER?
137 0087 8B 0080 R    JNE  K5        ; NO, CONTINUE
138 0088             MOV  BX,OBUFFER_START  ; YES, RESET TO BUFFER BEGINNING
139 008B C3          RET
140 008C             K4
141 008C             ENDP
142
143          ;----- HARDWARE INT 09 H -- ( IRQ LEVEL 1 ) -----
144
145          ;----- KEYBOARD INTERRUPT ROUTINE
146
147
148 008C             KB_INT_1  PROC FAR
149 008C FB          STI           ; ENABLE INTERRUPTS
150 008D 55          PUSH BP
151 008E 50          PUSH AX
152 008F 53          PUSH BX
153 0090 51          PUSH CX
154 0091 52          PUSH DX
155 0092 56          PUSH SI
156 0093 57          PUSH DI
157 0094 1E          PUSH DS
158 0095 06          PUSH ES
159 0096 FC          CLD
160 0097 E8 0000 E    CALL DDS        ; FORWARD DIRECTION
161
162          ;----- WAIT FOR KEYBOARD DISABLE COMMAND TO BE ACCEPTED
163
164 009A B0 AD          MOV  AL,D15_KBD  ; DISABLE THE KEYBOARD COMMAND
165 009C E8 0095 R    SHIP_IT
166 009F 5A          CALL
167 00A0 2B C9          CLI
168 00A2             SUB  CX,CX        ; DISABLE INTERRUPTS
169 00A2 E4 64          KB_INT_01:  IN   AL,STATUS_PORT  ; READ ADAPTER STATUS
170 00A4 A8 02          TEST AL,INPT_BUF_FULL  ; CHECK INPUT BUFFER FULL STATUS BIT
171 00A6 0E FA          LOOPNZ KB_INT_01  ; WAIT FOR COMMAND TO BE ACCEPTED
172
173          ;----- READ CHARACTER FROM KEYBOARD INTERFACE
174
175 00A8 E4 60          IN   AL,PORT_A        ; READ IN THE CHARACTER
176
177          ;----- SYSTEM HOOK INT 15H - FUNCTION 4FH (ON HARDWARE INTERRUPT LEVEL 9H)
178
179 00AA B4 4F          MOV  AH,04FH        ; SYSTEM INTERCEPT - KEY CODE FUNCTION
180 00AC F9          STC
181 00AD CD 15          INT  15H        ; SET CY= 1 (IN CASE OF RETI)
182
183 00AF 72 03          JC   KB_INT_02  ; CASSETTE CALL (AL)= KEY SCAN CODE
184
185 00B1 E9 02EE R    JMP  K26        ; RETURNS CY= 1 FOR INVALID FUNCTION
186
187          ;----- CONTINUE IF CARRY FLAG SET (AL)=CODE
188
189
190 00B4             KB_INT_02:  ; CONTINUE IF CARRY FLAG SET (AL)=CODE
191 00B4 FB          STI           ; ENABLE INTERRUPTS AGAIN
192 00B5 3C FE          CMP  AL,KB resend  ; IS THE INPUT A RESEND
193 00B7 74 0D          JE   KB_INT_4  ; GO IF RESEND
194
195
196
197 00B9 3C FA          KB_INT_4:  ;----- CHECK FOR RESPONSE TO A COMMAND TO KEYBOARD
198 00BB T5 12          CMP  AL,KB ACK  ; IS THE INPUT AN ACKNOWLEDGE
199 00C0             JNZ  KB_INT_2  ; GO IF NOT
200
201          ;----- A COMMAND TO THE KEYBOARD WAS ISSUED
202 00BD FA          CLI
203 00BE 80 0E 0097 R 10  OR   KB_FLAG_2,KB_FA  ; DISABLE INTERRUPTS
204 00C3 E9 02EE R    JMP  K26        ; INDICATE ACK RECEIVED
205
206          ;----- RESEND THE LAST BYTE
207
208 00C6             KB_INT_4:  ;----- RESEND THE LAST BYTE
209 00C6 FA          CLI
210 00C7 80 0E 0097 R 20  OR   KB_FLAG_2,KB_FE  ; DISABLE INTERRUPTS
211 00C8 E9 02EE R    JMP  K26        ; INDICATE RESEND RECEIVED
212
213 00CF             KB_INT_2:  ;----- UPDATE MODE INDICATORS IF CHANGE IN STATE
214
215 00CF 50          PUSH AX        ; SAVE DATA IN
216 0000 E8 0587 R    CALL MAKE_LED  ; GO GET MODE INDICATOR DATA BYTE
217 0003 8A 1E 0097 R  MOV  BL,KB_FLAG_2  ; GET PREVIOUS BITS
218 0004 8A 1E 0097 R  XOR  BL,AL        ; SET MODE INDICATOR
219 0009 80 E3 07          AND  BL,KB_LEDS  ; ISOLATE INDICATOR BITS
220 00DC T4 03          JZ   UP0        ; IF NO CHANGE BYPASS UPDATE
221
222 000E E8 0536 R    UP0:          CALL SND_LED  ; GO TURN ON MODE INDICATORS
223 0001 58          POP  AX        ; RESTORE DATA IN
224 0002 8A E0          MOV  AH,AL        ; SAVE SCAN CODE IN AH ALSO
225
226
227
228          ;----- TEST FOR OVERRUN SCAN CODE FROM KEYBOARD

```

```

229
230 00E4 3C FF           CMP    AL,_KB_OVER_RUN    ; IS THIS AN OVERRUN CHAR
231 00E6 T5 0D           JNZ    K16                 ; NO, TEST FOR SHIFT KEY
232 00E8 E9 04EB R       JMP    K62                 ; BUFFER_FULL_BEEP
233
234 ;----- THIS CODE CONTAINS THE KBX SUPPORT FOR INT 09H
235
236 ;----- EQUATES
237 = 00D9 F11_M EQU 217 ; FUNC 11 MAKE
238 = 00D7 F11_M EQU 215 ; FUNC 11 BREAK
239 = 00DA F12_M EQU 218 ; FUNC 12 MAKE
240 = 00D8 F12_B EQU 216 ; FUNC 12 BREAK
241 = 0056 K102_M EQU 86 ; KEY 102 MAKE
242 = 00D6 K102_B EQU 214 ; KEY 102 BREAK
243
244 = 0052 INS_M EQU 82  ; INSERT KEY MAKE
245 = 0053 DEL_M EQU 83  ; DELETE KEY MAKE
246 = 004B LEFT_M EQU 75 ; CURSOR LEFT MAKE
247 = 004D RIGHT_M EQU 77 ; CURSOR RIGHT MAKE
248 = 0048 UP_M EQU 72   ; CURSOR UP MAKE
249 = 0050 DN_M EQU 80   ; CURSOR DOWN MAKE
250 = 0059 PGUP_M EQU 73 ; PG UP MAKE
251 = 0051 PGDN_M EQU 81 ; PG DN MAKE
252 = 0047 HOME_M EQU 71 ; HOME MAKE
253 = 004F END_M EQU 79 ; END MAKE
254
255 = 0085 FUNC11 EQU 133 ; FUNCTION 11 KEY
256 = 00E0 HC EQU 224  ; HIDDEN CODE
257
258 ;----- TABLE OF KEYPAD CURSOR & CONTROL KEYS
259
260 00EB 48 50 52 53 4B 4D K_TABI DB UP_M, DN_M, INS_M, DEL_M, LEFT_M, RIGHT_M
261 00F1 49 51 47 4F 40 L_TABI EQU PGUP_M, PGDN_M, HOME_M, END_M
262 = 000A $-K_TABI
263
264 00F5 K16:
265 00F5 24 7F AND    AL,_07FH    ; REMOVE BREAK BIT
266 00F7 0E PUSH   CS
267 00F8 07 POP    ES    ; ESTABLISH ADDRESS OF TABLES
268
269 00F9 F6 06 0096 R C0 TEST   @KB_FLAG_3, RD_ID+LC_AB ; ARE WE DOING A READ ID?
270 00FE 74 33 JZ    NOT_ID ; CONTINUE IF NOT
271 0100 79 11 JNS   TST_ID_2 ; IS THE RD_ID FLAG ON?
272 0102 80 FC AB CMP    AH,_TD_I ; IS THIS THE 1ST ID CHARACTER?
273 0105 75 05 JNE   RST_RD_ID ; INDICATE 1ST ID WAS OK
274 0106 80 0E 0096 R 40 OR    @KB_FLAG_3, LC_AB
275 010C 80 26 0096 R 40 RST_RD_ID ; INDICATE 1ST ID WAS OK
276 010C 80 26 0096 R 7F AND   @KB_FLAG_3, NOT RD_ID ; RESET THE READ ID FLAG
277 0111 EB 4B JMP    SHORT_DO_EXIT
278
279 0113 0113 TST_ID_2:
280 0113 80 26 0096 R BF AND   @KB_FLAG_3, NOT LC_AB ; RESET FLAG
281 0118 80 FC 41 CMP    AH,_TD_2 ; IS THIS THE 2ND ID CHARACTER?
282 011B 75 41 JNE   DO_EXIT ; LEAVE IF NOT
283
284 ;----- A READ ID SAID THAT IT WAS KBX
285
286 011D 80 0E 0096 R 01 OR    @KB_FLAG_3, KBX ; INDICATE KBX WAS FOUND
287 0123 80 06 0096 R 20 TEST   @KB_FLAG_3, SET_NUM_LK ; SHOULD WE SET NUM LOCK?
288 0127 74 35 JZ    DO_EXIT ; EXIT IF NOT
289 0129 80 0E 0017 R 20 OR    @KB_FLAG_NUM_STATE ; FORCE NUM LOCK ON
290 012E EB 0536 R CALL   SND_LED ; GO SET THE NUM LOCK INDICATOR
291 0131 EB 70 JMP    SHORT_EXIT
292 0133
293 0133 F6 06 0096 R 02 NOT_ID:
294 0138 74 5F TEST   @KB_FLAG_3, LC_HC ; WAS THE LAST CHARACTER A HIDDEN CODE
295 JZ    NOT_LC_HC ; JUMP IF NOT
296
297 ;----- THE LAST CHARACTER WAS A HIDDEN CODE
298 013A 80 26 0096 R FD AND   @KB_FLAG_3, NOT LC_HC ; RESET LAST CHAR HIDDEN CODE FLAG
299 013B 80 06 0096 R 20 CMP    AL,_TNS_M ; WAS IT THE INSERT KEY?
300 0141 74 05 JE    NOT_I ; IGNORE BREAK ON REST OF THESE KEYS
301 0143 F6 C4 80 TEST   AH,_80H ; IS THIS A BREAK CODE
302 0146 75 5B JNZ   EXIT ; IGNORE BREAK ON REST OF THESE KEYS
303 0148 NOT_I:
304 0148 BF 00EB R MOV    DI, OFFSET K_TABI ; TEST FOR ONE OF THE KEYPAD CURSOR FUNC
305 0149 80 000A R CMP    CX,_TAB1 ; SCAN FOR THE KEY
306 014E F2 / AE REPNE SCASB ; GO ON IF NOT FOUND
307 0150 75 54 JNE   NOT_CUR ; ARE WE IN HOLD STATE?
308 0152 F6 06 0018 R 08 TEST   @KB_FLAG_1, HOLD_STATE ; ARE WE IN HOLD STATE?
309 0157 74 07 JZ    N_HLD ; EXIT
310 0159 80 26 0018 R F7 AND   @KB_FLAG_1, NOT HOLD_STATE ; EXIT HOLD STATE
311 0160 0110
312 015E EB 43 JMP    SHORT_EXIT ; IGNORE THIS KEY
313 0160 N_HLD:
314 0160 F6 06 0017 R 08 TEST   @KB_FLAG, ALT_SHIFT ; IS ALT DOWN?
315 0165 74 0E JZ    NOT_ALT ; HOW ABOUT CTRL?
316 0167 F6 06 0017 R 04 TEST   @KB_FLAG, CTRL_SHIFT ; HOW ABOUT CTRL?
317 0171 74 0E JZ    EXIT ; IGNORE IF ONLY ALT DOWN
318 016E 3C 53 CMP    AL, DEL_M ; WAS IT THE DELETE KEY?
319 0170 75 31 JNE   EXIT ; IGNORE IF NOT
320 0172 E9 030D R JMP    K29 ; GO DO THE CTL, ALT, DEL RESET
321
322 0175 NOT_ALT:
323 0175 F6 06 0017 R 04 TEST   @KB_FLAG, CTRL_SHIFT ; IS CTL DOWN?
324 0174 75 15 JNZ   CTL_ON ; SPECIAL CASE IF SO
325 017C 3C 52 CMP    AL,_TNS_M ; IS THIS THE INSERT KEY?
326 017E 75 0E JNE   N_INS ; IGNORE IF NOT
327
328 ;----- SPECIAL HANDLING FOR INSERT KEY
329
330 0180 8A C4 MOV    AL, AH ; RECOVER SCAN CODE
331 0182 B4 80 MOV    AH, INS_SHIFT ; AH = MASK FOR INSERT
332 0184 A8 80 TEST   AL,_80H ; WAS THIS A BREAK CODE?
333 0186 75 03 JNZ   B_C ; GO HANDLE INSERT SHIFT
334 0188 E9 028F R B_C:
335 0189 74 0E JMP    K24 ; HANDLE BREAK
336 018B E9 0202 R N_INS: JMP    K49 ; HANDLE & IGNORE NUMLOCK
337 018E N_INS: JMP    K49 ; HANDLE & IGNORE NUMLOCK
338 018E 09 0453 R CTL_ON:
339 0191 CMP    CL, 5 ; WAS IT INS, DEL, UP OR DOWN?
340 0191 80 F9 05 JZ    EXIT ; IGNORE IF SO
341 0194 77 0D JMP    K42 ; GO HANDLE CTRL CASE
342 0196 E9 0401 R

```

```

343
344 0199 NOT_LC_HC:
345 0199 80 FC E0 CMP AH,HC ; LAST CHARACTER WAS NOT A HIDDEN CODE
346 019C 75 08 JNE NOT_CUR ; IS THIS CHARACTER A HIDDEN CODE?
347 019E 80 0E 0096 R 03 OR 0KB_FLAG_3,LC_HC+KBX ; SET LAST CHAR WAS A HIDDEN CODE & KBX
348 01A2 EXIT: JMP K26 ; THROW AWAY THIS CODE
349 01A3 E9 02EE R
350
351 01A6 NOT_CUR:
352 01A6 80 FC D9 CMP AH,F11_M ; WAS IT F11?
353 01A9 75 04 JNE T_F12 ; HANDLE IF SO
354 01AB B1 85 MOV CL,FUNC11 ; SET BASE FUNCTION 11
355 01B0 EB 07 JMP SHORT DO_FN
356 01AF
357 01AF 80 FC DA CMP AH,F12_M ; WAS IT F12?
358 01B2 75 43 JNE T_SYS_KEY ; GO TEST FOR SYSTEM KEY
359 01B4 B1 86 MOV CL,FUNC11+1 ; SET BASE FUNCTION 12
360
361 01B6 80 FC D7 CMP AH,F11_B ; IS THIS A BREAK CODE
362 01B7 74 E8 JE EXIT ; IGNORE BREAK CODES
363 01BB 80 FC D8 CMP AH,F12_B ; IS THIS A BREAK CODE
364 01BE B4 E3 JE EXIT ; IGNORE BREAK CODES
365 01C0 F6 06 0018 R 08 TEST 0KB_FLAG_1,HOLD_STATE ; ARE WE IN HOLD STATE?
366 01C5 74 07 JZ N_HLD1 ; N_HLD1
367 01C6 80 26 0018 R F7 AND 0KB_FLAG_1,NOT HOLD_STATE ; EXIT HOLD STATE
368 01CC EB D5 JMP SHORT EXIT ; IGNORE THIS KEY
369 01CE
370 01CE B4 E1 N_HLD1: MOV AH,CL ; IGNORE THIS KEY
371
372 01D0 F6 06 0017 R 08 TEST 0KB_FLAG,ALT_SHIFT ; ARE WE IN ALT
373 01D1 74 05 JZ T_CTL ; CNVT TO ALT FN 11-12
374 01D7 B4 C4 06 ADD AH,6
375 01DA EB 16 JMP SHORT SET_FN
376 01DC
377 01DC F6 06 0017 R 04 T_CTL: TEST 0KB_FLAG,CTL_SHIFT ; ARE WE IN CTRL
378 01E1 74 05 JZ T_SRF ; CNVT TO CTRL FN 11-12
379 01E2 80 C4 04 ADD AH,4
380 01E5 EB 0A JMP SHORT SET_FN
381 01E8
382 01E8 F6 06 0017 R 03 T_SHF: TEST 0KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT ; IS EITHER SHIFT ON?
383 01ED 74 03 JZ SET_FN ; CNVT TO SHIFT FN 11-12
384 01EF 80 C4 02 ADD AH,2
385 01F0
386 01F2 2A C0 SET_FN: SUB AL,AL ; FORCE PSEUDO SCAN CODE
387 01F4 E9 04BA R JMP K61 ; PUT IT INTO BUFFER
388
389 ;----- TEST FOR SYSTEM KEY
390
391 01F7 T_SYS_KEY:
392 01F7 3C 54 CMP AL,SYS_KEY ; IS IT THE SYSTEM KEY?
393 01F9 75 3D JNZ K16A ; CONTINUE IF NOT
394
395 01FB F6 C4 80 TEST AH,080H ; CHECK IF THIS A BREAK CODE
396 01FE 75 21 JNZ K16C ; DO NOT TOUCH SYSTEM INDICATOR IF TRUE
397
398 0200 F6 06 0018 R 04 TEST 0KB_FLAG_1,SYS_SHIFT ; SEE IF IN SYSTEM KEY HELD DOWN
399 0205 75 17 JNZ K16B ; IF YES, DON'T PROCESS SYSTEM INDICATOR
400
401 0207 80 0E 0018 R 04 OR 0KB_FLAG_1,SYS_SHIFT ; INDICATE SYSTEM KEY DEPRESSED
402 020C B0 20 MOV AL,E01 ; END OF INTERRUPT COMMAND
403 020E E6 20 OUT INTA00,AL ; SEND COMMAND TO INTERRUPT CONTROL PORT
404
405 0210 B0 AE MOV AL,ENA_KBD ; INSURE KEYBOARD IS ENABLED
406 0212 E8 0595 R CALL SHIP_IT ; EXECUTE ENABLE
407 0215 B8 8500 MOV AX,08500H ; FUNCTION VALUE FOR MAKE OF SYSTEM KEY
408 0218 FB STI ; MAKE SURE INTERRUPTS ENABLED
409 0219 CD 15 INT 15H ; USER INTERRUPT
410 021E E9 02FB R JMP K27A ; END PROCESSING
411 021E
412 021E E9 02EE R K16B: JMP K26 ; IGNORE SYSTEM KEY
413 0221
414 0221 B0 26 0018 R FB K16C: AND 0KB_FLAG_1,NOT SYS_SHIFT ; TURN OFF SHIFT KEY HELD DOWN
415 0226 B0 20 MOV AL,E01 ; END OF INTERRUPT COMMAND
416 0228 E6 20 OUT INTA00,AL ; SEND COMMAND TO INTERRUPT CONTROL PORT
417
418 022A B0 AE MOV AL,ENA_KBD ; INSURE KEYBOARD IS ENABLED
419 022C E8 0595 R CALL SHIP_IT ; EXECUTE ENABLE
420 022F B8 8501 MOV AX,08501H ; FUNCTION VALUE FOR BREAK OF SYSTEM KEY
421 0232 FB STI ; MAKE SURE INTERRUPTS ENABLED
422 0233 CD 15 INT 15H ; USER INTERRUPT
423 0235 E9 02FB R JMP K27A ; IGNORE SYSTEM KEY
424 0236
425 0238 BF 0000 E K16A: MOV D1,OFFSET K6 ; SHIFT KEY TABLE
426 023B B9 0000 E MOV CX,OFFSET K6L ; LENGTH
427 023E F21 AE REPNE SCASB ; LOOK THROUGH THE TABLE FOR A MATCH
428 0240 B4 C0 MOV AH,K1 ; RECOVER SCAS CODE
429 0242 B4 03 JE K1 ; JUMP IF MATCH FOUND
430 0244 E9 02DA R JMP K25 ; IF NO MATCH, THEN SHIFT NOT FOUND
431
432 ;----- SHIFT KEY FOUND
433 0247 K17:
434 0247 B1 EF 0001 E SUB D1,OFFSET K6+1 ; ADJUST PTR TO SCAN CODE MATCH
435 0248 2E1 8A A5 0000 E MOV AH,CS1K7[D1] ; GET MASK INTO AH
436 0250 B4 80 TEST AL,80H ; TEST FOR BREAK KEY
437 0252 74 02 JZ K17C ; BREAK_SHIFT_FOUND
438 0254 EB 5D JMP SHORT K23 ; CONTINUE
439
440 ;----- DETERMINE SET OR TOGGLE
441 0256 K17C:
442 0256 80 FC 10 CMP AH,SCROLL_SHIFT ; IF SCROLL SHIFT OR ABOVE, TOGGLE KEY
443 0259 73 07 JAE K18
444
445 ;----- PLAIN SHIFT KEY, SET SHIFT ON
446
447 0258 B6 0017 R OR 0KB_FLAG,AH ; TURN ON SHIFT BIT
448 025F E9 02EE R JMP K26 ; INTERRUPT_RETURN
449
450 ;----- TOGGLED SHIFT KEY, TEST FOR 1ST MAKE OR NOT
451
452 0262 K18:
453 0262 F6 06 0017 R 04 TEST 0KB_FLAG,CTL_SHIFT ; SHIFT_TOGGLE
454 0267 75 71 JNZ K25 ; CHECK CTL SHIFT STATE
455
456 0269 3C 52 CMP AL, INS_KEY ; CHECK FOR INSERT KEY

```

```

457 026B F6 06 0017 R 08      JNZ      K22          ; JUMP IF NOT INSERT KEY
458 026D F6 06 0017 R 08      TEST    @KB_FLAG, ALT_SHIFT ; CHECK FOR ALTERNATE SHIFT
459 0272 75 66      JNZ      K25          ; JUMP IF ALTERNATE SHIFT
460
461 0274 F6 06 0017 R 20      TEST    @KB_FLAG, NUM_STATE ; CHECK FOR BASE STATE
462 0279 75 0D      JNZ      K21          ; JUMP IF NUM LOCK IS ON
463 027B F6 06 0017 R 03      TEST    @KB_FLAG, LEFT_SHIFT+RIGHT_SHIFT
464 0280 74 0D      JZ       K22          ; JUMP IF BASE STATE
465
466 0282 80 0018 R      K20:          ; NUMERIC ZERO, NOT INSERT KEY
467 0282 B8 5230      MOV     AX, 5230H
468 0285 E9 048A R      JMP     K57
469 0288
470 0288 F6 06 0017 R 03      K21:          ; CHECK FOR BASE STATE
471 028D 74 F3      TEST    @KB_FLAG, LEFT_SHIFT+RIGHT_SHIFT
472
473 028F
474 028F 84 26 0018 R      K22:          ; SHIFT TOGGLE KEY HIT; PROCESS IT
475 0293 74 02      TEST    AH,@KB_FLAG_I
476 0295 EB 57      JZ       K22A0
477 0297
478 0297 08 26 0018 R      K22A0:        ; IS KEY ALREADY DEPRESSED
479 0298 30 26 0017 R      OR     @KB_FLAG_I, AH ; GO IF NOT
480
481 0299          ;----- TOGGLE LED IF CAPS OR NUM KEY DEPRESSED
482
483 029F F6 C4 70      TEST    AH,CAPS_SHIFT+NUM_SHIFT+SCROLL_SHIFT ; SHIFT TOGGLE?
484 02A2 74 05      JZ       K22B
485
486 02A4 50      PUSH   AX          ; SAVE SCAN CODE AND SHIFT MASK
487 02A5 E8 0536 R      CALL   SND_LED
488 02A8 58          POP    AX          ; GO TURN MODE INDICATORS ON
489 02A9
490 02A9 3C 52      CMP    AL,INS_KEY
491 02A9 55 41      JNE   K26          ; TURN OFF SHIFT BIT
492 02AD B8 5200      MOV    AX,INS_KEY*H
493 02B0 E9 048A R      JMP   K57          ; SET SCAN CODE INTO AH, 0 INTO AL
494
495 02B1          ;----- BREAK SHIFT FOUND
496
497 02B3
498 02B3 80 FC 10      CMP    AH,SCROLL_SHIFT ; IS THIS A TOGGLE KEY
499 02B6 73 1A      JAE   K24          ; YES, HANDLE BREAK TOGGLE
500 02B8 F6 D4      NOT    AH          ; INVERT MASK
501 02BA 20 26 0017 R      AND    @KB_FLAG,AH ; TURN OFF SHIFT BIT
502 02BE 3C B8      CMP    AL,ALT_KEY+80H ; IS THIS ALTERNATE SHIFT RELEASE
503 02C0 75 2C      JNE   K26          ; INVERTERRUPT_RETURN
504
505 02C1          ;----- ALTERNATE SHIFT KEY RELEASED, GET THE VALUE INTO BUFFER
506
507 02C2 A0 0019 R      MOV    AL,ALT_INPUT
508 02C5 B4 00          MOV    AH,0          ; SCAN CODE OF 0
509 02C8 80 26 0019 R      MOV    AL,INPUT_AH ; ZERO OUT THE FIELD
510 02CB E0 00          CMP    AL,0          ; END THE INPUTS
511 02CD 74 1F      JE    K26          ; INTERRUPT_RETURN
512 02CF E9 0493 R      JMP   K58          ; IT WASN'T, SO PUT IN BUFFER
513
514 02D2
515 02D4 F6 D4      NOT    AH          ; BREAK-Toggle
516 02D4 20 26 0018 R      AND    @KB_FLAG_I,AH ; INVERT MASK
517 02D8 EB 14      JMP   SHORT K26 ; INDICATE NOT LONGER DEPRESSED
518
519 02D9          ;----- TEST FOR HOLD STATE
520
521 02DA
522 02DA 3C 80      CMP    AL,80H ; NO-SHIFT-FOUND
523 02DC 73 10      JAE   K26          ; TEST FOR BREAK KEY
524 02DE F6 06 0018 R 08      TEST    @KB_FLAG_I,HOLD_STATE ; NOTHING FOR BREAK CHARS FROM HERE ON
525 02E3 74 1E      JZ       K28          ; ARE WE IN HOLD STATE
526 02E5 3C 45      CMP    AL,NUM_KEY
527 02E7 74 05      JE    K26          ; CAN'T END HOLD ON NUM_LOCK
528 02E9 80 26 0018 R F7      AND    @KB_FLAG_I,NOT HOLD_STATE ; TURN OFF THE HOLD STATE BIT
529
530 02EE
531 02EE FA      CMP    AL,80H ; INTERRUPT-RETURN
532 02EF B0 20      MOV    AL,EO1 ; TURN OFF INTERRUPTS
533 02EF E6 20      OUT   INTA00,AL ; END OF INTERRUPT COMMAND
534 02F3 00 00      K27:          SEND COMMAND TO INTERRUPT CONTROL PORT
535 02F3 B0 AE      MOV    AL,ENA_KBD ; INTERRUPT-RETURN-NO-EOI
536 02F5 E8 0595 R      CALL   SHIP_IT ; MUSICAL KEYBOARD IS ENABLED
537 02F8
538 02F8 FA      CL I          ; EXECUTE ENABLE
539 02F9 07      POP   ES          ; DISABLE INTERRUPTS
540 02F9 F7      POP   DS          ; RESTORE REGISTERS
541 02FB 5F      POP   DI
542 02FC 5E      POP   SI
543 02FD 5A      POP   DX
544 02FE 59      POP   CX
545 02FB 5B      POP   BX
546 0300 58      POP   AX
547 0301 5D      POP   BP
548 0302 CF      IRET
549
550 0303          ; RETURN, INTERRUPTS ON WITH FLAG CHANGE
551
552 0303 F6 06 0017 R 08      K28:          ; NOT IN HOLD STATE
553 0304 75 03      TEST    @KB_FLAG,ALT_SHIFT ; NO-HOLD-STATE
554 0304 E9 03A5 R      JNZ   K29          ; ARE WE IN ALTERNATE SHIFT
555 0304 E9 03A5 R      JMP   K38          ; JUMP IF ALTERNATE SHIFT
556
557 0305          ;----- TEST FOR CONTROL KEY AND RESET KEY SEQUENCE (CTL ALT DEL)
558
559 030D
560 030D F6 06 0017 R 04      K29:          ; TEST-RESET
561 0312 74 39      TEST    @KB_FLAG,CTL_SHIFT ; ARE WE IN CONTROL SHIFT ALSO
562 0314 3C 45      JZ       K31          ; NO RESET
563 0316 3C 68      CMP    AL,NUM_KEY ; CHECK FOR INVALID NUM_LOCK KEY
564 0318 3C 68      JE    K26          ; THROW AWAY IF (ALT-CTL)+NUM_LOCK
565 031A 74 D2      CMP    AL,SCROLL_KEY ; CHECK FOR INVALID SCROLL_LOCK KEY
566 031C 3C 53      JE    K26          ; THROW AWAY IF (ALT-CTL)+SCROLL_LOCK
567 031E 75 2D      CMP    AL,DEL_KEY ; CTL-ALT STATE, TEST FOR DELETE KEY
568
569 031F          ;----- CTL-ALT-DEL HAS BEEN FOUND
570

```

```

571 0320 C7 06 0072 R 1234    MOV    @RESET_FLAG,1234H    ; SET FLAG FOR RESET FUNCTION
572 0326 E9 0000 E    JMP    START_T    ; JUMP TO POWER ON DIAGNOSTICS

573
574: ;----- ALT-INPUT-TABLE
575 0329 K30: LABEL BYTE
576 0329 52 4F 50 51 4B 4C    DB    82,79,80,81,75,76
577 032F 4D 47 48 49    DB    78,77,78,79,73    ; 10 NUMBERS ON KEYPAD
578: ;----- SUPER-KEYPAD-TABLE
579 0333 10 11 12 13 14 15    DB    16,17,18,20,21,22,23
580 0339 16 17 18 19 1E 1F    DB    22,23,24,25,30,31
581 033F 20 21 22 23 24 25    DB    32,33,34,35,36,37
582 0345 26 2C 2D 2E 2F 30    DB    38,44,45,46,47,48
583 034B 31 32    DB    49,50

584
585: ;----- IN ALTERNATE SHIFT, RESET NOT FOUND
586
587 034D K31: CMP    AL,57    ; NO-RESET
588 034D 3C 39    JNE    K32    ; TEST FOR SPACE KEY
589 034D 35 35    MOV    AL,' '    ; NOT THERE
590 0351 B0 20    MOV    AL,' '    ; SET SPACE CHAR
591 0353 E9 048A R    JMP    K37    ; BUFFER_FILL

592
593: ;----- LOOK FOR KEY PAD ENTRY
594
595 0356 K32: MOV    DI,OFFSET K30    ; ALT-KEY-PAD
596 0356 BF 0329 R    MOV    CX,10    ; ALT-INPUT-TABLE
597 0359 B9 000A    REPNE  SCASB    ; LOOK FOR ENTRY USING KEYPAD
598 035C F2/ AE    JNE    K33    ; LOOK FOR MATCH
599 035E 75 13    SUB    DI,OFFSET K30+1    ; NO_ALT_KEYPAD
600 0360 81 EF 032A R    MOV    AL,ALT_INPUT    ; DI_NOW HAS ENTRY VALUE
601 0360 00 00 0019 R    MOV    AH,10    ; GET THE CURRENT BYTE
602 0367 B4 04    MUL    AH,AH    ; MULTIPLY BY 10
603 0369 F6 E4    ADD    AX,DX    ; ADD IN THE LATEST ENTRY
604 036B 03 C7    MOV    @ALT_INPUT,AL    ; STORE IT AWAY
605 036D A2 0019 R    JMP    K26    ; THROW AWAY THAT KEYSTROKE
606 0370 E9 02EE R

608: ;----- LOOK FOR SUPERSHIFT ENTRY
609
610 0373 K33: MOV    @ALT_INPUT,0    ; NO-ALT-KEYPAD
611 0373 C6 06 0019 R 00    MOV    CX,26    ; ZERO ANY PREVIOUS ENTRY INTO INPUT
612 0378 B0 001A    REPNE  SCASB    ; (DI),(ES) ALREADY POINTING
613 037C 32/ AE    JNE    K34    ; LOOK FOR MATCH IN ALPHABET
614 037D 75 05    MOV    AL,0    ; NOT FOUND, SELECTION KEY OR OTHER
615 037F B0 00    MOV    AL,0    ; ASCII CODE OF ZERO
616 0381 E9 048A R    JMP    K57    ; PUT IT IN THE BUFFER

618: ;----- LOOK FOR TOP ROW OF ALTERNATE SHIFT
619
620 0384 K34: CMP    AL,2    ; ALT-TOP-ROW
621 0384 3C 02    JB    K35    ; KEY WITH '1' ON IT
622 0386 T2 0C    CMP    AL,14    ; NOT ONE OF INTERESTING KEYS
623 0388 3C 0E    JAE    K35    ; IS IT IN THE REGION
624 038A T3 05    CMP    AL,14    ; ALT-FUNCTION
625 038C 3C 04 76    ADD    AH,118    ; CONVERT PSEUDO SCAN CODE TO RANGE
626 038F B0 00    MOV    AH,10    ; AND MAKE IT SUCH
627 0391 E9 048A R    JMP    K57    ; BUFFER_FILL

628
629: ;----- TRANSLATE ALTERNATE SHIFT PSEUDO SCAN CODES
630
631 0394 K35: CMP    AL,59    ; ALT-FUNCTION
632 0394 3C 3B    TEST   AL,59    ; TEST FOR IN TABLE
633 0396 73 03    JAE    K37    ; ALT-CONTINUE
634 0398 K36: CMP    AL,14    ; CLOSE-RETURN
635 0398 E9 02EE R    JMP    K26    ; IGNORE THE KEY
636 039B K37: CMP    AL,71    ; ALT-CONTINUE
637 039B 3C 47    JAE    K37    ; IN KEYPAD REGION
638 039D T3 F9    MOV    BX,OFFSET K13    ; IF IT IS, IGNORE IT
639 039F BB 0000 E    MOV    BX,OFFSET K13    ; ALT SHIFT PSEUDO SCAN TABLE
640 03A2 E9 04E1 R    JMP    K63    ; TRANSLATE THAT

641
642: ;----- NOT IN ALTERNATE SHIFT
643
644 03A5 K38: TEST   @KB_FLAG,CTL_SHIFT    ; NOT-ALT-SHIFT
645 03A5 F6 06 0017 R 04    JZ    K44    ; ARE WE IN CONTROL SHIFT
646 03A8 T4 62    JZ    K44    ; NOT-CTL-SHIFT

648: ;----- CONTROL SHIFT, TEST SPECIAL CHARACTERS
649: ;----- TEST FOR BREAK AND PAUSE KEYS
650
651 03AC 3C 46    CMP    AL,SCROLL_KEY    ; TEST FOR BREAK
652 03AE 75 1D    JNE    K39    ; NO-BREAK
653 03B0 8B 1E 0080 R    MOV    BX,@BUFFER_START    ; RESET BUFFER TO EMPTY
654 03B4 89 1E 001A R    MOV    @BUFFER_HEAD,BX
655 03B8 89 1E 001C R    MOV    @BUFFER_TAIL,BX
656 03BC C6 08 0011 R 80    MOV    @BIOS_BREAK,80H    ; TURN ON @BIOS_BREAK BIT
657
658: ;----- ENABLE KEYBOARD
659
660 03CD B0 AE    MOV    AL,ENA_KBD    ; ENABLE_KEYBOARD
661 03CD B5 0595 R    CALL   SHIFT_IT    ; EXECUTE_ENABLE
662 03CD C0 1B    INT    1BH    ; BREAK_INTERRUPT_VECTOR
663 03CA B2 C0    SUB    AX,AX    ; PUT OUT DUMMY CHARACTER
664 03CA E9 048A R    JMP    K57    ; BUFFER_FILL

665
666 03CD K39: CMP    AL,NUM_KEY    ; NO-BREAK
667 03CD 3C 45    JNE    K41    ; LOOK FOR PAUSE KEY
668 03CF T5 25    OR    @KB_FLAG_1,HOLD_STATE    ; NO-PAUSE
669 03D1 B0 0E 0018 R 08    JMP    K57    ; TURN ON THE HOLD FLAG

670
671: ;----- ENABLE KEYBOARD
672
673 03D6 B0 AE    MOV    AL,ENA_KBD    ; ENABLE_KEYBOARD
674 03D6 B5 0595 R    CALL   SHIFT_IT    ; EXECUTE_ENABLE
675 03D9 B0 20    MOV    AL,E01    ; END OF INTERRUPT TO CONTROL_PORT
676 03DD E6 20    OUT    INTA00,AL    ; ALLOW FURTHER KEYSTROKE INTERRUPTS
677
678: ;----- DURING PAUSE INTERVAL, TURN COLOR CRT BACK ON
679
680 03DF 80 3E 0049 R 07    CMP    @CRT_MODE,7    ; IS THIS THE MONOCHROME CARD
681 03E4 74 07    JE    K40    ; YES, NOTHING TO DO
682 03E6 B0 03DB    MOV    DX,03DBH    ; PORT FOR COLOR CARD
683 03E9 A0 0065 R    MOV    AL,@CRT_MODE_SET    ; GET THE VALUE OF THE CURRENT MODE
684 03EC EE    OUT    DX,AL    ; SET THE CRT MODE, SO THAT CRT IS ON

```

```

685
686
687
688 03ED
689 03ED F6 06 0018 R 08
690 03F2 75 F9
691
692 03F4 E9 02F8 R
693
694
695
696 03F7
697 03F7 3C 37
698 03F9 75 06
699 03FB B8 7200
700 03FE E9 048A R
701
702
703
704 0401
705 0401 BB 0000 E
706 0404 3C 3B
707 0406 72 7E
708
709 0408 BB 0000 E
710 040B E9 04E1 R
711
712
713
714 040E
715 040E 3C 47
716 0410 73 33
717 0412 F6 06 0017 R 03
718 0417 74 62
719
720
721
722 0419 3C 0F
723 0419 75 05
724 041D BB 0000
725 0420 EB 68
726
727 0422
728 0422 3C 37
729 0424 75 10
730
731
732
733 0426 B0 AE
734 0428 E9 0595 R
735 0429 75 20
736 042D E4 20
737 042F 55
738 0430 CD 05
739 0432 5D
740 0433 E9 02F3 R
741
742 0436
743 0436 3C 3B
744 0438 72 06
745 043A BB 0000 E
746 043B E9 04E1 R
747
748 0440
749 0440 BB 0000 E
750 0443 EB 41
751
752
753
754 0445
755 0445 F6 06 0017 R 20
756 044A 75 21
757 044C F6 06 0017 R 03
758 0451 75 21
759
760
761
762 0453
763
764 0453 3C 4A
765 0455 74 0C
766 0456 74 0E
767 0459 74 0D
768 045B 2C 47
769 045D BB 0000 E
770 0460 E9 04E3 R
771
772 0463 B8 442D
773 0466 EB 22
774 0468
775 0468 B8 4E2B
776 046B EB 1D
777
778
779
780 046D
781 046D F6 06 0017 R 03
782 0472 75 DF
783
784 0474
785 0474 2C 46
786 0476 BB 0000 E
787 0479 EB 0B
788
789
790
791 047B
792 047B 3C 3B
793 047D 72 04
794 047F B0 00
795 0481 EB 07
796
797 0483
798 0483 BB 0000 E
      ;----- SUSPEND SYSTEM OPERATION (LOOP) TILL NEXT KEY CLEARS HOLD STATE FLAG
      K40: TEST 0KB_FLAG_1,HOLD_STATE ; PAUSE-LOOP
            JNZ K40 ; CHECK HOLD STATE FLAG
            ; LOOP UNTIL FLAG TURNED OFF
      JMP K27A ; INTERRUPT_RETURN_NO_EOI
      ;----- TEST SPECIAL CASE KEY 55
      K41: CMP AL,55 ; NO-PAUSE
            JNE K42 ; NOT-KEY-55
            MOV AX,114H ; START/STOP PRINTING SWITCH
            JMP K57 ; BUFFER_FILL
      ;----- SET UP TO TRANSLATE CONTROL SHIFT
      K42: MOV BX,OFFSET K8 ; NOT-KEY-55
            CMP AL,59 ; SET UP TO TRANSLATE CTL
            JB K56 ; IF IT IN TABLE
            ; YES, GO TRANSLATE CHAR
            ; CTL-TABLE-TRANSLATE
            ; CTL_TABLE SCAN
            ; TRANSLATE_SCAN
      ;----- NOT IN CONTROL SHIFT
      K44: CMP AL,71 ; NOT-CTL-SHIFT
            JAE K48 ; TEST FOR KEYPAD REGION
            TEST 0KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT ; HANDLE KEYPAD REGION
            JZ K54 ; TEST FOR SHIFT STATE
      ;----- UPPER CASE, HANDLE SPECIAL CASES
      K45: CMP AL,15 ; NOT-BACK-TAB
            JNE K46 ; PRINT SCREEN KEY
            MOV AX,15H ; SET PSEUDO SCAN CODE
            JMP SHORT K57 ; BUFFER_FILL
      ;----- ISSUE INTERRUPT TO INDICATE PRINT SCREEN FUNCTION
      K46: MOV AL,ENA_KBD ; BACK TAB KEY
            CALL SHIP_IT ; EXECUTE_ENABLE
            MOV AL,E01 ; END OF CURRENT INTERRUPT
            OUT INTA00,AL ; FOR THESE THINGS CAN HAPPEN
            PUSH BP ; SAVE POINTER
            INT 05H ; ISSUE PRINT SCREEN INTERRUPT
            POP BP ; RESTORE POINTER
            JMP K27 ; GO BACK WITHOUT EOI OCCURRING
      ;----- NOT-PRINT-SCREEN
      K47: CMP AL,59 ; FUNCTION KEYS
            JB K48 ; NOT-UPPER-FUNCTION
            MOV BX,OFFSET K12 ; UPPER CASE PSEUDO SCAN CODES
            JMP K63 ; TRANSLATE_SCAN
      ;----- NOT-UPPER-FUNCTION
      K48: MOV BX,OFFSET K11 ; POINT TO UPPER CASE TABLE
            JMP SHORT K56 ; OK, TRANSLATE THE CHAR
      ;----- KEYPAD KEYS, MUST TEST NUM LOCK FOR DETERMINATION
      K49: TEST 0KB_FLAG,NUM_STATE ; KEYPAD-REGION
            JNZ K52 ; ARE WE IN NUM_LOCK
            TEST 0KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT ; TEST FOR SURE
            JNZ K53 ; ARE WE IN SHIFT STATE
            ; IF SHIFTED, REALLY NUM STATE
      ;----- BASE CASE FOR KEYPAD
      K49: ; BASE-CASE
      ;----- SPECIAL CASE FOR A COUPLE OF KEYS
      K50: CMP AL,74 ; MINUS
            JE K50
            CMP AL,78 ; PLUS
            JE K51
            SUB AL,71 ; CONVERT ORIGIN
            MOV BX,OFFSET K15 ; BASE CASE TABLE
            JMP K64 ; CONVERT TO PSEUDO SCAN
      ;----- MINUS
      K51: MOV AX,74H+-+ ; MINUS
            JMP SHORT K57 ; BUFFER_FILL
      ;----- PLUS
      K51: MOV AX,78H+-+ ; PLUS
            JMP SHORT K57 ; BUFFER_FILL
      ;----- MIGHT BE NUM LOCK, TEST SHIFT STATUS
      K52: TEST 0KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT ; ALMOST-NUM-STATE
            JNZ K49 ; SHIFTED TEMP OUT OF NUM STATE
      ;----- REALLY_NUM_STATE
      K53: SUB AL,70 ; CONVERT ORIGIN
            MOV BX,OFFSET K14 ; NUM STATE TABLE
            JMP SHORT K56 ; TRANSLATE_CHAR
      ;----- PLAIN OLD LOWER CASE
      K54: CMP AL,59 ; NOT-SHIFT
            JB K55 ; TEST FOR FUNCTION KEYS
            MOV AL,0 ; NOT-LOWER-FUNCTION
            JMP SHORT K57 ; SCAN CODE IN AH ALREADY
            ; BUFFER_FILL
      ;----- NOT-LOWER-FUNCTION
      K55: MOV BX,OFFSET K10 ; LC TABLE

```

```

799
800
801
802 0486 K56:      ;----- TRANSLATE THE CHARACTER
803 0486 FE C8 DEC AL      ; TRANSLATE-CHAR
804 0488 2E; D7 XLAT CS:K11 ; CONVERT ORIGIN
805
806
807      ;----- PUT CHARACTER INTO BUFFER
808 048A K57:      CMP AL,-1      ; BUFFER-FILL
809 048A 3C FF JE K59      ; IS THIS AN IGNORE CHAR
810 048C 74 1F CMP AH,-1      ; YES, DO NOTHING WITH IT
811 048E 80 FC FF JE K59      ; LOOK FOR -1 PSEUDO SCAN
812 0491 74 1A
813
814      ;----- HANDLE THE CAPS LOCK PROBLEM
815 0493 K58:      TEST 0KB_FLAG,CAPS_STATE ; BUFFER-FILL-NOTEST
816 0493 F6 06 0017 R 40 JZ K61      ; ARE WE IN CAPS LOCK STATE
817 0498 74 20
818
819      ;----- IN CAPS LOCK STATE
820 049A F6 06 0017 R 03 TEST 0KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT ; TEST FOR SHIFT STATE
821 049F 74 0F JZ K60      ; IF NOT SHIFT, CONVERT LOWER TO UPPER
822
823
824
825      ;----- CONVERT ANY UPPER CASE TO LOWER CASE
826 04A1 3C 41 CMP AL,'A'      ; FIND OUT IF ALPHABETIC
827 04A3 72 15 JB K61      ; NOT_CAPS_STATE
828 04A5 3C 5A CMP AL,'Z'      ; NOT_CAPS_STATE
829 04A7 77 11 JA K61      ; NOT_CAPS_STATE
830 04A9 04 20 ADD AL,'a'-'A' ; CONVERT TO LOWER CASE
831 04AB EB 0D JMP SHORT K61 ; NOT_CAPS_STATE
832
833
834 04AD K59:      JMP K26      ; NEAR-INTERRUPT-RETURN
835 04AD E9 02EE R
836
837      ;----- CONVERT ANY LOWER CASE TO UPPER CASE
838 04B0 K60:      CMP AL,'a'      ; LOWER-TO-UPPER
839 04B0 3C 61 JB K61      ; FIND OUT IF ALPHABETIC
840 04B2 72 06 CMP AL,'z'      ; NOT_CAPS_STATE
841 04B4 3C 7A JA K61      ; NOT_CAPS_STATE
842 04B6 77 02 SUB AL,'a'-'A' ; CONVERT TO UPPER CASE
843 04B8 2C 20
844
845
846 04B8 K61:      ; NOT-CAPS-STATE
847 04B8 8B 0E 001C R MOV BX,0BUFFER_TAIL ; GET THE END POINTER TO THE BUFFER
848 04B8 8B F3 00 CALL SI,BX ; SAVE THE VALUE
849 04C2 8B 0E 007F R CMP BX,0BUFFER_HEAD ; ADVANCE THE TAIL
850 04C2 8B 0E 001A R JE K62      ; HAVE THE BUFFER WRAPPED AROUND
851 04C7 74 22 CMP BX,0BUFFER_HEAD ; BUFFER FULL BEEP
852 04C9 89 04 MOV [SI],AX ; STORE THE VALUE
853 04CB 89 1E 001C R MOV 0BUFFER_TAIL,BX ; MOVE THE POINTER UP
854 04CF FA CL1
855 04D0 B0 20 MOV AL,E0I      ; TURN OFF INTERRUPTS
856 04D2 B0 20 OUT INTA00_AL ; END OF INTERRUPT COMMAND
857 04D4 B0 AE MOV AL,ENA_KBD ; SEND COMMON INTERRUPT CONTROL PORT
858 04D6 E8 0595 R CALL SH1P_IT ; INSURE KEYBOARD IS ENABLED
859 04D9 B8 9102 MOV AX,09102H ; EXECUTE ENABLE
860 04DC CD 15 INT 15H ; MOVE IN POST CODE & TYPE
861 04DE E9 02F8 R JMP K27A ; PERFORM OTHER FUNCTION
862
863
864      ;----- TRANSLATE SCAN FOR PSEUDO SCAN CODES
865
866 04E1 K63:      SUB AL,59      ; TRANSLATE-SCAN
867 04E1 2C 3B
868 04E3
869 04E3 2E; D7 K64:      XLAT CS:K9      ; CONVERT ORIGIN TO FUNCTION KEYS
870 04E5 8A E0 MOV AH,AL ; TRANSLATE-SCAN-ORIG
871 04E7 B0 00 MOV AL,0 ; CTL TABLE SCAN
872 04E9 EB 9F JMP K57      ; PUT VALUE INTO AH
873
874 04EB KB_INT_1 ENDP
875
876 04EB K62:      MOV AL,E0I      ; ZERO ASCII CODE
877 04EB B0 20 OUT INTA00_AL ; ENABLE INTERRUPT CONTROLLER CHIP
878 04E0 E6 20 MOV CX,678 ; DIVISOR FOR 1760 Hz
879 04E0 B0 02A6 MOV BL,4 ; SHORT BEEP COUNT (1/16 + 1/64 DELAY)
880 04F2 B3 00 MOV BH,AL ; GO TO COMMON BEEP HANDLER
881 04F4 E8 0000 E CALL BEEP
882 04F7 E9 02F3 R JMP K27 ; EXIT
883
884
885
886
887
888
889
890
891
892
893 04FA SND_DATA PROC NEAR
894 04FA 50 PUSH AX      ; SAVE REGISTERS
895 04FB 53 PUSH BX
896 04FC 51 PUSH CX
897 04FD 8A F8 MOV BH,AL ; SAVE TRANSMITTED BYTE FOR RETRIES
898 04F8 B3 03 MOV BL,3 ; LOAD RETRY COUNT
899 0501 SDO:      CL.I      ; DISABLE INTERRUPTS
900
901 0501 FA AND 0KB_FLAG_2,NOT (KB_FE+KB_FA) ; CLEAR ACK AND RESEND FLAGS
902
903      ;----- WAIT FOR ANY PENDING COMMAND TO BE ACCEPTED
904
905 0507 2B C9 SD1:      SUB CX,CX ; MAXIMUM WAIT COUNT
906 0509
907 0509 E4 64 IN AL,STATUS_PORT ; READ KEYBOARD PROCESSOR STATUS PORT
908 050B A8 02 TEST AL,INPT_BUF_FULL ; CHECK FOR ANY PENDING COMMAND
909 050D 00 FA LOOPNZ SD1 ; WAIT FOR COMMAND TO BE ACCEPTED
910
911 050F 8A C7 MOV AL,BH ; REESTABLISH BYTE TO TRANSMIT
912 0511 E6 60 OUT PORT_A,AL ; SEND BYTE

```

```

913 0513 FB           STI           CX,01A00H      ; ENABLE INTERRUPTS
914 0514 B9 1A00       MOV           ; LOAD COUNT FOR 10 ms+
915 0517                 SD3:        TEST          @KB_FLAG_2,KB_FE+KB_FA ; SEE IF EITHER BIT SET
916 0517 F6 06 0097 R 30  JNZ          SD7          ; IF SET, SOMETHING RECEIVED GO PROCESS
917 051C 75 0D
918
919 051E E2 F7           LOOP         SD3          ; OTHERWISE WAIT
920 0520                 SD5:        DEC           BL            ; DECREMENT RETRY COUNT
921 0520 FE CB           JNZ          SD0          ; RETRY TRANSMISSION
922 0522 75 DD
923
924 0524 80 0E 0097 R 80  OR            @KB_FLAG_2,KB_ERR ; TURN ON TRANSMIT ERROR FLAG
925 0529 EB 07           JMP           SHORT SD9  ; RETRIES EXHAUSTED FORGET TRANSMISSION
926 0528
927 0528 F6 06 0097 R 10  SD7:        TEST          @KB_FLAG_2,KB_FA ; SEE IF THIS IS AN ACKNOWLEDGE
928 0530 74 EE           JZ            SD5          ; IF NOT, GO RESEND
929 0532
930 0532 59
931 0532 6B
932 0534 58
933 0535 C3
934 0536
935
936
937
938
939
940
941
942
943 0536
944 0536 FA           SND_LED PROC NEAR
945 0537 F6 06 0097 R 40  CLI           TEST          @KB_FLAG_2,KB_PR_LED ; TURN OFF INTERRUPTS
946 053C 75 47           JNZ          SL9          ; CHECK FOR MODE INDICATOR UPDATE
947
948 053E 80 0E 0097 R 40  OR             @KB_FLAG_2,KB_PR_LED ; DON'T UPDATE AGAIN IF UPDATE UNDERWAY
949 0543 80 20
950 0545 E6 20
951 0547 EB 0D
952
953 0549
954 0549 FA           SND_LEDI:      NEAR
955 054A F6 06 0097 R 40  CLI           TEST          @KB_FLAG_2,KB_PR_LED ; TURN OFF INTERRUPTS
956 054F 75 34           JNZ          SL9          ; CHECK FOR MODE INDICATOR UPDATE
957
958 0551 80 0E 0097 R 40  OR             @KB_FLAG_2,KB_PR_LED ; DON'T UPDATE AGAIN IF UPDATE UNDERWAY
959 0556
960 0556 B0 ED           SL3:        MOV           AL,_LED_CMD ; TURN ON UPDATE IN PROCESS
961 0556 E8 04FA R       CALL          SND_DATA    ; END OF INTERRUPT COMMAND
962 0558 FA           CALL          CLI          ; SEND COMMAND TO INTERRUPT CONTROL PORT
963 055C E8 0587 R
964 055F B0 26 0097 R F8  AND           @KB_FLAG_2,NOT KB_LEDS ; GO SEND MODE INDICATOR COMMAND
965 0564 08 06 0097 R
966 0564 F6 06 0097 R 80  OR             @KB_FLAG_2,AL ; LED CMD BYTE
967 056D 75 0B           TEST          @KB_FLAG_2,KB_ERR ; SEND DATA TO KEYBOARD
968
969 056F E8 04FA R       CALL          SND_DATA    ; CLEAR MODE INDICATOR BITS
970 0572 FA           CALL          CLI          ; SAVE INDICATORS STATES FOR NEXT TIME
971 0573 F6 06 0097 R 80  TEST          @KB_FLAG_2,KB_ERR ; TRANSMIT ERROR DETECTED
972 0578 74 06           JZ            SL7          ; IF SO, BYPASS SECOND BYTE TRANSMISSION
973
974 057A B0 F4           SL5:        MOV           AL,_KB_ENABLE ; LED CMD BYTE
975 057C E8 04FA R       CALL          SND_DATA    ; SEND DATA TO KEYBOARD
976 057F FA           CALL          CLI          ; TURN OFF INTERRUPTS
977 0580
978 0580 B0 26 0097 R 3F  SL7:        AND           @KB_FLAG_2,NOT(KB_PR_LED+KB_ERR) ; TURN OFF MODE INDICATOR
979 0584 74 06           SL9:        STI           ; UPDATE AND TRANSMIT ERROR FLAG
980 0585 FB           RET           ; ENABLE INTERRUPTS
981 0586 C3
982 0587
983
984
985
986
987
988
989
990
991 0587
992 0587 51
993 0588 A0 0017 R       MAKE_LED PROC NEAR
994 0588 24 70           PUSH          CX          ; SAVE CX
995 058B B1 04           MOV           AL,_KB_FLAG ; GET CAPS & NUM LOCK INDICATORS
996 058F D2 C0           AND           AL,_CAPS_STATE+NUM_STATE+SCROLL_STATE ; ISOLATE INDICATORS
997 0591 74 07           MOV           CL,4         ; SHIFT COUNT
998 0593 59
999 0594 C3
1000 0595
1001
1002
1003
1004
1005
1006
1007
1008 0595
1009 0595 50
1010
1011
1012 059A FA           SHIP_IT PROC NEAR
1013 0597 2B C9           PUSH          AX          ; SAVE DATA TO SEND
1014 0598 74 06           CLI           SUB          CX,CX ; CLEAR TIMEOUT COUNTER
1015 0599 E4 64           S10:        IN            AL,_STATUS_PORT ; READ KEYBOARD CONTROLLER STATUS
1016 059B A5 02           TEST          AL,_INPT_BUF_FULL ; CHECK FOR ITS INPUT BUFFER BUSY
1017 059B E0 FA           LOOPNZ        S10          ; WAIT FOR COMMAND TO BE ACCEPTED
1018
1019 059F 58
1020 05A0 E6 64
1021 05A2 74 FB
1022 05A3 C3
1023 05A4
1024 05A4
1025

```

```

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_ID_
8 EXTRN DDS:NEAR
9
10 ;---- INT 17 H -----
11 ; PRINTER_ID_
12 ; THIS ROUTINE PROVIDES COMMUNICATION WITH THE PRINTER
13 ; INPUT
14 ; (AH)= 00H PRINT THE CHARACTER IN (AL)
15 ; OR RETURN (AH)= 0 IF CHARACTER NOT BE PRINTED (TIME OUT)
16 ; OTHER BITS AS IN 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 ; 7 6 5 4 3 2-1 0
21 ; | | | | | | | |
22 ; | | | | | | | | TIME OUT
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 0000 ASSUME CS:CODE,DS:DATA
49 0000 PRINTER_ID_ PROC FAR
50 0001 STT: ; ENTRY POINT FOR ORG 0EFD2H
51 0001 IE PUSH DS ; INTERRUPTS BACK ON
52 0001 56 PUSH SI ; SAVE SEGMENT
53 0003 22 PUSH DX
54 0004 51 PUSH CX
55 0005 53 PUSH BX
56 0006 E8 0000 E CALL DDS ; ADDRESS DATA SEGMENT
57 0009 8B F2 0000 E MOV SI,DX ; GET PRINTER PARAMETER
58 000F 8C 0008 R SHL SI,1 ; LOAD TIMEOUT VALUE
59 0011 84 94 0008 R MOV DX,[@PRINTER_BASE+SI] ; WORD OFFSET INTO TABLE INTO (SI)
60 0015 8B D4 0008 R OR DX,DX ; GET BASE ADDRESS FOR PRINTER CARD
61 0017 74 0C JZ B10 ; TEST DX = ZERO, INDICATING NO PRINTER
62 0019 0A E4 OR AH,AH ; EXIT, NO PRINTER ADAPTER AT OFFSET
63 001B 0E E5 DEC AH ; TEST FOR (AH)= 00H
64 001C FE CC DEC AH ; PRINT CHARACTER IN (AL)
65 001F 74 58 JZ B20 ; INITIALIZE PRINTER
66 0021 FE CC DEC AH ; TEST FOR (AH)= 02H
67 0023 74 3F JZ B50 ; GET PRINTER STATUS
68 0025 B10: ;---- PRINT THE CHARACTER IN (AL)
69 0025 5B POP BX ; RETURN
70 0027 59 POP CX
71 0027 5A POP DX
72 0028 5E POP SI ; RECOVER REGISTERS
73 0029 1F POP DS
74 002A C0 IRET ; RETURN TO CALLING PROGRAM
75
76 ;---- CHECK FOR PRINTER BUSY
77 B20: ;---- INT 15 H ----- DEVICE BUSY
78 002B 50 PUSH AX ; SAVE VALUE TO PRINT
79 002B 50 OUT DX,AL ; OUTPUT CHARACTER TO DATA PORT
80 002C EE INC DX ; POINT TO STATUS PORT
81 002D 42
82
83 ;---- CHECK FOR PRINTER BUSY
84 002E 53 PUSH BX ; SAVE TIMEOUT BASE COUNT
85 002F EC IN AL,DX ; GET STATUS PORT VALUE
86 0030 A8 80 TEST AL,80H ; IS THE PRINTER CURRENTLY BUSY
87 0032 75 05 JNZ B25 ; SKIP SYSTEM DEVICE BUSY CALL IF NOT
88
89 ;---- INT 15 H ----- DEVICE BUSY
90 0034 B8 90FE MOV AX,90FEH ; FUNCTION 90 PRINTER ID
91 0037 CD 15 INT 15H ; SYSTEM CALL
92
93 ;---- WAIT BUSY
94 B25: ;---- INT 15 H ----- DEVICE BUSY
95 0039 SUB BH,BH ; ADJUST OUTER LOOP COUNT
96 0039 2A FF RCL BX,2 ; CLEAR (BH)
97 003B C1 D3 02 B30: ; MULTIPLY BY 4
98 0040 0040 B35: ; INNER LOOP (64K)
99 0040 EC IN AL,DX ; GET STATUS
100 0041 A8 E0 MOV AH,AL ; STATUS TO (AH) ALSO
101 0041 A8 80 TEST AL,80H ; IS THE PRINTER CURRENTLY BUSY
102 0045 55 DE JNZ B40 ; GO TO NEXT STROBE
103 0047 E2 FT LOOP B30 ; LOOP IF NOT
104 0049 48 DEC BX ; DECREMENT OUTER LOOP COUNT
105 004A 75 F2 JNZ B30 ; MAKE ANOTHER PASS IF NOT ZERO
106 004C 5B POP BX ; CLEAR (BX) FROM STACK
107 004D 80 CC 01 OR AH,1 ; SET ERROR FLAG
108 0050 80 E4 F9 AND AH,0F9H ; TURN OFF THE UNUSED BITS
109 0053 EB 1C JMP SHORT B70 ; RETURN WITH ERROR FLAG SET

```

```

115 0055 5B
116 0056 B0 0D
117 0058 42
119 0059 FA
120 005A EE
121 005B EB 00
122 005D EB 00
123 005F B0 0C
124 0061 EE
125 0062 FB
126 0063 58
127
128
129
130 0064
131 0064 50
132 0065
133 0066 4B 94 0008 R
134 0069 42
135 006A EC
136 006B EC
137 006C 8A E0
138 006E B0 E4 F8
139
140 0071 5A
141 0072 B0 C2
142 0074 B0 F4 4B
143 0077 EB AC
144
145
146
147 0079
148 0079 50
149 007A 42
150 007B 42
151 007C B0 08
152 007D 8E
153 007F B0 OFA0
154 0082
155 0082 4B
156 0083 T5 FD
157 0085 B0 0C
158 0087 EE
159 0088 EB DB
160
161 008A
162
163 008A
164

B40:          POP    BX
              MOV    AL,0DH
              INC    DX
              CLD
              OUT   DX,AL
              JMP   $+2
              JMP   $+2
              MOV    AL,0CH
              OUT   DX,AL
              STI
              POP    AX
              ;----- PRINTER STATUS
              ;----- B50:          PUSH   AX
              ;----- B60:          MOV    DX,0PRINTER_BASE[SI]
              ;----- INC    DX
              ;----- IN    AL,DX
              ;----- IN    AL,DX
              ;----- MOV    AH,AL
              ;----- AND   AH,0FH
              ;----- POP    DX
              ;----- MOV    AL,DL
              ;----- XOR   AH,48H
              ;----- JMP   B10
              ;----- B70:          PUSH   AX
              ;----- B80:          PUSH   AX
              ;----- INC    DX
              ;----- INC    DX
              ;----- MOV    AL,8
              ;----- OUT   DX,AL
              ;----- MOV    AX,1000*4
              ;----- B90:          DEC    AX
              ;----- JNZ   B90
              ;----- MOV    AL,0CH
              ;----- OUT   DX,AL
              ;----- JMP   B60
              ;----- PRINTER_I_O_I  ENDP
              ;----- CODE  ENDS
              ;----- END

```

:
 SEND STROBE PULSE  
 : RESTORE (BX) WITH TIMEOUT COUNT  
 : SET THE STROBE LOW (BIT ON)  
 : OUTPUT STROBE TO CONTROL PORT  
 : PREVENT INTERRUPT PULSE STRETCHING  
 : OUTPUT STROBE BIT 1 > 1us < 5us  
 : 1/4 DELAY AND ALLOW FOR LINE LOADING  
 : AND FOR CORRECT PULSE WIDTH  
 : SET THE -STROBE HIGH  
 : INTERRUPTS BACK ON  
 : RECOVER THE OUTPUT CHAR

:
 : SAVE (AL) REGISTER  
 : GET PRINTER ATTACHMENT BASE ADDRESS  
 : POINT TO PRINTER PORT  
 : PRE-CHARGE +BUSY LINE IF FLOATING  
 : GET PRINTER STATUS HARDWARE BITS  
 : SAVE  
 : TURN OFF UNUSED BITS

:
 : RECOVER (AL) REGISTER  
 : MOVE CHARACTER INTO (AL)  
 : FLIP A COUPLE OF BITS  
 : RETURN FROM ROUTINE WITH STATUS IN AH

:
 : SAVE (AL)  
 : POINT TO OUTPUT PORT

:
 : SET INIT LINE LOW

:
 : ADJUST FOR INITIALIZATION DELAY LOOP  
 : INIT\_LOOP  
 : LOOP FOR RESET TO TAKE  
 : INIT\_LOOP  
 : NO INTERRUPTS, NON AUTO LF, INIT HIGH

:
 : EXIT THROUGH STATUS ROUTINE

```

1 PAGE 118,121
2 TITLE RS232 ----- 06/10/85 COMMUNICATIONS BIOS (RS232)
3 .LIST
4 0000 CODE SEGMENT BYTE PUBLIC
5
6 PUBLIC RS232_10_I
7 EXTRN AI1NEAR
8 EXTRN DDS1NEAR
9
10 ---- INT 14 H -----
11 ;RS232_10
12 ; THIS ROUTINE PROVIDES BYTE STREAM I/O TO THE COMMUNICATIONS
13 ; PORT ACCORDING TO THE PARAMETERS:
14 ;
15 ; (AH)= 00H INITIALIZE THE COMMUNICATIONS PORT
16 ; (AL) HAS PARAMETERS FOR INITIALIZATION
17 ;
18 ;----- BAUD RATE -- -PARITY-- STOPBIT 1 WORD LENGTH--:
19 ;----- 7 6 5 4 3 2 1 0
20 ;----- 000 - 110 X0 - NONE 0 - 1 10 - 7 BITS
21 ;----- 001 - 150 01 - ODD 1 - 2 11 - 8 BITS
22 ;----- 010 - 300 11 - EVEN
23 ;----- 011 - 600
24 ;----- 100 - 1200
25 ;----- 110 - 2400
26 ;----- 111 - 4800
27 ;----- 111 - 9600
28 ;
29 ; ON RETURN, CONDITIONS SET AS IN CALL TO COMM STATUS (AH=03H)
30 ;
31 ; (AH)= 01H SEND THE CHARACTER IN (AL) OVER THE COMM LINE
32 ; (AL) REGISTER IS PRESERVED
33 ; ON EXIT, BIT 7 OF AH IS SET IF THE ROUTINE WAS UNABLE TO
34 ; TO TRANSMIT THE BYTE OF DATA OVER THE LINE.
35 ; IF BIT 7 OF AH IS NOT SET, THE
36 ; REMAINDER OF (AH) IS SET AS IN A STATUS REQUEST,
37 ; REFLECTING THE CURRENT STATUS OF THE LINE.
38 ;
39 ; (AH)= 02H RECEIVE CHARACTER INTO (AL) FROM COMM LINE BEFORE
40 ; RETURNING TO CALLER.
41 ; ON EXIT, (AH) HAS THE CURRENT LINE STATUS, AS SET BY THE
42 ; THE STATUS ROUTINE, EXCEPT THAT THE ONLY BITS
43 ; LEFT ON ARE THE ERROR BITS (7,4,3,2,1)
44 ; IF (AH) HAS BIT 7 ON (TIME OUT) THE REMAINING
45 ; BITS ARE NOT PRESENT.
46 ; THUS, (AH) IS NON ZERO ONLY WHEN AN ERROR OCCURRED.
47 ;
48 ; (AH)= 03H RETURN THE COMM PORT STATUS IN (AX)
49 ; (AH) CONTAINS THE LINE CONTROL STATUS
50 ; BIT 7 = TIME OUT
51 ; BIT 6 = TRANSMIT SHIFT REGISTER EMPTY
52 ; BIT 5 = TRANSMIT HOLDING REGISTER EMPTY
53 ; BIT 4 = BREAK DETECT
54 ; BIT 3 = FRAMING ERROR
55 ; BIT 2 = PARITY ERROR
56 ; BIT 1 = OVERRUN ERROR
57 ; BIT 0 = DATA READY
58 ;
59 ; (AL) CONTAINS THE MODE STATUS
60 ; BIT 7 = RECEIVE LINE SIGNAL DETECT
61 ; BIT 6 = RING INDICATOR
62 ; BIT 5 = DATA SET READY
63 ; BIT 4 = CLEAR TO SEND
64 ; BIT 3 = DELTA RECEIVE LINE SIGNAL DETECT
65 ; BIT 2 = TRAILING EDGE RING DETECTOR
66 ; BIT 1 = DELTA DATA SET READY
67 ; BIT 0 = DELTA CLEAR TO SEND
68 ;
69 ; (DX) = PARAMETER INDICATING WHICH RS232 CARD (0,1 ALLOWED)
70 ; DATA AREA RS232_BASE CONTAINS THE BASE ADDRESS OF THE 8250 ON THE CARD
71 ; LOCATION 100H CONTAINS UP TO 8 RS232 ADDRESSES POSSIBLE
72 ; DATA AREA LABEL @RS232_TIM_OUT (BYTE) CONTAINS OUTER LOOP COUNT
73 ; VALUE FOR TIMEOUT (DEFAULT=1)
74 ;
75 ; OUTPUT AX MODIFIED ACCORDING TO PARAMETERS OF CALL
76 ; ALL OTHERS UNCHANGED
77 ;
78 0000 ASSUME CS:CODE,DS:DATA
79
80 RS232_10_I PROC FAR
81 ;
82 ;----- VECTOR TO APPROPRIATE ROUTINE
83 0000 FB STI ; INTERRUPTS BACK ON
84 0001 1E PUSH DS ; SAVE SEGMENT
85 0002 52 PUSH DX
86 0003 56 PUSH SI
87 0005 51 PUSH DI
88 0006 53 PUSH BX
89 0007 BB F2 MOV SI,DX ; RS232 VALUE TO (SI)
90 0009 BB FA MOV DI,DX ; AND TO (DI) (FOR TIMEOUTS)
91 000B D1 E6 SHL SI,1 ; WORD OFFSET
92 000B E8 0000 E CALL DDS ; GET BASE ADDRESS
93 0010 00 0000 R MOV DS,RS232_BASE[SI] ; TEST FOR 0 BASE ADDRESS
94 0014 0B D2 OR DX,DX ; TEST FOR 0 BASE ADDRESS
95 0016 74 13 JZ A3 ; RETURN
96 001A 0A E4 OR AH,AH ; TEST FOR (AH)=00H
97 001A T4 16 JZ A4 ; COMM. INITIALIZATION
98 001C FE CC DEC AH ; TEST FOR (AH)=01H
99 001C 00 0B JZ A5 ; SEND
100 0020 FE CC DEC AH ; TEST FOR (AH)=02H
101 0022 T4 70 JZ A12 ; RECEIVE INTO (AL)
102 0024 A2: DEC AH ; TEST FOR (AH)=03H
103 0024 FE CC JNZ A3 ; COMMUNICATION STATUS
104 0028 00 03 R JMP A18 ; RETURN FROM RS232
105 0028 E9 00B6 R A3: POP BX ; RETURN TO CALLER, NO ACTION
106 0028 00 0F R POP CX
107 0028 5B POP DI
108 002C 59 POP SI
109 002D 5F POP DX
110 002E 5E POP DS
111 002F 5A POP DS
112 0030 1F POP DS
113 0031 CF IRET

```

```

114          PAGE
115          ;----- INITIALIZE THE COMMUNICATIONS PORT
116
117          0032
118          0032 8A E0          A4:
119          0034 83 C2 03          MOV  AH,AL          ; SAVE INITIALIZATION PARAMETERS IN (AH)
120          0037 80 80          ADD  DX,3          ; POINT TO 8250 CONTROL REGISTER
121          0039 EE          MOV  AL,80H          ; SET DLAB=1
122
123          ;----- DETERMINE BAUD RATE DIVISOR
124
125          003A 8A D4          MOV  DL,AH          ; GET PARAMETERS TO (DL)
126          003C B1 04          MOV  CL,4          ; GET ms OF DIVISOR
127          003E D2 C2          ROL  DL,CL          ; ISOLATE THEM
128          0040 81 E2 000E          AND  DX,0EH          ; BASE OF TABLE
129          0044 BF 0000          MOV  DI,OFFSET AI          ; PUT INTO INDEX REGISTER
130          0048 03 FA          ADD  DI,DX          ; POINT TO HIGH ORDER OF DIVISOR
131          0049 80 94 0000 R          MOV  DX,@RS232_BASE[SI]
132          004D 42
133          004E 2E; 8A 45 01          INC  DX          ; GET HIGH ORDER OF DIVISOR
134          0052 EE          MOV  AL,CS:[DI]+1          ; SET ms OF DIVISOR TO 0
135          0053 4A          DEC  DX
136          0056 EB 00          JMP  $+2          ; I/O DELAY
137          0057 05; 8A 05          MOV  AL,CS:[DI]
138          0059 EE          OUT  DX,AL          ; GET LOW ORDER OF DIVISOR
139          005A 83 C2 03          ADD  DX,3          ; SET LOW OF DIVISOR
140          0058 8A C4          MOV  AL,AH          ; GET PARAMETERS BACK
141          005F 24 1F          AND  AL,01FH          ; STRIP OFF THE BAUD BITS
142          0060 44
143          0062 44          DEC  DX          ; LINE CONTROL TO 8 BITS
144          0063 44          DEC  DX
145          0064 EB 00          JMP  $+2          ; I/O DELAY
146          0066 B0 00          MOV  AL,0          ; INTERRUPT ENABLES ALL OFF
147          0066 EE          OUT  DX,AL          ; COM_STATUS
148          0069 EB 4B          JMP  SHORT A1B          ; SEND CHARACTER IN (AL) OVER COMM LINE
149
150
151          ;----- SEND CHARACTER IN (AL) OVER COMM LINE
152          006B
153          006B 50          A5:
154          006C 83 C2 04          PUSH AX          ; SAVE CHAR TO SEND
155          006F 00 03          ADD  DX,4          ; MODEM CONTROL REGISTER
156          0071 EE          MOV  AL,3          ; DTR AND RTS
157          0072 42          OUT  DX,AL          ; DATA TERMINAL READY, REQUEST TO SEND
158          0073 42          INC  DX          ; MODEM STATUS REGISTER
159          0074 B7 30          MOV  BH,30H          ; DATA SET READY & CLEAR TO SEND
160          0076 E8 00C5 R          CALL  WAIT_FOR_STATUS          ; ARE BOTH TRUE
161          0078 14 08          JE   A9          ; YES, READY TO TRANSMIT CHAR
162          007B
163          007B 59          A7:
164          007C 8A C1          POP  CX          ; RELOAD DATA BYTE
165          007E
166          007E 80 CC 80          MOV  AL,CL          ; INDICATE TIME OUT
167          0081 EB A8          OR   AH,80H          ; RETURN
168
169          0083
170          0083 4A          A8:
171          0084          A10:
172          0084 B7 20          MOV  BH,20H          ; CLEAR_TO_SEND
173          0085 E8 00C5 R          CALL  WAIT_FOR_STATUS          ; LINE STATUS REGISTER
174          0089 15 F0          JNZ  A7          ; WAIT SEND
175          008B
176          008B 83 EA 05          A11:
177          008E 59          SUB  DX,5          ; IS TRANSMITTER READY
178          008F 8A C1          POP  CX          ; TEST FOR TRANSMITTER READY
179          0091 EE          MOV  AL,CL          ; RETURN WITH TIME OUT SET
180          0092 EB 97          OUT  DX,AL          ; OUT CHAR
181
182          ;----- RECEIVE CHARACTER FROM COMM LINE
183
184          0094
185          0094 83 C2 04          A12:
186          0097 B0 01          ADD  DX,4          ; MODEM CONTROL REGISTER
187          0099 EE          MOV  AL,1          ; DATA TERMINAL READY
188          009A 42          OUT  DX,AL          ; MODEM STATUS REGISTER
189          0098 42          INC  DX
190          009C B7 20          A13:
191          009E E8 00C5 R          MOV  BH,20H          ; WAIT_DSR
192          00A1 75 DB          CALL  WAIT_FOR_STATUS          ; DATA_SET READY
193          00A1 JZ   A8          IN   AH,00000001          ; TEST FOR DSR
194
195          00A3          A15:
196          00A4 4A          DEC  DX          ; RETURN WITH ERROR
197          0101 B7 01          A16:
198          00A6 E8 00C5 R          MOV  BH,1          ; WAIT_DSR_END
199          00A8 75 D3          CALL  WAIT_FOR_STATUS          ; LINE_STATUS_REGISTER
200          00AB
201          00AB B0 E4 1E          JNZ  A8          ; WAIT_RECV
202
203          00AE B8 94 0000 R          AND  AH,00000001          ; RECEIVE BUFFER FULL
204          00B2 EC 0000          MOV  DX,@RS232_BASE[SI]          ; TEST FOR RECEIVE BUFFER FULL
205          00B3 E9 002B R          IN   AL,DX          ; RETURN WITH RECEIVE ERROR
206
207          ;----- COMM PORT STATUS ROUTINE
208
209          00B6
210          00B6 B8 94 0000 R          A18:
211          00B6 83 C2 05          MOV  DX,@RS232_BASE[SI]          ; CONTROL_PORT
212          00B6 EC          ADD  DX,5          ; GET LINE CONTROL STATUS
213          00B6 8A E0          IN   AL,DX          ; PUT IN (AH) FOR RETURN
214          00C0 42          MOV  AH,AL          ; POINT TO MODEM STATUS REGISTER
215          00C1 EC          INC  DX          ; GET MODEM CONTROL STATUS
216          00C2 E9 002B R          IN   AL,DX          ; RETURN

```

```
217  
218  
219  
220 ;-----  
221 ; WAIT FOR STATUS ROUTINE  
222 ;ENTRY: (BH)= STATUS BIT(S) TO LOOK FOR  
223 ; (DH)= ADDRESS OF STATUS REG  
224 ;EXIT: ZERO FLAG ON = STATUS FOUND  
225 ; ZERO FLAG OFF = TIMEOUT.  
226 ; (AH)= LAST STATUS READ  
227  
228 00C5 00C5 8A 9D 007C R  
229  
230 ;-----  
231 ; ADJUST OUTER LOOP COUNT  
232 00C9 55  
233 00C9 53  
234 00B8 D0  
235 00C0 B1 E5 00FF  
236 00D0 D1 D5  
237 00D2 D1 D5  
238 00D4  
239 00D4 28 C9  
240 00D6  
241 00D6 EC  
242 00D7 5A E0  
243 00D9 22 C7  
244 00DB 3A C7  
245 00D7 74 07  
246 00DF E2 F5  
247 00E1 4D  
248 00E2 75 F0  
251  
252 00E4 0A FF  
253 00E6  
254 00E6 5D  
255 00E7 C3  
256  
257 00E8  
258  
259 00E8  
260  
261 00E8  
262  
PAGE  
;-----  
; WAIT FOR STATUS ROUTINE  
;ENTRY: (BH)= STATUS BIT(S) TO LOOK FOR  
; (DH)= ADDRESS OF STATUS REG  
;EXIT: ZERO FLAG ON = STATUS FOUND  
; ZERO FLAG OFF = TIMEOUT.  
; (AH)= LAST STATUS READ  
;-----  
; ADJUST OUTER LOOP COUNT  
;-----  
; SAVE (BP)  
; SAVE (BX)  
; USE BP FOR OUTER LOOP COUNT  
; SHIFT HIGH BITS  
; MULTIPLY OUTER COUNT BY 4  
WF50:  
SUB CX, CX  
WFS1:  
IN AL,DX  
MOV AH,AL  
AND AL,BH  
CMP AL,BH  
JE WFS_END  
LOOP WFS1  
; TRY AGAIN  
DEC BP  
JNZ WF50  
WFS_END:  
OR BH,BH  
POP BP  
RET  
; SET ZERO FLAG OFF  
; RESTORE (BP)  
WAIT_FOR_STATUS ENDP  
RS232_IO_1 ENDP  
CODE ENDS
```

```

PAGE 116,121
TITLE VIDEO1 --- 06/10/85 VIDEO DISPLAY BIOS
.286C
.LIST
0000      CODE      SEGMENT BYTE PUBLIC
.
PUBLIC ACT_DISP_PAGE
PUBLIC READ_AC_CURRENT
PUBLIC READ_CURSOR
PUBLIC READ_DOT
PUBLIC READ_LPEN
PUBLIC SCROLL_DOWN
PUBLIC SCROLL_UP
PUBLIC SET_COLOR
PUBLIC SET_CPOS
PUBLIC SET_CTYPE
PUBLIC SET_MODE
PUBLIC WRITE_AC_CURRENT
PUBLIC WRITE_C_CURRENT
PUBLIC WRITE_DOT
PUBLIC WRITE_LPEN
PUBLIC VIDEO_10_I
PUBLIC VIDEO_STATE

EXTRN BEEP:NEAR           ; SPEAKER BEEP ROUTINE
EXTRN GC_CHT:GEN:NEAR     ; CHARACTER GENERATOR GRAPHICS TABLE
EXTRN DOSSTAR:NEAR        ; CAD (DS) WITH DATA SEGMENT SELECTOR
EXTRN M5:WORD              ; REGEN BUFFER LENGTH TABLE
EXTRN M6:BYTE              ; COLUMNS PER MODE TABLE
EXTRN M7:BYTE              ; MODE SET VALUE PER MODE TABLE

;--- INT 10 H ---
VIDEO1:INT10H
;THESE ROUTINES PROVIDE THE CRT DISPLAY INTERFACE
;THE FOLLOWING FUNCTIONS ARE PROVIDED:
;
;(AH)= 00H SET MODE (AL) CONTAINS MODE VALUE
;(AL) = 00H 40X25 BW MODE (POWER ON DEFAULT)
;(AL) = 01H 40X25 COLOR
;(AL) = 02H 80X25 BW
;(AL) = 03H 80X25 COLOR
;GRAPHICS MODES
;(AL) = 04H 320X200 COLOR
;(AL) = 05H 320X200 BW MODE
;(AL) = 06H 640X200 BW MODE
;(AL) = 07H 80X25 MONOCHROME (USED INTERNAL TO VIDEO ONLY)
;*** NOTES -BW MODES OPERATE SAME AS COLOR MODES, BUT COLOR
;BURST IS NOT ENABLED
;CURSOR IS NOT DISPLAYED IN GRAPHICS MODE
;
;(AH)= 01H SET CURSOR TYPE
;(CH) = BITS 4-0 = START LINE FOR CURSOR
;** HARDWARE WILL ALWAYS CAUSE BLINK
;** SETTING BIT 5 OR 6 WILL CAUSE ERRATIC BLINKING
;ON NO CURSOR AT ALL
;(CL) = BITS 4-0 = END LINE FOR CURSOR
;
;(AH)= 02H SET CURSOR POSITION
;(DH,DL) = ROW,COLUMN (00H,00H) IS UPPER LEFT
;(BH) = PAGE NUMBER (MUST BE 00H FOR GRAPHICS MODES)
;ON EXIT (DH,DL) = ROW,COLUMN OF CURRENT CURSOR
;(CH,CL) = CURSOR MODE CURRENTLY SET
;
;(AH)= 04H READ LIGHT PEN POSITION
;ON EXIT:
;(AH) = 00H -- LIGHT PEN SWITCH NOT DOWN/NOT_TRIGGERED
;(AH) = 01H -- VALID LIGHT PEN VALUE IN REGISTERS
;(DH,DL) = ROW,COLUMN OF CHARACTER LP POSITION
;(CH) = RASTER LINE (0-199)
;(BH) = PAGE NUMBER (0-31,0-639)
;
;(AH)= 05H SELECT ACTIVE DISPLAY PAGE (VALID ONLY FOR ALPHA MODES)
;(AL) = NEW PAGE VALUE (10-7 FOR MODES 041, 0-3 FOR MODES 2&3)
;
;(AH)= 06H SCROLL ACTIVE PAGE UP
;(AL) = NUMBER OF LINES, INPUT LINES BLANKED AT BOTTOM OF WINDOW
;(AL) = 00H MEANS BLANK ENTIRE WINDOW
;(CH,CL) = ROW,COLUMN OF UPPER LEFT CORNER OF SCROLL
;(DH,DL) = ROW,COLUMN OF LOWER RIGHT CORNER OF SCROLL
;(BH) = ATTRIBUTE TO BE USED ON BLANK LINE
;
;(AH)= 07H SCROLL ACTIVE PAGE DOWN
;(AL) = NUMBER OF LINES, INPUT LINES BLANKED AT TOP OF WINDOW
;(AL) = 00H MEANS BLANK ENTIRE WINDOW
;(CH,CL) = ROW,COLUMN OF UPPER LEFT CORNER OF SCROLL
;(DH,DL) = ROW,COLUMN OF LOWER RIGHT CORNER OF SCROLL
;(BH) = ATTRIBUTE TO BE USED ON BLANK LINE
;
CHARACTER HANDLING ROUTINES
;
;(AH)= 08H READ ATTRIBUTE/CHARACTER AT CURRENT CURSOR POSITION
;(AL) = DISPLAY PAGE (VALID FOR ALPHA MODES ONLY)
;ON EXIT:
;(AL) = CHAR READ
;(AH) = ATTRIBUTE OF CHARACTER READ (ALPHA MODES ONLY)
;
;(AH)= 09H WRITE ATTRIBUTE/CHARACTER AT CURRENT CURSOR POSITION
;(BH) = DISPLAY PAGE (VALID FOR ALPHA MODES ONLY)
;(CX) = COUNT OF CHARACTERS TO WRITE
;(AL) = CHAR TO WRITE
;(BL) = ATTRIBUTE OF CHARACTER (ALPHA)/COLOR OF CHAR (GRAPHICS):
;      SEE NOTE ON WRITE DOT FOR BIT 7 OF BL = 1.
;
;(AH)= 0AH WRITE CHARACTER ONLY AT CURRENT CURSOR POSITION
;(BH) = DISPLAY PAGE (VALID FOR ALPHA MODES ONLY)
;(CX) = COUNT OF CHARACTERS TO WRITE
;(AL) = CHAR TO WRITE
;      NOTE: USE FUNCTION (AH)= 09H IN GRAPHICS MODE
;FOR READ/WRITE CHARACTER INTERFACE WHILE IN GRAPHICS MODE, THE
;CHARACTERS ARE FORMED FROM CHARACTER GENERATOR IMAGE
;MAINTAINED IN THE SYSTEM ROM. ONLY THE FIRST 128 CHARS
;ARE CONTAINED THERE. TO READ/WRITE THE SECOND 128 CHARS,
;THE USER MUST INITIALIZE THE POINTER AT INTERRUPT 1FH
;(LOCATION 0007CH) TO POINT TO THE 1K BYTE TABLE CONTAINING
;THE CODE POINTS FOR THE SECOND 128 CHARS (128-255).
;FOR WRITE CHARACTER INTERFACE IN GRAPHICS MODE, THE REPLICATION FACTOR
;CONTROLS (CX) COUNT ENTRY. THIS PRODUCES VALID RESULTS ONLY
;FOR CHARACTERS CONTAINED ON THE SAME ROW. CONTINUATION TO
;SUCCEEDING LINES WILL NOT PRODUCE CORRECTLY.
;
```

```

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 ;-----+
143 ;-----+
144 ;-----+
145 ;-----+
146 ;-----+
147 ;-----+
148 ;-----+
149 ;-----+
150 ;-----+
151 ;-----+
152 ;-----+
153 ;-----+
154 ;-----+
155 ;-----+
156 ;-----+
157 ;-----+
158 ;-----+
159 ;-----+
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 0000 0067 R
187 0002 0137 R
188 0004 015C R
189 0006 0184 R
190 0008 0771 R
191 000A 098 R
192 000C 028F R
193 000E 0247 R
194 0010 02F9 R
195 0012 0353 R
196 0014 0385 R
197 0016 0400 R
198 0018 0446 R
199 001A 0435 R
200 001C 06EA R
201 001E 01E5 R
202 0020 012E R
203 0022 012E R
204 0024 012E R
205 0026 03B2 R
206 = 0028
207
208 0028
209 0028 FB
210 002A 00
211 002A 06
212 002B 1E
213 002C 52
214 002D 51
215 002E 53
216 002F 56
217 0030 57
218 0031 55
219 0032 E8 0000 E
220 0035 BE B800
221 0038 BE 3E 0010 R
222 003A 81 FF 0030
223 0040 83 FF 30
224 0043 75 03
225 0045 BE B000
226 0046
227 0048 80 FC 13
228 0048 74 02
M1 DW OFFSET SET_MODE ; TABLE OF ROUTINES WITHIN VIDEO I/O
M1 DW OFFSET SET_CTYPE
M1 DW OFFSET SET_CPOS
M1 DW OFFSET READ_CURSOR
M1 DW OFFSET READ_LPEN
M1 DW OFFSET ACT_DISP_PAGE
M1 DW OFFSET SCROLL_PAGE
M1 DW OFFSET SET_SCROLL_DOWN
M1 DW OFFSET READ_AC_CURRENT
M1 DW OFFSET WRITE_AC_CURRENT
M1 DW OFFSET WRITE_C_CURRENT
M1 DW OFFSET SET_COLOR
M1 DW OFFSET READ_TELT
M1 DW OFFSET WRITE_TTY
M1 DW OFFSET VIDEO_STATE
M1 DW OFFSET VIDEO_RETURN ; RESERVED
M1 DW OFFSET VIDEO_RETURN ; RESERVED
M1 DW OFFSET VIDEO_RETURN ; RESERVED
M1 DW OFFSET VIDEO_RETURN ; CASE 19H, WRITE STRING
MIL EQU $-M1
VIDEO_IO_I PROC NEAR ; ENTRY POINT FOR ORG 0F065H
ST1 CDP
CDP
PUSH ES
PUSH DS ; SET DIRECTION FORWARD
PUSH DX
PUSH CX ; SAVE WORK AND PARAMETER REGISTERS
PUSH BX
PUSH SI
PUSH DI
PUSH BP
CALL DDS ; POINT DS: TO DATA SEGMENT
MOV SI,0B800H ; GET SEGMENT FOR COLOR CARD
MOV DI,*EQUIP_FLAG ; GET EQUIPMENT FLAGS SETTING
AND DI,30H ; IS COLOR CARD SW CARD?
CMP DI,30H ; IS SETTING FOR BW CARD?
JNE M2 ; SKIP IF NOT BW CARD
MOV SI,0B000H ; ELSE GET SEGMENT FOR BW CARD
M2: CMP AH,13H ; TEST FOR WRITE STRING OPERATION
JE M3 ; SKIP IF ES:BP VALID AS PASSED

```

```

229 004D 8E C6           MOV    ES,SI      ; SET UP TO POINT AT VIDEO MEMORY AREAS
230 004F                 M3:   MOV    SI,AX      ; MOVE COMMAND TO LOOK UP REGISTER
231 004F 8B F0           SHR    SI,8       ; SHIFT COMMAND TO FORM BYTE OFFSET
232 0051 C1 EE 08         SAL    SI,1       ; TIMES 2 FOR WORD TABLE LOOKUP
233 0054 D1 E6           CMP    SI,MIL    ; TEST FOR WITHIN TABLE RANGE
234 0056 83 FE 28         JNB    M4        ; BRANCH TO EXIT IF NOT A VALID COMMAND
235 0059 73 09
236
237 005B 8A 26 0049 R    MOV    AH,0CRT_MODE ; MOVE CURRENT MODE INTO AH
238 005F 2E; FF A4 0000 R JMP    WORD PTR CS:[SI+OFFSET M1] ; GO TO SELECTED FUNCTION
239
240 0064
241 0064 E9 012E R
242 0067
243
244
245
246
247
248
249
250
251
252 0067
253 0067 8A 03D4
254 0068 76 89 16 0632 R
255 006D 75 04
256 006F B0 07
257 0071 B2 B4
258
259 0073 A2 0049 R
260 0076 89 16 0632 R
261 0079 00 06 0084 R 18
262 007F 1E
263 0080 50
264 0081 98
265 0082 8B F0
266 0083 2E; 8A 84 0000 E
267 0089 00 06 0045 R
268 008C 24 37
269 008E 52
270 008F 83 C2 04
271 0092 EE
272 0093 5A
273
274 0094 2B DB
275 0096 8E DB
276 0098 C5 1E 0074 R
277
278 009C 56
279 009D B9 00 010
280 00A0 3C 02
281 00A2 72 0E
282 00A4 03 D9
283 00A6 3C 04
284 00A8 72 0B
285 00A9 D9
286 00AC 3C 07
287 00AE 72 02
288 00B0 03 D9
289
290
291
292 00B2
293 00B2 50
294 00B3 8B 47 0A
295 00B6 86 E0
296 00B8 1E
297
298 00B9 E8 0000 E
299 00BC A3 0060 R
300
301 00BF 1F
302 00C0 3C 04
303
304
305
306 00C2
307 00C2 8A C4
308 00C4 EE
309 00C5 02
310 00C6 FE C4
311 00C8 8A 07
312 00CA EE
313 00CB 43
314 00CC 4A
315 00CD 22 F3
316 00CF 5B
317 00D0 1F
318
319
320
321
322 00D1 33 FF
323 00D3 89 3E 004E R
324 00D7 C6 06 0062 R 00
325 00DC B9 2000
326 00DF 3C 04
327 00E0 74 0A
328 00E3 3C 07
329 00E5 74 04
330 00E7 33 C0
331 00E9 EB 05
332 00EB
333 00EB B5 08
334 00ED
335 00ED BB 0720
336 00F0
337 00F0 F3/ AB
338
339
340
341 00F2 8B 16 0063 R
342 00F6 83 C2 04

M3:   MOV    SI,AX      ; MOVE COMMAND TO LOOK UP REGISTER
      SHR    SI,8       ; SHIFT COMMAND TO FORM BYTE OFFSET
      SAL    SI,1       ; TIMES 2 FOR WORD TABLE LOOKUP
      CMP    SI,MIL    ; TEST FOR WITHIN TABLE RANGE
      JNB    M4        ; BRANCH TO EXIT IF NOT A VALID COMMAND

M4:   JMP    VIDEO_RETURN ; COMMAND NOT VALID
      ENDP

VIDEO1
VIDEO_ID
VIDEO_RETURN
ENDP

;-----SET_MODE
;-----THIS ROUTINE INITIALIZES THE ATTACHMENT TO
;-----THE SELECTED MODE. THE SCREEN IS BLANKED.
;-----INPUT
;-----(AL) = MODE SELECTED (RANGE 0-7)
;-----OUTPUT
;-----NONE
;-----SET_MODE
;-----PROC  NEAR
;-----MOV    DX,03D4H   ; ADDRESS OF COLOR CARD
;-----CMP    DI,30H     ; IS B/W CARD INSTALLED
;-----JNE    M1         ; NO, WITH COLOR
;-----MOV    AL,7       ; INDICATE INTERNAL BW CARD MODE
;-----MOV    DL,0B4H     ; ADDRESS OF BW (MONOCHROME) CARD

M8:   MOV    @CRT_MODE,AL ; SAVE MODE IN GLOBAL VARIABLE
      MOV    @ADDR_6845,DX ; SAVE ADDRESS OF BASE
      MOV    @ROWS,25-1    ; INITIALIZE DEFAULT ROW COUNT OF 25
      PUSH   DS          ; SAVE POINTER TO DATA SEGMENT
      PUSH   AX          ; SAVE MODE NUMBER (AL)
      CBW
      MOV    SI,AX      ; CLEAR HIGH BYTE OF MODE
      MOV    AL,CS:[SI + OFFSET M7] ; SET TABLE POINTER, INDEXED BY MODE
      MOV    @INIT_MODE_SET,AL ; GET THE MODE NUMBER VALUE FROM TABLE
      AND    AL,037H    ; SAVE MODE NUMBER SET
      PUSH   DX          ; VIDEO OFF, SAVE HIGH RESOLUTION BIT
      ADD    DX,4       ; SAVE OUTPUT PORT VALUE
      OUT    DX,AL      ; POINT TO CONTROL REGISTER
      POP    DX          ; RESET VIDEO TO OFF TO SUPPRESS ROLLING
      XOR    DX,DX      ; BACK TO BASE REGISTER

      ASSUME DS:AB50 ; SET UP FOR ABS0 SEGMENT
      SUB    BX,BX      ; ESTABLISH VECTOR TABLE ADDRESSING
      MOV    DS,BX      ; GET POINTER TO VIDEO PARMS
      LDS    BX,@PARM_PTR

      ASSUME DS:CODE ; GET ADDRESS OF CODE SEGMENT
      POP    AX          ; RECOVER MODE NUMBER IN (AL)
      CMP    AX,CX,16   ; LENGTH OF EACH ROW OF TABLE
      CMP    AL,2       ; DETERMINE WHICH ONE TO USE
      JC    M9         ; MODE IS 0 OR 1
      ADD    BX,CX      ; NEXT ROW OF INITIALIZATION TABLE
      CMP    AL,4       ; MODE IS 1 OR 3
      ADD    BX,CX      ; MODE IS 2 OR 3
      CMP    AL,7       ; MODE IS 4,5, OR 6
      ADD    BX,CX      ; MODE IS 4,5, OR 6
      XOR    AH,AH      ; MOVE TO B/W CARD ROW OF INIT_TABLE
      ASSUME DS:CODE ; MOVE TO B/W CARD ROW OF INIT_TABLE

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

M9:   PUSH   AX          ; OUT. INIT.
      MOV    AX,[BX+10]  ; SAVE MODE IN (AL)
      XCHG   AH,AL      ; GET THE CURSOR MODE FROM THE TABLE
      PUSH   DS          ; PUT CURSOR MODE IN CORRECT POSITION
      ASSUME DS:DATA ; SAVE TABLE SEGMENT POINTER
      CALL   DS:DS      ; POINT DS TO DATA SEGMENT
      MOV    @CURSOR_MODE,AX ; PLACE INTO BIOS DATA SAVE AREA
      ASSUME DS:CODE ; RESTORE THE TABLE SEGMENT POINTER
      POP    DS          ; AH IS REGISTER NUMBER DURING LOOP
      XOR    AH,AH      ; RECOVER SEGMENT VALUE

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

M10:  MOV    AL,AH      ; INITIALIZATION LOOP
      OUT    DX,AL      ; GET 6845 REGISTER NUMBER
      INC    DX,1L      ; POINT TO DATA PORT
      INC    AH        ; NEXT REGISTER VALUE
      MOV    AL,[BX]    ; GET TABLE VALUE
      OUT    DX,AL      ; OUT TO CHIP
      INC    BX        ; NEXT IN TABLE
      INC    DS,1L      ; BACK TO POINTER REGISTER
      DEC    DX        ; DO THE WHOLE TABLE
      LOOP  M10        ; GET MODE BACK INTO (AL)
      POP    DS          ; RECOVER SEGMENT VALUE
      ASSUME DS:DATA ; FILL REGEN AREA WITH BLANKS

;-----FILL REGEN AREA WITH BLANKS

M11:  XOR    DI,DI      ; SET UP POINTER FOR REGEN
      MOV    @CRT_START,DI ; START ADDRESS SAVED IN GLOBAL
      MOV    @ACTIVE_PAGE,0 ; SET PAGE VALUE
      MOV    CX,8192-    ; NUMBER OF WORDS IN COLOR CARD
      CMP    AL,4       ; TEST FOR GRAPHICS
      JC    M12         ; NO GRAPHICS INIT
      CMP    AL,7       ; TEST FOR B/W CARD
      JE    M11         ; B/W CARD INIT
      XOR    AX,AX      ; FILL FOR GRAPHICS MODE
      SHORT M13        ; CLEAR BUFFER
      XOR    AX,AX      ; B/W CARD INIT
      BufferSize=1024 ; BUFFER SIZE ON BW CARD (2048)
      BufferSize=512   ; NO GRAPHICS INIT
      BufferSize=1024 ; FILL CHAR FOR ALPHA + ATTRIBUTE
      BufferSize=512   ; CLEAR BUFFER
      BufferSize=1024 ; FILL THE REGEN BUFFER WITH BLANKS

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

M12:  MOV    DX,@ADDR_6845 ; PREPARE TO OUTPUT TO VIDEO ENABLE PORT
      ADD    DX,4       ; POINT TO THE MODE CONTROL REGISTER

;-----VIDEO1
;-----5-145

```

```

343 00F9 A0 0065 R      MOV     AL,OCR_MODE_SET      ; GET THE MODE SET VALUE
344 00FC EE      OUT    DX,AL      ; SET VIDEO ENABLE PORT
345
346
347      ;----- DETERMINE NUMBER OF COLUMNS, BOTH FOR ENTIRE DISPLAY
348      ;----- AND THE NUMBER TO BE USED FOR TTY INTERFACE
349 00FD 2E: 8A B4 0000 E  MOV     AL,CS:[SI + OFFSET M6] ; GET NUMBER OF COLUMNS ON THIS SCREEN
350 0102 98      CBW
351 0103 A3 004A R      MOV     OCR_COLC,AX      ; INITIALIZE NUMBER OF COLUMNS COUNT
352
353      ;----- SET CURSOR POSITIONS
354
355 0106 81 E6 000E      AND     S1,000EH      ; WORD OFFSET INTO CLEAR LENGTH TABLE
356 010A 2E: 8B B4 0000 E  MOV     AX,CS:[S1 + OFFSET M5] ; LENGTH TO CLEAR
357 010F A3 004C R      MOV     OCR_LEN,AX      ; SAVE LENGTH OF CRT -- NOT USED FOR BW
358 0110 B9 0008      MOV     CX,8
359 0115 00 0050 R      MOV     [SI+OFFSET OCR_CURSOR_POSN]
360 0118 1E      PUSH    DS
361 0119 07      POP     ES      ; ESTABLISH SEGMENT
362 011A 33 C0      XOR    AX,AX      ; ADDRESSING
363 011C F3/ AB      REP    STOSW      ; FILL WITH ZEROES
364
365      ;----- SET UP OVERSCAN REGISTER
366
367 011E 42      INC    DX      ; SET OVERSCAN PORT TO A DEFAULT
368 011F B0 30      MOV    AL,30H      ; 30H VALUE FOR ALL MODES EXCEPT 640X200
369 0121 80 3E 0049 R 06  CMP    OCR_MODE,6      ; SEE IF THE MODE IS 640X200 BW
370 0126 75 02      JNZ    M14      ; IF NOT 640X200, THEN GO TO REGULAR
371 0127 80 3F      MOV    AL,3FH      ; IF IT IS 640X200, THEN PUT IN 3FH
372 012A
373 012A EE      OUT    DX,AL      ; OUTPUT THE CORRECT VALUE TO 3D9 PORT
374 012B A2 0066 R      MOV    OCR_PALETTE,AL ; SAVE THE VALUE FOR FUTURE USE
375
376      ;----- NORMAL RETURN FROM ALL VIDEO RETURNS
377
378 012E RETURN:      VIDEO_RETURN:      POP    BP
379 012E 5D      POP    DI
380 012F 5F      POP    SI
382 0131 5B      POP    BX
383 0132
384 0132 59      M15:      POP    CX      ; VIDEO_RETURN_C
385 0133 5A      POP    DX
386 0134 1F      POP    DS
387 0135 07      POP    ES      ; RECOVER SEGMENTS
388 0136 CF      IRET
389 0137
390      ;----- SET_MODE
391      ;----- SET_CTYPE
392      ;----- THIS ROUTINE SETS THE CURSOR VALUE
393      ;----- INPUT (CX) HAS CURSOR VALUE CH-START LINE, CL-STOP LINE
394      ;----- OUTPUT NONE
395
396      ;----- SET_CTYPE
397 0137      SET_MODE      PROC   NEAR
398 0137 B4 0A      MOV    AH,10      ; 6845 REGISTER FOR CURSOR SET
399 0139 B9 0E 0060 R  MOV    OCR_MODE,CX      ; SAVE IN DATA AREA
400 013D E8 0142 R      CALL   M16      ; OUTPUT CX REGISTER
401 0140 EB EC      JMP    VIDEO_RETURN
402
403      ;----- THIS ROUTINE OUTPUTS THE CX REGISTER TO THE 6845 REGISTERS NAMED IN (AH)
404
405      ;----- M16:
406 0142 8B 16 0063 R  M16:      MOV    DX,0ADDR_6845      ; ADDRESS REGISTER
407 0146 8A C4      MOV    AL,AH      ; GET VALUE
408 014A EE      OUT    DX,AL      ; REGISTER SET
409 0149 42      INC    DX
410 014A EB 00      JMP    $+2      ; DATA REGISTER
411 014B C5      MOV    CX,CH      ; I/O DELAY
412 014E EC      OUT    DX,AL      ; DATA
413 014F 4A      DEC    DX
414 0150 8A C4      MOV    AL,AH      ; POINT TO OTHER DATA REGISTER
415 0152 FE C0      INC    AL      ; SET FOR SECOND REGISTER
416 0154 EE      OUT    DX,AL
417 0156 EB 00      INC    DX      ; I/O DELAY
418 0158 8A C1      MOV    CX,CL      ; SECOND DATA VALUE
419 015A EE      OUT    DX,AL
420 015B C3      RET
421 015C
422
423 015C      SET_CTYPE      ENDP
424
425      ;----- SET_CPOS
426      ;----- THIS ROUTINE SETS THE CURRENT CURSOR POSITION TO THE
427      ;----- NEW X-Y VALUES PASSED
428      ;----- INPUT DX - ROW,COLUMN OF NEW CURSOR
429      ;----- BH - DISPLAY PAGE OF CURSOR
430      ;----- OUTPUT
431      ;----- CURSOR IS SET AT 6845 IF DISPLAY PAGE IS CURRENT DISPLAY
432
433 015C 8A C7      SET_CPOS      PROC   NEAR
434 015C 00 0000 R  MOV    AL,BH      ; MOVE PAGE NUMBER TO WORK REGISTER
435 015D 00 0000 R  ADD    CX,BH      ; DIVIDE PAGE TO WORD VALUE
436 015F D1 E0      SAL    AX,1      ; WORD OFFSET
437 0161 96      XCHG   AX,S1      ; USE INDEX REGISTER
438 0162 B9 94 0050 R  MOV    [SI+OFFSET OCR_CURSOR_POSN],DX ; SAVE THE POINTER
439 0164 38 3E 0062 R  CMP    OCR_ACTIVE_PAGE,BH
440 016A 75 05      JNZ    M17      ; SET_CPOS RETURN
441 016C BB C2      MOV    AX,DX      ; GET THE ROW,COLUMN TO AX
442 016E E8 0173 R  CALL   M18      ; CURSOR SET
443 0171 EB BB      M17:      JMP    VIDEO_RETURN      ; SET_CPOS RETURN
444 0173
445 0173      SET_CPOS      ENDP
446
447 0173 00 0000 R  ;----- SET CURSOR POSITION, AX HAS ROW/COLUMN FOR CURSOR
448 0173
449 0173      M18      PROC   NEAR
450 0173 00 0000 R  POSITION      ; DETERMINE LOCATION IN REGEN BUFFER
451 0176 BB C8      MOV    CX,AX
452 0178 03 0E 004E R  ADD    CX,OCR_START      ; ADD IN THE START ADDRESS FOR THIS PAGE
453 017C D1 F9      SAR    CX,1      ; DIVIDE BY 2 FOR CHAR ONLY COUNT

```

```

457 017E B4 0E      MOV     AH,14      ; REGISTER NUMBER FOR CURSOR
458 0180 EB 0142 R  CALL    M16      ; OUTPUT THE VALUE TO THE 6845
459 0183 C3          RET
460 0184          M18      ENDP

461          ;-----+
462          ; READ_CURSOR
463          ; THIS ROUTINE READS THE CURRENT CURSOR VALUE FROM THE
464          ; 6845, FORMATS IT, AND SENDS IT BACK TO THE CALLER
465          ; INPUT
466          ; BH - PAGE OF CURSOR
467          ; OUTPUT
468          ; DX - ROW, COLUMN OF THE CURRENT CURSOR POSITION
469          ; CX - CURRENT CURSOR MODE
470          ;-----+
471 0184          READ_CURSOR PROC NEAR
472 0184 8A DF      MOV     BH,BH
473 0186 32 FF      XOR     BH,BH
474 0188 D1 E3      SAL     BX,1      ; WORD OFFSET
475 018A BB 97 0050 R MOV     DX,[BX+OFFSET @CURSOR_POSN]
476 018E BB 0E 0060 R MOV     CX,@CURSOR_MODE
477 018F 00 00       POP    BP
478 0193 8F          POP    DI
479 0194 5E          POP    SI
480 0195 5B          POP    BX
481 0196 58          POP    AX      ; DISCARD SAVED CX AND DX
482 0197 5B          POP    AX
483 0198 1F          POP    DS
484 0199 07          POP    ES
485 019A CF          IRET
486 019B          READ_CURSOR ENDP
487          ;-----+
488          ; ACT_DISP_PAGE
489          ; THIS ROUTINE SETS THE ACTIVE DISPLAY PAGE, ALLOWING
490          ; THE FULL USE OF THE MEMORY SET ASIDE FOR THE VIDEO ATTACHMENT
491          ; INPUT
492          ; AL HAS THE NEW ACTIVE DISPLAY PAGE
493          ; OUTPUT
494          ; THE 6845 IS RESET TO DISPLAY THAT PAGE
495          ;-----+
496 019B          ACT_DISP_PAGE PROC NEAR
497 019B A2 0062 R  MOV     @ACTIVE_PAGE,AL  ; SAVE ACTIVE PAGE VALUE
498 019E BB 0E 004C R MOV     CX,@CRT_LEN  ; GET SAVED LENGTH OF REGEN BUFFER
499 01A2 98          CBW
500 01A3 50          PUSH   AX
501 01A4 F0 E1      MUL    CX      ; CONVERT AL TO WORD
502 01A4 A3 004E R  MOV     AX,START,AX  ; DISPLAY PAGE TIMES REGEN LENGTH
503 01A9 BB C8      MOV     CX,AX  ; START ADDRESS FOR LATER
504 01AB D1 F9      SAR    CX,1      ; START ADDRESS TO CX
505 01AD B4 0C      MOV     AH,12  ; DIVIDE BY 2 FOR 6845 HANDLING
506 01AF EB 0142 R  CALL   M16
507 01B0 00 00       POP    BX
508 01B3 91 E3      SAL    BX,1      ; RECOVER PAGE VALUE
509 01B5 BB 87 0050 R MOV     AX,[BX + OFFSET @CURSOR_POSN]  ; * FOR WORD OFFSET
510 01B9 E9 0173 R  CALL   M18  ; GET CURSOR FOR THIS PAGE
511 01BC E9 012E R  JMP    VIDEO_RETURN  ; SET THE CURSOR POSITION
512 01BF          ACT_DISP_PAGE ENDP
513          ;-----+
514          ; SET COLOR
515          ; THIS ROUTINE WILL ESTABLISH THE BACKGROUND COLOR, THE OVERSCAN COLOR,
516          ; AND THE FOREGROUND COLOR SET FOR MEDIUM RESOLUTION GRAPHICS
517          ; INPUT
518          ; (BH) HAS COLOR ID
519          ; IF BH=0, THE BACKGROUND COLOR VALUE IS SET
520          ; FROM THE LOW BITS OF BL(0-3)
521          ; IF BH=1, THE PALETTE SELECTION IS MADE
522          ; BASED ON THE LOW BIT OF BL:
523          ; 0 = GREEN, RED, YELLOW FOR COLORS 1,2,3
524          ; 1 = BLUE, CYAN, MAGENTA FOR COLORS 1,2,3
525          ; (BL) HAS THE COLOR VALUE TO BE USED
526          ; OUTPUT
527          ; THE COLOR SELECTION IS UPDATED
528          ;-----+
529 01BF          SET_COLOR PROC NEAR
530 01BF BB 16 0063 R MOV     DX,@ADDR_6845  ; I/O PORT FOR PALETTE
531 01C3 B3 C2 05    ADD    DX,5      ; OVERSCAN PORT
532 01C4 00 00 0066 R MOV     AL,@CRT_PALETTE  ; GET CURRENT PALETTE VALUE
533 01C9 04 F0      OR     BH,BH  ; THIS COLOR?
534 01CB 75 0E      JNZ    M20
535          ;-----+
536          ; HANDLE COLOR 0 BY SETTING THE BACKGROUND COLOR
537 01CD 24 E0      AND    AL,0E0H  ; TURN OFF LOW 5 BITS OF CURRENT
538 01CF 80 E3 1F    AND    BL,01FH  ; TURN OFF HIGH 3 BITS OF INPUT VALUE
539 01D2 00 C3      OR     AL,BL  ; PUT VALUE INTO REGISTER
540 01D4          M19:    OUT   DX,AL  ; OUTPUT THE PALETTE
541          ;-----+
542 01D4 EE          OUT   DX,AL  ; OUTPUT COLOR SELECTION TO 3D9 PORT
543 01D5 A2 0066 R  MOV    @CRT_PALETTE,AL  ; SAVE THE COLOR VALUE
544 01D8 E9 012E R  JMP    VIDEO_RETURN
545          ;-----+
546          ; HANDLE COLOR 1 BY SELECTING THE PALETTE TO BE USED
547 01DB          M20:    AND    AL,0DFH  ; TURN OFF PALETTE SELECT BIT
548 01DB 24 DF      SHR    BL,1    ; TEST THE LOW ORDER BIT OF BL
549 01E0 90 EB      JNC    M19  ; IF AL=0, WE'RE ALREADY DONE
550 01E1 73 F3      OR     AL,20H  ; TURN ON PALETTE SELECT BIT
551 01E3 EB EF      JMP    M19  ; GO DO IT
552 01E5          SET_COLOR ENDP
553          ;-----+
554          ; VIDEO_STATE
555          ; RETURNS THE CURRENT VIDEO STATE IN AX
556          ; AH = NUMBER OF COLUMNS ON THE SCREEN
557          ; AL = CURRENT VIDEO MODE
558          ; BH = CURRENT ACTIVE PAGE
559          ;-----+
560 01E5          VIDEO_STATE PROC NEAR
561 01E5 BB 26 004A R MOV     AH,BYTE PTR @CRT_COLS  ; GET NUMBER OF COLUMNS
562 01E9 A0 0049 R  MOV     AL,@CRT_MODE  ; CURRENT MODE
563 01EC 8A 3E 0062 R MOV     BH,@ACTTVE_PAGE  ; GET CURRENT ACTIVE PAGE
564 01F0 5D          POP    BP
565 01F1 5F          POP    DI
566 01F2 00 00       POP    SI
567 01F3 59          POP    CX
568 01F4 E9 0132 R  JMP    M15  ; DISCARD SAVED BX
569 01F4 E9 0132 R  JMP    M15  ; RETURN TO CALLER

```

```

571 01F7  VIDEO_STATE  ENDP
572
573
574
575
576
577
578
579
580
581 01F7  ;----- POSITION
582 01F7 53  THIS SERVICE ROUTINE CALCULATES THE REGEN BUFFER ADDRESS
583 01F8 89 D8  OF A CHARACTER IN THE ALPHA MODE
584 01FA 84 C4  INPUT
585 01FC F6 26 004A R AX = ROW, COLUMN POSITION
586 0200 32 FF  OUTPUT
587 0202 03 C3  AX = OFFSET OF CHAR POSITION IN REGEN BUFFER
588 0204 D1 E0
589 0206 00 5B
590 0207 C3
591 0208
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608 0208  ASSUME DS:DATA,ES:DATA
609
610 0208 E8 02E4 R SCROLL_UP  PROC  NEAR
611 0208 80 FC 04  CALL TEST_LINE_COUNT
612 020E 72 08  CMP AH,4
613 0210 80 FC 07  JC N1 : HANDLE SEPARATELY
614 0212 74 03  CMP AH,7
615 0215 E9 04A3 R JE N1 : TEST FOR BW CARD
616 0218
617 0218 53  N1: PUSH BX : UP_CONTINUE
618 0219 BB C1  MOV AX,CX : SAVE_FILL_ATTRIBUTE_IN BH
619 021B E8 0255 R CALL SCROLL_POSITION : UPPER_LEFT_POSITION
620 0220 30 00  ADD DI,BP : DO SETUP FOR SCROLL
621 0220 03 00  JZ N7 : BLACK_FIELD
622 0220 8A E6  ADD S1,AX : FROM ADDRESS
623 0224 2A E3  MOV AH,DH : # ROWS_IN_BLOCK
624 0226  SUB AH,BL : # ROWS_TO_BE_MOVED
625 0226 E8 0297 R N2: PUSH DI : ROW_LOOP
626 0229 03 55  ADD S1,BP : MOVE ONE ROW
627 0229 FD 00  CALL SCROLL_POSITION : POINT_TO_NEXT_LINE_IN_BLOCK
628 022D FE CC  ADD DI,BP : COUNT_OF_LINES_TO_MOVE
629 022F 75 F5  DEC AH : ROW_LOOP
630 0231
631 0231 58  N3: JNZ N2 : CLEAR_ENTRY
632 0232 B0 20  POP AX : RECOVER_ATTRIBUTE_IN_AH
633 0232 00 00  MOV AL,` ` : FILL_WITH_BLANKS
634 0234 E8 02A0 R N4: CALL SCROLL_POSITION : CLEAR_LOOP
635 0237 03 FD  ADD D1,BP : POINT_TO_NEXT_LINE
636 0239 FE CB  DEC BL : COUNTER_OF_LINES_TO_SCROLL
637 023B 75 F7  JNZ N4 : CLEAR_LOOP
638 023D
639 0240 E8 0000 E N5: CALL SCROLL_END : SCROLL_END
640 0240 80 3E 0049 R 07 CALL DDS : IS THIS THE BLACK AND WHITE CARD
641 0245 74 07  CMP `CRT_MODE,? : IF SO, SKIP THE MODE RESET
642 0247 A0 0065 R JE N6 : GET THE VALUE OF THE MODE SET
643 024A BA 03D8  MOV AL,`CRT_MODE_SET` : ALWAYS_SET_COLOR_CARD_PORT
644 024D EE  OUT DX,AL
645 024E
646 0251 E9 012E R N6: JMP VIDEO_RETURN : VIDEO_RET_HERE
647 0251
648 0251 8A DE  N7: MOV BL,DH : BLANK_FIELD
649 0253 EB DC  JMP N3 : GET_ROW_COUNT
650 0255 SCROLL_UP  ENDP : GO CLEAR THAT AREA
651
652
653
654
655 0255  ASSUME DS:DATA,ES:DATA
656 0255 E8 01F7 R SCROLL_POSITION  PROC  NEAR
657 0258 03 00 004E R CALL POSITION : CONVERT_TO_REGEN_POINTER
658 025C BB F8  ADD AX,`CRT_START` : OFFSET_OF_ATTRIBUTE_PAGE
659 0260 00 00  MOV D1,AX : TO_ADDRESS_FOR_SCROLL
660 0260 2B D1  MOV S1,AX : FROM_ADDRESS_FOR_SCROLL
661 0262 FE C6  SUB DX,CX : DX = #ROWS, #COLS_IN_BLOCK
662 0264 FE C2  INC DH : INCREMENT_FOR_0_ORIGIN
663 0266 BB 26 004A R XOR CH,CH : SET_HIGH_BYTE_OF_COUNT_TO_ZERO
664 026C FE ED  MOV BP,`CRT_COLS` : GET_NUMBER_OF_COLUMNS_IN_DISPLAY
665 026E BB 00 00  ADD BP,BP : INDEX FOR_ATTRIBUTE_BYT
666 0270 26 004A R MOV AL,` ` : GET_LINE_COUN
667 0274 03 C0  MUL BYTE PTR `CRT_COLS` : DETERMINE_OFFSET_TO_FROM_ADDRESS
668 0276 00 5B  ADD AX,AX : #2 FOR_ATTRIBUTE_BYT
669 0277 A0 0049 R PUSH AX : SAVE_LINE_COUN
670 0278 00 00  MOV AL,`CRT_MODE` : GET_CURRENT_MODE
671 027B 1F 00  ESTABLISH_ADDRESSING_TO_REGEN_BUFFER
672 027C 3C 02  CMP AL,2 : FOR_BW_POINTS
673 027E 72 13  JB N9 : TEST_FOR_COLOR_CARD_SPECIAL_CASES_HERE
674 0280 3C 03  CMP AL,3 : HAVE_TO_HANDLE_80X25_SEPARATELY
675 0282 77 0F  JA N9
676
677
678 0284 52  ;----- PUSH DX : 80X25 COLOR CARD SCROLL
679 0285 BA 03DA  MOV DX,3DAH : GUARANTEED_TO_BE_COLOR_CARD_HERE
679 0285
680 0288 EC  N8: IN AL,DX : WAIT_DISP_ENABLE
681 0289 A8 08  TEST AL,`VRVT` : GET_PORT
682 028B 00 00  JZ N9 : WAIT_FOR_VERTICAL_RETRACE
683 028D B0 25  MOV AL,25H : WAIT_DISP_ENABLE
684 028F B2 D8  MOV DL,0DH : ADDRESS_CONTROL_PORT

```

```

685 0291 EE          OUT    DX,AL      ; TURN OFF VIDEO DURING VERTICAL RETRACE
686 0290 5A          POP    DX
687 0293              N9:   POP    AX      ; RESTORE LINE COUNT
688 0293 58          OR     BL,BL      ; 0 SCROLL MEANS BLANK FIELD
689 0294 0A          RET
690 0296 C3          SCROLL_POSITION ENDP
691 0297
692
693
694 0297              N10:  PROC  NEAR
695 0297 8A CA        MOV    CL,DL      ; GET # OF COLS TO MOVE
696 0299 56          PUSH   SI
697 029A 57          PUSH   DI      ; SAVE START ADDRESS
698 029B F3/A        REP    MOVSW      ; MOVE THAT LINE ON SCREEN
699 029C 57          POP    DI
700 029E 5E          POP    SI      ; RECOVER ADDRESSES
701 029F C3          RET
702 02A0              N10:  ENDP
703
704
705 02A0              N11:  PROC  NEAR
706 02A0 8A CA        MOV    CL,DL      ; GET # COLUMNS TO CLEAR
707 02A2 57          PUSH   DI
708 02A3 F3/AB       REP    STOSW      ; STORE THE FILL CHARACTER
709 02A5 5F          POP    DI
710 02A6 C3          RET
711 02A7              N11:  ENDP
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728 02A7              N12:  PROC  NEAR
729 02A7 FD          CALL   TEST_LINE_COUNT      ; DIRECTION FOR SCROLL DOWN
730 02A8 E8 02E4 R    CMP    AH,4        ; TEST FOR GRAPHICS
731 02A9 80 FC 04     JC    N12
732 02A8 72 08        CMP    AH,7        ; TEST FOR BW CARD
733 02B0 80 FC 07     JE    N12
734 02B3 74 03        JMP   GRAPHICS_DOWN
735 02B4 E9 04FA R    N12:  PUSH  BX      ; CONTINUE DOWN
736 02B6              CALL   SCROLL_POSITION      ; SAVE ATTRIBUTE IN BH
737 02B8 53          MOV    AX,DX      ; LOWER RIGHT CORNER
738 02B9 8B C2        CALL   SCROLL_POSITION      ; GET REGEN LOCATION
739 02B8 E8 0255 R    JZ    N16
740 02B8 74 20        SUB   SI,AX      ; SI IS FROM ADDRESS
741 02C0 2B F0        MOV    AH,DH      ; GET TOTAL # ROWS
742 02C1 6A E5        SUB   AH,BL      ; COUNT TO MOVE IN SCROLL
743 02C2 2A E3        N13:  CALL   N10      ; MOVE ONE ROW
744 02C6              PUSH  BX
745 02C6 E8 0297 R    SUB   SI,BP
746 02C9 2B F5        CALL   SCROLL_POSITION      ; MOVE ONE ROW
747 02CB 2B FD        SUB   DI,BP
748 02C9 FE CC        DEC   AH
749 02CF 75 F5        JNZ   N13
750 02D1              N14:  POP   AX      ; RECOVER ATTRIBUTE IN AH
751 02D1 58          MOV   AL,' '
752 02D2 B0 20        N15:  CALL   N11      ; CLEAR ONE ROW
753 02D4              POP   AX
754 02D4 E8 02A0 R    CALL   SCROLL_POSITION      ; GO TO NEXT ROW
755 02D7 2B FD        SUB   DI,BP
756 02D9 FE CB        DEC   BL
757 02DB 75 F7        JNZ   N15
758 02DD E9 023D R    JMP   N5      ; SCROLL_END
759 02E0              N16:  MOV   BL,DH
760 02E1 8A DE        MOV   AL,AL      ; SAVE LINE COUNT IN BL
761 02E2 EB ED        OR    AL,AL      ; TEST IF AL IS ALREADY ZERO
762 02E4              SCROLL_DOWN      ; IF IT IS THEN RETURN...
763
764
765
766
767 02E4              TEST_LINE_COUNT PROC NEAR
768
769 02E4 8A D8        MOV   BL,AL      ; SAVE LINE COUNT IN BL
770 02E6 0A C0        OR    AL,AL      ; TEST IF AL IS ALREADY ZERO
771 02E6 74 0E        JZ    BL_SET      ; IF IT IS THEN RETURN...
772
773 02EB 8A C6        PUSH  AX      ; SAVE AX
774 02ED 2A C5        MOV   AL,DH      ; SUBTRACT LOWER ROW FROM UPPER ROW
775 02EF FE C0        SUB   AL,CH
776 02F1 3A C3        INC   AL
777 02F3 58          CMP   AL,BL      ; ADJUST DIFFERENCE BY 1
778 02F4 75 02        POP   AX      ; LINE COUNT = AMOUNT OF ROWS IN WINDOW?
779 02F6 2A DB        JNE   BL_SET      ; RESTORE AX
780 02F8              BL_SET:  RET      ; IF NOT THEN WE'RE ALL SET
781 02F8 C3          TEST_LINE_COUNT ENDP      ; OTHERWISE SET BL TO ZERO
782 02F9

```

```

783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799 02F9
800 02F9 80 FC 04
801 02FC 72 08
802
803 02FE 80 FC 07
804 0301 74 03
805
806 0303 E9 062A R
807 0306
808 0306 E8 0322 R
809 0309 88 F7
810 0308 06
811 030C 1F
812
813
814
815 030D 0A DB
816 0310 75 0D
817 0311
818 0311 FB
819 0312 90
820 0313 FA
821 0314 EC
822 0315 A8 01
823 0316 75 F9
824 0319
825 0319 EC
826 031A A8 09
827 031C 74 FB
828 031E
829 031F AD
830 031F E9 012E R
831
832 0322
833
834
835 0322
836 0322 86 E3
837 0324 88 E8
838 0326 80 EB 02
839 0329 00 EB
840 032B 88 F3
841 032C 88 F7
842 032F 32 FF
843 0331 88 FB
844 0333 D1 E7
845 0335 88 85 0050 R
846 0339 74 09
847
848 033B 33 FF
849 033D
850 033D 03 3E 004C R
851 0341 4B
852 0342 75 F9
853 0343
854 0344 E8 01FF R
855 0347 03 F8
856 0349 88 16 0063 R
857 034D 83 C2 06
858 0350 88 DE
859 0352 C3
860
861
862 0353

PAGE
;----- READ_AC_CURRENT
;----- THIS ROUTINE READS THE ATTRIBUTE AND CHARACTER AT THE CURRENT
;----- CURSOR POSITION AND RETURNS THEM TO THE CALLER
;----- INPUT
;----- (AH) = CURRENT CRT MODE
;----- (BH) = DISPLAY PAGE ( ALPHA MODES ONLY )
;----- (DS) = DATA SEGMENT
;----- (ES) = REGEN SEGMENT
;----- OUTPUT
;----- (AL) = CHARACTER READ
;----- (AH) = ATTRIBUTE READ
;----- ASSUME DS:DATA,ES:DATA

READ_AC_CURRENT PROC NEAR
    CMP AH,4          ; IS THIS GRAPHICS
    JC P10
    CMP AH,7          ; IS THIS BW CARD
    JE P10
    JMP GRAPHICS_READ
P10:   GRAPHICS_READ
    CALL FIND_POSITION
    MOV SI,DT          ; GET REGEN LOCATION AND PORT ADDRESS
    PUSH ES            ; ESTABLISH ADDRESSING IN SI
    POP DS             ; GET REGEN SEGMENT FOR QUICK ACCESS

;----- WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
    OR BL,BL           ; CHECK MODE FLAG FOR COLOR CARD IN 80
    JNZ P13            ; ELSE SKIP RETRACE WAIT - DO FAST READ
P11:   STI              ; WAIT FOR HORIZONTAL RETRACE LOW OR VERTICAL
    NOP              ; ENABLE INTERRUPTS FIRST
    CLI              ; ALLOW FOR SMALL INTERRUPT WINDOW
    IN AL,DX           ; BLOCK INTERRUPTS FOR SINGLE LOOP
    TEST AL,RHRZ
    JNZ P11
P12:   IN AL,DX           ; GET STATUS FROM THE ADAPTER
    TEST AL,RRVT+RHRZ
    JZ P12
    P13:  LODSW            ; IS HORIZONTAL RETRACE LOW
    JMP VIDEO_RETURN   ; IS HORIZONTAL OR VERTICAL RETRACE HIGH
    P14:  GET THE CHARACTER AND ATTRIBUTE
    P15:  EXIT WITH (AX)
    READ_AC_CURRENT ENDP

FIND_POSITION PROC NEAR
    XCHG AH,BL           ; SETUP FOR BUFFER READ OR WRITE
    MOV BP,AX             ; SAVE CHARACTER/ATTR IN (BP) REGISTER
    SUB BL,2              ; CONVERT DISPLAY MODE TYPE TO A
    SHR BL,1              ; ZERO VALUE FOR COLOR IN 80 COLUMN
    MOV SI,BX             ; AND SAVE (2 OR 3 --> ZERO)
    MOV BL,BH             ; MOVE DISPLAY PAGE TO LOW BYTE
    XOR BH,BH             ; CLEAR HIGH BYTES OF COUNT/BYTE OFFSET
    MOV DX,BH             ; MOVE DISPLAY PAGE (COUNT) TO WORK REG
    SAL DI,1              ; TIMES 2 FOR WORD OFFSET
    MOV AX,[DI+OFFSET_CURSOR_POSN] ; GET ROW/COLUMN OF THAT PAGE
    JZ P21                ; SKIP BUFFER ADJUSTMENT IF PAGE ZERO
P20:   XOR DI,DI           ; ELSE SET BUFFER START ADDRESS TO ZERO
    ADD DI,0CRT_LEN        ; ADD LENGTH OF BUFFER FOR ONE PAGE
    DEC BX                ; DECREMENT PAGE COUNT
    JNZ P20                ; LOOP TILL PAGE COUNT EXHAUSTED

P21:   CALL POSITION        ; DETERMINE LOCATION IN REGEN IN PAGE
    ADD DI,AX             ; ADD LOCATION TO START OF REGEN PAGE
    MOV DX,0ADDR_6845        ; GET BASE ADDRESS OF ACTIVE DISPLAY
    INP PORT              ; POINT AT STATUS PORT
    P22:  RECOVER_CONVERTED_MODE_TYPE_IN(BL) ; RECOVER CONVERTED MODE TYPE IN (BL)
    P23:  DX=ATTRIBUTE/CHARACTER (FROM PORT/AL) ; DX=PORT ADDRESS OF BUFFER
    P24:  DX=STATUS PORT ADDRESS OF ADAPTER ; BL=MODE FLAG (ZERO FOR 80x25 COLOR)
    FIND_POSITION ENDP

```

```

863 PAGE
864
865 ;----- WRITE_AC_CURRENT
866 ;----- THIS ROUTINE WRITES THE ATTRIBUTE AND CHARACTER
867 ;----- AT THE CURRENT CURSOR POSITION
868
869 INPUT
870 ;----- (AH) = CURRENT CRT MODE
871 ;----- (BH) = DISPLAY PAGE
872 ;----- (CX) = COUNT OF CHARACTERS TO WRITE
873 ;----- (AL) = CHAR TO WRITE
874 ;----- (BL) = ATTRIBUTE OF CHAR TO WRITE
875 ;----- (DS) = DATA SEGMENT
876 ;----- (ES) = REGEN SEGMENT
877
878 OUTPUT
879 ;----- DISPLAY REGEN BUFFER UPDATED
880
881 0353 WRITE_AC_CURRENT PROC NEAR
882 0353 80 FC 04
883 0356 80 FB 05
884 0358 80 F0 07
885 035B 74 03
886 0350 E9 0582 R
887 0360
888 0361 E8 0322 R
889 0363 0A DB
890 0365 74 06
891
892 0367 95
893 0368 F3/ AB
894 036A EB 16
895
896
897 ;----- P30: WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
898 036C
899 036C 95
900 036D
901 036E FB
902 036E 90
903 036F FA
904 0370 EC
905 0371 A8 08
906 0373 75 09
907 0374 A8 01
908 0377 75 F4
909 0379
910 0379 EC
911 037A A8 09
912 037C 74 FB
913
914 037E 95
915 037F AB
916 0380 E2 EA
917 0382
918 0383 E9 012E R
919
920 0385
921
922 ;----- WRITE_C_CURRENT
923 ;----- THIS ROUTINE WRITES THE CHARACTER AT
924 ;----- THE CURRENT CURSOR POSITION, ATTRIBUTE UNCHANGED
925
926 INPUT
927 ;----- (AH) = CURRENT CRT MODE
928 ;----- (BH) = DISPLAY PAGE
929 ;----- (CX) = COUNT OF CHARACTERS TO WRITE
930 ;----- (AL) = CHAR TO WRITE
931 ;----- (DS) = DATA SEGMENT
932 ;----- (ES) = REGEN SEGMENT
933
934 OUTPUT
935 ;----- DISPLAY REGEN BUFFER UPDATED
936
937 0385 WRITE_C_CURRENT PROC NEAR
938 0385 80 FC 04
939 0388 80 FB 05
940 038A 80 F0 07
941 038D 74 03
942 038F E9 0582 R
943 0392
944 0392 E8 0322 R
945
946
947 ;----- P40: WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
948 0395
949 0395 FB
950 0396 0A DB
951 0396 00 0F
952 039A FA
953 039A FA
954 0398 EC
955 039C A8 08
956 039E 75 09
957 03A0 A8 01
958 03A1 75 F1
959 03A4
960 03A5 EC
961 03A5 A8 09
962 03A7 74 FB
963 03A9
964 03A9 8B C5
965 03AB AA
966 03AC 47
967 03AD E2 E6
968
969 03AF E9 012E R
970
971 ;----- P41: WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
972 0395
973 0395 FB
974 0396 0A DB
975 0396 00 0F
976 039A FA
977 039A FA
978 0398 EC
979 039C A8 08
980 039E 75 09
981 03A0 A8 01
982 03A1 75 F1
983 03A4
984 03A5 EC
985 03A5 A8 09
986 03A7 74 FB
987 03A9
988 03A9 8B C5
989 03AB AA
990 03AC 47
991 03AD E2 E6
992
993 ;----- P42: WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
994 0395
995 0395 FB
996 0396 0A DB
997 0396 00 0F
998 039A FA
999 039A FA
1000 0398 EC
1001 039C A8 08
1002 039E 75 09
1003 03A0 A8 01
1004 03A1 75 F1
1005 03A4
1006 03A5 EC
1007 03A5 A8 09
1008 03A7 74 FB
1009 03A9
1010 03A9 8B C5
1011 03AB AA
1012 03AC 47
1013 03AD E2 E6
1014
1015 ;----- P43: WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
1016 0395
1017 0395 FB
1018 0396 0A DB
1019 0396 00 0F
1020 039A FA
1021 039A FA
1022 0398 EC
1023 039C A8 08
1024 039E 75 09
1025 03A0 A8 01
1026 03A1 75 F1
1027 03A4
1028 03A5 EC
1029 03A5 A8 09
1030 03A7 74 FB
1031 03A9
1032 03A9 8B C5
1033 03AB AA
1034 03AC 47
1035 03AD E2 E6
1036
1037 ;----- P44: WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
1038 0395
1039 0395 FB
1040 0396 0A DB
1041 0396 00 0F
1042 039A FA
1043 039A FA
1044 0398 EC
1045 039C A8 08
1046 039E 75 09
1047 03A0 A8 01
1048 03A1 75 F1
1049 03A4
1050 03A5 EC
1051 03A5 A8 09
1052 03A7 74 FB
1053 03A9
1054 03A9 8B C5
1055 03AB AA
1056 03AC 47
1057 03AD E2 E6
1058
1059 ;----- P45: WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
1060 0395
1061 0395 FB
1062 0396 0A DB
1063 0396 00 0F
1064 039A FA
1065 039A FA
1066 0398 EC
1067 039C A8 08
1068 039E 75 09
1069 03A0 A8 01
1070 03A1 75 F1
1071 03A4
1072 03A5 EC
1073 03A5 A8 09
1074 03A7 74 FB
1075 03A9
1076 03A9 8B C5
1077 03AB AA
1078 03AC 47
1079 03AD E2 E6
1080
1081 ;----- P46: WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
1082 0395
1083 0395 FB
1084 0396 0A DB
1085 0396 00 0F
1086 039A FA
1087 039A FA
1088 0398 EC
1089 039C A8 08
1090 039E 75 09
1091 03A0 A8 01
1092 03A1 75 F1
1093 03A4
1094 03A5 EC
1095 03A5 A8 09
1096 03A7 74 FB
1097 03A9
1098 03A9 8B C5
1099 03AB AA
1100 03AC 47
1101 03AD E2 E6
1102
1103 ;----- P47: WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
1104 0395
1105 0395 FB
1106 0396 0A DB
1107 0396 00 0F
1108 039A FA
1109 039A FA
1110 0398 EC
1111 039C A8 08
1112 039E 75 09
1113 03A0 A8 01
1114 03A1 75 F1
1115 03A4
1116 03A5 EC
1117 03A5 A8 09
1118 03A7 74 FB
1119 03A9
1120 03A9 8B C5
1121 03AB AA
1122 03AC 47
1123 03AD E2 E6
1124
1125 ;----- P48: WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
1126 0395
1127 0395 FB
1128 0396 0A DB
1129 0396 00 0F
1130 039A FA
1131 039A FA
1132 0398 EC
1133 039C A8 08
1134 039E 75 09
1135 03A0 A8 01
1136 03A1 75 F1
1137 03A4
1138 03A5 EC
1139 03A5 A8 09
1140 03A7 74 FB
1141 03A9
1142 03A9 8B C5
1143 03AB AA
1144 03AC 47
1145 03AD E2 E6
1146
1147 ;----- P49: WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
1148 0395
1149 0395 FB
1150 0396 0A DB
1151 0396 00 0F
1152 039A FA
1153 039A FA
1154 0398 EC
1155 039C A8 08
1156 039E 75 09
1157 03A0 A8 01
1158 03A1 75 F1
1159 03A4
1160 03A5 EC
1161 03A5 A8 09
1162 03A7 74 FB
1163 03A9
1164 03A9 8B C5
1165 03AB AA
1166 03AC 47
1167 03AD E2 E6
1168
1169 ;----- P50: WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
1170 0395
1171 0395 FB
1172 0396 0A DB
1173 0396 00 0F
1174 039A FA
1175 039A FA
1176 0398 EC
1177 039C A8 08
1178 039E 75 09
1179 03A0 A8 01
1180 03A1 75 F1
1181 03A4
1182 03A5 EC
1183 03A5 A8 09
1184 03A7 74 FB
1185 03A9
1186 03A9 8B C5
1187 03AB AA
1188 03AC 47
1189 03AD E2 E6
1190
1191 ;----- P51: WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
1192 0395
1193 0395 FB
1194 0396 0A DB
1195 0396 00 0F
1196 039A FA
1197 039A FA
1198 0398 EC
1199 039C A8 08
1200 039E 75 09
1201 03A0 A8 01
1202 03A1 75 F1
1203 03A4
1204 03A5 EC
1205 03A5 A8 09
1206 03A7 74 FB
1207 03A9
1208 03A9 8B C5
1209 03AB AA
1210 03AC 47
1211 03AD E2 E6
1212
1213 ;----- P52: WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
1214 0395
1215 0395 FB
1216 0396 0A DB
1217 0396 00 0F
1218 039A FA
1219 039A FA
1220 0398 EC
1221 039C A8 08
1222 039E 75 09
1223 03A0 A8 01
1224 03A1 75 F1
1225 03A4
1226 03A5 EC
1227 03A5 A8 09
1228 03A7 74 FB
1229 03A9
1230 03A9 8B C5
1231 03AB AA
1232 03AC 47
1233 03AD E2 E6
1234
1235 ;----- P53: WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
1236 0395
1237 0395 FB
1238 0396 0A DB
1239 0396 00 0F
1240 039A FA
1241 039A FA
1242 0398 EC
1243 039C A8 08
1244 039E 75 09
1245 03A0 A8 01
1246 03A1 75 F1
1247 03A4
1248 03A5 EC
1249 03A5 A8 09
1250 03A7 74 FB
1251 03A9
1252 03A9 8B C5
1253 03AB AA
1254 03AC 47
1255 03AD E2 E6
1256
1257 ;----- P54: WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
1258 0395
1259 0395 FB
1260 0396 0A DB
1261 0396 00 0F
1262 039A FA
1263 039A FA
1264 0398 EC
1265 039C A8 08
1266 039E 75 09
1267 03A0 A8 01
1268 03A1 75 F1
1269 03A4
1270 03A5 EC
1271 03A5 A8 09
1272 03A7 74 FB
1273 03A9
1274 03A9 8B C5
1275 03AB AA
1276 03AC 47
1277 03AD E2 E6
1278
1279 ;----- P55: WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
1280 0395
1281 0395 FB
1282 0396 0A DB
1283 0396 00 0F
1284 039A FA
1285 039A FA
1286 0398 EC
1287 039C A8 08
1288 039E 75 09
1289 03A0 A8 01
1290 03A1 75 F1
1291 03A4
1292 03A5 EC
1293 03A5 A8 09
1294 03A7 74 FB
1295 03A9
1296 03A9 8B C5
1297 03AB AA
1298 03AC 47
1299 03AD E2 E6
1300
1301 ;----- P56: WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
1302 0395
1303 0395 FB
1304 0396 0A DB
1305 0396 00 0F
1306 039A FA
1307 039A FA
1308 0398 EC
1309 039C A8 08
1310 039E 75 09
1311 03A0 A8 01
1312 03A1 75 F1
1313 03A4
1314 03A5 EC
1315 03A5 A8 09
1316 03A7 74 FB
1317 03A9
1318 03A9 8B C5
1319 03AB AA
1320 03AC 47
1321 03AD E2 E6
1322
1323 ;----- P57: WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
1324 0395
1325 0395 FB
1326 0396 0A DB
1327 0396 00 0F
1328 039A FA
1329 039A FA
1330 0398 EC
1331 039C A8 08
1332 039E 75 09
1333 03A0 A8 01
1334 03A1 75 F1
1335 03A4
1336 03A5 EC
1337 03A5 A8 09
1338 03A7 74 FB
1339 03A9
1340 03A9 8B C5
1341 03AB AA
1342 03AC 47
1343 03AD E2 E6
1344
1345 ;----- P58: WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
1346 0395
1347 0395 FB
1348 0396 0A DB
1349 0396 00 0F
1350 039A FA
1351 039A FA
1352 0398 EC
1353 039C A8 08
1354 039E 75 09
1355 03A0 A8 01
1356 03A1 75 F1
1357 03A4
1358 03A5 EC
1359 03A5 A8 09
1360 03A7 74 FB
1361 03A9
1362 03A9 8B C5
1363 03AB AA
1364 03AC 47
1365 03AD E2 E6
1366
1367 ;----- P59: WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
1368 0395
1369 0395 FB
1370 0396 0A DB
1371 0396 00 0F
1372 039A FA
1373 039A FA
1374 0398 EC
1375 039C A8 08
1376 039E 75 09
1377 03A0 A8 01
1378 03A1 75 F1
1379 03A4
1380 03A5 EC
1381 03A5 A8 09
1382 03A7 74 FB
1383 03A9
1384 03A9 8B C5
1385 03AB AA
1386 03AC 47
1387 03AD E2 E6
1388
1389 ;----- P60: WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
1390 0395
1391 0395 FB
1392 0396 0A DB
1393 0396 00 0F
1394 039A FA
1395 039A FA
1396 0398 EC
1397 039C A8 08
1398 039E 75 09
1399 03A0 A8 01
1400 03A1 75 F1
1401 03A4
1402 03A5 EC
1403 03A5 A8 09
1404 03A7 74 FB
1405 03A9
1406 03A9 8B C5
1407 03AB AA
1408 03AC 47
1409 03AD E2 E6
1410
1411 ;----- P61: WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
1412 0395
1413 0395 FB
1414 0396 0A DB
1415 0396 00 0F
1416 039A FA
1417 039A FA
1418 0398 EC
1419 039C A8 08
1420 039E 75 09
1421 03A0 A8 01
1422 03A1 75 F1
1423 03A4
1424 03A5 EC
1425 03A5 A8 09
1426 03A7 74 FB
1427 03A9
1428 03A9 8B C5
1429 03AB AA
1430 03AC 47
1431 03AD E2 E6
1432
1433 ;----- P62: WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
1434 0395
1435 0395 FB
1436 0396 0A DB
1437 0396 00 0F
1438 039A FA
1439 039A FA
1440 0398 EC
1441 039C A8 08
1442 039E 75 09
1443 03A0 A8 01
1444 03A1 75 F1
1445 03A4
1446 03A5 EC
1447 03A5 A8 09
1448 03A7 74 FB
1449 03A9
1450 03A9 8B C5
1451 03AB AA
1452 03AC 47
1453 03AD E2 E6
1454
1455 ;----- P63: WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
1456 0395
1457 0395 FB
1458 0396 0A DB
1459 0396 00 0F
1460 039A FA
1461 039A FA
1462 0398 EC
1463 039C A8 08
1464 039E 75 09
1465 03A0 A8 01
1466 03A1 75 F1
1467 03A4
1468 03A5 EC
1469 03A5 A8 09
1470 03A7 74 FB
1471 03A9
1472 03A9 8B C5
1473 03AB AA
1474 03AC 47
1475 03AD E2 E6
1476
1477 ;----- P64: WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
1478 0395
1479 0395 FB
1480 0396 0A DB
1481 0396 00 0F
1482 039A FA
1483 039A FA
1484 0398 EC
1485 039C A8 08
1486 039E 75 09
1487 03A0 A8 01
1488 03A1 75 F1
1489 03A4
1490 03A5 EC
1491 03A5 A8 09
1492 03A7 74 FB
1493 03A9
1494 03A9 8B C5
1495 03AB AA
1496 03AC 47
1497 03AD E2 E6
1498
1499 ;----- P65: WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
1500 0395
1501 0395 FB
1502 0396 0A DB
1503 0396 00 0F
1504 039A FA
1505 039A FA
1506 0398 EC
1507 039C A8 08
1508 039E 75 09
1509 03A0 A8 01
1510 03A1 75 F1
1511 03A4
1512 03A5 EC
1513 03A5 A8 09
1514 03A7 74 FB
1515 03A9
1516 03A9 8B C5
1517 03AB AA
1518 03AC 47
1519 03AD E2 E6
1520
1521 ;----- P66: WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
1522 0395
1523 0395 FB
1524 0396 0A DB
1525 0396 00 0F
1526 039A FA
1527 039A FA
1528 0398 EC
1529 039C A8 08
1530 039E 75 09
1531 03A0 A8 01
1532 03A1 75 F1
1533 03A4
1534 03A5 EC
1535 03A5 A8 09
1536 03A7 74 FB
1537 03A9
1538 03A9 8B C5
1539 03AB AA
1540 03AC 47
1541 03AD E2 E6
1542
1543 ;----- P67: WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
1544 0395
1545 0395 FB
1546 0396 0A DB
1547 0396 00 0F
1548 039A FA
1549 039A FA
1550 0398 EC
1551 039C A8 08
1552 039E 75 09
1553 03A0 A8 01
1554 03A1 75 F1
1555 03A4
1556 03A5 EC
1557 03A5 A8 09
1558 03A7 74 FB
1559 03A9
1560 03A9 8B C5
1561 03AB AA
1562 03AC 47
1563 03AD E2 E6
1564
1565 ;----- P68: WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
1566 0395
1567 0395 FB
1568 0396 0A DB
1569 0396 00 0F
1570 039A FA
1571 039A FA
1572 0398 EC
1573 039C A8 08
1574 039E 75 09
1575 03A0 A8 01
1576 03A1 75 F1
1577 03A4
1578 03A5 EC
1579 03A5 A8 09
1580 03A7 74 FB
1581 03A9
1582 03A9 8B C5
1583 03AB AA
1584 03AC 47
1585 03AD E2 E6
1586
1587 ;----- P69: WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
1588 0395
1589 0395 FB
1590 0396 0A DB
1591 0396 00 0F
1592 039A FA
1593 039A FA
1594 0398 EC
1595 039C A8 08
1596 039E 75 09
1597 03A0 A8 01
1598 03A1 75 F1
1599 03A4
1600 03A5 EC
1601 03A5 A8 09
1602 03A7 74 FB
1603 03A9
1604 03A9 8B C5
1605 03AB AA
1606 03AC 47
1607 03AD E2 E6
1608
1609 ;----- P70: WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
1610 0395
1611 0395 FB
1612 0396 0A DB
1613 0396 00 0F
1614 039A FA
1615 039A FA
1616 0398 EC
1617 039C A8 08
1618 039E 75 09
1619 03A0 A8 01
1620 03A1 75 F1
1621 03A4
1622 03A5 EC
1623 03A5 A8 09
1624 03A7 74 FB
1625 03A9
1626 03A9 8B C5
1627 03AB AA
1628 03AC 47
1629 03AD E2 E6
1630
1631 ;----- P71: WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
1632 0395
1633 0395 FB
1634 0396 0A DB
1635 0396 00 0F
1636 039A FA
1637 039A FA
1638 0398 EC
1639 039C A8 08
1640 039E 75 09
1641 03A0 A8 01
1642 03A1 75 F1
1643 03A4
1644 03A5 EC
1645 03A5 A8 09
1646 03A7 74 FB
1647 03A9
1648 03A9 8B C5
1649 03AB AA
1650 03AC 47
1651 03AD E2 E6
1652
1653 ;----- P72: WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
1654 0395
1655 0395 FB
1656 0396 0A DB
1657 0396 00 0F
1658 039A FA
1659 039A FA
1660 0398 EC
1661 039C A8 08
1662 039E 75 09
1663 03A0 A8 01
1664 03A1 75 F1
1665 03A4
1666 03A5 EC
1667 03A5 A8 09
1668 03A7 74 FB
1669 03A9
1670 03A9 8B C5
1671 03AB AA
1672 03AC 47
1673 03AD E2 E6
1674
1675 ;----- P73: WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
1676 0395
1677 0395 FB
1678 0396 0A DB
1679 0396 00 0F
1680 039A FA
1681 039A FA
1682 0398 EC
1683 039C A8 08
1684 039E 75 09
1685 03A0 A8 01
1686 03A1 75 F1
1687 03A4
1688 03A5 EC
1689 03A5 A8 09
1690 03A7 74 FB
1691 03A9
1692 03A9 8B C5
1693 03AB AA
1694 03AC 47
1695 03AD E2 E6
1696
1697 ;----- P74: WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
1698 0395
1699 0395 FB
1700 0396 0A DB
1701 0396 00 0F
1702 039A FA
1703 039A FA
1704 0398 EC
1705 039C A8 08
1706 039E 75 09
1707 03A0 A8 01
1708 03A1 75 F1
1709 03A4
1710 03A5 EC
1711 03A5 A8 09
1712 03A7 74 FB
1713 03A9
1714 03A9 8B C5
1715 03AB AA
1716 03AC 47
1717 03AD E2 E6
1718
1719 ;----- P75: WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
1720 0395
1721 0395 FB
1722 0396 0A DB
1723 0396 00 0F
1724 039A FA
1725 039A FA
1726 0398 EC
1727 039C A8 08
1728 039E 75 09
1729 03A0 A8 01
1730 03A1 75 F1
1731 03A4
1732 03A5 EC
1733 03A5 A8 09
1734 03A7 74 FB
1735 03A9
1736 03A9 8B C5
1737 03AB AA
1738 03AC 47
1739 03AD E2 E6
1740
1741 ;----- P76: WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
1742 0395
1743 0395 FB
1744 0396 0A DB
1745 0396 00 0F
1746 039A FA
1747 039A FA
1748 0398 EC
1749 039C A8 08
1750 039E 75 09
1751 03A0 A8 01
1752 03A1 75 F1
1753 03A4
1754 03A5 EC
1755 03A5 A8 09
1756 03A7 74 FB
1757 03A9
1758 03A9 8B C5
1759 03AB AA
1760 03AC 47
1761 03AD E2 E6
1762
1763 ;----- P77: WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
1764 0395
1765 0395 FB
1766 0396 0A DB
1767 0396 00 0F
1768 039A FA
1769 039A FA
1770 0398 EC
1771 039C A8 08
1772 039E 75 09
1773 03A0 A8 01
1774 03A1 75 F1
1775 03A4
1776 03A5 EC
1777 03A5 A8 09
1778 03A7 74 FB
1779 03A9
1780 03A9 8B C5
1781 03AB AA
1782 03AC 47
1783 03AD E2 E6
1784
1785 ;----- P78: WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
1786 0395
1787 0395 FB
1788 0396 0A DB
1789 0396 00 0F
1790 039A FA
1791 039A FA
1792 0398 EC
1793 039C A8 08
1794 039E 75 09
1795 03A0 A8 01
1796 03A1 75 F1
1797 03A4
1798 03A5 EC
1799 03A5 A8 09
1800 03A7 74 FB
1801 03A9
1802 03A9 8B C5
1803 03AB AA
1804 03AC 47
1805 03AD E2 E6
1806
1807 ;----- P79: WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
1808 0395
1809 0395 FB
1810 0396 0A DB
1811 0396 00 0F
1812 039A FA
1813 039A FA
1814 0398 EC
1815 039C A8 08
1816 039E 75 09
1817 03A0 A8 01
1818 03A1 75 F1
```



```

1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085 0435 E8 0469 R
1086 0435 261 8A 04
1088 043B 22 C4
1089 043D D2 E0
1090 043E 0A E6
1091 0441 D0 C0
1092 0443 E9 012E R
1093 0446
1094
1095 0446
1096 0446 50
1097 0447 50
1098 0448 E8 0469 R
1099 044B D2 E8
1100 044D 22 C4
1101 044E 261 8A 0C
1102 0452 00 00
1103 0453 F6 C3 80
1104 0456 75 0D
1105 0458 F6 D4
1106 045A 22 CC
1107 045C 0A C1
1108 045D 00 00
1109 045E 261 88 04
1110 0461 58
1111 0462 E9 012E R
1112 0465
1113 0465 32 C1
1114 0465 EB F5
1115 0469
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129 0469
1130
1131
1132
1133
1134 0469 93
1135 046A B0 28
1136 046C F6 E2
1137 046E A8 08
1138 0470 14 03
1139 0472 05 1FD8
1140 0475
1141 0475 96
1142 0476 93
1143 0477 88 D1
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153 0479 BB 02C0
1154 047C B9 0302
1155 047F 80 3E 0049 R 06
1156 0484 82 08
1157 0486 BB 0180
1158 0489 B9 0703
1159
1160
1161 048C
1162 048C 22 EA
1163
1164
1165
1166 048E D3 EA
1167 0490 03 F2
1168 0492 B8 F7
1169
1170
1171
1172 0494 2A C9
1173 0496
1174 0497 D0 C8
1175 0498 02 CD
1176 049A FE CF
1177 049C T5 F8
1178 049E 8A E3
1179 04A0 D2 EC
1180 04A2 C3

PAGE
-----
; READ DOT -- WRITE DOT
; THESE ROUTINES WILL WRITE A DOT, OR READ THE
; DOT AT THE INDICATED LOCATION
ENTRY -
; DX = ROW (0-199) (THE ACTUAL VALUE DEPENDS ON THE MODE)
; CX = COLUMN (0-639) (THE VALUES ARE NOT RANGE CHECKED)
; AL = DOT VALUE TO WRITE (1,2 OR 4 BITS DEPENDING ON MODE,
; REQUIRED FOR WRITE DOT ONLY, RIGHT JUSTIFIED)
; 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
SHR AL,CL
AND AL,AH
MOV DH,CL
ROL AL,CL
JMP VIDEO_RETURN
READ_DOT ENDP
;----- 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
;----- WRITE_DOT PROC NEAR
WRITE_DOT PROC NEAR
PUSH AX
CALL R3
MOV AL,ES:[SI]
AND AL,AH
SHR AL,CL
AND AL,AH
MOV DH,CL
ROL AL,CL
POP AX
TEST BL,80H
JNZ R2
NOT AH
AND CL,AH
OR AL,CL
MOV DS:[SI],AL
POP AX
JMP VIDEO_RETURN
R1: ; DETERMINE BYTE POSITION OF THE DOT
; SHIFT TO SET UP THE BITS FOR OUTPUT
; STRIP OFF THE OTHER BITS
; GET THE CURRENT BYTE
; REVERSE FOR FLAG
; IS IT ON
; YES, XON THE DOT
; SET MASK TO REMOVE THE INDICATED BITS
; OR IN THE NEW VALUE OF THOSE BITS
; FINISH DOT
; RESTORE THE BYTE IN MEMORY
R2: ; XOR AL,CL
; XOR_DOT
; EXCLUSIVE OR THE DOTS
; FINISH UP THE WRITING
;----- 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 = # OF BITS IN RESULT
; BX = MODIFIED
;----- R3 PROC NEAR
R3 PROC NEAR
;----- DETERMINE 1ST BYTE IN INDICATED ROW BY MULTIPLYING ROW VALUE BY 40
;----- 1 LOW BIT OF ROW DETERMINES EVEN/ODD, 80 BYTES/ROW
;----- XCHG AX,BX
;----- MOV AL,40
;----- MUL DL
;----- TEST AL,008H
;----- JZ R4
;----- ADD AX,200H-40
;----- R4: ; WILL SAVE AL AND AH DURING OPERATION
;----- XCHG SI,AX
;----- XCHG AX,BX
;----- MOV DX,CX
;----- AX= ADDRESS OF START OF INDICATED ROW
;----- TEST FOR EVEN/ODD ROW CALCULATED
;----- JZ R4 IF EVEN ROW
;----- DS=OFFSET LOCATION OF ODD ROWS ADJUST
;----- EVEN ROW
;----- MOVE^POINTER TO SI
;----- RECOVER AL AND AH VALUES
;----- COLUMN VALUE TO DX
;----- DETERMINE GRAPHICS MODE CURRENTLY IN EFFECT
;----- SET UP THE REGISTERS ACCORDING TO THE MODE
;----- CH = MASK FOR LOW OF COLUMN ADDRESS ( 7/3 FOR HIGH/MED RES )
;----- CL = # OF ADDRESS BITS IN COLUMN VALUE ( 3/2 FOR H/M )
;----- BL = MASK TO SELECT BITS FROM POINTEED BYTE ( 80H/COH FOR H/M )
;----- BH = NUMBER OF VALID BITS IN POINTEED BYTE ( 1/2 FOR H/M )
;----- MOV BX,2C0H
;----- MOV CX,302H
;----- CMP .FCRT_MODE,6
;----- JC R5
;----- R5: ; SET PARM FOR MED RES
;----- MOV BX,180H
;----- MOV CX,703H
;----- SET PARM FOR HIGH RES
;----- DETERMINE BIT OFFSET IN BYTE FROM COLUMN MASK
;----- R5: ; ADDRESS OF PEL WITHIN BYTE TO CH
;----- AND CH,DL
;----- DETERMINE BYTE OFFSET FOR THIS LOCATION IN COLUMN
;----- SHR DX,CL
;----- ADD SI,DX
;----- MOV DH,BH
;----- SET BYT OFFSET TO DH
;----- DETERMINE BH (VALID BITS IN BYTE) BY CH (BIT OFFSET)
;----- SUB CL,CL
;----- ZERO INTO STORAGE LOCATION
;----- R6: ; LEFT JUSTIFY VALUE IN AL (FOR WRITE)
;----- ROR AL,1
;----- ADD CL,CH
;----- DEC BH
;----- JNZ R6
;----- R6: ; LOOP CONTROL
;----- ON EXIT, CL HAS COUNT TO RESTORE BITS
;----- GET MASK TO AH
;----- MOVE THE MASK TO CORRECT LOCATION
;----- RETURN WITH EVERYTHING SET UP

```

```

181 04A3
182
183 ; SCROLL UP
184 ; THIS ROUTINE SCROLLS UP THE INFORMATION ON THE CRT
185 ; ENTRY --
186 ; CH,CL = UPPER LEFT CORNER OF REGION TO SCROLL
187 ; DH,DL = LOWER RIGHT CORNER OF REGION TO SCROLL
188 ; BOTH OF THE ABOVE ARE IN CHARACTER POSITIONS
189 ; BH = FILL VALUE FOR BLANKED LINES
190 ; AL = # LINES TO SCROLL (AL=0 MEANS BLANK THE ENTIRE FIELD)
191 ; DS = DATA SEGMENT
192 ; ES = REGEN SEGMENT
193 ; EXIT --
194
195 ; NOTHING, THE SCREEN IS SCROLLED
196 -----
197 04A3 8A D8
198 04A5 BB C1
199
200 ;----- USE CHARACTER SUBROUTINE FOR POSITIONING
201 ;----- ADDRESS RETURNED IS MULTIPLIED BY 2 FROM CORRECT VALUE
202
203 04A7 E8 06D8 R
204 04AA BB F8
205
206 ;----- DETERMINE SIZE OF WINDOW
207
208 04AC 2B D1
209 04AE 81 C2 0101
210 04B0 C0 E6 02
211
212 ;----- DETERMINE CRT MODE
213
214 04B5 80 3E 0049 R 06
215 04BA 73 04
216
217 ;----- MEDIUM RES UP
218 04BC D0 E2
219 04B0 D1 E7
220
221 ;----- DETERMINE THE SOURCE ADDRESS IN THE BUFFER
222 04C0
223 04C0 06
224 04C0 1F
225 04C2 2A ED
226 04C4 C0 E3 02
227 04C7 74 2D
228 04C8 A8 C3
229 04C9 B4 50
230 04D0 F6 E4
231 04D1 7C 77
232 04D1 03 0F
233 04D3 8A E6
234 04D5 2A E3
235
236 ;----- LOOP THROUGH, MOVING ONE ROW AT A TIME, BOTH EVEN AND ODD FIELDS
237 04D7 E8 0558 R
238 04DA 81 EE 1FB0
239 04DE 81 EF 1FB0
240 04E2 FE CC
241 04E4 75 F1
242
243 ;----- FILL IN THE VACATED LINE(S)
244
245 04E6
246 04E6 8A C7
247 04E8
248 04E8 E8 0571 R
249 04E9 81 EF 1FB0
250 04E9 80 CB
251 04E9 75 F5
252 04F3 E9 012E R
253
254 04F6
255 04F6 8A DE
256 04FB EB EC
257 04FA
258
259 ; SCROLL DOWN
260 ; THIS ROUTINE SCROLLS DOWN THE INFORMATION ON THE CRT
261 ; ENTRY --
262 ; CH,CL = UPPER LEFT CORNER OF REGION TO SCROLL
263 ; DH,DL = LOWER RIGHT CORNER OF REGION TO SCROLL
264 ; BOTH OF THE ABOVE ARE IN CHARACTER POSITIONS
265 ; BH = FILL VALUE FOR BLANKED LINES
266 ; AL = # LINES TO SCROLL (AL=0 MEANS BLANK THE ENTIRE FIELD)
267 ; DS = DATA SEGMENT
268 ; ES = REGEN SEGMENT
269 ; EXIT --
270
271 ; NOTHING, THE SCREEN IS SCROLLED
272
273 04FA
274 04FA FD
275 04FB 8A D8
276 04FD BB C2
277
278 ;----- USE CHARACTER SUBROUTINE FOR POSITIONING
279 ;----- ADDRESS RETURNED IS MULTIPLIED BY 2 FROM CORRECT VALUE
280
281 04FF E8 06D8 R
282 0502 BB F8
283
284 ;----- DETERMINE SIZE OF WINDOW
285
286 0504 2B D1
287 0506 81 C2 0101
288 050A C0 E6 02
289
290 ;----- DETERMINE CRT MODE
291
292 0500 80 3E 0049 R 06
293 0512 73 05
294
295 ;----- GRAPHICS_UP PROC NEAR
296
297 MOV BL,AL ; SAVE LINE COUNT IN BL
298 MOV AX,CX ; GET UPPER LEFT POSITION INTO AX REG
299
300 ;----- CALL GRAPH_POSN
301 MOV DI,AX ; SAVE RESULT AS DESTINATION ADDRESS
302
303 ;----- SUB DX,CX
304 ADD DX,101H ; ADJUST VALUES
305 SAL DH,2 ; MULTIPLY ROWS BY 4 AT 8 VERT DOTS/CHAR
306
307 ;----- CMP OCT_MODE,6
308 JNC R12 ; TEST FOR MEDIUM RES
309 ;----- FIND_SOURCE
310
311 ;----- PUSH ES
312 POP DS ; FIND_SOURCE, GET SEGMENTS BOTH POINTING TO REGEN
313 SUB CH,CH
314 SAL BL,2 ; ZERO TO HIGH OF COUNT REGISTER
315 ;----- JZ R11
316 ;----- IF ZERO, THEN BLANK ENTIRE FIELD
317 ;----- MOV AL,BL
318 ;----- GET NUMBER OF LINES IN AL
319 ;----- MOV AH,80
320 ;----- 80 BYTES/ROW
321 ;----- MUL AL,AH ; DETERMINE OFFSET TO SOURCE
322 ;----- ADD SI,DI ; SET UP SOURCE
323 ;----- ADD SI,AX ; ADD IN OFFSET TO IT
324 ;----- MOV AH,0H ; NUMBER OF ROWS IN FIELD
325 ;----- SUB AH,BL ; DETERMINE NUMBER TO MOVE
326
327 ;----- CALL R17 ; ROW_LOOP
328 ;----- SUB SI,2000H-80 ; MOVE ONE ROW
329 ;----- SUB DI,2000H-80 ; MOVE TO NEXT ROW
330
331 ;----- DEC AH ; NUMBER OF ROWS TO MOVE
332 ;----- JNZ R8 ; CONTINUE TILL ALL MOVED
333
334 ;----- R9: ;----- FILL_ENTRY
335 ;----- MOV AL,BH ; ATTRIBUTE TO FILL WITH
336
337 ;----- R10: ;----- CLEAR THAT ROW
338 ;----- CALL R18 ; POINT TO NEXT LINE
339 ;----- SUB DI,2000H-80 ; NUMBER OF LINES TO FILL
340 ;----- DEC BL ; CLEAR_LOOP
341 ;----- JNZ R10 ; EVERYTHING DONE
342 ;----- JMP VIDEO_RETURN
343
344 ;----- R11: ;----- BLANK_FIELD
345 ;----- MOV BL,DH ; SET BLANK COUNT TO EVERYTHING IN FIELD
346 ;----- JNC R9 ; CLEAR THE FIELD
347 ;----- ENDP
348
349 ;----- GRAPHICS_UP ENDP
350
351 ;----- GRAPHICS_DOWN PROC NEAR
352
353 STD ; SET DIRECTION
354 MOV BL,AL ; SAVE LINE COUNT IN BL
355 MOV AX,DX ; GET LOWER RIGHT POSITION INTO AX REG
356
357 ;----- CALL GRAPH_POSN
358 MOV DI,AX ; SAVE RESULT AS DESTINATION ADDRESS
359
360 ;----- SUB DX,CX
361 ADD DX,101H ; ADJUST VALUES
362 SAL DH,2 ; MULTIPLY ROWS BY 4 AT 8 VERT DOTS/CHAR
363
364 ;----- CMP OCT_MODE,6
365 JNC R12 ; TEST FOR MEDIUM RES
366 ;----- FIND_SOURCE_DOWN
367
368 ;----- END

```

```

1295 ;----- MEDIUM RES DOWN
1296 0514 D0 E2 ; SAL DL,1 ; # COLUMNS * 2, SINCE 2 BYTES/CHAR
1297 0516 D1 E7 ; SAL D1,1 ; OFFSET * 2 SINCE 2 BYTES/CHAR
1298 0518 47 ; INC DI ; POINT TO LAST BYTE
1299
1300 ;----- DETERMINE THE SOURCE ADDRESS IN THE BUFFER
1301 0519 ; R12: PUSH ES ; FIND SOURCE DOWN
1302 0519 06 ; POP DS ; BOTH SEGMENTS TO REGEN
1303 051A 1F ; SUB CH,CH ; ZERO TO HIGH OF COUNT REGISTER
1304 051B 2A E0 ; ADD DI,240 ; POINT TO LAST ROW OF PIXELS
1305 051C 20 00F0 ; SAL BL,4 ; MULTIPLE OF NUMBER OF LINES BY 4
1306 0521 20 E3 02 ; JZ R16 ; IF ZERO, THEN BLANK ENTIRE FIELD
1307 0524 74 2E ; MOV AL,BL ; GET NUMBER OF LINES IN AL
1308 0526 8A C3 ; MOV AH,80 ; 80 BYTES/ROW
1309 0528 BA 50 ; MUL AH ; DETERMINE OFFSET TO SOURCE
1310 052A FE E4 ; MOV SI,DI ; SOURCE COLUMN NUMBER
1311 052B 8B F7 ; SUB SI,AX ; SUBTRACT THE OFFSET
1312 052E 2B F0 ; MOV AH,DH ; NUMBER OF ROWS IN FIELD
1313 0530 8A E6 ; MOV SUB AH,BL ; DETERMINE NUMBER TO MOVE
1314 0532 2A E3 ; R15
1315
1316 ;----- LOOP THROUGH, MOVING ONE ROW AT A TIME, BOTH EVEN AND ODD FIELDS
1317 0534 ; R13: CALL R17 ; ROW LOOP DOWN
1318 0534 E8 0558 R ; SUB SI,2000H+80 ; MOVE ONE ROW
1319 0537 81 EE 2050 ; SUB DI,2000H+80 ; MOVE TO NEXT ROW
1320 053B 81 EF 2050 ; DEC AH ; NUMBER OF ROWS TO MOVE
1321 053F FE CC ; JNZ R13 ; CONTINUE TILL ALL MOVED
1322 0541 75 F1
1323
1324 ;----- FILL IN THE VACATED LINE(S)
1325 0543 ; R14: MOV AL,BH ; CLEAR ENTRY DOWN
1326 0543 8A C7 ; R15: CALL R18 ; ATTRIBUTE TO FILL WITH
1327 0545 ; SUB DI,2000H+80 ; CLEAR LOOP DOWN
1328 0546 E8 0571 R ; DEC BL ; CLEAR ROW DOWN
1329 0548 81 EF 2050 ; JNZ R15 ; POINT TO NEXT LINE
1330 054C FE CB ; CLD ; NUMBER OF LINES TO FILL
1331 054E 75 F5 ; JMP VIDEO_RETURN ; CLEAR LOOP DOWN
1332 0550 FC ; R16: MOV BL,DH ; RESET THE DIRECTION FLAG
1333 0551 E9 012E R ; R17: MOV R14 ; EVERYTHING DONE
1334
1335 0554 ; R16: MOV BL,DH ; BLANK DOWN
1336 0554 8A DE ; JMP R14 ; SET BLANK COUNT TO EVERYTHING IN FIELD
1337 0556 EB EB ; GRAPHICS_DOWN ; CLEAR THE FIELD
1338 0558
1339
1340 ;----- ROUTINE TO MOVE ONE ROW OF INFORMATION
1341
1342 0558 ; R17: PROC NEAR ; NUMBER OF BYTES IN THE ROW
1343 0558 8A CA ; MOV CL,DL ; SAVE POINTERS
1344 055A 56 ; PUSH SI ; MOVE THE EVEN FIELD
1345 055B 57 ; PUSH DI ; POINT TO THE ODD FIELD
1346 055C F3/ AA ; REP MOVSB ; SAVE THE POINTERS
1347 055E 5F ; POP DI ; COUNT BACK
1348 055F 5E ; PUSH DI ; MOVE THE ODD FIELD
1349 0560 81 C6 2000 ; ADD SI,2000H ; POP SI ; POINTERS BACK
1350 0564 81 C7 2000 ; ADD DI,2000H ; RET ; RETURN TO CALLER
1351 0568 56
1352 0569 57
1353 056A 8A CA ; R18: ENDP ; RETURN TO CALLER
1354 056C F3/ AA
1355 056E 5F
1356 056F 5E
1357 0570 C3
1358 0571
1359
1360 ;----- CLEAR A SINGLE ROW
1361
1362 0571 ; R18: PROC NEAR ; NUMBER OF BYTES IN FIELD
1363 0571 8A CA ; MOV CL,DL ; SAVE POINTER
1364 0573 57 ; PUSH DI ; STORE THE NEW VALUE
1365 0574 F3/ AA ; REP STOSB ; POINTER BACK
1366 0576 5F ; POP DI ; POINT TO ODD FIELD
1367 0577 81 C7 2000 ; ADD DI,2000H ; PUSH DI ; FILL THE ODD FIELD
1368 057B 57 ; MOV CL,DL ; RET ; RETURN TO CALLER
1369 057C 80 CA ; REP STOSB
1370 057E F3/ AA ; POP DI
1371 0580 5F ; RET
1372 0581 C3 ; R18: ENDP ; RETURN TO CALLER
1373 0582
1374
1375 ;----- GRAPHICS WRITE
1376 ; THIS ROUTINE WRITES THE ASCII CHARACTER TO THE CURRENT
1377 ; POSITION ON THE SCREEN.
1378 ; ENTRY --
1379 ; AL = CHARACTER TO WRITE
1380 ; BL = COLOR ATTRIBUTE TO BE USED FOR FOREGROUND COLOR
1381 ; CX = NUMBER OF CHARACTERS TO BE WRITTEN
1382 ; (0 IS USED FOR THE BACKGROUND COLOR)
1383 ; CX = NUMBER OF CHARS TO WRITE
1384 ; DS = DATA SEGMENT
1385 ; ES = REGEN SEGMENT
1386 ; EXIT --
1387 ; NOTHING IS RETURNED
1388
1389 ;----- GRAPHICS READ
1390 ; THIS ROUTINE READS THE ASCII CHARACTER AT THE CURRENT CURSOR
1391 ; POSITION ON THE SCREEN BY MATCHING THE DOTS ON THE SCREEN TO THE
1392 ; CHARACTER GENERATOR CODE POINTS
1393 ; ENTRY --
1394 ; NONE (0 IS ASSUMED AS THE BACKGROUND COLOR)
1395 ; EXIT --
1396 ; AL = CHARACTER READ AT THAT POSITION (0 RETURNED IF NONE FOUND)
1397
1398 ;----- FOR BOTH ROUTINES, THE IMAGES USED TO FORM CHARS ARE CONTAINED IN ROM
1399 ; FOR THE 1ST 128 CHARS. TO ACCESS CHARS IN THE SECOND HALF, THE USER
1400 ; MUST INITIALIZE THE VECTOR AT INTERRUPT 1FH (LOCATION 0007CH) TO
1401 ; POINT TO THE USER SUPPLIED TABLE OF GRAPHIC IMAGES (8X8 BOXES).
1402 ; FAILURE TO DO SO WILL CAUSE IN STRANGE RESULTS
1403
1404 ;----- ASSUME DS=DATA,ES=DATA
1405 0582 ; GRAPHICS_WRITE PROC NEAR ; ZERO TO HIGH OF CODE POINT
1406 0582 B4 00 ; MOV AH,0 ; SAVE CODE POINT VALUE
1407 0584 50

```

```

1409      ;----- DETERMINE POSITION IN REGEN BUFFER TO PUT CODE POINTS
1410      CALL    S26          ; FIND LOCATION IN REGEN BUFFER
1411 0585 E8 06D5 R  MOV    DI,AX      ; REGEN POINTER IN DI
1412 0588 BB F8
1413
1414      ;----- DETERMINE REGION TO GET CODE POINTS FROM
1415      POP    AX          ; RECOVER CODE POINT
1416 058A 58  CMP    AL,80H    ; IS IT IN SECOND HALF
1417 058B 3C 80  JAE    SI      ; YES
1418 058D 73 06
1419
1420      ;----- IMAGE IS IN FIRST HALF, CONTAINED IN ROM
1421      MOV    SI,OFFSET CRT_CHAR_GEN ; OFFSET OF IMAGES
1422 058F BE 0000 E  PUSH   CS          ; SAVE SEGMENT ON STACK
1423 0592 0E
1424 0593 EB 0F  JMP    SHORT S2      ; DETERMINE_MODE
1425
1426
1427      ;----- IMAGE IS IN SECOND HALF, IN USER MEMORY
1428 0595 S1:      SUB    AL,80H    ; EXTEND CHAR
1429 0595 2C 80  PUSH   DS          ; ZERO ORIGIN FOR SECOND HALF
1430 0597 1E
1431 0598 2B F6
1432 059A 8E 0D  MOV    DI,SI      ; SAVE DATA POINTER
1433
1434 059C 95 C5 36 007C R  ASSUME DS1AB50 ; ESTABLISH VECTOR ADDRESSING
1435 05A0 8C DA  LDS    SI,DX_PTR  ; GET THE OFFSET OF THE TABLE
1436
1437 05A2 1F  MOV    DX,DS      ; GET THE SEGMENT OF THE TABLE
1438 05A3 52  ASSUME DS:DATA  ; RECOVER DATA SEGMENT
1439
1440      ;----- DETERMINE GRAPHICS MODE IN OPERATION
1441      S2:      SAL    AX,3      ; DETERMINE_MODE
1442 05A4 CI E0 03  ADD    SI,AX    ; MULTIPLY CODE POINT VALUE BY 8
1443 05A7 03 F0
1444 05A9 80 3E 0049 R 06  CMP    #CRT_MODE,6 ; S1 HAS OFFSET OF DESIRED CODES
1445 05A6 0F
1446 05A7 T2 2C  POP    DS          ; RECOVER TABLE POINTER SEGMENT
1447 05A8 51  JC    S1      ; TEST FOR MEDIUM RESOLUTION MODE
1448
1449      ;----- HIGH RESOLUTION MODE
1450 05B1 S3:      PUSH   DI          ; HIGH_CHAR
1451 05B1 57  PUSH   SI          ; SAVE REGEN POINTER
1452 05B2 56  MOV    DH,4      ; SAVE CODE POINTER
1453 05B3 B6 04
1454 05B5
1455 05B5 AC  S4:      LODSB   TEST   BL,80H    ; NUMBER OF TIMES THROUGH LOOP
1456 05B5 C3 80  JNZ    S6          ; SHOULD USE THE FUNCTION
1457 05B9 75 16  STOSB   S6          ; TO PUT CHAR IN
1458 05B8 AA
1459 05B8 AC  LODSB
1460 05BD
1461 05BD 26: 88 85 1FFF  MOV    E5:[DI+2000H-1],AL ; STORE IN SECOND HALF
1462 05C0 05 00 C7 4F  ADD    DI,79    ; MOVE TO NEXT ROW IN REGEN
1463 05C5 FE CE  LODSB   DI
1464 05C7 75 EC  JNZ    S4          ; DONE WITH LOOP
1465 05C9 5E
1466 05CA 5F  POP    SI          ; RECOVER REGEN POINTER
1467 05CB 47  INC    DI          ; POINT TO NEXT CHAR POSITION
1468 05CC E2 E3  LOOP   S3          ; MORE CHARS TO WRITE
1469 05C9 E9 012E R  JMP    VIDEO_RETURN
1470
1471 05D1 S5:      XOR    AL,ES:[DI] ; EXCLUSIVE OR WITH CURRENT
1472 05D1 26: 32 05  STOSB   ; STORE THE CODE POINT
1473 05D4 AA
1474 05D5 AC  LODSB   ; AGAIN FOR ODD FIELD
1475 05D6 26: 32 85 1FFF  XOR    AL,ES:[DI+2000H-1] ; BACK TO MAINSTREAM
1476 05D8 EB E0
1477
1478      ;----- MEDIUM RESOLUTION WRITE
1479 05DD S7:      MOV    DL,BL    ; MED_RES_WRITE
1480 05D0 8A D3  SAL    DI,1      ; SAVE FOR COLOR BIT
1481 05D1 D1 E1
1482
1483 05E1 80 E3 03  AND    BL,3      ; OFFSET*2 SINCE 2 BYTES/CHAR
1484 05E4 B5 55  MOV    AL,055H    ; EXPAND BL TO FULL WORD OF COLOR
1485 05E6 F6 E3  MUL    BL
1486 05E8 8A D8  MOV    BL,AL
1487 05F0 8A F8  MOV    BH,AL
1488 05EC
1489 05EC 57  S8:      PUSH   DI          ; ISOLATE THE COLOR BITS ( LOW 2 BITS )
1490 05ED 56  PUSH   SI          ; GET BL CONVERSION MULTIPLIER
1491 05EB B8 04  MOV    DH,4      ; EXPAND 2 COLOR BITS TO 4 REPLICATIONS
1492 05F0
1493 05F0 AC  S9:      LODSB   TEST   DL,80H    ; PLACE BACK IN WORK REGISTER
1494 05F1 E8 06AD R  CALL   S21        ; EXPAND TO 8 REPLICATIONS OF COLOR BITS
1495 05F4 23 C3  AND    AX,BX    ; MED_CHAR
1496 05F6 80 E0  XCHG   AH,AL    ; SAVE REGEN POINTER
1497 05F8 F6 C2 80  TEST   DL,80H    ; SAVE THE CODE POINTER
1498 05F9 74 03  JZ    S10        ; NUMBER OF LOOPS
1499 05FD 26: 33 05  XOR    AX,ES:[DI] ; GET CODE POINT
1500 0600
1501 0600 26: 89 05  S10:      MOV    ES:[DI],AX ; DOUBLE UP ALL THE BITS
1502 0603 AC  LODSB   S21        ; CONVERT TO FOREGROUND COLOR ( 0 BACK )
1503 0604 E8 06AD R  CALL   S21        ; SWAP HIGH/LOW BYTES FOR WORD MOVE
1504 0606 80 E0  AND    AX,BX    ; AGAIN, IS THIS XOR FUNCTION
1505 0609 86 E0  XCHG   AH,AL    ; NO, STORE IT IN AS IT IS
1506 060B F6 C2 80  TEST   DL,80H    ; DO FUNCTION WITH LOW/HIGH
1507 060E 74 05  JZ    S11        ; FUNCTION WITH FIRST HALF LOW
1508 0610 26: 33 85 2000  XOR    AX,ES:[DI+2000H] ; STORE FIRST BYTE HIGH, SECOND LOW
1509 0615
1510 0615 26: 89 85 2000  S11:      MOV    ES:[DI+2000H],AX ; GET CODE POINT
1511 0616 80 C1 50  ADD    DI,80    ; STORE SECOND PORTION HIGH
1512 0616 FE CE  DEC    DI        ; POINT TO NEXT LOCATION
1513 061F 75 CF  JNZ    S9        ; KEEP GOING
1514 0621 5E  POP    SI          ; RECOVER CODE POINTER
1515 0622 5F  POP    DI          ; RECOVER REGEN POINTER
1516 0623 47  INC    DI          ; POINT TO NEXT CHAR POSITION
1517 0624 47
1518 0625 E2 C5  LOOP   S8        ; MORE TO WRITE
1519 0627 E9 012E R  JMP    VIDEO_RETURN
1520 062A
1521
1522 ;----- GRAPHICS_WRITE ENDP

```

```

1523
1524 062A
1525 062A E8 06D5 R
1526 062D 8B F0
1527 062F 83 EC 08
1528 0632 8B EC
1529
1530
1531
1532 0634 80 3E 0049 R 06
1533 0639 06
1534 063A 1F
1535 063B 72 19
1536
1537
1538
1539
1540 063D B6 04
1541 063F 8B 04
1542 0641 8B 46 00
1544 0644 45
1545 0645 8A 84 2000
1546 0649 8A 46 00
1547 064B 45
1548 064D 43 C6 50
1549 0650 FE CE
1550 0652 75 EB
1551 0654 E8 16
1552
1553
1554 0655
1555 0656 D1 E6
1556 0658 B6 04
1557 065A
1558 065A E8 06BC R
1559 065D 81 C6 1FFE
1560 065E 81 00 00BC R
1561 0664 B1 EE 1FB2
1562 0668 FE CE
1563 066A 75 EE
1564
1565
1566 066C
1567 066C BF 0000 E
1568 066F 0E
1569 0670 07
1570 0671 83 ED 08
1571 0674 88 F5
1572 0676 FC
1573 0678 B0 00
1574 0679
1575 0679 16
1576 067A 1F
1577 067B B8 0080
1578 067E
1579 0680 56
1580 067F 57
1581 0680 B9 0004
1582 0683 F3/ A7
1583 0685 5F
1584 0686 5E
1585 0687 41 1E
1586 0689 FE C0
1587 068B B3 C7 08
1588 068E 4A
1589 068F 75 ED
1590
1591
1592
1593 0691 3C 00
1594 0693 74 12
1595 0695 2B C0
1596 0697 8E D8
1597
1598 0699 C4 3E 007C R
1599 069D 8C C0
1600 069F 0B C7
1601 06A1 74 04
1602 06A3 B8 80
1603 06A5 E8 D2
1604
1605
1606
1607 06A7
1608 06A7 83 C4 08
1609 06A9 E9 012E R
1610 06AD
1611
1612
1613
1614
1615
1616
1617 06AD
1618 06AD 51
1619 06AE B9 0008
1620 06B1
1621 06B3 00 C8
1622 06B3 D1 00
1623 06B5 D1 FD
1624 06B7 E2 F8
1625
1626 06B9 95
1627 06B8 59
1628 06B9 C3
1629 06BC
1630
1631
1632
1633
1634
1635
1636

;----- GRAPHICS_READ PROC NEAR
    CALL S26
    MOV SI,AX
    SUB SP,8
    MOV BP,SP
;----- DETERMINE GRAPHICS MODES
    CMP 0CRT_MODE,6
    PUSH ES
    POP DS
    JC S13
;----- HIGH RESOLUTION READ
    ;----- GET VALUES FROM REGEN BUFFER AND CONVERT TO CODE POINT
    MOV DH,4
;----- MEDIUM RESOLUTION READ
    S12:
    MOV AL,[SI]
    MOV [BP],AL
    INC BP
    MOV AL,[SI+2000H]
    MOV [BP],AL
    INC BP
    ADD SI,80
    DEC DH
    JNZ S12
    JMP SHORT S15
;----- MEDIUM RESOLUTION READ
    S13:
    SAL SI,1
    MOV DH,4
;----- SAVE AREA HAS CHARACTER IN IT, MATCH IT
    S15:
    MOV DI,OFFSET CRT_CHAR_GEN
    PUSH CS
    POP ES
    SUB BP,8
    MOV SI,BP
    CLD
    MOV AL,0
;----- CHAR NOT MATCHED, MIGHT BE IN USER SUPPLIED SECOND HALF
    S16:
    PUSH SS
    POP DS
    MOV DX,128
;----- CHARACTER IS FOUND ( AL=0 IF NOT FOUND )
    S18:
    ADD SP,8
    JMP VIDEO_RETURN
;----- GRAPHICS_READ ENDP
;----- EXPAND_BYTE
;----- THIS ROUTINE TAKES THE BYTE IN AL AND DOUBLES ALL
;----- OF THE BITS, TURNING THE 8 BITS INTO 16 BITS.
;----- THE RESULT IS LEFT IN AX
;----- PROC NEAR
    S21:
    PUSH CX
    MOV CX,8
;----- EXPAND_BYTE
;----- THIS ROUTINE WILL TAKE 2 BYTES FROM THE REGEN BUFFER,
;----- COMPARE AGAINST THE CURRENT FOREGROUND COLOR, AND PLACE
;----- THE CORRESPONDING ON/OFF BIT PATTERN INTO THE CURRENT
;----- POSITION IN THE SAVE AREA
    ENTRY S21
;----- MED_READ_BYTE
;----- THIS ROUTINE WILL TAKE 2 BYTES FROM THE REGEN BUFFER,
;----- SHIFT COUNT REGISTER FOR ONE BYTE
;----- SHIFT COUNT REGISTER FOR ONE BYTE
;----- SHIFT BITS, LOW BIT INTO CARRY FLAG
;----- MOVE CARRY FLAG (LOW BIT) INTO RESULTS
;----- SIGN EXTEND HIGH BIT (DOUBLE IT)
;----- REPEAT FOR ALL 8 BITS
;----- PROC NEAR
    S22:
    ROR AL,1
    RCR BP,1
    SAR BP,1
    LOOP S22
;----- XCHG AX,BP
    POP CX
    RET
    BNDP
;----- MED_READ_BYTE
;----- THIS ROUTINE WILL TAKE 2 BYTES FROM THE REGEN BUFFER,
;----- COMPARE AGAINST THE CURRENT FOREGROUND COLOR, AND PLACE
;----- THE CORRESPONDING ON/OFF BIT PATTERN INTO THE CURRENT
;----- POSITION IN THE SAVE AREA
    ENTRY S21

```

```
1637 ; S1,DS = POINTER TO REGEN AREA OF INTEREST
1638 ; BX = EXPANDED FOREGROUND COLOR
1639 ; BP = POINTER TO SAVE AREA
1640 ; EXIT --
1641 ; S1 AND BP ARE INCREMENTED
1642
1643 06BC PROC NEAR
1644 06BC AD LODSW ; GET FIRST BYTE AND SECOND BYTES
1645 06BD B6 C4 XCHG AL,AH ; SWAP FOR COMPARE
1646 06BF B9 C000 MOV CX,0C000H ; 2 BIT MASK TO TEST THE ENTRIES
1647 06C2 B2 00 MOV DL,0 ; RESULT REGISTER
1648 06C4
1649 06C5 85 C1 S24: TEST AX,CX ; IS THIS SECTION BACKGROUND?
1650 06C6 74 01 JZ S25 ; IF ZERO, IT IS BACKGROUND (CARRY=0)
1651 06C8 F9 STC ; WASN'T, SO SET CARRY
1652 06C9 S25:
1653 06C9 D0 D2 RCL DL,1 ; MOVE THAT BIT INTO THE RESULT
1654 06CB C1 E9 02 SHR CX,2 ; MOVE THE MASK INTO THE RIGHT BY 2 BITS
1655 06C9 80 F0 JNC S26 ; DO IT AGAIN IF MASK DIDN'T FALL OUT
1656 06D0 88 56 00 MOV [BP],DL ; STORE RESULT IN SAVE AREA
1657 06D3 45 INC BP ; ADJUST POINTER
1658 06D4 C3 RET ; ALL DONE
1659 06D5 S23 ENDP
1660
1661 ;----- V4 POSITION
1662 ; THIS ROUTINE TAKES THE CURSOR POSITION CONTAINED IN
1663 ; THE MEMORY LOCATION, AND CONVERTS IT INTO AN OFFSET
1664 ; INTO THE REGEN BUFFER, ASSUMING ONE BYTE/CHAR.
1665 ; FOR MEDIUM RESOLUTION GRAPHICS, THE NUMBER MUST
1666 ; BE DOUBLED.
1667 ; ENTRY -- NO REGISTERS, MEMORY LOCATION @CURSOR_POSN IS USED
1668 ; EXIT --
1669 ; AX CONTAINS OFFSET INTO REGEN BUFFER
1670
1671 06D5 PROC NEAR
1672 06D5 A1 0050 R MOV AX, @CURSOR_POSN ; GET CURRENT CURSOR
1673 06D6 0000 00 PUSH AX ; SAVE REGISTER
1674 06D8 53 MOV BX,AX ; SAVE A COPY OF CURRENT CURSOR
1675 06D9 8B D8 MOV AL,AH ; GET ROWS TO AL
1676 06D8 BA C4 MUL WORD PTR @CRT_COLS ; MULTIPLY BY BYTES/COLUMN
1677 06D0 F6 26 004A R SHL AX,2 ; MULTIPLY BY 4 SINCE 4 ROWS/BYTE
1678 06E1 C1 E0 02 SUB BH,BH ; IS AL THE COLUMN VALUE
1679 06E2 80 20 FF ADD AX,BX ; DETERMINE OFFSET
1680 06E5 E3 C3 POP BX ; RECOVER POINTER
1681 06E8 5B RET ; ALL DONE
1682 06E9 C3
1683 06EA S26 ENDP
1684 ;----- WRITE_TTY
1685
1686 ; THIS INTERFACE PROVIDES A TELETYPE LIKE INTERFACE TO THE
1687 ; VIDEO CARDS. THE INPUT CHARACTER IS WRITTEN TO THE CURRENT
1688 ; CURSOR POSITION, AND THE CURSOR IS MOVED TO THE NEXT POSITION.
1689 ; IF THE CURSOR LEAVES THE LAST COLUMN OF THE FIELD, THE COLUMN
1690 ; IS SET TO ZERO, AND THE ROW VALUE IS INCREMENTED. IF THE ROW
1691 ; ROW VALUE LEAVES THE LAST ROW, THE CURSOR IS PLACED ON THE LAST ROW,
1692 ; FIRST COLUMN, AND THE ENTIRE SCREEN IS SCROLLED UP ONE LINE.
1693 ; WHEN THE SCREEN IS SCROLLED UP, THE ATTRIBUTE FOR FILLING THE
1694 ; NEWLY BLANKED LINE IS READ FROM THE CURSOR POSITION ON THE PREVIOUS
1695 ; LINE BEFORE THE SCROLL, IN CHARACTER MODE. IN GRAPHICS MODE,
1696 ; THE 0 COLOR IS USED.
1697 ; ENTRY -- NO REGISTERS, MEMORY LOCATION @CURSOR_POSN IS USED
1698 ; (AH) = CURRENT CRT MODE
1699 ; (AL) = CHARACTER TO BE WRITTEN
1700 ; NOTE THAT BACK SPACE, CARRIAGE RETURN, BELL AND LINE FEED ARE
1701 ; HANDLED AS COMMANDS RATHER THAN AS DISPLAY GRAPHICS CHARACTERS
1702 ; (BL) = FOREGROUND COLOR FOR CHAR WRITE IF CURRENTLY IN A GRAPHICS MODE
1703 ; EXIT -- NO REGISTERS, MEMORY LOCATION @CURSOR_POSN IS USED
1704 ; ALL REGISTERS SAVED
1705
1706 ASSUME DS:DATA
1707 06EA WRITE_TTY PROC NEAR
1708 06EA 50 PUSH AX ; SAVE REGISTERS
1709 06E9 00 PUSH AX ; SAVE CHARACTER TO WRITE
1710 06EC B4 03 MOV AH,03H ; GET CURRENT PAGE SETTING
1711 06EE 8A 3E 0062 R MOV BH,ACTIVE_PAGE ; READ THE CURRENT CURSOR POSITION
1712 06F2 CD 10 INT 10H ; RECOVER CHARACTER
1713 06F4 58 POP AX
1714
1715 ;----- DX NOW HAS THE CURRENT CURSOR POSITION
1716
1717 06F5 3C 0D CMP AL,CR ; IS IT CARRIAGE RETURN OR CONTROL
1718 06F7 76 46 JBE U8 ; GO TO CONTROL CHECKS IF IT IS
1719
1720 ;----- WRITE THE CHAR TO THE SCREEN
1721 06F9 U01 MOV AH,0AH ; WRITE CHARACTER ONLY COMMAND
1722 06F9 B4 0A MOV CX,1 ; ONLY ONE CHARACTER
1723 06FB B9 0001 INT 10H ; WRITE THE CHARACTER
1724 06F6 CD 10
1725
1726 ;----- POSITION THE CURSOR FOR NEXT CHAR
1727
1728 0700 FE C2 INC DL ; TEST FOR COLUMN OVERFLOW
1729 0702 3A 16 004A R CMP DL,WORD PTR @CRT_COLS ; SET_CURSOR
1730 0706 75 33 JNZ U7 ; COLUMN FOR CURSOR
1731 0708 B2 00 MOV DL,0 ; CHECK FOR LAST ROW
1732 070A 80 FE 1B CMP DH,25-1 ; SET_CURSOR_INC
1733 070D 75 2A JNZ U6
1734
1735 ;----- SCROLL REQUIRED
1736 070F U11: MOV AH,02H ; SET THE CURSOR
1737 070F B4 02 INT 10H
1738 0711 CD 10
1739
1740 ;----- DETERMINE VALUE TO FILL WITH DURING SCROLL
1741
1742 0713 A0 0049 R MOV AL, @CRT_MODE ; GET THE CURRENT MODE
1743 0716 3C 04 CMP AL,4 ; READ-CURSOR
1744 0718 72 06 JC U2 ; SCROLL-UP
1745 071C 00 07 CMP AL,7 ; FILL WITH BACKGROUND
1746 071C B7 00 MOV BH,0 ; READ-CURSOR
1747 071E 75 06 JNE U3 ; GET READ CURSOR COMMAND
1748 0720 U2: MOV AH,0BH ; READ CHAR/ATTR AT CURRENT CURSOR
1749 0720 B4 08 INT 10H ; READ CHAR/ATTR AT CURRENT CURSOR
1750 0722 CD 10
```

```

1751 0724 8A FC          MOV    BH,AH          ; STORE IN BH
1752 0726                   U3:   MOV    AX,0601H      ; SCROLL-UP
1753 0726 BB 0601          MOV    CX,CX          ; SCROLL ONE LINE
1754 0729 2B C9          SUB    DH,25-1      ; UPPER LEFT CORNER
1755 072A 00 18          MOV    DL,BYTE PTR .CRT_COLS ; LOWER RIGHT ROW
1756 072D 8A 16 004A R      MOV    DL,0           ; LOWER RIGHT COLUMN
1757 0731 FE CA          DEC    DL             ; RETURN TO CALLER
1758 0733 U4:             INT    10H           ; VIDEO-CALL-RETURN
1759 0733 CD 10          U5:   POP    AX             ; SCROLL UP THE SCREEN
1760 0735                   JMP    VIDEO_RETURN ; TTY-RETURN
1761 0735 58          U6:   INC    DH             ; RESTORE THE CHARACTER
1762 0736 E9 012E R          JMP    U4             ; RETURN TO CALLER
1763
1764 0739                   U7:   INC    DH             ; SET-CURSOR-INC
1765 0739 FE C6          INC    AH,02H       ; NEXT ROW
1766 073B                   U8:   MOV    AH,02H       ; SET-CURSOR
1767 073B B4 02          JMP    U4             ; ESTABLISH THE NEW CURSOR
1768
1769 073F                   U9:   ----- CHECK FOR CONTROL CHARACTERS
1770 073F T4 13          JE    U9             ; WAS IT A CARRIAGE RETURN
1771 073F 30 0A          CMP   AL,LF          ; IF IT IS, LINE FEED
1772 0743 T4 13          JE    U10            ; GO TO LINE FEED
1773 0743 30 0A          CMP   AL,07H         ; IS IT A BELL
1774 0745 3C 07          JE    U11            ; GO TO BELL
1775 0747 T4 16          CMP   AL,08H         ; IS IT A BACKSPACE
1776 0749 3C 08          JNE   U0             ; IF NOT A CONTROL, DISPLAY IT
1777 074B T5 AC          U10:  ----- BACK SPACE FOUND
1778
1779 074F T4 13          OR    DL,DL          ; IS IT ALREADY AT START OF LINE
1780 074F 30 0A          JE    U7             ; SET_CURSOR
1781 074F EA EA          DEC    DX             ; NO -- JUST MOVE IT BACK
1782 0752 EB E7          JMP    U7             ; SET_CURSOR
1783
1784 0754 B2 00          U11:  ----- CARRIAGE RETURN FOUND
1785 0754 BB 00          MOV    DL,0           ; MOVE TO FIRST COLUMN
1786 0756 EB E3          JMP    U7             ; SET_CURSOR
1787
1788 0755                   U12:  ----- LINE FEED FOUND
1789 0755 80 FE 18          CMP   DH,25-1      ; BOTTOM OF SCREEN
1790 0755 75 DC          JNE   U6             ; YES, SCROLL THE SCREEN
1791 075D EB B0          JMP    U1            ; NO, JUST SET THE CURSOR
1792
1793
1794 0758                   U13:  ----- BELL FOUND
1795 0758 80 FE 18          CMP   DH,25-1      ; DIVISOR FOR 896 Hz TONE
1796 0758 75 DC          JNE   U6             ; SET COUNT FOR 31/64 SECOND FOR BEEP
1797 075D EB B0          CALL   BEEP          ; SOUND THE BEEP
1798 0760                   U14:  ----- BELL FOUND
1799 0760 EB CC          JMP    U5             ; TTY_RETURN
1800
1801 075F                   U15:  ----- ASSUME DS:DATA
1802 075F B9 0533          MOV    CX,1331      ; 3,3,5,5,3,3,4,           ; SUBTRACT_TABLE
1803 0762 B3 1F          MOV    BL,31          ; BY CARRY ARE DESTROYED
1804 0762 E8 0000 E          CALL   BEEP          ; SET COUNT FOR 31/64 SECOND FOR BEEP
1805 0764 E8 0000 E          JMP    U5             ; SOUND THE BEEP
1806 0767 EB CC          WRITE_TTY        ; TTY_RETURN
1807 0769                   U16:  ----- LIGHT PEN
1808
1809 0771                   U17:  ----- THIS ROUTINE TESTS THE LIGHT PEN SWITCH AND THE LIGHT
1810 0771 8B 00             PEN TRIGGER. IF BOTH ARE SET, THE LOCATION OF THE LIGHT
1811 0773 BB 16 0063 R      PEN IS DETERMINED. OTHERWISE, A RETURN WITH NO INFORMATION
1812 0777 B3 C2 06          IS MADE.
1813
1814 0777 EC             U18:  ----- ON EXIT:
1815 0777 8B 00             (AH) = 0 IF NO LIGHT PEN INFORMATION IS AVAILABLE
1816 0777 BB 16 0063 R      BY CARRY ARE DESTROYED
1817 0777 B3 C2 06             (AH) = 1 IF LIGHT PEN IS AVAILABLE
1818 0777 EC             (DH,DL) = ROW,COLUMN OF CURRENT LIGHT PEN POSITION
1819 0777 8B 00             (CH) = RASTER POSITION
1820 0777 BB 16 0063 R      (BX) = BEST GUESS AT PIXEL HORIZONTAL POSITION
1821
1822
1823 0769 03 03 05 05 03 03  V1   ASSUME DS:DATA
1824 0769 03 04          VI    DB 3,3,5,5,3,3,4,           ; SUBTRACT_TABLE
1825
1826 0771                   U19:  ----- WAIT FOR LIGHT PEN TO BE DEPRESSED
1827 0771 EC             READ_LPEN        PROC   NEAR
1828 0771 BB 00          MOV    AH,0           ; SET NO LIGHT PEN RETURN CODE
1829 0773 BB 16 0063 R      MOV    DX,ADDR_6845      ; GET BASE ADDRESS OF 6845
1830 0777 B3 C2 06          ADD    DX,6           ; POINT TO STATUS REGISTER
1831 077A EC             IN    AL,DX          ; GET STATUS REGISTER
1832 077B T4 04          TEST   AL,004H      ; TEST LIGHT PEN SWITCH
1833 077D T4 03          JZ    V6_A          ; GO IF YES
1834 077F E9 0603 R      JMP    V6             ; NOT SET, RETURN
1835
1836
1837 0782 A8 02          U20:  ----- NOW TEST FOR LIGHT PEN TRIGGER
1838 0784 75 03          V6_A:  TEST   AL,2           ; TEST LIGHT PEN TRIGGER
1839 0784 BB 16 0063 R      JMP    VT             ; RETURN WITHOUT RESETTING TRIGGER
1840 0786 E9 060D R
1841
1842 0789                   U21:  ----- TRIGGER HAS BEEN SET, READ THE VALUE IN
1843 0789 BB 16 0063 R      V7A:  MOV    AH,16           ; LIGHT PEN REGISTERS ON 6845
1844 0789
1845 0789 B4 10          MOV    AH,16           ; LIGHT PEN REGISTERS ON 6845
1846
1847 0789                   U22:  ----- INPUT REGISTERS POINTED TO BY AH, AND CONVERT TO ROW COLUMN IN (DX)
1848 078B BB 16 0063 R      MOV    DX,ADDR_6845      ; ADDRESS REGISTER FOR 6845
1849 078F 8A C4          MOV    AL,AH          ; REGISTER TO READ
1850 0791 EE             OUT   DX,AL          ; SET TO DATA REGISTER
1851 0792 BB 16 0063 R      JMP    $+2           ; I/O DELAY
1852 0794 42          INC    DX             ; DATA REGISTER
1853 0794 42          IN    AL,DX          ; GET THE VALUE
1854 0795 EC             MOV    CH,AL          ; SAVE IN CX
1855 0796 8A E8          DEC    DX             ; ADDRESS REGISTER
1856 0796 8A E8          INC    AH             ; SECOND DATA REGISTER
1857 0796 8A E8          MOV    AL,AH          ; POINT TO DATA REGISTER
1858 0798 8A C4          OUT   DX,AL          ; I/O DELAY
1859 079D EE             INC    DX             ; GET SECOND DATA VALUE
1860 079E 42          JMP    $+2           ; AX HAS INPUT VALUE
1861 079F EB 00
1862 07A1 EC
1863 07A2 8A E5
1864

```

```

1865           ;----- AX HAS THE VALUE READ IN FROM THE 6845
1866
1867 07A4 8A 1E 0049 R   MOV    BL, *CRT_MODE
1868 07A8 2A FF           SUB    BH,BH
1869 07AA 2E: 8A 9F 0769 R  MOV    BL,CS1+1[BX]      ; MODE VALUE TO BX
1870 07AF 2B C3           SUB    AX,BX      ; DETERMINE AMOUNT TO SUBTRACT
1871 07B1 8B 1E 004E R   MOV    BX,*CRT_START
1872 07B5 D1 E8           SHR    BX,1       ; TAKE IT AWAY
1873 07B7 2B C3           SUB    AX,BX      ; CONVERT TO CORRECT PAGE ORIGIN
1874 07B9 19 02           JNS    V2          ; IF POSITIVE, DETERMINE MODE
1875 07BB 2B C0           SUB    AX,AX      ; <0 PLAYS AS 0
1876
1877           ;----- DETERMINE MODE OF OPERATION
1878
1879 07BD V2:             MOV    CL,3       ; DETERMINE_MODE
1880 07BD B1 03           CMP    *CRT_MODE,4
1881 07BF 80 3E 0049 R 04  JE    V4          ; SET * SHFT COUNT
1882 07C4 72 29           JB    V4          ; DETERMINE IF GRAPHICS OR ALPHA
1883 07C6 80 3E 0049 R 07  CMP    *CRT_MODE,7
1884 07CB T4 22           JE    V4          ; ALPHA_PEN
1885
1886           ;----- GRAPHICS MODE
1887
1888 07CD B2 28           MOV    DL,40      ; DIVISOR FOR GRAPHICS
1889 07CF F6 F2           DIV    DL          ; DETERMINE ROW(AL) AND COLUMN(AH)
1890
1891           ;----- DETERMINE GRAPHIC ROW POSITION
1892
1893 07D1 8A E8           MOV    CH,AL      ; SAVE ROW VALUE IN CH
1894 07D3 02 ED           ADD    CH,CH      ; *2 FOR EVEN/ODD FIELD
1895 07D5 8A DC           MOV    BL,AH      ; COLUMN VALUE TO BX
1896 07D7 2A FF           SUB    BH,BH      ; MULTIPLY BY 16 FOR MEDIUM RES
1897 07D9 80 3E 0049 R 06  CMP    *CRT_MODE,6
1898 07E0 75 04           JNE    V3          ; DETERMINE MEDIUM OR HIGH RES
1899 07E0 B1 04           MOV    CL,4       ; NOT HIGH RES
1900 07E2 D0 E4           SAL    AH,1       ; SHIFT VALUE FOR HIGH RES
1901 07E4 V3:             SHL    BX,CL      ; COLUMN VALUE TIMES 2 FOR HIGH RES
1902 07E4 D3 E3           SHL    BX,CL      ; NOT HIGH RES
1903
1904           ;----- DETERMINE ALPHA CHAR POSITION
1905
1906 07E6 8A D4           MOV    DL,AH      ; COLUMN VALUE FOR RETURN
1907 07E8 8A F0           MOV    DH,AL      ; ROW VALUE
1908 07EA C0 EE 02           SHR    DH,2       ; DIVIDE BY 4 FOR VALUE IN 0-24 RANGE
1909 07ED EB 12           JMP    SHORT V5  ; LIGHT_PEN_RETURN_SET
1910
1911           ;----- ALPHA MODE ON LIGHT PEN
1912
1913 07EF V4:             DIV    BYTE PTR *CRT_COLS
1914 07EF F6 30 004A R   MOV    DH,AL      ; DETERMINE ROW,COLUMN VALUE
1915 07F3 8A F0           MOV    DL,AH      ; ROWS TO DH
1916 07F5 80 D4           MOV    DL,AL      ; COLS TO DL
1917 07F7 02 00           SAL    AX,CL      ; MULTIPLY ROWS * 8
1918 07F9 8A E8           MOV    CH,AL      ; GET RASTER VALUE TO RETURN REGISTER
1919 07FB 8A DC           MOV    BL,AH      ; COLUMN VALUE
1920 07FD 32 FF           XOR    BH,BH      ; TO BX
1921 07FF D3 E3           SAL    BX,CL      ; LIGHT_PEN_RETURN_SET
1922 0800 00 00           V5:             MOV    AH,1       ; INDICATE EVERYTHING SET
1923 0801 B4 01           MOV    AH,1       ; LIGHT_PEN_RETURN
1924 0803 V6:             PUSH   DX          ; SAVE RETURN VALUE (IN CASE)
1925 0803 52           MOV    DX, @ADDR_6845
1926 0804 BB 16 0063 R   ADD    DX,7       ; GET BASE ADDRESS
1927 0808 B3 C2 07           OUT    DX,AL      ; POINT TO RESET PARM
1928 0808 EE           POP    DX          ; ADDRESS, NOT DATA, IS IMPORTANT
1929 0808 5A           POP    DX          ; RECOVER VALUE
1930 0800 V7:             POP    ES          ; RETURN_NO_RESET
1931 0800 5D           POP    BP          ; DISCARD SAVED BX,CX,DX
1932 0800 5F           POP    D1
1933 080F 5E           POP    SI
1934 0810 1F           POP    DS
1935 0811 1F           POP    DS
1936 0812 1F           POP    DS
1937 0813 1F           POP    DS
1938 0814 07           POP    ES
1939 0815 CF           IRET
1940 0816           READ_LPEN  ENDP
1941 0816           CODE   ENDS
1942           END

```

```
1 PAGE 118,121
2 TITLE BIOS ----- 06/10/85 BIOS ROUTINES
3 .286C
4 .LIST
5 0000 CODE SEGMENT BYTE PUBLIC
6
7 PUBLIC EQUIPMENT_
8 PUBLIC MEMORY_SIZE_DET_
9 PUBLIC NMI_INT_
10
11 EXTRN CB943:NEAR ; POST SEND 8042 COMMAND ROUTINE
12 EXTRN CMDS_READ:NEAR ; READ CMDS LOCATION ROUTINE
13 EXTRN D1:NEAR ; *PARITY CHECK 1* MESSAGE
14 EXTRN D2:NEAR ; *PARITY CHECK 2* MESSAGE
15 EXTRN D2A: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_1000:NEAR ; DISPLAY FIVE CHARACTER ADDRESS ROUTINE
19 EXTRN PRT_1001:NEAR ; DISPLAY FIVE CHARACTER ADDRESS ROUTINE
20 EXTRN P_MSG:NEAR ; DISPLAY MESSAGE STRING ROUTINE
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 0000
49 0000 FB
50 0001 1E
51 0002 E8 0000 E
52 0005 A1 0013 R
53 0008 1F
54 0009 CF
55 000A
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 000A
94 000A FB
95 000B 1E
96 000C 0000 E
97 000F A1 0010 R
98 0012 1F
99 0013 CF
100 0014
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
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
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
188
189
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
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
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
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
418
419
419
420
421
422
423
424
425
426
427
428
429
429
430
431
432
433
434
435
436
437
438
439
439
440
441
442
443
444
445
446
447
448
449
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
486
487
488
489
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
537
538
539
539
540
541
542
543
544
545
546
547
547
548
549
549
550
551
552
553
554
555
556
557
557
558
559
559
560
561
562
563
564
565
566
567
567
568
569
569
570
571
572
573
574
575
575
576
577
577
578
579
579
580
581
582
583
584
585
585
586
587
587
588
589
589
590
591
592
593
594
595
596
597
597
598
599
599
600
601
602
603
604
605
606
607
607
608
609
609
610
611
612
613
614
615
615
616
617
617
618
619
619
620
621
622
623
624
625
625
626
627
627
628
629
629
630
631
632
633
634
635
635
636
637
637
638
639
639
640
641
642
643
644
644
645
646
646
647
648
648
649
649
650
651
652
653
654
654
655
656
656
657
658
658
659
659
660
661
662
663
664
664
665
666
666
667
668
668
669
669
670
671
672
673
674
674
675
676
676
677
678
678
679
679
680
681
682
683
684
685
685
686
687
687
688
689
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
717
718
719
719
720
721
722
723
724
725
726
727
727
728
729
729
730
731
732
733
734
735
736
737
737
738
739
739
740
741
742
743
744
745
746
747
747
748
749
749
750
751
752
753
754
755
756
757
757
758
759
759
760
761
762
763
764
765
766
767
767
768
769
769
770
771
772
773
774
775
776
777
777
778
779
779
780
781
782
783
784
785
786
787
787
788
789
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
817
818
819
819
820
821
822
823
824
825
826
827
827
828
829
829
830
831
832
833
834
835
836
837
837
838
839
839
840
841
842
843
844
845
846
847
847
848
849
849
850
851
852
853
854
855
856
857
857
858
859
859
860
861
862
863
864
865
866
867
867
868
869
869
870
871
872
873
874
875
876
877
877
878
879
879
880
881
882
883
884
885
886
887
887
888
889
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
917
918
919
919
920
921
922
923
924
925
926
927
927
928
929
929
930
931
932
933
934
935
936
937
937
938
939
939
940
941
942
943
944
945
946
947
947
948
949
949
950
951
952
953
954
955
956
957
957
958
959
959
960
961
962
963
964
965
966
966
967
968
968
969
970
971
972
973
974
975
976
977
977
978
979
979
980
981
982
983
984
985
986
987
987
988
989
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1047
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1057
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1077
1078
1079
1079
1080
1081
1082
1083
1084
1085
1086
1087
1087
1088
1089
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1116
1117
1118
1118
1119
1120
1121
1122
1123
1124
1125
1126
1126
1127
1128
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1137
1138
1139
1139
1140
1141
1142
1143
1144
1145
1146
1146
1147
1148
1148
1149
1150
1151
1152
1153
1154
1155
1156
1156
1157
1158
1158
1159
1160
1161
1162
1163
1164
1165
1166
1166
1167
1168
1168
1169
1170
1171
1172
1173
1174
1175
1176
1176
1177
1178
1178
1179
1180
1181
1182
1183
1184
1185
1186
1186
1187
1188
1188
1189
1190
1191
1192
1193
1194
1195
1195
1196
1197
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1206
1207
1208
1208
1209
1210
1211
1212
1213
1214
1215
1215
1216
1217
1217
1218
1219
1219
1220
1221
1222
1223
1224
1225
1226
1226
1227
1228
1228
1229
1230
1231
1232
1233
1234
1235
1236
1236
1237
1238
1238
1239
1240
1241
1242
1243
1244
1245
1245
1246
1247
1247
1248
1249
1249
1250
1251
1252
1253
1254
1255
1256
1256
1257
1258
1258
1259
1260
1261
1262
1263
1264
1265
1265
1266
1267
1267
1268
1269
1269
1270
1271
1272
1273
1274
1275
1276
1276
1277
1278
1278
1279
1280
1281
1282
1283
1284
1285
1285
1286
1287
1287
1288
1289
1289
1290
1291
1292
1293
1294
1295
1295
1296
1297
1297
1298
1299
1300
1301
1302
1303
1304
1305
1305
1306
1307
1307
1308
1309
1310
1311
1312
1313
1314
1315
1315
1316
1317
1317
1318
1319
1319
1320
1321
1322
1323
1324
1325
1326
1326
1327
1328
1328
1329
1330
1331
1332
1333
1334
1335
1335
1336
1337
1337
1338
1339
1339
1340
1341
1342
1343
1344
1345
1345
1346
1347
1347
1348
1349
1349
1350
1351
1352
1353
1354
1355
1356
1356
1357
1358
1358
1359
1360
1361
1362
1363
1364
1365
1365
1366
1367
1367
1368
1369
1369
1370
1371
1372
1373
1374
1375
1376
1376
1377
1378
1378
1379
1380
1381
1382
1383
1384
1385
1385
1386
1387
1387
1388
1389
1389
1390
1391
1392
1393
1394
1395
1395
1396
1397
1397
1398
1399
1399
1400
1401
1402
1403
1404
1405
1405
1406
1407
1407
1408
1409
1409
1410
1411
1412
1413
1414
1415
1415
1416
1417
1417
1418
1419
1419
1420
1421
1422
1423
1424
1425
1425
1426
1427
1427
1428
1429
1429
1430
1431
1432
1433
1434
1435
1435
1436
1437
1437
1438
1439
1439
1440
1441
1442
1443
1444
1445
1445
1446
1447
1447
1448
1449
1449
1450
1451
1452
1453
1454
1455
1455
1456
1457
1457
1458
1459
1459
1460
1461
1462
1463
1464
1465
1465
1466
1467
1467
1468
1469
1469
1470
1471
1472
1473
1474
1475
1475
1476
1477
1477
1478
1479
1479
1480
1481
1482
1483
1484
1485
1485
1486
1487
1487
1488
1489
1489
1490
1491
1492
1493
1494
1495
1495
1496
1497
1497
1498
1499
1499
1500
1501
1502
1503
1504
1505
1505
1506
1507
1507
1508
1509
1509
1510
1511
1512
1513
1514
1515
1515
1516
1517
1517
1518
1519
1519
1520
1521
1522
1523
1524
1525
1525
1526
1527
1527
1528
1529
1529
1530
1531
1532
1533
1534
1535
1535
1536
1537
1537
1538
1539
1539
1540
1541
1542
1543
1544
1545
1545
1546
1547
1547
1548
1549
1549
1550
1551
1552
1553
1554
1555
1555
1556
1557
1557
1558
1559
1559
1560
1561
1562
1563
1564
1565
1565
1566
1567
1567
1568
1569
1569
1570
1571
1572
1573
1574
1575
1575
1576
1577
1577
1578
1579
1579
1580
1581
1582
1583
1584
1585
1585
1586
1587
1587
1588
1589
1589
1590
1591
1592
1593
1594
1595
1595
1596
1597
1597
1598
1599
1599
1600
1601
1602
1603
1604
1605
1605
1606
1607
1607
1608
1609
1609
1610
1611
1612
1613
1614
1615
1615
1616
1617
1617
1618
1619
1619
1620
1621
1622
1623
1624
1625
1625
1626
1627
1627
1628
1629
1629
1630
1631
1632
1633
1634
1635
1635
1636
1637
1637
1638
1639
1639
1640
1641
1642
1643
1644
1645
1645
1646
1647
1647
1648
1649
1649
1650
1651
1652
1653
1654
1655
1655
1656
1657
1657
1658
1659
1659
1660
1661
1662
1663
1664
1665
1665
1666
1667
1667
1668
1669
1669
1670
1671
1672
1673
1674
1675
1675
1676
1677
1677
1678
1679
1679
1680
1681
1682
1683
1684
1685
1685
1686
1687
1687
1688
1689
1689
1690
1691
1692
1693
1694
1695
1695
1696
1697
1697
1698
1699
1699
1700
1701
1702
1703
1704
1705
1705
1706
1707
1707
1708
1709
1709
1710
1711
1712
1713
1714
1715
1715
1716
1717
1717
1718
1719
1719
1720
1721
1722
1723
1724
1725
1725
1726
1727
1727
1728
1729
1729
1730
1731
1732
1733
1734
1735
1735
1736
1737
1737
1738
1739
1739
1740
1741
1742
1743
1744
1745
1745
1746
1747
1747
1748
1749
1749
1750
1751
1752
1753
1754
1755
1755
1756
1757
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1765
1766
1767
1767
1768
1769
1769
1770
1771
1772
1773
1774
1775
1775
1776
1777
1777
1778
1779
1779
1780
1781
1782
1783
1784
1785
1785
1786
1787
1787
1788
1789
1789
1790
1791
1792
1793
1794
1795
1795
1796
1797
1797
1798
1799
1799
1800
1801
1802
1803
1804
1805
1805
1806
1807
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1815
1816
1817
1817
1818
1819
1819
1820
1821
1822
1823
1824
1825
1825
1826
1827
1827
1828
1829
1829
1830
1831
1832
1833
1834
1835
1835
1836
1837
1837
1838
1839
1839
1840
1841
1842
1843
1844
1845
1845
1846
1847
1847
1848
1849
1849
1850
1851
1852
1853
1854
1855
1855
1856
1857
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1865
1866
1867
1867
1868
1869
1869
1870
1871
1872
1873
1874
1875
1875
1876
1877
1877
1878
1879
1879
1880
1881
1882
1883
1884
1885
1885
1886
1887
1887
1888
1889
1889
1890
1891
1892
1893
1894
1894
1895
1896
1896
1897
1898
1898
1899
1900
1901
1902
1903
1904
1904
1905
1906
1906
1907
1908
1908
1909
1910
1911
1912
1913
1914
1915
1915
1916
1917
1917
1918
1919
1919
1920
1921
1922
1923
1924
1925
1925
1926
1927
1927
1928
1929
1929
1930
1931
1932
1933
1934
1935
1935
1936
1937
1937
1938
1939
1939
1940
1941
1942
1943
1944
1945
1945
1946
1947
1947
1948
1949
1949
1950
1951
1952
1953
1954
1955
1955
1956
1957
1957
1958
1959
1959
1960
1961
1962
1963
1964
1965
1965
1966
1967
1967
1968
1969
1969
1970
1971
1972
1973
1974
1975
1975
1976
1977
1977
1978
1979
1979
1980
1981
1982
1983
1984
1985
1985
1986
1987
1987
1988
1989
1989
1990
1991
1992
1993
1994
1994
1995
1996
1996
1997
1998
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2029
2030
2031
2032
2033
2034
2035
2035
2036
2037
2037
2038
2039
2039
2040
2041
2042
2043
2044
2045
2045
2046
2047
2047
2048
2049
2049
2050
2051
2052
2053
2054
2055
2055
2056
2057
2057
2058
2059
2059
2060
2061
2062
2063
2064
2065
2065
2066
2067
2067
2068
2069
2069
2070
2071
2072
2073
2074
2075
2075
2076
2077
2077
2078
2079
2079
2080
2081
2082
2083
2084
2085
2085
2086
2087
2087
2088
2089
2089
2090
2091
2092
2093
2094
2094
2095
2096
2096
2097
2098
2098
2099
2100
2101
2102
2103
2104
2105
2105
2106
2107
2107
2108
2109
2109
2110
2111
2112
2113
2114
2115
2115
2116
2117
2117
2118
2119
2119
2120
2121
2122
2123
2124
2125
2125
2126
2127
2127
2128
2129
2129
2130
2131
2132
2133
2134
2135
2135
2136
2137
2137
2138
2139
2139
2140
2141
2142
2143
2144
2145
2145
2146
2147
2147
2148
2149
2149
2150
2151
2152
2153
2154
2155
2155
2156
2157
2157
2158
2159
2159
2160
2161
2162
2163
2164
2165
2165
2166
2167
2167
2168
2169
2169
2170
2171
2172
2173
2174
2175
2175
2176
2177
2177
2178
2179
2179
2180
2181
2182
2183
2184
2185
2185
2186
2187
2187
2188
2189
2189
2190
2191
2192
2193

```

```

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 BAD 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
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 B0 0D MOV AL,CMOS_REG_D ; 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: PUSH AX ; HARDWARE ERROR
128 0022 50 ; SAVE INITIAL CHECK MASK IN (AL)
129 0023 B0 8D MOV AL,CMOS_REG_D+NMI ; MASK TRAP (NMI) INTERRUPTS OFF
130 0024 E5 00 OUT AL,DT5_KBD
131 0027 00 AD CALL CB042 ; DISABLE THE KEYBOARD
132 0029 E8 0000 E SEND COMMAND TO ADAPTER
133 002C E8 0000 E CALL DDS ; ADDRESS DATA SEGMENT
134 002F B4 00 MOV AH,0 ; INITIALIZE AND SET MODE FOR VIDEO
135 0031 A0 0049 R MOV AL,FCRT_MODE ; GET CURRENT MODE
136 0034 CD 10 INT 10H ; CALL VIDEO_ID TO CLEAR SCREEN
137
138 ;----- DISPLAY "PARITY CHECK ?" ERROR MESSAGES
139
140 0036 58 POP AX ; RECOVER INITIAL CHECK STATUS
141 0037 BE 0000 E MOV SI,OFFSET D1 ; PLANAR ERROR, ADDRESS "PARITY CHECK 1"
142 003A A8 80 TEST AL,PARITY_CHECK ; CHECK FOR PLANAR ERROR
143 003C 74 05 JZ NMI_2 ; SKIP IF NOT
144
145 003E 50 PUSH AX ; SAVE STATUS
146 003F E8 0000 E CALL P_MSG ; DISPLAY "PARITY CHECK 1" MESSAGE
147 0042 58 POP AX ; AND RECOVER STATUS
148
149 0043 BE 0000 E NMI_2: MOV SI,OFFSET D2 ; ADDRESS OF "PARITY CHECK 2" MESSAGE
150 0046 A8 40 TEST AL,IO_CHECK ; I/O PARITY CHECK ?
151 0048 74 03 JZ NMI_3 ; SKIP IF CORRECT ERROR DISPLAYED
152 004A E8 0000 E CALL P_MSG ; DISPLAY "PARITY CHECK 2" ERROR
153
154 ;----- TEST FOR HOT NMI ON PLANAR PARITY LINE
155
156 004D NMI_3: IN AL,PORT_B ; SET DIRECTION FLAG TO INCREMENT
157 004D E4 61 OR AL,RAM_PAR_OFF ; POINT (DX) AT START OF REAL MEMORY
158 004F 0C 0C OUT PORT_B,AL ; SET (SI) TO START OF (DS:)
159 0051 E6 61 AND AL,RAM_PAR_ON ; READ CURRENT PARITY CHECK LATCH
160 0053 24 F3 TEST AL,PARITY_ERR ; CHECK FOR HOT NMI SOURCE
161 0055 E6 61 JNZ NMI_5 ; SKIP IF ERROR NOT RESET (DISPLAY ???)
162
163 0057 FC CLD ; GO PRINT SEGMENT ADDRESS IF ERROR
164 0058 2B D2 SUB DX,DX
165 005A 2B F6 SUB SI,SI
166 005C E7 61 IN AL,PORT_B ; SET WORD COUNT FOR 64 KB SCAN
167 005E A8 C0 TEST AL,PARITY_ERR ; READ 64 KB OF MEMORY
168 0060 75 19 JNZ NMI_5 ; READ PARITY CHECK LATCHES
169
170 ;----- SEE IF LOCATION THAT CAUSED PARITY CHECK CAN BE FOUND IN BASE MEMORY
171
172 0062 8B 1E 0013 R MOV BX,MEMORY_SIZE ; GET BASE MEMORY SIZE WORD
173 0064
174 0066 8E DA MOV DS,DX ; POINT TO 64K SEGMENT
175 0068 B9 8000 MOV CX,4000H*2 ; SET WORD COUNT FOR 64 KB SCAN
176 006B F3 /AD REP LODSW ; READ 64 KB OF MEMORY
177 006D E4 61 IN AL,PORT_B ; READ PARITY CHECK LATCHES
178 006F A8 C0 TEST AL,PARITY_ERR ; CHECK FOR ANY PARITY ERROR PENDING
179 0071 75 10 JNZ NMI_6 ; GO PRINT SEGMENT ADDRESS IF ERROR
180
181 0073 80 C0 10 ADD DH,010H ; POINT TO NEXT 64K BLOCK
182 0076 83 EB 40 SUB BX,16D*4 ; DECREMENT COUNT OF 1024 BYTE SEGMENTS
183 0079 77 EB JA NMI_4 ; LOOP TILL ALL 64K SEGMENTS DONE
184
185 007B BE 0000 E NMI_5: MOV SI,OFFSET D2A ; PRINT ROW OF ????? IF PARITY
186 007E E8 0000 E CALL P_MSG ; CHECK COULD NOT BE RE-CREATED
187 0081 FA CLI ; HALT SYSTEM
188 0082 F4 HLT
189
190 0083
191 0083 E8 0000 E NMI_6: CALL PRT_SEG ; PRINT SEGMENT VALUE (IN DX)
192 0086 B0 28 MOV AL,'1' ; PRINT (S)
193 0088 E8 0000 E CALL PRT_HEX
194 008B B5 53 MOV AL,'5'
195 008D E8 0000 E CALL PRT_HEX
196 0090 B0 29 MOV AL,'1'
197 0092 E8 0000 E CALL PRT_HEX
198 0095 FA CLI ; HALT SYSTEM
199 0096 F4 HLT
200
201 0097 NMI_INT_1 ENDP
202
203 0097 CODE ENDS
204 END

```

```

1 PAGE 118,121
2 TITLE BIOS1 ---- 06/10/85 INTERRUPT 15H BIOS ROUTINES
3 .286C
4 .LIST
5 0000  SEGMENT BYTE PUBLIC
6
7 PUBLIC  CASSETTE_10_1
8 PUBLIC  GATE_A20
9 PUBLIC  SHUT5
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          ::::: RETURN AL = SWITCH SETTINGS (BITS 7-4)
62          ::::: (DX) = 01H - READ THE RESISTIVE INPUTS
63          ::::: RETURN BX = A(i)y VALUE
64          ::::: BX = B(i)y VALUE
65          ::::: CX = B(i)x VALUE
66          ::::: DX = B(i)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 SERIALIZED ACCESS
92          ::::: 80H -> BFH
93          ::::: REENTRANT DEVICES; ES:BX IS USED TO
94          ::::: DISTINGUISH DIFFERENT CALLS (MULTIPLE I/O
95          ::::: CALLS ARE ALLOWED SIMULTANEOUSLY)
96          ::::: C0H -> 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      ::::: TYPE DESCRIPTION TIMEOUT
103          ::::: 00H = DISK YES
104          ::::: 01H = DISKETTE YES
105          ::::: 02H = KEYBOARD NO
106          ::::: 80H = NETWORK NO
107          ::::: ES:BX -> NCB
108          ::::: FDH = DISKETTE MOTOR START YES
109          ::::: FEH = PRINTER YES
110
111

```

```

PAGE
F      (AH) = COH  RETURN CONFIGURATION PARAMETERS POINTER
:
:           RETURNS
:           (AH) = 00H AND CY= 0 (IF PRESENT ELSE 86 AND CY= 1)
:           (ES:BX) = PARAMETER TABLE ADDRESS POINTER
:           WHERE;
:
:           DW 8      LENGTH OF FOLLOWING TABLE
:           DB MODEL_BYTE SYSTEM MODEL BYTE
:           DB TYPE_BYTE SYSTEM TYPE BYTE
:           DB BIOS_LEVEL BIOS REVISION LEVEL
:           DB ?
:           00000000 = DMA CHANNEL 3 USE BY BIOS
:           01000000 = CASCADED INTERRUPT LEVEL 2
:           00100000 = REAL TIME CLOCK AVAILABLE
:           00010000 = KEYBOARD SCAN CODE HOOK 1AH
:
:           DB 0      RESERVED
:           DB 0      RESERVED
:           DB 0      RESERVED
:
ASSUME CS:CODE
CASSETTE_IO_1 PROC FAR
ST1_      ; ENABLE INTERRUPTS
CMP AH,080H
JB 0FH
CMP AH,000H
JE CONF_PARMS
SUB AH,080H
OR AH,AH
JZ DEV_OPEN
DEC AH
JZ DEV_CLOSE
DEC AH
JZ PROG_TERM
DEC AH
JZ EVENT_WAIT
DEC AH
JNZ NOT_JOYSTICK
JMP JOY_STICK
NOT_JOYSTICK:
DEC AH
JZ SYS_REQ
DEC AH
JZ C1_A
DEC AH
JNZ NOT_BLOCKMOVE
JMP BLOCKMOVE
C1_A:  JMP WAIT
C1_B:  DEC AH
C1_C:  JNZ C1_D
JMP SET_VMODE
C1_D:  SUB AH,7
JNZ C1_E
JMP DEVICE_BUSY
C1_E:  DEC AH
JNZ C1_F
JMP INT_COMPLETE
C1_F:  MOV AH,86H
STC
RET 2
FAR RETURN EXIT FROM ROUTINES
:
005A:  DEY_OPEN:          ; NULL HANDLERS
005A:  DEV_CLOSE:
005A:  PROG_TERM:
005A:  SYS_REQ:
JMP C1_F
CASSETTE_IO_1 ENDP
:
CONF_PARMS PROC NEAR
PUSH CS
POP ES
MOV BX,[OFFSET CONF_TBL]
XOR AH, AH
JMP C1_F
CONF_PARMS ENDP
:
0065 EVENT_WAIT PROC NEAR
ASSUME DS:DATA
PUSH DS
CALL DDS
OR AL,AL
JZ EVENT_WAIT_2
DEC AH
JZ EVENT_WAIT_3
POP DS
STC
JMP C1_F
:
EVENT_WAIT_2:
ASSUME DS:DATA
CLT
TEST ORTC_WAIT_FLAG,01
JZ EVENT_WAIT_1
:
EVENT_WAIT_1:
NO_INTERRUPTS_ALLOWED
:
0075 FA
TEST ORTC_WAIT_FLAG,01
JZ EVENT_WAIT_1
:
0076 F6 06 00A0 R 01
0077 T4 05
007D FB
ST1
007E 1F
007F F9
0080 EB D5
POP DS
STC
JMP C1_F
:
NO_INTERRUPTS_ALLOWED
:
CHECK FOR FUNCTION ACTIVE
:
ENABLE_INTERRUPTS
:
SET_ERROR
:
RETURN

```

```

226
227 0062          EVENT_WAIT_1:    AL,_INTB01          ; ENSURE INTERRUPT UNMASKED
228 0062 E4 A1
229 0084 EB 00
230 0086 24 FE
231 0088 E6 A1
232 008A 8C 06 009A R
233 008E 89 1E 0098 R
234 0092 89 0E 009E R
235 0094 89 0E 009C R
236 009A C6 06 00A0 R 01
237 009F 80 0B
238 00A1 EB 0000 E
239 00A4 24 7F
240 00A6 00 40
241 00A8 80
242 00A9 8A E0
243 00AB 80 0B
244 00AD EB 0000 E
245 00B0 58
246 00B1 1F
247 00B2 FB
248 00B3 F8
249 00B4 EB A1
250
251          ;----- CANCEL
252
253 00B6          EVENT_WAIT_3:    PUSH AX          ; SAVE
254 00B6 50          PUSH CX          ; DISABLE_INTERRUPTS
255 00B7 FA          MOV AX,X*CMOS_REG_B    ; TURN OFF PIE
256 00B8 0B0B          CALL CMOS_READ    ; GET ALARM REGISTER
257 00B8 EB 0000 E
258 00B9 24 BF
259 00B9 80 E0
260 00C2 EB 0000 E
261 00C5 58
262 00C6 C6 06 00A0 R 00
263 00CB FB
264 00CC 1F
265 00CD F8
266 00CE EB 87
267
268 00D0          EVENT_WAIT:    ENDP
269          ;--- JOY STICK -----
270          ; THIS ROUTINE WILL READ THE JOYSTICK PORT
271          ;:
272          ; INPUT
273          ; (DX)=0 READ THE CURRENT SWITCHES
274          ; RETURNS (AL)= SWITCH SETTINGS IN BITS 7-4
275          ;:
276          ; (DX)=1 READ THE RESISTIVE INPUTS
277          ; RETURNS (AX)=A(X) VALUE
278          ; (DX)=2 READ B(X) VALUE
279          ; (DX)=3 READ B(Y) VALUE
280          ; (DX)=4 READ A(Y) VALUE
281          ;:
282          ; CY FLAG ON IF NO ADAPTER CARD OR INVALID CALL
283
284
285 00D0          JOY_STICK:    PROC NEAR
286 00D0 FB          STI             ; INTERRUPTS BACK ON
287 00D1 BB C2          MOV AX,DX          ; GET SUB FUNCTION CODE
288 00D3 BA 201H        MOV DX,201H        ; ADDRESS OF PORT
289 00D4 00
290 00D8 74 0B          OR AL,AL          ; READ SWITCHES
291 00DA FE C8
292 00DC 74 0C
293 00DE E9 0054 R
294 00E1
295 00E2 FB
296 00E2 E9 0057 R
297
298 00E5          JOY_1:       STI             ; READ SWITCHES
299 00E5 EC          IN AL,DX          ; READ RESISTIVE INPUTS
300 00E6 24 F0          AND AL,0F0H        ; STRIP UNWANTED BITS OFF
301 00E8 EB F7          JMP JOY_1          ; FINISHED
302
303 00EA          JOY_2:       IN AL,DX          ; READ RESISTIVE INPUTS
304 00EA BB 01          AND AL,0F0H        ; STRIP UNWANTED BITS OFF
305 00EC EB 0108 R
306 00EF 51
307 00F0 BB 02
308 00F0 EB 0108 R
309 00F5 51
310 00F6 EB 03 04
311 00F8 EB 0108 R
312 00FB 51
313 00FC BB 08
314 00F8 EB 0108 R
315 0101 BB D1
316 0103 59
317 0104 5B
318 0105 58
319 0106 EB D9
320
321 0108          TEST_CORD:  PROC NEAR
322 0108 52          PUSH DX          ; SAVE
323 0109 FA          CLI             ; BLOCK INTERRUPTS WHILE READING
324 010A BB 00          MOV AL,0          ; SET UP TO LATCH TIMER 0
325 010B 43
326 010E EB 00
327 0110 EB 40
328 0112 EB 00
329 0114 EB E0
330 0116 EB 40
331 0118 EB 00
332 011A 50
333 011B BB 04FF
334 011E EE
335 011F EB 00
336 0121
337 0121 EC
338 0122 84 C3
339 0124 EB FB
TEST_CORD_1:    IN AL,DX          ; READ VALUES
TEST_1:          TEST AL,BL          ; HAS PULSE ENDED?
TEST_CORD_1:    LOOPNZ TEST_CORD_1

```

```

340 012C 83 F9 00      CMP    CX,0
341 0129 59      POP    CX
342 012A 75 04      JNZ    SHORT TEST_CORD_2
343 012C 28 C9      SUB    CX,CX      ; SET 0 COUNT FOR RETURN
344 012D EB 28      JMP    SHORT TEST_CORD_3      ; EXIT WITH COUNT = 0
345 0130
TEST_CORD_2:
346 0130 B0 00      MOV    AL,0      ; SET UP TO LATCH TIMER 0
347 0132 E4 43      OUT   TIMER+3,AL
348 0134 EB 00      JMP    $+2
349 0136 E4 40      IN    AL,TIMER      ; READ LOW BYTE OF TIMER 0
350 0138 00 00      MOV    AH,AL
351 013A EB 00      JMP    $+2
352 013C E4 40      IN    AL,TIMER      ; READ HIGH BYTE OF TIMER 0
353 013E 86 E0      XCHG   AH,AL      ; REARRANGE TO HIGH,LOW
354
355 0140 3B C8      CMP    CX,AX      ; CHECK FOR COUNTER WRAP
356 0142 73 0B      JAE    TEST_CORD_4
357 0144 00 00      PUSH   DX
358 0145 BA FFFF      MOV    DX,-1
359
360 0148 2B D0      SUB    DX,AX      ; ADJUST FOR WRAP
361 014A 03 CA      ADD    CX,DX
362 014C 5A      POP    DX
363 014D EB 02      JMP    SHORT TEST_CORD_5
364
365 014F
TEST_CORD_4:
366 014F 2B C8      SUB    CX,AX
367 0151
TEST_CORD_5:
368 0151 81 E1 IFF0      AND    CX,IFF0H      ; ADJUST
369 0155 C1 E9 04      SHR    CX,4
370
371 0158
TEST_CORD_3:
372 0158 FB      STI
373 0159 BA 0201      MOV    DX,201H      ; INTERRUPTS BACK ON
374 015B 81 00      PUSH   CX      ; FLUSH OTHER INPUTS
375 015D 00 00
376 015E B9 04FF      PUSH   AX
377 0161
TEST_CORD_6:
378 0161 EC      IN    AL,DX
379 0162 A8 0F      TEST   AL,0FH
380 0164 EB FB      LOOPNZ TEST_CORD_6
381
382 0166 5B      POP    AX
383 0167 59      POP    CX
384 0168 5A      POP    DX      ; SET COUNT
385
386 0169 C3      RET
387
388 016A
TEST_CORD:
389 016A JOY_STICK      ENDP
390
391 016A
WAIT:
392 016A 1E      PROC   NEAR
393 016B E9 0000 E      PUSH   DS      ; SAVE
394 016E F6 00 00A0 R 01      CALL   DS
395 0173 74 05      TEST   RTC_WAIT_FLAG,01      ; TEST FOR FUNCTION ACTIVE
396 0175 1F      POP    DS
397 0176 F9      STC
398 0177 E9 0057 R      JMP    C1_F      ; SET ERROR
399 0178
WAIT_1:
400 017A FA      CLI
401 017B E4 A1      IN    AL,INTB01      ; NO INTERRUPTS ALLOWED
402 017D EB 00      JMP    $+2      ; ENSURE INTERRUPT UNMASKED
403 017F 24 FE      AND    AL,0FEH
404 0181 E6 A1      OUT   INTB01,AL
405 0183 01 E0 009A R      MOV    USER_FLAG,SEG.DS      ; SET UP TRANSFER TABLE
406 0187 C7 06 0098 R 00A0 R      MOV    DS,INTB01FFSET RTC_WAIT_FLAG
407 018D 89 0E 009E R      MOV    RTC_HIGH,CX
408 0193 89 16 009C R      MOV    RTC_LOW,DX
409 0195 C6 06 00A0 R 01      MOV    RTC_WAIT_FLAG,01      ; SET ON FUNCTION ACTIVE SWITCH
410 019A 50      PUSH   AX
411 019B 00 00      MOV    AX,X*CMOS_REG_B      ; SAVE (AH)
412 019E E8 0000 E      CALL   CMOS_READ      ; ENABLE PIE
413 01A1 24 7F      AND    AL,07FH      ; READ ALARM BYTE
414 01A3 0C 40      OR    AL,040H      ; CLEAR SIT BIT
415 01A5 86 E0      XCHG   AH,AL      ; ENABLE PIE BIT
416 01A7 E8 0000 E      CALL   CMOS_WRITE      ; DATA TO WORK REGISTER
417 01A8 58      POP    AX      ; WRITE NEW ALARM BYTE
418
419
WAIT_TILL_RTC_TIMEOUT_POSTED: (WITH ERROR TIMEOUT)
420
421 01AB FB      STI      ; ENABLE INTERRUPTS
422 01AC 51      PUSH   CX
423 01AD 52      PUSH   DX      ; SAVE CALLERS PARAMETERS
424 01AE 87 D1      XCHG   DX,CX      ; SWAP COUNT WORK REGISTERS
425 01B0
WAIT_2:
426 01B0 F6 06 00A0 R 80      TEST   RTC_WAIT_FLAG,080H      ; CHECK FOR END OF WAIT - CLEAR CARRY
427 01B5 E1 F9      LOOPZ  WAIT_2      ; DECREMENT TIMEOUT DELAY TILL WAIT END
428 01B7 75 05      JNZ    WAIT_9      ; EXIT IF RTC TIMER WAIT ENDED FLAG SET
429 01B9 83 EA 01      SUB    DX,1      ; DECREMENT ERROR TIMEOUT COUNTER
430 01BC 73 F2      JNC    WAIT_2      ; LOOP TILL COUNTERS TIMEOUT
431 01BE
WAIT_9:
432 01BE C6 06 00A0 R 00      MOV    RTC_WAIT_FLAG,0      ; SET FUNCTION INACTIVE
433 01C3 5A      POP    DX
434 01C4 59      POP    CX      ; RESTORE CALLERS PARAMETERS
435 01C5 1F      POP    DS
436 01C6 F8      CLC
437 01C7 E9 0057 R      JMP    C1_F      ; CLEAR CARRY FLAG
438
439 01CA
WAIT: ENDP

```



```

554 01D1 8C 16 0069 R MOV    $10_ROM_SEG,SS      ; SAVE USERS STACK SEGMENT
555 01D5 89 26 0067 R MOV    $10_ROM_INIT,SP    ; SAVE USERS STACK POINTER
556
557 ===== SET UP THE PROTECTED MODE DEFINITIONS =====
558
559 ===== MAKE A 24 BIT ADDRESS OUT OF THE ES:SI FOR THE GDT POINTER
560
561 ASSUME DS:NOTHING      ; POINT (DS) TO USERS CONTROL BLOCK
562 MOV    AX,ES             ; GET THE GDT DATA SEGMENT
563 MOV    DS,AX             ; MOVE THE GDT SEGMENT POINTER TO (DS)
564 MOV    DH,AH             ; BUILD HIGH BYTE OF THE 24 BIT ADDRESS
565 SHR    DH,4              ; USE ONLY HIGH NIBBLE SHIFT - RIGHT 4
566 SHL    AX,4              ; STRIP HIGH NIBBLE FROM (AX)
567 ADD    AX,SI             ; ADD THE GDT OFFSET TO DEVELOP LOW WORD
568 ADC    DH,0              ; ADJUST HIGH BYTE IF CARRY FROM LOW
569
570 ===== SET THE GDT_LOC
571
572 01E4 C7 44 08 FFFF MOV    [SI].CGDT_LOC,SEG LIMIT,MAX SEG LEN
573 01EF 89 04 0A MOV    [SI].CGDT_LOC,LO_WORD,AX      ; SET THE LOW WORD
574 01F2 88 74 0C MOV    [SI].CGDT_LOC,BASE_HI_BYT,DH      ; SET THE HIGH BYTE
575 01F5 C7 44 0E 0000 MOV    [SI].CGDT_LOC,DATA_RESERVED,0 ; RESERVED
576
577 ===== SET UP THE CODE SEGMENT DESCRIPTOR
578
579 01FA C7 44 20 FFFF MOV    [SI].BIOS_CS,SEG LIMIT,MAX SEG LEN
580 01FF C7 44 22 0000 MOV    [SI].BIOS_CS,BASE LO_WORD,CSEG@ LO      ; LOW WORD OF (CS)= 0
581 0204 C6 44 24 0F MOV    [SI].BIOS_CS,BASE_HI_BYT,CSEG@ HI      ; HIGH BYTE OF (CS)= 0FH
582 0208 C6 44 25 98 MOV    [SI].BIOS_CS,DATA_ACC_RIGHTS,CPL0_CODE_ACCESS
583 020C C7 44 26 0000 MOV    [SI].BIOS_CS,DATA_RESERVED,0 ; RESERVED
584
585 ===== MAKE A 24 BIT ADDRESS OUT OF THE (SS) - ( (SP) REMAINS USER (SP) )
586
587 0211 8C D0 MOV    AX,SS             ; GET THE CURRENT STACK SEGMENT
588 0213 8A F4 MOV    DH,AH             ; FORM HIGH BYTE OF 24 BIT ADDRESS
589 0215 C0 EE 04 SHR    DH,4              ; FORM HIGH BYTE - SHIFT RIGHT 4
590 0219 C1 E0 04 SHL    AX,4              ; STRIP HIGH NIBBLE FROM (AX)
591
592 ===== SS IS NOW IN POSITION FOR A 24 BIT ADDRESS --> SETUP THE (SS) DESCRIPTOR
593
594 021B C7 44 28 FFFF MOV    [SI].TEMP_SS,SEG LIMIT,MAX SEG LEN ; SET THE SS SEGMENT LIMIT
595 0220 89 44 2A MOV    [SI].TEMP_SS,BASE LO_WORD,AX      ; SET THE LOW WORD
596 0223 88 74 2C MOV    [SI].TEMP_SS,BASE_HI_BYT,DH      ; SET THE HIGH BYTE
597 0226 C6 44 24 D9 MOV    [SI].TEMP_SS,DATA_ACC_RIGHTS,CPL0_DATA_ACCESS ; SET CPL0
598
599 ===== GATE ADDRESS BIT 20 ON (DISABLE INTERRUPTS)
600
601 022A B4 DF MOV    AH,ENABLE_BIT20      ; GET ENABLE MASK
602 022C E8 03CC R CALL   GATE_A20      ; ENABLE A20 AND CLEAR INTERRUPTS
603 022F 3C 00 CMP    AL,0              ; WAS THE COMMAND ACCEPTED?
604 0231 74 06 JZ    BL4              ; GO IF YES
605
606 0233 B0 03 MOV    AL,03H             ; SET THE ERROR FLAG IF NOT
607 0233 E8 80 OUT   MFG_PORT,AL      ; EARLY ERROR EXIT
608 0237 EB 51 JMP    SHORT_SHUTS
609
610 ===== SET SHUTDOWN RETURN ADDRESS AND DISABLE NMI
611 0239 BL4: MOV    AX,9*H+CMOS_SHUT_DOWN+NMI ; SET THE SHUTDOWN BYTE LOCATION
612 0239 B8 09BF CALL   CMOS_WRITE      ; TO SHUT DOWN 9 AND DISABLE NMI
613 023C E8 0000 E
614
615 ===== CLEAR EXCEPTION ERROR FLAG
616
617 023F 2A C0 SUB    AL,AL             ; SET ERROR FLAG LOCATION TO 0
618 0241 E8 80 OUT   MFG_PORT,AL
619
620 ===== LOAD THE IDT AND GDT
621
622 0243 BD 02C6 R MOV    BP,OFFSET ROM_IDT_LOC
623         SEG09 CS             ; LOAD THE IDT
624 0246 2E +    DB    02EH             ; REGISTER FROM THIS AREA
625         LIDT [BP]
626 0247 0F +    DB    00FH             ; REGISTER FROM THIS AREA
627 0248 08 +    ??0001 LABEL  BYTE
628 0248 8B 5E 00 MOV    BX,WORD PTR [BP]
629 0248 08 +    ??0002 LABEL  BYTE
630 0248 +    ??0002 ORG   CS:??0001
631 0248 01 +    DB    001H             ; REGISTER FROM THIS AREA
632 0248 +    ORG   OFFSET CS:??0002
633
634         LGDT [SI].CGDT_LOC      ; LOAD GLOBAL DESCRIPTOR TABLE REGISTER
635 0248 0F +    DB    00FH             ; REGISTER FROM THIS AREA
636 024C +    ??0003 LABEL  BYTE
637 024C 8B 54 08 MOV    DX,WORD PTR [SI].CGDT_LOC
638 024F +    ??0004 LABEL  BYTE
639 024C +    ORG   CS:??0003
640 024C 01 +    DB    001H             ; REGISTER FROM THIS AREA
641 024F +    ORG   OFFSET CS:??0004
642
643 ===== SWITCH TO VIRTUAL MODE
644
645 024F B8 0001 MOV    AX,VIRTUAL_ENABLE ; MACHINE STATUS WORD NEEDED TO
646 LMSW   AX             ; SWITCH TO VIRTUAL MODE
647 0252 0F 01 F0 +    DB    00FH,001H,00FH
648 0255 EA DB    DEAH             ; PURGE PRE-FETCH QUEUE WITH FAR JUMP
649 0256 025A R DW    OFFSET VIRT      ; - TO OFFSET
650 0256 0200 DW    BIOS_CS          ; - IN SEGMENT -PROTECTED MODE SELECTOR
651 025A VIRT:
652
653 ===== IN PROTECTED MODE - SETUP STACK SELECTOR AND SOURCE/TARGET SELECTORS
654
655 025A B8 0028 MOV    AX,TEMP_SS      ; USER'S SS+SP IS NOT A DESCRIPTOR
656 025D 8E D0 MOV    SS,AX             ; LOAD STACK SELECTOR
657 025E 0010 MOV    AX,SOURCE        ; GET SOURCE SELECTOR ENTRY
658 0262 8E 08 MOV    DS,AX             ; LOAD SOURCE SELECTOR
659 0264 B8 0018 MOV    AX,TARGET        ; GET THE TARGET ENTRY
660 0267 8E C0 MOV    ES,AX             ; LOAD TARGET SELECTOR
661 0269 2B F6 SUB    S1,S1             ; SET SOURCE INDEX REGISTER TO ZERO
662 026B 2B FF SUB    D1,D1             ; SET TARGET INDEX REGISTER TO ZERO
663
664 026D F3/ A5 REP    MOVSW      ; MOVE THE BLOCK COUNT PASSED IN (CX)
665
666
667 ===== CHECK FOR MEMORY PARITY BEFORE SHUTDOWN

```

```

668
669 026F E4 61      IN    AL,_PORT_B      ; GET THE PARITY LATCHES
670 0271 24 C0      AND   AL,_PARITY_ERR  ; STRIP UNWANTED BITS
671 0273 T4 12      JZ    DONE1          ; GO IF NO PARITY ERROR
672
673
674      ;---- CLEAR PARITY BEFORE SHUTDOWN
675 0275 8B 05      MOV   AX,DS:[DI]      ; FETCH CURRENT SOURCE DATA
676 0277 89 05      MOV   DS:[DI],AX    ; WRITE IT BACK
677 0279 B0 01      MOV   AL,_01        ; SET PARITY CHECK ERROR = 01
678 027B E6 80      OUT   MFC_PORT,AL  ; AL,_PORT_B
679 027C E4 61      IN    AL,_PORT_B      ; AL,_RAM_PAR_OFF
680 027F E0 0C      OUT   PORT_B,AL    ; TOGGLE PARITY CHECK LATCHES
681 0281 E6 61      OUT   AL,_RAM_PAR_ON ; TO CLEAR THE PENDING ERROR
682 0283 24 F3      AND   AL,_RAM_PAR_ON ; AND ENABLE CHECKING
683 0285 E6 61      OUT   PORT_B,AL
684
685
686
687 0287          DONE1:  JMP   PROC_SHUTDOWN ; GO RESET PROCESSOR AND SHUTDOWN
688 0287 E9 0000 E
689
690
691      ;===== RETURN FROM SHUTDOWN =====
692
693 028A          SHUT9:  ASSUME DS:DATA ; RESTORE USERS STACK
694
695 028A B8 ---- R  MOV   AX,DATA
696 028D B8 D8      MOV   DS,AX        ; SET DS TO DATA AREA
697 028F B8 16 0069 R MOV   SS,@IO_ROM_SEG ; GET USER STACK SEGMENT
698 0293 B8 26 0067 R MOV   SP,@IO_ROM_INIT ; GET USER STACK POINTER
699
700      ;---- GATE ADDRESS BIT 20 OFF
701
702 0297 B4 DD      MOV   AH,DISABLE_BIT20 ; DISABLE MASK
703 0299 E8 03CC R  CALL  GATE_A20    ; GATE ADDRESS 20 LINE OFF
704 029C 3C 00      CMP   AL,0         ; COMMAND ACCEPTED?
705 029E T4 0A      JZ    DONE3          ; GO IF YES
706
707 02A0 E4 80      IN    AL,MFC_PORT ; CHECK FOR ANY OTHER ERROR FIRST
708 02A2 3C 00      CMP   AL,0         ; WAS THERE AN ERROR?
709 02A4 T5 04      JNZ   DONE3          ; REPORT FIRST ERROR IF YES
710 02A6 B0 03      MOV   AL,03H        ; ELSE SET GATE A20 ERROR FLAG
711 02A8 E6 80      OUT  MFC_PORT,AL
712
713
714
715 02AA          DONE3:  MOV   AX,CMOS_REG_D ; CLEAR (AH) TO ZERO AND (AL) TO DEFAULT
716 02AA B8 0000 D  OUT  CMOS_PORT,AL ; ENABLE NMI INTERRUPTS
717 02AD E6 70
718
719 02AF 1F          POP   DS        ; RESTORE USER DATA SEGMENT
720 02B0 07          POP   ES        ; RESTORE USER EXTRA SEGMENT
721 02B1 E4 80      IN    AL,MFC_PORT ; GET THE ENDING STATUS RETURN CODE
722 02B2 3C 00      MOV   BP,SP        ; POINT TO THE END OF USER STACK
723 02B5 B8 46 0F    MOV   [BP+15],AL ; PLACE ERROR CODE INTO STACK AT (AH)
724 02B8 3A E0      CMP   AH,AL        ; SET THE ZF & CY FLAGS WITH RETURN CODE
725 02B8 61          POPA
726 02BB FB          STI   STI        ; RESTORE THE GENERAL PURPOSE REGISTERS
727 02BC          DONE4:  PROC  FAR      ; TURN INTERRUPTS ON
728 02BC RET 2        ; RETURN WITH FLAGS SET -- (AH)= CODE
729 02BF          DONE4:  ENDP
730
731      ;---- RESTORE THE USERS REGISTERS AND SET RETURN CODES
732
733 02BF          EX_INT:  MOV   AL,02H        ; GET EXCEPTION ERROR CODE
734 02BF B0 02      OUT  MFC_PORT,AL ; SET EXCEPTION INTERRUPT OCCURRED FLAG
735 02C1 E6 80      JMP   PROC_SHUTDOWN ; CAUSE A EARLY SHUTDOWN
736 02C3 E9 0000 E
737
738
739
740 02C6          ROM_IDT_LOC: DW    ROM_IDT_END-ROM_IDT ; LENGTH OF ROM IDT TABLE
741 02C6 0100        DW    ROM_IDT        ; LOW WORD OF BASE ADDRESS
742 02C8 02CC R      DB    CSEG@_HI ; HIGH BYTE OF BASE ADDRESS
743 02C8 0F          DB    0             ; RESERVED
744 02C8 00          DB    0
745
746
747 02CC          ROM_IDT:  DW    EX_INT        ; EXCEPTION 00
748 02CC 02BF R      DW    BIOS_CS      ; DESTINATION OFFSET
749 02CC 0020        DW    0             ; DESTINATION SEGMENT SELECTOR
750 02C0 0020        DB    0             ; WORD COPY COUNT
751 02D0 00          DB    TRAP_GATE   ; GATE TYPE - ACCESS RIGHTS BYTE
752 02D1 67          DW    0             ; RESERVED
753 02D2 0000        DW    0
754
755 02D4 02BF R      DW    EX_INT        ; EXCEPTION 01
756 02D6 0020        DW    BIOS_CS      ; DESTINATION OFFSET
757 02D8 00          DW    0             ; DESTINATION SEGMENT SELECTOR
758 02D9 67          DB    0             ; WORD COPY COUNT
759 02DA 0000        DB    TRAP_GATE   ; GATE TYPE - ACCESS RIGHTS BYTE
760 02D0 0000        DW    0             ; RESERVED
761 02DC 02BF R      DW    EX_INT        ; EXCEPTION 02
762 02DE 0020        DW    BIOS_CS      ; DESTINATION OFFSET
763 02E0 00          DB    0             ; DESTINATION SEGMENT SELECTOR
764 02E1 67          DB    0             ; WORD COPY COUNT
765 02E2 0000        DW    TRAP_GATE   ; GATE TYPE - ACCESS RIGHTS BYTE
766 02E0 0000        DW    0             ; RESERVED
767 02E4 02BF R      DW    EX_INT        ; EXCEPTION 03
768 02E6 0020        DW    BIOS_CS      ; DESTINATION OFFSET
769 02E8 00          DB    0             ; DESTINATION SEGMENT SELECTOR
770 02E9 67          DB    0             ; WORD COPY COUNT
771 02EA 0000        DB    TRAP_GATE   ; GATE TYPE - ACCESS RIGHTS BYTE
772 02E0 0000        DW    0             ; RESERVED
773 02EC 02BF R      DW    EX_INT        ; EXCEPTION 04
774 02EE 0020        DW    BIOS_CS      ; DESTINATION OFFSET
775 02F0 00          DB    0             ; DESTINATION SEGMENT SELECTOR
776 02F2 67          DB    0             ; WORD COPY COUNT
777 02F2 0000        DW    TRAP_GATE   ; GATE TYPE - ACCESS RIGHTS BYTE
778
779 02F4 02BF R      DW    EX_INT        ; EXCEPTION 05
780 02F6 0020        DW    BIOS_CS      ; DESTINATION OFFSET
781 02F8 00          DB    0             ; WORD COPY COUNT

```

782 02F9 87	DB	TRAP_GATE	; GATE TYPE - ACCESS RIGHTS BYTE
783 02FA 0000	DW	0	RESERVED
784			
785 02FC 02BF R	DW	EX_INT	; EXCEPTION 06
786 02FE 0020	DW	BIOS_CS	; DESTINATION OFFSET
787 0300 00	DB	0	; DESTINATION SEGMENT SELECTOR
788 0301 87	DB	TRAP_GATE	; WORD COPY COUNT
789 0302 0000	DW	0	; GATE TYPE - ACCESS RIGHTS BYTE
790			RESERVED
791 0304 02BF R	DW	EX_INT	; EXCEPTION 07
792 0306 0020	DW	BIOS_CS	; DESTINATION OFFSET
793 0308 00	DB	0	; DESTINATION SEGMENT SELECTOR
794 0309 87	DB	TRAP_GATE	; WORD COPY COUNT
795 030A 0000	DW	0	; GATE TYPE - ACCESS RIGHTS BYTE
796			RESERVED
797 030C 02BF R	DW	EX_INT	; EXCEPTION 08
798 030E 0020	DW	BIOS_CS	; DESTINATION OFFSET
799 0310 00	DB	0	; DESTINATION SEGMENT SELECTOR
800 0311 87	DB	TRAP_GATE	; WORD COPY COUNT
801 0312 0000	DW	0	; GATE TYPE - ACCESS RIGHTS BYTE
802			RESERVED
803 0314 02BF R	DW	EX_INT	; EXCEPTION 09
804 0316 0020	DW	BIOS_CS	; DESTINATION OFFSET
805 0318 00	DB	0	; DESTINATION SEGMENT SELECTOR
806 0319 87	DB	TRAP_GATE	; WORD COPY COUNT
807 031A 0000	DW	0	; GATE TYPE - ACCESS RIGHTS BYTE
808			RESERVED
809 031C 02BF R	DW	EX_INT	; EXCEPTION 10
810 031E 0020	DW	BIOS_CS	; DESTINATION OFFSET
811 0320 00	DB	0	; DESTINATION SEGMENT SELECTOR
812 0321 87	DB	TRAP_GATE	; WORD COPY COUNT
813 0322 0000	DW	0	; GATE TYPE - ACCESS RIGHTS BYTE
814			RESERVED
815 0324 02BF R	DW	EX_INT	; EXCEPTION 11
816 0326 0020	DW	BIOS_CS	; DESTINATION OFFSET
817 0328 00	DB	0	; DESTINATION SEGMENT SELECTOR
818 0329 87	DB	TRAP_GATE	; WORD COPY COUNT
819 032A 0000	DW	0	; GATE TYPE - ACCESS RIGHTS BYTE
820			RESERVED
821 032C 02BF R	DW	EX_INT	; EXCEPTION 12
822 032E 0020	DW	BIOS_CS	; DESTINATION OFFSET
823 0330 00	DB	0	; DESTINATION SEGMENT SELECTOR
824 0331 87	DB	TRAP_GATE	; WORD COPY COUNT
825 0332 0000	DW	0	; GATE TYPE - ACCESS RIGHTS BYTE
826			RESERVED
827 0334 02BF R	DW	EX_INT	; EXCEPTION 13
828 0336 0020	DW	BIOS_CS	; DESTINATION OFFSET
829 0338 00	DB	0	; DESTINATION SEGMENT SELECTOR
830 0339 87	DB	TRAP_GATE	; WORD COPY COUNT
831 033A 0000	DW	0	; GATE TYPE - ACCESS RIGHTS BYTE
832			RESERVED
833 033C 02BF R	DW	EX_INT	; EXCEPTION 14
834 033E 0020	DW	BIOS_CS	; DESTINATION OFFSET
835 0340 00	DB	0	; DESTINATION SEGMENT SELECTOR
836 0341 87	DB	TRAP_GATE	; WORD COPY COUNT
837 0342 0000	DW	0	; GATE TYPE - ACCESS RIGHTS BYTE
838			RESERVED
839 0344 02BF R	DW	EX_INT	; EXCEPTION 15
840 0346 0020	DW	BIOS_CS	; DESTINATION OFFSET
841 0348 00	DB	0	; DESTINATION SEGMENT SELECTOR
842 0349 87	DB	TRAP_GATE	; WORD COPY COUNT
843 034A 0000	DW	0	; GATE TYPE - ACCESS RIGHTS BYTE
844			RESERVED
845 034C 02BF R	DW	EX_INT	; EXCEPTION 16
846 034E 0020	DW	BIOS_CS	; DESTINATION OFFSET
847 0350 00	DB	0	; DESTINATION SEGMENT SELECTOR
848 0351 87	DB	TRAP_GATE	; WORD COPY COUNT
849 0352 0000	DW	0	; GATE TYPE - ACCESS RIGHTS BYTE
850			RESERVED
851 0354 02BF R	DW	EX_INT	; EXCEPTION 17
852 0356 0020	DW	BIOS_CS	; DESTINATION OFFSET
853 0358 00	DB	0	; DESTINATION SEGMENT SELECTOR
854 0359 87	DB	TRAP_GATE	; WORD COPY COUNT
855 035A 0000	DW	0	; GATE TYPE - ACCESS RIGHTS BYTE
856			RESERVED
857 035C 02BF R	DW	EX_INT	; EXCEPTION 18
858 035E 0020	DW	BIOS_CS	; DESTINATION OFFSET
859 0360 00	DB	0	; DESTINATION SEGMENT SELECTOR
860 0361 87	DB	TRAP_GATE	; WORD COPY COUNT
861 0362 0000	DW	0	; GATE TYPE - ACCESS RIGHTS BYTE
862			RESERVED
863 0364 02BF R	DW	EX_INT	; EXCEPTION 19
864 0366 0020	DW	BIOS_CS	; DESTINATION OFFSET
865 0368 00	DB	0	; DESTINATION SEGMENT SELECTOR
866 0369 87	DB	TRAP_GATE	; WORD COPY COUNT
867 036A 0000	DW	0	; GATE TYPE - ACCESS RIGHTS BYTE
868			RESERVED
869 036C 02BF R	DW	EX_INT	; EXCEPTION 20
870 036E 0020	DW	BIOS_CS	; DESTINATION OFFSET
871 0370 00	DB	0	; DESTINATION SEGMENT SELECTOR
872 0371 87	DB	TRAP_GATE	; WORD COPY COUNT
873 0372 0000	DW	0	; GATE TYPE - ACCESS RIGHTS BYTE
874			RESERVED
875 0374 02BF R	DW	EX_INT	; EXCEPTION 21
876 0376 0020	DW	BIOS_CS	; DESTINATION OFFSET
877 0378 00	DB	0	; DESTINATION SEGMENT SELECTOR
878 0379 87	DB	TRAP_GATE	; WORD COPY COUNT
879 037A 0000	DW	0	; GATE TYPE - ACCESS RIGHTS BYTE
880			RESERVED
881 037C 02BF R	DW	EX_INT	; EXCEPTION 22
882 037E 0020	DW	BIOS_CS	; DESTINATION OFFSET
883 0380 00	DB	0	; DESTINATION SEGMENT SELECTOR
884 0381 87	DB	TRAP_GATE	; WORD COPY COUNT
885 0382 0000	DW	0	; GATE TYPE - ACCESS RIGHTS BYTE
886			RESERVED
887 0384 02BF R	DW	EX_INT	; EXCEPTION 23
888 0386 0020	DW	BIOS_CS	; DESTINATION OFFSET
889 0388 00	DB	0	; DESTINATION SEGMENT SELECTOR
890 0389 87	DB	TRAP_GATE	; WORD COPY COUNT
891 038A 0000	DW	0	; GATE TYPE - ACCESS RIGHTS BYTE
892			RESERVED
893 038C 02BF R	DW	EX_INT	; EXCEPTION 24
894 038E 0020	DW	BIOS_CS	; DESTINATION OFFSET
895 0390 00	DW	0	; DESTINATION SEGMENT SELECTOR

```

898 0391 87      DB     TRAP_GATE      ; GATE TYPE - ACCESS RIGHTS BYTE
897 0392 0000    DW     0             ; RESERVED
898
899 0394 02BF R  DW     EX_INT        ; EXCEPTION 25
900 0396 0020    DW     BIOS_CS       ; DESTINATION OFFSET
901 0398 00      DB     0             ; DESTINATION SEGMENT SELECTOR
902 0399 87      DB     TRAP_GATE      ; WORD COPY COUNT
903 039A 0000    DW     0             ; GATE TYPE - ACCESS RIGHTS BYTE
904
905 039C 02BF R  DW     EX_INT        ; RESERVED
906 039E 0020    DW     BIOS_CS       ; EXCEPTION 26
907 03A0 00      DB     0             ; DESTINATION OFFSET
908 03A1 87      DB     TRAP_GATE      ; DESTINATION SEGMENT SELECTOR
909
910 03A2 0000    DW     0             ; WORD COPY COUNT
911 03A4 02BF R  DW     EX_INT        ; GATE TYPE - ACCESS RIGHTS BYTE
912 03A6 0020    DW     BIOS_CS       ; RESERVED
913 03A8 00      DB     0             ; EXCEPTION 27
914 03A9 87      DB     TRAP_GATE      ; DESTINATION OFFSET
915 03AA 0000    DW     0             ; DESTINATION SEGMENT SELECTOR
916
917 03AC 02BF R  DW     EX_INT        ; WORD COPY COUNT
918 03AE 0020    DW     BIOS_CS       ; GATE TYPE - ACCESS RIGHTS BYTE
919 03B0 00      DB     0             ; RESERVED
920 03B1 87      DB     TRAP_GATE      ; EXCEPTION 28
921 03B2 0000    DW     0             ; DESTINATION OFFSET
922
923 03B4 02BF R  DW     EX_INT        ; DESTINATION SEGMENT SELECTOR
924 03B6 0020    DW     BIOS_CS       ; WORD COPY COUNT
925 03B8 00      DB     0             ; GATE TYPE - ACCESS RIGHTS BYTE
926 03B9 87      DB     TRAP_GATE      ; RESERVED
927 03BA 0000    DW     0             ; EXCEPTION 29
928
929 03BC 02BF R  DW     EX_INT        ; DESTINATION OFFSET
930 03BE 0020    DW     BIOS_CS       ; DESTINATION SEGMENT SELECTOR
931 03C0 00      DB     0             ; WORD COPY COUNT
932 03C1 87      DB     TRAP_GATE      ; GATE TYPE - ACCESS RIGHTS BYTE
933 03C2 0000    DW     0             ; RESERVED
934
935 03C4 02BF R  DW     EX_INT        ; EXCEPTION 30
936 03C6 0020    DW     BIOS_CS       ; DESTINATION OFFSET
937 03C8 00      DB     0             ; DESTINATION SEGMENT SELECTOR
938 03C9 87      DB     TRAP_GATE      ; WORD COPY COUNT
939 03CA 0000    DW     0             ; GATE TYPE - ACCESS RIGHTS BYTE
940 03CC
941
942 03CC          BLOCKMOVE    ENDP      ; RESERVED
ROM_IDT_END:
```

```

943 PAGE
944
945 ;----- GATE_A20
946 ;----- THIS ROUTINE CONTROLS A SIGNAL WHICH GATES ADDRESS BIT 20.
947 ;----- THE GATE A20 SIGNAL IS AN OUTPUT OF THE 8042 SLAVE PROCESSOR.
948 ;----- ADDRESS BIT 20 SHOULD BE GATED ON BEFORE ENTERING PROTECTED
949 ;----- MODE. INTERRUPTS ARE LEFT DISABLED ON EXIT.
950
951 ;----- INPUT
952 ;----- (AH)= DDH ADDRESS BIT 20 GATE OFF. (A20 ALWAYS ZERO)
953 ;----- (AH)= DHF ADDRESS BIT 20 GATE ON. (A20 CONTROLLED BY 80286)
954 ;----- OUTPUT
955 ;----- (AL)= 00H OPERATION SUCCESSFUL. 8042 HAS ACCEPTED COMMAND.
956 ;----- (AL)= 02H FAILURE-8042 UNABLE TO ACCEPT COMMAND.
957
958 03CC GATE_A20 PROC
959 03CC 51 PUSH CX ; SAVE USERS (CX)
960 03CD FA CL1 ; DISABLE INTERRUPTS WHILE USING 8042
961 03CE E8 03E5 R CALL EMPTY_8042 ; INSURE 8042 INPUT BUFFER EMPTY
962 03D1 75 10 JNZ GATE_A20_RETURN ; EXIT IF 8042 UNABLE TO ACCEPT COMMAND
963 03D3 01 MOV AL,1 ; 8042 INPUT BUFFER FULL
964 03D5 E6 64 OUT STATUS_PORT,AL ; 8042 COMMAND TO WRITE OUTPUT PORT
965 03D7 E8 03E5 R CALL EMPTY_8042 ; 8042 PORT DATA
966 03DA 75 07 JNZ GATE_A20_RETURN ; EXIT IF 8042 UNABLE TO ACCEPT COMMAND
967 03DC 8A C4 MOV AL, AH ; 8042 PORT DATA
968 03DC E6 60 OUT PORT_A_AL ; OUTPUT PORT DATA TO 8042
969 03E0 E8 03E5 R CALL EMPTY_8042 ; WAIT FOR 8042 TO ACCEPT PORT DATA
970
971 ;----- 8042 OUTPUT WILL SWITCH WITHIN 20 MICRO SECONDS OF ACCEPTING PORT DATA
972
973 03E3 GATE_A20_RETURN: POP CX ; RESTORE USERS (CX)
974 03E3 59 RET
975 03E4 C3
976
977 ;----- EMPTY_8042
978 ;----- THIS ROUTINE WAITS FOR THE 8042 INPUT BUFFER TO EMPTY.
979 ;----- INPUT
980 ;----- NONE
981 ;----- OUTPUT
982 ;----- (AL)= 00H 8042 INPUT BUFFER EMPTY (ZERO FLAG SET)
983 ;----- (AL)= 02H TIME OUT, 8042 INPUT BUFFER FULL (NON-ZERO FLAG SET)
984 ;----- (CX) - MODIFIED
985
986 03E5 EMPTY_8042: SUB CX,CX ; (CX)=0, WILL BE USED AS TIME OUT VALUE
987 03E7 IN AL,STATUS_PORT ; READ 8042 STATUS PORT
988 03E7 E4 64 AND AL,INPT_BUF_FULL ; TEST INPUT BUFFER FULL FLAG (BIT 1)
989 03E9 24 02 LOOPNZ EMPTY_L ; LOOP UNTIL BUFFER EMPTY OR TIME OUT
990 03EB E0 FA RET
991 03ED C3
992 03EE GATE_A20 ENDP
993
994
995 ;----- INT 15H -- ( FUNCTION 88H - I/O MEMORY SIZE DETERMINE )
996 ;----- EXT_MEMORY
997 ;----- THIS ROUTINE RETURNS THE AMOUNT OF MEMORY IN THE SYSTEM THAT IS
998 ;----- LOCATED STARTING AT THE 1024K ADDRESSING RANGE, AS DETERMINED BY
999 ;----- THE 8042 STATUS PORT.
1000 ;----- NOTE THAT THE SYSTEM MAY NOT BE ABLE TO USE I/O MEMORY UNLESS THERE
1001 ;----- IS A FULL COMPLEMENT OF 512K OR 640 BYTES ON THE PLANAR. THIS SIZE
1002 ;----- SIZE IS STORED IN CMOS AT ADDRESS LOCATIONS 30H AND 31H.
1003
1004 ;----- INPUT
1005 ;----- AH = 88H
1006
1007 ;----- THE I/O MEMORY SIZE VARIABLE IS SET DURING POWER ON
1008 ;----- DIAGNOSTICS ACCORDING TO THE FOLLOWING ASSUMPTIONS:
1009
1010 ;----- 1. ALL INSTALLED MEMORY IS FUNCTIONAL.
1011 ;----- 2. ALL MEMORY FROM 0 TO 640K MUST BE CONTIGUOUS.
1012
1013 ;----- OUTPUT
1014 ;----- (AX) = NUMBER OF CONTIGUOUS 1K BLOCKS OF MEMORY A
1015 ;----- AVAILABLE STARTING AT ADDRESS 1024K.
1016
1017
1018 03EE EXT_MEMORY PROC
1019
1020 03EE B8 3031 MOV AX,CMOS_U_M_S_LO*H+CMOS_U_M_S_HI ; ADDRESS HIGH/LOW BYTES
1021 03F1 E8 0000 E CALL CMOS_READ ; GET THE HIGH BYTE OF I/O MEMORY
1022 03F4 86 C4 XCHG AL, AH ; PUT HIGH BYTE IN POSITION (AH)
1023 03F6 E8 0000 E CALL CMOS_READ ; GET THE LOW BYTE OF I/O MEMORY
1024 03F9 CF IRET ; RETURN TO USER
1025
1026 03FA EXT_MEMORY ENDP

```

```

1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134

PAGE
---- INT 15H ( FUNCTION 89H )
PURPOSE:
THIS BIOS FUNCTION PROVIDES A MEANS TO THE USER TO SWITCH INTO
VIRTUAL (PROTECTED) MODE. UPON COMPLETION OF THIS FUNCTION THE
PROCESSOR WILL BE IN VIRTUAL (PROTECTED) MODE AND CONTROL WILL
BE TRANSFERRED TO THE CODE SEGMENT THAT WAS SPECIFIED BY THE USER.

ENTRY REQUIREMENTS:
(ES:SI) POINTS TO A DESCRIPTOR TABLE (GDT) BUILT BEFORE INTERRUPTING
TO THIS FUNCTION. THESE DESCRIPTORS ARE USED BY THIS FUNCTION TO
INITIALIZE THE IDTR, THE GDTR AND THE STACK SEGMENT SELECTOR. THE
DATA SEGMENT (DS) SELECTOR AND THE EXTRA SEGMENT (ES) SELECTOR WILL
BE INITIALIZED BY THIS FUNCTION BUT THE USER MUST SPECIFY THE SAME
BH - OFFSET INTO THE INTERRUPT DESCRIPTOR TABLE STATING WHERE THE
FIRST EIGHT HARDWARE INTERRUPTS WILL BEGIN. ( INTERRUPT LEVEL 1 )
BL - OFFSET INTO THE INTERRUPT DESCRIPTOR TABLE STATING WHERE THE
SECOND EIGHT HARDWARE INTERRUPTS BEGIN. ( INTERRUPT LEVEL 2 )

THE DESCRIPTORS ARE DEFINED AS FOLLOWS:
1. THE FIRST DESCRIPTOR IS THE REQUIRED DUMMY.
    (USER INITIALIZED TO 0)
2. THE SECOND DESCRIPTOR POINTS TO THE GDT TABLE AS
    A DATA SEGMENT.
    (USER INITIALIZED)
3. THE THIRD DESCRIPTOR POINTS TO THE USER DEFINED
    INTERRUPT DESCRIPTOR TABLE (IDT).
    (USER INITIALIZED)
4. THE FORTH DESCRIPTOR POINTS TO THE USER'S DATA
    SEGMENT (DS).
    (USER INITIALIZED)
5. THE FIFTH DESCRIPTOR POINTS TO THE USER'S EXTRA
    SEGMENT (ES).
    (USER INITIALIZED)
6. THE SIXTH DESCRIPTOR POINTS TO THE USER'S STACK
    SEGMENT (SS).
    (USER INITIALIZED)
7. THE SEVENTH DESCRIPTOR POINTS TO THE CODE SEGMENT
    THAT THIS FUNCTION WILL RETURN TO.
    (USER INITIALIZED TO THE USER'S CODE SEGMENT.)
8. THE EIGHTH DESCRIPTOR IS USED BY THIS FUNCTION TO
    ESTABLISH A CODE SEGMENT FOR ITSELF. THIS IS
    NEEDED SO THAT THIS FUNCTION CAN COMPLETE IT'S
    OPERATION WHILE IN PROTECTED MODE. WHEN CONTROL
    GETS PASSED TO THE USER'S CODE THIS DESCRIPTOR CAN
    BE USED BY HIM IN ANY WAY HE CHOOSES.

NOTE - EACH DESCRIPTOR MUST CONTAIN ALL THE NECESSARY DATA
I.E. THE LIMIT, BASE ADDRESS AND THE ACCESS RIGHTS BYTE.

Aha: 89H (FUNCTION CALL)
ES:SI = LOCATION OF THE GDT TABLE BUILD BY ROUTINE
USING THIS FUNCTION.

EXIT PARAMETERS:
AH = 0 IF SUCCESSFUL
ALL SEGMENT REGISTERS ARE CHANGED, (AX) AND (BP) DESTROYED

CONSIDERATIONS:
1. NO BIOS AVAILABLE TO USER. USER MUST HANDLE ALL
    I/O COMMANDS.
2. INTERRUPTS - INTERRUPT VECTOR LOCATIONS MUST BE
    MOVED, DUE TO THE 286 RESERVED AREAS. THE
    HARDWARE INTERRUPT CONTROLLERS MUST BE REINITIALIZED
    TO DEFINE LOCATIONS THAT DO NOT RESIDE IN THE 286
    RESERVED AREAS.
3. EXCEPTION INTERRUPT TABLE AND HANDLER MUST BE
    INITIALIZED BY THE USER.
4. THE INTERRUPT DESCRIPTOR TABLE MUST NOT OVERLAP
    THE REAL MODE BIOS INTERRUPT DESCRIPTOR TABLE.
5. THE FOLLOWING GIVES AN IDEA OF WHAT THE USER CODE
    SHOULD LOOK LIKE WHEN INVOKING THIS FUNCTION.

REAL MODE ---> "USER CODE"
    . MOV AX,GDT SEGMENT
    . MOV ES,AX
    . MOV SI,GDT OFFSET
    . MOV BH,HARDWARE INT LEVEL 1 OFFSET
    . MOV BL,HARDWARE INT LEVEL 2 OFFSET
    . MOV AH,89H
    . INT 15H

VIRTUAL MODE ---> "USER CODE"

DESCRIPTION:
1. CLI (NO INTERRUPTS ALLOWED) WHILE THIS FUNCTION IS EXECUTING.
2. ADDRESS LINE 20 IS GATED ACTIVE.
3. THE CURRENT USER STACK SEGMENT DESCRIPTOR IS INITIALIZED.
4. THE GDTR IS LOADED WITH THE GDT BASE ADDRESS.
5. THE DS SELECTOR IS LOADED WITH THE USER DEFINED
6. THE 8259 IS REINITIALIZED WITH THE NEW INTERRUPT OFFSETS.
7. THE PROCESSOR IS PUT IN VIRTUAL MODE WITH THE CODE
    SEGMENT DESIGNATED FOR THIS FUNCTION.
8. DATA SEGMENT IS LOADED WITH THE USER DEFINED
    SELECTOR FOR THE DS REGISTER.
9. EXTRA SEGMENT IS LOADED WITH THE USER DEFINED
    SELECTOR FOR THE ES REGISTER.
10. STACK SEGMENT IS LOADED WITH THE USER DEFINED
    SELECTOR FOR THE SS REGISTER.
11. CODE SEGMENT DESCRIPTOR SELECTOR VALUE IS
    SUBSTITUTED ON THE STACK FOR RETURN TO USER.
12. WE TRANSFER CONTROL TO THE USER WITH INTERRUPTS DISABLED.

```



```

1249
1250 0442 C7 44 38 FFFF
1251 0447 C6 44 3C 0F
1252 044B C7 44 3A 0000
1253 0450 C6 44 3D 98
1254 0454 C7 44 3E 0000
1255
1256
1257
1258
1259
1260 0459 0F
1261 045A
1262 045A BB 54 08
1263 045D
1264 045A
1265 045E 01
1266 045D 01
1267
1268 045D 0F
1269 045E
1270 045E BB 5C 10
1271 045E
1272 045E
1273 045E 01
1274 0461
1275
1276 0461 BB 0001
1277
1278 0464 0F 01 F0
1279 0467 EA
1280 0468 046C R
1281 046A 0038
1282
1283 046C
1284
1285
1286
1287 046C BB 0018
1288 046F 8E D8
1289 0471 BB 0020
1290 0474 8E 00
1291 0476 BB 0028
1292 0479 8E D0
1293
1294
1295
1296
1297 0478 5B
1298 047C B3 C4 04
1299 047F 6A 30
1300 0481 53
1301 0482 CB
1302
1303 0483
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313 0483 F8
1314 0484 E9 0057 R
1316 0487
1317
1318 0487
1319 0487 CF
1320 0488
1321
1322 0488
1323

1249
1250 0442 C7 44 38 FFFF
1251 0447 C6 44 3C 0F
1252 044B C7 44 3A 0000
1253 0450 C6 44 3D 98
1254 0454 C7 44 3E 0000
1255
1256
1257
1258
1259
1260 0459 0F
1261 045A
1262 045A BB 54 08
1263 045D
1264 045A
1265 045E 01
1266 045D 01
1267
1268 045D 0F
1269 045E
1270 045E BB 5C 10
1271 045E
1272 045E
1273 045E 01
1274 0461
1275
1276 0461 BB 0001
1277
1278 0464 0F 01 F0
1279 0467 EA
1280 0468 046C R
1281 046A 0038
1282
1283 046C
1284
1285
1286
1287 046C BB 0018
1288 046F 8E D8
1289 0471 BB 0020
1290 0474 8E 00
1291 0476 BB 0028
1292 0479 8E D0
1293
1294
1295
1296
1297 0478 5B
1298 047C B3 C4 04
1299 047F 6A 30
1300 0481 53
1301 0482 CB
1302
1303 0483
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313 0483 F8
1314 0484 E9 0057 R
1316 0487
1317
1318 0487
1319 0487 CF
1320 0488
1321
1322 0488
1323

;-----[ENABLE PROTECTED MODE]-----;
MOV [SI].BIO_CS.SEG LIMIT,MAX SEG LEN ; SET LENGTH
MOV [SI].BIO_CS.BASE_HI_BYTE,CSEG@HI ; SET HIGH BYTE OF CS=0F
MOV [SI].BIO_CS.BASE_LO_WORD,CSEG@LO ; SET LOW WORD OF CS=0
MOV [SI].BIO_CS.DATA_ACC RIGHTS,CPL0_CODE_ACCESS
MOV [SI].BIO_CS.DATA_RESERVED,0 ; ZERO RESERVED AREA

;-----[LOAD GLOBAL DESCRIPTOR TABLE REGISTER]-----;
LGDTR [SI].GDTPTR ; LOAD GLOBAL DESCRIPTOR TABLE REGISTER
DB 00H
+ ??0005 LABEL BYTE
MOV DX,WORD PTR [SI].GDTPTR
+ ??0006 LABEL BYTE
ORG ??0005
DB 00H
+ ??0007 LABEL BYTE
ORG ??0006
LIDT [SI].IDTPTR ; INTERRUPT DESCRIPTOR TABLE REGISTER
DB 0FH
+ ??0007 LABEL BYTE
MOV BX,WORD PTR [SI].IDTPTR
+ ??0008 LABEL BYTE
ORG ??0007
DB 001H
+ ??0008 LABEL BYTE
ORG ??0008
DB 00FH,001H,0FH ; PURGE PRE-FETCH QUEUE WITH FAR JUMP
DB 0EAH ; - TO OFFSET
DW OFFSET VMODE ; - IN SEGMENT -PROTECTED MODE SELECTOR
DW BIO_CS

;-----[SETUP USER SEGMENT REGISTERS]-----;
VMODE:
MOV AX,VIRTUAL_ENABLE ; MACHINE STATUS WORD NEEDED TO
LMSW AX ; SWITC TO VIRTUAL MODE
MOV DS,AX
MOV AX,USER_ES ; SETUP USER'S DATA SEGMENT
MOV ES,ES
MOV AX,USER_SS ; SETUP USER'S EXTRA SEGMENT
MOV SS,AX ; SETUP USER'S STACK SEGMENT

;-----[PUT TRANSFER ADDRESS ON STACK]-----;
;-----[AND RETURN TO THE USER]-----;
POP BX ; GET RETURN IP FROM THE STACK
ADD SP,4 ; NORMALIZE STACK POINTER
PUSH USER_CS ; SET STACK FOR A RETURN FAR
PUSH BX
RET ; RETURN TO USER IN VIRTUAL MODE

X_VIRTUAL ENDP

;-----[DEVICE BUSY AND INTERRUPT COMPLETE]-----;
;-----[THIS ROUTINE IS A TEMPORARY HANDLER FOR DEVICE BUSY AND INTERRUPT COMPLETE]-----;
;-----[INPUT - SEE PROLOGUE]-----;

DEVICE_BUSY PROC NEAR
CLC ; TURN CARRY OFF
CMP C1_F ; RETURN WITH CARRY FLAG
DEVICE_BUSY ENDP

INT_COMPLETE PROC NEAR
IRET ; RETURN
INT_COMPLETE ENDP

CODE ENDS
END

```



```

PAGE 14
105 0031 A0 0070 R
116 0031 A0 0070 R 00
117 0034 C6 06 0070 R 00
118 0039 BB 0E 006E R
119 003D BB 16 006C R
120 0041 C3
121
122 0042
123 0042 B9 16 006C R
124 0044 B9 0E 006E R
125 0044 C6 06 0070 R 00
126 004F C3
127
128 0050
129 0054 E8 016B R
130 0053 72 1F
131
132 0055 B0 00
133 0057 E8 0000 E
134 0054 A8 F0
135 0056 B0 0B
136 0056 E8 0000 E
137 0061 24 01
138 0063 8A D0
139 0063 B0 02
140 0067 E8 0000 E
141 0064 B8 C8
142 0064 E8 0000 E
143 0066 E8 0000 E
144 0071 8A E8
145 0073 F8
146 0074
147 0074 C3
148
149 0075
150 0075 E8 016B R
151 0074 73 03
152 0074 E8 0154 R
153 0076
154 0070 8A E6
155 0070 FB 00
156 0081 E8 0000 E
157 0084 A8 E1
158 0086 B0 02
159 0087 E8 0000 E
160 0088 B8 E5
161 0080 B0 04
162 008F E8 0000 E
163 0092 BB 0B0B
164 0095 E8 0000 E
165 0096 0A C7
166 0094 0C 02
167 009C 80 E2 01
168 0099 0A C2
169 00A1 86 E0
170 00A4 E8 0000 E
171 00A5 F8
172 00A7 C3
173
174 00A8
175 00A8 E8 016B R
176 00A2 72 1D
177
178 00AD B0 07
179 00AE E8 0000 E
180 00B2 B8 D0
181 00B4 B0 08
182 00B5 E8 0000 E
183 00B9 8A F0
184 00B9 B0 09
185 00B8 E8 0000 E
186 00C4 A8 C8
187 00C2 B0 32
188 00C2 E8 0000 E
189 00C7 8A E8
190 00C9 F8
191 00CA
192 00CA C3
193
194 00CB
195 00CB E8 016B R
196 00C6 73 03
197 00D4 E8 0154 R
198 00D3
199 00D3 B8 0006
200 00D3 E8 0000 E
201 00D9 8A E2
202 00D9 B0 07
203 00DE E8 0000 E
204 00E0 A8 E6
205 00E0 B0 08
206 00E4 E8 0000 E
207 00E7 8A E1
208 00E9 B0 09
209 00E9 E8 0000 E
210 00E9 A8 E5
211 00E9 B0 32
212 00F5 BB 0B0B
213 00F5 BB 0B0B
214 00F8 E8 0000 E
215 00F8 24 1F
216 00FD B8 E0
217 00FD E8 0000 E
218 0102 F8
219 0103 C3
220
221 0104
222 0104 B0 0B
223 0106 E8 0000 E
224 0108 A8 20
225 0108 F9
226 0105 T5 33
227
228 0106
229 0106 E8 0000 E
230 0108 A8 20
231 0108 F9
232 0105 T5 33
233
234 0106
235 0106 E8 0000 E
236 0108 A8 20
237 0108 F9
238 0105 T5 33
239
240 0106
241 0106 E8 0000 E
242 0108 A8 20
243 0108 F9
244 0105 T5 33
245
246 0106
247 0106 E8 0000 E
248 0108 A8 20
249 0108 F9
250 0105 T5 33
251
252 0106
253 0106 E8 0000 E
254 0108 A8 20
255 0108 F9
256 0105 T5 33
257
258 0106
259 0106 E8 0000 E
260 0108 A8 20
261 0108 F9
262 0105 T5 33
263
264 0106
265 0106 E8 0000 E
266 0108 A8 20
267 0108 F9
268 0105 T5 33
269
270 0106
271 0106 E8 0000 E
272 0108 A8 20
273 0108 F9
274 0105 T5 33
275
276 0106
277 0106 E8 0000 E
278 0108 A8 20
279 0108 F9
280 0105 T5 33
281
282 0106
283 0106 E8 0000 E
284 0108 A8 20
285 0108 F9
286 0105 T5 33
287
288 0106
289 0106 E8 0000 E
290 0108 A8 20
291 0108 F9
292 0105 T5 33
293
294 0106
295 0106 E8 0000 E
296 0108 A8 20
297 0108 F9
298 0105 T5 33
299
200 0106
201 0106 E8 0000 E
202 0108 A8 20
203 0108 F9
204 0105 T5 33
205
206 0106
207 0106 E8 0000 E
208 0108 A8 20
209 0108 F9
210 0105 T5 33
211
212 0106
213 0106 E8 0000 E
214 0108 A8 20
215 0108 F9
216 0105 T5 33
217
218 0106
219 0106 E8 0000 E
220 0108 A8 20
221 0108 F9
222 0105 T5 33
223
224 0106
225 0106 E8 0000 E
226 0108 A8 20
227 0108 F9
228 0105 T5 33
229
230 0106
231 0106 E8 0000 E
232 0108 A8 20
233 0108 F9
234 0105 T5 33
235
236 0106
237 0106 E8 0000 E
238 0108 A8 20
239 0108 F9
240 0105 T5 33
241
242 0106
243 0106 E8 0000 E
244 0108 A8 20
245 0108 F9
246 0105 T5 33
247
248 0106
249 0106 E8 0000 E
250 0108 A8 20
251 0108 F9
252 0105 T5 33
253
254 0106
255 0106 E8 0000 E
256 0108 A8 20
257 0108 F9
258 0105 T5 33
259
260 0106
261 0106 E8 0000 E
262 0108 A8 20
263 0108 F9
264 0105 T5 33
265
266 0106
267 0106 E8 0000 E
268 0108 A8 20
269 0108 F9
270 0105 T5 33
271
272 0106
273 0106 E8 0000 E
274 0108 A8 20
275 0108 F9
276 0105 T5 33
277
278 0106
279 0106 E8 0000 E
280 0108 A8 20
281 0108 F9
282 0105 T5 33
283
284 0106
285 0106 E8 0000 E
286 0108 A8 20
287 0108 F9
288 0105 T5 33
289
290 0106
291 0106 E8 0000 E
292 0108 A8 20
293 0108 F9
294 0105 T5 33
295
296 0106
297 0106 E8 0000 E
298 0108 A8 20
299 0108 F9
200 0105 T5 33
201
202 0106
203 0106 E8 0000 E
204 0108 A8 20
205 0108 F9
206 0105 T5 33
207
208 0106
209 0106 E8 0000 E
210 0108 A8 20
211 0108 F9
212 0105 T5 33
213
214 0106
215 0106 E8 0000 E
216 0108 A8 20
217 0108 F9
218 0105 T5 33
219
220 0106
221 0106 E8 0000 E
222 0108 A8 20
223 0108 F9
224 0105 T5 33
225
226 0106
227 0106 E8 0000 E
228 0108 A8 20
229 0108 F9
230 0105 T5 33
231
232 0106
233 0106 E8 0000 E
234 0108 A8 20
235 0108 F9
236 0105 T5 33
237
238 0106
239 0106 E8 0000 E
240 0108 A8 20
241 0108 F9
242 0105 T5 33
243
244 0106
245 0106 E8 0000 E
246 0108 A8 20
247 0108 F9
248 0105 T5 33
249
250 0106
251 0106 E8 0000 E
252 0108 A8 20
253 0108 F9
254 0105 T5 33
255
256 0106
257 0106 E8 0000 E
258 0108 A8 20
259 0108 F9
260 0105 T5 33
261
262 0106
263 0106 E8 0000 E
264 0108 A8 20
265 0108 F9
266 0105 T5 33
267
268 0106
269 0106 E8 0000 E
270 0108 A8 20
271 0108 F9
272 0105 T5 33
273
274 0106
275 0106 E8 0000 E
276 0108 A8 20
277 0108 F9
278 0105 T5 33
279
280 0106
281 0106 E8 0000 E
282 0108 A8 20
283 0108 F9
284 0105 T5 33
285
286 0106
287 0106 E8 0000 E
288 0108 A8 20
289 0108 F9
290 0105 T5 33
291
292 0106
293 0106 E8 0000 E
294 0108 A8 20
295 0108 F9
296 0105 T5 33
297
298 0106
299 0106 E8 0000 E
300 0108 A8 20
301 0108 F9
302 0105 T5 33
303
304 0106
305 0106 E8 0000 E
306 0108 A8 20
307 0108 F9
308 0105 T5 33
309
310 0106
311 0106 E8 0000 E
312 0108 A8 20
313 0108 F9
314 0105 T5 33
315
316 0106
317 0106 E8 0000 E
318 0108 A8 20
319 0108 F9
320 0105 T5 33
321
322 0106
323 0106 E8 0000 E
324 0108 A8 20
325 0108 F9
326 0105 T5 33
327
328 0106
329 0106 E8 0000 E
330 0108 A8 20
331 0108 F9
332 0105 T5 33
333
334 0106
335 0106 E8 0000 E
336 0108 A8 20
337 0108 F9
338 0105 T5 33
339
340 0106
341 0106 E8 0000 E
342 0108 A8 20
343 0108 F9
344 0105 T5 33
345
346 0106
347 0106 E8 0000 E
348 0108 A8 20
349 0108 F9
350 0105 T5 33
351
352 0106
353 0106 E8 0000 E
354 0108 A8 20
355 0108 F9
356 0105 T5 33
357
358 0106
359 0106 E8 0000 E
360 0108 A8 20
361 0108 F9
362 0105 T5 33
363
364 0106
365 0106 E8 0000 E
366 0108 A8 20
367 0108 F9
368 0105 T5 33
369
370 0106
371 0106 E8 0000 E
372 0108 A8 20
373 0108 F9
374 0105 T5 33
375
376 0106
377 0106 E8 0000 E
378 0108 A8 20
379 0108 F9
380 0105 T5 33
381
382 0106
383 0106 E8 0000 E
384 0108 A8 20
385 0108 F9
386 0105 T5 33
387
388 0106
389 0106 E8 0000 E
390 0108 A8 20
391 0108 F9
392 0105 T5 33
393
394 0106
395 0106 E8 0000 E
396 0108 A8 20
397 0108 F9
398 0105 T5 33
399
400 0106
401 0106 E8 0000 E
402 0108 A8 20
403 0108 F9
404 0105 T5 33
405
406 0106
407 0106 E8 0000 E
408 0108 A8 20
409 0108 F9
410 0105 T5 33
411
412 0106
413 0106 E8 0000 E
414 0108 A8 20
415 0108 F9
416 0105 T5 33
417
418 0106
419 0106 E8 0000 E
420 0108 A8 20
421 0108 F9
422 0105 T5 33
423
424 0106
425 0106 E8 0000 E
426 0108 A8 20
427 0108 F9
428 0105 T5 33
429
430 0106
431 0106 E8 0000 E
432 0108 A8 20
433 0108 F9
434 0105 T5 33
435
436 0106
437 0106 E8 0000 E
438 0108 A8 20
439 0108 F9
440 0105 T5 33
441
442 0106
443 0106 E8 0000 E
444 0108 A8 20
445 0108 F9
446 0105 T5 33
447
448 0106
449 0106 E8 0000 E
450 0108 A8 20
451 0108 F9
452 0105 T5 33
453
454 0106
455 0106 E8 0000 E
456 0108 A8 20
457 0108 F9
458 0105 T5 33
459
460 0106
461 0106 E8 0000 E
462 0108 A8 20
463 0108 F9
464 0105 T5 33
465
466 0106
467 0106 E8 0000 E
468 0108 A8 20
469 0108 F9
470 0105 T5 33
471
472 0106
473 0106 E8 0000 E
474 0108 A8 20
475 0108 F9
476 0105 T5 33
477
478 0106
479 0106 E8 0000 E
480 0108 A8 20
481 0108 F9
482 0105 T5 33
483
484 0106
485 0106 E8 0000 E
486 0108 A8 20
487 0108 F9
488 0105 T5 33
489
490 0106
491 0106 E8 0000 E
492 0108 A8 20
493 0108 F9
494 0105 T5 33
495
496 0106
497 0106 E8 0000 E
498 0108 A8 20
499 0108 F9
500 0105 T5 33
501
502 0106
503 0106 E8 0000 E
504 0108 A8 20
505 0108 F9
506 0105 T5 33
507
508 0106
509 0106 E8 0000 E
510 0108 A8 20
511 0108 F9
512 0105 T5 33
513
514 0106
515 0106 E8 0000 E
516 0108 A8 20
517 0108 F9
518 0105 T5 33
519
520 0106
521 0106 E8 0000 E
522 0108 A8 20
523 0108 F9
524 0105 T5 33
525
526 0106
527 0106 E8 0000 E
528 0108 A8 20
529 0108 F9
530 0105 T5 33
531
532 0106
533 0106 E8 0000 E
534 0108 A8 20
535 0108 F9
536 0105 T5 33
537
538 0106
539 0106 E8 0000 E
540 0108 A8 20
541 0108 F9
542 0105 T5 33
543
544 0106
545 0106 E8 0000 E
546 0108 A8 20
547 0108 F9
548 0105 T5 33
549
550 0106
551 0106 E8 0000 E
552 0108 A8 20
553 0108 F9
554 0105 T5 33
555
556 0106
557 0106 E8 0000 E
558 0108 A8 20
559 0108 F9
560 0105 T5 33
561
562 0106
563 0106 E8 0000 E
564 0108 A8 20
565 0108 F9
566 0105 T5 33
567
568 0106
569 0106 E8 0000 E
570 0108 A8 20
571 0108 F9
572 0105 T5 33
573
574 0106
575 0106 E8 0000 E
576 0108 A8 20
577 0108 F9
578 0105 T5 33
579
580 0106
581 0106 E8 0000 E
582 0108 A8 20
583 0108 F9
584 0105 T5 33
585
586 0106
587 0106 E8 0000 E
588 0108 A8 20
589 0108 F9
590 0105 T5 33
591
592 0106
593 0106 E8 0000 E
594 0108 A8 20
595 0108 F9
596 0105 T5 33
597
598 0106
599 0106 E8 0000 E
600 0108 A8 20
601 0108 F9
602 0105 T5 33
603
604 0106
605 0106 E8 0000 E
606 0108 A8 20
607 0108 F9
608 0105 T5 33
609
610 0106
611 0106 E8 0000 E
612 0108 A8 20
613 0108 F9
614 0105 T5 33
615
616 0106
617 0106 E8 0000 E
618 0108 A8 20
619 0108 F9
620 0105 T5 33
621
622 0106
623 0106 E8 0000 E
624 0108 A8 20
625 0108 F9
626 0105 T5 33
627
628 0106
629 0106 E8 0000 E
630 0108 A8 20
631 0108 F9
632 0105 T5 33
633
634 0106
635 0106 E8 0000 E
636 0108 A8 20
637 0108 F9
638 0105 T5 33
639
640 0106
641 0106 E8 0000 E
642 0108 A8 20
643 0108 F9
644 0105 T5 33
645
646 0106
647 0106 E8 0000 E
648 0108 A8 20
649 0108 F9
650 0105 T5 33
651
652 0106
653 0106 E8 0000 E
654 0108 A8 20
655 0108 F9
656 0105 T5 33
657
658 0106
659 0106 E8 0000 E
660 0108 A8 20
661 0108 F9
662 0105 T5 33
663
664 0106
665 0106 E8 0000 E
666 0108 A8 20
667 0108 F9
668 0105 T5 33
669
670 0106
671 0106 E8 0000 E
672 0108 A8 20
673 0108 F9
674 0105 T5 33
675
676 0106
677 0106 E8 0000 E
678 0108 A8 20
679 0108 F9
680 0105 T5 33
681
682 0106
683 0106 E8 0000 E
684 0108 A8 20
685 0108 F9
686 0105 T5 33
687
688 0106
689 0106 E8 0000 E
690 0108 A8 20
691 0108 F9
692 0105 T5 33
693
694 0106
695 0106 E8 0000 E
696 0108 A8 20
697 0108 F9
698 0105 T5 33
699
700 0106
701 0106 E8 0000 E
702 0108 A8 20
703 0108 F9
704 0105 T5 33
705
706 0106
707 0106 E8 0000 E
708 0108 A8 20
709 0108 F9
710 0105 T5 33
711
712 0106
713 0106 E8 0000 E
714 0108 A8 20
715 0108 F9
716 0105 T5 33
717
718 0106
719 0106 E8 0000 E
720 0108 A8 20
721 0108 F9
722 0105 T5 33
723
724 0106
725 0106 E8 0000 E
726 0108 A8 20
727 0108 F9
728 0105 T5 33
729
730 0106
731 0106 E8 0000 E
732 0108 A8 20
733 0108 F9
734 0105 T5 33
735
736 0106
737 0106 E8 0000 E
738 0108 A8 20
739 0108 F9
740 0105 T5 33
741
742 0106
743 0106 E8 0000 E
744 0108 A8 20
745 0108 F9
746 0105 T5 33
747
748 0106
749 0106 E8 0000 E
750 0108 A8 20
751 0108 F9
752 0105 T5 33
753
754 0106
755 0106 E8 0000 E
756 0108 A8 20
757 0108 F9
758 0105 T5 33
759
760 0106
761 0106 E8 0000 E
762 0108 A8 20
763 0108 F9
764 0105 T5 33
765
766 0106
767 0106 E8 0000 E
768 0108 A8 20
769 0108 F9
770 0105 T5 33
771
772 0106
773 0106 E8 0000 E
774 0108 A8 20
775 0108 F9
776 0105 T5 33
777
778 0106
779 0106 E8 0000 E
780 0108 A8 20
781 0108 F9
782 0105 T5 33
783
784 0106
785 0106 E8 0000 E
786 0108 A8 20
787 0108 F9
788 0105 T5 33
789
790 0106
791 0106 E8 0000 E
792 0108 A8 20
793 0108 F9
794 0105 T5 33
795
796 0106
797 0106 E8 0000 E
798 0108 A8 20
799 0108 F9
800 0105 T5 33
801
802 0106
803 0106 E8 0000 E
804 0108 A8 20
805 0108 F9
806 0105 T5 33
807
808 0106
809 0106 E8 0000 E
810 0108 A8 20
811 0108 F9
812 0105 T5 33
813
814 0106
815 0106 E8 0000 E
816 0108 A8 20
817 0108 F9
818 0105 T5 33
819
820 0106
821 0106 E8 0000 E
822 0108 A8 20
823 0108 F9
824 0105 T5 33
825
826 0106
827 0106 E8 0000 E
828 0108 A8 20
829 0108 F9
830 0105 T5 33
831
832 0106
833 0106 E8 0000 E
834 0108 A8 20
835 0108 F9
836 0105 T5 33
837
838 0106
839 0106 E8 0000 E
840 0108 A8 20
841 0108 F9
842 0105 T5 33
843
844 0106
845 0106 E8 0000 E
846 0108 A8 20
847 0108 F9
848 0105 T5 33
849
850 0106
851 0106 E8 0000 E
852 0108 A8 20
853 0108 F9
854 0105 T5 33
855
856 0106
857 0106 E8 0000 E
858 0108 A8 20
859 0108 F9
860 0105 T5 33
861
862 0106
863 0106 E8 0000 E
864 0108 A8 20
865 0108 F9
866 0105 T5 33
867
868 0106
869 0106 E8 0000 E
870 0108 A8 20
871 0108 F9
872 0105 T5 33
873
874 0106
875 0106 E8 0000 E
876 0108 A8 20
877 0108 F9
878 0105 T5 33
879
880 0106
881 0106 E8 0000 E
882 0108 A8 20
883 0108 F9
884 0105 T5 33
885
886 0106
887 0106 E8 0000 E
888 0108 A8 20
889 0108 F9
890 0105 T5 33
891
892 0106
893 0106 E8 0000 E
894 0108 A8 20
895 0108 F9
896 0105 T5 33
897
898 0106
899 0106 E8 0000 E
900 0108 A8 20
901 0108 F9
902 0105 T5 33
903
904 0106
905 0106 E8 0000 E
906 0108 A8 20
907 0108 F9
908 0105 T5 33
909
910 0106
911 0106 E8 0000 E
912 0108 A8 20
913 0108 F9
914 0105 T5 33
915
916 0106
917 0106 E8 0000 E
918 0108 A8 20
919 0108 F9
920 0105 T5 33
921
922 0106
923 0106 E8 0000 E
924 0108 A8 20
925 0108 F9
926 0105 T5 33
927
928 0106
929 0106 E8 0000 E
930 0108 A8 20
931 0108 F9
932 0105 T5 33
933
934 0106
935 0106 E8 0000 E
936 0108 A8 20
937 0108 F9
938 0105 T5 33
939
940 0106
941 0106 E8 0000 E
942 0108 A8 20
943 0108 F9
944 0105 T5 33
945
946 0106
947 0106 E8 0000 E
948 0108 A8 20
949 0108 F9
950 0105 T5 33
951
952 0106
953 0106 E8 0000 E
954 0108 A8 20
955 0108 F9
956 0105 T5 33
957
958 0106
959 0106 E8 0000 E
960 0108 A8 20
961 0108 F9
962 0105 T5 33
963
964 0106
965 0106 E8 0000 E
966 0108 A8 20
967 0108 F9
9
```

```

228 010 E6 16B R CALL UPD_IPR : CHECK FOR UPDATE IN PROCESS
229 0111 73 03 JNC RTC_65 : SKIP INITIALIZATION IF NO ERROR
230 0113 E6 0154 R CALL RTC_STA : ELSE INITIALIZE CLOCK
231

RTC_65: MOV AH,DH : GET SECONDS BYTE
        MOV AL,CMOS_SEC_ALARM : ADDRESS THE SECONDS ALARM REGISTER
        CALL CMOS_WRITE : INSERT SECONDS
        MOV AH,CL : GET MINUTES PARAMETER
        MOV AL,CMOS_MIN_ALARM : ADDRESS MINUTES ALARM REGISTER
        CALL CMOS_WRITE : INSERT MINUTES
        MOV AH,CH : GET HOURS PARAMETER
        MOV AL,CMOS_HR_ALARM : ADDRESS HOUR ALARM REGISTER
        CALL CMOS_WRITE : INSERT HOURS
        IN AL,INTBO1 : READ SECOND INTERRUPT MASK REGISTER
        AND AL,0FEH : ENABLE ALARM TIMER BIT (CY=0)
        OUT INTBO1,AL : WRITE UPDATED MASK
        MOV AX,X*CMOS_REG_B : ADDRESS ALARM REGISTER
        CALL CMOS_READ : READ CURRENT ALARM REGISTER
        AND AL,07FH : ENSURE SET BIT TURNED OFF
        OR AL,20H : TURN ON ALARM ENABLE
        CALL CMOS_WRITE : MOVE MASK TO OUTPUT REGISTER
        XCHG AH,AL : WRITE NEW ALARM MASK
        CLC : SET CY=0
232 0116 8A E6 CALL RTC_69: : RETURN WITH CY FLAG
233 0118 80 01
234 011A E6 0000 E
235 011D 8A E1
236 011F 80 03
237 0121 E6 0000 E
238 0124 8A E5
239 0126 80 05
240 0128 E6 0000 E
241 012B E4 A1
242 012D 24 FE
243 012F E6 A1
244 0131 BB 0B0B
245 0133 E6 0000 E
246 0137 24 FF
247 0139 DC 20
248 013B E6 0000 E
249 013D E6 0000 E
250 0140 F8
251 0141 C3
252 0141 B8 0000
253 0144 C3
254
255 0145 RTC_70: : CLEAR AX REGISTER
256 0146 B8 0B0B : RETURN WITH RESULTS IN CARRY FLAG
257 0148 E6 0000 E
258 014B 24 57
259 014D B6 E0
260 014F E6 0000 E
261 0152 F8
262 0153 C3
263
264 0154 RTC_00 ENDP : RETURN WITH NO CARRY
265
266 0154 RTC_STA PROC NEAR : INITIALIZE REAL TIME CLOCK
267 0154 B6 260A : ADDRESS REGISTERS A AND LOAD DATA MASK
268 0155 E8 0000 E : INITIALIZATION STATUS REGISTER
269 0155 B8 820B : SET "SET BIT" FOR CLOCK INITIALIZATION
270 0156 E8 0000 E : AND 24 HOUR MODE TO REGISTER B
271 0160 B0 0C : ADDRESS REGISTER C
272 0162 E8 0000 E : READ REGISTER C TO INITIALIZE
273 0163 B0 00 : ADDRESS REGISTER D
274 0167 E6 0000 E : READ REGISTER D TO INITIALIZE
275 0164 C3
276
277 016B RTC_STA ENDP : READ REGISTER D TO INITIALIZE
278
279 016B UPD_IPR PROC NEAR : WAIT TILL UPDATE NOT IN PROGRESS
280 016B B9 0320 : SAVE CALLERS REGISTER
281 016C B9 0320 : SET TIMEOUT LOOP COUNT
282 016F
283 017F B0 0A : ADDRESS STATUS REGISTER A
284 017A FA : NO TIMER SET, SO WAITING UPDATES
285 017D 00 0000 E : READ TIME IN PROCESS, NO AG
286 0175 A8 80 : IF UIP BIT IS ON I CANNOT READ TIME, I
287 0177 T4 06 : EXIT WITH CY=0 IF CAN READ CLOCK NOW
288 0179 FB : ALLOW INTERRUPTS WHILE WAITING
289 017A E2 F3 : LOOP TILL READY OR TIMEOUT
290 017B 93 C0 : CLEAR RESULTS IF ERROR
291 017E F9 : SET CARRY FOR ERROR
292 017F UPD_90: : RESTORE CALLERS REGISTER
293 017F 59 : INTERRUPTS OFF DURING SET
294 0180 FA : RETURN WITH CY FLAG SET
295 0181 C3
296
297 0182 UPD_IPR ENDP

```

```

298
299
300
301
302
303
304
305
306
307
308
309
310
311
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 BB BB8C MOV 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 0186 90 NOP ; I/O DELAY
323 0186 70 71 IN AL,CMOS_DATA ; READ AND RESET INTERRUPT REQUEST FLAGS
324 018D A8 60 TEST AL,01000000B ; CHECK FOR EITHER INTERRUPT PENDING
325 018E 74 4D 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 0192 86 70 OUT CMOS_PORT,AL ; WRITE ALARM ENABLE MASK ADDRESS
329 0195 00 00 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 86 40 TEST AL,01000000B ; CHECK FOR PERIODIC INTERRUPT
333 019B 74 30 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 03D0 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 20 JNC RTC_T_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 BB BB8B MOV AX,X*(CMOS_REG_B+NMI) ; INTERRUPT ENABLE REGISTER
346 01B2 E6 70 OUT CMOS_PORT,AL ; WRITE ADDRESS TO CMOS CLOCK
347 01B3 00 00 NOP ; I/O DELAY
348 01B5 E4 71 IN AL,CMOS_DATA ; READ CURRENT ENABLES
349 01B7 24 BF AND AL,0BFH ; TURN OFF PIE
350 01B9 86 C4 XCHG AL,AH ; GET CMOS ADDRESS AND SAVE VALUE
351 01B9 E6 70 OUT CMOS_PORT,AL ; ADDRESS REGISTER B
352 01BD 86 C4 XCHG AL,AH ; GET NEW INTERRUPT ENABLE MASK
353 01C1 00 00 INT AL,CMOS_DATA,AL ; SET MASK IN INTERRUPT ENABLE REGISTER
354 01C1 C6 06 0040 R 00 MOV #RTC_USER_FLAG,0 ; POINT TO USER FLAG OFF
355 01C6 C5 3E 0098 R LDS D1,DWORD PTR #USER_FLAG ; SET UP (DS:D1) TO POINT TO USER FLAG
356 01C6 C6 05 80 MOV BYTE PTR [D1],80H ; TURN ON USERS FLAG
357 01C6 58 POP AX ; GET INTERRUPT SOURCE BACK
358
359 01CE 86 A2 TEST AL,00010000B ; TEST FOR ALARM INTERRUPT
360 01D0 74 0A JZ RTC_I_7 ; SKIP USER INTERRUPT CALL IF NOT ALARM
361
362 01D2 B0 0D MOV AL,CMOS_REG_D ; POINT TO DEFAULT READ ONLY REGISTER
363 01D4 E6 70 OUT CMOS_PORT,AL ; ENABLE NMI AND CMOS ADDRESS TO DEFAULT
364 01D6 FB STI ; INTERRUPTS BACK ON NOW
365 01D8 00 00 DX
366 01D8 CD 4A INT 4AH ; TRANSFER TO USER ROUTINE
367 01DA 5A POP DX
368 01DB FA CLD ; BLOCK INTERRUPT FOR RETRY
369 01DC CLD ; RESTART ROUTINE TO HANDLE DELAYED
370 01DE EB A7 JMP RTC_I_1 ; ENTRY AND SECOND EVENT BEFORE DONE
371
372
373 01DE RTC_I_9: ; EXIT - NO PENDING INTERRUPTS
374 01DE B0 0D MOV AL,CMOS_REG_D ; POINT TO DEFAULT READ ONLY REGISTER
375 01E0 E6 70 OUT CMOS_PORT,AL ; ENABLE NMI AND CMOS ADDRESS TO DEFAULT
376 01E2 B0 20 MOV AL,E0I ; END OF INTERRUPT MASK TO 8259 - 2
377 01E3 00 00 OUT INTB00,AL ; TO 8259 - 2
378 01E6 E6 20 OUT INTA00,AL ; TO 8259 - 1
379 01E8 5F POP DI ; RESTORE REGISTERS
380 01E9 58 POP AX
381 01EA 1F POP DS
382 01EB CF IRET ; END OF INTERRUPT
383
384 01EC RTC_INT ENDP

```

```

385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403 01EC PRINT_SCREEN_I PROC FAR
404
405 01EC 1E PUSH DS ; DELAY INTERRUPT ENABLE TILL FLAG SET
406 01ED 50 PUSH AX ; SAVE WORK REGISTERS
407 01EE 53 PUSH BX
408 01EF 51 PUSH CX
409 01F0 52 PUSH DX
410 01F1 E8 0000 E CALL DDS ; USE 0040:0100 FOR STATUS AREA STORAGE
411 01F4 80 3E 0100 R 01 CMP *STATUS_BYTE,1 ; GET STATUS BYTE DATA SEGMENT
412 01F5 74 00 JE PR190 ; SEE IF PRINT ALREADY IN PROGRESS
413 01FB C6 06 0100 R 01 MOV *STATUS_BYTE,1 ; EXIT IF PRINT ALREADY IN PROGRESS
414 0200 FB STI ; INDICATE PRINT NOT IN PROGRESS
415 0201 B4 0F MOV AH,0FH ; MUST RUN WITH INTERRUPTS ENABLED
416 0203 CD 10 INT 10H ; WILL REQUEST THE CURRENT SCREEN MODE
417
418
419 0205 8A CC MOV CL, AH ; (AL)= MODE
420 0207 8A 2E 0084 R MOV CH, #ROWS ; (AH)= NUMBER COLUMNS/LINE
421 0208 FE C5 INC CH ; (BH)= VISUAL PAGE
422
423
424
425
426
427
428
429
430 020D 33 D2 XOR DX,DX ; WILL MAKE USE OF (DX) REGISTER TO
431 020F B4 02 MOV AH,02H ; CONTROL ROWS ON SCREEN & COLUMNS
432 0211 CD 17 INT 10H ; ADJUST ROWS ON DISPLAY COUNT
433 0213 B0 F4 80 XOR AH,080H ; (CL)= NUMBER COLUMNS/LINE
434 0216 F6 C4 A0 TEST AH,0AOH ; (CH)= NUMBER OF ROWS ON DISPLAY
435 0219 75 4E JNZ PR180 ; ERROR EXIT IF PRINTER STATUS ERROR
436
437 021B E8 0275 R CALL CRLF ; CARRIAGE RETURN LINE FEED TO PRINTER
438
439 021E 51 PUSH CX ; SAVE SCREEN BOUNDS
440 021F B4 03 MOV AH,03H ; NOW READ THE CURRENT CURSOR POSITION
441 0221 CD 10 INT 10H ; AND RESTORE AT END OF ROUTINE
442 0223 59 XOR AH,080H ; RECALC SCREEN BOUNDS
443 0224 52 PUSH DX ; PRESERVE THE ORIGINAL POSITION
444 0225 33 D2 XOR DX,DX ; INITIAL CURSOR (0,0) AND FIRST PRINTER
445
446
447
448
449 0227 PR110: ; THIS LOOP IS TO READ EACH CURSOR POSITION FROM THE
450 0227 B4 02 MOV AH,02H ; SCREEN AND PRINT IT. (BH)= VISUAL PAGE (CH)= ROWS
451 0229 CD 10 INT 10H ; -
452 022B B4 08 MOV AH,08H ; -
453 022D CD 10 INT 10H ; -
454 022E 00 CD OR AL,AL ; -
455 0231 55 C2 JNZ PR120 ; SEE IF VALID CHAR
456 0233 B0 20 MOV AL,1 ; JUMP IF VALID CHAR
457 0235
458 0235 52 PUSH DX ; ELSE MAKE IT A BLANK
459 0236 33 D2 XOR DX,DX ; -
460 0238 3D E4 MOV AH,0AH ; -
461 023A CD 17 INT 10H ; -
462 023C 5A POP DX ; -
463 023D F6 C4 29 TEST AH,29H ; -
464 0240 75 22 JNZ PR170 ; -
465 0242 FE C2 INC DL ; -
466 0244 3C CA CMP CL,DL ; -
467 0246 45 FF JNZ PR110 ; -
468 0248 32 D2 XOR DL,DL ; -
469 024A 8A E2 MOV AH,DL ; -
470 024C 52 PUSH DX ; -
471 024D E8 0275 R CALL CRLF ; -
472 024E 5A POP DX ; -
473 0251 EE C6 INC DH ; -
474 0253 3A EE CMP CH, DH ; -
475 0255 75 D0 JNZ PR110 ; -
476
477 0257 5A POP DX ; -
478 0258 B4 02 MOV AH,02H ; -
479 025A CD 10 INT 10H ; -
480 025C FA CLI ; -
481 025D C6 06 0100 R 00 MOV *STATUS_BYTE,0 ; -
482 0262 EB 0B JMP SHORT PR190 ; MOVE OK RESULTS FLAG TO STATUS_BYTE
483
484 0264 PR170: ; EXIT PRINTER ROUTINE
485 0264 5A POP DX ; -
486 0265 B4 02 MOV AH,02H ; -
487 0267 CD 10 INT 10H ; -
488 0269 PR180: ; CURSOR POSITION RESTORED
489 0269 FA CLI ; -
490 026A C6 06 0100 R FF MOV *STATUS_BYTE,0FFH ; -
491 026F 5A POP DX ; -
492 0270 59 POP CX ; -
493 0271 5B POP BX ; -
494 0272 58 POP AX ; -
495 0273 1F POP DS ; -
496 0274 CF IRET ; -
497 0275 PRINT_SCREEN_I ENDP ; RETURN WITH INITIAL INTERRUPT MASK

```

```

499 ;----- CARRIAGE RETURN, LINE FEED SUBROUTINE
501
502 0275 CRLF PROC NEAR
503
504 0275 33 D2 XOR DX,DX ; SEND CR,LF TO FIRST PRINTER
505 0277 B8 0000 MOV AX,CR ; ASSUME FIRST PRINTER (DX=0)
506 027A CD 17 INT 17H ; GET THE PRINTER CHARACTER COMMAND
507 027C B8 000A MOV AX,LF ; GET THE CARRIAGE RETURN CHARACTER
508 027F CD 17 INT 17H ; NOW GET THE LINE FEED AND
509 0281 C3 RET ; SEND IT TO THE BIOS PRINTER ROUTINE
510 0282 CRLF ENDP
511
512
513
514 ;-- HARDWARE INT 08 H -- ( IRQ LEVEL 0 ) -----
515
516 ; THIS ROUTINE HANDLES THE TIMER INTERRUPT FROM CHANNEL 0 OF
517 ; THE 8254 TIMER. INPUT FREQUENCY IS 1.19318 MHZ AND THE DIVISOR
518 ; IS 65536, RESULTING IN APPROXIMATELY 18.2 INTERRUPTS EVERY SECOND.
519
520 ; THE INTERRUPT HANDLER MAINTAINS A COUNT (40:16C) OF INTERRUPTS SINCE
521 ; POWER ON TIME, WHICH MAY BE USED TO ESTABLISH TIME OF DAY.
522 ; THE INTERRUPT HANDLER ALSO DECREMENTS THE MOTOR CONTROL COUNT (40:40)
523 ; OF THE DISKETTE, AND WHEN IT EXPIRES, WILL TURN OFF THE
524 ; DISKETTE MOTOR. IT ALSO SETS THE MOTOR RUNNING FLAG.
525 ; THE INTERRUPT HANDLER WILL ALSO INVOKES A USER ROUTINE THROUGH
526 ; INTERRUPT ICH AT EVERY TIME TICK. THE USER MUST CODE A
527 ; ROUTINE AND PLACE THE CORRECT ADDRESS IN THE VECTOR TABLE.
528
529
530 0282 TIMER_INT_1 PROC FAR
531 0282 FB ST1: ; INTERRUPTS BACK ON
532 0283 1E PUSH DS
533 0284 50 PUSH AX
534 0285 52 PUSH DX
535 0286 E 0000 E CALL DDS ; SAVE MACHINE STATE
536 0287 FF 06 006C R INC @TIMER_LOW ; ESTABLISH ADDRESSABILITY
537 028D 75 04 JNZ T4 ; INCREMENT COUNT
538 028F FF 06 006E R INC @TIMER_HIGH ; GO TO TEST DAY
539 0293 T4: INC FF 0001H ; INCREMENT HIGH WORD OF TIME
540 0293 83 3E 006E R 18 CMP @TIMER_HIGH,018H ; TEST FOR COUNT EQUALING 24 HOURS
541 0293 75 15 JNZ T5 ; GO TO TEST DAY
542 029A 81 3E 006C R 00B0 CMP @TIMER_LOW,0B0H ; GO TO DISKETTE_CTL
543 02A0 75 0D JNZ T5 ; GO TO DISKETTE_CTL
544
545 ;----- TIMER HAS GONE 24 HOURS
546
547 02A2 2B C0 SUB AX,AX
548 02A4 A3 006E R MOV @TIMER_HIGH,AX
549 02A7 A3 006C R MOV @TIMER_LOW,AX
550 02A8 C6 06 0070 R 01 MOV @TIMER_OFLOW
551
552 ;----- TEST FOR DISKETTE TIME OUT
553
554 02AF T5: DEC @MOTOR_COUNT ; DECREMENT DISKETTE MOTOR CONTROL
555 02B0 FF 0E 0040 R JNZ T6 ; RETURN IF COUNT NOT OUT
556 02B3 75 08 T6: AND @MOTOR_STATUS,0F0H ; TURN OFF MOTOR RUNNING BITS
557 02B5 B0 26 003F R F0 MOV AL,0CH
558 02B8 B0 0C MOV DX,03F2H ; FDC CTL PORT
559 02B8 BA 03F2 MOV OUT DX,AL ; TURN OFF THE MOTOR
560 02B9 EE
561
562 02C0 T6: INT ICH ; TIMER TICK INTERRUPT
563 02C0 CD 1C INT ICH ; TRANSFER CONTROL TO A USER ROUTINE
564
565 02C2 5A POP DX ; RESTORE (DX)
566 02C3 B0 20 MOV AL,EOI ; GET END OF INTERRUPT MASK
567 02C3 FA CLI ; DISABLE INTERRUPTS TILL STACK CLEARED
568 02C4 00 20 OUT INTAA00,AL ; END OF INTERRUPT TO 8259 - 1
569 02C8 58 POP AX
570 02C9 1F POP DS ; RESET MACHINE STATE
571 02CA CF IRET ; RETURN FROM INTERRUPT
572
573 02CB TIMER_INT_1 ENDP
574
575 02CB CODE ENDS
576
577

```

```

1 PAGE 118,121
2 TITLE ORGS ----- 06/10/85 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 E111
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 F3A
44 PUBLIC F3D
45 PUBLIC F3D1
46 PUBLIC FD_TBL
47 PUBLIC FLOPPY
48 PUBLIC K0
49 PUBLIC K10
50 PUBLIC K11
51 PUBLIC K12
52 PUBLIC K13
53 PUBLIC K4
54 PUBLIC K15
55 PUBLIC K6
56 PUBLIC K6L
57 PUBLIC K7
58 PUBLIC K8
59 PUBLIC K9
60 PUBLIC M4
61 PUBLIC M5
62 PUBLIC M6
63 PUBLIC M7
64 PUBLIC NMI_INT
65 PUBLIC PRINTSCREEN
66 PUBLIC PRINTER
67 PUBLIC SEEKS_I
68 PUBLIC SLAVE_VECTOR_TABLE
69 PUBLIC TUTOR
70 PUBLIC VECTOR_TABLE
71 PUBLIC VIDEO_PARMS
72
73 EXTRN BOOT_STRAP_1:NEAR
74 EXTRN CASSETTE_10_1:NEAR
75 EXTRN D1:NEAR
76 EXTRN DISK_INT_1:NEAR
77 EXTRN DISK_SETUP:NEAR
78 EXTRN DISK_10_1:NEAR
79 EXTRN DSKETTE_SETUP:NEAR
80 EXTRN EQUIPMENT_1:NEAR
81 EXTRN INT_287:NEAR
82 EXTRN K16:NEAR
83 EXTRN KEYBOARD_10_1:NEAR
84 EXTRN KEYBOARD_1:NEAR
85 EXTRN MEMORY_SIZE_DET_1:NEAR
86 EXTRN NMI_INT_1:NEAR
87 EXTRN PRINT_SCREEN_1:NEAR
88 EXTRN PRINTER_10_1:NEAR
89 EXTRN RE_DIRECT:NEAR
90 EXTRN RTC_287_1:NEAR
91 EXTRN RTE_INT:NEAR
92 EXTRN SEEK:NEAR
93 EXTRN START_1:NEAR
94 EXTRN TIME_OF_DAY_1:NEAR
95 EXTRN TIMER_INT_1:NEAR
96 EXTRN VIDEO_10_1:NEAR
97
98 ASSUME CS:CODE,DS:DATA
99
100
101
102
103
104
105
106
107
108
109

```

---

```

; THIS MODULE HAS BEEN ADDED TO FACILITATE THE EXPANSION OF THIS PROGRAM.
; IT ALLOWS FOR THE FIXED ORG STATEMENT ENTRY POINTS THAT HAVE TO REMAIN
; AT THE SAME ADDRESSES. THE USE OF ENTRY POINTS AND TABLES WITHIN THIS
; MODULE SHOULD BE AVOIDED AND ARE INCLUDED ONLY TO SUPPORT EXISTING CODE
; THAT VIOLATE THE STRUCTURE AND DESIGN OF BIOS. ALL BIOS ACCESS SHOULD
; USE THE DOCUMENTED INTERRUPT VECTOR INTERFACE FOR COMPATIBILITY.
;
```

---

```

110          PAGE
111
112          ;-----: COPYRIGHT NOTICE :-----;
113
114          ;-----: ORG 0E000H
115          ORG 00000H
116
117 0000 36 34 38 30 30 39          DB     '6480090 COPR. IBM 1981, 1985
118          30 20 43 4F 50 52
119          2E 20 49 42 4D 20
120          31 39 38 31 2C 20
121          31 39 38 35 20 20
122          20 20
123
124
125          ;-----: PARITY ERROR MESSAGES :-----;
126
127
128 0020 50 41 52 49 54 59          D1    DB     'PARITY CHECK 1',CR,LF ; PLANAR BOARD PARITY CHECK LATCH SET
129          20 43 48 45 43 48
130          20 31 0D 0A
131 0030 50 41 52 49 54 59          D2    DB     'PARITY CHECK 2',CR,LF ; I/O CHANNEL CHECK LATCH SET
132          20 43 48 45 43 48
133          20 32 0D 0A
134 0040 3F 3F 3F 3F 0D          D2A   DB     '??????',CR,LF
135          0A
136  = 0047          IP    =  $0E05BH
137          ;-----: ORG 0005BH
138 005B
139 005B
140 005B E9 0000 E          RESET: JMP   START_1          ; RESET START
141
142
143          ;-----: POST ERROR MESSAGES :-----;
144
145
146 005E 20 31 30 31 2D 53          E101  DB     ' 101-System Board Error',CR,LF ; INTERRUPT FAILURE
147          79 73 74 65 6D 20
148          42 6F 61 72 64 20
149          45 72 72 6F 72 0D
150          0A
151 0077 20 31 30 32 2D 53          E102  DB     ' 102-System Board Error',CR,LF ; TIMER FAILURE
152          79 73 74 65 6D 20
153          42 6F 61 72 64 20
154          45 72 72 6F 72 0D
155          0A
156 0090 20 31 30 33 2D 53          E103  DB     ' 103-System Board Error',CR,LF ; TIMER INTERRUPT FAILURE
157          79 73 74 65 6D 20
158          42 6F 61 72 64 20
159          45 72 72 6F 72 0D
160          0A
161 0049 20 31 30 34 2D 53          E104  DB     ' 104-System Board Error',CR,LF ; PROTECTED MODE FAILURE
162          79 73 74 65 6D 20
163          42 6F 61 72 64 20
164          45 72 72 6F 72 0D
165          0A
166 00C2 20 31 30 35 2D 53          E105  DB     ' 105-System Board Error',CR,LF ; LAST 8042 COMMAND NOT ACCEPTED
167          79 73 74 65 6D 20
168          42 6F 61 72 64 20
169          45 72 72 6F 72 0D
170          0A
171 00B8 20 31 30 36 2D 53          E106  DB     ' 106-System Board Error',CR,LF ; CONVERTING LOGIC TEST
172          79 73 74 65 6D 20
173          42 6F 61 72 64 20
174          45 72 72 6F 72 0D
175          0A
176 00F4 20 31 30 37 2D 53          E107  DB     ' 107-System Board Error',CR,LF ; HOT NMI TEST
177          79 73 74 65 6D 20
178          42 6F 61 72 64 20
179          45 72 72 6F 72 0D
180          0A
181 010D 20 31 30 38 2D 53          E108  DB     ' 108-System Board Error',CR,LF ; TIMER BUS TEST
182          79 73 74 65 6D 20
183          42 6F 61 72 64 20
184          45 72 72 6F 72 0D
185          0A
186 0126 20 31 30 39 2D 53          E109  DB     ' 109-System Board Error',CR,LF ; LOW MEG CHIP SELECT TEST
187          79 73 74 65 6D 20
188          42 6F 61 72 64 20
189          45 72 72 6F 72 0D
190          0A
191 013F 20 31 36 31 2D 53          E110  DB     ' 110-System Board Error',CR,LF ; LOW MEG CHIP SELECT TEST
192          79 73 74 65 6D 20
193          42 6F 61 72 64 20
194          45 72 72 6F 72 0D
195          0A
196 0168 20 31 36 32 2D 53          E111  DB     ' 111-System Options Not Set-(Run SETUP)',CR,LF ; DEAD BATTERY
197          55 50 29 00 0A
198 0168 20 31 36 32 2D 53          E112  DB     ' 112-System Options Not Set-(Run SETUP)',CR,LF ; CHECKSUM/CONFIG
199          79 73 74 65 6D 20
200          42 6F 61 72 64 20
201          45 72 72 6F 72 0D
202          53 65 74 20 28 52
203          75 6E 20 52 45 54
204          55 50 29 00 0A
205 0191 20 31 36 33 2D 54          E113  DB     ' 113-Time & Date Not Set-(Run SETUP)',CR,LF ; CLOCK NOT UPDATING
206          69 6D 6F 20 45 54
207          44 61 66 20 45 54
208          6F 74 20 52 65 74
209          29 28 52 75 6E 20
210          53 45 54 55 50 29
211          0A
212 01B7 20 31 36 34 2D 4D          E114  DB     ' 114-Memory Size Error-(Run SETUP)',CR,LF ; CMOS DOES NOT MATCH
213          65 6D 6F 72 79 20
214          53 69 7A 65 20 45
215          72 72 6F 72 2D 28
216          52 75 6E 20 53 45
217          54 55 65 20 0D 0A
218 01DB 20 31 36 35 2D 5D          E201  DB     ' 201-Memory Error',CR,LF
219          65 6D 6F 72 79 20
220          45 72 72 6F 72 0D
221          0A
222 01EE 20 32 30 32 2D 4D          E202  DB     ' 202-Memory Address Error',CR,LF
223          65 6D 6F 72 79 20          ; LINE ERROR 00->15

```

224 41 64 64 72 65 73  
225 73 20 45 72 72 6F  
226 65 79 62 6F 61 72  
227 0209 20 33 30 32 2D 4D E203 DB \* 203-Memory Address Error',CR,LF ; LINE ERROR 16->23  
228 65 60 6F 72 79 20  
229 41 64 64 72 65 73  
230 73 20 45 72 6F  
231 72 00 0A  
232 0224 20 33 30 31 2D 4B E301 DB \* 301-Keyboard Error',CR,LF ; KEYBOARD ERROR  
233 65 79 62 6F 61 72  
234 64 20 45 72 72 6F  
235 72 00 0A  
236 0239 20 33 30 32 2D 53 E302 DB \* 302-System Unit Keylock is Locked',CR,LF ; KEYBOARD LOCK ON  
237 79 73 74 65 6D 20  
238 65 6E 69 74 20 4B  
239 65 79 62 6F 68 6B  
240 20 33 30 2D 4C 7F  
241 63 6B 65 64 0D 0A  
242 025D 20 28 52 45 53 55 F3D DB \* (RESUME = "F1" KEY)',CR,LF  
243 4D 45 20 3D 20 22  
244 46 31 22 20 4B 45  
245 59 29 0D 0A  
246  
247 I----- NMI ENTRY  
248 = 0273 IP = \$  
249 02C3 :-- ORG 0E2C3H  
250 = 02C3 NMI\_INT EQU \$  
251 02C3 E9 0000 E JMP NMI\_INT\_1 ; VECTOR ON TO MOVED NMI CODE  
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 45 72 20 53  
256 79 73 74 64 6D 20  
257 55 6E 69 74 20 45  
258 72 72 6F 72 00 0A  
259  
260 I----- KEYBOARD/SYSTEM ERROR  
261 02E4 20 33 30 34 2D 4B E304 DB \* 304-Keyboard Or System Unit Error',CR,LF ; KEYBOARD CLOCK HIGH  
262 65 79 62 6F 61 72  
263 64 20 45 72 20 53  
264 79 73 74 64 6D 20  
265 55 6E 69 74 20 45  
266 72 72 6F 72 00 0A  
267  
268 030E 20 34 30 31 2D 43 E401 DB \* 401-CRT Error',CR,LF ; MONOCHROME  
269 52 54 20 45 72 72  
270 6F 72 00 0A  
271 031E 20 34 30 31 2D 43 E501 DB \* 501-CRT Error',CR,LF ; COLOR  
272 52 54 20 45 72 72  
273 6F 72 00 0A  
274 032E 20 36 30 31 2D 44 E601 DB \* 601-Diskette Error',CR,LF ; DISKETTE ERROR  
275 69 73 6B 65 74 74  
276 65 20 45 72 72 6F  
277 72 00 0A  
278 I----- DISKETTE BOOT RECORD IS NOT VALID  
279 0343 20 36 30 32 2D 44 E602 DB \* 602-Diskette Boot Record Error',CR,LF  
280 69 73 6B 65 74 74  
281 65 20 42 6F 6F 74  
282 20 52 65 63 6F 72  
283 64 20 45 72 72 6F  
284 72 00 0A  
285 I----- HARD FILE ERROR MESSAGE  
286 0364 31 37 38 30 2D 44 F1780 DB \* 1780-Disk 0 Failure',CR,LF  
287 69 73 6B 20 30 20  
288 46 61 69 6C 75 72  
289 55 6D 0D 0A  
290 0379 31 37 38 31 2D 44 F1781 DB \* 1781-Disk 1 Failure',CR,LF  
291 69 73 6B 20 31 20  
292 46 61 69 6C 75 72  
293 55 6D 0D 0A  
294 038E 31 37 38 32 2D 44 F1782 DB \* 1782-Disk Controller Failure',CR,LF  
295 69 73 6B 20 33 6F  
296 6E 74 72 6F 6C 6C  
297 65 72 20 46 61 69  
298 6C 75 72 65 6D 0A  
299 03AC 31 37 39 30 2D 44 F1790 DB \* 1790-Disk 0 Error',CR,LF  
300 69 73 6B 20 30 20  
301 45 72 72 6F 72 0D  
302 0A  
303 03BF 31 37 39 31 2D 44 F1791 DB \* 1791-Disk 1 Error',CR,LF  
304 69 73 6B 20 31 20  
305 45 72 72 6F 72 0D  
306 0A  
307  
308 03D2 52 4F 4D 20 20 45 F3A DB \* ROM Error ',CR,LF ; ROM CHECKSUM  
309 72 72 6F 72 20 0D  
310 0A  
311 03DF 20 20 20 20 2D 55 F3D1 DB \* -Unlock System Unit Keylock ',CR,LF  
312 6E 6C 63 63 6B 20  
313 53 62 63 65 60  
314 20 55 6E 69 74 20  
315 4B 65 79 6C 6F 63  
316 6B 20 0D 0A



```

431 0466 0100      DW  0256D      ; WRITE PRE-COMPENSATION CYLINDER
432 0466 00      DW  0      ; HEADS
433 0469 00      DW  0      ; CONTROL BYTE
434 046A 00 00 00  DW  0,0,0      ; LANDING ZONE
435 046D 01FF      DW  0511D      ; SECTORS/TRACK
436 046F 11      DB  17D      ; SECTORS/TRACK
437 0470 00      DB  0      ; CYLINDERS
438
439      ----- DRIVE TYPE 08
440
441 0471 02DD      DW  0733D      ; CYLINDERS
442 0473 05      DB  05D      ; HEADS
443 0474 0000      DW  0      ; CONTROL BYTE
444 0476 FFFF      DW  0FFFFH      ; NO WRITE PRE-COMPENSATION
445 0477 00      DW  0      ; LANDING ZONE
446 0479 00      DW  0      ; SECTORS/TRACK
447 047A 00 00 00  DW  0,0,0      ; CYLINDERS
448 047D 02DD      DW  0733D      ; HEADS
449 047F 11      DB  17D      ; CONTROL BYTE
450 0480 00      DB  0      ; LANDING ZONE
451
452      ----- DRIVE TYPE 09
453
454 0481 0384      DW  0900D      ; CYLINDERS
455 0483 0F      DB  15D      ; HEADS
456 0484 0000      DW  0      ; CONTROL BYTE
457 0485 FFFF      DW  0FFFFH      ; NO WRITE PRE-COMPENSATION
458 0488 00      DW  0      ; LANDING ZONE
459 0489 08      DB  008H      ; SECTORS/TRACK
460 048A 00 00 00  DW  0,0,0      ; CYLINDERS
461 048D 0385      DW  0901D      ; HEADS
462 048F 11      DB  17D      ; CONTROL BYTE
463 0490 00      DB  0      ; LANDING ZONE
464
465      ----- DRIVE TYPE 10
466
467 0491 0334      DW  0820D      ; CYLINDERS
468 0493 03      DB  03D      ; HEADS
469 0495 0000      DW  0      ; CONTROL BYTE
470 0496 FFFF      DW  0FFFFH      ; NO WRITE PRE-COMPENSATION
471 0498 00      DW  0      ; LANDING ZONE
472 0499 00      DW  0      ; SECTORS/TRACK
473 049A 00 00 00  DW  0,0,0      ; CYLINDERS
474 049D 0334      DW  0820D      ; HEADS
475 049F 11      DB  17D      ; CONTROL BYTE
476 04A0 00      DB  0      ; LANDING ZONE
477
478      ----- DRIVE TYPE 11
479
480 04A1 0357      DW  0855D      ; CYLINDERS
481 04A3 05      DB  05D      ; HEADS
482 04A4 0000      DW  0      ; CONTROL BYTE
483 04A6 FFFF      DW  0FFFFH      ; NO WRITE PRE-COMPENSATION
484 04A8 00      DW  0      ; LANDING ZONE
485 04A9 00      DW  0      ; SECTORS/TRACK
486 04AA 00 00 00  DW  0,0,0      ; CYLINDERS
487 04AD 0357      DW  0855D      ; HEADS
488 04AE 11      DB  17D      ; CONTROL BYTE
489 04B0 00      DB  0      ; LANDING ZONE
490
491      ----- DRIVE TYPE 12
492
493 04B1 0357      DW  0855D      ; CYLINDERS
494 04B3 07      DB  07D      ; HEADS
495 04B4 0000      DW  0      ; CONTROL BYTE
496 04B6 FFFF      DW  0FFFFH      ; NO WRITE PRE-COMPENSATION
497 04B8 00      DW  0      ; LANDING ZONE
498 04B9 00      DW  0      ; SECTORS/TRACK
499 04B0 00 00 00  DW  0,0,0      ; CYLINDERS
500 04B2 0357      DW  0855D      ; HEADS
501 04B4 11      DB  17D      ; CONTROL BYTE
502 04C0 00      DB  0      ; LANDING ZONE
503
504      ----- DRIVE TYPE 13
505
506 04C1 0132      DW  0306D      ; CYLINDERS
507 04C3 08      DB  08D      ; HEADS
508 04C4 0000      DW  0      ; CONTROL BYTE
509 04C6 0080      DW  0128D      ; WRITE PRE-COMPENSATION CYLINDER
510 04C8 00      DW  0      ; LANDING ZONE
511 04C9 00      DW  0      ; SECTORS/TRACK
512 04CA 00 00 00  DW  0,0,0      ; CYLINDERS
513 04CD 013F      DW  0319D      ; HEADS
514 04CF 11      DB  17D      ; CONTROL BYTE
515 04D0 00      DB  0      ; LANDING ZONE
516
517      ----- DRIVE TYPE 14
518
519 04D1 02DD      DW  0733D      ; CYLINDERS
520 04D3 07      DB  07D      ; HEADS
521 04D4 0000      DW  0      ; CONTROL BYTE
522 04D6 FFFF      DW  0FFFFH      ; NO WRITE PRE-COMPENSATION
523 04D8 00      DW  0      ; LANDING ZONE
524 04D9 00      DW  0      ; SECTORS/TRACK
525 04DA 00 00 00  DW  0,0,0      ; CYLINDERS
526 04DD 02DD      DW  0733D      ; HEADS
527 04DF 11      DB  17D      ; CONTROL BYTE
528 04E0 00      DB  0      ; LANDING ZONE
529
530      ----- DRIVE TYPE 15  RESERVED  ***** DO NOT USE*****
531
532 04E1 0000      DW  0000D      ; CYLINDERS
533 04E3 00      DB  00D      ; HEADS
534 04E4 0000      DW  0      ; CONTROL BYTE
535 04E6 0000      DW  0000D      ; WRITE PRE-COMPENSATION CYLINDER
536 04E7 00      DW  0      ; LANDING ZONE
537 04E9 00      DW  0      ; SECTORS/TRACK
538 04EA 00 00 00  DW  0,0,0      ; CYLINDERS
539 04ED 0000      DW  0000D      ; HEADS
540 04EF 00      DB  00D      ; CONTROL BYTE
541 04F0 00      DB  0      ; LANDING ZONE
542
543      ----- DRIVE TYPE 16
544

```

```

545 04F1 0264          DW    0612D      ; CYLINDERS
546 04F3 04          DB    04D      ; HEADS
547 04F4 0000          DW    00      ; WRITE PRE-COMPENSATION ALL CYLINDER
548 04F6 0000          DW    0000D
549 04F8 00          DB    0
550 04F9 00          DB    0
551 04FA 00 00 00      DB    0,0,0
552 04FD 0297          DW    0663D      ; CONTROL BYTE
553 04FF 11          DB    17D      ; LANDING ZONE
554 0500 00          DB    0      ; SECTORS/TRACK
555
556          ;----- DRIVE TYPE 17
557
558 0501 03D1          DW    0977D      ; CYLINDERS
559 0502 05          DB    05D      ; HEADS
560 0504 0000          DW    0
561 0506 012C          DW    0300D      ; WRITE PRE-COMPENSATION CYL
562 0508 00          DB    0
563 0509 00          DB    0
564 050A 00 00 00      DB    0,0,0
565 050D 03D1          DW    0977D      ; LANDING ZONE
566 050F 11          DB    17D      ; SECTORS/TRACK
567 0510 00          DB    0
568
569          ;----- DRIVE TYPE 18
570
571 0511 03D1          DW    0977D      ; CYLINDERS
572 0513 07          DB    07D      ; HEADS
573 0514 0000          DW    0
574 0516 FFFF          DW    0FFFFH
575 0518 00          DB    0
576 0519 00          DB    0
577 051A 00 00 00      DB    0,0,0
578 051D 03D1          DW    0977D      ; LANDING ZONE
579 051F 11          DB    17D      ; SECTORS/TRACK
580 0520 00          DB    0
581
582          ;----- DRIVE TYPE 19
583
584 0521 0400          DW    1024D      ; CYLINDERS
585 0523 07          DB    07D      ; HEADS
586 0524 0000          DW    0
587 0526 0200          DW    0512D      ; WRITE PRE-COMPENSATION CYLINDER
588 0528 00          DB    0
589 0529 00          DB    0
590 052B 00 00 00 00      DB    0,0,0
591 052D 03FF          DW    1033D      ; LANDING ZONE
592 052F 11          DB    17D      ; SECTORS/TRACK
593 0530 00          DB    0
594
595          ;----- DRIVE TYPE 20
596
597 0531 02DD          DW    0733D      ; CYLINDERS
598 0533 05          DB    05D      ; HEADS
599 0534 0000          DW    0
600 0536 012C          DW    0300D      ; WRITE PRE-COMPENSATION CYL
601 0538 00          DB    0
602 0539 00          DB    0
603 053A 00 00 00      DB    0,0,0
604 053D 02DC          DW    0732D      ; LANDING ZONE
605 053F 11          DB    17D      ; SECTORS/TRACK
606 0540 00          DB    0
607
608          ;----- DRIVE TYPE 21
609
610 0541 02DD          DW    0733D      ; CYLINDERS
611 0543 07          DB    07D      ; HEADS
612 0544 0000          DW    0
613 0546 012C          DW    0300D      ; WRITE PRE-COMPENSATION CYL
614 0547 00          DB    0
615 0549 00          DB    0
616 054A 00 00 00      DB    0,0,0
617 054D 02DC          DW    0732D      ; LANDING ZONE
618 054F 11          DB    17D      ; SECTORS/TRACK
619 0550 00          DB    0
620
621          ;----- DRIVE TYPE 22
622
623 0551 02DD          DW    0733D      ; CYLINDERS
624 0553 05          DB    05D      ; HEADS
625 0554 0000          DW    0
626 0556 012C          DW    0300D      ; WRITE PRE-COMPENSATION CYL
627 0558 00          DB    0
628 0559 00          DB    0
629 055A 00 00 00      DB    0,0,0
630 055D 02DD          DW    0733D      ; LANDING ZONE
631 055F 11          DB    17D      ; SECTORS/TRACK
632 0560 00          DB    0
633
634          ;----- DRIVE TYPE 23
635
636 0561 0132          DW    0306D      ; CYLINDERS
637 0563 04          DB    04D      ; HEADS
638 0565 0000          DW    0
639 0566 0000          DW    0000D      ; WRITE PRE-COMPENSATION ALL CYL
640 0568 00          DB    0
641 0569 00          DB    0
642 056A 00 00 00      DB    0,0,0
643 056D 0150          DW    0336D      ; LANDING ZONE
644 056F 11          DB    17D      ; SECTORS/TRACK
645 0570 00          DB    0
646
647          ;----- DRIVE TYPE 24    *** RESERVED ***
648
649 0571 0000          DW    0000D      ; CYLINDERS
650 0573 00          DB    00D      ; HEADS
651 0574 0000          DW    0000D      ; WRITE PRE-COMPENSATION CYL
652 0576 0000          DW    0000D
653 0578 00          DB    0
654 0579 00          DB    0
655 057A 00 00 00      DB    0,0,0
656 057D 0000          DW    0000D      ; LANDING ZONE
657 057F 00          DB    0      ; SECTORS/TRACK
658 0580 00          DB    0

```

659  
660 ;----- DRIVE TYPE 25 \*\*\* RESERVED\*\*\*  
661  
662 0581 0000 DW 0000D ; CYLINDERS  
663 0583 00 DW 00D ; HEADS  
664 0584 0000 DW 0000D ; WRITE PRE-COMPENSATION CYL  
665 0586 0000 DW 0000D ;  
666 0588 00 DW 0000D ;  
667 0589 00 DW 0000D ; CONTROL BYTE  
668 058A 0000 00 00 DW 0,0,0 ;  
669 058D 0000 DW 0000D ; LANDING ZONE  
670 058F 00 DW 00D ; SECTORS/TRACK  
671 0590 00 DW 0000D ;  
672 ;----- DRIVE TYPE 26 \*\*\* RESERVED\*\*\*  
673  
674  
675 0591 0000 DW 0000D ; CYLINDERS  
676 0593 00 DW 00D ; HEADS  
677 0594 0000 DW 0000D ; WRITE PRE-COMPENSATION CYL  
678 0596 0000 DW 0000D ;  
679 0598 00 DW 0000D ;  
680 0599 00 DW 0000D ; CONTROL BYTE  
681 059A 0000 00 00 DW 0,0,0 ;  
682 059D 0000 DW 0000D ; LANDING ZONE  
683 059F 0000 DW 00D ; SECTORS/TRACK  
684 05A0 00 DW 0000D ;  
685 ;----- DRIVE TYPE 27 \*\*\* RESERVED\*\*\*  
686  
687  
688 05A1 0000 DW 0000D ; CYLINDERS  
689 05A3 00 DW 00D ; HEADS  
690 05A4 0000 DW 0000D ; WRITE PRE-COMPENSATION CYL  
691 05A6 0000 DW 0000D ;  
692 05A7 00 DW 0000D ;  
693 05A9 00 DW 0000D ; CONTROL BYTE  
694 05AA 0000 00 00 DW 0,0,0 ;  
695 05AD 0000 DW 0000D ; LANDING ZONE  
696 05AF 00 DW 00D ; SECTORS/TRACK  
697 05B0 00 DW 0000D ;  
698 ;----- DRIVE TYPE 28 \*\*\* RESERVED\*\*\*  
699  
700  
701 05B1 0000 DW 0000D ; CYLINDERS  
702 05B3 00 DW 00D ; HEADS  
703 05B4 0000 DW 0000D ; WRITE PRE-COMPENSATION CYL  
704 05B6 0000 DW 0000D ;  
705 05B7 00 DW 0000D ;  
706 05B9 00 DW 0000D ; CONTROL BYTE  
707 05B9 00 00 00 DW 0,0,0 ;  
708 05BD 0000 DW 0000D ; LANDING ZONE  
709 05BF 00 DW 00D ; SECTORS/TRACK  
710 05C0 00 DW 0000D ;  
711 ;----- DRIVE TYPE 29 \*\*\* RESERVED\*\*\*  
712  
713  
714 05C1 0000 DW 0000D ; CYLINDERS  
715 05C3 00 DW 00D ; HEADS  
716 05C4 0000 DW 0000D ; WRITE PRE-COMPENSATION CYL  
717 05C5 0000 DW 0000D ;  
718 05C8 00 DW 0000D ;  
719 05C9 00 DW 0000D ; CONTROL BYTE  
720 05CA 0000 00 00 DW 0,0,0 ;  
721 05CD 0000 DW 0000D ; LANDING ZONE  
722 05CF 00 DW 00D ; SECTORS/TRACK  
723 05D0 00 DW 0000D ;  
724 ;----- DRIVE TYPE 30 \*\*\* RESERVED\*\*\*  
725  
726  
727 05D1 0000 DW 0000D ; CYLINDERS  
728 05D3 00 DW 00D ; HEADS  
729 05D4 0000 DW 0000D ; WRITE PRE-COMPENSATION CYL  
730 05D6 0000 DW 0000D ;  
731 05D8 00 DW 0000D ;  
732 05D9 00 DW 0000D ; CONTROL BYTE  
733 05DA 0000 00 00 DW 0,0,0 ;  
734 05DD 0000 DW 0000D ; LANDING ZONE  
735 05DF 00 DW 00D ; SECTORS/TRACK  
736 05E0 00 DW 0000D ;  
737 ;----- DRIVE TYPE 31 \*\*\* RESERVED\*\*\*  
738  
739  
740 05E1 0000 DW 0000D ; CYLINDERS  
741 05E3 00 DW 00D ; HEADS  
742 05E4 0000 DW 0000D ; WRITE PRE-COMPENSATION CYL  
743 05E6 0000 DW 0000D ;  
744 05E8 00 DW 0000D ;  
745 05E9 00 DW 0000D ; CONTROL BYTE  
746 05EA 0000 00 00 DW 0,0,0 ;  
747 05ED 0000 DW 0000D ; LANDING ZONE  
748 05EF 00 DW 00D ; SECTORS/TRACK  
749 05F0 00 DW 0000D ;  
750 ;----- DRIVE TYPE 32 \*\*\* RESERVED\*\*\*  
751  
752  
753 05F1 0000 DW 0000D ; CYLINDERS  
754 05F3 00 DW 00D ; HEADS  
755 05F4 0000 DW 0000D ; WRITE PRE-COMPENSATION CYL  
756 05F6 0000 DW 0000D ;  
757 05F8 00 DW 0000D ;  
758 05F9 00 DW 0000D ; CONTROL BYTE  
759 05FA 0000 00 00 DW 0,0,0 ;  
760 05FB 0000 DW 0000D ; LANDING ZONE  
761 05FF 00 DW 00D ; SECTORS/TRACK  
762 0600 00 DW 0000D ;  
763 ;----- DRIVE TYPE 33 \*\*\* RESERVED\*\*\*  
764  
765  
766 0601 0000 DW 0000D ; CYLINDERS  
767 0603 00 DW 00D ; HEADS  
768 0604 0000 DW 0000D ; WRITE PRE-COMPENSATION CYL  
770 0608 00 DW 0000D ;  
771 0609 00 DW 0000D ; CONTROL BYTE  
772 060A 00 00 00 DW 0,0,0 ;

```

773 060D 0000      DW  0000D      ; LANDING ZONE
774 060F 00      DB  00D      ; SECTORS/TRACK
775 0610 00      DB  0
776
777 ;----- DRIVE TYPE 34 *** RESERVED ***
778
779 0611 0000      DW  0000D      ; CYLINDERS
780 0613 00      DB  00D      ; HEADS
781 0614 0000      DW  0
782 0616 0000      DW  0000D      ; WRITE PRE-COMPENSATION CYL
783 0618 00      DB  0
784 0619 00      DB  0
785 061A 00 00 00  DW  0,0,0
786 061D 0000      DW  0000D      ; LANDING ZONE
787 061F 00      DB  00D      ; SECTORS/TRACK
788 0620 00      DB  0
789
790 ;----- DRIVE TYPE 35 *** RESERVED ***
791
792 0621 0000      DW  0000D      ; CYLINDERS
793 0623 00      DB  00D      ; HEADS
794 0624 0000      DW  0
795 0626 0000      DW  0000D      ; WRITE PRE-COMPENSATION CYL
796 0628 00      DB  0
797 0629 00      DB  0
798 062A 00 00 00  DW  0,0,0
799 062D 0000      DW  0000D      ; LANDING ZONE
800 062F 00      DB  00D      ; SECTORS/TRACK
801 0630 00      DB  0
802
803 ;----- DRIVE TYPE 36 *** RESERVED ***
804
805 0631 0000      DW  0000D      ; CYLINDERS
806 0633 00      DB  00D      ; HEADS
807 0634 0000      DW  0
808 0636 0000      DW  0000D      ; WRITE PRE-COMPENSATION CYL
809 0638 00      DB  0
810 0639 00      DB  0
811 063A 00 00 00  DW  0,0,0
812 063D 0000      DW  0000D      ; LANDING ZONE
813 063F 00      DB  00D      ; SECTORS/TRACK
814 0640 00      DB  0
815
816 ;----- DRIVE TYPE 37 *** RESERVED ***
817
818 0641 0000      DW  0000D      ; CYLINDERS
819 0643 00      DB  00D      ; HEADS
820 0644 0000      DW  0
821 0646 0000      DW  0000D      ; WRITE PRE-COMPENSATION CYL
822 0648 00      DB  0
823 0649 00      DB  0
824 064A 00 00 00  DW  0,0,0
825 064D 0000      DW  0000D      ; LANDING ZONE
826 064F 00      DB  00D      ; SECTORS/TRACK
827 0650 00      DB  0
828
829 ;----- DRIVE TYPE 38 *** RESERVED ***
830
831 0651 0000      DW  0000D      ; CYLINDERS
832 0653 00      DB  00D      ; HEADS
833 0654 0000      DW  0
834 0655 0000      DW  0000D      ; WRITE PRE-COMPENSATION CYL
835 0658 00      DB  0
836 0659 00      DB  0
837 065A 00 00 00  DW  0,0,0
838 065D 0000      DW  0000D      ; LANDING ZONE
839 065F 00      DB  00D      ; SECTORS/TRACK
840 0660 00      DB  0
841
842 ;----- DRIVE TYPE 39 *** RESERVED ***
843
844 0661 0000      DW  0000D      ; CYLINDERS
845 0663 00      DB  00D      ; HEADS
846 0665 0000      DW  0
847 0666 0000      DW  0000D      ; WRITE PRE-COMPENSATION CYL
848 0668 00      DB  0
849 0669 00      DB  0
850 066A 00 00 00  DW  0,0,0
851 066D 0000      DW  0000D      ; LANDING ZONE
852 066F 00      DB  00D      ; SECTORS/TRACK
853 0670 00      DB  0
854
855 ;----- DRIVE TYPE 40 *** RESERVED ***
856
857 0671 0000      DW  0000D      ; CYLINDERS
858 0673 00      DB  00D      ; HEADS
859 0675 0000      DW  0
860 0676 0000      DW  0000D      ; WRITE PRE-COMPENSATION CYL
861 0678 00      DB  0
862 0679 00      DB  0
863 067A 00 00 00  DW  0,0,0
864 067D 0000      DW  0000D      ; LANDING ZONE
865 067F 00      DB  00D      ; SECTORS/TRACK
866 0680 00      DB  0
867
868 ;----- DRIVE TYPE 41 *** RESERVED ***
869
870 0681 0000      DW  0000D      ; CYLINDERS
871 0683 00      DB  00D      ; HEADS
872 0684 0000      DW  0
873 0686 0000      DW  0000D      ; WRITE PRE-COMPENSATION CYL
874 0688 00      DB  0
875 0689 00      DB  0
876 068A 00 00 00  DW  0,0,0
877 068D 0000      DW  0000D      ; LANDING ZONE
878 068F 00      DB  00D      ; SECTORS/TRACK
879 0690 00      DB  0
880
881 ;----- DRIVE TYPE 42 *** RESERVED ***
882
883 0691 0000      DW  0000D      ; CYLINDERS
884 0693 00      DB  00D      ; HEADS
885 0694 0000      DW  0
886 0696 0000      DW  0000D      ; WRITE PRE-COMPENSATION CYL

```

887 0698 00 DB 0  
888 0699 00 DB 0  
889 069A 00 00 00 DB 0,0,0 ; CONTROL BYTE  
890 069D 0000 DW 0000D ; LANDING ZONE  
891 069F 00 DB 00D ; SECTORS/TRACK  
892 06A0 00 DB 0  
893  
894 ;----- DRIVE TYPE 43 \*\*\* RESERVED\*\*\*  
895  
896 06A1 0000 DW 0000D ; CYLINDERS  
897 06A3 00 DB 00D ; HEADS  
898 06A4 0000 DW 0  
899 06A5 0000 DW 0000D ; WRITE PRE-COMPENSATION CYL  
900 06A8 00 DB 0  
901 06A9 00 DB 0  
902 06AA 00 00 00 DB 0,0,0 ; CONTROL BYTE  
903 06AD 0000 DW 0000D ; LANDING ZONE  
904 06AF 00 DB 00D ; SECTORS/TRACK  
905 06B0 00 DB 0  
906  
907 ;----- DRIVE TYPE 44 \*\*\* RESERVED\*\*\*  
908  
909 06B1 0000 DW 0000D ; CYLINDERS  
910 06B3 00 DB 00D ; HEADS  
911 06B4 0000 DW 0  
912 06B5 0000 DW 0000D ; WRITE PRE-COMPENSATION CYL  
913 06B8 00 DB 0  
914 06B9 00 DB 0  
915 06BA 00 00 00 DB 0,0,0 ; CONTROL BYTE  
916 06BD 0000 DW 0000D ; LANDING ZONE  
917 06BF 00 DB 00D ; SECTORS/TRACK  
918 06C0 00 DB 0  
919  
920 ;----- DRIVE TYPE 45 \*\*\* RESERVED\*\*\*  
921  
922 06C1 0000 DW 0000D ; CYLINDERS  
923 06C3 00 DB 00D ; HEADS  
924 06C4 0000 DW 0  
925 06C6 0000 DW 0000D ; WRITE PRE-COMPENSATION CYL  
926 06C8 00 DB 0  
927 06C9 00 DB 0  
928 06CA 00 00 00 DB 0,0,0 ; CONTROL BYTE  
929 06CB 0000 DW 0000D ; LANDING ZONE  
930 06CF 00 DB 00D ; SECTORS/TRACK  
931 06D0 00 DB 0  
932  
933 ;----- DRIVE TYPE 46 \*\*\* RESERVED\*\*\*  
934  
935 06D1 0000 DW 0000D ; CYLINDERS  
936 06D3 00 DB 00D ; HEADS  
937 06D4 0000 DW 0  
938 06D6 0000 DW 0000D ; WRITE PRE-COMPENSATION CYL  
939 06D8 00 DB 0  
940 06D9 00 DB 0  
941 06D9 00 00 00 DB 0,0,0 ; CONTROL BYTE  
942 06DD 0000 DW 0000D ; LANDING ZONE  
943 06DF 00 DB 00D ; SECTORS/TRACK  
944 06E0 00 DB 0  
945  
946 ;----- DRIVE TYPE 47 \*\*\* RESERVED\*\*\*  
947  
948 06E1 0000 DW 0000D ; CYLINDERS  
949 06E3 00 DB 00D ; HEADS  
950 06E4 0000 DW 0  
951 06E6 0000 DW 0000D ; WRITE PRE-COMPENSATION CYL  
952 06E8 00 DB 0  
953 06E9 00 DB 0  
954 06EA 00 00 00 DB 0,0,0 ; CONTROL BYTE  
955 06ED 0000 DW 0000D ; LANDING ZONE  
956 06EF 00 DB 00D ; SECTORS/TRACK  
957 06F0 00 DB 0  
958  
959  
960 ;----- BOOT LOADER INTERRUPT  
961  
962 = 06F1 IP = \$  
963 ;---- ORG 0E6F2H  
964 06F2 ORG 006F2H  
965 = 06F2 BOOT\_STRAP EQU \$  
966 06F2 E9 0000 E JMP BOOT\_STRAP\_1 ; VECTOR ON TO MOVED BOOT CODE  
967  
968  
969 06F5 CONF\_TBL:  
970 06F5 0008 DW CONF\_E-CONF\_TBL-2 ; USE INT 15 H AH=0C0H  
971 06F6 0C DB MODEL\_BYTE ; CONFIGURATION TABLE FOR THIS SYSTEM  
972 06F7 00 DB SYSTEM\_MODEL\_BYTE ; LENGTH OF FOLLOWING TABLE  
973 06F9 01 DB BIOS\_LEVEL ; SYSTEM MODEL BYTE  
974 06FA 70 DB 0111000B ; SYSTEM MODEL TYPE BYTE  
975  
976  
977 06FB 00 DB 0 ; BIOS REVISION LEVEL  
978 06FC 00 DB 0 ; 00000000 = DMA CHANNEL 3 USE BY BIOS  
980 06FD 00 DB 0 ; 01000000 = CASCADED INTERRUPT LEVEL 2  
981 06FE 00 DB 0 ; 00100000 = REAL TIME CLOCK AVAILABLE  
982 = 06FF CONF\_E EQU \$ ; 00010000 = KEYBOARD SCAN CODE HOOK 1AH  
983  
984  
985 ;----- BAUD RATE INITIALIZATION TABLE  
986 = 06FF IP = \$  
987 ;---- ORG 0E729H  
988 0729 ORG 00729H  
989 072A 0417 A1 DW 1047 ; 110 BAUD ; TABLE OF VALUES  
990 072B 0300 DW 165 ; 150 ; FOR INITIALIZATION  
991 072D 0180 DW 344 ; 300  
992 072F 00C0 DW 192 ; 600  
993 0731 0060 DW 96 ; 1200  
994 0733 0030 DW 48 ; 2400  
995 0735 0018 DW 24 ; 4800  
996 0737 000C DW 12 ; 9600  
997  
998 ;----- RS232  
999  
1000 ;---- ORG 0E739H

```

1001 0739 ORG 00739H
1002 = 0739 EQU $
1003 0739 E9 0000 E JMP RS232_10_I ; VECTOR ON TO MOVED RS232 CODE
1004
1005 ;----- KEYBOARD
1006
1007 ;----- ORG 0082EH
1008 082E ;----- ORG 0082EH
1009 = 082E KEYBOARD_10 EQU $
1010 082E E9 0000 E JMP KEYBOARD_10_I ; VECTOR ON TO MOVED KEYBOARD CODE
1011
1012 ;----- TABLE OF SHIFT KEYS AND MASK VALUES (EARLY PC)
1013
1014 ;----- ORG 0E87EH
1015 087E ;----- ORG 0E87EH
1016 087E 52 K6 DB INS_KEY ; INSERT KEY
1017 087F 3A 45 46 38 ID DB CAPS_KEY,NUM_KEY,SCROLL_KEY,ALT_KEY,CTL_KEY
1018 0884 2A 36 DB LEFT_KEY,RIGHT_KEY
1019 = 0008 K6L EQU $-K6
1020
1021 ;----- SHIFT_MASK_TABLE
1022
1023 0886 80 K7 DB INS_SHIFT ; INSERT MODE SHIFT
1024 0887 40 20 10 08 04 DB CAPS_SHIFT,NUM_SHIFT,SCROLL_SHIFT,ALT_SHIFT,CTL_SHIFT
1025 088C 02 01 DB LEFT_SHIFT,RIGHT_SHIFT
1026
1027 ;----- SCAN CODE TABLES
1028
1029 088E 1B FF 00 FF FF FF FF K8 DB 27,-1,0,-1,-1,-1,30,-1,-1,-1,-1,31
1030 089A 1E FF 00 FF FF FF FF DB -1,127,-1,17,23,5,18,20,25,21,9,15
1031 089B 1F FF 00 FF FF FF FF
1032 12 14 19 15 09 0F
1033 08A6 10 1B 1D 0A FF 01
1034 13 04 06 07 08 0A
1035 08B2 08 0C FF FF FF FF
1036 08B3 1C 1A 18 02 16 02
1037 08B4 0E 0C FF FF FF FF
1038 FF FF 20 FF
1039 ;----- CTL TABLE SCAN
1040
1041 08C8 5E 5F 60 61 62 63 K9 DB 94,95,96,97,98,99,100,101,102,103,-1,-1
1042 64 65 66 67 FF FF FF DB 119,-1,132,-1,115,-1,116,-1,117,-1,118,-1
1043 08D4 17 FF 84 FF 73 FF
1044 74 FF 75 FF 76 FF
1045 08E0 FF
1046
1047 ;----- LC TABLE
1048
1049 08E1 1B 31 32 33 34 35 K10 DB 01BH,'1234567890=-',08H,09H
1050 36 37 38 39 30 2D
1051 3D 08 09
1052 08F0 71 77 65 72 74 79 K11 DB 'qwertyuiop[],0DH,-1,'asdfghjkl:,027H
1053 75 69 6F 72 5B 5D
1054 00 FF 61 73 64 66
1055 65 68 6A 6B 6C 3B
1056 27
1057 0909 60 FF 5C 7A 78 63
1058 76 62 6E 6D 2C 2E
1059 2F FF 2A FF 20
1060 091A FF
1061 ;----- UC TABLE
1062
1063
1064 091B 1B 21 40 23 24 25 K12 DB 27,!*#$*,37,05EH,'&*()_-=',08H,0
1065 5E 26 2A 28 29 5F
1066 2B 08 00
1067 092A 1C 20 41 52 54 59
1068 55 49 4F 50 7D
1069 00 FF 41 53 44 46
1070 47 48 4A 4B 4C 3A
1071 22
1072 0943 7E FF 7C 5A 58 43
1073 56 42 4E 4D 3C 3E
1074 3F FF 00 FF 20 FF
1075 ;----- UC TABLE SCAN
1076
1077 0955 54 55 56 57 58 59 K13 DB 104,105,106,107,108
1078 095B 5A 5B 5C 5D 5D DB 109,110,111,112,113
1079 ;----- ALT TABLE SCAN
1080
1081 095F 68 69 6A 6B 6C
1082 0964 6D 6E 6F 70 71
1083 ;----- NUM STATE TABLE
1084
1085
1086 0969 37 38 39 2D 34 35 K14 DB '789-456+1230.'
1087 36 2B 31 32 33 30
1088 2E
1089 ;----- BASE CASE TABLE
1090
1091 0976 47 48 49 FF 4B FF K15 DB 71,72,73,-1,75,-1
1092 097C 4D FF 4F 50 51 52 DB 77,-1,79,80,81,82,83
1093 53
1094 ;----- KEYBOARD INTERRUPT
1095
1096 ;----- ORG 0E987H
1097 ;----- ORG 00987H
1098 0987 KB_INT EQU $
1099 = 0987 JMP KB_INT_I ; VECTOR ON TO MOVED KEYBOARD HANDLER
1100 0987 E9 0000 E

```

```

1101
1102 ;----- DISKETTE I/O
1103
1104 ;+ ORG 00C59H
1105 0C59
1106 = 0C59
1107 0C59 E9 0000 E
1108
1109 ;----- DISKETTE INTERRUPT
1110
1111 ;+ ORG 00F57H
1112 0F57
1113 = 0F57
1114 0F57 E9 0000 E
1115
1116 ;----- DISKETTE PARAMETERS
1117
1118 ;+ ORG 00FC7H
1119 0FC7
1120
1121 ;----- DISK_BASE
1122 ; THIS IS THE SET OF PARAMETERS REQUIRED FOR
1123 ; DISKETTE OPERATION. THEY ARE POINTED AT BY THE
1124 ; DATA VARIABLE DISK_POINTER. TO MODIFY THE PARAMETERS,
1125 ; BUILD ANOTHER PARAMETER BLOCK AND POINT AT IT
1126
1127
1128
1129 0FC7
1130 DISK_BASE LABEL BYTE
1131 0FC7 DF DB 1101111B ; SRT=0, HD UNLOAD=0F - 1ST SPECIFY BYTE
1132 0FC8 2 DB 10000000 ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
1133 0FC9 25 DB 00000000 ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
1134 0FC4 02 DB 2 ; 512 BYTES/SECTOR
1135 0FCB 0F DB 15 ; EOT ( LAST SECTOR ON TRACK )
1136 0FC4 1B DB 01BH ; GAP LENGTH
1137 0FCF FF DB 0FFH ; DTL
1138 0FC4 54 DB 054H ; GAP LENGTH FOR FORMAT
1139 0FC4 F6 DB 0F6H ; FILE BYTE FOR FORMAT
1140 0FD0 0F DB 15 ; HEAD SETTLE TIME (MILLI SECONDS)
1141 0FD1 08 DB 8 ; MOTOR START TIME (1/8 SECONDS)
1142
1143 ;----- PRINTER I/O
1144
1145 ;+ ORG 00FD2H
1146 0FD2
1147 = 0FD2
1148 0FD2 E9 0000 E
1149
1150 ;----- FOR POSSIBLE COMPATIBILITY ENTRY POINTS
1151
1152 ;+ ORG 00F45H
1153 1045
1154
1155 ASSUME CS:CODE,DS:DATA
1156
1157 EXTRN SET_MODE:NEAR
1158 EXTRN SET_CTYPE:NEAR
1159 EXTRN SET_CURSOR:NEAR
1160 EXTRN READ_LPEN:NEAR
1161 EXTRN ACT_DISP_PAGE:NEAR
1162 EXTRN SCROLL_UP:NEAR
1163 EXTRN SCROLL_DOWN:NEAR
1164 EXTRN READ_AC_CURSOR:NEAR
1165 EXTRN WRITE_AC_CURRENT:NEAR
1166 EXTRN WRITE_C_CURRENT:NEAR
1167 EXTRN SET_COLOR:NEAR
1168 EXTRN WRITE_DOT:NEAR
1169 EXTRN READ_DOT:NEAR
1170 EXTRN WRITE_TTY:NEAR
1171 EXTRN VIDEO_STATE:NEAR
1172
1173 1045 0000 E M1 DW OFFSET SET_MODE ; TABLE OF ROUTINES WITHIN VIDEO I/O
1174 1045 0000 E DW OFFSET SET_CTYPE ; EXIT STACK VALUES MAY BE
1175 1045 0000 E DW OFFSET SET_CURSOR ; DIFFERENT DEPENDING ON THE
1176 1045 0000 E DW OFFSET READ_CURSOR ; SYSTEM AND MODEL
1177 104D 0000 E DW OFFSET READ_LPEN
1178 104F 0000 E DW OFFSET ACT_DISP_PAGE
1179 1051 0000 E DW OFFSET SCROLL_UP
1180 1053 0000 E DW OFFSET SCROLL_DOWN
1181 1055 0000 E DW OFFSET READ_AC_CURRENT
1182 1057 0000 E DW OFFSET WRITE_AC_CURRENT
1183 1059 0000 E DW OFFSET WRITE_C_CURRENT
1184 105B 0000 E DW OFFSET SET_COLOR
1185 105D 0000 E DW OFFSET WRITE_DOT
1186 105F 0000 E DW OFFSET READ_DOT
1187 1061 0000 E DW OFFSET WRITE_TTY
1188 1063 0000 E DW OFFSET VIDEO_STATE
1189 = 0020 M1 EQU $-M1
1190
1191 ;+ ORG 00F65H
1192 1065
1193 = 1065
1194 1065 E9 0000 E
1195
1196 ;----- VIDEO PARAMETERS --- INIT_TABLE
1197
1198 ;+ ORG 00F44H
1199 1044
1200
1201 1044 VIDEO_PARMS LABEL BYTE
1202 1044 38 28 2D 0A 1F 06 DB 38H,28H,2DH,0AH,1FH,6,19H ; SET UP FOR 40X25
1203 19
1204 10AB 1C 02 07 06 07 DB 1CH,2,7,6,7
1205 108B 00 00 00 00 00 DB 0,0,0,0
1206 = 0010 M4 EQU $-VIDEO_PARMS
1207
1208 10B4 71 50 5A 0A 1F 06 DB 71H,50H,5AH,0AH,1FH,6,19H ; SET UP FOR 80X25
1209 19
1210 10BB 1C 02 07 06 07 DB 1CH,2,7,6,7
1211 10C0 00 00 00 00 00 DB 0,0,0,0
1212
1213 10C4 38 28 2D 0A 7F 06 DB 38H,28H,2DH,0AH,7FH,6,64H ; SET UP FOR GRAPHICS
1214 64

```

```

1215 10CB 70 02 01 06 07      DB    70H,2,1,6,7
1216 10D0 00 00 00 00          DB    0,0,0,0
1217
1218 10D4 61 50 52 0F 19 06      DB    61H,50H,52H,0FH,19H,6,19H      ; SET UP FOR 80X25 B&W CARD
1219 19
1220 10DB 19 02 0D 0B 0C      DB    19H,2,0DH,0BH,0CH
1221 10E0 00 00 00 00          DB    0,0,0,0
1222
1223 10E4 0800      M5      DW    2048      ; TABLE OF REGEN LENGTHS
1224 10E5 1000      DW    4096      ; 40X25
1225 10E8 4000      DW    16384      ; 80X25
1226 10EA 4000      DW    16384      ; GRAPHICS
1227
1228 ;----- COLUMNS
1229
1230 10EC 28 28 50 50 28 28      M6      DB    40,40,80,80,40,40,80,80
1231 50 50
1232 ;----- C_REG_TAB
1233
1234 10F4 2C 28 2D 29 2A 2E      M7      DB    2CH,28H,2DH,29H,2AH,2EH,1EH,29H ; TABLE OF MODE SETS
1235 1E 29
1236 ;----- MEMORY SIZE
1237
1238 ;:- ORG 0F841H
1239 1841      ORG 01841H
1240 = 1841      MEMORY_SIZE_DET EQU $
1241 1841 E9 0000 E      JMP  MEMORY_SIZE_DET_1      ; VECTOR ON TO MOVED BIOS CODE
1242
1243 ;----- EQUIPMENT DETERMINE
1244
1245 ;:- ORG 0F840H
1246 184D      ORG 01840H
1247 = 184D      EQUIPMENT EQU $
1248 184D E9 0000 E      JMP  EQUIPMENT_1      ; VECTOR ON TO MOVED BIOS CODE
1249
1250 ;----- CASSETTE (N BIOS SUPPORT)
1251
1252 ;:- ORG 0F859H
1253 1859      ORG 01859H
1254 = 1859      CASSETTE_10 EQU $
1255 1859 E9 0000 E      JMP  CASSETTE_10_1      ; VECTOR ON TO MOVED BIOS CODE
1256
1257 ;----- CHARACTER GENERATOR GRAPHICS FOR 320X200 AND 640X200 GRAPHICS :
1258
1259 ;:- ORG 0FA46EH
1260 1A6E      ORG 01A46EH
1261 1A6E      CRT_CHAR_GEN LABEL BYTE
1262 1A6E      DB 000H,000H,000H,000H,000H,000H,000H,000H ; D_00 BLANK
1263 1A6E 00 00 00 00 00 00 00      DB 0TEH,081H,0A5H,081H,08DH,099H,081H,07EH ; D_01 SMILING FACE
1264 00 00
1265 1A76 00 01 A5 81 BD 99      DB 07EH,0FFH,0DBH,0FFH,0C3H,0E7H,0FFH,07EH ; D_02 SMILING FACE N
1266 1A76 01 0E
1267 1A7E 7E FF DB FF C3 E7      DB 06CH,0FEH,0FEH,0FEH,07CH,038H,010H,000H ; D_03 HEART
1268 FF TE
1269 1A86 6C FE FE FE TC 38      DB 010H,038H,07CH,0FEH,07CH,038H,010H,000H ; D_04 DIAMOND
1270 10 00
1271 1A8E 38 TC FE TC 38      DB 038H,07CH,038H,0FEH,0FEH,07CH,038H,07CH ; D_05 CLUB
1272 10 00
1273 1A96 38 TC 38 FE FE TC      DB 010H,010H,038H,07CH,0FEH,07CH,038H,07CH ; D_06 SPADE
1274 38 TC
1275 1A9E 10 10 38 TC FE TC      DB 000H,000H,018H,03CH,03CH,018H,000H,000H ; D_07 BULLET
1276 38 TC
1277 1A9A 00 00 18 3C 3C 18      DB 0FFH,0FFH,0E7H,0C3H,0C3H,0E7H,0FFH,0FFH ; D_08 BULLET NEG
1278 00 00
1279 1A9A FF FF E7 C3 C3 E7      DB 000H,03CH,066H,042H,042H,066H,03CH,000H ; D_09 CIRCLE
1280 FF FF
1281 1A9B 00 3C 66 42 42 66      DB 0FFH,0C3H,099H,08DH,08DH,099H,0C3H,0FFH ; D_10 CIRCLE NEG
1282 3C 00
1283 1A9E FF 99 BD BD 99      DB 000H,000H,000H,000H,000H,000H,000H,000H ; D_11 MALE
1284 03 FF
1285 1A9C 0F 07 0F 7D CC CC      DB 000H,007H,0FFH,07DH,0CCH,0CCH,0CCH,078H ; D_12 FEMALE
1286 CC T8
1287 1A9C 3C 66 66 3C 18      DB 03CH,066H,066H,066H,03CH,018H,07EH,018H ; D_13 EIGHTH NOTE
1288 TE 18
1289 1A9D 3C 66 3F 30 30 70      DB 03FH,033H,03FH,030H,030H,070H,0F0H,0E0H ; D_14 TWO 1/16 NOTE
1290 20 E0
1291 1A9E 7F 63 7F 63 63 67      DB 07FH,063H,07FH,063H,063H,067H,0E6H,0C0H ; D_15 SUN
1292 E6 C0
1293 1A9E 99 5A 3C ET E7 3C      DB 099H,05AH,03CH,0E7H,0E7H,03CH,05AH,099H ; D_16
1294 5A 99
1295
1296 1A9E 80 E0 F8 FE F8 E0      DB 080H,0E0H,0F8H,0FEH,0F8H,0E0H,080H,000H ; D_17 R ARROWHEAD
1297 80 00
1298 1A9F 02 0E 3E FE 3E 0E      DB 002H,00EH,03EH,0FEH,03EH,00EH,002H,000H ; D_18 L ARROWHEAD
1299 02 00
1300 1A9F 18 3C TE 18 18 7E      DB 018H,03CH,07EH,018H,018H,07EH,03CH,018H ; D_19 ARROW 2 VERT
1301 3C 00
1302 1B06 66 66 66 66 00      DB 066H,066H,066H,066H,066H,000H,066H,000H ; D_13 2 EXCLAMATIONS
1303 66 00
1304 1B0E 7F DB DB 7B 1B 1B      DB 07FH,0DBH,0DBH,07BH,01BH,01BH,01BH,000H ; D_14 PARAGRAPH
1305 1B 00
1306 1B16 3E 63 38 6C 6C 38      DB 03EH,063H,038H,06CH,06CH,038H,0CCH,078H ; D_15 SECTION
1307 C0 00
1308 1B1E 00 00 00 00 TE TE      DB 000H,000H,000H,000H,07EH,07EH,07EH,000H ; D_16 RECTANGLE
1309 TE 00
1310 1B26 18 3C TE 18 7E 3C      DB 018H,03CH,07EH,018H,07EH,03CH,018H,0FFH ; D_17 ARROW 2 VRT UP
1311 18 FF
1312 1B2E 18 3C TE 18 18 18      DB 018H,03CH,07EH,018H,018H,018H,000H ; D_18 ARROW VRT UP
1313 18 00
1314 1B36 18 00 18 18 7E 3C      DB 018H,018H,018H,018H,07EH,03CH,018H,000H ; D_19 ARROW VRT DOWN
1315 18 00
1316 1B3E 00 18 0C FE 0C 18      DB 000H,018H,00CH,0FEH,00CH,018H,000H,000H ; D_1A ARROW RIGHT
1317 00 00
1318 1B46 00 30 60 FE 60 30      DB 000H,030H,060H,0FEH,060H,030H,000H,000H ; D_1B ARROW LEFT
1319 00 00
1320 1B4E 00 00 CO CO FE      DB 000H,000H,0C0H,0C0H,0C0H,0FEH,000H,000H ; D_1C NOT INVERTED
1321 00 00
1322 1B56 00 24 66 FF 66 24      DB 000H,024H,066H,0FFH,066H,024H,000H,000H ; D_1D ARROW 2 HZ
1323 00 00
1324 1B5E 00 00 00 3C 7E FF FF      DB 000H,018H,03CH,07EH,0FFH,0FFH,000H,000H ; D_1E ARROWHEAD UP
1325 00 00
1326 1B66 00 00 FF FF TE 3C 18      DB 000H,0FFH,0FFH,07EH,03CH,018H,000H,000H ; D_1F ARROWHEAD DOWN
1327 00 00
1328

```

1329 IB6E 00 00 00 00 00 00 DB 000H,000H,000H,000H,000H,000H ; D\_20 SPACE  
 1330 IB76 30 08 78 30 30 00 DB 030H,078H,078H,030H,030H,000H ; D\_21 ! EXCLAMATION  
 1331 IB76 30 00 00 00 00 DB 06CH,06CH,06CH,000H,000H,000H ; D\_22 " QUOTATION  
 1334 IB76 6C 6C 6C 00 00 00 DB 06CH,06CH,0FEH,06CH,0FEH,06CH,000H ; D\_23 # LB.  
 1335 IB86 6C 6C FE 6C FE 6C DB 030H,07CH,0C0H,078H,00CH,0FBH,030H,000H ; D\_24 \$ DOLLAR SIGN  
 1337 IB8E 30 7C CO 78 0C F8 DB 000H,0C6H,0CCH,018H,030H,066H,0C6H,000H ; D\_25 % PERCENT  
 1338 IB8E 30 7C CO 78 0C F8 DB 038H,06CH,038H,076H,0DCH,0CCH,076H,000H ; D\_26 & AMPERSAND  
 1342 IB8E 76 00 00 00 00 DB 060H,060H,0C0H,000H,000H,000H,000H ; D\_27 \* APOSTROPHE  
 1343 IB8E 60 60 CO 00 00 00 DB 018H,030H,060H,060H,060H,030H,018H,000H ; D\_28 ( L. PARENTHESIS  
 1344 IB8E 18 00 60 60 30 00 DB 060H,030H,018H,018H,018H,030H,060H,000H ; D\_29 ) R. PARENTHESIS  
 1345 IB8E 18 00 60 60 30 00 DB 000H,000H,000H,0FCH,000H,000H,000H,000H ; D\_2D - DASH  
 1346 IB8E 18 00 60 60 30 00 DB 000H,000H,000H,0FCH,000H,000H,000H,000H ; D\_2E . PERIOD  
 1347 IB8E 60 30 18 18 18 30 DB 006H,00CH,018H,030H,060H,0C0H,080H,000H ; D\_2F / SLASH  
 1348 IB8E 60 00 00 00 00 00 DB 07CH,0C6H,0CEH,0DEH,0F6H,0E6H,07CH,000H ; D\_30 0  
 1349 IB8E 60 00 00 00 00 00 DB 030H,070H,030H,030H,030H,030H,0FCCH,000H ; D\_31 !  
 1350 IB8E 30 00 30 30 FC 30 30 DB 078H,0CCH,0C0H,038H,060H,0CCH,0FCCH,000H ; D\_32 2  
 1351 IB8E 00 00 00 00 00 30 DB 078H,0CCH,0C0H,038H,00CH,0CCH,078H,000H ; D\_33 3  
 1352 IB8E 30 60 00 00 00 00 DB 01CH,03CH,06CH,0CCH,0FEH,00CH,01EH,000H ; D\_34 4  
 1353 IB8E 00 00 00 00 00 30 DB 0FCCH,0C0H,0F8H,00CH,0CCH,078H,000H ; D\_35 5  
 1354 IB8E 30 60 00 00 00 00 DB 038H,060H,0C0H,0F8H,0CCH,078H,000H ; D\_36 6  
 1355 IB8E 00 00 00 FC 00 00 DB 0FCCH,0CCH,0C0H,018H,030H,030H,030H,000H ; D\_37 7  
 1356 IB8E 00 00 00 00 00 00 DB 078H,0CCH,0CCH,078H,0CCH,0CCH,078H,000H ; D\_38 8  
 1357 IB8E 00 00 00 00 00 00 DB 078H,0CCH,0CCH,07CH,00CH,018H,070H,000H ; D\_39 9  
 1358 IB8E 30 00 00 00 00 00 DB 000H,030H,030H,000H,000H,030H,030H,000H ; D\_3A : COLON  
 1359 IB8E 00 00 00 00 00 00 DB 000H,030H,030H,000H,000H,030H,030H,060H ; D\_3B : SEMICOLON  
 1360 IB8E 30 60 00 00 00 00 DB 018H,030H,060H,0C0H,060H,030H,018H,000H ; D\_3C < LESS THAN  
 1361 IB8E 30 60 00 00 00 00 DB 000H,000H,0FCH,000H,000H,0FCH,000H,000H ; D\_3D = EQUAL  
 1362 IB8E 00 30 30 00 00 30 DB 060H,030H,018H,0C0H,018H,030H,060H,000H ; D\_3E > GREATER THAN  
 1363 IB8E 30 00 00 00 00 00 DB 078H,0CCH,0C0H,018H,030H,000H,030H,000H ; D\_3F ? QUESTION MARK  
 1364 IB8E 30 60 00 00 00 00 DB 07CH,0C6H,0DEH,0DEH,0DEH,0C0H,078H,000H ; D\_40 \* AT  
 1365 IB8E 78 00 00 00 00 00 DB 030H,078H,0CCH,0CCH,0FCH,0CCH,0CCH,000H ; D\_41 A  
 1366 IB8E 30 78 CC CC FC CC DB 0FCCH,066H,066H,07CH,066H,066H,0FCCH,000H ; D\_42 B  
 1367 IB8E 30 78 CC CC FC CC DB 03CH,066H,0C0H,0C0H,0C0H,066H,03CH,000H ; D\_43 C  
 1368 IB8E 30 78 CC CC FC CC DB 0F8H,06CH,066H,066H,06CH,0F8H,000H ; D\_44 D  
 1369 IB8E 30 78 CC CC FC CC DB 0FEH,062H,068H,078H,068H,062H,0FEH,000H ; D\_45 E  
 1370 IB8E 62 68 78 68 60 DB 0FEH,062H,068H,078H,068H,060H,0F0H,000H ; D\_46 F  
 1371 IB8E 62 68 78 68 60 DB 03CH,066H,0C0H,0C0H,0CEH,066H,03EH,000H ; D\_47 G  
 1372 IB8E 62 68 78 68 60 DB 0CCH,0CCH,0CCH,0FCH,0CCH,0CCH,0CCH,000H ; D\_48 H  
 1373 IB8E 62 68 78 68 60 DB 078H,030H,030H,030H,030H,030H,030H,078H,000H ; D\_49 I  
 1374 IB8E 62 68 78 68 60 DB 01EH,0C0H,0C0H,0CCH,0CCH,0CCH,078H,000H ; D\_4A J  
 1375 IB8E 62 68 78 68 60 DB 0E6H,066H,06CH,078H,06CH,066H,0E6H,000H ; D\_4B K  
 1376 IB8E 62 68 78 68 60 DB 0F0H,060H,060H,062H,066H,062H,0FEH,000H ; D\_4C L  
 1377 IB8E 62 68 78 68 60 DB 0C6H,0EEH,0FEH,0FEH,0D6H,0C6H,0C6H,000H ; D\_4D M  
 1378 IB8E 62 68 78 68 60 DB 0C6H,0E6H,0F6H,0DEH,0CEH,0C6H,0C6H,000H ; D\_4E N  
 1379 IB8E 62 68 78 68 60 DB 038H,06CH,0C6H,0C6H,0C6H,0C6H,06CH,038H,000H ; D\_4F O  
 1380 IB8E 62 68 78 68 60 DB 0FCCH,066H,066H,07CH,060H,060H,0F0H,000H ; D\_50 P  
 1381 IB8E 62 68 78 68 60 DB 078H,0CCH,0CCH,0CCH,0DCH,078H,01CH,000H ; D\_51 Q  
 1382 IB8E 62 68 78 68 60 DB 0FCCH,066H,066H,07CH,06CH,066H,0E6H,000H ; D\_52 R  
 1383 IB8E 62 68 78 68 60 DB 078H,0CCH,0C0H,0CCH,0CCH,0CCH,078H,000H ; D\_53 S  
 1384 IB8E 62 68 78 68 60 DB 0FCCH,06CH,0C0H,070H,01CH,0CCH,078H,000H ; D\_54 T  
 1385 IB8E 62 68 78 68 60 DB 0FCCH,084H,030H,030H,030H,030H,030H,078H,000H ; D\_55 U  
 1386 IB8E 62 68 78 68 60 DB 0CCH,0CCH,0CCH,0CCH,0CCH,0CCH,078H,000H ; D\_56 V  
 1387 IB8E 62 68 78 68 60 DB 0C6H,0C6H,0C6H,0C6H,0C6H,0C6H,0C6H,000H ; D\_57 W

```

1443 C6 00
1444 1D2E C6 C0 6C 38 38 6C      DB  0C6H,0C6H,06CH,038H,038H,06CH,0C6H,000H ; D_58 X
1445 C6 00
1446 1D36 CC C0 CC T8 30 30      DB  0CCH,0CCH,0CCH,078H,030H,030H,078H,000H ; D_59 Y
1447 1D3E FC C6 8C 18 32 66      DB  0FEH,0C6H,08CH,018H,032H,066H,0FEH,000H ; D_5A Z
1448 FE 00
1450 1D46 T8 60 60 60 60 60      DB  078H,060H,060H,060H,060H,060H,078H,000H ; D_5B [ LEFT BRACKET
1451 T8 00
1452 1D4E C0 60 30 18 0C 06      DB  0C0H,060H,030H,018H,00CH,006H,002H,000H ; D_5C \ BACKSLASH
1453 02 00
1454 1D56 T8 18 18 18 18 18      DB  078H,018H,018H,018H,018H,078H,000H ; D_5D ] RIGHT BRACKET
1455 T8 00
1456 1D5E 10 38 6C C0 00 00      DB  010H,038H,06CH,0C6H,000H,000H,000H,000H ; D_5E ^ CIRCUMFLEX
1457 00 00
1458 1D66 00 00 00 00 00 00      DB  000H,000H,000H,000H,000H,000H,000H,0FFH ; D_5F _ UNDERSCORE
1459 00 FF
1460
1461 1D6E 30 30 18 00 00 00      DB  030H,030H,018H,000H,000H,000H,000H,000H ; D_60 ' APOSTROPHE REV
1462 00 00
1463 1D76 00 00 78 0C 7C CC      DB  000H,000H,078H,0CCH,07CH,0CCH,076H,000H ; D_61 a
1464 T6 00
1465 1D7E E0 60 70 TC 66 66      DB  0E0H,060H,060H,07CH,066H,066H,0DCH,000H ; D_62 b
1466 00 00
1467 1D86 00 00 78 CC C0 CC      DB  000H,000H,078H,0CCH,0C0H,0CCH,078H,000H ; D_63 c
1468 T8 00
1469 1D8E 1C 0C 0C TC CC CC      DB  01CH,00CH,00CH,07CH,0CCH,0CCH,076H,000H ; D_64 d
1470 T6 00
1471 1D96 00 00 78 CC FC C0      DB  000H,000H,078H,0CCH,0FCH,0C0H,078H,000H ; D_65 e
1472 00 00
1473 1D9E 38 6C 60 F0 60 60      DB  038H,06CH,060H,0F0H,060H,060H,0F0H,000H ; D_66 f
1474 F0 00
1475 1DA6 00 00 76 CC CC 7C      DB  000H,000H,076H,0CCH,0CCH,07CH,00CH,0F8H ; D_67 g
1476 0C F8
1477 1DAE E0 60 6C T6 66 66      DB  0E0H,060H,060H,076H,066H,066H,0E6H,000H ; D_68 h
1478 00 00
1479 1DB6 00 00 70 30 30 30      DB  030H,000H,070H,030H,030H,030H,078H,000H ; D_69 i
1480 T8 00
1481 1DBE 0C 00 0C 0C CC CC      DB  00CH,000H,00CH,0CCH,00CH,0CCH,0CCH,078H ; D_6A j
1482 CC 78
1483 1DC6 E0 60 66 6C 78 6C      DB  0E0H,060H,066H,06CH,078H,06CH,0E6H,000H ; D_6B k
1484 00 00
1485 1DCE 70 00 30 30 30 30      DB  070H,030H,030H,030H,030H,030H,078H,000H ; D_6C l
1486 T8 00
1487 1DDE 00 00 CC FE FE D6      DB  000H,000H,0CCH,0FEH,0FEH,0D6H,0C6H,000H ; D_6D m
1488 C6 00
1489 1DDE 00 00 F8 CC CC CC      DB  000H,000H,0F8H,0CCH,0CCH,0CCH,0CCH,000H ; D_6E n
1490 CC 00
1491 1DE6 00 00 78 CC CC CC      DB  000H,000H,078H,0CCH,0CCH,0CCH,078H,000H ; D_6F o
1492 T8 00
1493
1494 1DEE 00 00 DC 66 66 7C      DB  000H,000H,0DCH,066H,066H,07CH,060H,0F0H ; D_70 p
1495 60 F0
1496 1DF6 00 00 76 CC CC 7C      DB  000H,000H,076H,0CCH,0CCH,07CH,00CH,01EH ; D_71 q
1497 0C 1E
1498 1DFF 00 00 DC 76 66 60      DB  000H,000H,0DCH,076H,066H,060H,0F0H,000H ; D_72 r
1499 F0 00
1500 1E06 00 00 TC C0 78 0C      DB  000H,000H,07CH,0C0H,078H,00CH,0F8H,000H ; D_73 s
1501 F8 00
1502 1E0E 00 30 7C 30 30 34      DB  010H,030H,07CH,030H,030H,034H,018H,000H ; D_74 t
1503 18 00
1504 1E16 00 00 CC CC CC CC      DB  000H,000H,0CCH,0CCH,0CCH,0CCH,076H,000H ; D_75 u
1505 T6 00
1506 1E1E 00 00 CC CC CC 78      DB  000H,000H,0CCH,0CCH,0CCH,078H,030H,000H ; D_76 v
1507 30 00
1508 1E26 00 00 C6 D6 FE FE      DB  000H,000H,0C6H,0D6H,0FEH,0FEH,06CH,000H ; D_77 w
1509 6C 00
1510 1E2E 00 00 C6 6C 38 6C      DB  000H,000H,0C6H,06CH,038H,06CH,0C6H,000H ; D_78 x
1511 C6 00
1512 1E36 00 00 CC CC CC 7C      DB  000H,000H,0CCH,0CCH,0CCH,0CCH,07CH,00CH,0F8H ; D_79 y
1513 0C F8
1514 1E3E 00 00 FC 98 30 64      DB  000H,000H,0FCH,098H,030H,064H,0FCH,000H ; D_7A z
1515 00 00
1516 1E46 1C 30 30 E0 30 30      DB  01CH,030H,030H,0E0H,030H,030H,01CH,000H ; D_7B { LEFT BRACE
1517 1C 00
1518 1E4E 18 18 00 18 18      DB  018H,018H,018H,000H,018H,018H,018H,000H ; D_7C } BROKEN STROKE
1519 18 00
1520 1E56 1D 30 30 1C 30 30      DB  0E0H,030H,030H,01CH,030H,030H,0E0H,000H ; D_7D } RIGHT BRACE
1521 1D 00
1522 1E5E 76 DC 00 00 00 00      DB  076H,0DCH,000H,000H,000H,000H,000H,000H ; D_7E ~ TILDE
1523 00 00
1524 1E66 00 10 38 6C C6 C6      DB  000H,010H,038H,06CH,0C6H,0C6H,0FEH,000H ; D_7F ^ DELTA
1525 FE 00
1526
1527
1528
1529
1530 1E6E
1531 = 1E6E
1532 1E6E E9 0000 E
1533
1534
1535
1536
1537 1EA5
1538 = 1EA5
1539 1EA5 E9 0000 E

:----- TIME OF DAY
:-- ORG 0FE6EH
:-- ORG 01E6EH
TIME_OF_DAY EQU $
JMP TIME_OF_DAY_1 ; VECTOR ON TO MOVED BIOS CODE

:----- TIMER INTERRUPT
:-- ORG 0FEA5H
:-- ORG 01EA5H
TIMER_INT EQU $
JMP TIMER_INT_1 ; VECTOR ON TO MOVED BIOS CODE

```

```

1540 PAGE
1541 1----- VECTOR TABLE
1542
1543 11- ORG 0FEF3H
1544 IEF3 11- ORG 0FEF3H
1545 VECTOR_TABLE LABEL WORD : AT LOCATION 0FEF3H
1546 11- VECTOR_TABLE LABEL WORD : VECTOR TABLE VALUES FOR POST TESTS
1547 IEF5 0E5 R DW OFFSET TIMER_INT : INT 10H - HARDWARE TIMER 0 IRQ 0
1548 IEF5 0967 R DW OFFSET KEYBOARD : INT 09H - KEYBOARD IRQ 1
1549 IEF7 0000 E DW OFFSET DTI : INT 0AH - SLAVE INTERRUPT INPUT IRQ 3
1550 IEF8 0000 E DW OFFSET D11 : INT 0CH - IRQ 4
1551 IEF9 0000 E DW OFFSET D11 : INT 0DH - IRQ 5
1552 IEFF 0F57 R DW OFFSET DISK_INT : INT 0EH - DISKETTE IRQ 6
1553 IF01 0000 E DW OFFSET D11 : INT 0FH - IRQ 7
1554
1555
1556 1----- SOFTWARE INTERRUPTS ( BIOS CALLS AND POINTERS )
1557 IF03 1065 R DW OFFSET VIDEO_IO : INT 10H -- VIDEO DISPLAY
1558 IF05 1841 R DW OFFSET EQUIPMENT : INT 11H -- GET EQUIPMENT FLAG WORD
1559 IF07 1441 R DW OFFSET EQUIPMENT_SIZE_DET : INT 12H -- EQUIPMENT MODE/MEMORY SIZE
1560 IF09 0C59 R DW OFFSET DISKETTE_10 : INT 13H -- DISKETTE
1561 IF08 0739 R DW OFFSET RS232_10 : INT 14H -- COMMUNICATION ADAPTER
1562 IF0D 1859 R DW OFFSET CASSETTE_10 : INT 15H -- EXPANDED BIOS FUNCTION CALL
1563 IF0F 082E R DW OFFSET KEYBOARD_10 : INT 16H -- KEYBOARD INPUT
1564 IF10 0002 R DW OFFSET PRINTER_TO : INT 17H -- PRINTER OUTPUT
1565 IF13 0000 R DW OFFSET BOOT_STRAP : INT 18H -- SYSTEM CALLS FOR BASIC
1566 IF15 06F2 R DW OFFSET TIME_OF_DAY : INT 19H -- TIME OF DAY
1567 IF17 1E6E R DW OFFSET DUMMY_RETURN : INT 1AH -- TIMER BREAK ADDRESS
1568 IF19 1F53 R DW OFFSET VIDEO_PARMS : INT 1DH -- VIDEO PARAMETERS
1569 IF1B 1F53 R DW OFFSET DISK_BASE : INT 1EH -- DISKETTE PARAMETERS
1570 IF1D 104A R
1571 IF1F 0FC7 R
1572 IF21 0000 R DW 00000H : INT 1FH -- POINTER TO VIDEO EXTENSION
1573
1574 IF23 SLAVE_VECTOR_TABLE LABEL WORD : ( INTERRUPT 70H THRU 7FH )
1575
1576 IF23 0000 E DW OFFSET RTC_INT : INT 70H - REAL TIME CLOCK IRQ 8
1577 IF25 0000 E DW OFFSET _DIRECT : INT 71H - REDIRECT TO INT 0AH IRQ 9
1578 IF27 0000 E DW OFFSET _DTI : INT 72H - IRQ 10
1579 IF29 0000 E DW OFFSET D11 : INT 73H - IRQ 11
1580 IF2B 0000 E DW OFFSET D11 : INT 74H - IRQ 12
1581 IF2D 0000 E DW OFFSET INT_287 : INT 75H - -MATH COPROCESSOR IRQ 13
1582 IF2F 0000 E DW OFFSET D1T : INT 76H - -FIXED DISK IRQ 14
1583 IF31 0000 E DW OFFSET D11 : INT 77H - IRQ 15
1584
1585 1----- DUMMY INTERRUPT HANDLER
1586
1587 11- ORG 0FF53H
1588 IF53 11- ORG 01F53H
1589 = IF53 DUMMY_RETURN EQU $ : BIOS DUMMY (NULL) INTERRUPT RETURN
1590
1591 IF53 CF IRET
1592
1593 1----- PRINT SCREEN
1594
1595 11- ORG 0FF54H
1596 11- ORG 01F54H
1597 IF54 PRINT_SCREEN EQU $ : VECTOR ON TO MOVED BIOS CODE
1598 = IF54 PRINT_SCREEN_1 : TUTOR
1599 IF54 E9 0000 E .LIST JMP PRINT_SCREEN_1
1600
1601
1602
1603
1604
1605 1----- POWER ON RESET VECTOR
1606
1607 IFF0 11- ORG 0FFF0H : POWER ON RESTART EXECUTION LOCATION
1608 ORG 01FFF0H
1609
1610 1----- POWER ON RESET
1611 IFF0 P_O_R LABEL FAR
1612
1613 IFF0 EA DB 0EAH : HARD CODE FAR JUMP TO SET
1614 IFF1 005B R DW OFFSET RESET : OFFSET
1615 IFF3 F000 DW 0F00H : SEGMENT
1616
1617 IFF5 30 36 2F 31 30 2F DB '06/10/85' : RELEASE MARKER
1618 38 35
1619
1620 IFFE ORG 01FFEH
1621 IFFE FC DB MODEL_BYTE : THIS PC'S ID (MODEL BYTE)
1622
1623 IFFF CODE ENDS : CHECKSUM AT LAST LOCATION
1624 END

```

# SECTION 6. INSTRUCTION SET

## Contents

<b>80286 Instruction Set</b> .....	<b>6-3</b>
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
<b>80287 Coprocessor Instruction Set</b> .....	<b>6-22</b>
Data Transfer .....	6-22
Comparison .....	6-23
Constants .....	6-24
Arithmetic .....	6-25
Transcendental .....	6-26

## Notes:

# 80286 Instruction Set

## Data Transfer

### MOV = move

Register to Register/Memory

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

Register/Memory to Register

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

Immediate to Register/Memory

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

Immediate to Register

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

Memory to Accumulator

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

Accumulator to Memory

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

Register/Memory to Segment Register

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

Segment Register to Register/Memory

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

### PUSH = Push

Memory

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

Register

01010reg
----------

Segment Register

000reg110
-----------

Immediate

011010s0	data	data if s = 0
----------	------	---------------

**PUSHA = Push All**

01100000
----------

**POP = Pop**

Memory

10001111	mod000 r/m
----------	------------

Register

01011reg
----------

Segment Register

000reg111	reg ≠ 01
-----------	----------

**POPA = Pop All**

01100001
----------

**XCHG = Exchange**

Register/Memory with Register

1000011w	mod reg r/m
----------	-------------

Register with Accumulator

10010reg
----------

## IN = Input From

Fixed Port

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

Variable Port

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

## OUT = Output To

Fixed Port

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

Variable Port

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

## XLAT = Translate Byte to AL

11010111
----------

## LEA = Load EA to Register

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

## LDS = Load Pointer to DS

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

**DEC = Decimal Adjust for Add**

00100111
----------

**AAS = ASCII Adjust for Subtract**

00111111
----------

**DAS = Decimal Adjust for Subtract**

00110111
----------

**MUL = Multiply (Unsigned)**

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

**IMUL = Integer Multiply (Signed)**

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

## IIMUL = Integer Immediate Multiply (Signed)

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

## DIV = Divide (Unsigned)

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

## IDIV = Integer Divide (Signed)

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

## AAM = ASCII Adjust for Multiply

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

## AAD = ASCII Adjust for Divide

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

## CBW = Convert Byte to Word

10011000
----------

## CWD = Convert Word to Double Word

10011001
----------

# Logic

## Shift/Rotate Instructions

Register/Memory by 1

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

Register/Memory by CL

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

### Register/Memory by Count

1100000w	mod TTT r/m	count
TTT		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

0000 0dw	mod reg r/m	
----------	-------------	--

### Immediate to Register/Memory

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

### Immediate to Accumulator

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

## XOR = Exclusive OR

### Register/Memory and Register to Either

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

### Immediate to Register/Memory

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

### Immediate to Accumulator

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

## NOT = Invert Register/Memory

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

## String Manipulation

### MOVS = Move Byte Word

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

### CMPS = Compare Byte Word

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

### SCAS = Scan Byte Word

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

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

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

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

1010101w

**INS = Input Byte from DX Port**

0110110w

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

0110111w

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

Repeat Move String

11110011 | 1010010w

Repeat Compare String (z/Not z)

1111001z | 1010011w

Repeat Scan String (z/Not z)

1111001z | 1010111w

Repeat Load String

11110011 | 1010110w

Repeat Store String

11110011 | 1010101w

Repeat Input String

11110011 | 0110110w

Repeat Output String

11110011 | 1010011w

# Control Transfer

## CALL = Call

Direct Within Segment

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

Register/Memory Indirect Within Segment

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

Direct Intersegment

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

Indirect Intersegment

11111111	mod011 r/m (mod ≠ 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 Log<sub>2</sub> [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 $\neq$ 0
ST	Stack top	Stack top	$(i) =$ <i>i</i> th register from the top
d	Destination	Dest. is ST(0)	Dest. is ST( <i>i</i> )
P	Pop	No pop	Pop
R	Reverse*	Dest. (op) source	Source (op) dest.

\* When d=1, reverse the sense of R.

## Notes:

# SECTION 7. CHARACTERS, KEYSTROKES, AND COLORS

## Contents

<b>Character Codes</b>	.....	<b>7-3</b>
<b>Quick Reference</b>	.....	<b>7-14</b>

## Notes:

# Character Codes

Value		As Characters			As Text Attributes		
					Color/Graphics Monitor Adapter		IBM Monochrome Display Adapter
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground	
00	0	Blank (Null)	Ctrl 2		Black	Black	Non-Display
01	1	☺	Ctrl A		Black	Blue	Underline
02	2	☻	Ctrl B		Black	Green	Normal
03	3	♥	Ctrl C		Black	Cyan	Normal
04	4	♦	Ctrl D		Black	Red	Normal
05	5	♣	Ctrl E		Black	Magenta	Normal
06	6	♠	Ctrl F		Black	Brown	Normal
07	7	●	Ctrl G		Black	Light Grey	Normal
08	8	●	Ctrl H, Backspace, Shift Backspace		Black	Dark Grey	Non-Display
09	9	○	Ctrl I		Black	Light Blue	High Intensity Underline
0A	10	○	Ctrl J, Ctrl ←		Black	Light Green	High Intensity
0B	11	♂	Ctrl K		Black	Light Cyan	High Intensity
0C	12	♀	Ctrl L		Black	Light Red	High Intensity
0D	13	♪	Ctrl M, ←, Shift ←		Black	Light Magenta	High Intensity
0E	14	♪	Ctrl N		Black	Yellow	High Intensity
0F	15	★	Ctrl O		Black	White	High Intensity
10	16	►	Ctrl P		Blue	Black	Normal
11	17	◀	Ctrl Q		Blue	Blue	Underline
12	18	↑	Ctrl R		Blue	Green	Normal
13	19	!!	Ctrl S		Blue	Cyan	Normal
14	20	¶	Ctrl T		Blue	Red	Normal
15	21	§	Ctrl U		Blue	Magenta	Normal
16	22	▬	Ctrl V		Blue	Brown	Normal
17	23	▬	Ctrl W		Blue	Light Grey	Normal

Value		As Characters			As Text Attributes			
					Color/Graphics Monitor Adapter		IBM Monochrome Display Adapter	
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground		
18	24	↑	Ctrl X		Blue	Dark Grey	High Intensity	
19	25	↓	Ctrl Y		Blue	Light Blue	High Intensity	Underline
1A	26	→	Ctrl Z		Blue	Light Green	High Intensity	
1B	27	←	Ctrl [, Esc, Shift Esc, 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	

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	△	Ctrl -		Light Grey	White	High Intensity

Value		As Characters			As Text Attributes		
					Color/Graphics Monitor Adapter		IBM Monochrome Display Adapter
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground	
* * * * 80 to FF Hex are Flashing in both Color & IBM Monochrome * * * *							
80	128	ç	Alt 128	Note 6	Black	Black	Non-Display
81	129	ü	Alt 129	Note 6	Black	Blue	Underline
82	130	é	Alt 130	Note 6	Black	Green	Normal
83	131	â	Alt 131	Note 6	Black	Cyan	Normal
84	132	ä	Alt 132	Note 6	Black	Red	Normal
85	133	à	Alt 133	Note 6	Black	Magenta	Normal
86	134	å	Alt 134	Note 6	Black	Brown	Normal
87	135	ç	Alt 135	Note 6	Black	Light Grey	Normal
88	136	ê	Alt 136	Note 6	Black	Dark Grey	Non-Display
89	137	ë	Alt 137	Note 6	Black	Light Blue	High Intensity Underline
8A	138	è	Alt 138	Note 6	Black	Light Green	High Intensity
8B	139	ï	Alt 139	Note 6	Black	Light Cyan	High Intensity
8C	140	î	Alt 140	Note 6	Black	Light Red	High Intensity
8D	141	ì	Alt 141	Note 6	Black	Light Magenta	High Intensity
8E	142	Ä	Alt 142	Note 6	Black	Yellow	High Intensity
8F	143	Å	Alt 143	Note 6	Black	White	High Intensity
90	144	É	Alt 144	Note 6	Blue	Black	Normal
91	145	æ	Alt 145	Note 6	Blue	Blue	Underline
92	146	Æ	Alt 146	Note 6	Blue	Green	Normal
93	147	ô	Alt 147	Note 6	Blue	Cyan	Normal
94	148	ö	Alt 148	Note 6	Blue	Red	Normal
95	149	ò	Alt 149	Note 6	Blue	Magenta	Normal
96	150	û	Alt 150	Note 6	Blue	Brown	Normal
97	151	ù	Alt 151	Note 6	Blue	Light Grey	Normal
98	152	ÿ	Alt 152	Note 6	Blue	Dark Grey	High Intensity
99	153	Ö	Alt 153	Note 6	Blue	Light Blue	High Intensity Underline
9A	154	Ü	Alt 154	Note 6	Blue	Light Green	High Intensity



Value		As Characters		As Text Attributes			
				Color/Graphics Monitor Adapter		IBM Monochrome Display Adapter	
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground	
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
CO	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
DO	208	█	Alt 208	Note 6	Magenta	Black	Normal

Value		As Characters			As Text Attributes		
					Color/Graphics Monitor Adapter		IBM Monochrome Display Adapter
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground	
D1	209		Alt 209	Note 6	Magenta	Blue	Underline
D2	210		Alt 210	Note 6	Magenta	Green	Normal
D3	211		Alt 211	Note 6	Magenta	Cyan	Normal
D4	212		Alt 212	Note 6	Magenta	Red	Normal
D5	213		Alt 213	Note 6	Magenta	Magenta	Normal
D6	214		Alt 214	Note 6	Magenta	Brown	Normal
D7	215		Alt 215	Note 6	Magenta	Light Grey	Normal
D8	216		Alt 216	Note 6	Magenta	Dark Grey	High Intensity
D9	217		Alt 217	Note 6	Magenta	Light Blue	High Intensity Underline
DA	218		Alt 218	Note 6	Magenta	Light Green	High Intensity
DB	219		Alt 219	Note 6	Magenta	Light Cyan	High Intensity
DC	220		Alt 220	Note 6	Magenta	Light Red	High Intensity
DD	221		Alt 221	Note 6	Magenta	Light Magenta	High Intensity
DE	222		Alt 222	Note 6	Magenta	Yellow	High Intensity
DF	223		Alt 223	Note 6	Magenta	White	High Intensity
E0	224	$\alpha$	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	$\infty$	Alt 236	Note 6	Brown	Light Red	High Intensity
ED	237	$\phi$	Alt 237	Note 6	Brown	Light Magenta	High Intensity
EE	238	$\epsilon$	Alt 238	Note 6	Brown	Yellow	High Intensity
EF	239	$\cap$	Alt 239	Note 6	Brown	White	High Intensity
F0	240	$\equiv$	Alt 240	Note 6	Light Grey	Black	Reverse Video
F1	241	$\pm$	Alt 241	Note 6	Light Grey	Blue	Underline
F2	242	$\geq$	Alt 242	Note 6	Light Grey	Green	Normal
F3	243	$\leq$	Alt 243	Note 6	Light Grey	Cyan	Normal
F4	244	$\int$	Alt 244	Note 6	Light Grey	Red	Normal
F5	245	$\int$	Alt 245	Note 6	Light Grey	Magenta	Normal
F6	246	$\div$	Alt 246	Note 6	Light Grey	Brown	Normal
F7	247	$\approx$	Alt 247	Note 6	Light Grey	Light Grey	Normal
F8	248	$\circ$	Alt 248	Note 6	Light Grey	Dark Grey	Reverse Video
F9	249	$\bullet$	Alt 249	Note 6	Light Grey	Light Blue	High Intensity Underline
FA	250	$\bullet$	Alt 250	Note 6	Light Grey	Light Green	High Intensity
FB	251	$\sqrt{-}$	Alt 251	Note 6	Light Grey	Light Cyan	High Intensity
FC	252	$^n$	Alt 252	Note 6	Light Grey	Light Red	High Intensity
FD	253	$^2$	Alt 253	Note 6	Light Grey	Light Magenta	High Intensity
FE	254	$\blacksquare$	Alt 254	Note 6	Light Grey	Yellow	High Intensity
FF	255	<b>BLANK</b>	Alt 255	Note 6	Light Grey	White	High Intensity

## Notes

1. Asterisk (\*) can be typed using two methods: press the (\*) key or, in the shift mode, press the 8 key.
2. Period (.) can be typed using two methods: press the . key or, in the shift or Num Lock mode, press the Del key.
3. Numeric characters 0-9 can be typed using two methods: press the numeric keys on the top row of the keyboard or, in the shift or Num Lock mode, press the numeric keys in the keypad portion of the keyboard.
4. Uppercase alphabetic characters (A-Z) can be typed in two modes: the shift mode or the Caps Lock mode.
5. Lowercase alphabetic characters (a-z) can be typed in two modes: in the normal mode or in Caps Lock and shift mode combined.
6. The three digits after the Alt key must be typed from the numeric keypad. Character codes 1-255 may be entered in this fashion (with Caps Lock activated, character codes 97-122 will display uppercase).

# Quick Reference

DECIMAL VALUE	◆	0	16	32	48	64	80	96	112
↓	HEXA- DECIMAL VALUE	0	1	2	3	4	5	6	7
0	0	BLANK (NULL)	▶	BLANK (SPACE)	0	@	P	‘	p
1	1	☺	◀	!	1	A	Q	a	q
2	2	☻	↑	॥	2	B	R	b	r
3	3	♥	!!	#	3	C	S	c	s
4	4	◆	¶	\$	4	D	T	d	t
5	5	♣	§	%	5	E	U	e	u
6	6	♠	▬	&	6	F	V	f	v
7	7	•	↑	’	7	G	W	g	w
8	8	•	↑	(	8	H	X	h	x
9	9	○	↓	)	9	I	Y	i	y
10	A	○	→	*	:	J	Z	j	z
11	B	♂	←	+	;	K	[	k	{
12	C	♀	▬	,	<	L	\	l	▬
13	D	♪	↔	—	=	M	]	m	}
14	E	♪	▲	.	>	N	^	n	~
15	F	☀	▼	/	?	O	—	o	△



## Notes:

# SECTION 8. COMMUNICATIONS

## Contents

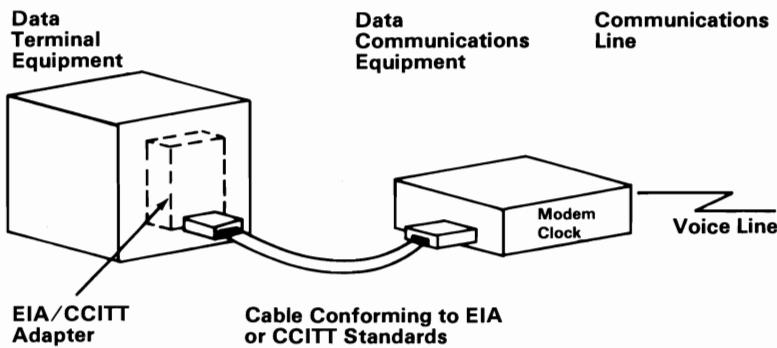
<b>Hardware</b> .....	<b>8-3</b>
<b>Establishing a Communications Link</b> .....	<b>8-5</b>

## Notes:

# Hardware

Information-processing equipment used for communication is called data terminal equipment (DTE.) Equipment used to connect the DTE to the communication line is called data communication equipment (DCE.)

An adapter connects the data terminal equipment to the data communication line as shown in the following figure:



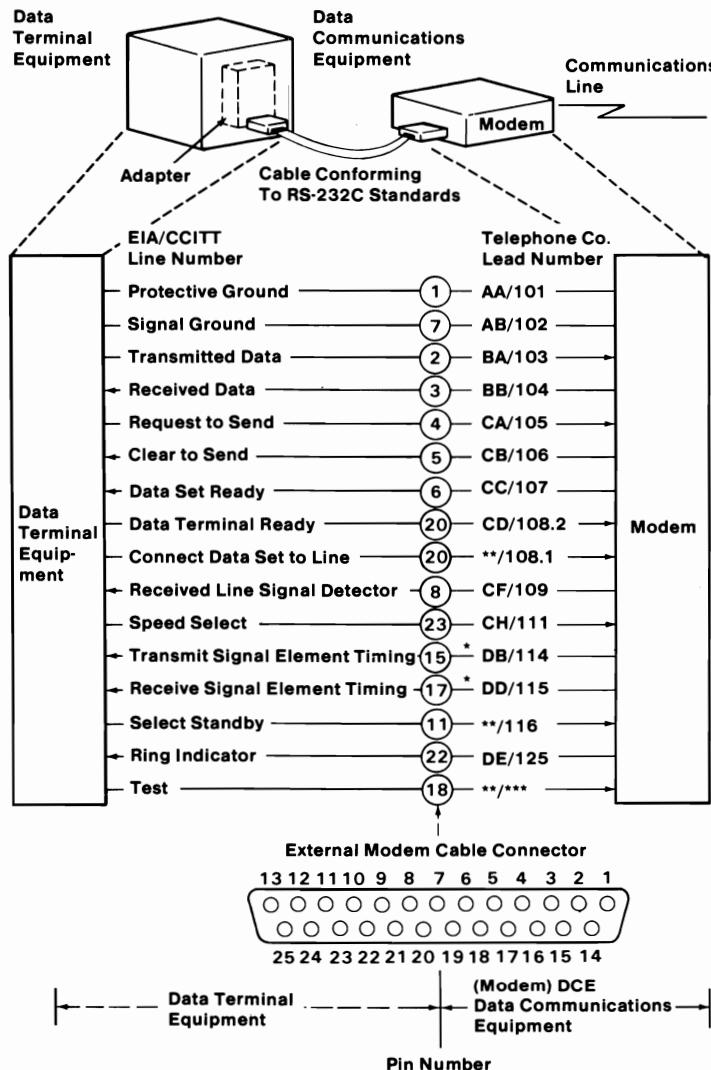
The EIA/CCITT adapter allows the data terminal equipment to be connected to the data communications equipment using EIA or CCITT standardized connections. An external modem is shown in the figure; however, other types of data communications equipment also can be connected to the data terminal equipment using EIA or CCITT standardized connections.

EIA standards are labeled RS-x (recommended standards-x), and CCITT standards are labeled V.x or X.x, where x is the number of the standard.

The EIA RS-232 interface standard defines the connector type, pin numbers, line names, and signal levels used to connect data terminal equipment to data communications equipment for the purpose of transmitting and receiving data. Since the RS-232 standard was developed, it has been revised three times. The three revised standards are RS-232A, RS-232B, and the presently used RS-232C.

The CCITT V.24 interface standard is equivalent to the RS-232C standard; therefore, the descriptions of the EIA standards also apply to the CCITT standards.

The following is an illustration of data terminal equipment connected to an external modem using connections defined by the RS-232C interface standard:



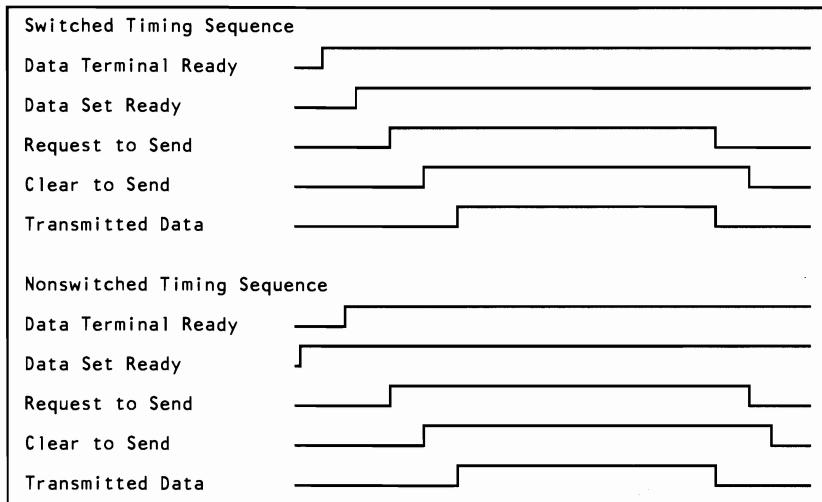
\*Not used when business machine clocking is used.

\*\*Not standardized by EIA (Electronics Industry Association).

\*\*\*Not standardized by CCITT

# Establishing a Communications Link

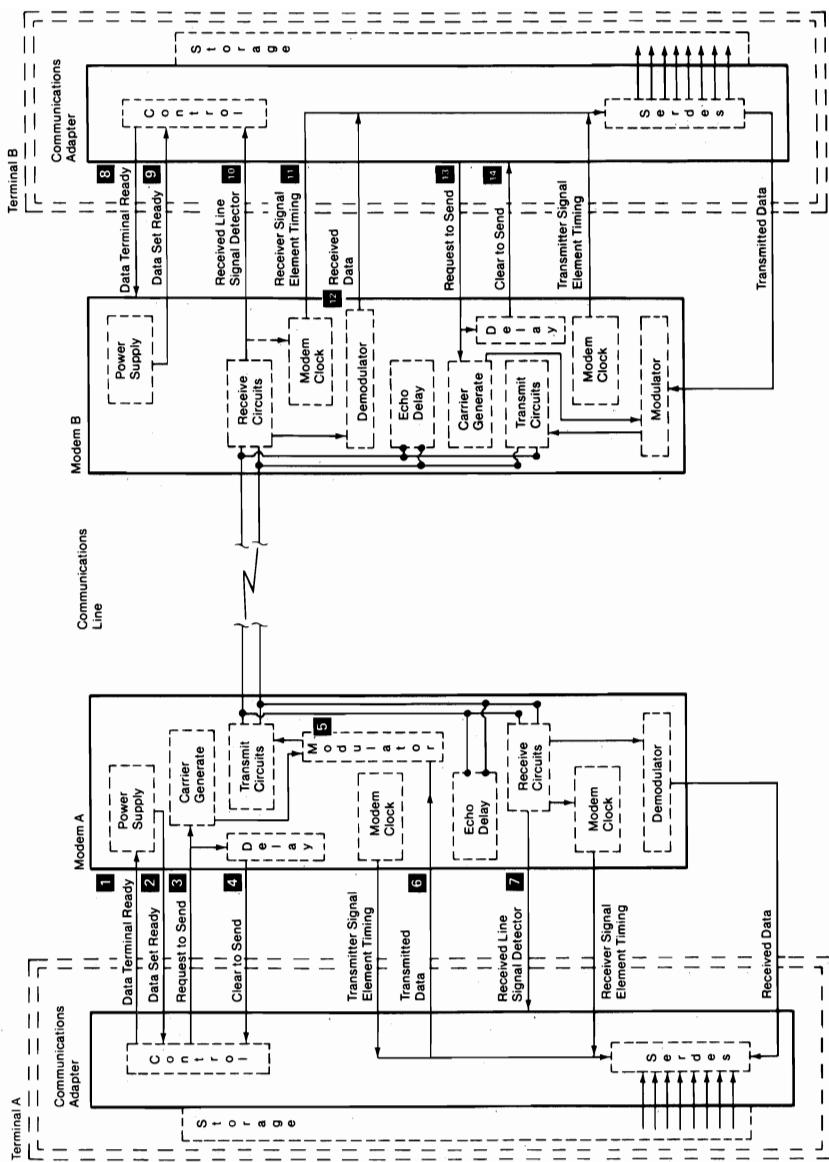
The following bar graphs represent normal timing sequences of operation during the establishment of communication for both switched (dial-up) and nonswitched (direct line) networks.



The following examples show how a link is established on a nonswitched point-to-point line, a nonswitched multipoint line, and a switched point-to-point line.

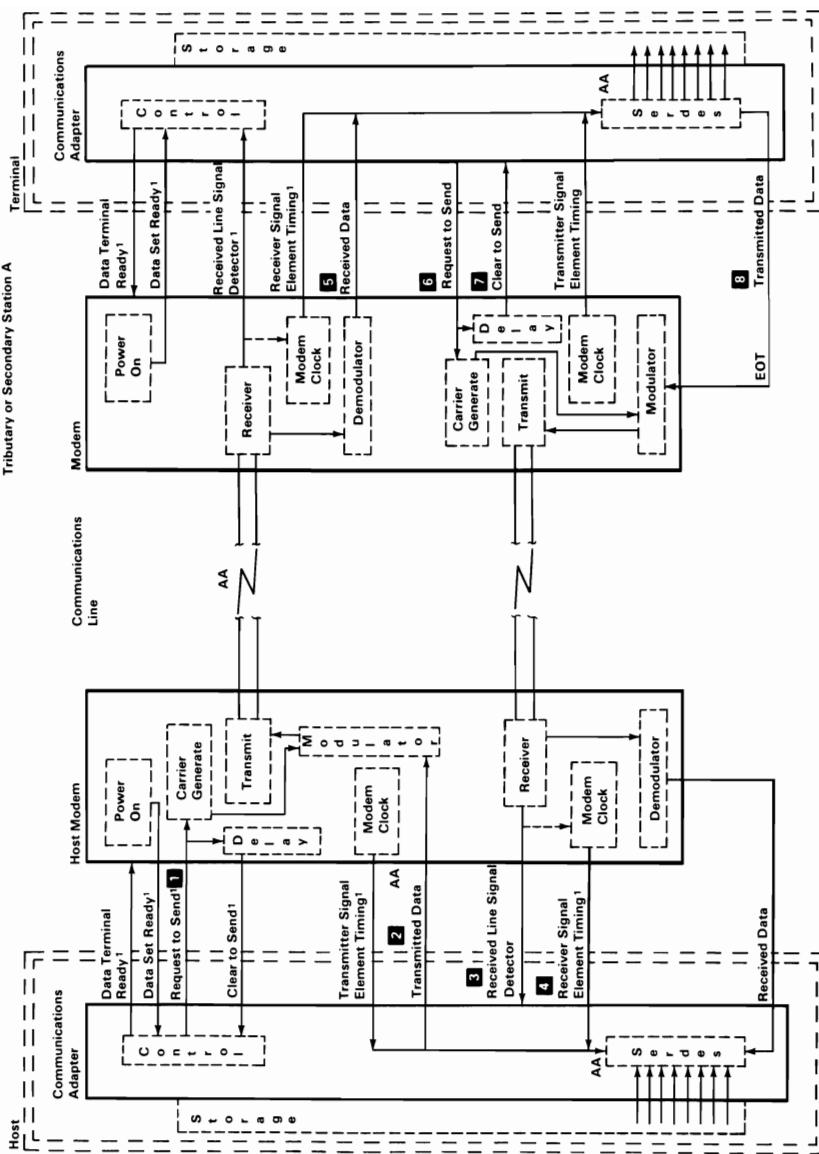
## Establishing a Link on a Nonswitched Point-to-Point Line

- The terminals at both locations activate the data terminal ready' lines **1** and **8**.
- Normally the 'data set ready' lines **2** and **9** from the modems are active whenever the modems are powered on.
- Terminal A activates the 'request to send' line **3**, which causes the modem at terminal A to generate a carrier signal.
- Modem B detects the carrier, and activates the 'received line signal detector' line (sometimes called data carrier detect) **10**. Modem B also activates the 'receiver signal element timing' line (sometimes called receive clock) **11** to send receive clock signals to the terminal. Some modems activate the clock signals whenever the modem is powered on.
- After a specified delay, modem A activates the 'clear to send' line **4**, which indicates to terminal A that the modem is ready to transmit data.
- Terminal A serializes the data to be transmitted (through the serdes) and transmits the data one bit at a time (synchronized by the transmit clock) onto the 'transmitted data' line **6** to the modem.
- The modem modulates the carrier signal with the data and transmits it to the modem B **5**.
- Modem B demodulates the data from the carrier signal and sends it to terminal B on the 'received data' line **12**.
- Terminal B deserializes the data (through the serdes) using the receive clock signals (on the 'receiver signal element timing' line) **11** from the modem.
- After terminal A completes its transmission, it deactivates the 'request to send' line **3**, which causes the modem to turn off the carrier and deactivate the 'clear to send' line **4**.
- Terminal A and modem A now become receivers and wait for a response from terminal B, indicating that all data has reached terminal B. Modem A begins an echo delay (50 to 150 milliseconds) to ensure that all echoes on the line have diminished before it begins receiving. An echo is a reflection of the transmitted signal. If the transmitting modem changed to receive too soon, it could receive a reflection (echo) of the signal it just transmitted.
- Modem B deactivates the 'received line signal detector' line **10** and, if necessary, deactivates the receive clock signals on the 'receiver signal element timing' line **11**.
- Terminal B now becomes the transmitter to respond to the request from terminal A. To transmit data, terminal B activates the 'request to send' line **13**, which causes modem B to transmit a carrier to modem A.
- Modem B begins a delay that is longer than the echo delay at modem A before turning on the 'clear to send' line. The longer delay (called request-to-send to clear-to-send delay) ensures that modem A is ready to receive when terminal B begins transmitting data. After the delay, modem B activates the 'clear to send' line **14** to indicate that terminal B can begin transmitting its response.
- After the echo delay at modem A, modem A senses the carrier from modem B (the carrier was activated in step 13 when terminal B activated the 'request to send' line) and activates the 'received line signal detector' line **7** to terminal A.
- Modem A and terminal A are now ready to receive the response from terminal B. Remember, the response was not transmitted until after the request-to-send to clear-to-send delay at modem B (step 14).



## Establishing a Link on a Nonswitched Multipoint Line

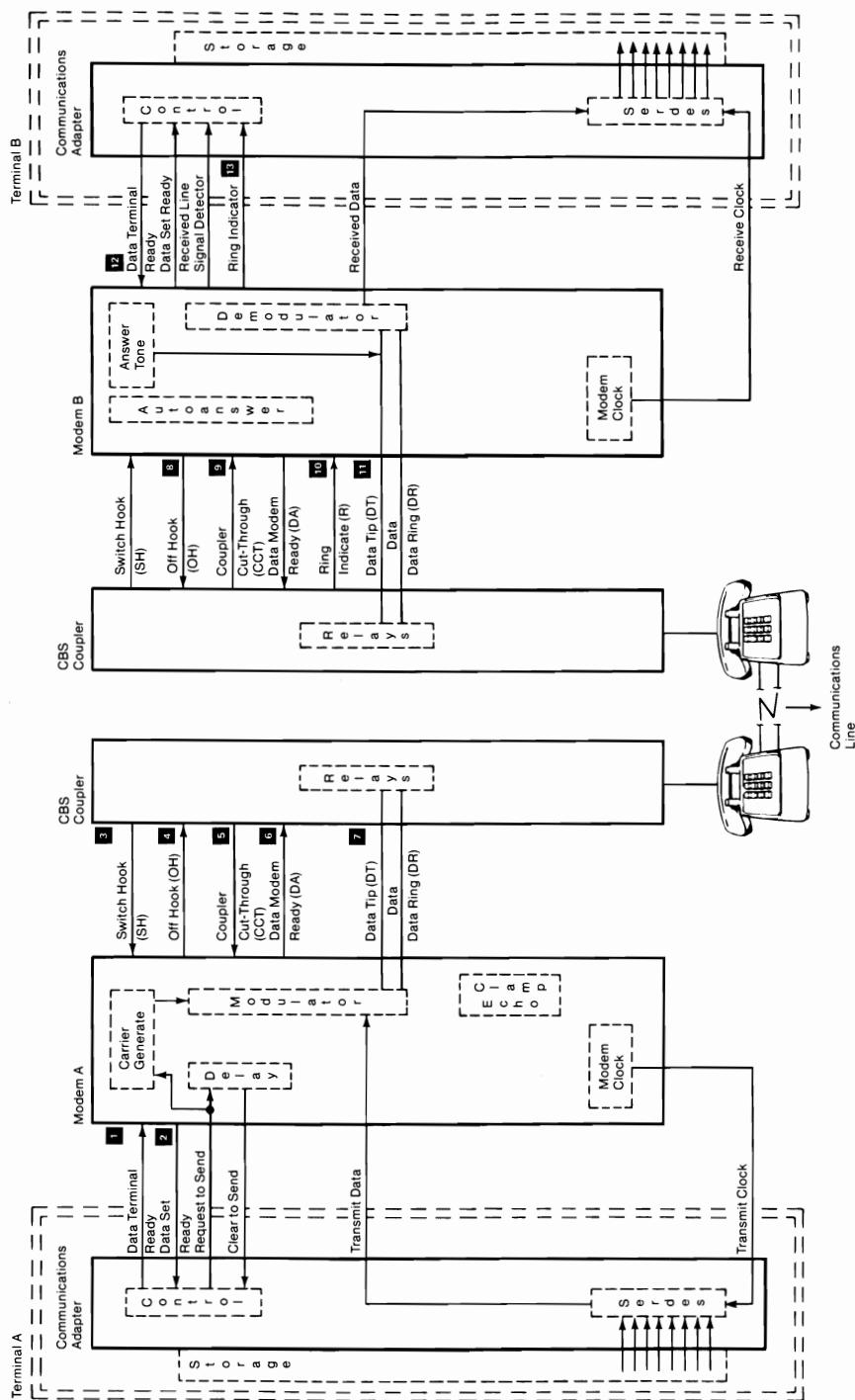
1. The control station serializes the address for the tributary or secondary station (AA) and sends its address to the modem on the 'transmitted data' line **2**.
2. Since the 'request to send' line and, therefore, the modem carrier, is active continuously **1**, the modem immediately modulates the carrier with the address, and, thus, the address is transmitted to all modems on the line.
3. All tributary modems, including the modem for station A, demodulate the address and send it to their terminals on the 'received data' line **5**.
4. Only station A responds to the address; the other stations ignore the address and continue monitoring their 'received data' line. To respond to the poll, station A activates its 'request to send' line **6** which causes the modem to begin transmitting a carrier signal.
5. The control station's modem receives the carrier and activates the 'received line signal detector' line **3** and the 'receiver signal element timing' line **4** (to send clock signals to the control station). Some modems activate the clock signals as soon as they are powered on.
6. After a short delay to allow the control station modem to receive the carrier, the tributary modem activates the 'clear to send' line **7**.
7. When station A detects the active 'clear to send' line, it transmits its response. (For this example, assume that station A has no data to send; therefore, it transmits an EOT **8**.)
8. After transmitting the EOT, station A deactivates the 'request to send' line **6**. This causes the modem to deactivate the carrier and the 'clear to send' line **7**.
9. When the modem at the control station (host) detects the absence of the carrier, it deactivates the 'received line signal detector' line **3**.
10. Tributary station A is now in receive mode waiting for the next poll or select transmission from the control station.



<sup>1</sup>These lines are active continuously.

## Establishing a Link on a Switched Point-to-Point Line

1. Terminal A is in communications mode; therefore, the 'data terminal ready' line **1** is active. Terminal B is in communication mode waiting for a call from terminal A.
2. When the terminal A operator lifts the telephone handset, the 'switch hook' line from the coupler is activated **3**.
3. Modem A detects the 'switch hook' line and activates the 'off hook' line **4**, which causes the coupler to connect the telephone set to the line and activate the 'coupler cut-through' line **5** to the modem.
4. Modem A activates the 'data modem ready' line **6** to the coupler (the 'data modem ready' line is on continuously in some modems).
5. The terminal A operator sets the exclusion key or talk/data switch to the talk position to connect the handset to the communications line. The operator then dials the terminal B number.
6. When the telephone at terminal B rings, the coupler activates the 'ring indicate' line to modem B **10**. Modem B indicates that the 'ring indicate' line was activated by activating the 'ring indicator' line **13** to terminal B.
7. Terminal B activates the 'data terminal ready' line to modem B **12**, which activates the autoanswer circuits in modem B. (The 'data terminal ready' line might already be active in some terminals.)
8. The autoanswer circuits in modem B activate the 'off hook' line to the coupler **8**.
9. The coupler connects modem B to the communications line through the 'data tip' and 'data ring' lines **11** and activates the 'coupler cut-through' line **9** to the modem. Modem B then transmits an answer tone to terminal A.
10. The terminal A operator hears the tone and sets the exclusion key or talk/data switch to the data position (or performs an equivalent operation) to connect modem A to the communications line through the 'data tip' and 'data ring' lines **7**.
11. The coupler at terminal A deactivates the 'switch hook' line **3**. This causes modem A to activate the 'data set ready' line **2**, indicating to terminal A that the modem is connected to the communications line. The sequence of the remaining steps to establish the data link is the same as the sequence required on a nonswitched point-to-point line. When the terminals have completed their transmission, they both deactivate the 'data terminal ready' line to disconnect the modems from the line.



## Notes:

# SECTION 9. IBM PERSONAL COMPUTER COMPATIBILITY

## Contents

<b>Hardware Considerations</b> .....	<b>9-3</b>
System Board .....	9-3
Fixed Disk Drive .....	9-5
Diskette Drive Compatibility .....	9-5
Copy Protection .....	9-5
Bypassing BIOS .....	9-6
Diskette Drive Controls .....	9-6
Write Current Control .....	9-6
<b>Application Guidelines</b> .....	<b>9-7</b>
High-Level Language Considerations .....	9-7
Assembler Language Programming Considerations .....	9-8
Multitasking Provisions .....	9-16
Interfaces .....	9-16
Classes .....	9-17
Time-Outs .....	9-19
Machine-Sensitive Code .....	9-19

## Notes:

This section describes the differences among the members of the IBM Personal Computer family. It also contains information necessary to design hardware and programs that will be compatible with all members of the IBM Personal Computer family.

## Hardware Considerations

To design compatible hardware or programs, you must consider hardware differences among the IBM Personal Computers. The following are hardware features of the IBM Personal Computer AT that are not supported by all of the IBM Personal Computer family.

### System Board

The IBM Personal Computer AT system board uses an Intel 80286 Microprocessor. This microprocessor is compatible with the 80287 Math Coprocessor used in the Personal Computer AT, and is generally compatible with the Intel 8088 Microprocessor used in other IBM Personal Computers.

The following table identifies the microprocessor and describes the I/O channel used with each type of IBM Personal Computer.

System Name	System Unit Microprocessor	I/O Channel Description
Personal Computer	8088	5 62-Pin
PCjr	8088	Not Compatible
Personal Computer XT	8088	8 62-Pin
Portable Personal Computer	8088	8 62-Pin
Personal Computer AT	80286	2 62-pin 6 98-Pin (62 Pin + 36 Pin)

### System Hardware Identification Chart

The faster processing capability of the 80286, compared to the 8088, creates special programming considerations, which are discussed later in this section under "Application Guidelines."

Some adapters use a 36-pin connector in addition to the 62-pin connector. Adapters designed to use the 36-pin connectors are not compatible with all members of the IBM Personal Computer family. Refer to the "System to Adapter Compatibility Chart" in the *Technical Reference Options and Adapters* manual, Volume 1, to identify the adapters supported by each system. The IBM Personal Computer AT does not support an expansion unit.

On the I/O channel:

- The system clock signal should be used only for synchronization and not for applications requiring a fixed frequency.
- The 14.31818-MHz oscillator is not synchronous with the system clock.
- The ALE signal is activated during DMA cycles.
- The -IOW signal is not active during refresh cycles.
- Pin B04 supports IRQ 9.

## Fixed Disk Drive

Reading from and writing to this drive is initiated in the same way as with other IBM Personal Computers; however, the Fixed Disk and Diskette Drive Adapter may be addressed from different BIOS locations.

## Diskette Drive Compatibility

The following chart shows the read, write, and format capabilities for each of the diskette drives used by IBM Personal Computers.

Diskette Drive Name	160/180K Mode	320/360K Mode	1.2M Mode
5-1/4 In. Diskette Drive:			
Type 1	R W F	---	---
Type 2	R W F	R W F	---
Type 3	R W F	R W F	---
Slimline Diskette Drive	R W F	R W F	---
Double Sided Diskette Drive	R W F	R W F	---
High Capacity Diskette Drive	R W*	R W*	R W F
R-Read W-Write F-Format W*-If a diskette is formatted in either 160/180K mode or 320/360K mode and written on by a High Capacity Drive, that diskette may be read by only a High Capacity Drive.			

### Diskette Drive Compatibility Chart

**Note:** Diskettes designed for the 1.2M mode may not be used in either a 160/180K or a 320/360K diskette drive.

## Copy Protection

The following methods of copy protection may not work on systems using the High Capacity Diskette Drive:

- Bypassing BIOS

- Diskette drive controls
- Write current control

## Bypassing BIOS

Copy protection that tries to bypass the following BIOS routines will not work on the High Capacity Diskette Drive:

**Track Density:** The High Capacity Diskette Drive records tracks at a density of 96 TPI (tracks per inch). This drive has to double-step in the 48 TPI mode, which is performed by BIOS.

**Data Transfer Rate:** BIOS selects the proper data transfer rate for the media being used.

**Disk Base:** Copy protection, which creates its own disk base will not work on the High Capacity Diskette Drive.

## Diskette Drive Controls

Copy protection that uses the following will not work on the High Capacity Diskette Drive:

**Rotational Speed:** The time between two events on a diskette is controlled by the Fixed Disk and Diskette Drive Adapter.

**Access Time:** Diskette BIOS routines must set the track-to-track access time for the different types of media used on the IBM Personal Computer AT.

**Head Geometry:** See "Diskette Drive Compatibility" on page 9-5

**Diskette Change Signal:** Copy protection may not be able to reset this signal.

## Write Current Control

Copy protection that uses write current control will not work because the Fixed Disk and Diskette Drive Adapter selects the proper write current for the media being used.

## Application Guidelines

The following information should be used to develop application programs for the IBM Personal Computer family.

### High-Level Language Considerations

The IBM-supported languages of BASIC, FORTRAN, COBOL, Pascal, and APL are the best choices for writing compatible programs.

If a program uses specific features of the hardware, that program may not be compatible with all IBM Personal Computers. Specifically, the use of assembler language subroutines or hardware-specific commands (In, Out, Peek, Poke, ...) must follow the assembler language rules (see "Assembler Language Programming Considerations" on page 9-8 ).

Any program that requires precise timing information should obtain it through a DOS or language interface; for example, TIME\$ in BASIC. If greater precision is required, the assembler techniques in "Assembler Language Programming Considerations" are available. The use of programming loops may prevent a program from being compatible with other IBM Personal Computers.

# Assembler Language Programming Considerations

The following OP codes work differently on systems using the 80286 microprocessor than they do on systems using the 8088 microprocessor.

- If the system microprocessor executes a POPF instruction in either the real or the virtual address mode with  $CPL \leq IOPL$ , 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    ; CALL within segment
E8 FB FF  CALL $-2  ; 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 in GDT, LDT, or IDT, beyond the defined limit of the descriptor table.
- Exception #GP(0) from attempted read or write (except for "PUSH" onto stack) beyond the defined limit of segment.
- Exception #SS(0) from attempted "PUSH" below the defined limit of the stack segment.

Restarting applications which generate the above exceptions may result in errors.

In the protected mode, when any of the null selector values (0000H, 0001H, 0002H, 0003H) are loaded into the DS or ES registers via a MOV or POP instruction or a task switch, the 80286 always loads the null selector 0000H into the corresponding register.

If a coprocessor (80287) operand is read from an "executable and readable" and conforming (ERC) code segment, and the coprocessor operand is sufficiently near the segment's limit that the second or subsequent byte lies outside the limit, no protection exception #9 will be generated.

The following correctly describes the operation of all 80286 parts:

- Instructions longer than 10 bytes (instructions using multiple redundant prefixes) generate exception #13 (General Purpose Exception) in both the real and protected modes.
- If the second operand of an ARPL instruction is a null selector, the instruction generates an exception #13.

Assembler language programs should perform all I/O operations through ROM BIOS or DOS function calls.

- Program interrupts are used for access to these functions. This practice removes the absolute addressing from the program. Only the interrupt number is required.
- The coprocessor detects six different exception conditions that can occur during instruction execution. If the appropriate exception mask within the coprocessor is not set, the coprocessor sets its error signal. This error signal generates a hardware interrupt (interrupt 13) and causes the 'busy' signal to the coprocessor to be held in the busy state. The 'busy' signal may be cleared by an 8-bit I/O Write command to address hex F0 with D0 through D7 equal to 0.

The power-on-self-test code in the system ROM enables hardware IRQ 13 and sets up its vector to point to a routine in ROM. The ROM routine clears the 'busy' signal latch and then transfers control to the address pointed to by the NMI interrupt vector. This allows code written for any IBM Personal Computer to work on an IBM Personal Computer AT. The NMI interrupt handler should read the coprocessor's status to determine if the NMI was caused by the coprocessor. If the interrupt was not generated by the coprocessor, control should be passed to the original NMI interrupt handler.

- Back to back I/O commands to the same I/O ports will not permit enough recovery time for I/O chips. To ensure enough time, a JMP SHORT \$+2 must be inserted between IN/OUT instructions to the same I/O chip.

**Note:** MOV AL,AH type instruction does not allow enough recovery time. An example of the correct procedure follows:

```
OUT  IO_ADD,AL
JMP  SHORT $+2
MOV  AL,AH
OUT  IO_ADD,AL
```

- In systems using the 80286 microprocessor, IRQ 9 is redirected to INT hex 0A (hardware IRQ 2). This insures

that hardware designed to use IRQ 2 will operate in the IBM Personal Computer AT.

- The system can mask hardware sensitivity. New devices can change the ROM BIOS to accept the same programming interface on the new device.
- In cases where BIOS provides parameter tables, such as for video or diskette, a program may substitute new parameter values by building a new copy of the table and changing the vector to point to that table. However, the program should copy the current table, using the current vector, and then modify those locations in the table that need to be changed. In this way, the program will not inadvertently change any values that should be left the same.
- Disk Base consists of 11 parameters required for diskette operation. They are pointed at by the data variable, Disk Pointer, at absolute address 0:78. It is strongly recommended that the values supplied in ROM be used. If it becomes necessary to modify any of the parameters, build another parameter block and modify the address in Disk Pointer to point to the new block.

The parameters were established to operate both the High Capacity Diskette Drive and the Double Sided Diskette Drive. Three of the parameters in this table are under control of BIOS in the following situations.

The Gap Length Parameter is no longer retrieved from the parameter block.

The gap length used during diskette read, write, and verify operations is derived from within diskette BIOS.

The gap length for format operations is still obtained from the parameter block.

Special considerations are required for formatting operations. See the prolog of Diskette BIOS for the required details. If a parameter block contains a head settle time parameter value of 0 milliseconds, and a write operation is being performed, at least 15 milliseconds of head settle time will be enforced

for a High Capacity Diskette Drive and 20 milliseconds will be enforced for a Double Sided Diskette Drive. If a parameter block contains a motor start wait parameter of less than 1 second for a write or format operation of 625 milliseconds for a read or verify operation, Diskette BIOS will enforce those times listed above.

- The following procedure is used to determine the type of media inserted in the High Capacity Diskette Drive:
  1. Read Track 0, Head 0, Sector 1 to allow diskette BIOS to establish the media/drive combination. If this is successful, continue with the next step.
  2. Read Track 0, Sector 15. If an error occurs, a double sided diskette is in the drive.

**Note:** Refer to the *DOS Technical Reference* manual for the File Allocation Table (FAT) parameters for single- and double-sided diskettes.

If a successful read occurs, a high capacity diskette is in the drive.

3. If Step 1 fails, issue the reset function (AH=0) to diskette BIOS and retry. If a successful read cannot be done, the media needs to be formatted or is defective.

ROM BIOS and DOS do not provide for all functions. The following are the allowable I/O operations with which IBM will maintain compatibility in future systems.

- Control of the sound, using port hex 61, and the sound channel of the timer/counter. A program can control timer/counter channels 0 and 2, ports hex 40, 42, and 43. A program must not change the value in port hex 41, because this port controls the dynamic-memory refresh. Channel 0 provides the time-of-day interrupt, and can also be used for timing short intervals. Channel 2 of the timer/counter is the output for the speaker and cassette ports. This channel may also be used for timing short intervals, although it cannot interrupt at the end of the period.

- Control of the Game Control Adapter, port hex 201
  - Note:** Programs should use the timer for delay on the paddle input rather than a program loop.
- Interrupt Mask Register (IMR), port hex 21, can be used to selectively mask and unmask the hardware features.

The following information pertains to absolute memory locations.

- Interrupt Vectors Segment (hex 0)--A program may change these to point at different processing routines. When an interrupt vector is modified, the original value should be retained. If the interrupt, either hardware or program, is not directed toward this device handler, the request should be passed to the next item in the list.
- Video Display Buffers (hex B0000 and B8000)-- For each mode of operation defined in the video display BIOS, the memory map will remain the same. For example, the bit map for the 320 x 200 medium-resolution graphics mode of the Color/Graphics Monitor adapter will be retained on any future adapter that supports that mode. If the bit map is modified, a different mode number will be used.
- ROM BIOS Data Area (hex 40:0)--Any variables in this area will retain their current definition, whenever it is reasonable to do so. IBM may use these data areas for other purposes when the variable no longer has meaning in the system. In general, ROM BIOS data variables should be read or modified through BIOS calls whenever possible, and not with direct access to the variable.

A program that requires timing information should use either the time-of-day clock or the timing channels of the timer/counter. The input frequency to the timer will be maintained at 1.19 MHz, providing a constant time reference. Program loops should be avoided.

Programs that use copy protection schemes should use the ROM BIOS diskette calls to read and verify the diskette and should not be timer dependent. Any method can be used to create the diskette, although manufacturing capability should be considered.

The verifying program can look at the diskette controller's status bytes in the ROM BIOS data area for additional information about embedded errors. More information about copy protection may be found on page 9-5 under "Copy Protection".

Any DOS program must be relocatable and insensitive to the size of DOS or its own load addresses. A program's memory requirement should be identified and contiguous with the load module. A program should not assume that all of memory is available to it.

There are several 80286 instructions that, when executed, lock out external bus signals. DMA requests are not honored during the execution of these instructions. Consecutive instructions of this type prevent DMA activity from the start of the first instruction to the end of the last instruction. To allow for necessary DMA cycles, as required by the diskette controller in a multitasking system, multiple lock-out instructions must be separated by JMP SHORT \$+2.

## Multitasking Provisions

The IBM Personal Computer AT BIOS contains a feature to assist multitasking implementation. "Hooks" are provided for a multitasking dispatcher. Whenever a busy (wait) loop occurs in the BIOS, a hook is provided for the program to break out of the loop. Also, whenever BIOS services an interrupt, a corresponding wait loop is exited, and another hook is provided. Thus a program may be written that employs the bulk of the device driver code. The following is valid only in the microprocessor's real address mode and must be taken by the code to allow this support.

The program is responsible for the serialization of access to the device driver. The BIOS code is not reentrant.

The program is responsible for matching corresponding wait and post calls.

## Interfaces

There are four interfaces to be used by the multitasking dispatcher:

### Startup

First, the startup code hooks interrupt hex 15. The dispatcher is responsible to check for function codes of AH= hex 90 or 91. The "Wait" and "Post" sections describe these codes. The dispatcher must pass all other functions to the previous user of interrupt hex 15. This can be done by a JMP or a CALL. If the function code is hex 90 or 91, the dispatcher should do the appropriate processing and return by the IRET instruction.

### Serialization

It is up to the multitasking system to ensure that the device driver code is used serially. Multiple entries into the code can result in serious errors.

## Wait (Busy)

Whenever the BIOS is about to enter a busy loop, it first issues an interrupt hex 15 with a function code of hex 90 in AH. This signals a wait condition. At this point, the dispatcher should save the task status and dispatch another task. This allows overlapped execution of tasks when the hardware is busy. The following is an outline of the code that has been added to the BIOS to perform this function.

```
MOV AX, 90XXH      ; wait code in AH and
                    ; type code in AL
INT 15H           ; issue call
JC  TIMEOUT       ; optional: for time-out or
                    ; if carry is set, time-out
                    ; occurred
NORMAL TIMEOUT LOGIC ; normal time-out
```

## Post (Interrupt)

Whenever the BIOS has set an interrupt flag for a corresponding busy loop, an interrupt 15 occurs with a function code of hex 91 in AH. This signals a post condition. At this point, the dispatcher should set the task status to "ready to run" and return to the interrupt routine. The following is an outline of the code added to BIOS that performs this function.

```
MOV AX, 91XXH      ; post code AH and
                    ; type code AL
INT 15H           ; issue call
```

## Classes

The following types of wait loops are supported:

- The class for hex 0 to 7F is serially reusable. This means that for the devices that use these codes, access to the BIOS must be restricted to only one task at a time.

- The class for hex 80 to BF is reentrant. There is no restriction on the number of tasks that may access the device.
- The class for hex C0 to FF is non-interrupt. There is no corresponding interrupt for the wait loop. Therefore, it is the responsibility of the dispatcher to determine what satisfies this condition to exit the loop.

## Function Code Classes

Type Code (AL)	Description
00H->7FH	Serially reusable devices; operating system must serialize access
80H->0BFH	Reentrant devices; ES:BX is used to distinguish different calls (multiple I/O calls are allowed simultaneously)
0C0H->0FH	Wait only calls; there is no complementary POST for these waits--these are time-out only. Times are function-number dependent.

## Function Code Assignments

The following are specific assignments for the IBM Personal Computer AT BIOS. Times are approximate. They are grouped according to the classes described under "Function Code Classes".

Type Code (AL)	Time-out	Description
00H	yes (6 second)	fixed disk
01H	yes (2 second)	diskette
02H	no	keyboard
0FDH	yes (1 second-write)	diskette motor start

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

### Machine Identification Code

IBM will define methods for uniquely determining the specific machine type or I/O feature for any new device.

## Notes:

# Glossary

This glossary includes definitions developed by the American National Standards Institute (ANSI) and the International Organization for Standardization (ISO). This material is reproduced from the *American National Dictionary for Information Processing*, copyright 1977 by the Computer and Business Equipment Manufacturers Association, copies of which may be purchased from the American National Standards Institute, 1430 Broadway, New York, New York 10018.

$\mu$ . 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.

**CAS.** Column address strobe.

**cathode ray tube (CRT).** A vacuum tube in which a stream of electrons is projected onto a fluorescent screen producing a luminous spot. The location of the spot can be controlled.

**cathode ray tube display (CRT display).** (1) A CRT used for displaying data. For example, the electron beam can be controlled to form alphanumeric data by use of a dot matrix.

(2) Synonymous with monitor.

**CCITT.** International Telegraph and Telephone Consultative Committee.

**Celsius (C).** A temperature scale. Contrast with Fahrenheit (F).

**central processing unit (CPU).** Term for processing unit.

**channel.** A path along which signals can be sent; for example, data channel, output channel.

**character generator.** (1) In computer graphics, a functional unit that converts the coded representation of a graphic character into the shape of the character for display. (2) In word processing, the means within equipment for generating visual characters or symbols from coded data.

**character set.** (1) A finite set of different characters upon which agreement has been reached and that is considered complete for some purpose. (2) A set of unique representations called characters. (3) A defined collection of characters.

**characters per second (cps).** A standard unit of measurement for the speed at which a printer prints.

**check key.** A group of characters, derived from and appended to a data item, that can be used to detect errors in the data item during processing.

**clipping.** In computer graphics, removing parts of a display image that lie outside a window.

**closed circuit.** A continuous unbroken circuit; that is, one in which current can flow. Contrast with open circuit.

**CMOS.** Complementary metal oxide semiconductor.

**code.** (1) A set of unambiguous rules specifying the manner in which data may be represented in a discrete form. Synonymous with coding scheme. (2) A set of items, such as abbreviations, representing the members of another set. (3) To represent data or a computer program in a symbolic form that can be accepted by a data processor. (4) Loosely, one or more computer programs, or part of a computer program.

**coding scheme.** Synonym for code.

**collector.** An element in a transistor toward which current flows.

**color cone.** An arrangement of the visible colors on the surface of a double-ended cone where lightness varies along the axis of the cone, and hue varies around the circumference. Lightness includes both the intensity and saturation of color.

**column address strobe (CAS).** A signal that latches the column addresses in a memory chip.

**compile.** (1) To translate a computer program expressed in a problem-oriented language into a computer-oriented language. (2) To prepare a machine-language program from a computer program written in another programming language by making use of the overall logic structure of the program, or generating more

than one computer instruction for each symbolic statement, or both, as well as performing the function of an assembler.

**complement.** A number that can be derived from a specified number by subtracting it from a second specified number.

**complementary metal oxide semiconductor (CMOS).** A logic circuit family that uses very little power. It works with a wide range of power supply voltages.

**computer.** A functional unit that can perform substantial computation, including numerous arithmetic operations or logic operations, without human intervention during a run.

**computer instruction code.** A code used to represent the instructions in an instruction set. Synonymous with machine code.

**computer program.** A sequence of instructions suitable for processing by a computer.

**computer word.** A word stored in one computer location and capable of being treated as a unit.

**configuration.** (1) The arrangement of a computer system or network as defined by the nature, number, and the chief characteristics of its functional units. More specifically, the term configuration may refer to a hardware configuration or a software configuration. (2) The devices and programs that make up a system, subsystem, or network.

**conjunction.** Synonym for AND operation.

**contiguous.** Touching or joining at the edge or boundary; adjacent.

**control character.** A character whose occurrence in a particular context initiates, modifies, or stops a control operation.

**control operation.** An action that affects the recording, processing, transmission, or interpretation of data; for example, starting or stopping a process, carriage return, font change, rewind, and end of transmission.

**control storage.** A portion of storage that contains microcode.

**coordinate space.** In computer graphics, a system of Cartesian coordinates in which an object is defined.

**cps.** Characters per second.

**CPU.** Central processing unit.

**CRC.** Cyclic redundancy check.

**CRT.** Cathode ray tube.

**CRT display.** Cathode ray tube display.

**CTS.** Clear to send. Associated with modem control.

**cursor.** (1) In computer graphics, a movable marker that is used to indicate position on a display. (2) A displayed symbol that acts as a marker to help the user locate a point in text, in a system command, or in storage. (3) A movable spot of light on the screen of a display device, usually indicating where the next character is to be entered, replaced, or deleted.

**cyclic redundancy check (CRC).** (1) A redundancy check in which the check key is generated by a cyclic algorithm. (2) A system of error checking performed at both the sending and receiving station after a block-check character has been accumulated.

**cylinder.** (1) The set of all tracks with the same nominal distance from the axis about which the disk rotates. (2) The tracks of a disk storage device that can be accessed without repositioning the access mechanism.

**daisy-chained cable.** A type of cable that has two or more connectors attached in series.

**data.** (1) A representation of facts, concepts, or instructions in a formalized manner suitable for communication, interpretation, or

processing by human or automatic means. (2) Any representations, such as characters or analog quantities, to which meaning is, or might be assigned.

**data base.** A collection of data that can be immediately accessed and operated upon by a data processing system for a specific purpose.

**data processing system.** A system that performs input, processing, storage, output, and control functions to accomplish a sequence of operations on data.

**data transmission.** Synonym for transmission.

**dB.** Decibel.

**dBa.** Adjusted decibels.

**dc.** Direct current.

**debounce.** (1) An electronic means of overcoming the make/break bounce of switches to obtain one smooth change of signal level. (2) The elimination of undesired signal variations caused by mechanically generated signals from contacts.

**decibel.** (1) A unit that expresses the ratio of two power levels on a logarithmic scale. (2) A unit for measuring relative power.

**decoupling capacitor.** A capacitor that provides a low impedance path to ground to prevent common coupling between circuits.

**Deutsche Industrie Norm (DIN).** (1) German Industrial Norm. (2) The committee that sets German dimension standards.

**digit.** (1) A graphic character that represents an integer; for example, one of the characters 0 to 9. (2) A symbol that represents one of the non-negative integers smaller than the radix. For example, in decimal notation, a digit is one of the characters 0 to 9.

**digital.** (1) Pertaining to data in the form of digits. (2) Contrast with analog.

**DIN.** Deutsche Industrie Norm.

**DIN connector.** One of the connectors specified by the DIN committee.

**DIP.** Dual in-line package.

**DIP switch.** One of a set of small switches mounted in a dual in-line package.

**direct current (dc).** A current that always flows in one direction.

**direct memory access (DMA).** A method of transferring data between main storage and I/O devices that does not require processor intervention.

**disable.** To stop the operation of a circuit or device.

**disabled.** Pertaining to a state of a processing unit that prevents the occurrence of certain types of interruptions. Synonymous with masked.

**disk.** Loosely, a magnetic disk.

**diskette.** A thin, flexible magnetic disk and a semirigid protective jacket, in which the disk is permanently enclosed. Synonymous with flexible disk.

**diskette drive.** A device for storing data on and retrieving data from a diskette.

**display.** (1) A visual presentation of data. (2) A device for visual presentation of information on any temporary character imaging device. (3) To present data visually. (4) See cathode ray tube display.

**display attribute.** In computer graphics, a particular property that is assigned to all or part of a display; for example, low intensity, green color, blinking status.

**display element.** In computer graphics, a basic graphic element that can be used to construct a display image; for example, a dot, a line segment, a character.

**display group.** In computer graphics, a collection of display elements that can be manipulated as a unit and that can be further combined to form larger groups.

**display image.** In computer graphics, a collection of display elements or display groups that are represented together at any one time in a display space.

**display space.** In computer graphics, that portion of a display surface available for a display image. The display space may be all or part of a display surface.

**display surface.** In computer graphics, that medium on which display images may appear; for example, the entire screen of a cathode ray tube.

**DMA.** Direct memory access.

**dot matrix.** (1) In computer graphics, a two-dimensional pattern of dots used for constructing a display image. This type of matrix can be used to represent characters by dots. (2) In word processing, a pattern of dots used to form characters. This term normally refers to a small section of a set of addressable points; for example, a representation of characters by dots.

**dot printer.** Synonym for matrix printer.

**dot-matrix character generator.** In computer graphics, a character generator that generates character images composed of dots.

**drawing primitive.** A group of commands that draw defined geometric shapes.

**DSR.** Data set ready. Associated with modem control.

**DTR.** In the IBM Personal Computer, data terminal ready. Associated with modem control.

**dual in-line package (DIP).** A widely used container for an integrated circuit. DIPs have pins in two parallel rows. The pins are spaced 1/10 inch apart. See also DIP switch.

**duplex.** (1) In data communication, pertaining to a simultaneous two-way independent transmission in both directions. (2) Contrast with half-duplex.

**duty cycle.** In the operation of a device, the ratio of on time to idle time. Duty cycle is expressed as a decimal or percentage.

**dynamic memory.** RAM using transistors and capacitors as the memory elements. This memory requires a refresh (recharge) cycle every few milliseconds. Contrast with static memory.

**EBCDIC.** Extended binary-coded decimal interchange code.

**ECC.** Error checking and correction.

**edge connector.** A terminal block with a number of contacts attached to the edge of a printed-circuit board to facilitate plugging into a foundation circuit.

**EIA.** Electronic Industries Association.

**electromagnet.** Any device that exhibits magnetism only while an electric current flows through it.

**enable.** To initiate the operation of a circuit or device.

**end of block (EOB).** A code that marks the end of a block of data.

**end of file (EOF).** An internal label, immediately following the last record of a file, signaling the end of that file. It may include control totals for comparison with counts accumulated during processing.

**end-of-text (ETX).** A transmission control character used to terminate text.

**end-of-transmission (EOT).** A transmission control character used to indicate the conclusion of a transmission, which may have included one or more texts and any associated message headings.

**end-of-transmission-block (ETB).** A transmission control character used to indicate the end of a transmission block of data when data is divided into such blocks for transmission purposes.

**EOB.** End of block.

**EOF.** End of file.

**EOT.** End-of-transmission.

**EPROM.** Erasable programmable read-only memory.

**erasable programmable read-only memory (EPROM).** A PROM in which the user can erase old information and enter new information.

**error checking and correction (ECC).** The detection and correction of all single-bit errors, plus the detection of double-bit and some multiple-bit errors.

**ESC.** The escape character.

**escape character (ESC).** A code extension character used, in some cases, with one or more succeeding characters to indicate by some convention or agreement that the coded representations following the character or the group of characters are to be

interpreted according to a different code or according to a different coded character set.

**ETB.** End-of-transmission-block.

**ETX.** End-of-text.

**extended binary-coded decimal interchange code (EBCDIC).** A set of 256 characters, each represented by eight bits.

**F.** Fahrenheit.

**Fahrenheit (F).** A temperature scale. Contrast with Celsius (C).

**falling edge.** Synonym for negative-going edge.

**FCC.** Federal Communications Commission.

**fetch.** To locate and load a quantity of data from storage.

**FF.** The form feed character.

**field.** (1) In a record, a specified area used for a particular category of data. (2) In a data base, the smallest unit of data that can be referred to.

**field-programmable logic sequencer (FPLS).** An integrated circuit containing a programmable, read-only memory that responds to external inputs and feedback of its own outputs.

**FIFO (first-in-first out).** A queuing technique in which the next item to be retrieved is the item that has been in the queue for the longest time.

**fixed disk drive.** In the IBM Personal Computer, a unit consisting of nonremovable magnetic disks, and a device for storing data on and retrieving data from the disks.

**flag.** (1) Any of various types of indicators used for identification. (2) A character that signals the occurrence of some condition, such as the end of a word. (3) Deprecated term for mark.

**flexible disk.** Synonym for diskette.

**flip-flop.** A circuit or device containing active elements, capable of assuming either one of two stable states at a given time.

**font.** A family or assortment of characters of a given size and style; for example, 10 point Press Roman medium.

**foreground.** (1) In multiprogramming, the environment in which high-priority programs are executed. (2) On a color display screen, the characters as opposed to the background.

**form feed.** (1) Paper movement used to bring an assigned part of a form to the printing position. (2) In word processing, a function that advances the typing position to the same character position on a predetermined line of the next form or page.

**form feed character.** A control character that causes the print or display position to move to the next predetermined first line on the next form, the next page, or the equivalent.

**format.** The arrangement or layout of data on a data medium.

**FPLS.** Field-programmable logic sequencer.

**frame.** (1) In SDLC, the vehicle for every command, every response, and all information that is transmitted using SDLC procedures. Each frame begins and ends with a flag. (2) In data transmission, the sequence of contiguous bits bracketed by and including beginning and ending flag sequences.

g. Gram.

**G.** (1) Prefix giga; 1,000,000,000. (2) When referring to computer storage capacity, 1,073,741,824. (1,073,741,824 = 2 to the 30th power.)

**gate.** (1) A combinational logic circuit having one output channel and one or more input channels, such that the output channel state is completely determined by the input channel states. (2) A signal that enables the passage of other signals through a circuit.

**Gb.** 1,073,741,824 bytes.

**general-purpose register.** A register, usually explicitly addressable within a set of registers, that can be used for different purposes; for example, as an accumulator, as an index register, or as a special handler of data.

**giga (G).** Prefix 1,000,000,000.

**gram (g).** A unit of weight (equivalent to 0.035 ounces).

**graphic.** A symbol produced by a process such as handwriting, drawing, or printing.

**graphic character.** A character, other than a control character, that is normally represented by a graphic.

**half-duplex.** (1) In data communication, pertaining to an alternate, one way at a time, independent transmission. (2) Contrast with duplex.

**hardware.** (1) Physical equipment used in data processing, as opposed to programs, procedures, rules, and associated documentation. (2) Contrast with software.

**head.** A device that reads, writes, or erases data on a storage medium; for example, a small electromagnet used to read, write, or erase data on a magnetic disk.

**hertz (Hz).** A unit of frequency equal to one cycle per second.

**hex.** Common abbreviation for hexadecimal.

**hexadecimal.** (1) Pertaining to a selection, choice, or condition that has 16 possible different values or states. These values or states are usually symbolized by the ten digits 0 through 9 and the six letters A through F. (2) Pertaining to a fixed radix numeration system having a radix of 16.

**high impedance state.** A state in which the output of a device is effectively isolated from the circuit.

**highlighting.** In computer graphics, emphasizing a given display group by changing its attributes relative to other display groups in the same display field.

**high-order position.** The leftmost position in a string of characters. See also most-significant digit.

**hither plane.** In computer graphics, a plane that is perpendicular to the line joining the viewing reference point and the view point and that lies between these two points. Any part of an object between the hither plane and the view point is not seen. See also yon plane.

**housekeeping.** Operations or routines that do not contribute directly to the solution of the problem but do contribute directly to the operation of the computer.

**Hz.** Hertz

**image.** A fully processed unit of operational data that is ready to be transmitted to a remote unit; when loaded into control storage in the remote unit, the image determines the operations of the unit.

**immediate instruction.** An instruction that contains within itself an operand for the operation specified, rather than an address of the operand.

**index register.** A register whose contents may be used to modify an operand address during the execution of computer instructions.

**indicator.** (1) A device that may be set into a prescribed state, usually according to the result of a previous process or on the occurrence of a specified condition in the equipment, and that usually gives a visual or other indication of the existence of the prescribed state, and that may in some cases be used to determine the selection among alternative processes; for example, an overflow indicator. (2) An item of data that may be interrogated to determine whether a particular condition has been satisfied in the execution of a computer program; for example, a switch indicator, an overflow indicator.

**inhibited.** (1) Pertaining to a state of a processing unit in which certain types of interruptions are not allowed to occur. (2) Pertaining to the state in which a transmission control unit or an audio response unit cannot accept incoming calls on a line.

**initialize.** To set counters, switches, addresses, or contents of storage to 0 or other starting values at the beginning of, or at prescribed points in, the operation of a computer routine.

**input/output (I/O).** (1) Pertaining to a device or to a channel that may be involved in an input process, and, at a different time, in an output process. In the English language, "input/output" may be used in place of such terms as "input/output data," "input/output signal," and "input/output terminals," when such usage is clear in a given context. (2) Pertaining to a device whose parts can be performing an input process and an output process at the same time. (3) Pertaining to either input or output, or both.

**instruction.** In a programming language, a meaningful expression that specifies one operation and identifies its operands, if any.

**instruction set.** The set of instructions of a computer, of a programming language, or of the programming languages in a programming system.

**intensity.** In computer graphics, the amount of light emitted at a display point

**interface.** A device that alters or converts actual electrical signals between distinct devices, programs, or systems.

**interleave.** To arrange parts of one sequence of things or events so that they alternate with parts of one or more other sequences of the same nature and so that each sequence retains its identity.

**interrupt.** (1) A suspension of a process, such as the execution of a computer program, caused by an event external to that process, and performed in such a way that the process can be resumed. (2) In a data transmission, to take an action at a receiving station that causes the transmitting station to terminate a transmission. (3) Synonymous with interruption.

**I/O.** Input/output.

**I/O area.** Synonym for buffer.

**irrecoverable error.** An error that makes recovery impossible without the use of recovery techniques external to the computer program or run.

**joystick.** In computer graphics, a lever that can pivot in all directions and that is used as a locator device.

**k.** Prefix kilo; 1000.

**K.** When referring to storage capacity, 1024. (1024 = 2 to the 10th power.)

**Kb.** 1024 bytes.

**key lock.** A device that deactivates the keyboard and locks the cover on for security.

**kg.** Kilogram; 1000 grams.

**kHz.** Kilohertz; 1000 hertz.

**kilo (k).** Prefix 1000

**kilogram (kg).** 1000 grams.

**kilohertz (kHz).** 1000 hertz

**latch.** (1) A simple logic-circuit storage element. (2) A feedback loop in sequential digital circuits used to maintain a state.

**least-significant digit.** The rightmost digit. See also low-order position.

**LED.** Light-emitting diode.

**light-emitting diode (LED).** A semiconductor device that gives off visible or infrared light when activated.

**load.** In programming, to enter data into storage or working registers.

**look-up table (LUT).** (1) A technique for mapping one set of values into a larger set of values. (2) In computer graphics, a table that assigns a color value (red, green, blue intensities) to a color index.

**low power Schottky TTL.** A version (LS series) of TTL giving a good compromise between low power and high speed. See also transistor-transistor logic and Schottky TTL.

**low-order position.** The rightmost position in a string of characters. See also least-significant digit.

**luminance.** The luminous intensity per unit projected area of a given surface viewed from a given direction.

**LUT.** Look-up table.

**m.** (1) Prefix milli; 0.001. (2) Meter.

**M.** (1) Prefix mega; 1,000,000. (2) When referring to computer storage capacity, 1,048,576. (1,048,576 = 2 to the 20th power.)

**mA.** Milliampere; 0.001 ampere.

**machine code.** The machine language used for entering text and program instructions onto the recording medium or into storage and which is subsequently used for processing and printout.

**machine language.** (1) A language that is used directly by a machine. (2) Deprecated term for computer instruction code.

**magnetic disk.** (1) A flat circular plate with a magnetizable surface layer on which data can be stored by magnetic recording. (2) See also diskette.

**main storage.** (1) Program-addressable storage from which instructions and other data can be loaded directly into registers for subsequent execution or processing. (2) Contrast with auxiliary storage.

**mark.** A symbol or symbols that indicate the beginning or the end of a field, of a word, of an item of data, or of a set of data such as a file, a record, or a block.

**mask.** (1) A pattern of characters that is used to control the retention or elimination of portions of another pattern of characters. (2) To use a pattern of characters to control the retention or elimination of portions of another pattern of characters.

**masked.** Synonym for disabled.

**matrix.** (1) A rectangular array of elements, arranged in rows and columns, that may be manipulated according to the rules of

matrix algebra. (2) In computers, a logic network in the form of an array of input leads and output leads with logic elements connected at some of their intersections.

**matrix printer.** A printer in which each character is represented by a pattern of dots; for example, a stylus printer, a wire printer. Synonymous with dot printer.

**Mb.** 1,048,576 bytes.

**mega (M).** Prefix 1,000,000.

**megahertz (MHz).** 1,000,000 hertz.

**memory.** Term for main storage.

**meter (m).** A unit of length (equivalent to 39.37 inches).

**MFM.** Modified frequency modulation.

**MHz.** Megahertz; 1,000,000 hertz.

**micro ( $\mu$ ).** 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 (RGBI).** The description of a direct-drive color monitor that accepts input signals of red, green, blue, and intensity.

**redundancy check.** A check that depends on extra characters attached to data for the detection of errors. See cyclic redundancy check.

**register.** (1) A storage device, having a specified storage capacity such as a bit, a byte, or a computer word, and usually intended for a special purpose. (2) A storage device in which specific data is stored.

**retry.** To resend the current block of data (from the last EOB or ETB) a prescribed number of times, or until it is entered correctly or accepted.

**reverse video.** A form of highlighting a character, field, or cursor by reversing the color of the character, field, or cursor with its background; for example, changing a red character on a black background to a black character on a red background.

**RF.** Radio frequency.

**RF modulator.** The device used to convert the composite video signal to the antenna level input of a home TV.

**RGBI.** Red-green-blue-intensity.

**rising edge.** Synonym for positive-going edge.

**ROM.** Read-only memory.

**ROM/BIOS.** The ROM resident basic input/output system, which provides the level control of the major I/O devices in the computer system.

**row address strobe (RAS).** A signal that latches the row address in a memory chip.

**RS-232C.** A standard by the EIA for communication between computers and external equipment.

**RTS.** Request to send. Associated with modem control.

**run.** A single continuous performance of a computer program or routine.

**saturation.** In computer graphics, the purity of a particular hue. A color is said to be saturated when at least one primary color (red, blue, or green) is completely absent.

**scaling.** In computer graphics, enlarging or reducing all or part of a display image by multiplying the coordinates of the image by a constant value.

**schematic.** The representation, usually in a drawing or diagram form, of a logical or physical structure.

**Schottky TTL.** A version (S series) of TTL with faster switching speed, but requiring more power. See also transistor-transistor logic and low power Schottky TTL.

**SDLC.** Synchronous Data Link Control.

**sector.** That part of a track or band on a magnetic drum, a magnetic disk, or a disk pack that can be accessed by the magnetic heads in the course of a predetermined rotational displacement of the particular device.

**SERDES.** Serializer/deserializer.

**serial.** (1) Pertaining to the sequential performance of two or more activities in a single device. In English, the modifiers serial and parallel usually refer to devices, as opposed to sequential and consecutive, which refer to processes. (2) Pertaining to the sequential or consecutive occurrence of two or more related activities in a single device or channel. (3) Pertaining to the sequential processing of the individual parts of a whole, such as the bits of a character or the characters of a word, using the same facilities for successive parts. (4) Contrast with parallel.

**serializer/deserializer (SERDES).** A device that serializes output from, and deserializes input to, a business machine.

**setup.** (1) In a computer that consists of an assembly of individual computing units, the arrangement of interconnections between the units, and the adjustments needed for the computer to operate. (2) The preparation of a computing system to perform a job or job step. Setup is usually performed by an operator and often involves performing routine functions, such as mounting tape reels. (3) The preparation of the system for normal operation.

**short circuit.** A low-resistance path through which current flows, rather than through a component or circuit.

**signal.** A variation of a physical quantity, used to convey data.

**sink.** A device or circuit into which current drains.

**software.** (1) Computer programs, procedures, and rules concerned with the operation of a data processing system. (2) Contrast with hardware.

**source.** The origin of a signal or electrical energy.

**square wave.** An alternating or pulsating current or voltage whose waveshape is square.

**square wave generator.** A signal generator delivering an output signal having a square waveform.

**SS.** Start-stop.

**start bit.** (1) A signal to a receiving mechanism to get ready to receive data or perform a function. (2) In a start-stop system, a signal preceding a character or block that prepares the receiving device for the reception of the code elements.

**start-of-text (STX).** A transmission control character that precedes a text and may be used to terminate the message heading.

**start-stop system.** A data transmission system in which each character is preceded by a start bit and is followed by a stop bit.

**start-stop (SS) transmission.** (1) Asynchronous transmission such that a group of signals representing a character is preceded by a start bit and followed by a stop bit. (2) Asynchronous transmission in which a group of bits is preceded by a start bit that prepares the receiving mechanism for the reception and registration of a character and is followed by at least one stop bit that enables the receiving mechanism to come to an idle condition pending the reception of the next character.

**static memory.** RAM using flip-flops as the memory elements. Data is retained as long as power is applied to the flip-flops. Contrast with dynamic memory.

**stop bit.** (1) A signal to a receiving mechanism to wait for the next signal. (2) In a start-stop system, a signal following a character or block that prepares the receiving device for the reception of a subsequent character or block.

**storage.** (1) A storage device. (2) A device, or part of a device, that can retain data. (3) The retention of data in a storage device. (4) The placement of data into a storage device.

**strobe.** An instrument that emits adjustable-rate flashes of light. Used to measure the speed of rotating or vibrating objects.

**STX.** Start-of-text.

**symbol.** (1) A conventional representation of a concept. (2) A representation of something by reason of relationship, association, or convention.

**synchronization.** The process of adjusting the corresponding significant instants of two signals to obtain the desired phase relationship between these instants.

**Synchronous Data Link Control (SDLC).** A protocol for management of data transfer over a data link.

**synchronous transmission.** (1) Data transmission in which the time of occurrence of each signal representing a bit is related to a fixed time frame. (2) Data transmission in which the sending and receiving devices are operating continuously at substantially the same frequency and are maintained, by means of correction, in a desired phase relationship.

**syntax.** (1) The relationship among characters or groups of characters, independent of their meanings or the manner of their interpretation and use. (2) The structure of expressions in a language. (3) The rules governing the structure of a language. (4) The relationships among symbols.

**text.** In ASCII and data communication, a sequence of characters treated as an entity if preceded and terminated by one STX and one ETX transmission control character, respectively.

**time-out.** (1) A parameter related to an enforced event designed to occur at the conclusion of a predetermined elapsed time. A time-out condition can be cancelled by the receipt of an appropriate time-out cancellation signal. (2) A time interval allotted for certain operations to occur; for example, response to polling or addressing before system operation is interrupted and must be restarted.

**track.** (1) The path or one of the set of paths, parallel to the reference edge on a data medium, associated with a single reading or writing component as the data medium moves past the

**component.** (2) The portion of a moving data medium such as a drum, or disk, that is accessible to a given reading head position.

**transistor-transistor logic (TTL).** A popular logic circuit family that uses multiple-emitter transistors.

**translate.** To transform data from one language to another.

**transmission.** (1) The sending of data from one place for reception elsewhere. (2) In ASCII and data communication, a series of characters including headings and text. (3) The dispatching of a signal, message, or other form of intelligence by wire, radio, telephone, or other means. (4) One or more blocks or messages. For BSC and start-stop devices, a transmission is terminated by an EOT character. (5) Synonymous with data transmission.

**TTL.** Transistor-transistor logic.

**typematic key.** A keyboard key that repeats its function when held pressed.

**V.** Volt.

**vector.** In computer graphics, a directed line segment.

**video.** Computer data or graphics displayed on a cathode ray tube, monitor, or display.

**view point.** In computer graphics, the origin from which angles and scales are used to map virtual space into display space.

**viewing reference point.** In computer graphics, a point in the modeling coordinate space that is a defined distance from the view point.

**viewing transformation.** Operations on the coordinates of an object (usually matrix multiplications) that cause the view of the object to be rotated about any axis, translated (moved without

rotating), and/or scaled (changed in size along any or all dimensions). Viewing transformation differs from modeling transformation in that perspective is considered. See also modeling transformation.

**viewplane.** The visible plane of a CRT display screen that completely contains a defined window.

**viewport.** In computer graphics, a predefined part of the CRT display space.

**volt.** The basic practical unit of electric pressure. The potential that causes electrons to flow through a circuit.

**W.** Watt.

**watt.** The practical unit of electric power.

**window.** (1) A predefined part of the virtual space. (2) The visible area of a viewplane.

**word.** (1) A character string or a bit string considered as an entity. (2) See computer word.

**write.** To make a permanent or transient recording of data in a storage device or on a data medium.

**write precompensation.** The varying of the timing of the head current from the outer tracks to the inner tracks of the diskette to keep a constant 'write' signal.

**yon plane.** In computer graphics, a plane that is perpendicular to the line joining the viewing reference point and the view point, and that lies beyond the viewing reference point. Any part of an object beyond the yon plane is not seen. See also hither plane.

# Bibliography

- Microprocessor and Peripheral Handbook
  - INTEL Corporation. *210844.001*
- Introduction to the iAPX 286
  - INTEL Corporation. *210308.001*
- iAPX 286 Operating Systems Writer's Guide
  - INTEL Corporation. *121960.001*
- iAPX 286 Programmer's Reference Manual
  - INTEL Corporation. *210498.001*
- iAPX 286 Hardware Reference Manual
  - INTEL Corporation. *210760.001*
- Numeric Processor Extension Data Sheet
  - INTEL Corporation. *210920*
- 80287 Support Library Reference Manual
  - INTEL Corporation. *122129*
- National Semiconductor Corporation. *NS16450*
- Motorola Microprocessor's Data Manual
  - Motorola Inc. *Series B*

## **Notes:**

# Index

## A

AAA 6-8  
AAD 6-9  
AAM 6-9  
AAS 6-8  
access time,  
track-to-track 9-6  
ADC 6-6  
ADD 6-6  
additional ROM  
modules 5-13  
address generation, DMA 1-9  
address latch enable 1-35  
address latch enable,  
buffered 1-32  
address mode  
real 1-4  
address space, I/O 1-24  
address, segment 1-4  
addresses, CMOS RAM 1-56  
addresses, page register 1-10  
AEN 1-35  
ALE 9-4  
alternate key 5-20  
AND 6-10  
APL 9-7  
application guidelines 9-7  
arithmetic instructions 6-6,  
6-25  
ARPL 6-19  
ASCII characters 7-3  
ASCII, extended 5-14

## B

BALE 1-32  
bandwidth 1-7  
BASIC 9-7  
basic assurance test 4-5  
BASIC interrupts 5-6  
BAT 4-5  
battery connector 1-72  
BHE 1-9  
BIOS  
quick reference 5-24  
BIOS fixed disk  
parameters 1-63  
BIOS memory map 5-10  
BIOS programming  
hints 5-10  
block diagram  
keyboard interface 1-49  
system xiv  
system board 1-6  
system timer 1-22  
board, system 1-3  
BOUND 6-16  
break code 4-4, 4-11  
break key 5-21  
buffer, keyboard 4-3  
buffered address latch  
enable 1-32  
buffers, video display 9-14  
bus controller 1-32  
bus cycle 1-7  
busy loop 9-17  
bypassing BIOS 9-6  
byte high enable 1-9

# C

CALL 6-13  
capacitor, variable 1-41  
caps lock key 5-20  
CBW 6-9  
channel, I/O 1-24  
    connectors 1-25  
    pin assignments 1-28  
    signals 1-31  
channels, DMA 1-7, 1-9  
character codes 5-14  
characters 7-3  
classes, wait loop 9-17  
CLC 6-17  
CLD 6-17  
CLEX 6-27  
CLI 6-17  
CLK 1-31  
clock  
    real-time 1-56, 1-57  
clock and data signals  
clock cycle 1-7  
clock line, keyboard 1-54, 4-5, 4-12, 4-13  
clock, system 1-7  
CMC 6-17  
CMOS RAM 1-56  
CMOS RAM addresses 1-56  
CMOS RAM  
    configuration 1-59  
CMOS RAM I/O  
    operations 1-68  
CMP 6-7  
CMPS 6-11  
COBOL 9-7  
code  
    device driver 9-16  
    machine  
        identification 9-19  
    machine-sensitive 9-19

codes  
    character 5-14  
    extended 5-18  
    multitasking  
        function 9-18  
color burst signal 1-41  
command codes, DMA  
    controller 1-11  
commands  
    I/O 9-11  
    keyboard 4-9  
    keyboard controller 1-51  
    keyboard system 4-5  
commands from the system  
commands to the system  
comparison instructions 6-23  
compatibility, hardware 9-3  
condition, wait 9-17  
configuration record 1-56  
configuration, CMOS  
    RAM 1-59  
connectors  
    battery 1-72  
    I/O channel 1-25  
    J-1 through J-16 1-26  
    keyboard 1-73, 4-3  
    power LED and key  
        lock 1-72  
    power supply 1-71  
    power supply output 3-7  
    speaker 1-72  
    system board 1-71  
constants instructions 6-24  
control  
    game 9-14  
    sound 9-13  
control key 5-20  
control transfer  
    instructions 6-13  
controller, keyboard 1-42  
controllers  
    bus 1-32  
    DMA 1-7, 1-9, 1-10

interrupt 1-12  
refresh 1-7  
coprocessor controls 1-39  
coprocessor programming 2-3  
coprocessor, math 2-3  
copy protection 9-5, 9-14  
Ctrl state 5-18  
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 9-14  
data communication  
    equipment 8-3  
data input, keyboard 4-13  
data line, keyboard 1-54, 4-5,  
    4-12, 4-13  
data output, keyboard 4-13  
data stream 4-12  
data terminal equipment 8-3  
data transfer instructions 6-3,  
    6-22  
data transfer rate,  
    diskette 9-6  
DEC 6-7, 6-8  
decodes, memory 1-11, 1-31  
DECSTP 6-28  
default segment  
    workspace 5-9  
description  
descriptors 1-5

device driver code 9-16  
diagnostic checkpoint  
    port 1-39  
direct memory access 1-9  
disk pointer 9-12  
disk\_base 9-6, 9-12  
diskette change signal 9-6  
diskette data transfer rate 9-6  
diskette rotational speed 9-6  
diskette track density 9-6  
diskette write current 9-7  
DIV 6-9  
divide error exception 9-9  
DMA address generation 1-9  
DMA channels 1-7, 1-9  
DMA controller 1-7  
DMA controller command  
    codes 1-11  
DMA controller 1 1-9  
DMA controller 2 1-10  
DMA controllers 1-9  
DOS 9-7  
DOS function calls 9-10  
DOS interrupts 5-6  
DRQ0-DRQ3 1-34  
DRQ5-DRQ7 1-34

## E

EIA/CCITT 8-3  
enable NMI 1-38  
encoding, keyboard 5-13  
ENTER 6-16  
ESC 6-18  
exception, divide error 9-9  
extended ASCII 5-14  
extended codes 5-18

# F

FABS 6-26  
FADD 6-25  
FCHS 6-26  
FCOM 6-23  
FCOMP 6-23  
FCOMPP 6-24  
FDIV 6-25  
FIFO 4-3  
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 9-7  
FPREM 6-26  
FREE 6-28  
French keyboard 4-16  
FRNDINT 6-26  
FSCALE 6-26  
FSQRT 6-25  
FST 6-22  
FSTP 6-22  
FSUB 6-25  
FTST 6-24  
function calls, DOS 9-10  
function codes,  
multitasking 9-18  
FXAM 6-24  
FXCH 6-23  
FXTRACT 6-26

# G

game control 9-14  
gap length parameter 9-12  
generator, refresh  
request 1-22  
German keyboard 4-17  
graphics modes 5-8  
guidelines, application 9-7

# H

hard code 5-10  
hardware compatibility 9-3  
hardware interrupts 5-6  
HLT 6-17  
hooks 9-16

# I

I/O address map 1-37  
I/O address space 1-24  
I/O CH CK 1-32, 1-40  
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 9-11  
I/O CS16 1-36  
I/O ports, keyboard  
controller 1-54

I/O read 1-33  
I/O write 1-33  
IDIV 6-9  
IIMUL 6-9  
IMR 9-14  
IMUL 6-8  
IN 6-5  
INC 6-6  
INCSTP 6-28  
inhibit keyboard 1-48  
input buffer, keyboard  
    controller 1-51  
input port, keyboard  
    controller 1-54  
input requirements 3-3  
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  
interface, keyboard 4-3  
interfaces, multitasking 9-16  
interrupt controller 1-12  
interrupt mask register 9-14  
interrupt service routine 1-33  
interrupt sharing 1-14  
interrupt vectors 9-14  
interrupt, single step 9-8  
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 9-11  
IRQ 9 9-4, 9-11  
IRQ3-IRQ15 1-33  
Italian keyboard 4-18

## 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 1-40

# K

key lock 4-3  
key scan codes 4-11  
keyboard  
    buffer 4-3  
    clock line 1-54, 4-5, 4-12, 4-13  
    commands 4-9  
    connector 1-73, 4-3  
    controller 1-42  
    controller commands 1-51  
    controller I/O ports 1-54  
    controller input  
        buffer 1-51  
    controller input port 1-54  
    controller output  
        buffer 1-51  
    controller output  
        port 1-54  
    controller status  
        register 1-49  
    controller test inputs 1-54  
    data input 4-13  
    data line 1-54, 4-5, 4-12, 4-13  
    data output 4-13  
    encoding 5-13  
    inhibit switch 1-48  
    interface 4-3  
    interface block  
        diagram 1-49  
    layout 1-44, 5-15  
    outputs 4-11  
    routine 5-23  
    specifications 4-22  
    system commands 4-5  
keyboard layouts  
keyboard scan-code outputs  
keyboard, French 4-16  
keyboard, German 4-17

keyboard, Italian 4-18  
keyboard, Spanish 4-19  
keyboard, U.K. English 4-20  
keyboard, U.S. English 4-21  
keys 4-4  
    alternate 5-20  
    break 5-21  
    caps lock 5-20  
    combinations 5-21  
    control 5-20  
    number lock 5-21  
    pause 5-22  
    print screen 5-22  
    scroll lock 5-20  
    shift 5-19  
    system request 5-6, 5-22  
keys, typematic 4-4

# L

LAHF 6-5  
LAR 6-19  
layout system board 1-74  
layout, keyboard 1-44, 5-15  
LA17-LA23 1-31  
LDCW 6-27  
LDENV 6-27  
LDS 6-5  
LEA 6-5  
LEAVE 6-16  
LED 4-5  
LES 6-5  
LGDT 6-18  
LIDT 6-18  
light emitting diodes 4-5  
line contention 4-13  
line, multipoint 8-5  
line, point-to-point 8-5  
LLDT 6-18  
LMSW 6-19

load current 3-4  
LOCK 6-17  
LODS 6-11  
logic diagrams  
logic instructions 6-9  
LOOP 6-15  
loop, busy 9-17  
LOOPNZ/LOOPNE 6-16  
loops, program 9-14  
LOOPZ/LOOPE 6-15  
LSL 6-19  
LTR 6-18

## M

machine identification  
code 9-19  
machine-sensitive code 9-19  
make code 4-4, 4-11  
mask on and off 1-39  
master 1-35  
math coprocessor 2-3, 9-11  
math coprocessor  
controls 1-39  
MEM chip select 1-36  
MEM CS16 1-36  
memory 1-4  
memory decodes 1-11, 1-31  
memory locations,  
reserved 5-9  
memory map, BIOS 5-10  
MEMR 1-34  
MEMW 1-34  
microprocessor 1-4, 1-7  
microprocessor cycle 1-7  
modes, graphic 5-8  
modules, RAM 1-24  
modules,  
ROM/EPROM 1-23  
MOV 6-3

MOVS 6-11  
MUL 6-8  
multi-tasking  
function codes 9-18  
interfaces 9-16  
provisions 9-16  
serialization 9-16  
startup 9-16  
multipoint line 8-5

## N

NEG 6-8  
network, nonswitched 8-5  
network, switched 8-5  
NMI 1-12, 1-38  
no load protection 3-5  
non-maskable interrupt 1-38  
nonswitched network 8-5  
NOP 6-26, 6-28  
NOT 6-11  
Num Lock state 5-18  
number lock key 5-21

## O

operations, CMOS RAM  
I/O 1-68  
OR 6-10  
OSC 1-36, 1-41  
oscillator 1-36  
OUT 6-5  
output buffer, keyboard  
controller 1-51  
output port, keyboard  
controller 1-54  
output protection 3-4

output voltage sense  
levels 3-6  
output voltage  
sequencing 3-4  
outputs, keyboard 4-11  
outputs, power supply 3-4  
OUTS 6-12

## P

page register addresses 1-10  
parameter  
gap length 9-12  
passing 5-4  
tables 9-12  
parameters, BIOS fixed  
disk 1-63  
Pascal 9-7  
PATAN 6-26  
pause key 5-22  
performance, system 1-7  
point-to-point line 8-5  
POP 6-4  
POPA 6-4  
POPF 6-6, 9-8  
POR 4-4  
port, diagnostic  
checkpoint 1-39  
post 9-17  
power good signal 3-5  
power LED and key lock  
connector 1-72  
power on reset 4-4  
power supply  
connectors 1-71  
inputs 3-3  
output connectors 3-7  
outputs 3-4  
power-on routine  
print screen key 5-22

priorities, shift key 5-21  
processor control  
instructions 6-17  
program interrupts 5-3  
program loops 9-14  
programming hints,  
BIOS 5-10  
programming,  
coprocessor 2-3  
protected mode 1-5, 5-6  
protection control  
instructions 6-18  
protection, no load 3-5  
provisions, multitasking 9-16  
PTAN 6-26  
PUSH 6-3  
PUSH SP 9-8  
PUSHA 6-4  
PUSHF 6-6

## Q

quick reference charts 7-14

## R

RAM jumper 1-40  
RAM modules 1-24  
RAM subsystem 1-24  
RAM, CMOS 1-56  
rate, typematic 4-4, 4-7  
real address mode 1-4, 2-5  
real mode 5-3  
real-time clock 1-56, 1-57  
record, configuration 1-56  
refid=admod.virtual 1-4  
REFRESH 1-35  
refresh controller 1-7

refresh request  
generator 1-22  
regulation tolerance 3-4  
REP/REPNE,  
  REPZ/REPNZ 6-12  
requirements, input 3-3  
reserved memory  
  locations 5-9  
reserved scan codes 1-47  
RESET DRV 1-32  
reset, system 5-21  
RET 6-13  
ROM BIOS 9-10  
ROM BIOS data area 9-14  
ROM modules,  
  additional 5-13  
ROM scan codes 5-13  
ROM subsystem 1-23  
ROM/EPROM  
  modules 1-23  
rotate instructions 6-9  
rotational, speed 9-6  
routine, interrupt  
  service 1-33  
routine, keyboard 5-23  
RS-232 8-3  
RSTOR 6-28

## S

SAHF 6-5  
SAVE 6-28  
SA0-SA19 1-31  
SBB 6-7  
SBHE 1-35  
scan code translation 1-43  
scan codes 4-11  
scan codes, key 4-11  
scan codes, ROM 5-13  
SCAS 6-11

scroll lock key 5-20  
SD0-SD15 1-32  
segment address 1-4  
segments 1-5  
sense levels, output  
  voltage 3-6  
sequencing, output  
  voltage 3-4  
serialization,  
  multitasking 9-16  
SETPM 6-27  
SGDT 6-18  
shift counts 9-9  
shift instructions 6-9  
shift key 5-19  
shift key priorities 5-21  
Shift state 5-18  
shift states 5-19  
SIDT 6-18  
signals  
  diskette change 9-6  
  I/O channels 1-31  
  power good 3-5  
  system clock 9-4  
single step interrupt 9-8  
SLDT 6-18  
SMEMR 1-34  
SMEMW 1-34  
SMSW 6-19  
sound control 9-13  
Spanish keyboard 4-19  
speaker 1-40  
speaker connector 1-72  
speaker tone generation 1-22  
special vectors 5-6  
specifications  
  specifications, keyboard 4-22  
startup, multitasking 9-16  
states  
  Ctrl 5-18  
  Num Lock 5-18  
  Shift 5-18, 5-19

status register, keyboard controller 1-49  
STC 6-17  
STCW 6-27  
STD 6-17  
STENV 6-27  
STI 6-17  
STOS 6-12  
STR 6-19  
string manipulation instructions 6-11  
STS W 6-27  
STS WAX 6-27  
SUB 6-7  
subsystem, RAM 1-24  
subsystem, ROM 1-23  
support joystick 5-6  
switched network 8-5  
switches  
  keyboard inhibit 1-48  
  type of display 1-41  
system BIOS usage 5-3  
system block diagram xiv  
system board 1-3  
system board block diagram -  
  type 1 1-6  
system board block diagram -  
  type 2 1-6  
system board  
  connectors 1-71  
system board layout 1-74  
system bus high enable 1-35  
system clock 1-7  
system clock signal 9-4  
system interrupts 1-12  
system performance 1-7  
system request key 5-6, 5-22  
system reset 5-21  
system timer block  
  diagram 1-22  
system timers 1-22

## T

T/C 1-35  
table, translation 1-45  
tables, parameter 9-12  
terminal count 1-35  
TEST 6-10  
test inputs, keyboard controller 1-54  
time-outs 9-19  
timer/counter 1-22  
timer/counters 1-22  
timers, system 1-22  
tone generation, speaker 1-22  
track density, diskette 9-6  
track-to-track access  
  time 9-6  
translation table 1-45  
translation, scan code 1-43  
tri-state 1-36  
type of display adapter  
  switch 1-41  
typematic keys 4-4  
typematic rate 4-4, 4-7

## U

U.K. English keyboard 4-20  
U.S. English keyboard 4-21

## V

variable capacitor 1-41  
vectors, special 5-6  
VERR 6-19  
video display buffers 9-14

virtual address mode 1-4, 2-5

**Y**

**W**

YL2XP1 6-27

WAIT 6-17  
wait condition 9-17  
wait loop classes 9-17  
workspace, default  
    segment 5-9  
write current, diskette 9-7

**Z**

zero wait state 1-36

**X**

XCHG 6-4  
XLAT 6-5  
XOR 6-11

0WS 1-36  
2XM1 6-26  
80286 1-4  
8042 1-42  
82288 1-32  
8237A-5 1-9  
8254-2 1-22  
8259A Interrupt 1-12

## **Notes:**



The Personal Computer  
Hardware Reference  
Library

**Reader's Comment Form**

**Technical Reference**

**6280070**

Your comments assist us in improving the usefulness of our publication; they are an important part of the input used for revisions.

IBM may use and distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Please do not use this form for technical questions regarding the IBM Personal Computer or programs for the IBM Personal Computer, or for requests for additional publications; this only delays the response. Instead, direct your inquiries or request to your authorized IBM Personal Computer dealer.

**Comments:**



NO POSTAGE  
NECESSARY  
IF MAILED  
IN THE  
UNITED STATES

# BUSINESS REPLY MAIL

FIRST CLASS

PERMIT NO. 40

ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM PERSONAL COMPUTER  
READER COMMENT DEPARTMENT  
P.O. BOX 1328-C  
BOCA RATON, FLORIDA 33429-9960



.....  
Fold here

Tape

Please do not staple

Tape