

IBM

*Personal Computer
Hardware Reference
Library*

Technical Reference

6139362





*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

SECTION 1. SYSTEM BOARD	1-1
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
SECTION 2. COPROCESSOR	2-1
Description	2-3
Programming Interface	2-3
Hardware Interface	2-4
SECTION 3. POWER SUPPLY	3-1
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
SECTION 4. KEYBOARD	4-1
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
SECTION 5. SYSTEM BIOS	5-1
System BIOS Usage	5-3
Keyboard Encoding and Usage	5-13
Quick Reference	5-24
SECTION 6. INSTRUCTION SET	6-1
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
SECTION 7. CHARACTERS, KEYSTROKES, AND	
COLORS	7-1
Character Codes	7-3
Quick Reference	7-14

SECTION 8. COMMUNICATIONS	8-1
Hardware	8-3
Establishing a Communications Link	8-5
SECTION 9. IBM PERSONAL COMPUTER	
COMPATIBILITY	9-1
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
Glossary	Glossary-1
Bibliography	Bibliography-1
Index	Index-1

Notes:

(

(

(

INDEX TAB LISTING

Section 1: System Board

SECTION 1

Section 2: Coprocessor

SECTION 2

Section 3: Power Supply

SECTION 3

Section 4: Keyboard

SECTION 4

Section 5: System BIOS

SECTION 5

Section 6: Instruction Set

SECTION 6

Notes:



Section 7: Characters, Keystrokes, and Colors

SECTION 7

Section 8: Communications

SECTION 8

Section 9: Compatibility

SECTION 9

Glossary

GLOSSARY

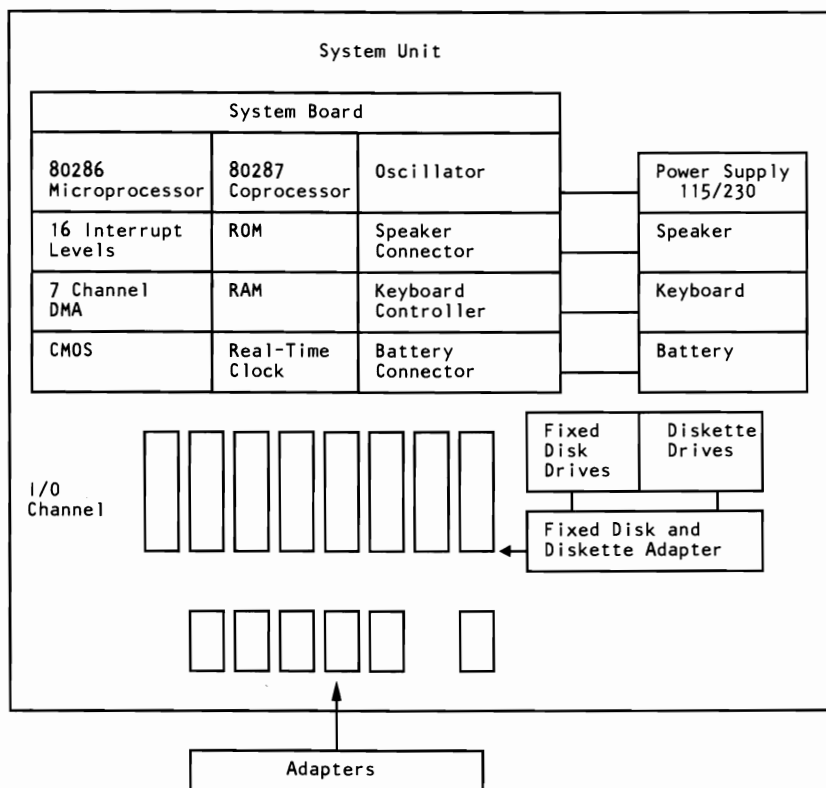
Bibliography

BIBLIOGRAPHY

Index

INDEX

System Block Diagram



SECTION 1. SYSTEM BOARD

Contents

- Memory** 1-4
- Microprocessor** 1-4
 - Real Address Mode 1-4
 - Protected (Virtual Address) Mode 1-5
- System Performance** 1-7
- Direct Memory Access** 1-9
- System Interrupts** 1-12
 - Hardware Interrupt Listing 1-13
 - Interrupt Sharing 1-14
 - Design Overview 1-14
 - Program Support 1-15
 - Precautions 1-17
 - Examples 1-18
- System Timers** 1-22
- System Clock** 1-23
- ROM Subsystem** 1-23
- RAM Subsystem** 1-24
- I/O Channel** 1-24
 - Connectors 1-25
 - I/O Channel Signal Description 1-31
 - 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
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
Specifications	1-69
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 interface¹, an extensive instruction set, DMA and interrupt support capabilities, a hardware fixed-point multiply and divide, integrated memory management, four-level memory protection, 1G (1,073,741,824 bytes) of virtual address space for each task, and two operating modes: the 8086-compatible real address mode and the protected or virtual address mode. More detailed descriptions of the microprocessor may be found in the publications listed in the Bibliography of this manual.

Real Address Mode

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

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

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

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

Protected (Virtual Address) Mode

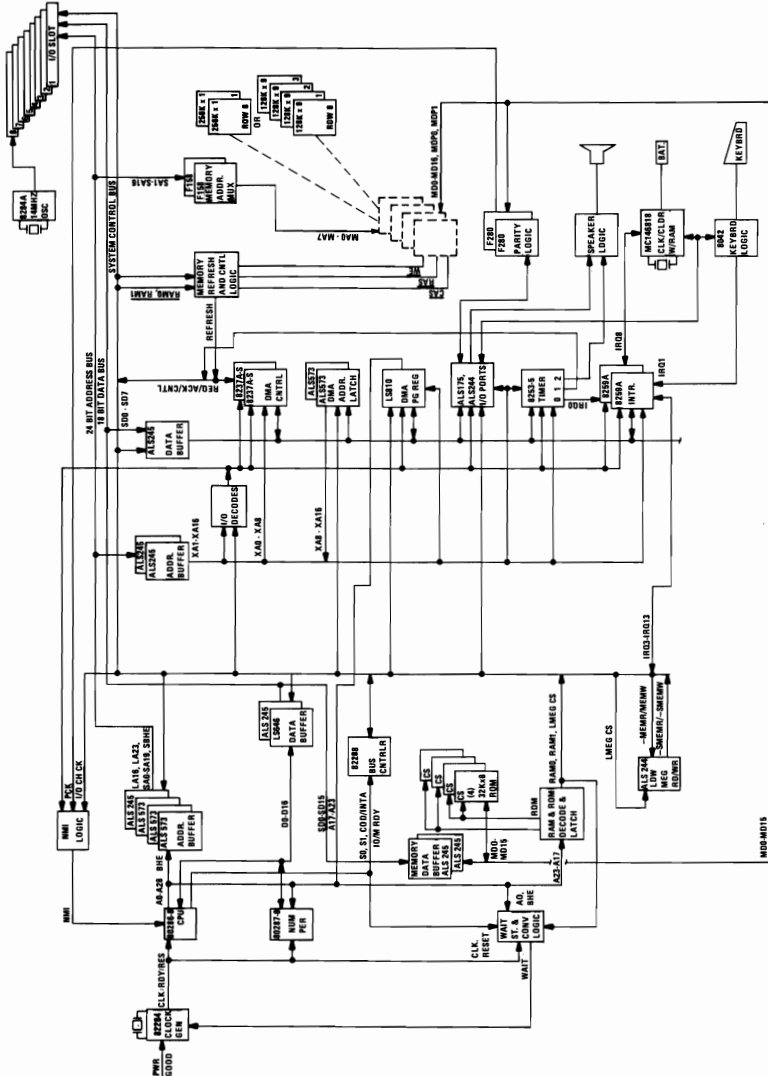
The protected mode offers extended physical and virtual memory address space, memory protection mechanisms, and new operations to support operating systems and virtual memory.

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

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

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

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.

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

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 0E0000
FF0000 to FFFFFFFF	64K ROM on the system board	Duplicated code assignment at address 0F0000

System Memory Map

Direct Memory Access

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

Controller 1	Controller 2
Ch 0 - Reserved	Ch 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	CH0 base and current address
0C2	CH0 base and current word count
0C4	CH1 base and current address
0C6	CH1 base and current word count
0C8	CH2 base and current address
0CA	CH2 base and current word count
0CC	CH3 base and current address
0CE	CH3 base and current word count
0D0	Read Status Register/Write Command Register
0D2	Write Request Register
0D4	Write Single Mask Register Bit
0D6	Write Mode Register
0D8	Clear Byte Pointer Flip-Flop
0DA	Read Temporary Register/Write Master Clear
0DC	Clear Mask Register
0DE	Write All Mask Register Bits

DMA Controller

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

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

System Interrupts

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

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

Hardware Interrupt Listing

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

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

Hardware Interrupt Listing

Interrupt Sharing

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

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

Design Overview

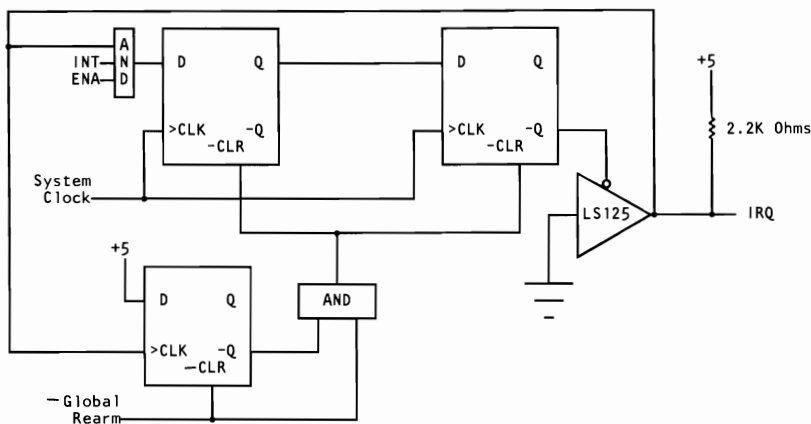
Most interrupt-supporting adapters hold the 'interrupt request' line (IRQ) at a low level and then drive the line high to cause an interrupt. In contrast, the shared-interrupt hardware design allows IRQ to float high through pull-up resistors on each adapter. Each adapter on the line may cause an interrupt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

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

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

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

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



Shared Interrupt Logic Diagram

Program Support

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

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

Linking to a Chain

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

Sharing the Interrupt Level

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

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

Unlinking from the Chain

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

Error Recovery

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

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

ROS Considerations

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

Precautions

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

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

- Treat words as words, not bytes. Remember that data is stored in memory using the Intel format (word 424B is stored as 4B42).

Interrupt Chaining Structure

```

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

```

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

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

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

Examples

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

reenabled) before another interrupt is serviced, protecting the stack from excessive build up.

Example of an Interrupt Handler

```

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

MYCSEG     SEGMENT PARA
           ASSUME  CS:MYCSEG,DS:DSEG
ENTRY      PROC    FAR
           JMP     SHORT PAST      ; Entry point of handler
FPTR       DD      0              ; Forward Pointer
SIGNATURE  DW      424BH          ; Used when unlinking to identify
                                           ; compatible interrupt handlers

FLAGS      DB      0             ; Flags
FIRST      EQU      80H
JMP        SHORT  RESET
RES_BYTES  DB      DUP 7 (0)      ; Future expansion
PAST:      STI                ; Actual start of handler code
           PUSH   ...           ; Save needed registers
           MOV    DX, YOUR_CARD  ; Select your status register
           IN     AL, DX         ; Read the status register
           TEST   AL, ISB        ; Your card caused the interrupt?
           JNZ   SERVICE        ; Yes, branch to service logic
           TEST   CS:FLAGS, FIRST ; Are we the first ones in?
           JNZ   EXIT           ; If yes, branch for EOI and Rearm
           POP    ...           ; Restore registers
           CLI    ...           ; Disable interrupts
           JMP    DWORD PTR CS:FPTR ; Pass control to next guy on chain

SERVICE:  ...                 ; Service the interrupt
EXIT:     CLI                ; Disable the interrupts
           MOV    AL,EOI        ;
           OUT   OCR,AL         ; Issue non-specific EOI to 8259
           MOV   DX,REARM       ; Rearm the cards
           OUT   DX,AL         ;
           POP    ...           ; Restore registers
           IRET

RESET:    ...                 ; Disable your card
           RET                  ; Return FAR to operating system

ENTRY    ENDP
MYCSEG   ENDS
END      ENTRY

```

Linking Code Example

```

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

Unlinking Code Example

```

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

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

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

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

UNCHAIN_X: STI                          ; Enable interrupts
POP     ES
POP     DS

```

System Timers

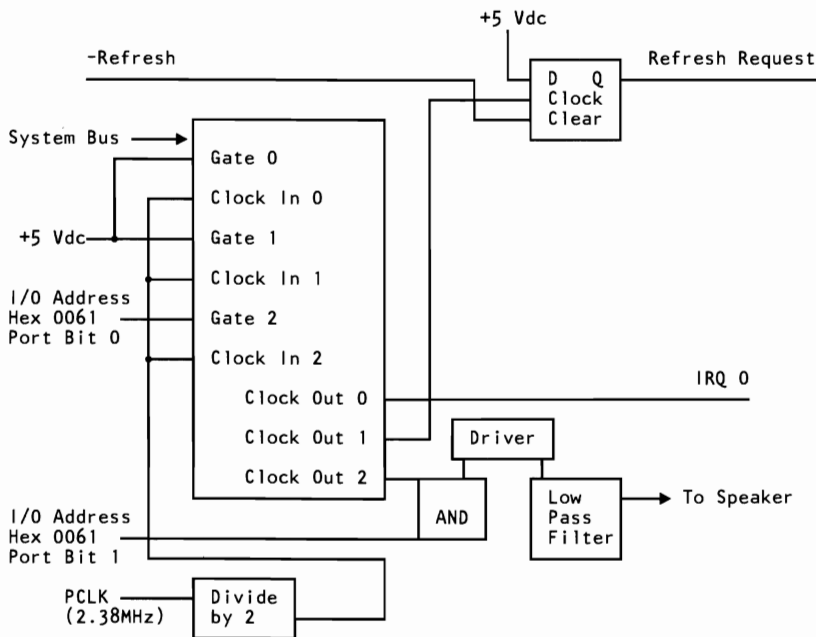
The system has three programmable timer/counters, Channels 0 through 2. They are controlled by an Intel 8254-2 Timer/Counter chip, and are defined as follows:

Channel 0	System Timer
GATE 0	Tied on
CLK IN 0	1.190 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:

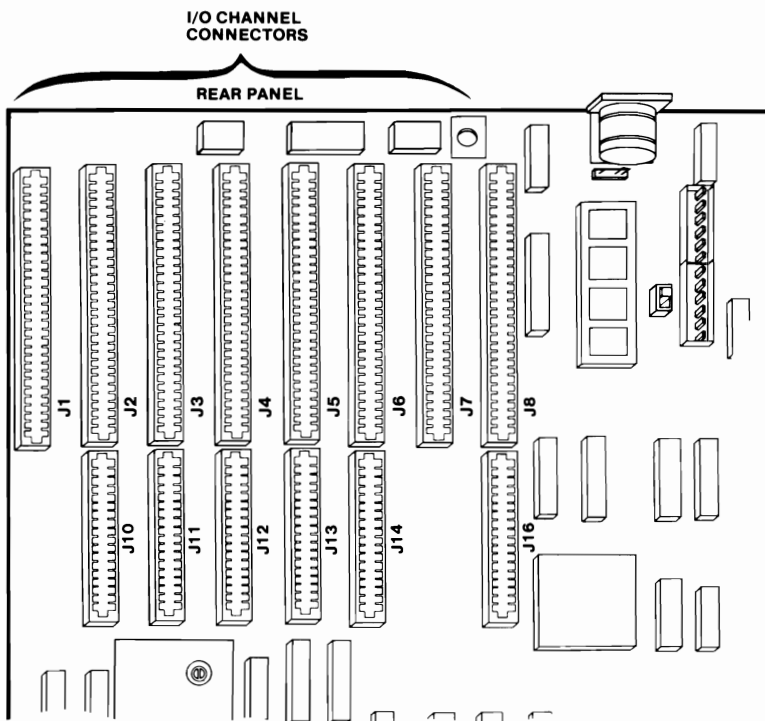
- I/O address space hex 100 to hex 3FF
- 24-bit memory addresses (16M)
- Selection of data accesses (either 8- or 16-bit)
- Interrupts
- DMA channels
- I/O wait-state generation

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

Connectors

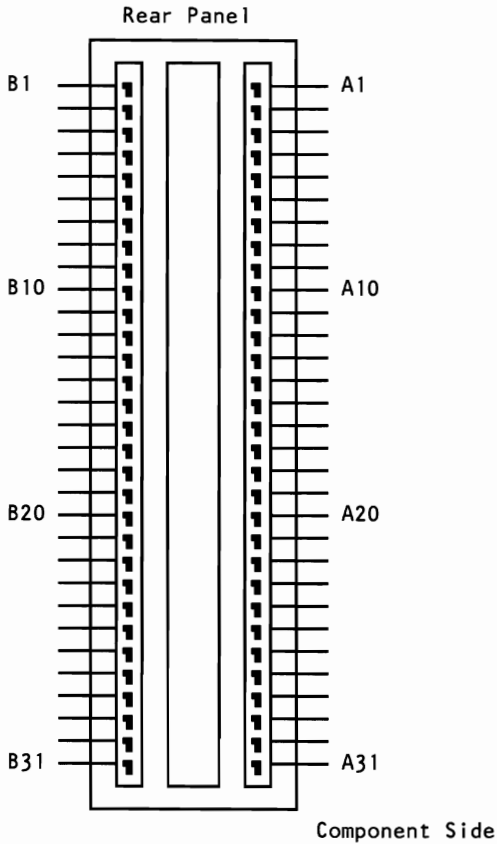
The following figure shows the location and the numbering of the I/O channel connectors. These connectors consist of six 36-pin and eight 62-pin edge connector sockets.

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.



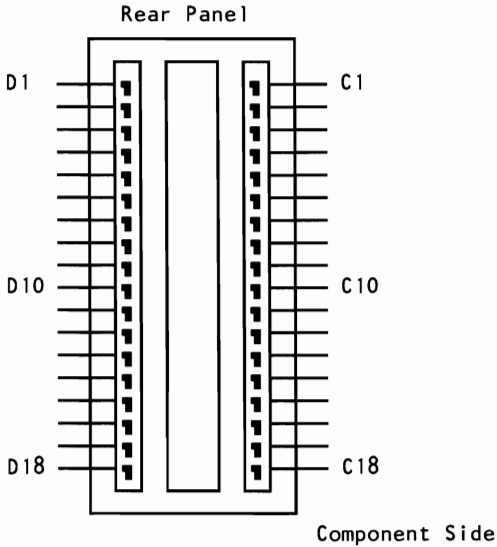
I/O Channel Connector Location

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	I
B5	-5 Vdc	Power
B6	DRQ2	I
B7	-12 Vdc	Power
B8	OWS	I
B9	+12 Vdc	Power
B10	GND	Ground
B11	-SMEMW	0
B12	-SMEMR	0
B13	-IOW	I/O
B14	-IOR	I/O
B15	-DACK3	0
B16	DRQ3	I
B17	-DACK1	0
B18	DRQ1	I
B19	-REFRESH	I/O
B20	CLK	0
B21	IRQ7	I
B22	IRQ6	I
B23	IRQ5	I
B24	IRQ4	I
B25	IRQ3	I
B26	-DACK2	0
B27	T/C	0
B28	BALE	0
B29	+5Vdc	Power
B30	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	I
D2	-I/O CS16	I
D3	IRQ10	I
D4	IRQ11	I
D5	IRQ12	I
D6	IRQ15	I
D7	IRQ14	I
D8	-DACK0	O
D9	DRQ0	I
D10	-DACK5	O
D11	DRQ5	I
D12	-DACK6	O
D13	DRQ6	I
D14	-DACK7	O
D15	DRQ7	I
D16	+5 Vdc	POWER
D17	-MASTER	I
D18	GND	GROUND

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

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.

OWS (I)

The 'zero wait state' signal tells the microprocessor that it can complete the present bus cycle without inserting any additional wait cycles. In order to run a memory cycle to a 16-bit device without wait cycles, **OWS** is derived from an address decode gated with a Read or Write command. In order to run a memory cycle to an 8-bit device with a minimum of two wait states, **OWS** 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. OWS is active low and should be driven with an open collector or tri-state driver capable of sinking 20 mA.

The following figure is an I/O address map.

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

Note: I/O Addresses, hex 000 to 0FF, are reserved for the system board I/O. Hex 100 to 3FF are available on the I/O channel.

I/O Address Map (Part 1 of 2)

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

Note: I/O Addresses, hex 000 to 0FF, are reserved for the system board I/O. Hex 100 to 3FF are available on the I/O channel.

I/O Address Map (Part 2 of 2)

NMI and Coprocessor Controls

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

Mask On Write to I/O address hex 070, with data bit 7 equal to a logic 0.

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

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

The following is a description of the Math Coprocessor controls.

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

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

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

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

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

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

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

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

Other Circuits

Speaker

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

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

RAM Jumpers

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

Pin	Assignments
1	No Connection
2	- RAM SEL
3	Ground

RAM Jumper Connector (J18)

The following shows how the jumpers affect RAM.

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

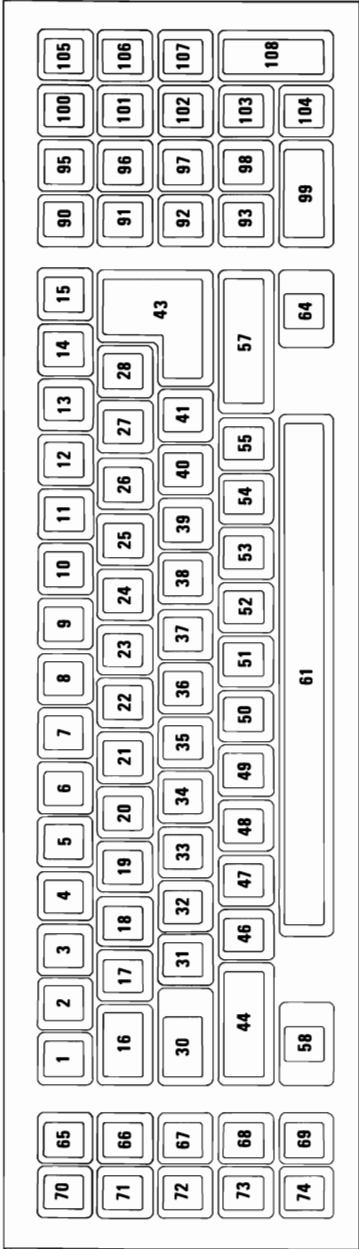
The keyboard sends data in a serial format using an 11-bit frame. The first bit is a start bit, and is followed by eight data bits, an odd parity bit, and a stop bit. Data sent is synchronized by a clock supplied by the keyboard. At the end of a transmission, the keyboard controller disables the interface 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	0B	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	7B	107
4B	6B	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.

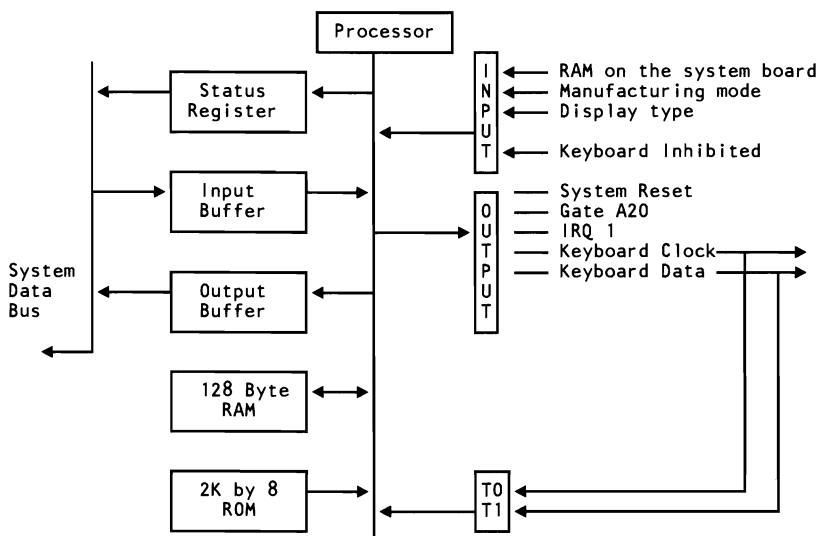
The keyboard is required to respond to all transmissions. The keyboard responds to any valid command and parameter, other than Echo and Resend, with an Acknowledge (ACK) response, hex FA. If the response contains a parity error, the keyboard controller places a hex FE in its output buffer, and the transmit 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 the square-wave frequency as set by the rate selection bits in register A, and a 0 disables the square wave. The system initializes this bit to 0.

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

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

Status Register C

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

Status Register D

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

CMOS RAM Configuration Information

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

Diagnostic Status Byte (Hex 0E)

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

- Bit 6** Configuration Record (Checksum Status Indicator)—A 0 indicates that checksum is good, and a 1 indicates it is bad.
- Bit 5** Incorrect Configuration Information—This is a check, at power-on time, of the equipment byte of the configuration record. A 0 indicates that the configuration information is valid, and a 1 indicates it is invalid. Power-on checks require:
- At least one diskette drive to be installed (bit 0 of the equipment byte set to 1).
 - The primary display adapter setting in configuration matches the system board's display switch setting and the actual display adapter hardware in the system.
- Bit 4** Memory Size Comparison—A 0 indicates that the power-on check determined the same memory size as in the configuration record, and a 1 indicates the memory size is different.
- Bit 3** Fixed Disk Adapter/Drive C Initialization Status—A 0 indicates that the adapter and drive are functioning properly and the system can attempt "boot up." A 1 indicates that the adapter and/or drive C failed initialization, which prevents the system from attempting to "boot up."
- Bit 2** Time Status Indicator (POST validity check)— A 0 indicates that the time is valid, and a 1 indicates that it is invalid.
- Bit 1–Bit 0** Reserved

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** listing at label FD_TBL)
- 1111** Type 16 through 255. See “Drive C Extended Byte (Hex 19)” on page 1-65 .

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

- 0000** No fixed disk drive is present.
- 0001** Define type 1 through type 14 as shown to in the following table (also see BIOS listing at label FD_TBL)
- 1110** listing at label FD_TBL)
- 1111** 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	All Cyl	663
17	977	5	300	977
18	977	7	None	977
19	1024	7	512	1023
20	733	5	300	732
21	733	7	300	732
22	733	7	300	733
23	Reserved			
.				
255				

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

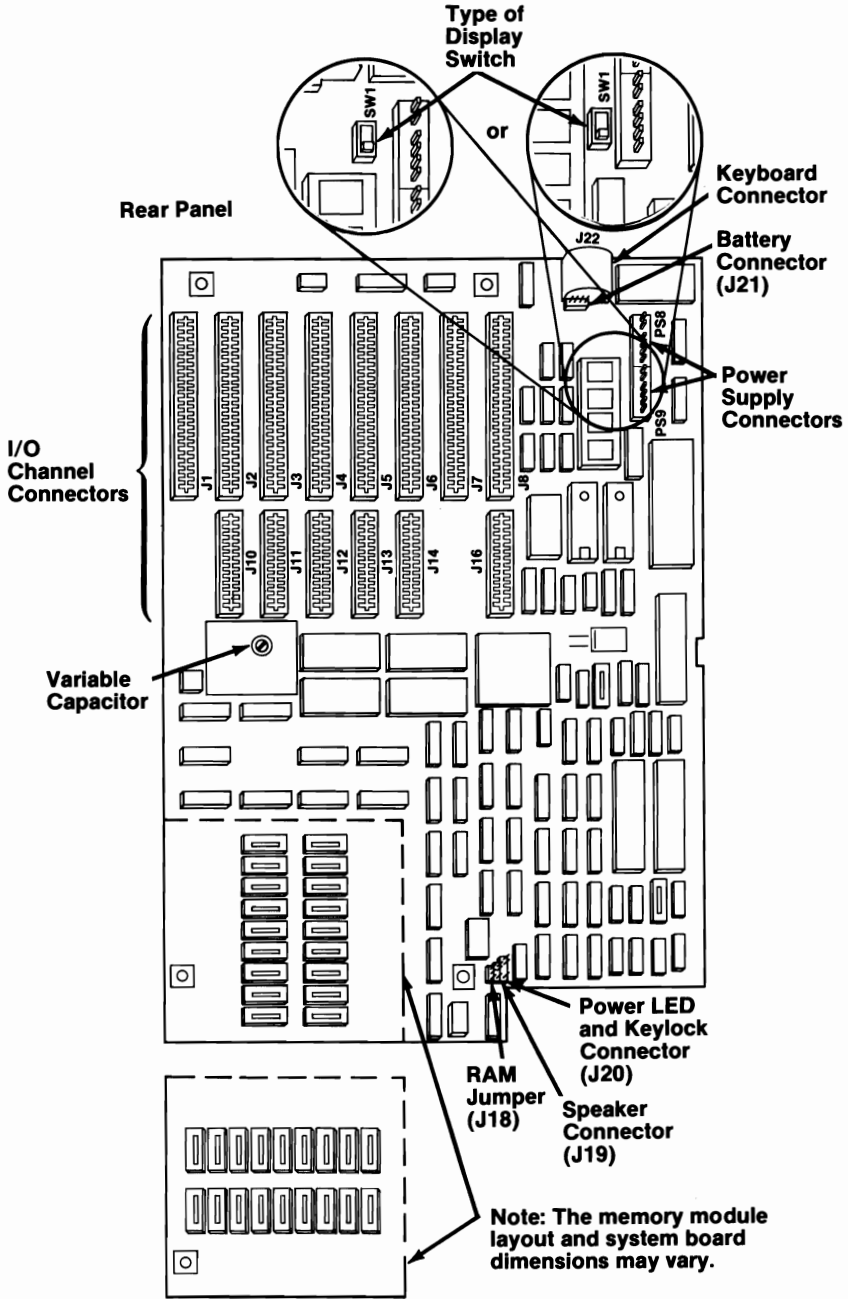
Battery Connector (J21)

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:

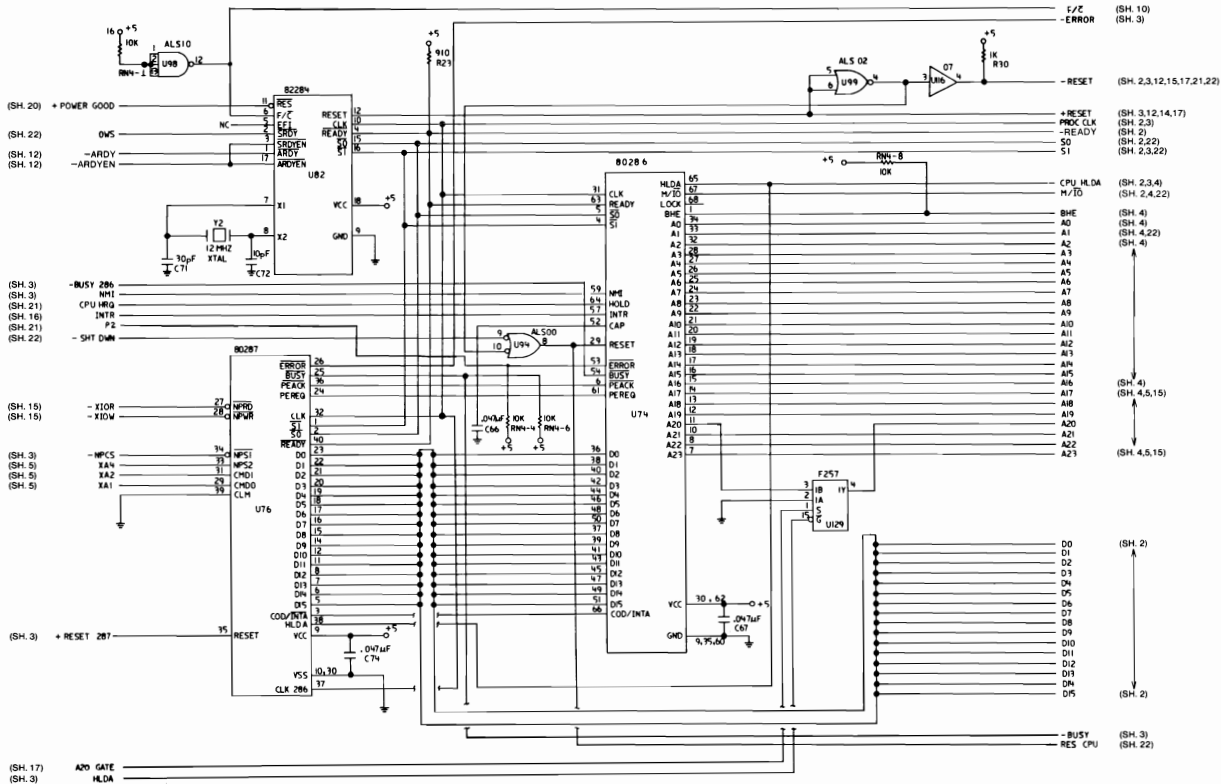
Pin	Assignments
1	Keyboard Clock
2	Keyboard Data
3	Reserved
4	Ground
5	+5 Vdc

Keyboard Connector (J22)

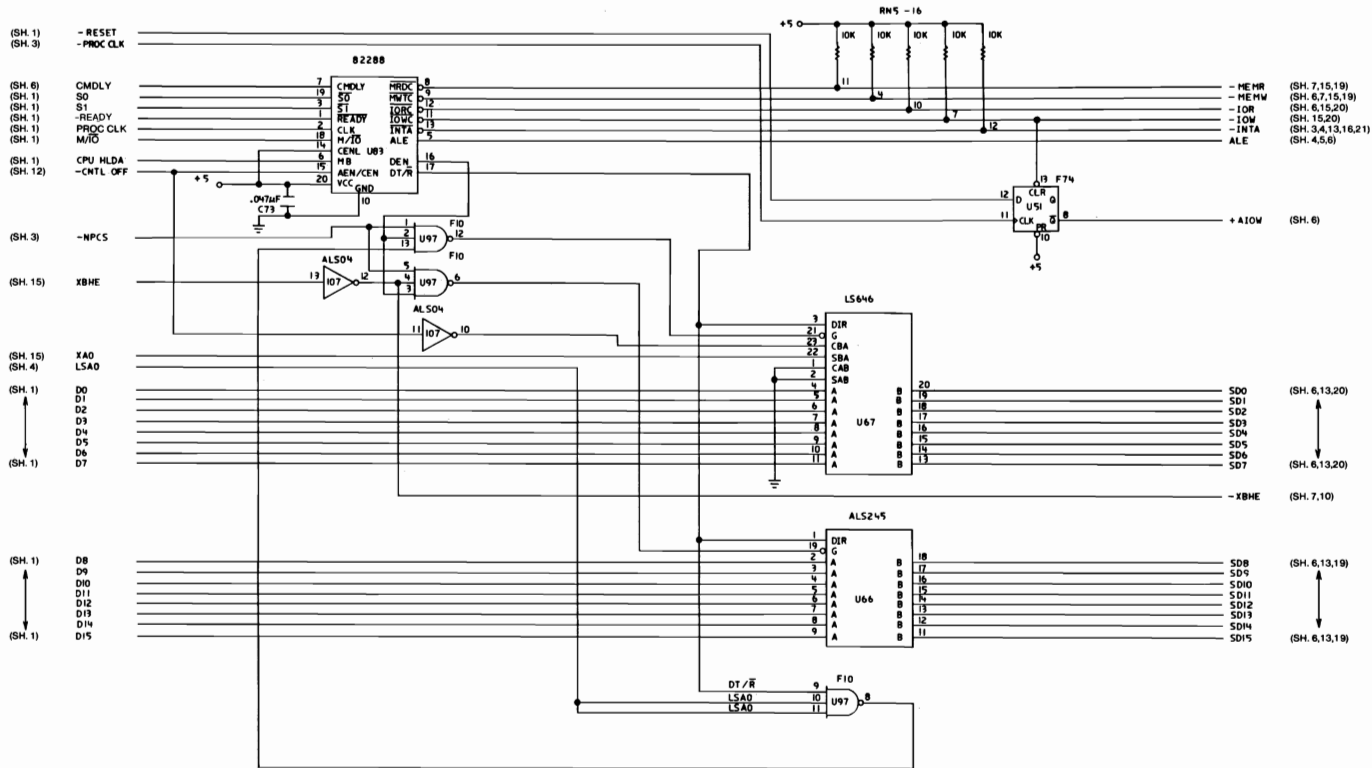
The following figure shows the layout of the system board.



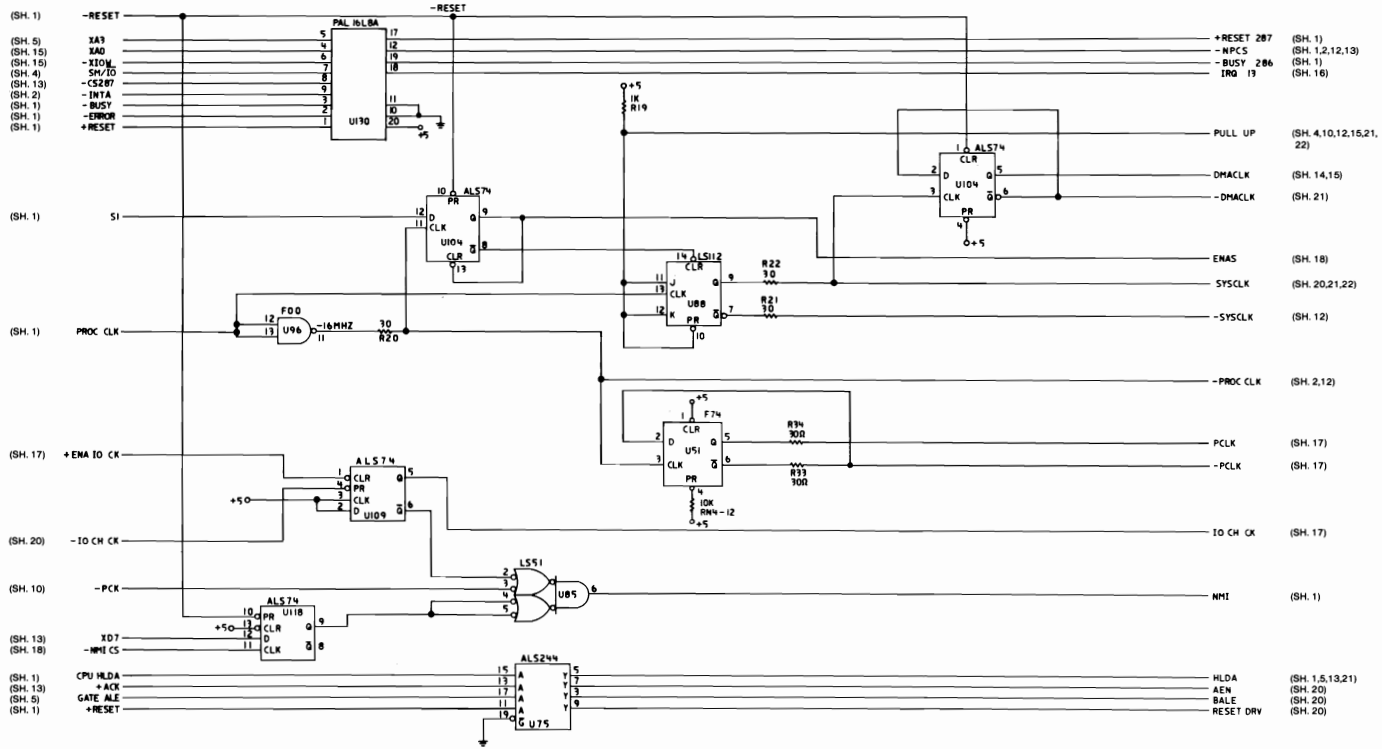
Notes:



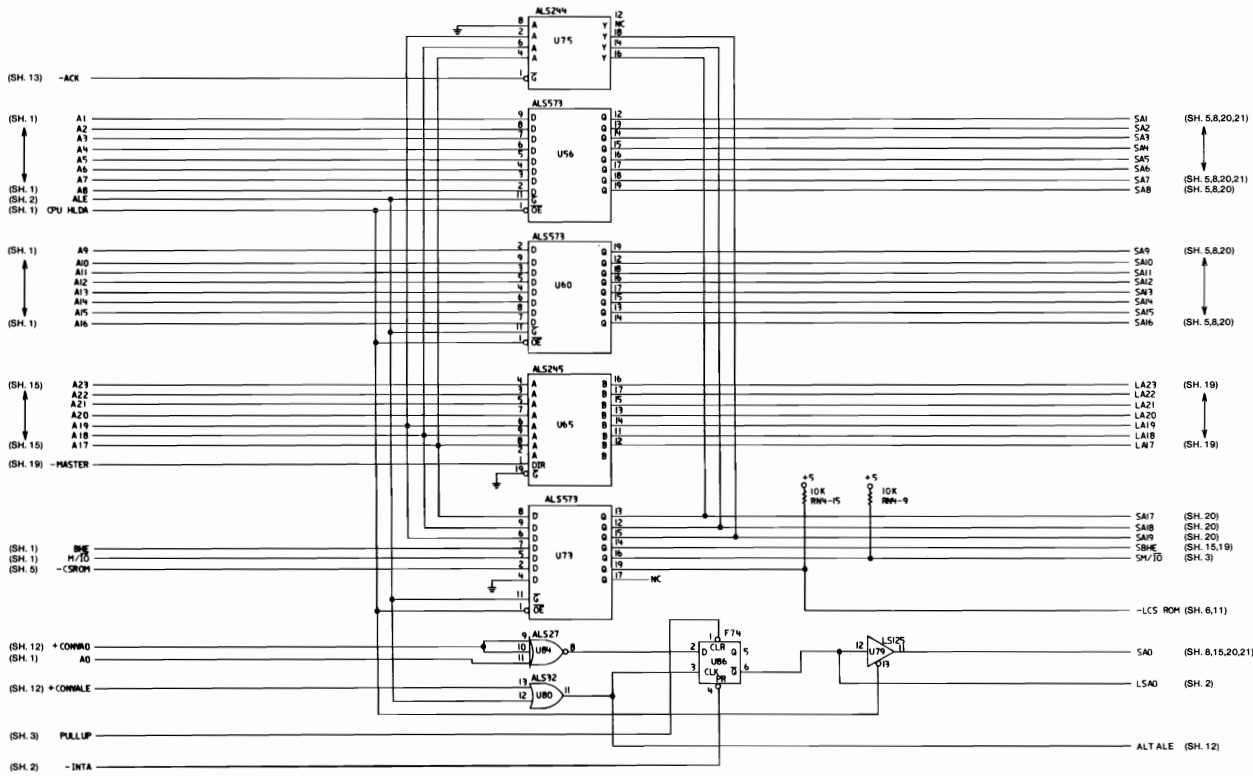
Type 1 512KB Planar (Sheet 1 of 22)



Type 1 512KB Planar (Sheet 2 of 22)

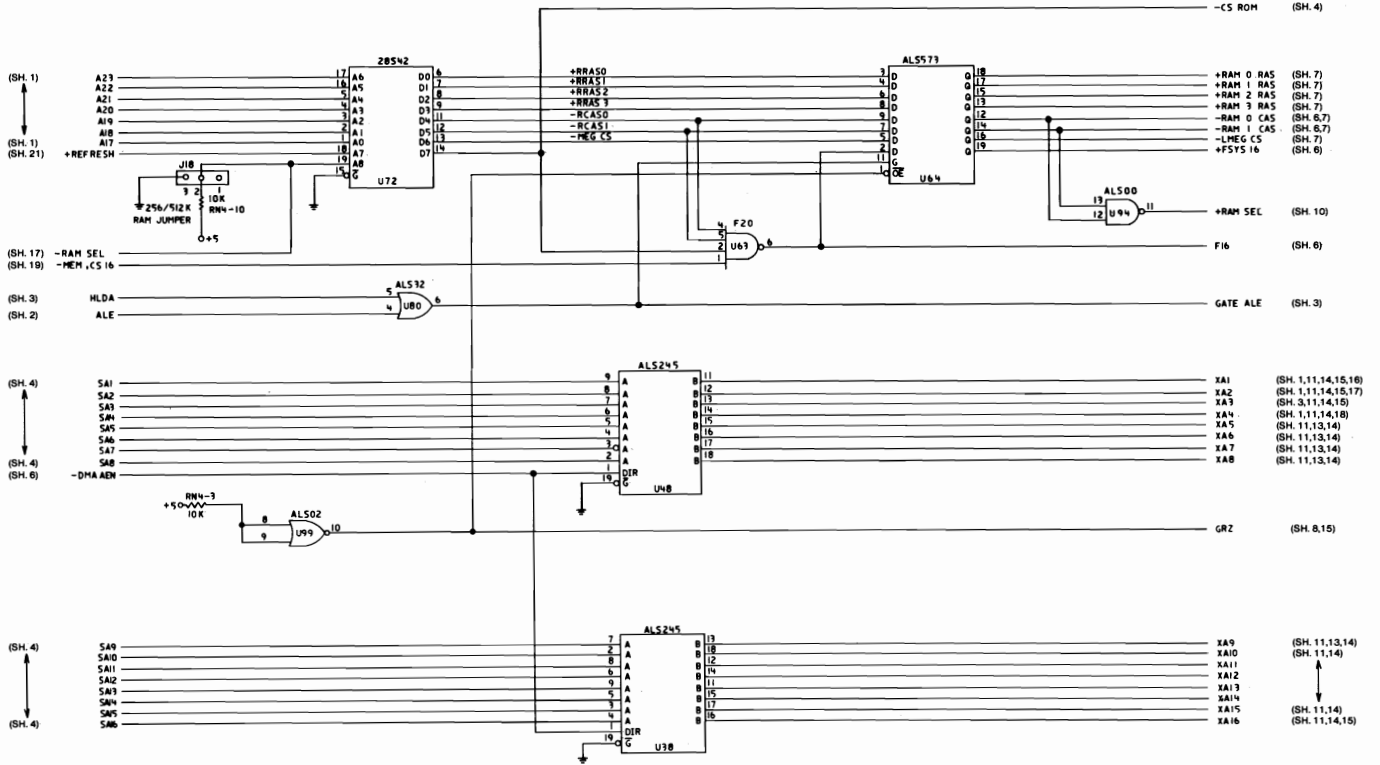


Type 1 512KB Planar (Sheet 3 of 22)

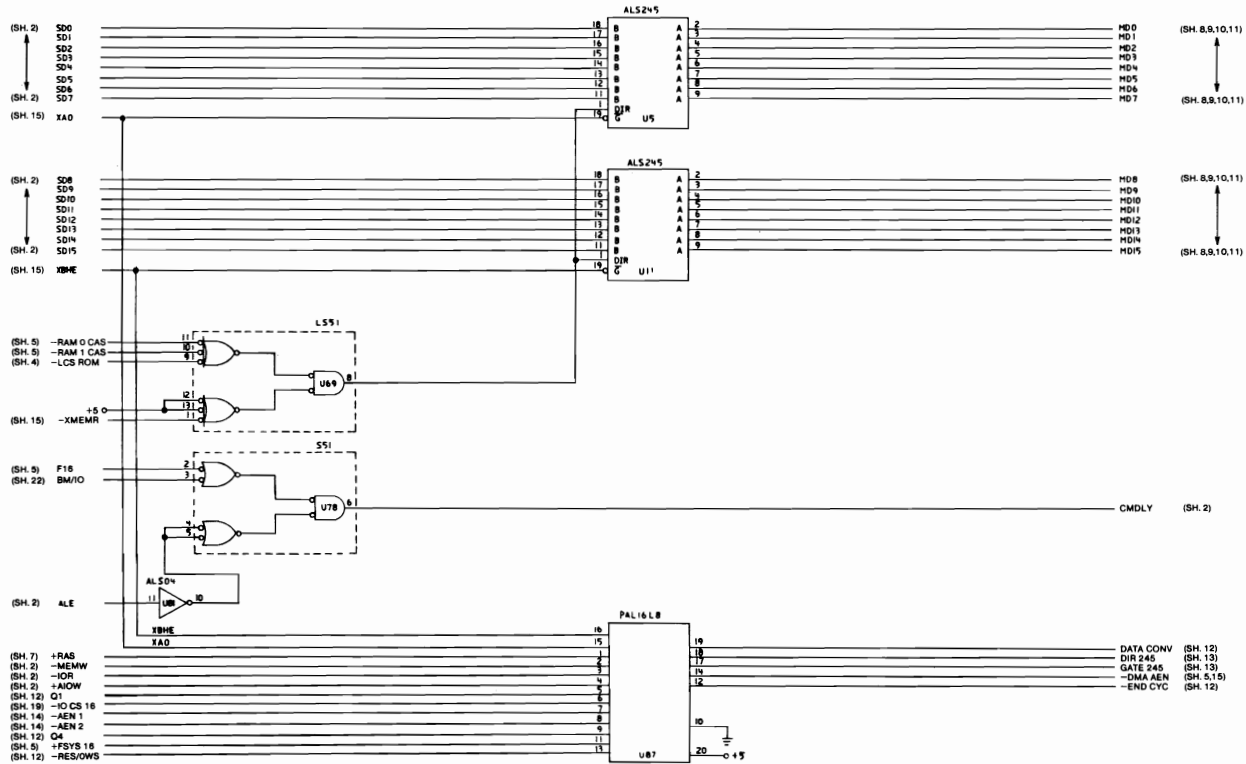


Type 1 512KB Planar (Sheet 4 of 22)

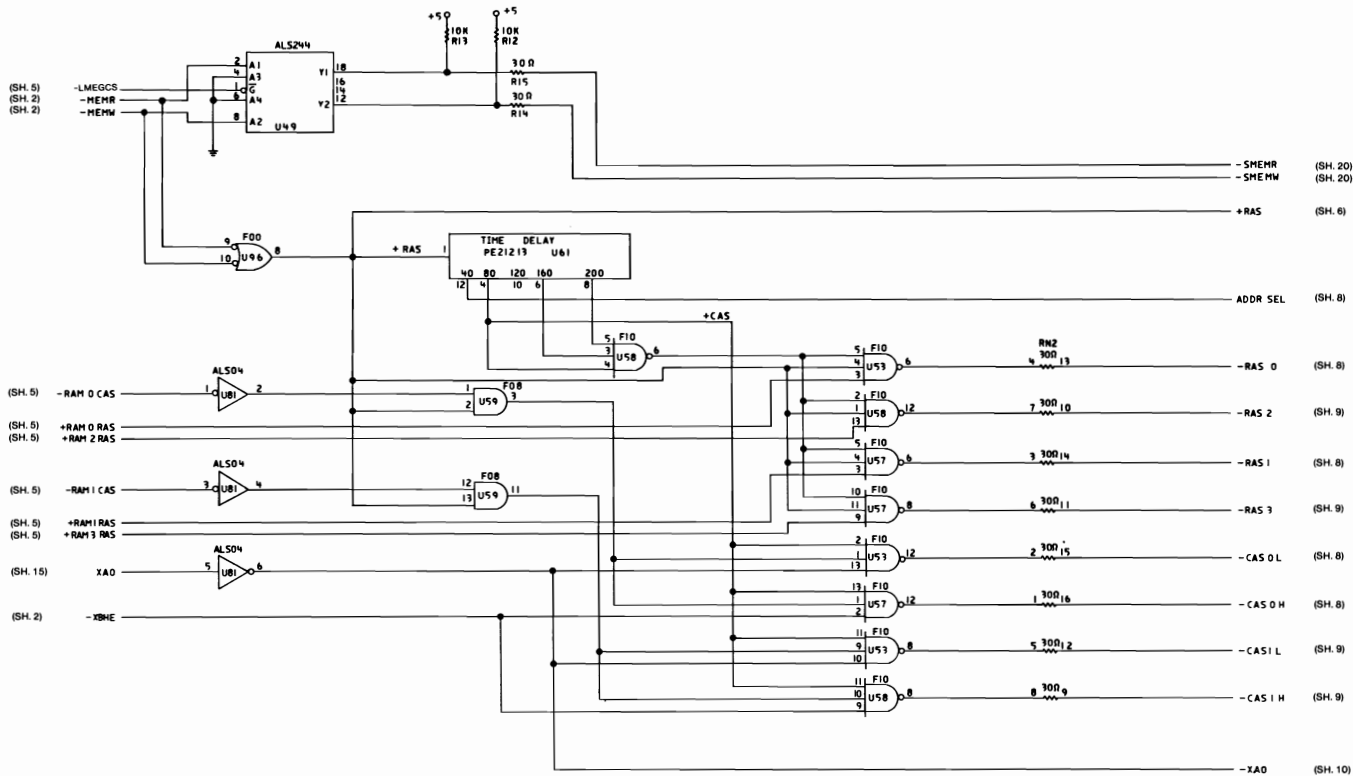
1-80 System Board



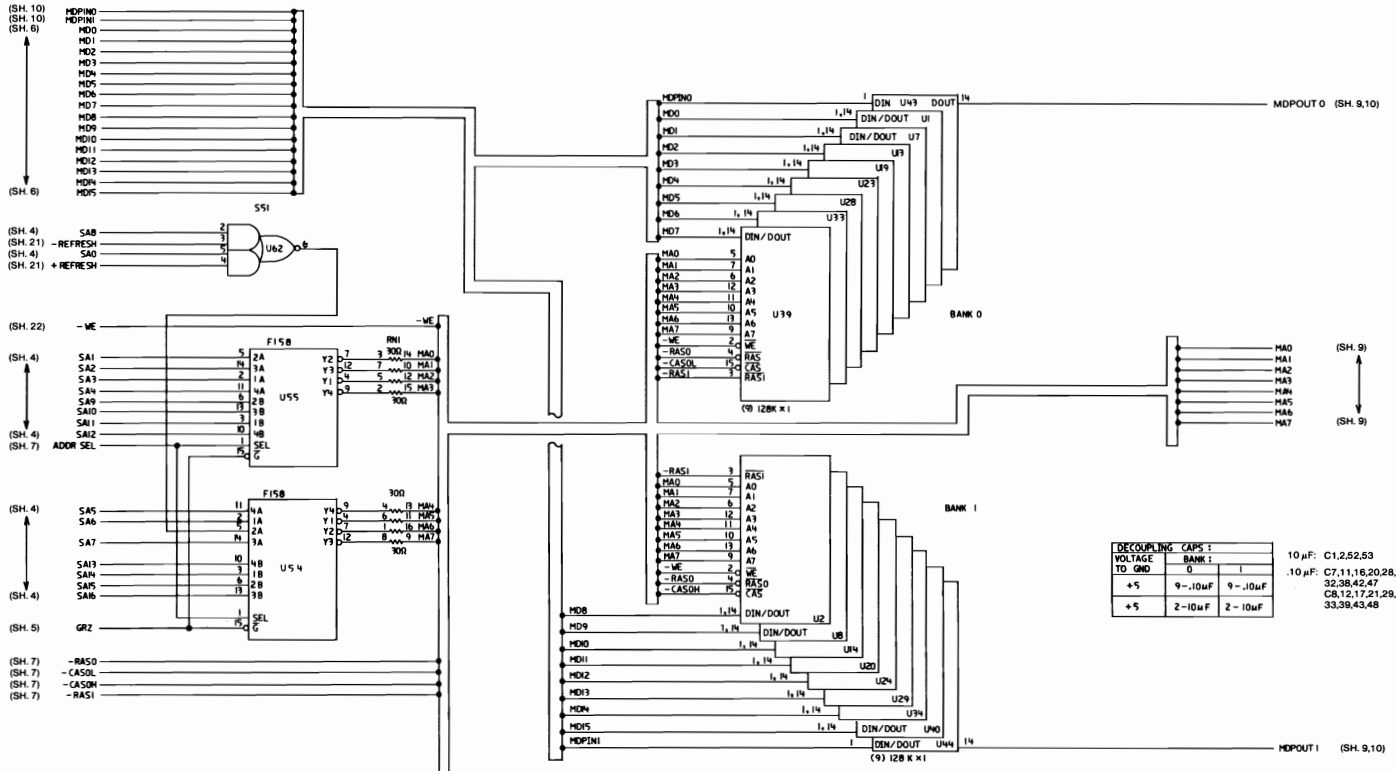
Type 1 512KB Planar (Sheet 5 of 22)



Type 1 512KB Planar (Sheet 6 of 22)

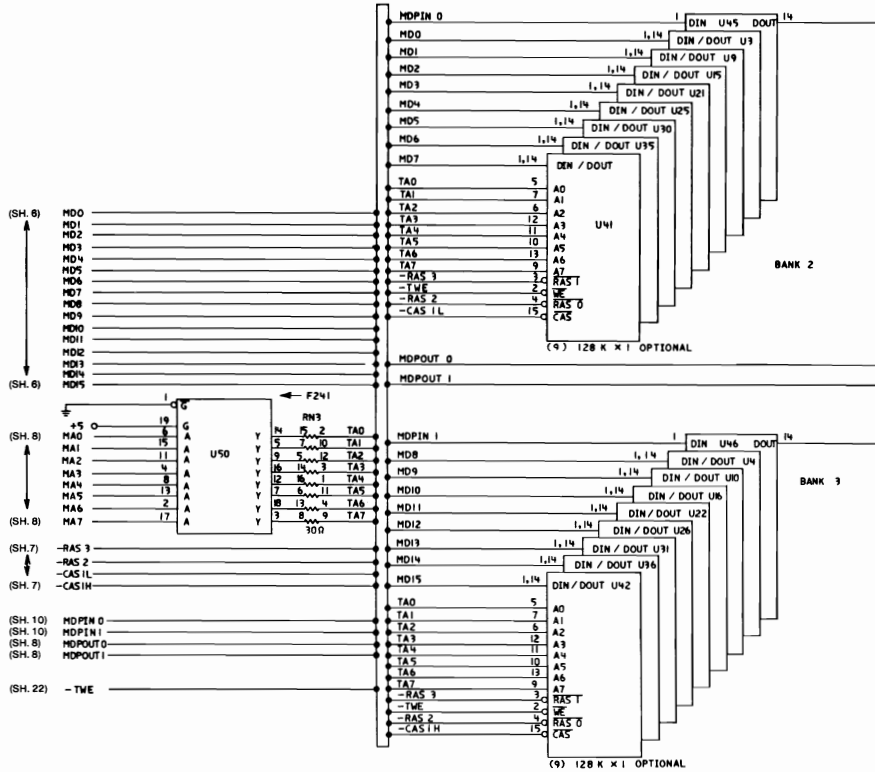


Type 1 512KB Planar (Sheet 7 of 22)



DECOUPLING CAPS :			
VOLTAGE TO GND	BANK :	CAPS :	
+5	9-10uF	9-10uF	10 μF: C1,2,52,53
+5	2-10uF	2-10uF	10 μF: C7,11,16,20,28, 32,38,42,47, C8,12,17,21,29, 33,39,43,48

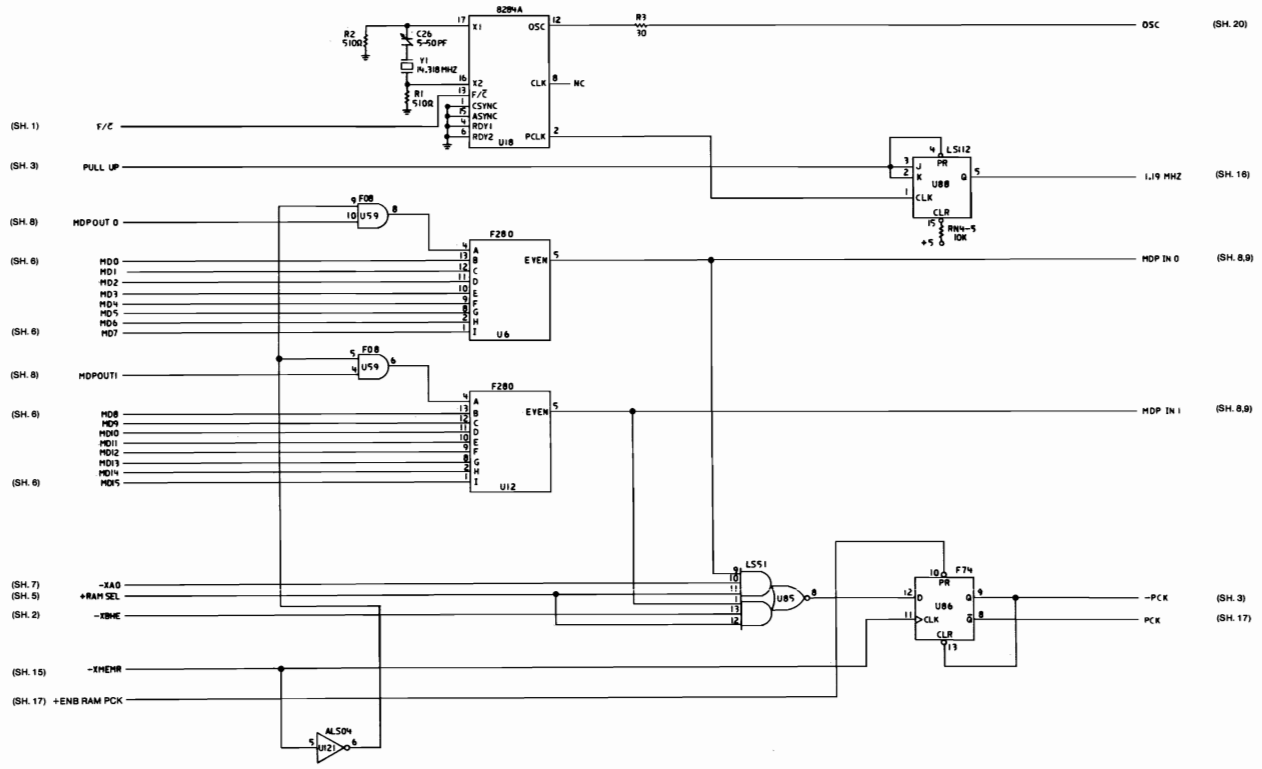
Type 1 512KB Planar (Sheet 8 of 22)



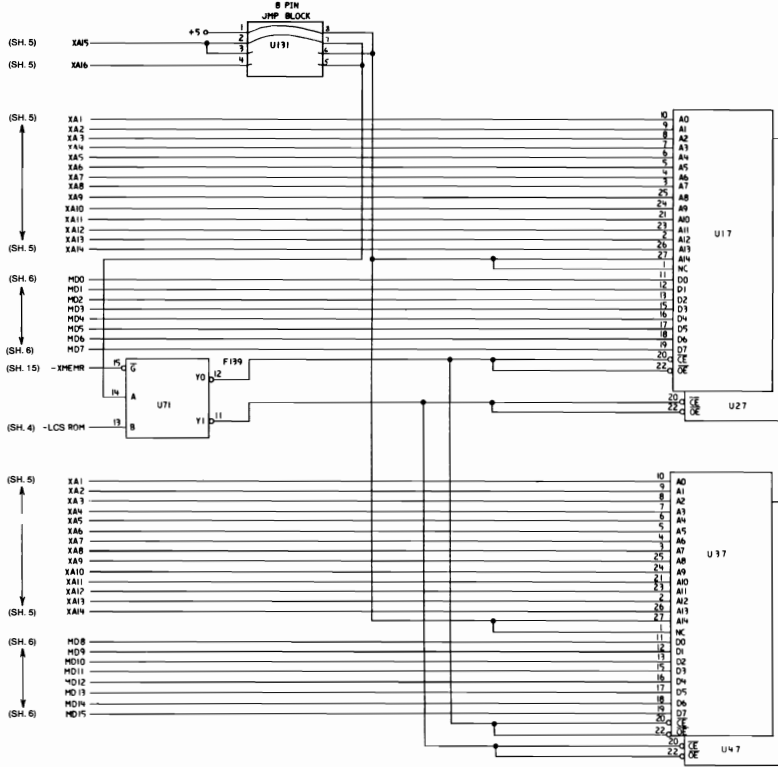
DECOUPLING CAP:		
VOLTAGE TO GND	2 BANK	3
+ 5	9 - .10uF	9 - .10uF
+ 5	2 - 10uF	2 - 10uF

- 10uF: C3,4,5,55
- 10uF: C9,13,18,22,30, 34,40,44,49
- C10,14,19,23,31, 35,41,45,50

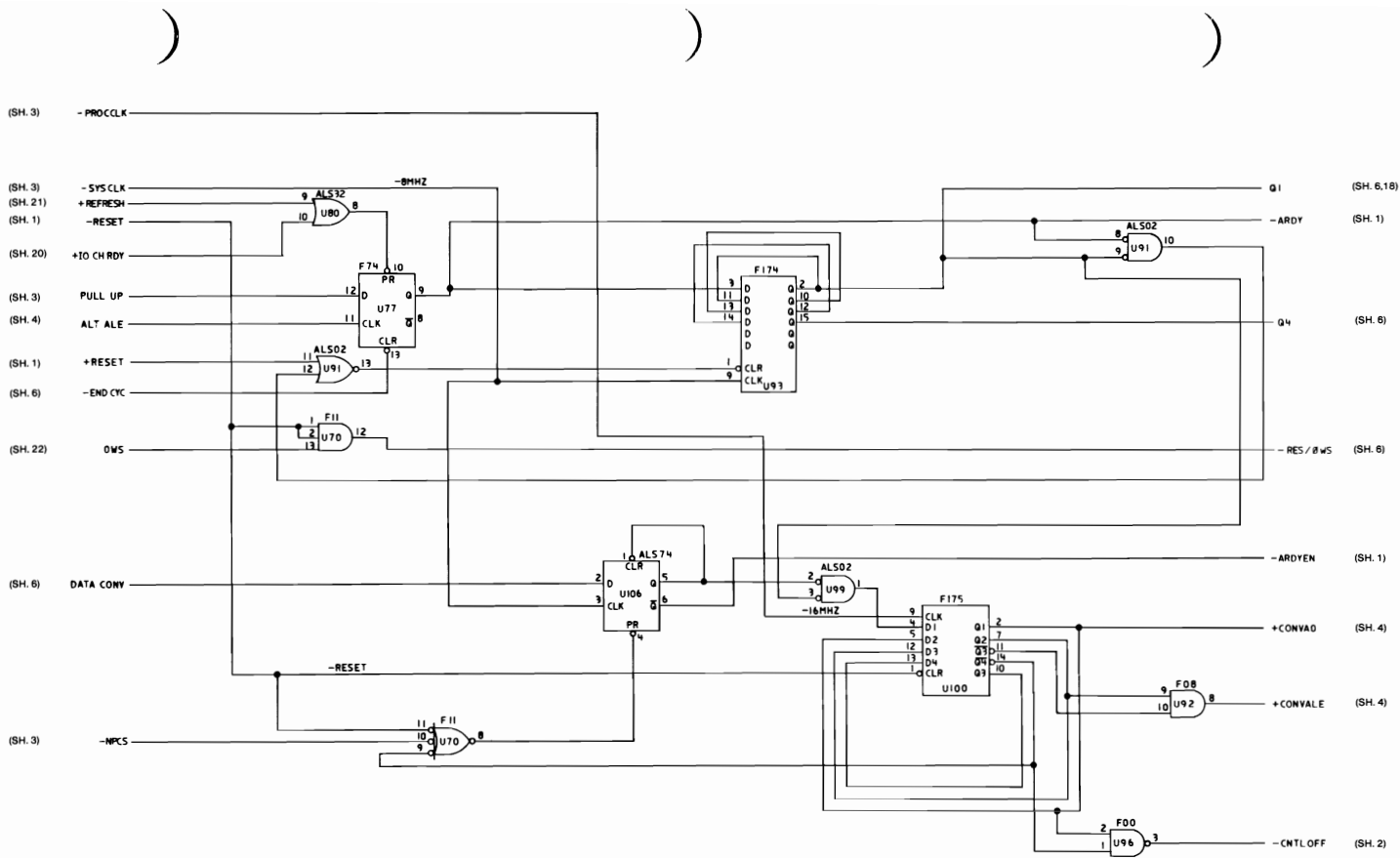
Type 1 512KB Planar (Sheet 9 of 22)



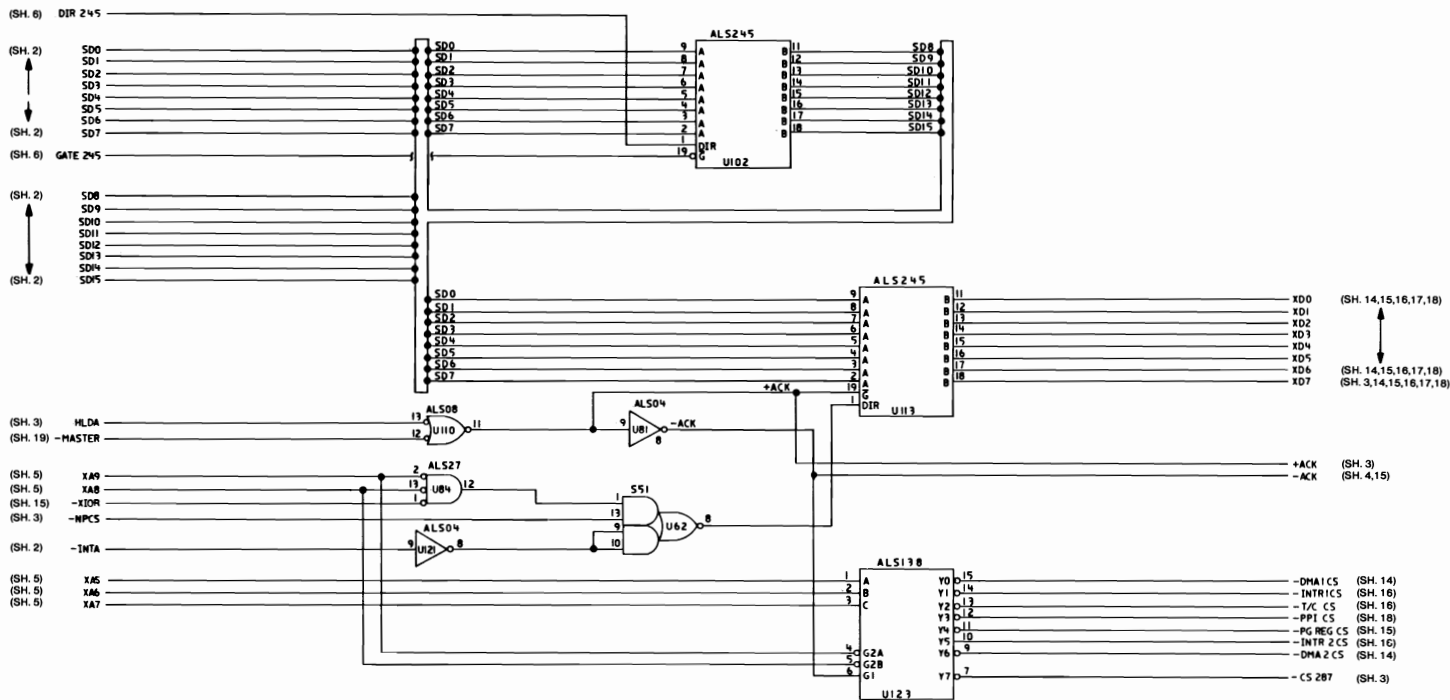
Type 1 512KB Planar (Sheet 10 of 22)



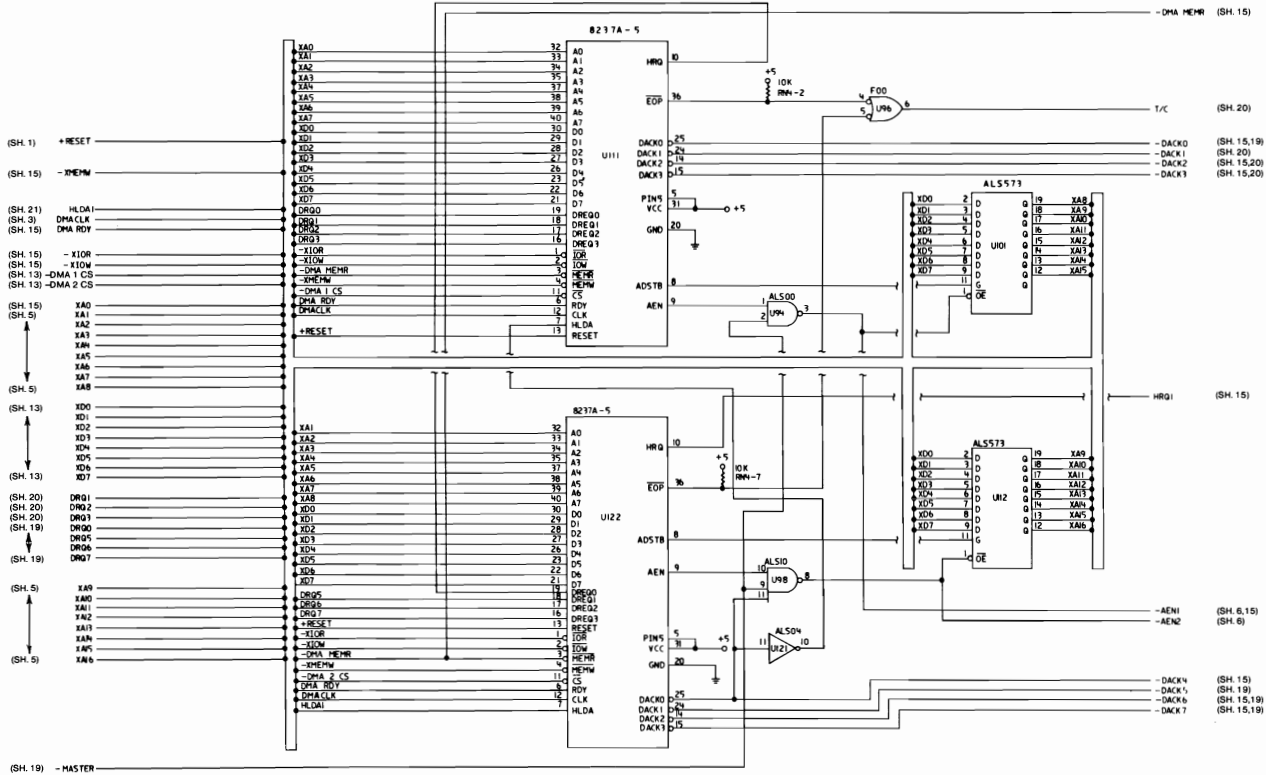
Type 1 512KB Planar (Sheet 11 of 22)



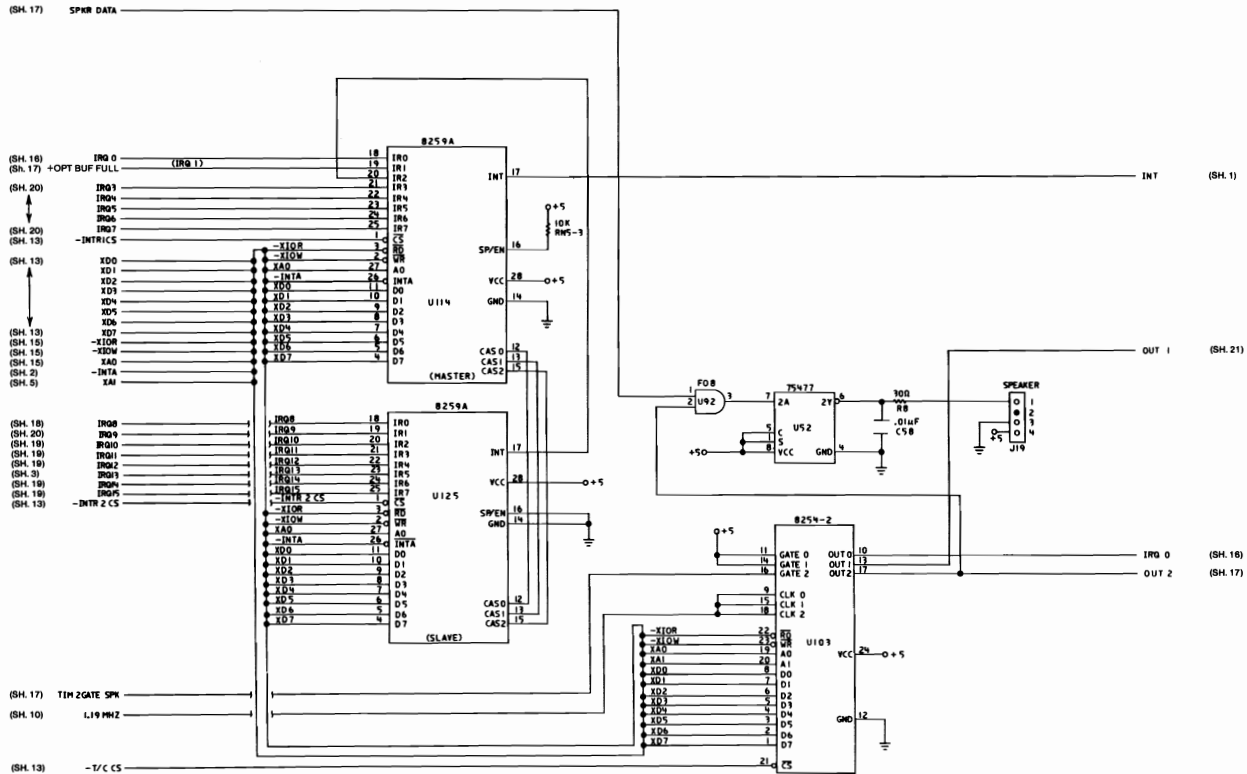
Type 1 512KB Planar (Sheet 12 of 22)



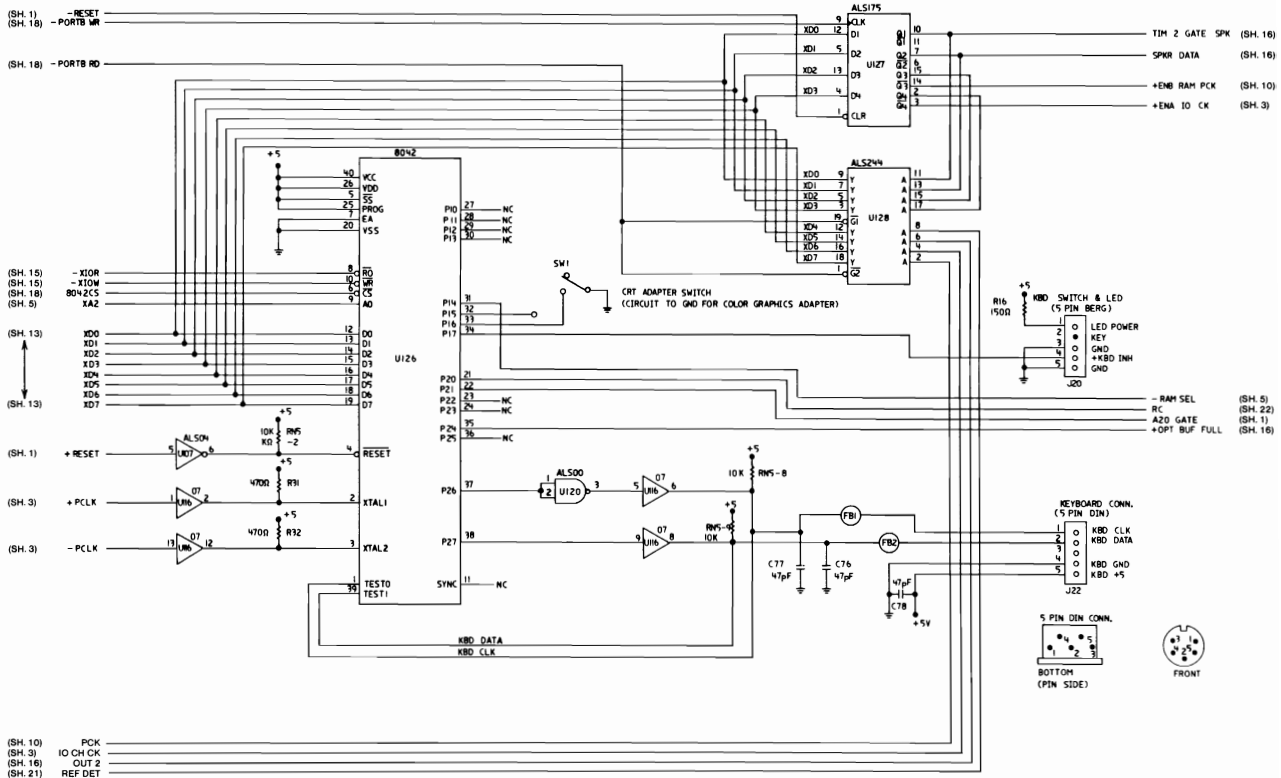
Type 1 512KB Planar (Sheet 13 of 22)



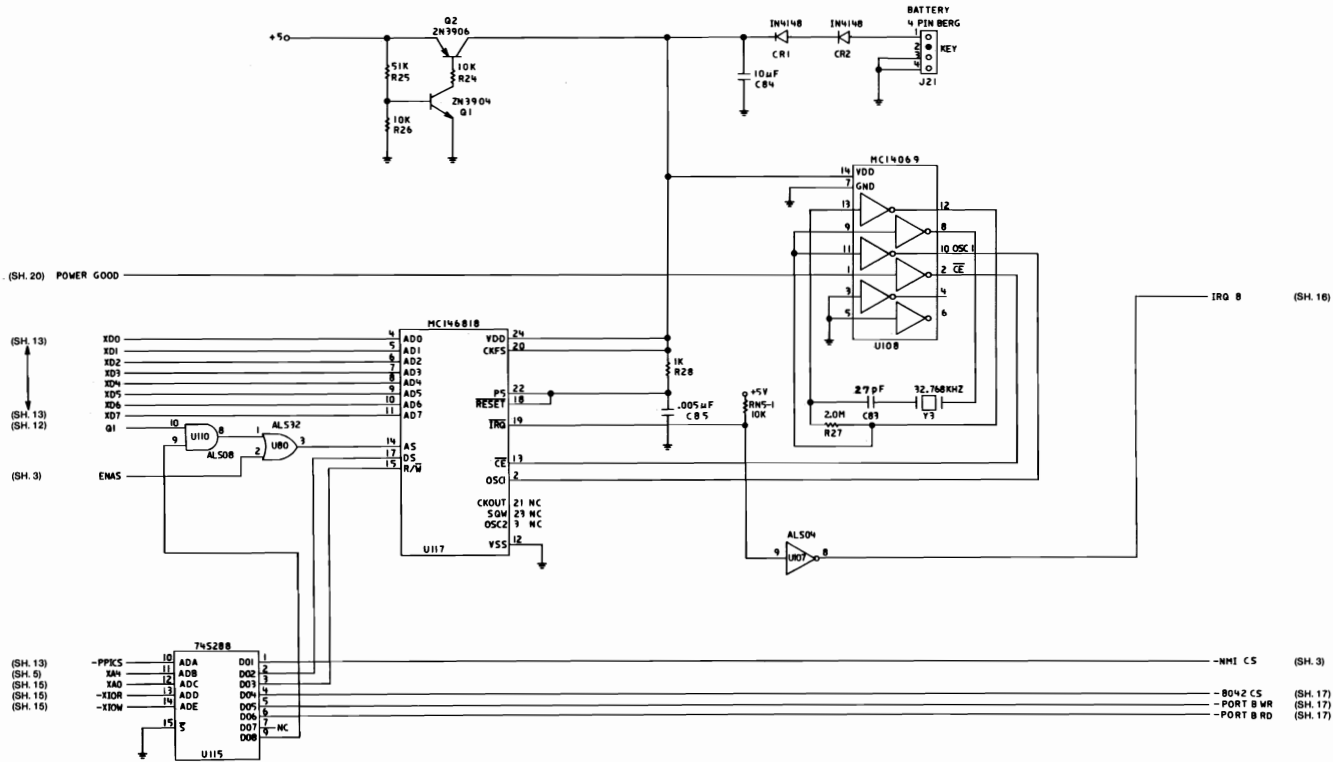
Type 1 512KB Planar (Sheet 14 of 22)



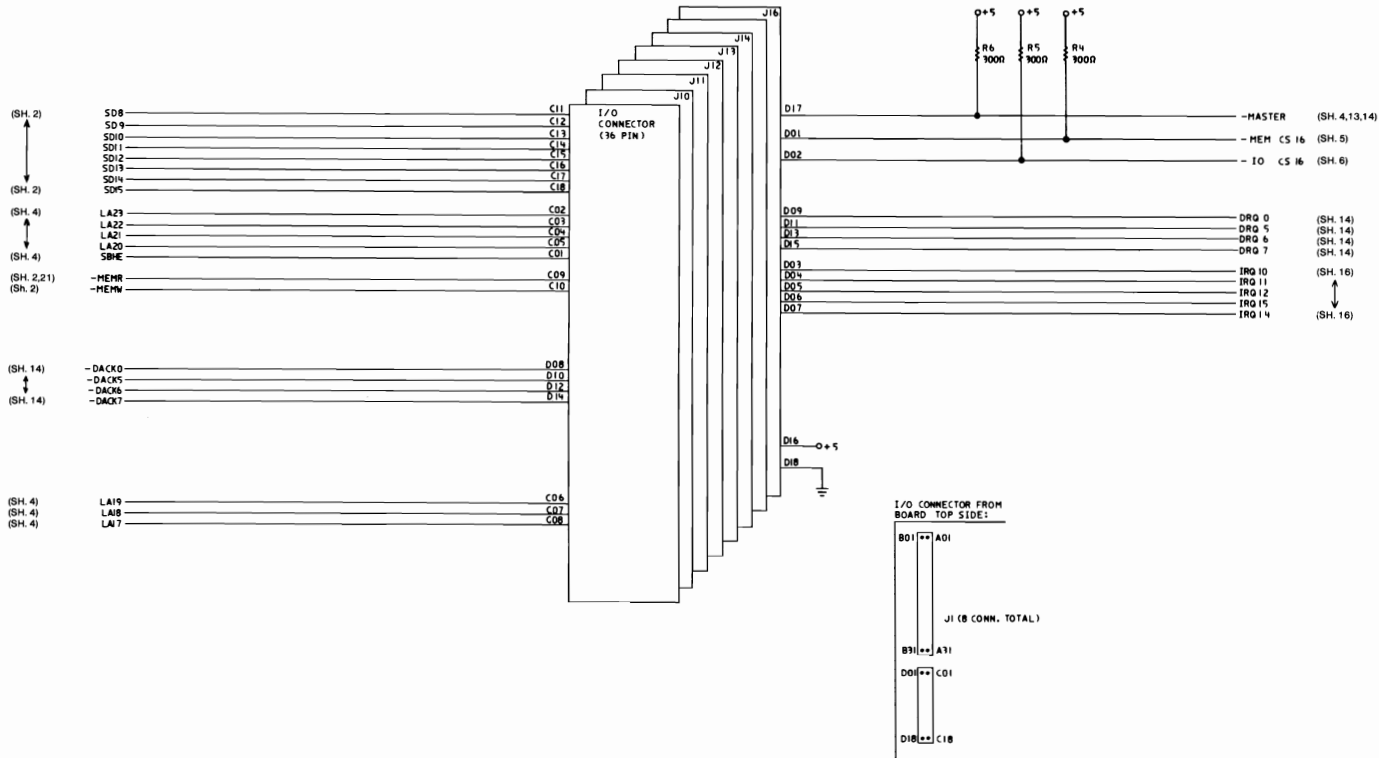
Type 1 512KB Planar (Sheet 16 of 22)



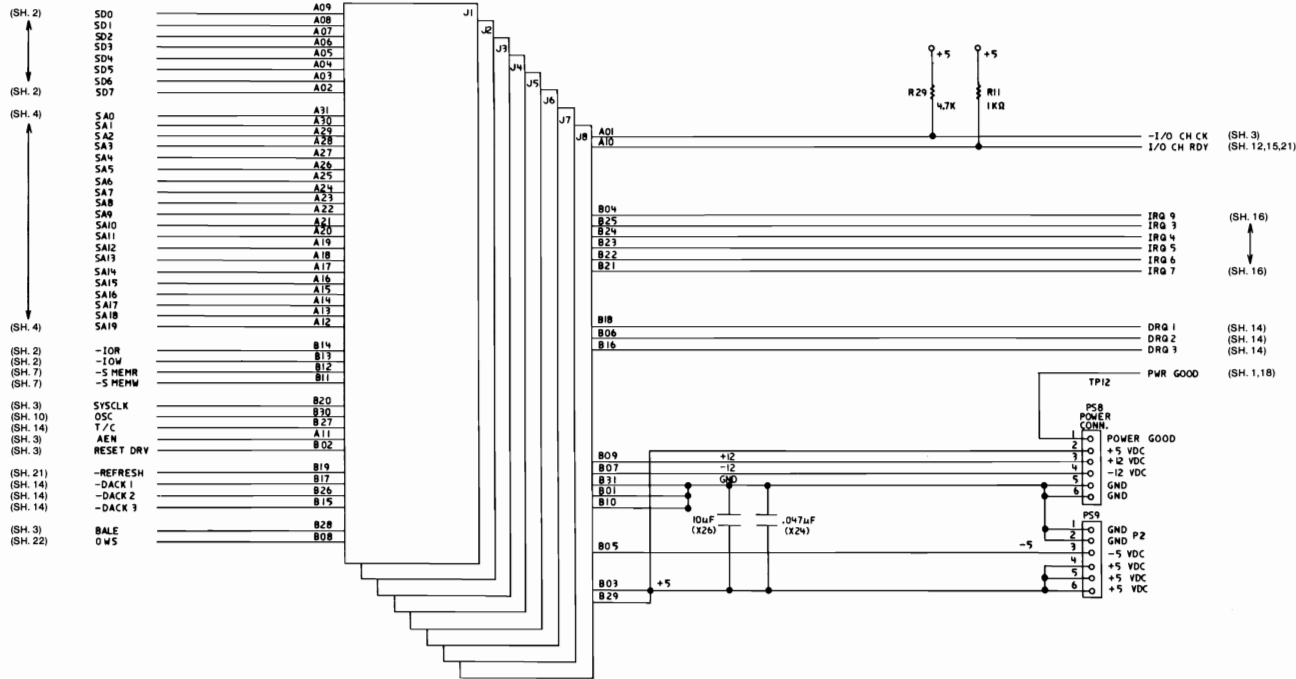
Type 1 512KB Planar (Sheet 17 of 22)



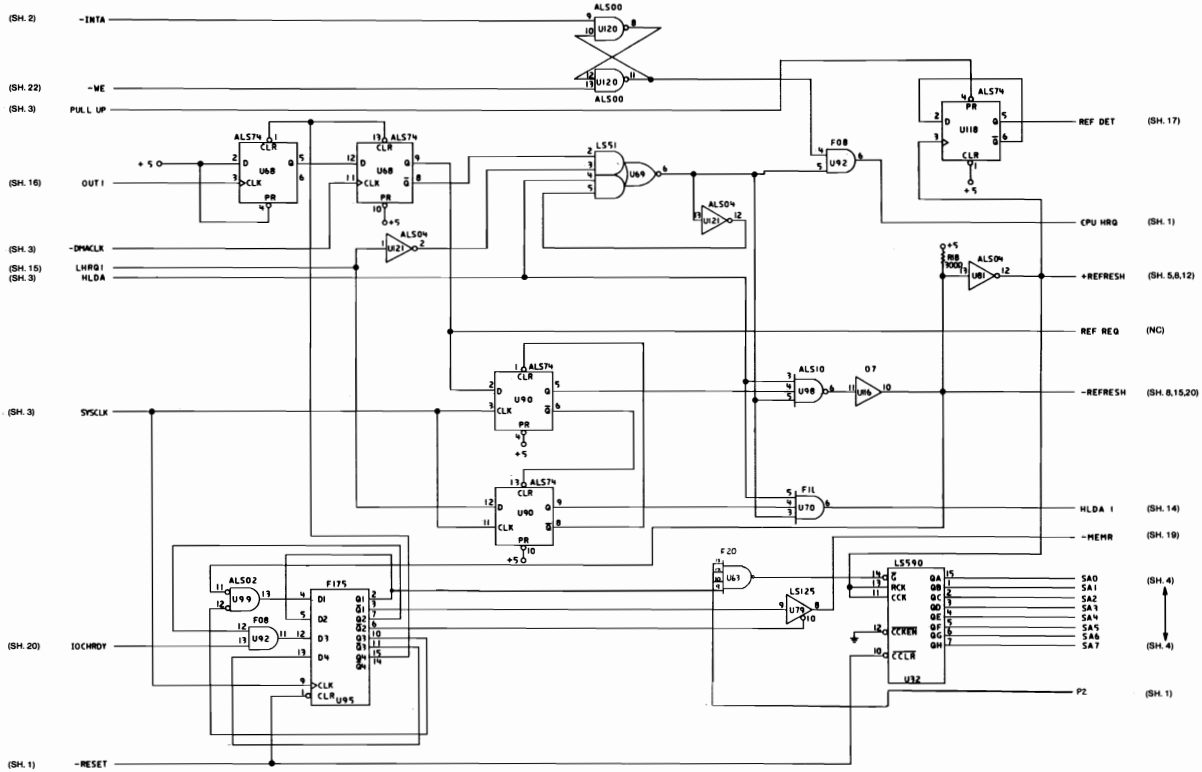
Type 1 512KB Planar (Sheet 18 of 22)



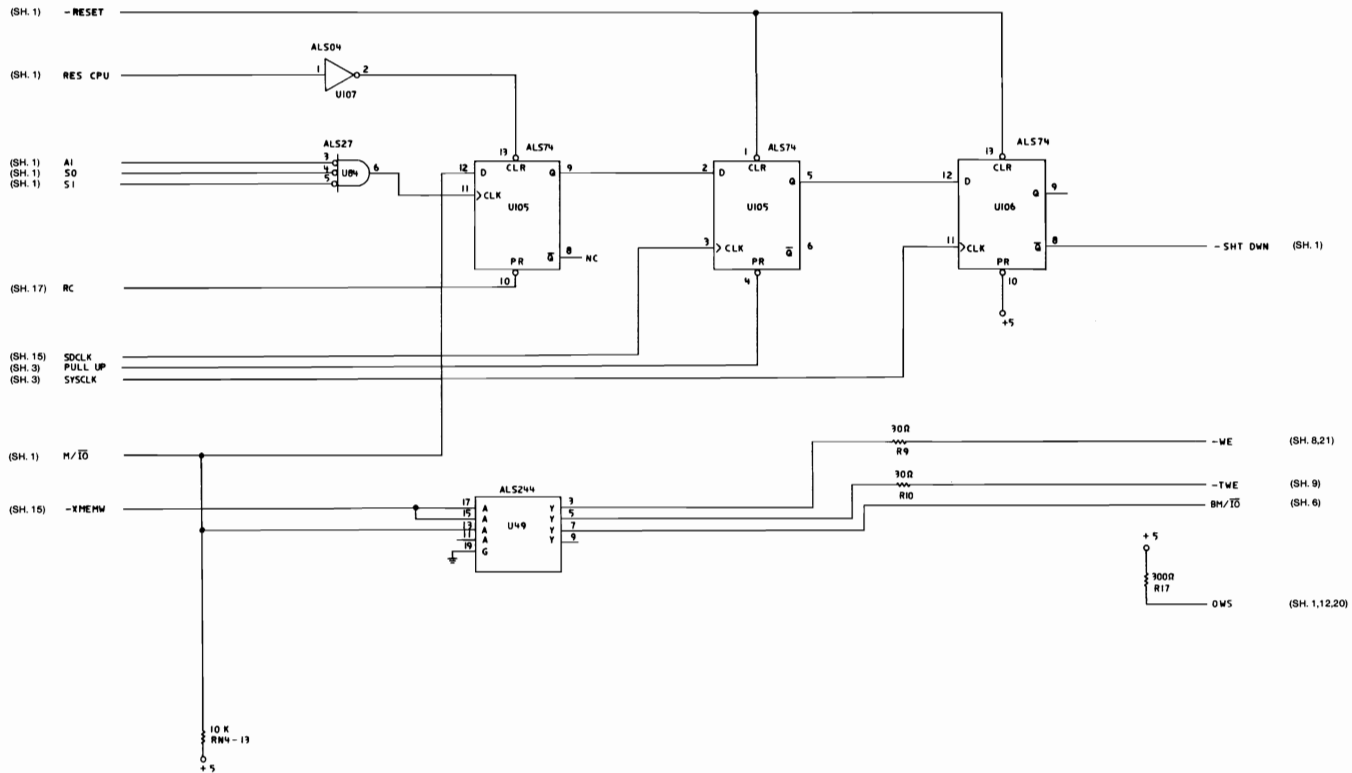
Type 1 512KB Planar (Sheet 19 of 22)



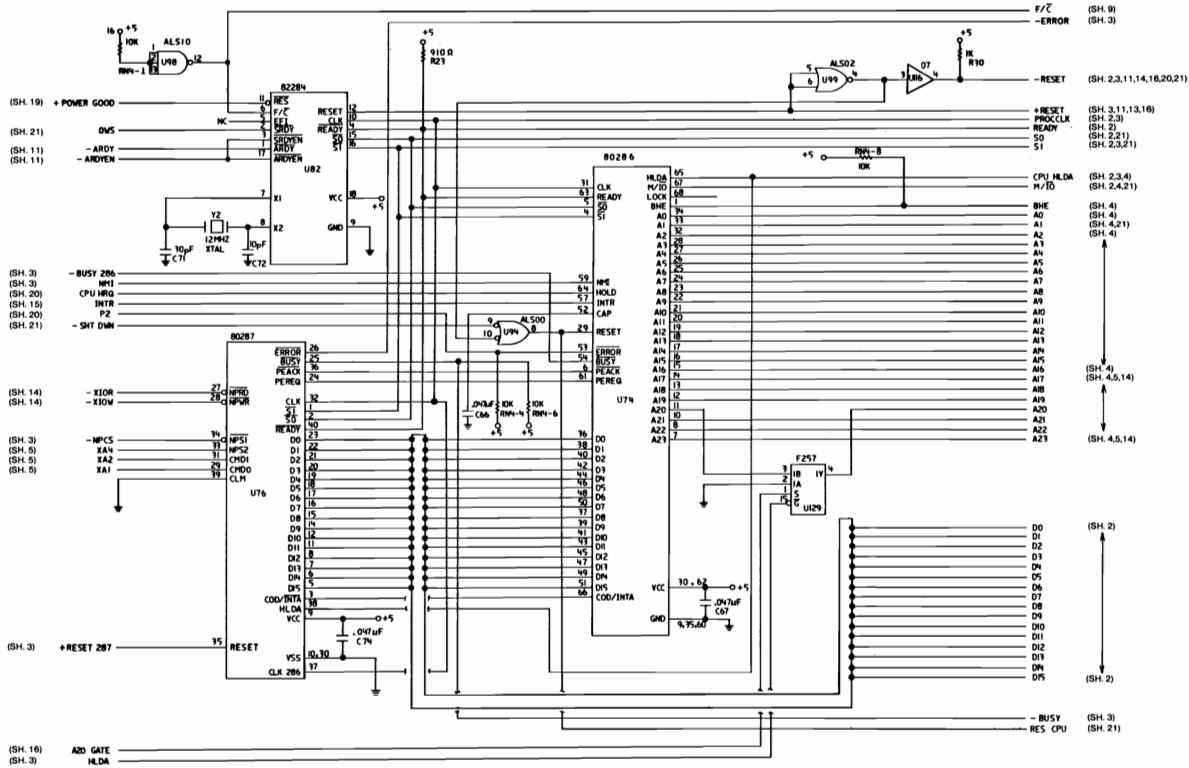
Type 1 512KB Planar (Sheet 20 of 22)



Type 1 512KB Planar (Sheet 21 of 22)

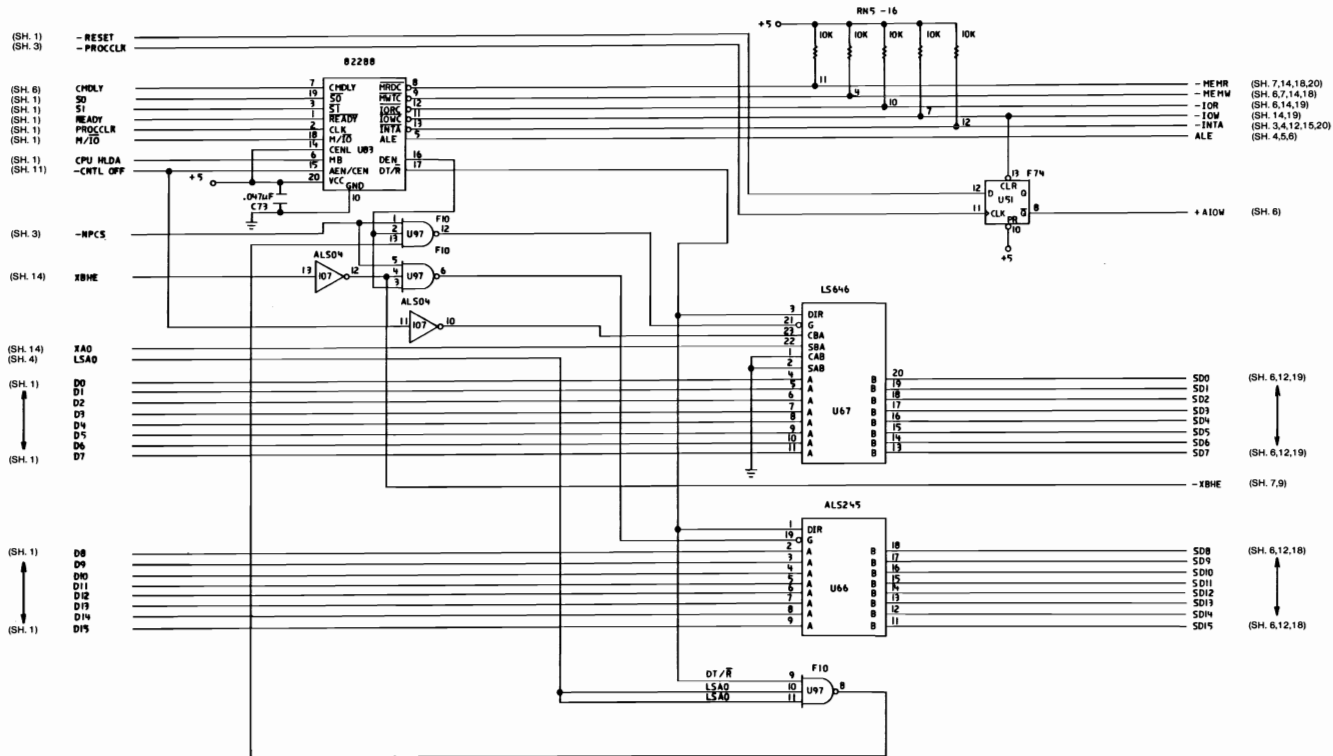


Type 1 512KB Planar (Sheet 22 of 22)



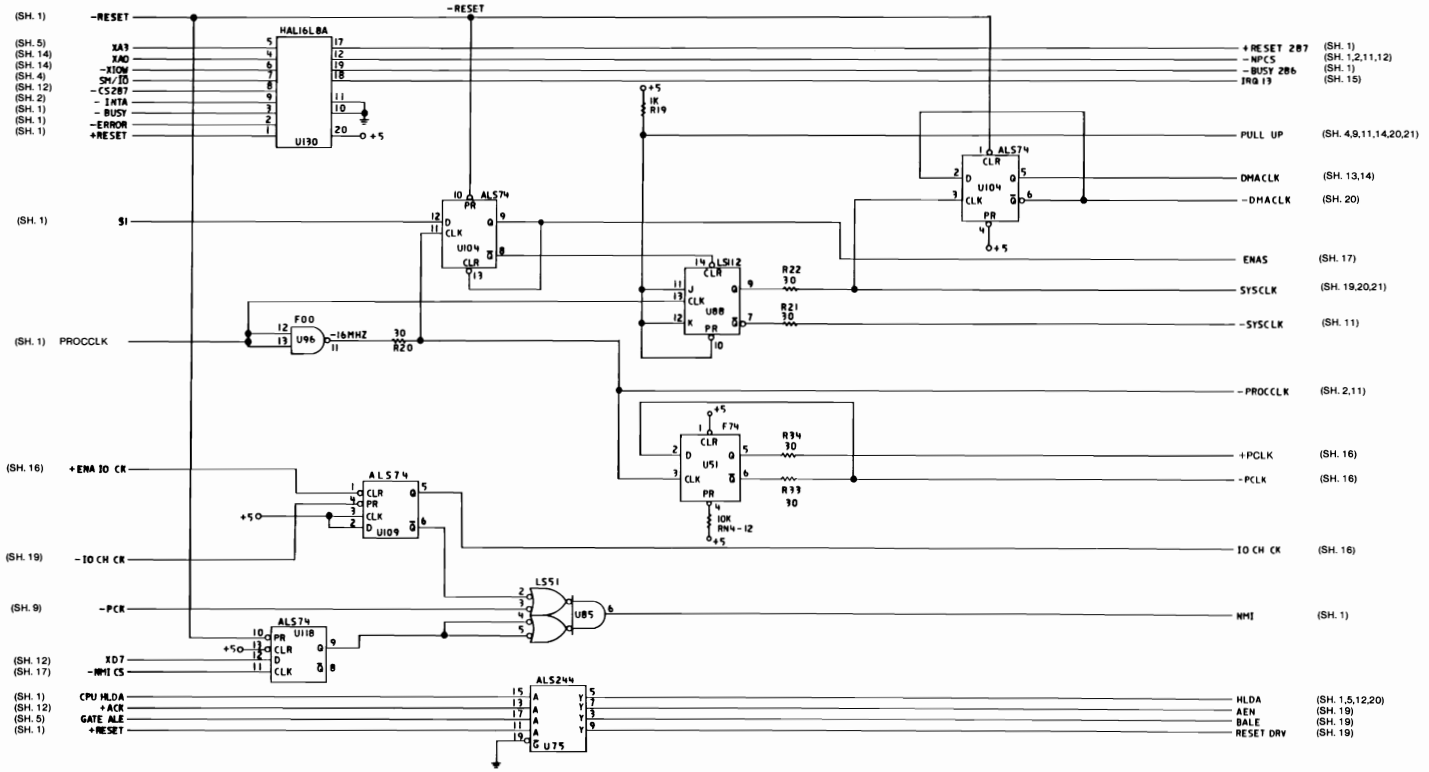
Type 2 512KB Planar (Sheet 1 of 21)



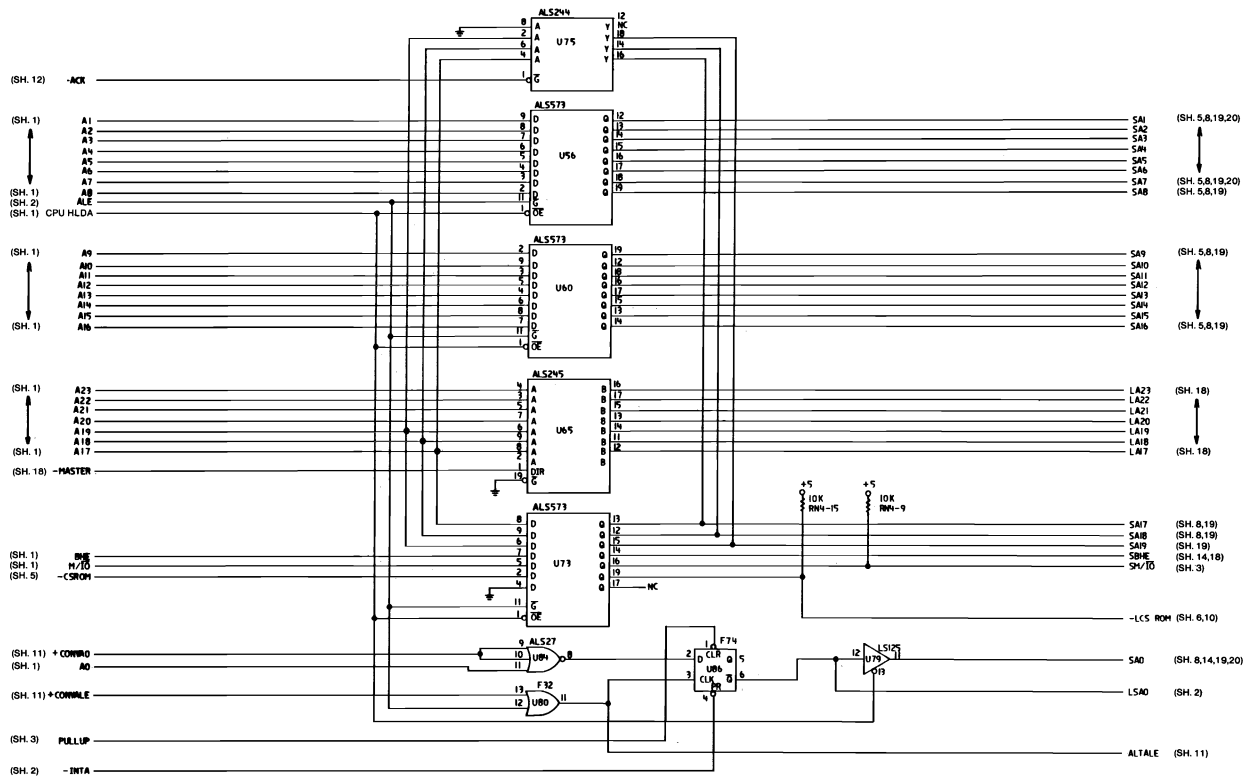


Type 2 512KB Planar (Sheet 2 of 21)

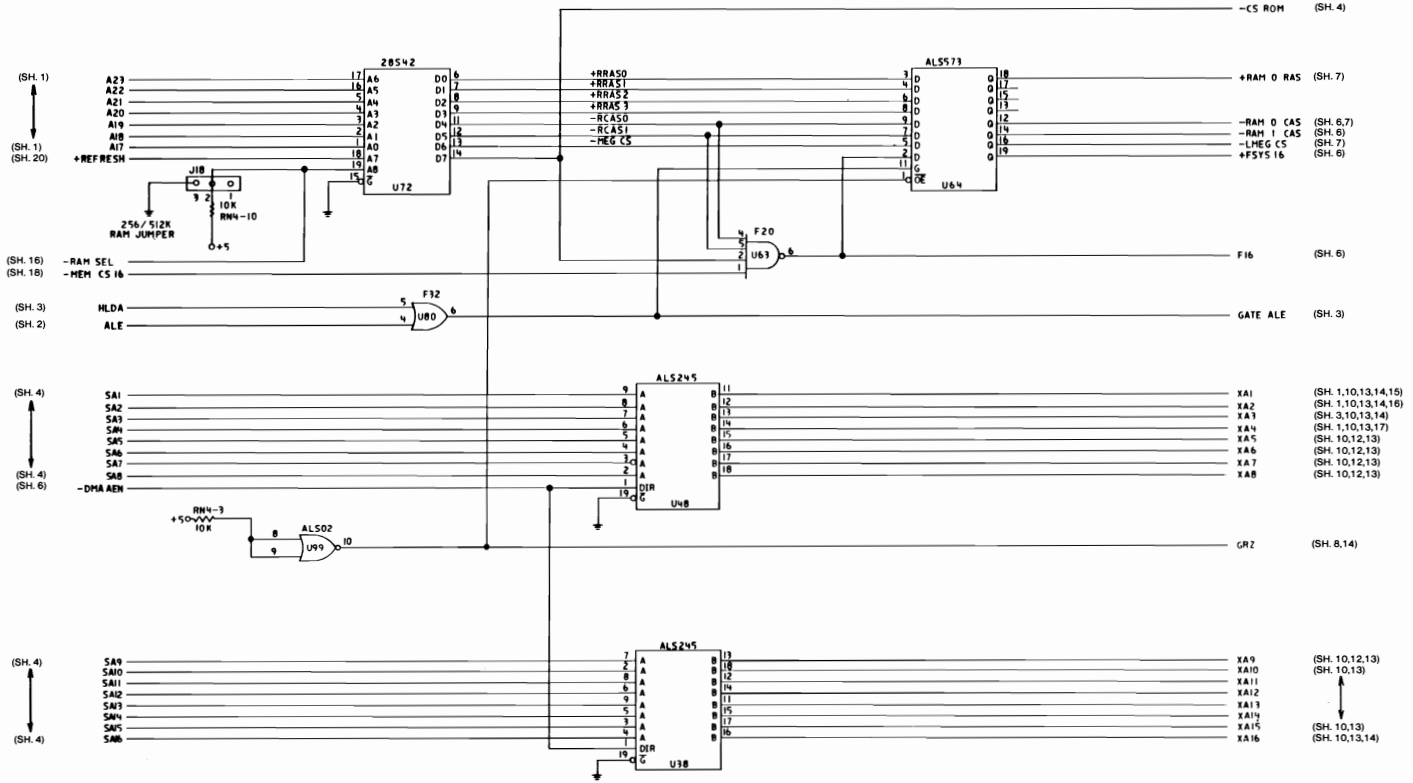
I-100 System Board



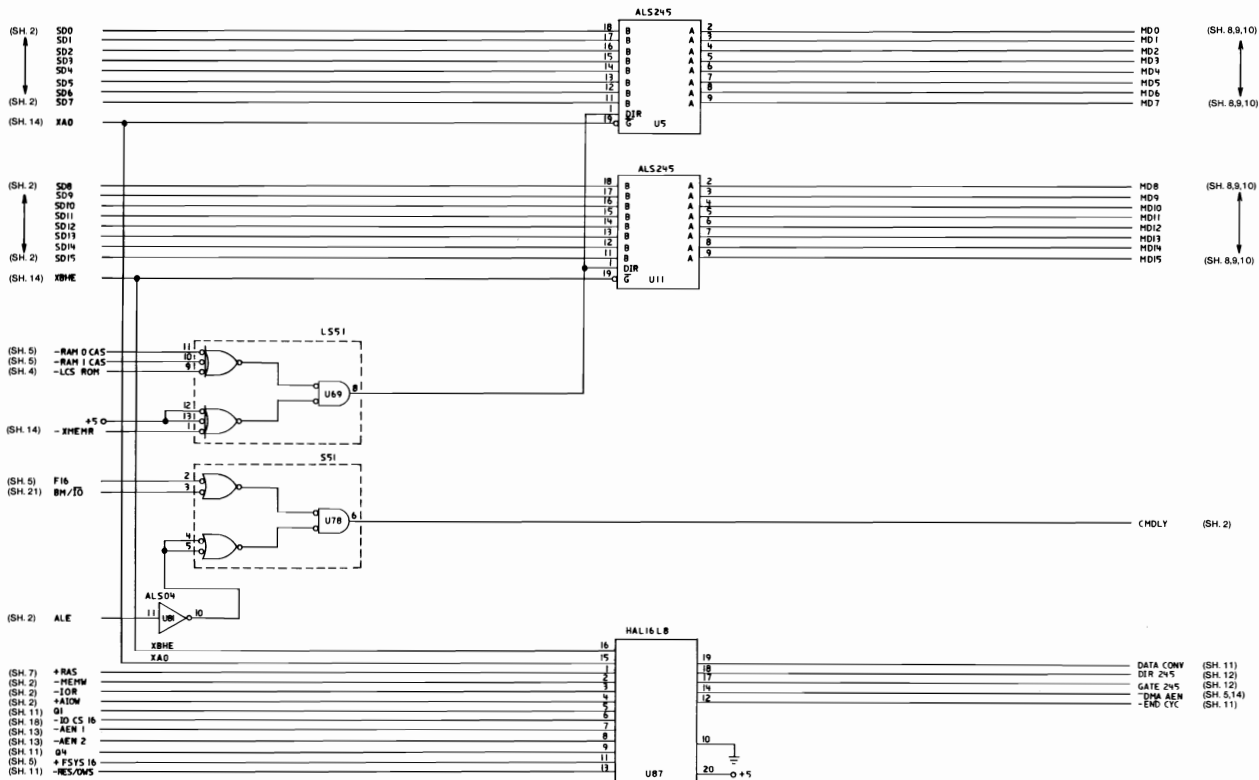
Type 2 512KB Planar (Sheet 3 of 21)



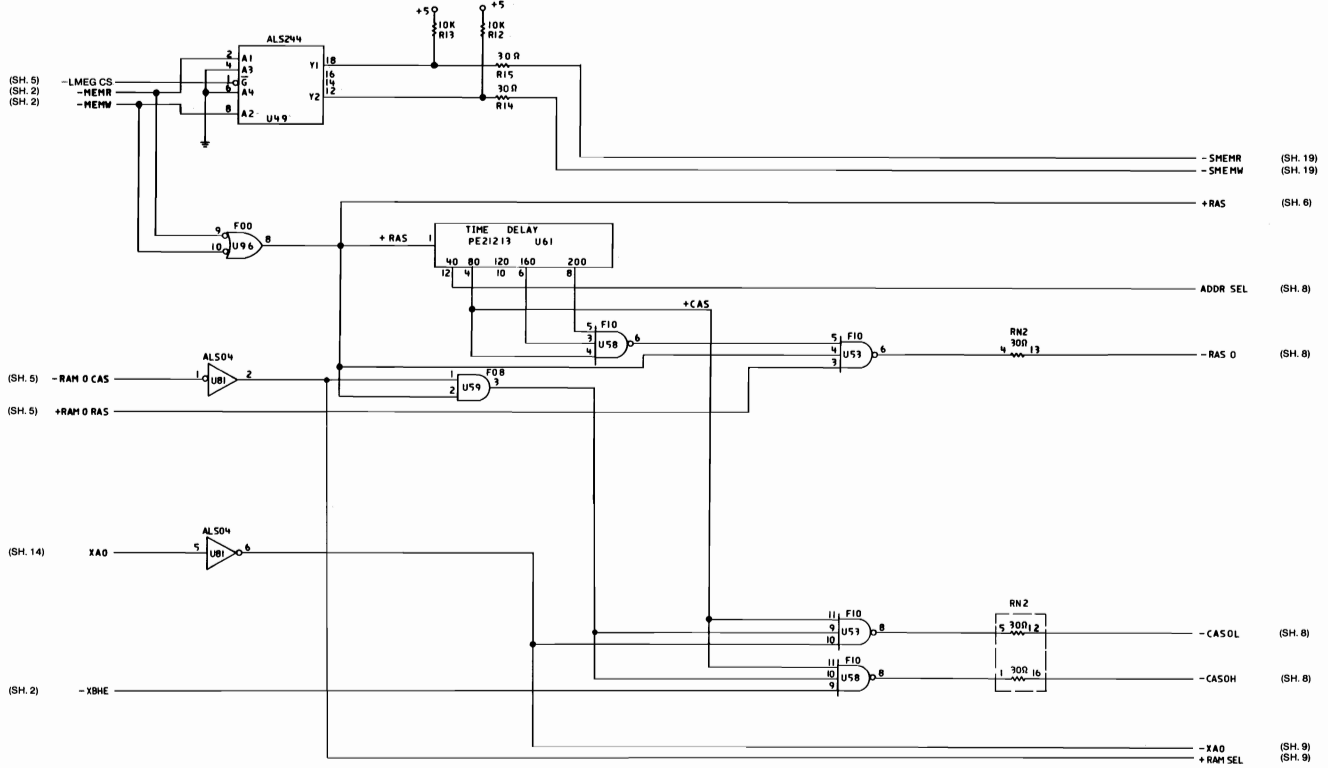
Type 2 512KB Planar (Sheet 4 of 21)



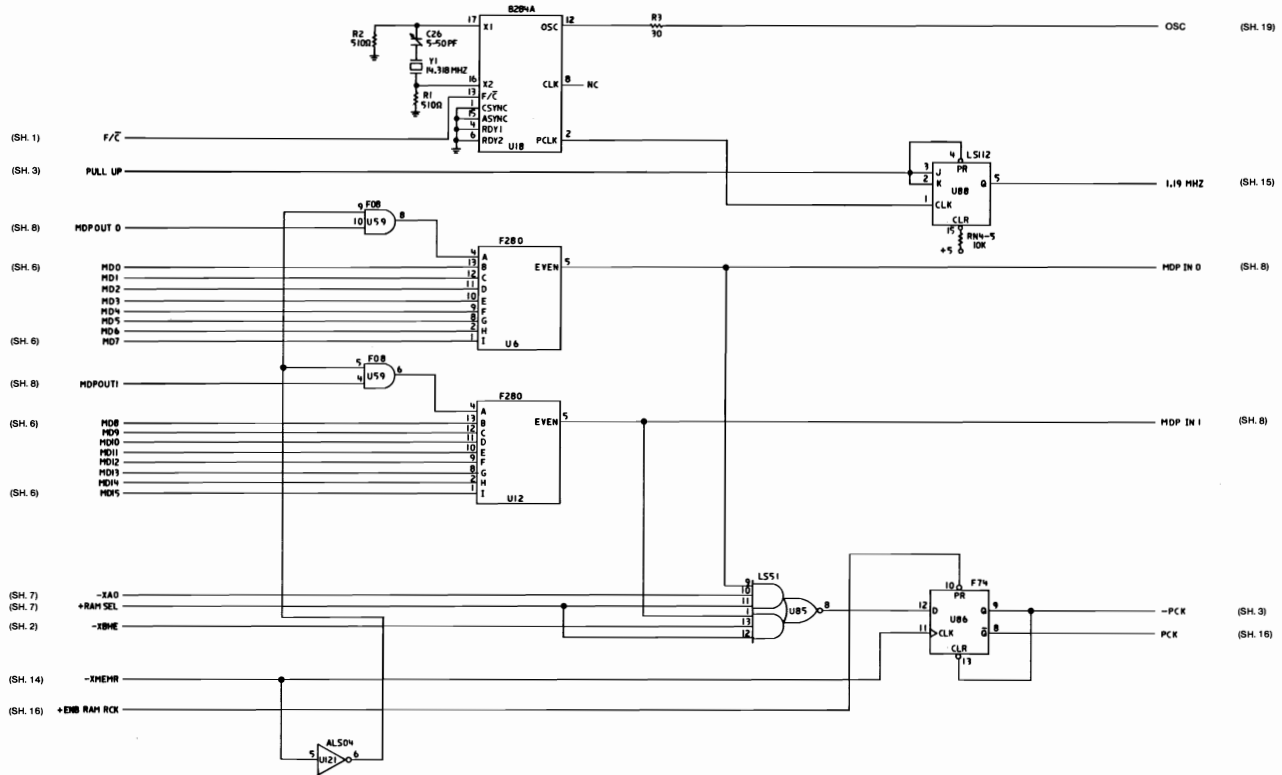
Type 2 512KB Planar (Sheet 5 of 21)



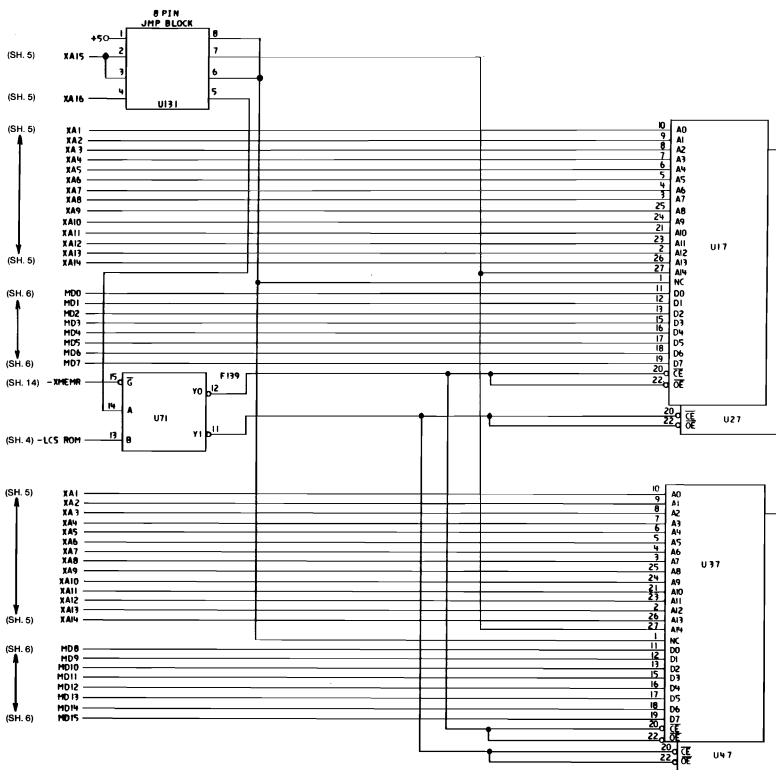
Type 2 512KB Planar (Sheet 6 of 21)



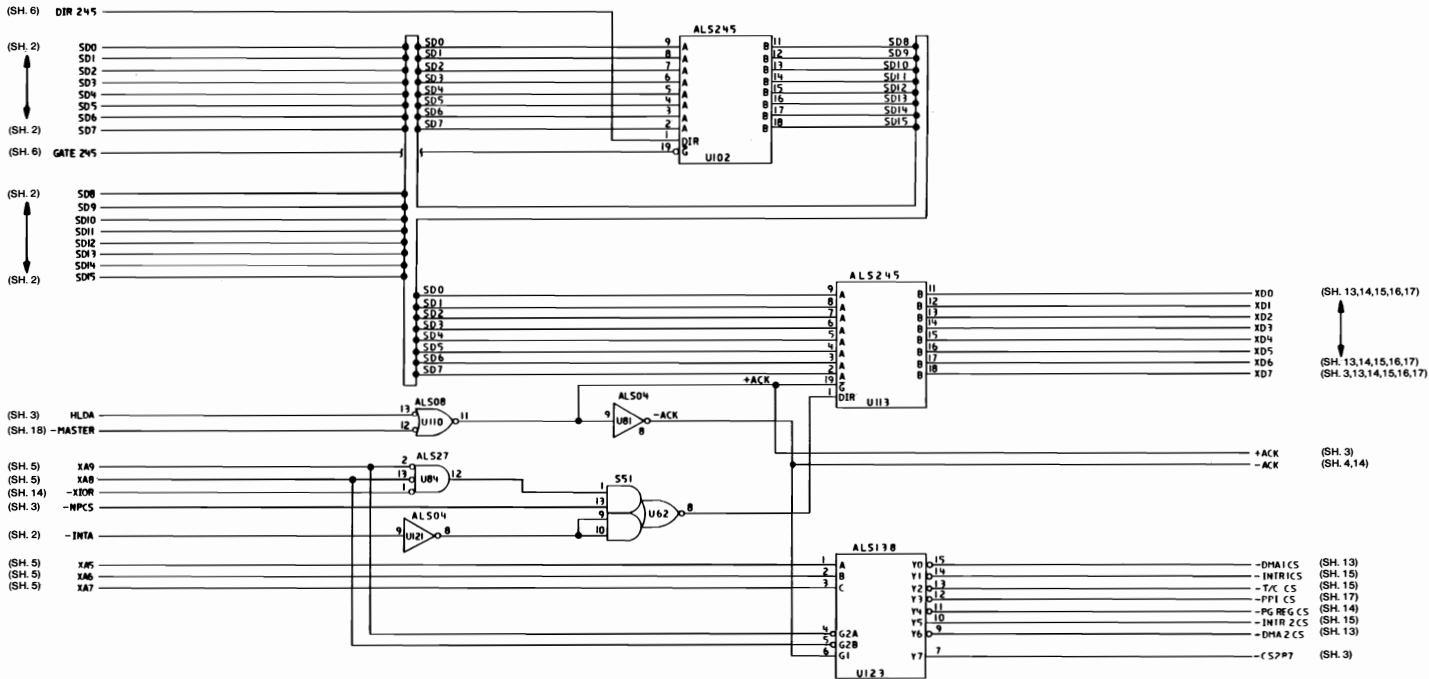
Type 2 512KB Planar (Sheet 7 of 21)



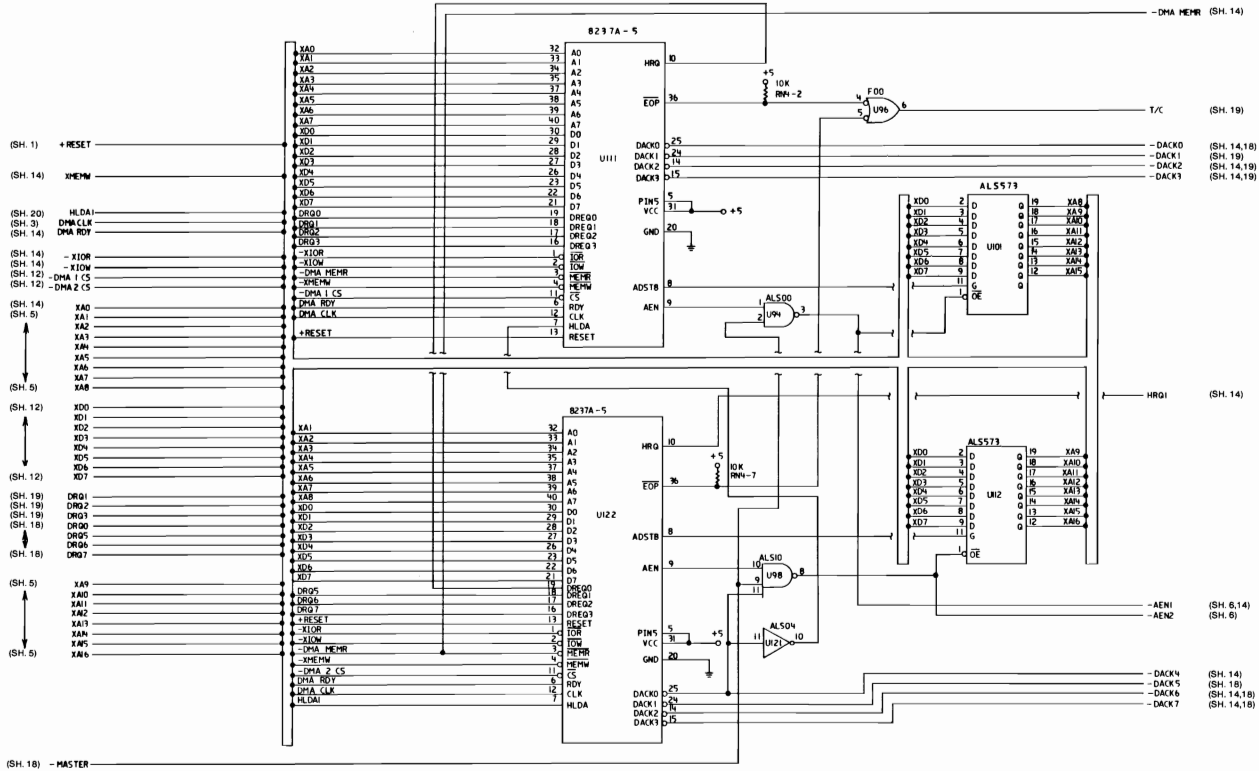
Type 2 512KB Planar (Sheet 9 of 21)

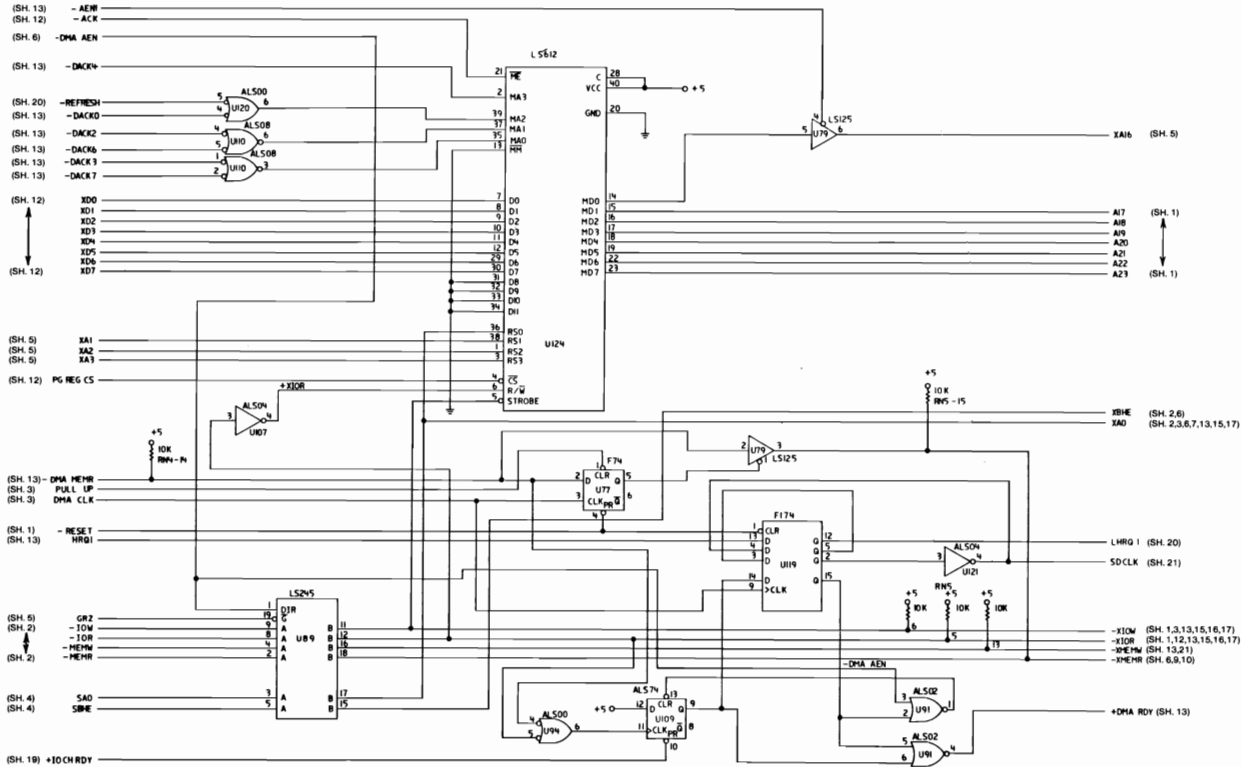


Type 2 512KB Planar (Sheet 10 of 21)

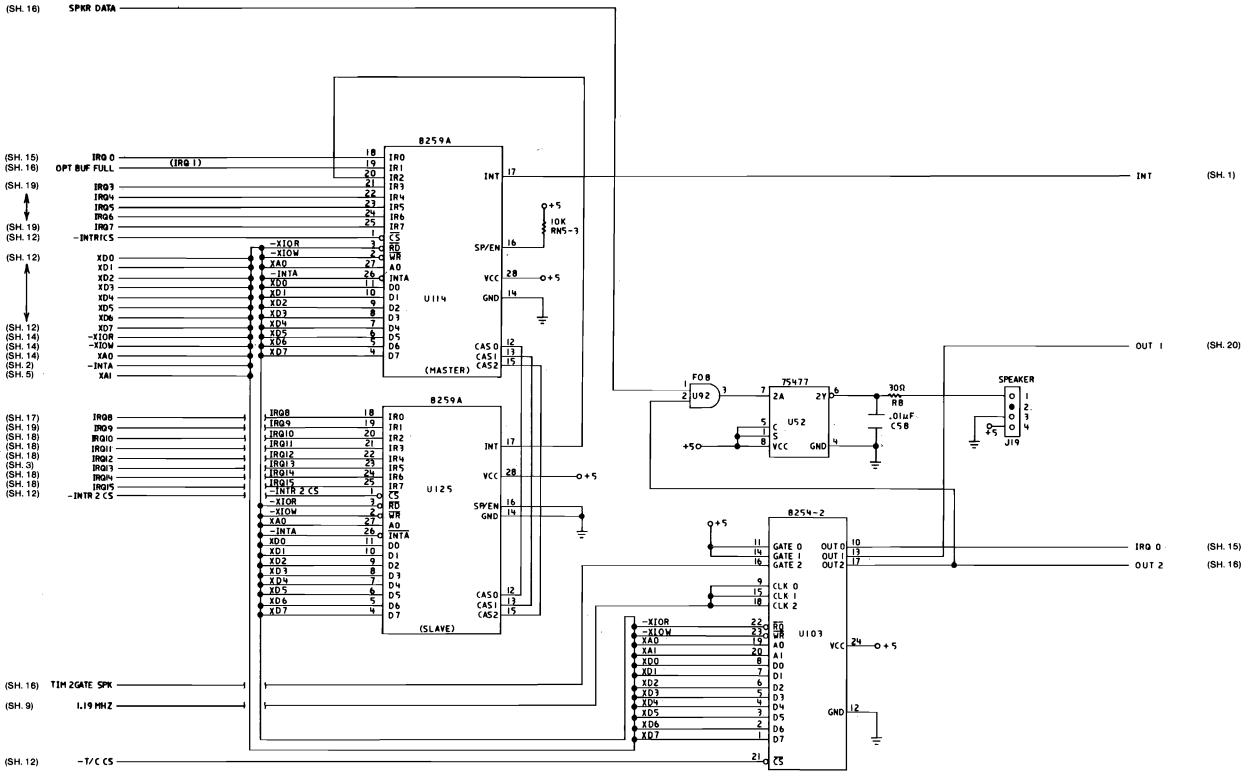


Type 2 512KB Planar (Sheet 12 of 21)

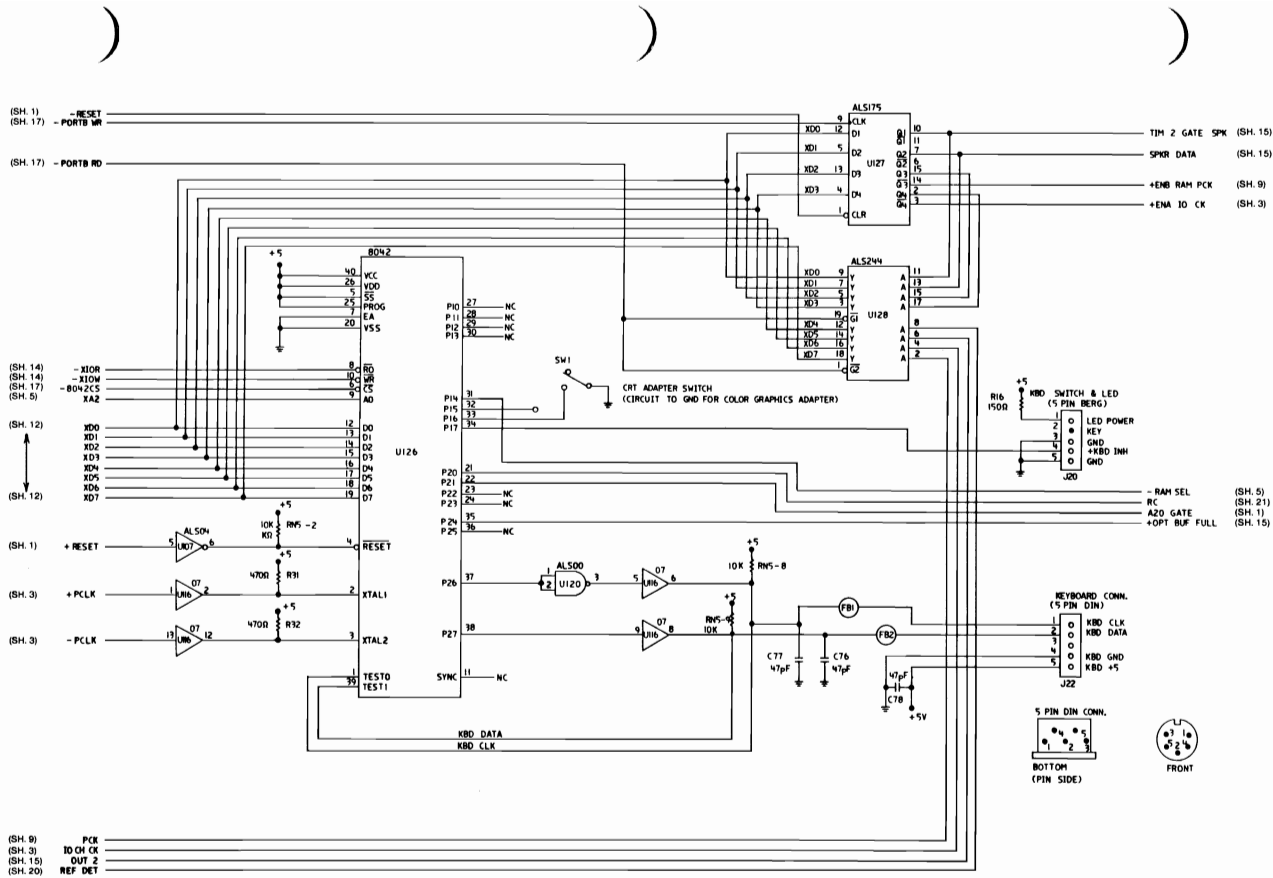




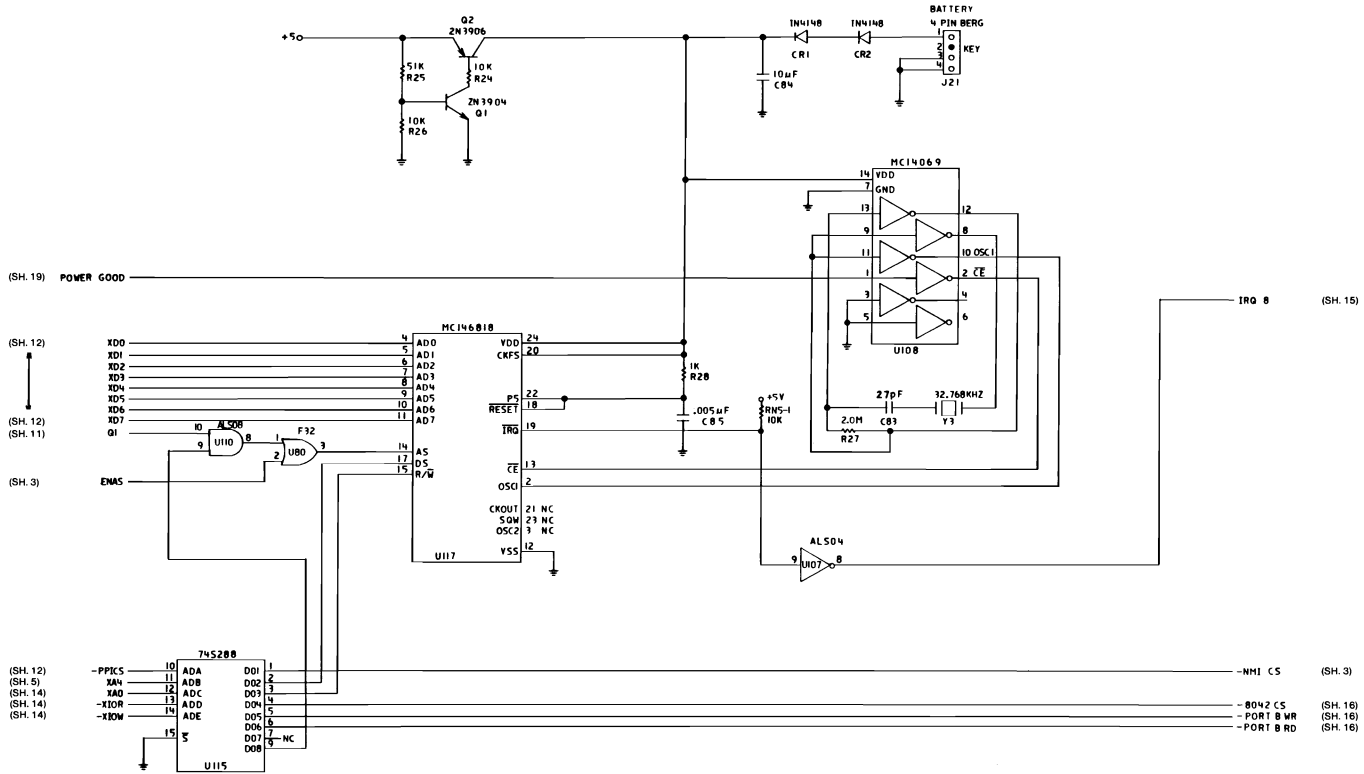
Type 2 512KB Planar (Sheet 14 of 21)



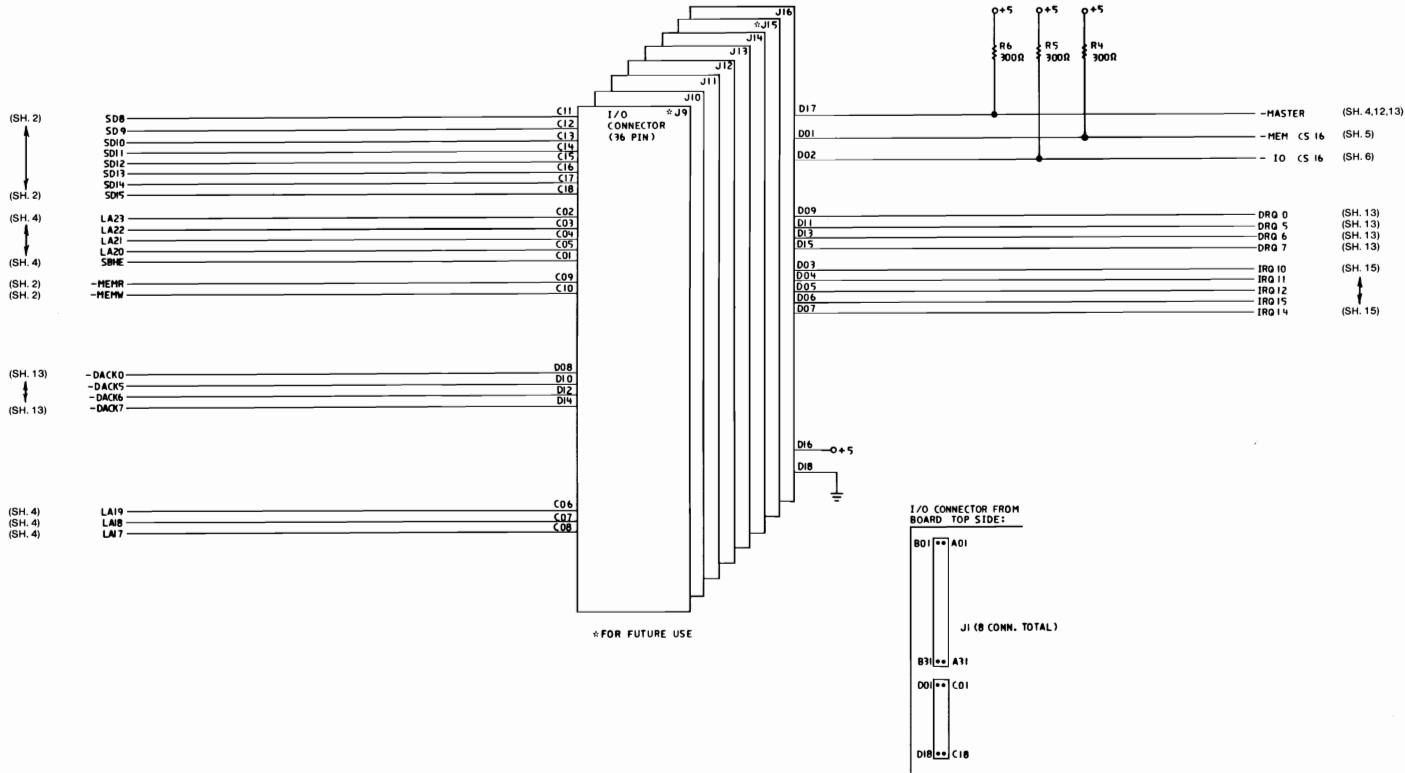
Type 2 512KB Planar (Sheet 15 of 21)



Type 2 512KB Planar (Sheet 16 of 21)

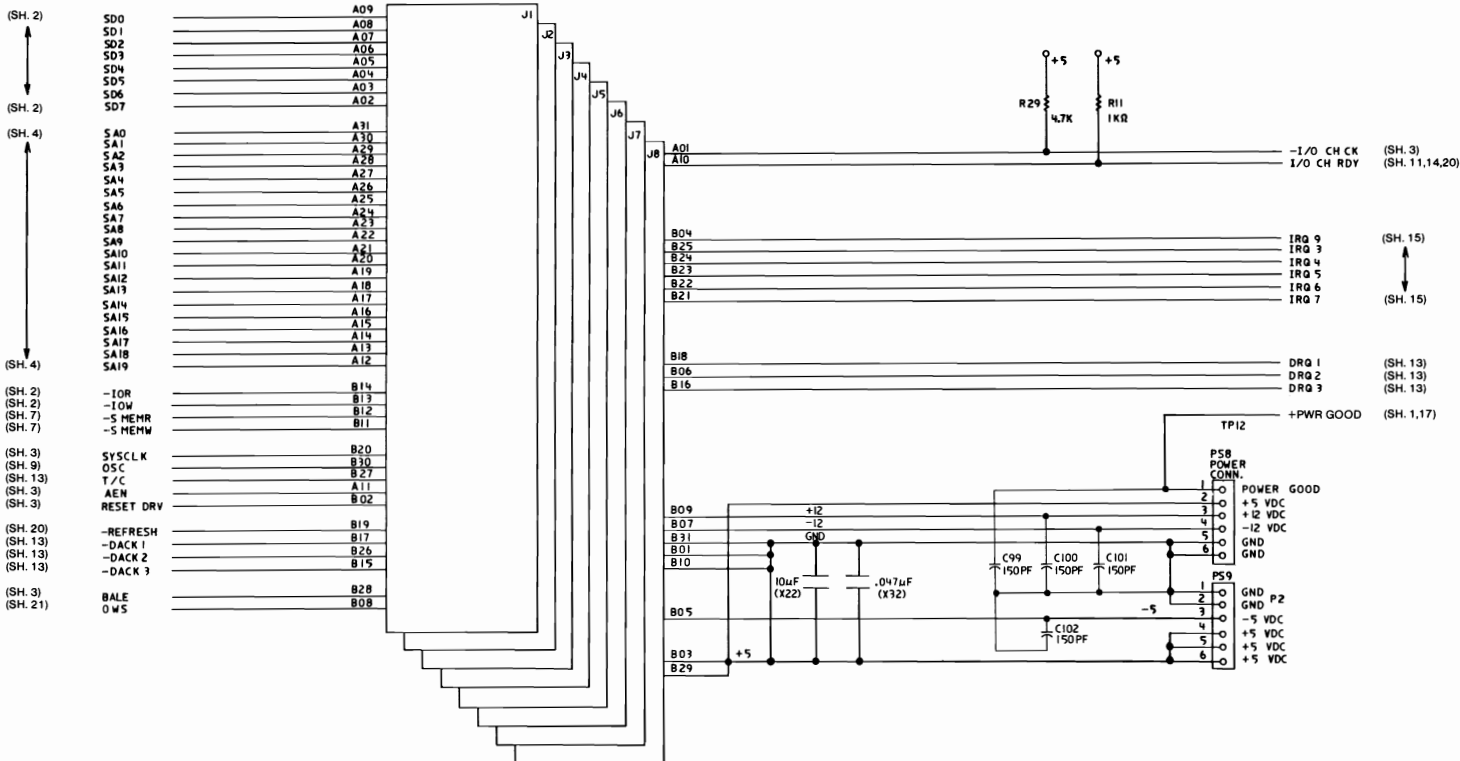


Type 2 512KB Planar (Sheet 17 of 21)

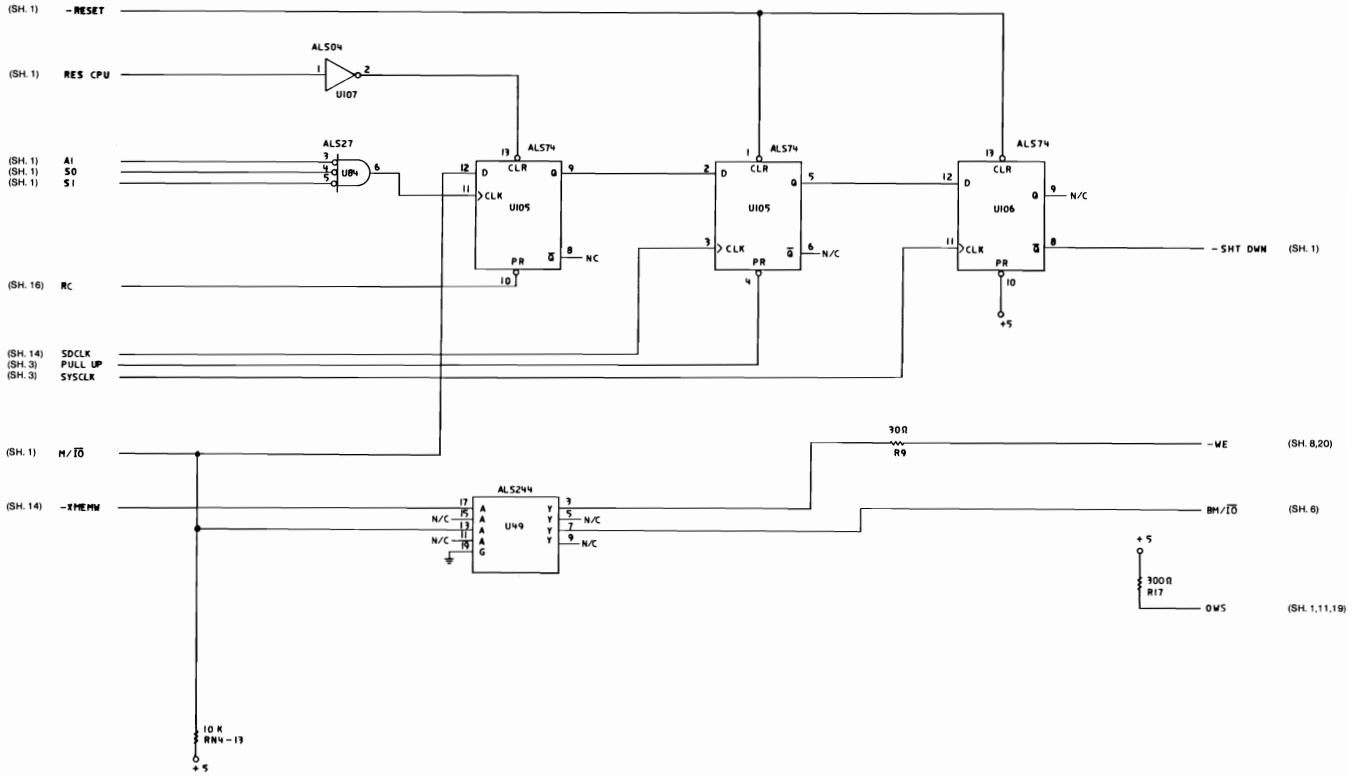


Type 2 512KB Planar (Sheet 18 of 21)

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	$-9.99 \leq x \leq +9.99$ (18 digits)
Short Real *	32	6-7	$8.43 \times 10^{-37} \leq x \leq 3.37 \times 10^{38}$
Long Real *	64	15-16	$4.19 \times 10^{-307} \leq x \leq 1.67 \times 10^{308}$
Temporary Real	80	19	$3.4 \times 10^{-4932} \leq x \leq 1.2 \times 10^{4932}$

Data Types

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

Hardware Interface

The coprocessor uses 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

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
Load Resistor	3-5
Connectors	3-7

Notes:



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

Inputs

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

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

Input Requirements

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

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

Clock and Data Signals	4-12
Keyboard Data Output	4-13
Keyboard Data Input	4-13
Keyboard Layouts	4-15
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
Specifications	4-22
Size	4-22
Weight	4-22
Logic Diagram	4-23

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-overflow 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 $1/\text{period}$. 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.

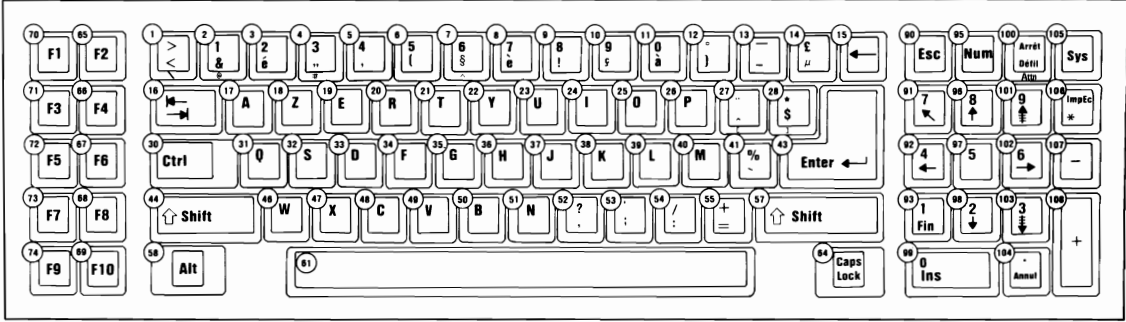
Each system command or data transmission to the keyboard requires a response from the keyboard before the system can send its next output. The keyboard will respond within 20 milliseconds unless the system prevents keyboard output. If the keyboard response is invalid or has a parity error, the system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

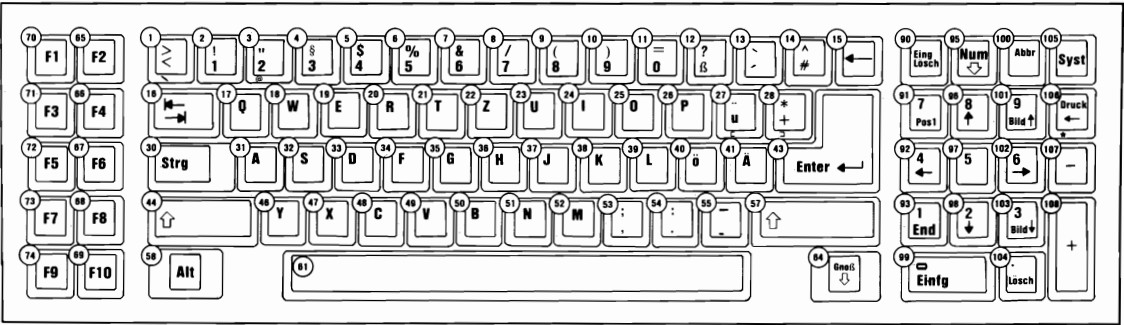
Keyboard Layouts

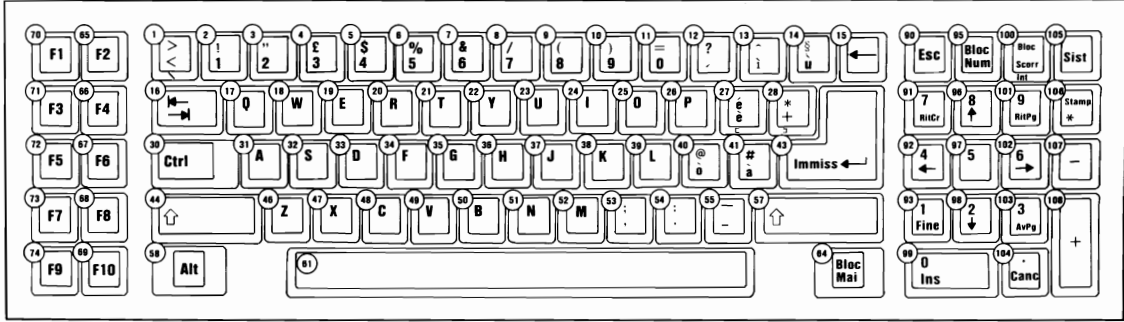
The keyboard has six different layouts:

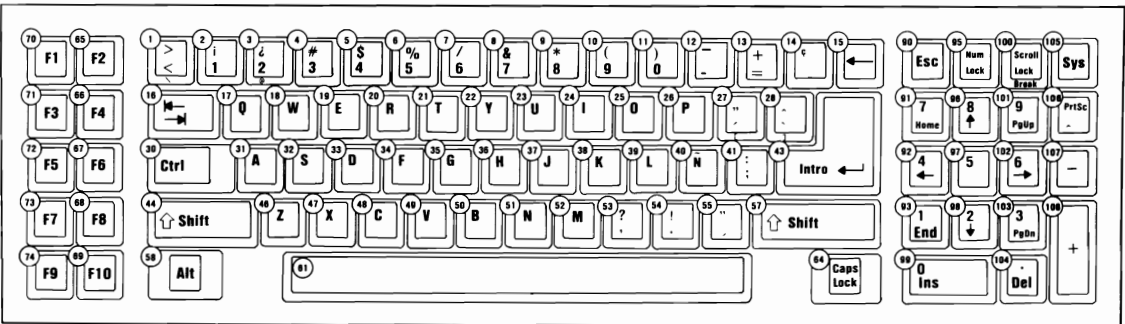
- French
- German
- Italian
- Spanish
- U.K. English
- U.S. English

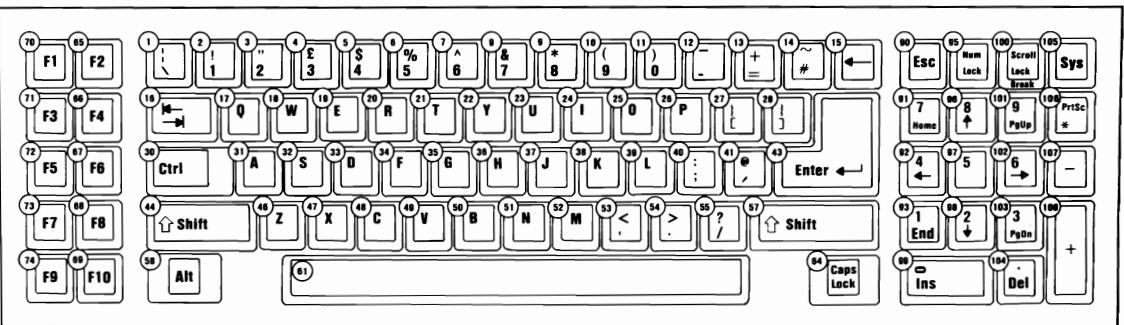
The following pages show the six keyboard layouts.



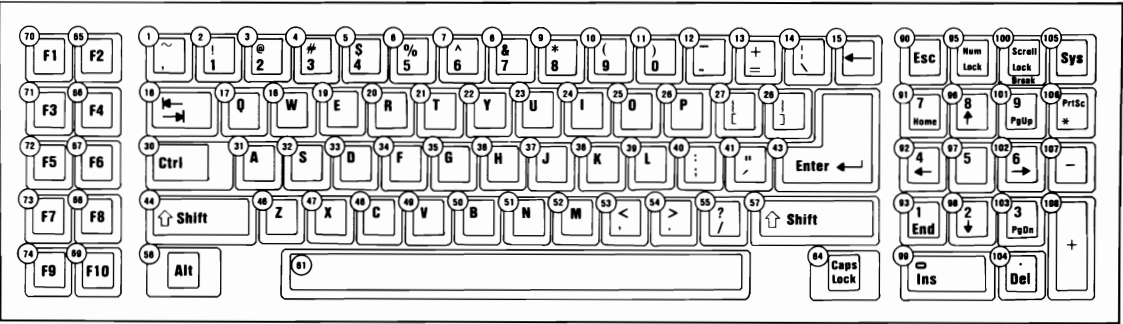








U.S. English Keyboard



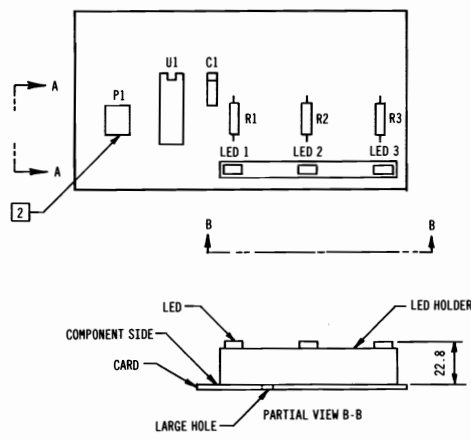
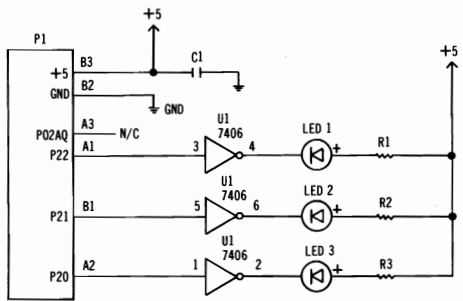
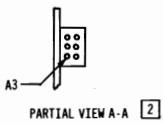
Specifications

Size

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

Weight

- 2.8 kilograms (6.2 pounds)



Enhancement Logic Card Assembly

Notes:



SECTION 5. SYSTEM BIOS

Contents

System BIOS Usage	5-3
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
Keyboard Encoding and Usage	5-13
Character Codes	5-14
Extended Functions	5-18
Shift States	5-19
Special Handling	5-21
Quick Reference	5-24

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_
13	4C-4F	Diskette/Disk	DISKETTE IO
14	50-53	Communications	RS232 IO
15	54-57	Cassette	CASSETTE
16	58-5B	Keyboard	IO/System
17	5C-5F	Printer	Extensions
18	60-63	Resident Basic	KEYBOARD IO
19	64-67	Bootstrap	PRINTER IO
1A	68-6B	Time of Day	F600:0000
1B	6C-6F	Keyboard Break	BOOTSTRAP
1C	70-73	Timer Tick	TIME OF DAY
1D	74-77	Video Initialization	DUMMY_RETURN
1E	78-7B	Diskette Parameters	DUMMY_RETURN
1F	7C-7F	Video Graphics Chars	VIDEO_PARMS
			DISK_BASE
			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 D11)
73	1CC-1CF	IRQ 11 (BIOS entry D11)
74	1D0-1D3	IRQ 12 (BIOS entry D11)
75	1D4-1D7	IRQ 13 BIOS Redirect to NMI interrupt (BIOS entry INT_287)
76	1D8-1DB	IRQ 14 (BIOS entry D11)
77	1DC-1DF	IRQ 15 (BIOS entry D11)
78-7F	1E0-1FF	Not used
80-85	200-217	Reserved for BASIC
86-F0	218-3C3	Used by BASIC interpreter while BASIC is running
F1-FF	3C4-3FF	Not used

Hardware, Basic, and DOS Interrupts

Vectors with Special Meanings

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

- Device open
- Device closed
- Program termination
- Event wait
- Joystick support

- System Request key pressed
- Wait
- Move block
- Extended memory size determination
- Processor to protected mode

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

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

This routine may retain control with the following considerations:

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

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

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	DOS	Reserved for DOS and BASIC
500		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		Single drive mode status byte
510-511		BASIC's segment address store
512-515		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	0.98
	Even and odd	1.04
	Both odd	1.13
Maximum 64K	Both Even	37.0
	Even and odd	55.0
	Both odd	72.0

Time is approximate

Move Block BIOS Timing

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

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

Adapters with System-Accessible ROM Modules

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

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

Byte 0 Hex 55

Byte 1 Hex AA

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

Byte 3 Entry by a CALL FAR

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

When the POST identifies a valid ROM, it does a CALL FAR to byte 3 of the ROM, which should be executable code. The adapter can now perform its power-on initialization tasks. The

adapter's ROM should then return control to the BIOS routines by executing a RETURN FAR.

Additional System Board ROM Modules

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

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

Byte 0 Hex 55

Byte 1 Hex AA

Byte 2 Not used

Byte 3 Entry by a CALL FAR

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

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

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

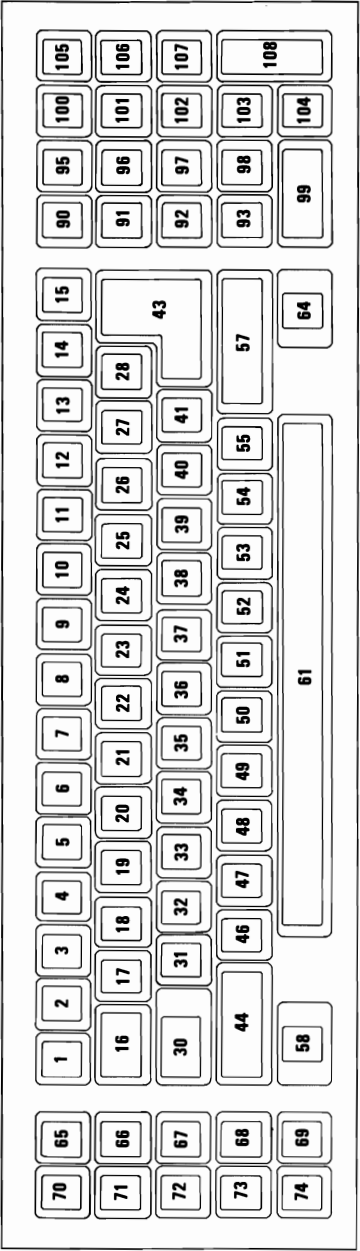
layout. Some operating systems may make provisions for alternate keyboard layouts by providing an interrupt replacer, which resides in the read/write memory. This section discusses only the ROM routine.

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	@	Nul(000) (*)	(*)
4	3	#	-1	(*)
5	4	\$	-1	(*)
6	5	%	-1	(*)
7	6	^	RS(030)	(*)
8	7	&	-1	(*)
9	8	*	-1	(*)
10	9	(-1	(*)
11	0)	-1	(*)
12	-	_	US(031)	(*)
13	=	+	-1	(*)
14	\		FS(028)	-1
15	Backspace (008)	Backspace (008)	Del(127)	-1
16	→ (009)	← (*)	-1	-1
17	q	Q	DC1(017)	(*)
18	w	W	ETB(023)	(*)
19	e	E	ENQ(005)	(*)
20	r	R	DC2(018)	(*)
21	t	T	DC4(020)	(*)
22	y	Y	EM(025)	(*)
23	u	U	NAK(021)	(*)
24	i	I	HT(009)	(*)
25	o	O	SI(015)	(*)
26	p	P	DLE(016)	(*)
27	[{	Esc(027)	(*)
28]	}	GS(029)	-1
30 Ctrl	-1	-1	-1	-1
31	a	A	SOH(001)	(*)
32	s	S	DC3(019)	(*)
33	d	D	EOT(004)	(*)
34	f	F	ACK(006)	(*)
35	g	G	BEL(007)	(*)
36	h	H	BS(008)	(*)
37	j	J	LF(010)	(*)
38	k	K	VT(011)	(*)
39	l	L	FF(012)	(*)
40	;	:	-1	-1
41	,	,"	-1	-1
43	CR	CR	LF(010)	-1
44 Shift (Left)	-1	-1	-1	-1
46	z	Z	SUB(026)	(*)
47	x	X	CAN(024)	(*)
48	c	C	ETX(003)	(*)

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

Character Codes (Part 1 of 2)

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

Test1	5-28
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
Test2	5-58
Test .17 Through Test .23	5-58
Test3. POST Exception Interrupt Tests	5-75
Test4. POST and BIOS Utility Routines	5-81
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
Test5. Exception Interrupt Tests	5-88
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
Test6	5-93
STGTST_CNT	5-93
ROM_ERR	5-95
XMIT_8042	5-95
BOOT_STRAP	5-95
Diskette BIOS	5-97
Fixed Disk (Hard File) BIOS	5-116

Keyboard BIOS	5-129
Printer BIOS	5-138
RS232 BIOS	5-140
Video BIOS	5-143
BIOS	5-161
Memory Size Determine	5-161
Equipment Determine	5-161
NMI	5-162
BIOS1.	5-163
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
BIOS2	5-176
Time of Day	5-176
Alarm Interrupt Handler	5-179
Print Screen	5-180
Timer 1 Interrupt Handler	5-181
ORGS - PC Compatibility and Tables	5-182
POST Error Messages	5-182

Address	Publics by Name	Address	Publics by Value
F000:1E729	A1	F000:0000	POST1
F000:13A23	ACT_DISP_PAGE	F000:0008	K6L
F000:16000	BASTC	F000:0010	Abs M4
F000:19F0	BEEP	F000:0050	START 1
F000:1B1A	BLINK_INT	F000:0396	C8042
F000:12022	BOOT_STRAP_1	F000:03A2	OBP_42
F000:10C96	C21	F000:0C96	POST2
F000:10396	C8042	F000:0C96	C21
F000:14135	CASSETTE_IO_1	F000:1052	SHUT3
F000:1941	CMOS_READ	F000:10B6	SHUT2
F000:195B	CMOS_WRITE	F000:10B9	SHUT7
F000:1A45	CONFTG_BAD	F000:10DA	SHUT6
F000:1E6F5	CONF_TBL	F000:1613	SHUT4
F000:1F6E	CRT_CHAR_GEN	F000:1671	POST3
F000:E020	D1	F000:1941	CMOS_READ
F000:1BCA	D11	F000:1941	POST4
F000:E030	D2	F000:195B	CMOS_WRITE
F000:E040	D2A	F000:1975	DD5
F000:1975	DD5	F000:197D	E_MSG
F000:20E3	DISKETTE_IO_1	F000:19A4	P_MSG
F000:1EFC7	DISK_BASE	F000:19B2	ERR_BEEP
F000:12A17	DISK_INT_1	F000:19F0	BEEP
F000:2C2B	DISK_IO_1	F000:1A36	WAITF
F000:2A82	DISK_SETUP	F000:1A45	CONFIG_BAD
F000:2A2E	DSKETTE_SETUP	F000:1A59	XPC_BYTE
F000:FF53	DUMMY_RETURN	F000:1A69	PRT_HEX
F000:1C18	DUMMY_RETURN_1	F000:1A70	PRT_SEG
F000:E05E	E101	F000:1A85	PROT_PRT_HEX
F000:E077	E102	F000:1AB1	ROM_CHECKSUM
F000:E090	E103	F000:1AB0	ROM_CHECK
F000:E0A9	E104	F000:1AEF	KBD_RESET
F000:E0C2	E105	F000:1B1A	BLINK_INT
F000:E0DB	E106	F000:1B28	SET_TOD
F000:E0F4	E107	F000:1BCA	D11
F000:E10D	E108	F000:1C18	DUMMY_RETURN_1
F000:E126	E109	F000:1C19	RE_DIRECT
F000:E13F	E141	F000:1C22	INT_Z81
F000:E168	E162	F000:1C31	PROC_SHUTDOWN
F000:E191	E163	F000:1C38	POST5
F000:E1B7	E164	F000:1D2A	SYNINI11
F000:E1D6	E201	F000:1EB5	POST6
F000:E1EE	E202	F000:1EB5	STGTST_CNT
F000:E209	E203	F000:1FB5	ROM_ERR
F000:E224	E301	F000:1FE1	XMIT_8042
F000:E239	E302	F000:2022	BOOT_STRAP_1
F000:E2C6	E303	F000:20E3	DISKETTE_IO_1
F000:E2EA	E304	F000:28C1	SEEK
F000:E30E	E401	F000:2A17	DISK_INT_1
F000:E31E	E501	F000:2A2E	DSKETTE_SETUP
F000:E32E	E601	F000:2A82	DISK_SETUP
F000:E343	E602	F000:2C2B	DISK_IO
F000:40A8	EQUIPMENT_1	F000:314F	HD_INT
F000:19B2	ERR_BEEP	F000:3172	KEYBOARD_IO_1
F000:197D	E_MSG	F000:31FE	KB_INT_1
F000:E364	F1780	F000:3267	K16
F000:E379	F1781	F000:326C	SND_DATA
F000:E38E	F1782	F000:3716	PRINTER_IO_1
F000:E3AC	F1790	F000:37A0	RS232_IO_1
F000:E3BF	F1791	F000:38B0	VIDEO_IO_1
F000:E3D2	F3A	F000:38EF	SET_MODE
F000:E25D	F3D	F000:39BF	SET_CTYPE
F000:E3DF	F3D1	F000:39E4	SET_CPOS
F000:E401	FD_TBL	F000:3A0C	READ_CURSOR
F000:4888	FILL	F000:3A23	ACT_DISP_PAGE
F000:FF5E	FLOPPY	F000:3A47	SET_COLOR
F000:4501	GATE_A20	F000:3A6D	VIDEO_STATE
F000:314F	HD_INT	F000:3A90	SCROLL_UP
F000:FF5A	HRD	F000:3B2F	SCROLL_DOWN
F000:1C22	INT_Z81	F000:3B81	READ_AC_CURRENT
F000:E8E1	K10	F000:3B0B	WRITE_AC_CURRENT
F000:E91B	K11	F000:3C0D	WRITE_C_CURRENT
F000:E955	K12	F000:3CB0	READ_DOT
F000:E95F	K13	F000:3CCE	WRITE_DOT
F000:E969	K14	F000:3F72	WRITE_TTY
F000:E976	K15	F000:3FF9	READ_LPEN
F000:3267	K16	F000:409E	MEMORY_SIZE_DET_1
F000:E87E	K6	F000:40A8	EQUIPMENT_1
F000:1008	Abs K6L	F000:40B2	NMI_INT_1
F000:E886	K7	F000:4136	CASSETTE_IO_1
F000:E88E	K8	F000:43BF	SHUT9
F000:E8C8	K9	F000:4501	GATE_A20
F000:1AEF	KBD_RESET	F000:45B0	TIME_OF_DAY_1
F000:31FE	KB_INT_1	F000:473F	RTC_INT
F000:3172	KEYBOARD_IO_1	F000:47A9	PRINT_SCREEN_1
F000:0010	Abs M4	F000:483F	TIMER_INT_1
F000:F0E4	M5	F000:4888	FILL
F000:F0EC	M6	F000:6000	BASIC
F000:F0F4	M7	F000:E020	D1
F000:409E	MEMORY_SIZE_DET_1	F000:E030	D2
F000:E2C3	NMI_INT	F000:E040	D2A
F000:40B2	NMI_INT_1	F000:E05E	E101
F000:03A2	OBP_42	F000:E077	E102
F000:0000	POST1	F000:E090	E103
F000:0C96	POST2	F000:E0A9	E104
F000:1671	POST3	F000:E0C2	E105
F000:1941	POST4	F000:E0DB	E106
F000:1C38	POST5	F000:E0F4	E107
F000:1EB5	POST6	F000:E10D	E108

F000:3716	PRINTER_IO_I	F000:E126	E109
F000:FF54	PRINT_SCREEN	F000:E13F	E161
F000:41A9	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:E1EE	E202
F000:19A4	P_MSG	F000:E209	E203
F000:FFF0	P_O_R	F000:E224	E301
F000:3B81	READ_AC_CURRENT	F000:E239	E302
F000:3A0C	READ_CURSOR	F000:E25D	F3D
F000:3CB0	READ_DOT	F000:E2C3	NMI_INT
F000:3FF9	READ_LPEN	F000:E2C6	E303
F000:1C19	RE_DRECT	F000:E2EA	E304
F000:1AB0	ROM_CHECK	F000:E30E	E401
F000:1AB1	ROM_CHECKSUM	F000:E31E	E501
F000:1FB5	ROM_ERR	F000:E32E	E601
F000:37A0	RSD2_IO_I	F000:E343	E602
F000:417F	RTC_INT	F000:E364	F1780
F000:3B2F	SCROLL_DOWN	F000:E379	F1781
F000:3A90	SCROLL_UP	F000:E38E	F1782
F000:28C1	SEK	F000:E3AC	F1790
F000:FF62	SEEKS_I	F000:E3BF	F1791
F000:3A47	SET_COLOR	F000:E3D2	F3A
F000:39E4	SET_CPOS	F000:E3DF	F3D1
F000:398F	SET_CTYPE	F000:E401	FD_TBL
F000:38EF	SET_MODE	F000:E6F5	CONF_TBL
F000:1B28	SET_TOD	F000:E729	A1
F000:10B6	SHUT2	F000:E87E	K6
F000:1052	SHUT3	F000:E886	K7
F000:1613	SHUT4	F000:E88E	K8
F000:10DA	SHUT5	F000:E8C8	K9
F000:10B9	SHUTT	F000:E8E1	K10
F000:43BF	SHUT9	F000:E91B	K11
F000:FF23	SLAVE_VECTOR_TABLE	F000:E955	K12
F000:366C	SND_DATA	F000:E95F	K13
F000:0050	START_I	F000:E969	K14
F000:1EB5	STGTST_CNT	F000:E976	K15
F000:1D2A	SYSINIT1	F000:EF7C	DISK_BASE
F000:483F	TIMER_INT_I	F000:FOA4	VIDEO_PARMS
F000:45B0	TIME_OF_DAY_I	F000:FOE4	M5
F000:FF66	TUTOR	F000:FOEC	M6
F000:FEF3	VECTOR_TABLE	F000:F0F4	M7
F000:3880	VIDEO_IO_I	F000:FA6E	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	HRD
F000:3CCE	WRITE_DOT	F000:FF5E	FLOPPY
F000:3F72	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 ABSOLUTE ADDRESSES WITHIN THE CODE SEGMENTS 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

26	TEST1.ASM	-->	POST AND MANUFACTURING TEST ROUTINES		
27	DSEG.INC	-->	DATA SEGMENTS LOCATIONS		
28	POSTQU.INC	-->	COMMON EQUATES FOR POST AND BIOS		
29	SYSDATA.INC	-->	POWER ON SELF TEST EQUATES FOR PROTECTED MODE		
30			POST TEST.01 THROUGH TEST.16		
31	TEST2.ASM	-->	POST TEST AND INITIALIZATION ROUTINES		
32			POST TEST.17 THROUGH TEST.22		
33	TEST3.ASM	-->	POST EXCEPTION INTERRUPT TESTS		
34	TEST4.ASM	-->	POST AND BIOS UTILITY ROUTINES		
35			CMOS_READ - READ CMOS LOCATION ROUTINE		
36			CMOS_WRITE - WRITE CMOS LOCATION ROUTINE		
37			DDS - LOAD (DS:) WITH DATA SEGMENT		
38			E_MSG - POST ERROR MESSAGE HANDLER		
39			MFG_HALT - MANUFACTURING ERROR TRAP		
40			P_MSG - POST STRING DISPLAY ROUTINE		
41			ERR_BEEP - POST ERROR BEEP PROCEDURE		
42			BEEP - SPEAKER BEEP CONTROL ROUTINE		
43			WAITF - FIXED TIME WAIT ROUTINE		
44			CONFIG_BAD - SET BAD CONFIG IN CMOS_DIAG		
45			XPC_BYTE - DISPLAY HEX BYTE AS 00 - FF		
46			PRT_HEX - DISPLAY CHARACTER		
47			PRT_SEG - DISPLAY SEGMENT FORMAT ADDRESS		
48			PROT_PRT_HEX - POST PROTECTED MODE DISPLAY		
49			ROM_CHECKSUM - CHECK ROM MODULES FOR CHECKSUM		
50			ROM_SCAN - ROM SCAN AND INITIALIZE		
51			KBD_RESET - POST KEYBOARD RESET ROUTINE		
52			BLINK_INT - MANUFACTURING TOGGLE BIT ROUTINE		
53			SET_T0D - SET TIMER FROM CMOS RTC		
54			DUMI - DUMMY INTERRUPT HANDLER ->INT 7?H		
55			RE_DIRECT - HARDWARE INT 9 REDIRECT (L 2)		
56			INT_287 - HARDWARE INT 13 REDIRECT (287)		
57			PROC_SHUTDOWN - 80286 RESET ROUTINE		
58	TEST5.ASM	-->	EXCEPTION INTERRUPT TEST HANDLERS FOR POST TESTS		
59			SYSNITI - BUILD PROTECTED MODE POINTERS		
60			GDT_BLD - BUILD THE GDT FOR POST		
61			SIDT_BLD - BUILD THE IDT FOR POST		
62	TEST6.ASM	-->	POST TESTS AND SYSTEM BOOT STRAP		
63			STGST_CNT - SEGMENT STORAGE TEST		
64			ROM_ERR - ROM ERROR DISPLAY ROUTINE		
65			XHIT_8042 - KEYBOARD DIAGNOSTIC OUTPUT		
66			BOOT_STRAP - BOOT STRAP LOADER -INT 19H		
67					
68	DSKETTE.ASM	-->	DISKETTE BIOS		
69			DISKETTE_IO_I - INT 13H BIOS ENTRY (40H) -INT 13H		
70			DISK_INT_I - HARDWARE INTERRUPT HANDLER -INT 0EH		
71			DSKETTE_SETUP - POST SETUP DRIVE TYPES		
72	DISK.ASM	-->	FIXED DISK BIOS		
73			DISK_SETUP - SETUP DISK VECTORS AND TEST		
74			DISK_IO - INT 13H BIOS ENTRY -INT 13H		
75			HD_INT - HARDWARE INTERRUPT HANDLER -INT 76H		
76	KYBD.ASM	-->	KEYBOARD BIOS		
77			KEYBOARD_IO_I - INT 16H BIOS ENTRY -INT 16H		
78			KB_INT_I - HARDWARE INTERRUPT -INT 09H		
79			SND_DATA - KEYBOARD TRANSMISSION		
80	PRT.ASM	-->	PRINTER ADAPTER BIOS		
81	RS232.ASM	-->	COMMUNICATIONS BIOS FOR RS232		
82	VIDEO1.ASM	-->	VIDEO BIOS		
83	BIOS.ASM	-->	BIOS ROUTINES		
84			MEMORY_SIZE_DET_I - REAL MODE SIZE -INT 12H		
85			EQUIPMENT_I - EQUIPMENT DETERMINATION -INT 11H		
86			NMI_INT_I - NMI HANDLER -INT 02H		
87	BIOS1.ASM	-->	INTERRUPT 15H BIOS ROUTINES		
88			DEV_OPEN - NULL DEVICE OPEN HANDLER		
89			DEV_CLOSE - NULL DEVICE CLOSE HANDLER		
90			PROG_TERM - NULL PROGRAM TERMINATION		
91			EVENT_WAIT - RTC EVENT WAIT/TIMEOUT ROUTINE		
92			JOY_STICK - JOYSTICK PORT HANDLER		
93			SYS_REQ - NULL SYSTEM REQUEST KEY		
94			WAIT - RTC TIMED WAIT ROUTINE		
95			BLOCKMOVE - EXTENDED MEMORY MOVE INTERFACE		
96			GATE_A20 - ADDRESS BIT 20 CONTROL		
97			EXT_MEMORY - EXTENDED MEMORY SIZE DETERMINE		
98			SET_VMODE - SWITCH PROCESSOR TO VIRTUAL MODE		
99			DEVICE_BUSY - NULL DEVICE BUSY HANDLER		
100			INT_COMPLETE - NULL INTERRUPT COMPLETE HANDLER		
101	BIOS2.ASM	-->	BIOS INTERRUPT ROUTINES		
102			TIME_OF_DAY_I - TIME OF DAY ROUTINES -INT 1AH		
103			RTC_INT_HANDLER - IRQ LEVEL 8 ALARM HANDLER -INT 70H		
104			PRINT_SCREEN_I - PRINT SCREEN ROUTINE -INT 05H		
105			TIMER_INT_I - TIMER1 INTERRUPT HANDLER ->INT 1CH		
106	ORGS.ASM	-->	COMPATIBILITY MODULE		
107			POST_ERROR_MESSAGES - POST ERROR MESSAGES		
108			DISKETTE - DISK - VIDEO DATA TABLES		

 .LIST


```

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

```

```

175 C PAGE
176 C |-----|
177 C | ROM BIOS DATA AREAS |
178 C |-----|
179 C
180 0000 C DATA SEGMENT AT 40H | ADDRESS= 0040:0000
181 C
182 0000 7777 C *RS232_BASE DW ? | BASE ADDRESSES OF RS232 ADAPTERS
183 0002 7777 C DW ? | SECOND LOGICAL RS232 ADAPTER
184 0004 7777 C DW ? | RESERVED
185 0006 7777 C *PRINTER_BASE DW ? | BASE ADDRESSES OF PRINTER ADAPTERS
186 0008 7777 C DW ? | SECOND LOGICAL PRINTER ADAPTER
187 000A 7777 C DW ? | THIRD LOGICAL PRINTER ADAPTER
188 000C 7777 C DW ? | RESERVED
189 000E 7777 C *EQUIP_FLAG DW ? | INSTALLED HARDWARE FLAGS
190 0010 7777 C *MFG_TST DB ? | INITIALIZATION FLAGS
191 0012 ?? C *MEMORY_SIZE DW ? | BASE MEMORY SIZE IN K BYTES (X 1024)
192 0013 7777 C *MFG_ERR_FLAG DB ? | SCRATCHPAD FOR MANUFACTURING
193 0015 ?? C *MFG_ERR_FLAG DB ? | ERROR CODES
194 0016 ?? C
195 C
196 C |-----|
197 C | KEYBOARD DATA AREAS |
198 C |-----|
199 C
200 0017 ?? C *KB_FLAG DB ? | KEYBOARD SHIFT STATE AND STATUS FLAGS
201 0018 ?? C *KB_FLAG DB ? | SECOND BYTE OF KEYBOARD STATUS
202 0019 ?? C *ALT_INPDT DB ? | STORAGE FOR ALTERNATE KEY PAD ENTRY
203 001A 7777 C *BUFFER_HEAD DW ? | POINTER TO HEAD OF KEYBOARD BUFFER
204 001C 7777 C *BUFFER_TAIL DW ? | POINTER TO TAIL OF KEYBOARD BUFFER
205 C
206 C |-----|
207 C | HEAD = TAIL INDICATES THAT THE BUFFER IS EMPTY |
208 001E 10 [ ??? ] C *KB_BUFFER DW 16 DUP(?) | ROOM FOR 15 SCAN CODE ENTRIES
209 C
210 C
211 C |-----|
212 C | DISKETTE DATA AREAS |
213 C |-----|
214 C
215 C
216 003E ?? C *SEEK_STATUS DB ? | DRIVE RECALIBRATION STATUS
217 C | BIT 3-0 = DRIVE 3-0 RECALIBRATION
218 C | BEFORE NEXT SEEK IF BIT IS = 0
219 003F ?? C *MOTOR_STATUS DB ? | MOTOR STATUS
220 C | BIT 3-0 = DRIVE 3-0 CURRENTLY RUNNING
221 C | BIT 7 = CURRENT OPERATION IS A WRITE
222 0040 ?? C *MOTOR_COUNT DB ? | TIME OUT COUNTER FOR MOTOR(S) TURN OFF
223 0041 ?? C *DSKETTE_STATUS DB ? | RETURN CODE STATUS BYTE
224 C | CMD BLOCK IN STACK FOR DISK OPERATION
225 0042 07 [ ?? ] C *NEC_STATUS DB 7 DUP(?) | STATUS BYTES FROM DISKETTE OPERATION
226 C
227 C
228 C
229 C
230 C |-----|
231 C | VIDEO DISPLAY DATA AREA |
232 C |-----|
233 C
234 0049 ?? C *CRT_MODE DB ? | CURRENT DISPLAY MODE (TYPE)
235 004A 7777 C *CRT_COLS DW ? | NUMBER OF COLUMNS ON SCREEN
236 004C 7777 C *CRT_LEN DW ? | LENGTH OF REGEN BUFFER IN BYTES
237 004E 7777 C *CRT_START DW ? | STARTING ADDRESS IN REGEN BUFFER
238 0050 08 [ ??? ] C *CURSOR_POSN DW 8 DUP(?) | CURSOR FOR EACH OF UP TO 8 PAGES
239 C
240 C
241 C
242 0060 7777 C *CURSOR_MODE DW ? | CURRENT CURSOR MODE SETTING
243 0062 ?? C *ACTIVE_PAGE DB ? | CURRENT PAGE BEING DISPLAYED
244 0063 7777 C *ADDR_6845 DW ? | BASE ADDRESS FOR ACTIVE DISPLAY CARD
245 0065 ?? C *CRT_MODE_SET DB ? | CURRENT SETTING OF THE 3x8 REGISTER
246 0066 ?? C *CRT_PALETTE DB ? | CURRENT PALETTE SETTING - COLOR CARD
247 C
248 C |-----|
249 C | POST AND BIOS WORK DATA AREA |
250 C |-----|
251 C
252 C | STACK SAVE, etc. |
253 0067 7777 C *IO_ROM_INIT DW ? | POINTER TO ROM INITIALIZATION ROUTINE
254 0069 7777 C *IO_ROM_SEG DW ? | POINTER TO I/O ROM SEGMENT
255 006B ?? C *INTR_FLAG DB ? | FLAG INDICATING AN INTERRUPT HAPPENED
256 C
257 C |-----|
258 C | TIMER DATA AREA |
259 C |-----|
260 C
261 006C 7777 C *TIMER_LOW DW ? | LOW WORD OF TIMER COUNT
262 006E 7777 C *TIMER_HIGH DW ? | HIGH WORD OF TIMER COUNT
263 0070 ?? C *TIMER_OFL DB ? | TIMER HAS ROLLED OVER SINCE LAST READ
264 C
265 C |-----|
266 C | SYSTEM DATA AREA |
267 C |-----|
268 C
269 0071 ?? C *BIOS_BREAK DB ? | BIT 7=1 IF BREAK KEY HAS BEEN PRESSED
270 0072 7777 C *RESET_FLAG DW ? | WORD=1234H IF KEYBOARD RESET UNDERWAY
271 C
272 C |-----|
273 C | FIXED DISK DATA AREAS |
274 C |-----|
275 C
276 0074 ?? C *DISK_STATUS1 DB ? | FIXED DISK STATUS
277 0075 ?? C *HF_NUM DB ? | COUNT OF FIXED DISK DRIVES
278 0076 ?? C *CONTROL_BYTE DB ? | HEAD CONTROL BYTE
279 0077 ?? C *PORT_OFF DB ? | RESERVED (PORT OFFSET)

```

```

280 C PAGE
281 C |-----|
282 C | TIME-OUT VARIABLES |
283 C |-----|
284 C
285 C *PRINT_TIM_OUT DB ? | TIME OUT COUNTERS FOR PRINTER RESPONSE
286 C 0078 ?? DB ? | SECOND LOGICAL PRINTER ADAPTER
287 C 007A ?? DB ? | THIRD LOGICAL PRINTER ADAPTER
288 C 007B ?? DB ? | RESERVED
289 C *RS232_TIM_OUT DB ? | TIME OUT COUNTERS FOR RS232 RESPONSE
290 C 007D ?? DB ? | SECOND LOGICAL RS232 ADAPTER
291 C 007E ?? DB ? | RESERVED
292 C 007F ?? DB ? | RESERVED
293 C
294 C |-----|
295 C | ADDITIONAL KEYBOARD DATA AREA |
296 C |-----|
297 C
298 C | BUFFER LOCATION WITHIN SEGMENT 40H
299 C 0080 ???? DW ? | OFFSET OF KEYBOARD BUFFER START
300 C 0082 ???? DW ? | OFFSET OF END OF BUFFER
301 C
302 C |-----|
303 C | EGA/PGA DISPLAY WORK AREA |
304 C |-----|
305 C
306 C *ROWS DB ? | ROWS ON THE ACTIVE SCREEN (LESS 1)
307 C 0085 ???? DW ? | BYTES PER CHARACTER
308 C 0087 ?? DB ? | MODE OPTIONS
309 C 0088 ?? DB ? | FEATURE BIT SWITCHES
310 C 0089 ?? DB ? | RESERVED FOR DISPLAY ADAPTERS
311 C 008A ?? DB ? | RESERVED FOR DISPLAY ADAPTERS
312 C
313 C |-----|
314 C | ADDITIONAL MEDIA DATA |
315 C |-----|
316 C
317 C *LAstrate DB ? | LAST DISKETTE DATA RATE SELECTED
318 C 008C ?? DB ? | STATUS REGISTER
319 C 008D ?? DB ? | ERROR REGISTER
320 C *HF_INT_FLAG DB ? | FIXED DISK INTERRUPT FLAG
321 C 008F ?? DB ? | COMBO FIXED DISK/DISKETTE CARD BIT 0=1
322 C 0090 ?? DB ? | DRIVE 0 MEDIA STATE
323 C 0091 ?? DB ? | DRIVE 1 MEDIA STATE
324 C 0092 ?? DB ? | DRIVE 0 OPERATION START STATE
325 C 0093 ?? DB ? | DRIVE 1 OPERATION START STATE
326 C 0094 ?? DB ? | DRIVE 0 PRESENT CYLINDER
327 C 0095 ?? DB ? | DRIVE 1 PRESENT CYLINDER
328 C
329 C |-----|
330 C | ADDITIONAL KEYBOARD FLAGS |
331 C |-----|
332 C
333 C *KB_FLAG_3 DB ? | KEYBOARD MODE STATE AND TYPE FLAGS
334 C 0097 ?? DB ? | KEYBOARD LED FLAGS
335 C
336 C |-----|
337 C | REAL TIME CLOCK DATA AREA |
338 C |-----|
339 C
340 C *USER_FLAG DW ? | OFFSET ADDRESS OF USERS WAIT FLAG
341 C 009A ???? DW ? | SEGMENT ADDRESS OF USER WAIT FLAG
342 C *RTC_LOW DW ? | LOW WORD OF USER WAIT FLAG
343 C 009E ???? DW ? | HIGH WORD OF USER WAIT FLAG
344 C 00A0 ?? DB ? | WAIT ACTIVE FLAG (01=BUSY, 00=POSTED)
345 C | (00=POST ACKNOWLEDGED) |
346 C
347 C |-----|
348 C | AREA FOR NETWORK ADAPTER |
349 C |-----|
350 C 00A1 07 [ ?? ] *NET DB ? DUP(?) | RESERVED FOR NETWORK ADAPTERS
351 C
352 C
353 C |-----|
354 C | EGA/PGA PALETTE POINTER |
355 C |-----|
356 C
357 C *SAVE_PTR DD ? | POINTER TO EGA PARAMETER CONTROL BLOCK
358 C 00A8 ???????? | RESERVED
359 C
360 C |-----|
361 C | DATA AREA - PRINT SCREEN |
362 C |-----|
363 C
364 C *STATUS_BYTE DB ? | ADDRESS= 0040:0100 (REF 0050:0000)
365 C 0100 ORG 100H | PRINT SCREEN STATUS BYTE
366 C 0100 ?? DB ? | 00=READY/OK, 01=BUSY, FF=ERROR
367 C
368 C
369 C DATA ENDS | END OF BIOS DATA SEGMENT
370 C
371 C .LIST
372 C

```

```

373 PAGE
374 C INCLUDE POSTEQU.INC
375 C -----
376 C | EQUATES USED BY POST AND BIOS |
377 C |-----|
378
379 = 00FC MODEL_BYTE EQU 0FCH ; SYSTEM MODEL BYTE
380 = 0000 SUB_MODEL_BYTE EQU 000H ; SYSTEM SUB-MODEL TYPE
381 = 0001 BIOS_LEVEL EQU 001H ; BIOS REVISION LEVEL
382 = FBAT RATE_UPPER EQU 0FBATH ; 0F952H +10X
383 = F9FD RATE_LOWER EQU 0F9FDH ; 0F952H -10X
384
385 C |-----| 8042 KEYBOARD INTERFACE AND DIAGNOSTIC CONTROL REGISTERS |-----|
386 PORT_A EQU 060H ; 8042 KEYBOARD SCAN CODE/CONTROL PORT
387 PORT_B EQU 061H ; PORT B READ/WRITE DIAGNOSTIC REGISTER
388 RAM_PAR_ON EQU 11110011B ; AND MASK FOR PARITY CHECKING ENABLE ON
389 RAM_PAR_OFF EQU 00001100B ; OR MASK FOR PARITY CHECKING ENABLE OFF
390 PARITY_ERR EQU 11000000B ; R/W MEMORY - 1/0 CHANNEL PARITY ERROR
391 = 0001 GATE2 EQU 00000010B ; TIMER 2 INPUT GATE CLOCK BIT
392 = 0002 SPK2 EQU 00000100B ; SPEAKER OUTPUT DATA ENABLE BIT
393 = 0010 REFRESH_BIT EQU 00100000B ; REFRESH TEST BIT
394 = 0020 OUT2 EQU 01000000B ; SPEAKER TIMER OUT2 INPUT BIT
395 = 0040 IO_CHECK EQU 01000000B ; I/O (MEMORY) CHECK OCCURRED BIT MASK
396 = 0080 PARITY_CHECK EQU 10000000B ; MEMORY PARITY CHECK OCCURRED BIT MASK
397 = 0064 STATUS_PORT EQU 064H ; 8042 STATUS PORT
398 = 0001 OUT_BUF_FULL EQU 00000010B ; 0 = +OUTPUT BUFFER FULL
399 = 0002 INPT_BUF_FULL EQU 00000100B ; 1 = +INPUT BUFFER FULL
400 = 0004 SYS_FLAG EQU 00000100B ; 2 = -SYSTEM FLAG -POST/-SELF TEST
401 = 0008 CMD_DATA EQU 00010000B ; 3 = -COMMAND/-DATA
402 = 0010 KYBD_INH EQU 00100000B ; 4 = +KEYBOARD INHIBITED
403 = 0020 TRANS_TMOUT EQU 01000000B ; 5 = +TRANSMIT TIMEOUT
404 = 0040 RCV_TMOUT EQU 10000000B ; 6 = +RECEIVE TIME OUT
405 = 0080 PARTTY_EVN EQU 01000000B ; 7 = +PARITY IS EVEN
406
407 C |-----| 8042 INPUT PORT BIT DEFINITION SAVED IN 06FG.TST |-----|
408 BASE_MEMB EQU 00010000B ; BASE PLANAR R/W MEMORY EXTENSION 640/X
409 = 0010 BASE_MEM EQU 00100000B ; BASE PLANAR R/W MEMORY SIZE 256/512
410 = 0020 MFG_LOOP EQU 01000000B ; LOOP POST JUMPER BIT FOR MANUFACTURING
411 = 0040 DSP_JMP EQU 01000000B ; DISPLAY TYPE SWITCH JUMPER BIT
412 = 0080 KEY_BD_INHIB EQU 10000000B ; KEYBOARD INHIBIT SWITCH BIT
413
414 C |-----| 8042 COMMANDS |-----|
415 = 0060 WRITE_8042_LOC EQU 060H ; WRITE 8042 COMMAND BYTE
416 = 00AA SELF_TEST EQU 0AAH ; 8042 SELF TEST
417 = 00AB INTR_FACE_CK EQU 0ABH ; CHECK 8042 INTERFACE COMMAND
418 = 00AD DIS_KBD EQU 0ADH ; DISABLE KEYBOARD COMMAND
419 = 00AE ENA_KBD EQU 0AEH ; ENABLE KEYBOARD COMMAND
420 = 00C0 READ_8042_INPUT EQU 0C0H ; READ 8042 INPUT PORT
421 = 00DD DISABLE_BIT20 EQU 0DDH ; DISABLE ADDRESS LINE BIT 20
422 = 00DF ENABLE_BIT20 EQU 0DFH ; ENABLE ADDRESS LINE BIT 20
423 = 00E0 KYBD_CLK_DATA EQU 0E0H ; GET KEYBOARD CLOCK AND DATA COMMAND
424 = 00FE SHUT_CMD_FULL EQU 0FEH ; CAUSE A SHUTDOWN COMMAND
425 = 0001 KYBD_CLK EQU 001H ; KEYBOARD CLOCK BIT 0
426
427 C |-----| KEYBOARD/LED COMMANDS |-----|
428 LED_CMD EQU 0EDH ; LED WRITE COMMAND
429 = 00F2 KB_READ_ID EQU 0F2H ; READ KEYBOARD ID COMMAND
430 = 00F4 KB_ENABLE EQU 0F4H ; KEYBOARD ENABLE
431
432 C |-----| 8042 KEYBOARD RESPONSE |-----|
433 = 00AA KB_OK EQU 0AAH ; RESPONSE FROM SELF DIAGNOSTIC
434 = 00FA KB_ACK EQU 0FAH ; ACKNOWLEDGE FROM TRANSMISSION
435 = 00FF KB_RESEND EQU 0FFH ; RESEND REQUEST
436 = 00FF KB_OVER_RUN EQU 0FFH ; OVER RUN SCAN CODE
437
438 C |-----| FLAG EQUATES WITHIN *KB_FLAG |-----|
439 = 0001 RIGHT_SHIFT EQU 00000010B ; RIGHT SHIFT KEY DEPRESSED
440 = 0002 LEFT_SHIFT EQU 00000100B ; LEFT SHIFT KEY DEPRESSED
441 = 0004 CTL_SHIFT EQU 00001000B ; CONTROL SHIFT KEY DEPRESSED
442 = 0008 ALT_SHIFT EQU 00001000B ; ALTERNATE SHIFT KEY DEPRESSED
443 = 0010 SCROLL_STATE EQU 01000000B ; SCROLL LOCK STATE HAS BEEN TOGGLED
444 = 0020 NUM_STATE EQU 01000000B ; NUM LOCK STATE HAS BEEN TOGGLED
445 = 0040 CAPS_STATE EQU 01000000B ; CAPS LOCK STATE HAS BEEN TOGGLED
446 = 0080 INS_STATE EQU 10000000B ; INSERT STATE IS ACTIVE
447
448 C |-----| FLAG EQUATES WITHIN *KB_FLAG_1 |-----|
449 = 0004 SYS_SHIFT EQU 00000100B ; SYSTEM KEY DEPRESSED AND HELD
450 = 0008 HOLD_STATE EQU 00001000B ; SUSPEND KEY HAS BEEN TOGGLED
451 = 0010 SCROLL_SHIFT EQU 00100000B ; SCROLL LOCK KEY IS DEPRESSED
452 = 0020 NUM_SHIFT EQU 01000000B ; NUM LOCK KEY IS DEPRESSED
453 = 0040 CAPS_SHIFT EQU 01000000B ; CAPS LOCK KEY IS DEPRESSED
454 = 0080 INS_SHIFT EQU 10000000B ; INSERT KEY IS DEPRESSED
455
456 C |-----| FLAG EQUATES WITHIN *KB_FLAG_2 |-----|
457 = 0007 KB_LEDS EQU 00000111B ; KEYBOARD LED STATE BITS
458 = 0000 KB_FA EQU 00010000B ; RESERVED (MUST BE ZERO)
459 = 0010 KB_FE EQU 00100000B ; ACKNOWLEDGMENT RECEIVED
460 = 0020 KB_FR_LED EQU 01000000B ; RESEND RECEIVED FLAG
461 = 0040 KB_ERR EQU 01000000B ; MODE INDICATOR UPDATE
462 = 0080 KB_TRR EQU 10000000B ; KEYBOARD TRANSMIT ERROR FLAG
463
464 C |-----| FLAG EQUATES WITHIN *KB_FLAG_3 |-----|
465 = 0001 KBX EQU 00000010B ; KBX INSTALLED
466 = 0002 LC_HC EQU 00000100B ; LAST SCAN CODED WAS A HIDDEN CODE
467 = 0004 GRAPH_ON EQU 00001000B ; ALL GRAPHICS KEY DOWN (W.T. ONLY)
468 = 0001 SET_NUM_LK EQU 00010000B ; RESERVED (MUST BE ZERO)
469 = 0020 LC_AB EQU 01000000B ; FORCE NUM LOCK IF READ ID AND KBX
470 = 0040 RD_ID EQU 01000000B ; LAST CHARACTER WAS FIRST ID CHARACTER
471 = 0080 RD_ID EQU 10000000B ; DOING A READ ID (MUST BE BIT0)
472
473 C |-----| KEYBOARD SCAN CODES |-----|
474 = 00AB ID_1 EQU 0ABH ; 1ST ID CHARACTER FOR KBX
475 = 00A1 ID_2 EQU 0A1H ; 2ND ID CHARACTER FOR KBX
476 = 0038 ALT_KEY EQU 56 ; SCAN CODE FOR ALTERNATE SHIFT KEY
477 = 001D CTL_KEY EQU 29 ; SCAN CODE FOR CONTROL KEY
478 = 003A CAPS_KEY EQU 58 ; SCAN CODE FOR SHIFT LOCK KEY
479 = 0053 DEL_KEY EQU 83 ; SCAN CODE FOR DELETE KEY
480 = 0052 INS_KEY EQU 82 ; SCAN CODE FOR INSERT KEY
481 = 002A LEFT_KEY EQU 42 ; SCAN CODE FOR LEFT SHIFT
482 = 0045 NUM_KEY EQU 69 ; SCAN CODE FOR NUMBER LOCK KEY
483 = 0036 RIGHT_KEY EQU 54 ; SCAN CODE FOR RIGHT SHIFT
484 = 0046 SCROLL_KEY EQU 70 ; SCAN CODE FOR SCROLL LOCK KEY
485 = 0054 SYS_KEY EQU 84 ; SCAN CODE FOR SYSTEM KEY
    
```

```

486 C PAGE
487 C |-----|
488 C | CMOS EQUATES FOR THIS SYSTEM |
489 C |-----|
490 = 0070 C CMOS_PORT EQU 070H ; I/O ADDRESS OF CMOS ADDRESS PORT
491 = 0071 C CMOS_DATA EQU 071H ; I/O ADDRESS OF CMOS DATA PORT
492 = 0080 C NMI EQU 1000000B ; DISABLE NMI INTERRUPT MASK
493 C ; HIGH BIT OF CMOS LOCATION ADDRESS
494 C |-----|
495 C |-----| CMOS TABLE LOCATION ADDRESS'S #
496 = 0000 C CMOS_SECONDS EQU 000H ; SECONDS
497 = 0001 C CMOS_SEC_ALARM EQU 001H ; SECONDS ALARM ## NOTE: ALL LOCATIONS
498 = 0002 C CMOS_MINUTES EQU 002H ; MINUTES ; IN THE CMOS AREA
499 = 0003 C CMOS_MIN_ALARM EQU 003H ; MINUTES ALARM ARE IBM USE ONLY
500 = 0004 C CMOS_HOURS EQU 004H ; HOURS AND SUBJECT TO
501 = 0005 C CMOS_HR_ALARM EQU 005H ; HOURS ALARM CHANGE, ONLY THE
502 = 0006 C CMOS_DAY_WEEK EQU 006H ; DAY OF THE WEEK POST & BIOS CODE
503 = 0007 C CMOS_DAY_MONTH EQU 007H ; DAY OF THE MONTH SHOULD DIRECTLY
504 = 0008 C CMOS_MONTH EQU 008H ; MONTH ACCESS LOCATIONS
505 = 0009 C CMOS_YEAR EQU 009H ; YEAR (TWO DIGITS) IN CMOS STORAGE.
506 = 000A C CMOS_REG_A EQU 00AH ; STATUS REGISTER A
507 = 000B C CMOS_REG_B EQU 00BH ; STATUS REGISTER B ALARM
508 = 000C C CMOS_REG_C EQU 00CH ; STATUS REGISTER C FLAGS
509 = 000D C CMOS_REG_D EQU 00DH ; STATUS REGISTER D BATTERY
510 = 000E C CMOS_DIAG EQU 00EH ; POST DIAGNOSTIC STATUS RESULTS BYTE
511 = 000F C CMOS_SHUT_DOWN EQU 00FH ; SHUTDOWN STATUS COMMAND BYTE
512 = 0010 C CMOS_DISKETTE EQU 010H ; DISKETTE DRIVE TYPE BYTE
513 C ; - RESERVED ; C
514 = 0012 C CMOS_DISK EQU 012H ; FIXED DISK TYPE BYTE ; 1C
515 C ; - RESERVED ; 1E
516 = 0014 C CMOS_EQUIP EQU 014H ; EQUIPMENT WORD LOW BYTE ; 1H
517 = 0015 C CMOS_B_M_S_LO EQU 015H ; BASE MEMORY SIZE - LOW BYTE (X1024) ; 1K
518 = 0016 C CMOS_B_M_S_HI EQU 016H ; BASE MEMORY SIZE - HIGH BYTE ; 1S
519 = 0017 C CMOS_E_M_S_LO EQU 017H ; EXPANSION MEMORY SIZE - LOW BYTE ; 1U
520 = 0018 C CMOS_E_M_S_HI EQU 018H ; EXPANSION MEMORY SIZE - HIGH BYTE ; 1M
521 = 0019 C CMOS_DTSK_1 EQU 019H ; FIXED DISK TYPE - DRIVE C EXTENSION ; 1E
522 = 001A C CMOS_DTSK_2 EQU 01AH ; FIXED DISK TYPE - DRIVE D EXTENSION ; 1D
523 C ; IBM THROUGH ZDH - RESERVED ;
524 = 002E C CMOS_CKSUM_H1 EQU 02EH ; CMOS CHECKSUM - HIGH BYTE ; *
525 = 002F C CMOS_CKSUM_L1 EQU 02FH ; CMOS CHECKSUM - LOW BYTE ; *
526 = 0030 C CMOS_U_M_S_LO EQU 030H ; USABLE MEMORY ABOVE 1 MEG - LOW BYTE ;
527 = 0031 C CMOS_U_M_S_HI EQU 031H ; USABLE MEMORY ABOVE 1 MEG - HIGH BYTE ;
528 = 0032 C CMOS_CENTURY EQU 032H ; DATE CENTURY BYTE (BCD) ;
529 = 0033 C CMOS_INFO128 EQU 033H ; 128KB INFORMATION STATUS FLAG BYTE ;
530 C ; - 34H THROUGH 3FH - RESERVED ;
531 C |-----|
532 C |-----| CMOS DIAGNOSTIC STATUS ERROR FLAGS WITHIN CMOS DIAG
533 C CMOS_CLK_FAIL EQU 00000100B ; CMOS CLOCK NOT UPDATING OR NOT VALID
534 = 0008 C HF_FAIL EQU 00000100B ; FIXED DISK FAILURE ON INITIALIZATION
535 = 0010 C W_MEM_SIZE EQU 00010000B ; MEMORY SIZE NOT EQUAL TO CONFIGURATION
536 = 0020 C BAD_CONFIG EQU 00100000B ; MINIMUM CONFIG USED INSTEAD OF CMOS
537 = 0040 C BAD_CKSUM EQU 01000000B ; CHECKSUM ERROR
538 = 0080 C BAD_BAT EQU 10000000B ; DEAD BATTERY - CMOS LOST POWER
539 C |-----|
540 C |-----| CMOS INFORMATION FLAGS
541 = 0080 C M40K EQU 10000000B ; 512K -> 640K OPTION INSTALLED (128K)
542 C ; EQU 01000000B ; FLAG USED BY CMOS SETUP UTILITY
543 C |-----|
544 C |-----| DISKETTE EQUATES
545 C |-----|
546 = 0001 C DUAL EQU 00000001B ; MASK FOR COMBO/DSP ADAPTER
547 = 0080 C INT_FLAG EQU 10000000B ; INTERRUPT OCCURRENCE FLAG
548 = 0080 C DSK_CHG EQU 10000000B ; DISKETTE CHANGE FLAG MASK BIT
549 = 0000 C DETERMIND EQU 00000000B ; DISK STATE DETERMINED IN STATE BITS
550 = 0010 C HOME EQU 00010000B ; TRACK 0 MASK
551 = 0004 C SENSE_DRY_ST EQU 00000100B ; SENSE DRIVE STATUS COMMAND
552 = 0030 C TRK_SCAP EQU 030H ; CRASH STOP (48 TPI DRIVES)
553 = 000A C QUIET_SEEK EQU 00AH ; SEEK TO TRACK 10
554 = 0002 C MAX_DRV EQU 2 ; MAX NUMBER OF DRIVES
555 = 000F C HD12_SETTLE EQU 15 ; 1.2 M HEAD SETTLE TIME
556 = 0014 C HD320_SETTLE EQU 20 ; 320 K HEAD SETTLE TIME
557 = 0025 C MOTOR_WAIT EQU 37 ; 2 SECONDS OF COUNTS FOR MOTOR TURN OFF
558 C |-----|
559 C |-----| DISKETTE ERRORS
560 = 0080 C TIME_OUT EQU 080H ; ATTACHMENT FAILED TO RESPOND
561 = 0040 C BAD_SEEK EQU 040H ; SEEK OPERATION FAILED
562 = 0020 C BAD_NEC EQU 020H ; DISKETTE CONTROLLER HAS FAILED
563 = 0010 C BAD_CRC EQU 010H ; BAD CRC ON DISKETTE READ
564 = 0009 C DMA_BOUNDARY EQU 009H ; ATTEMPT TO DMA ACROSS 64K BOUNDARY
565 = 0008 C BAD_DMA EQU 008H ; DMA OVERRUN ON OPERATION
566 = 0006 C MEDIA_CHANGE EQU 006H ; MEDIA REMOVED ON DUAL ATTACH CARD
567 = 0004 C RECORD_NOT_FND EQU 004H ; REQUESTED SECTOR NOT FOUND
568 = 0003 C WRITE_PROTECT EQU 003H ; WRITE ATTEMPTED ON WRITE PROTECT DISK
569 = 0002 C BAD_ADDR_MARK EQU 002H ; ADDRESS MARK NOT FOUND
570 = 0001 C BAD_CMD EQU 001H ; BAD COMMAND PASSED TO DISKETTE I/O
571 C |-----|
572 C |-----| DISK CHANGE LINE EQUATES
573 = 0001 C NOCHGLN EQU 001H ; NO DISK CHANGE LINE AVAILABLE
574 = 0002 C CHGLN EQU 002H ; DISK CHANGE LINE AVAILABLE
575 C |-----|
576 C |-----| MEDIA/DRIVE STATE INDICATORS
577 = 0001 C TRK_CAPA EQU 00000001B ; 80 TRACK CAPABILITY
578 = 0002 C FMT_CAPA EQU 00000010B ; MULTIPLE FORMAT CAPABILITY (1.2M)
579 = 0000 C DRV_DET EQU 00000010B ; DRIVE DETERMINED
580 = 0010 C MED_DET EQU 00010000B ; MEDIA DETERMINED BIT
581 = 0020 C DBL_STEP EQU 00100000B ; DOUBLE STEP BIT
582 = 00C0 C RATE_MSK EQU 11000000B ; MASK FOR CLEARING ALL BUT RATE
583 = 0000 C RATE_500 EQU 00000000B ; 500 KBS DATA RATE
584 = 0040 C RATE_300 EQU 01000000B ; 300 KBS DATA RATE
585 = 0080 C RATE_250 EQU 10000000B ; 250 KBS DATA RATE
586 = 000C C STRT_MSK EQU 0000000C ; OPERATION STARTED RATE MASK
587 = 00C0 C SEND_MSK EQU 11000000B ; MASK FOR SEND RATE BITS
588 C |-----|
589 C |-----| MEDIA/DRIVE STATE INDICATORS COMPATIBILITY
590 = 0000 C M3D3U EQU 00000000B ; 360 MEDIA/DRIVE NOT ESTABLISHED
591 = 0001 C M3D1U EQU 00000001B ; 360 MEDIA, 1.2DRIVE NOT ESTABLISHED
592 = 0002 C M1D1U EQU 00000010B ; 1.2 MEDIA/DRIVE NOT ESTABLISHED
593 = 0007 C MED_UNK EQU 00000111B ; NONE OF THE ABOVE
    
```

```

594          C PAGE
595          C |----- INTERRUPT EQUATES -----|
596 = 0020    C EQ1          EQU 020H          ; END OF INTERRUPT COMMAND TO 8259
597 = 0020    C INTA00     EQU 020H          ; 8259 PORT
598 = 0021    C INTA01     EQU 021H          ; 8259 PORT
599 = 00A0    C INTB00     EQU 0A0H          ; 2ND 8259
600 = 00A1    C INTB01     EQU 0A1H          ;
601 = 0070    C INT_TYPE   EQU 070H          ; START OF 8259 INTERRUPT TABLE LOCATION
602 = 0010    C INT_VECTOR EQU 010H          ; VIDEO VECTOR
603          C |-----|
604 = 0008    C DMA08      EQU 008H          ; DMA STATUS REGISTER PORT ADDRESS
605 = 0000    C DMA       EQU 000H          ; DMA CH.0 ADDRESS REGISTER PORT ADDRESS
606 = 00D0    C DMA18      EQU 0D0H          ; 2ND DMA STATUS PORT ADDRESS
607 = 00C0    C DMA1      EQU 0C0H          ; 2ND DMA CH.0 ADDRESS REGISTER ADDRESS
608          C |-----|
609 = 0040    C TIMER      EQU 040H          ; 8254 TIMER - BASE ADDRESS
610          C |-----|
611          C |----- MANUFACTURING PORT -----|
612 = 0080    C MFG_PORT    EQU 80H          ; MANUFACTURING AND POST CHECKPOINT PORT
613          C |-----|
614          C |----- MANUFACTURING BIT DEFINITION FOR 0MFG ERR FLAG+1 -----|
615          C |-----|
616 = 0001    C MEM_FAIL    EQU 00000010B    ; STORAGE TEST FAILED (ERROR 20X)
617 = 0002    C PRO_FAIL    EQU 00000010B    ; VIRTUAL MODE TEST FAILED (ERROR 104)
618 = 0004    C LMCS_FAIL   EQU 00001000B    ; LOW MEG CHIP SELECT FAILED (ERROR 109)
619 = 0008    C KYCLK_FAIL  EQU 00001000B    ; KEYBOARD CLOCK TEST FAILED (ERROR 304)
620 = 0010    C KY_SYS_FAIL EQU 00010000B    ; KEYBOARD DR SYSTEM FAILED (ERROR 303)
621 = 0020    C KYBD_FAIL   EQU 00100000B    ; KEYBOARD FAILED (ERROR 301)
622 = 0040    C DSK_FAIL    EQU 01000000B    ; DISKETTE TEST FAILED (ERROR 601)
623 = 0080    C KEY_FAIL    EQU 10000000B    ; KEYBOARD LOCKED (ERROR 302)
624          C |-----|
625          C |-----|
626 = 0081    C DMA_PAGE    EQU 081H          ; START OF DMA PAGE REGISTERS
627 = 008F    C LAST_DMA_PAGE EQU 08FH          ; LAST DMA PAGE REGISTER
628          C |-----|
629          C |-----|
630 = 00F0    C X287       EQU 0F0H          ; MATH COPROCESSOR CONTROL PORT
631          C |-----|
632          C |-----|
633 = 0000    C POST_SS     EQU 000000H    ; POST STACK SEGMENT
634 = 8000    C POST_SP     EQU 080000H    ; POST STACK POINTER
635          C |-----|
636          C |-----|
637 = 000D    C CR          EQU 000DH          ; CARRIAGE RETURN CHARACTER
638 = 000A    C LF          EQU 000AH          ; LINE FEED CHARACTER
639 = 0008    C RVRT       EQU 00010000B    ; VIDEO VERTICAL RETRACE BIT
640 = 0001    C RHRZ       EQU 00000010B    ; VIDEO HORIZONTAL RETRACE BIT
641 = 0100    C H          EQU 256          ; HIGH BYTE FACTOR (X 100H)
642 = 0101    C X          EQU H+1          ; HIGH AND LOW BYTE FACTOR (X 101H)
643          C |-----|
644          C .LIST
    
```

```

645 PAGE
646 C INCLUDE SYSDATA.INC
647 C -----
648 C ;
649 C ; PROTECTED MODE EQUATES FOR POST TESTS AND BIOS ROUTINES
650 C ;-----
651 C ;----- LENGTH EQUATES FOR PROTECTED MODE TESTS
652 C
653 C SDA_LEN EQU 00300H ; SYSTEM DATA AREA LENGTH
654 C SYS_IDT_LEN EQU 256*8 ; 256 SYSTEM IDT ENTRIES, 8 BYTES EACH
655 C GDT_LEN EQU TYPE GDT DEF ; GDT STRUCTURE LENGTH
656 C DESC_LEN EQU TYPE DATA_DESC ; LENGTH OF A DESCRIPTOR
657 C MCRT_SIZE EQU 4*1024 ; MONOCHROME CRT SIZE
658 C CCRT_SIZE EQU 16*1024 ; COMPATIBLE COLOR CRT SIZE
659 C ECRT_SIZE EQU 0FFFFH ; SIZE OF EACH PORTION OF THE ENHANCED
660 C MAX_SEG_LEN EQU 0FFFFH ; MAXIMUM SEGMENT LENGTH = 64K
661 C NULL_SEG_LEN EQU 00000H ; NULL SEGMENT LENGTH = 0
662 C
663 C ;----- LOCATION EQUATES FOR PROTECTED MODE TESTS
664 C
665 C SYS_IDT_LOC EQU 0D0A0H ; THE SYSTEM IDT IS AT THE BOTTOM
666 C SDA_LOC EQU 00400H ; SAME AS REAL
667 C GDT_LOC EQU (SYS_IDT_LOC + SYS_IDT_LEN)
668 C MCRT_HI EQU 0000H ; MONOCHROME CRT ADDRESS
669 C MCRT_LO EQU 0BH ; (0B0000H)
670 C CCRT_HI EQU 8000H ; COMPATIBLE COLOR CRT ADDRESS
671 C CCRT_LO EQU 0BH ; (0B8000H)
672 C ECRT_HI_LO EQU 0000H
673 C ECRT_LO_HI EQU 0AH ; (0A0000H)
674 C ECRT_HI_LO EQU 0000H
675 C ECRT_HI_HI EQU 0BH ; (0B0000H)
676 C CSEG_LO EQU 0000H ; CODE SEGMENT POST/BIOS
677 C CSEG_HI EQU 0FH ; (0F0000H) FOR TESTS
678 C NSEG_LO EQU 0000H ; ABS0
679 C NSEG_HI EQU 00H
680 C
681 C ;----- DEFINITIONS FOR ACCESS RIGHTS BYTES
682 C
683 C CPL3_DATA_ACCESS EQU 11110011B ; PRESENT
684 C ; DPL = 3
685 C ; CODE/DATA SEGMENT
686 C ; NOT EXECUTABLE
687 C ; GROW-UP (OFFSET <= LIMIT)
688 C ; WRITABLE
689 C ; ACCESSED
690 C ; DPL = 0
691 C ; CPL 0 - NON-CONFORMING
692 C
693 C CPL0_DATA_ACCESS EQU 10010011B ; DPL = 0
694 C CPL0_CODE_ACCESS EQU 10011011B ; CPL 0 - NON-CONFORMING
695 C LDT_DESC EQU 11100010B
696 C FREE_TSS EQU 10000001B
697 C INT_GATE EQU 10000110B
698 C TRAP_GATE EQU 10000111B
699 C
700 C VIRTUAL_ENABLE EQU 0000000000000011B ; PROTECTED MODE ENABLE
701 C
702 C ;----- THE GLOBAL DESCRIPTOR TABLE DEFINITION FOR POWER ON SELF TESTS
703 C
704 C GDT_DEF STRUC ?
705 C ; UNUSED ENTRY
706 C GDT_PTR DQ ? ; THIS ENTRY POINTS TO THIS TABLE
707 C SYS_IDT_PTR DQ ? ; POST INTERRUPT DESCRIPTOR TABLE
708 C RSDT_PTR DQ ? ; THE REAL SYSTEM DATA AREA FOR POST
709 C BWCRT_PTR DQ ? ; COMPATIBLE BW CRT FOR POST
710 C CCRT_PTR DQ ? ; COMPATIBLE COLOR CRT FOR POST
711 C ECRT_PTR DQ ? ; ENHANCED COLOR GRAPHICS CRT (16 BYTES)
712 C E_CRT_PTR2 DQ ?
713 C STS_ROM_CS DQ ? ; CS - POST IDT, ROM RESIDENT
714 C ES_TEMP DQ ? ; DYNAMIC POINTER FOR ES
715 C CS_TEMP DQ ? ; DYNAMIC POINTER FOR CS
716 C SS_TEMP DQ ? ; DYNAMIC POINTER FOR SS
717 C DS_TEMP DQ ? ; DYNAMIC POINTER FOR DS
718 C POST_TR DQ ? ; TR VALUE FOR THIS MACHINE'S TSS
719 C POST_TSS_PTR DQ ?
720 C POST_LDR DQ ? ; LDR VALUE FOR THIS MACHINE'S LDT
721 C POST_LDT_PTR DQ ?
722 C GDT_DEF ENDS
723 C
724 C ;----- SEGMENT DESCRIPTOR TABLE ENTRY STRUCTURE
725 C
726 C DATA_DESC STRUC ?
727 C SEG_LIMIT DW ? ; SEGMENT LIMIT (1 - 65535 BYTES)
728 C BASE_LO_WORD DW ? ; 24 BIT SEGMENT PHYSICAL
729 C BASE_HI_BYTE DB ? ; ADDRESS (0 - (16M-1))
730 C DATA_ACC_RIGHTS DB ? ; ACCESS RIGHTS BYTE
731 C DATA_RESERVED DW ? ; RESERVED - MUST BE 0000 FOR THE 80286
732 C DATA_DESC ENDS
733 C
734 C ;----- GATE DESCRIPTOR TABLE ENTRY STRUCTURE
735 C
736 C GATE_DESC STRUC ?
737 C ENTRY_POINT DW ? ; DESTINATION ROUTINE ENTRY POINT
738 C CS_SELECTOR DW ? ; SELECTOR FOR DESTINATION SEGMENT
739 C WORD_COUNT DB ? ; NUMBER OF WORDS TO COPY FROM STACK
740 C GATE_ACC_RIGHTS DB ? ; ACCESS RIGHTS BYTE
741 C GATE_RESERVED DW ? ; RESERVED - MUST BE 0000 FOR THE 80286
742 C GATE_DESC ENDS
743 C
744 C .LIST
    
```

```

743          PAGE
744 0000      CODE      SEGMENT WORD PUBLIC
745
746          PUBLIC C8042
747          PUBLIC OBF_42
748          PUBLIC POST1
749          PUBLIC START_1
750
751          EXTRN CMOS_READ;NEAR
752          EXTRN CMOS_WRITE;NEAR
753          EXTRN CONFG_BAD;NEAR
754          EXTRN D11;NEAR
755          EXTRN DDS;NEAR
756          EXTRN DUMMY_RETURN;NEAR
757          EXTRN ERR_BEEP;NEAR
758          EXTRN GATE_A20;NEAR
759          EXTRN KBD_RESET;NEAR
760          EXTRN NMI_INT;NEAR
761          EXTRN POST2;NEAR
762          EXTRN PRINT_SCREEN;NEAR
763          EXTRN PROC_SHUTDOWN;NEAR
764          EXTRN ROM_CHECK;NEAR
765          EXTRN SHUT2;NEAR
766          EXTRN SHUT3;NEAR
767          EXTRN SHUT4;NEAR
768          EXTRN SHUT6;NEAR
769          EXTRN SHUT7;NEAR
770          EXTRN SHUT9;NEAR
771          EXTRN SLAVE_VECTOR_TABLE;NEAR
772          EXTRN STGTEST;NEAR
773          EXTRN SYSINIT;NEAR
774          EXTRN VECTOR_TABLE;NEAR
775          EXTRN VIDEO_FARDS;BYTE
776
777          ASSUME CS:CODE,DS:NOTHING,ES:NOTHING,SS:NOTHING
778
779 0000      POST1     PROC      NEAR
780
781 = 0000
782 0000     36 34 38 30 30 39
783         30 43 4F 50 52 2E
784         20 49 42 4D 20 43
785         4F 52 50 2E 20 31
786         39 38 31 2C 31 39
787         38 35 20 20
788
789          EVEN
790          I          6 4 8 0 0 9 0   C O P R .   I B M   1 9 8 5   ;EVEN BOUNDARY
791          DB        '64800901 C O P R .   I B M   1 9 8 5   ;EVEN MODULE
792          DB        '64480009901 CCOPPR... I B M   1 9 8 5   ;ODD MODULE
793          DB        '64480009901 CCOPPR... I B M   1 9 8 5   ;COPYRIGHT NOTICE
794
795 0022     36 36 34 34 38 38
796         30 30 30 30 39 39
797         30 31 20 20 43 43
798         4F 4F 50 50 52 52
799         2E 2E 20 20 49 49
800         42 42 4D 4D 20 20
801         31 31 39 39 38 38
802
803 004E     20 20
804          DB        ' ' ;PAD
805
806          ;-----
807          ; INITIAL RELIABILITY TESTS -- (POST) ;
808          ;-----
809          ; TEST.01 ;
810          ; 80286 PROCESSOR TEST (REAL MODE) ;
811          ; DESCRIPTION ;
812          ; VERIFY FLAGS, REGISTERS ;
813          ; AND CONDITIONAL JUMPS. ;
814          ;-----
815          ASSUME DS:DATA
816
817          START_1:
818          CLI          ; DISABLE INTERRUPTS
819          MOV          AX,0D500H+CMOS_REG_D+NM1 ; FLAG MASK IN (AH) AND NM1 MASK IN (AL)
820          OUT          CMOS_PORT,AL          ; DISABLE NM1 INTERRUPTS
821          SAHF         ; SET "SF" "ZF" "AF" "PF" "CF" FLAGS ON
822          JNC          ERR02                 ; GO TO ERROR ROUTINE IF "CF" NOT SET
823          JNZ          ERR02                 ; GO TO ERROR ROUTINE IF "ZF" NOT SET
824          JNP          ERR02                 ; GO TO ERROR ROUTINE IF "PF" NOT SET
825          JNS          ERR02                 ; GO TO ERROR ROUTINE IF "SF" NOT SET
826          LAHF        ; LOAD FLAG IMAGE TO (AH)
827          MOV          CL,5                  ; LOAD COUNT REGISTER WITH SHIFT COUNT
828          SHR          AH,CL                 ; SHIFT "AF" INTO CARRY BIT POSITION
829          JNC          ERR02                 ; GO TO ERROR ROUTINE IF "AF" NOT SET
830          MOV          AL,40H                ; SET THE "OF" FLAG ON
831          SHL          AL,1                  ; SETUP FOR TESTING
832          JNO          ERR02                 ; GO TO ERROR ROUTINE IF "OF" NOT SET
833          XOR          AH,AH                 ; SET (AH) = 0
834          SAHF        ; CLEAR "SF", "CF", "ZF", AND "PF"
835          JBE          ERR02                 ; GO TO ERROR ROUTINE IF "CF" ON
836          JCF          ERR02                 ; GO TO ERROR ROUTINE IF "ZF" ON
837          JPF          ERR02                 ; GO TO ERROR ROUTINE IF "SF" ON
838          JNF          ERR02                 ; GO TO ERROR ROUTINE IF "PF" ON
839          LAHF        ; LOAD FLAG IMAGE TO (AH)
840          MOV          AH,CL                 ; SHIFT "AF" INTO CARRY BIT POSITION
841          JNC          ERR02                 ; GO TO ERROR ROUTINE IF ON
842          SHL          AH,1                  ; CHECK THAT "OF" IS CLEAR
843          JO          ERR02                 ; GO TO ERROR ROUTINE IF ON
844          JZ          CTA                    ; CONTINUE CONFIDENCE TESTS IF "ZF" SET
845          HLT          ; ERROR HALT
846          JMP          ERR02                 ; ERROR LOOP TRAP
847
848          CTA:
849          MOV          AX,DATA                ; SET DATA SEGMENT
850          MOV          DS,AX                 ; INTO THE (DS) SEGMENT REGISTER
851
852          ;----- CHECK FOR PROCESSOR SHUTDOWN
853          IN          AL,STATUS_PORT         ; READ CURRENT KEYBOARD PROCESSOR STATUS
854          TEST        AL,SYS_FLAG           ; CHECK FOR SHUTDOWN IN PROCESS FLAG
855          JNZ         CTB                    ; GO IF YES
856          JMP         SHJT0                 ; ELSE CONTINUE NORMAL POWER ON CODE

```



```

857 PAGE
858 ;----- CHECK FOR SHUTDOWN 09
859 0091
860 0091 B0 8F      MOV     AL,CMOS_SHUT_DOWN+NMI ; CMOS ADDRESS FOR SHUTDOWN BYTE
861 0093 E6 70      OUT     CMOS_PORT,AL
862 0095 EB 00      JMP     $+2
863 0097 E4 71      IN      AL,CMOS_DATA          ; GET REQUEST NUMBER
864 0099 3C 09      CMP     AL,09H                ; WAS IT SHUTDOWN REQUEST 9?
865 009B 86 C4      XCHG   AL,AH                  ; SAVE THE SHUTDOWN REQUEST
866 009D 74 41      JEC    CTC                    ; BYPASS INITIALIZING INTERRUPT CHIPS
867
868 ;----- CHECK FOR SHUTDOWN 0A
869
870 009F 80 FC 0A    CMP     AH,0AH                 ; WAS IT SHUTDOWN REQUEST A?
871 00A2 74 3C      JEC    CTC                    ; BYPASS INITIALIZING INTERRUPT CHIPS
872
873 00A4 2A C0      SUB     AL,AL                  ; INSURE MATH PROCESSOR RESET
874 00A6 E6 F1      OUT     X287+1,AL
875
876
877 ;----- RE-INITIALIZE THE 8259 INTERRUPT #1 CONTROLLER CHIP :
878
879 00AB B0 11      MOV     AL,11H                ; ICW1 - EDGE, MASTER, ICW4
880 00AA E6 20      OUT     INTA00,AL
881 00AC EB 00      JMP     $+2                   ; WAIT STATE FOR I/O
882 00AE B0 08      MOV     AL,08H                ; SETUP ICW2 - INTERRUPT TYPE 8H (8-F)
883 00B0 E6 21      OUT     INTA01,AL
884 00B2 EB 00      JMP     $+2                   ; WAIT STATE FOR I/O
885 00B4 B0 04      MOV     AL,04H                ; SETUP ICW3 - MASTER LEVEL 2
886 00B6 E6 21      OUT     INTA01,AL
887 00B8 EB 00      JMP     $+2                   ; I/O WAIT STATE
888 00BA B0 01      MOV     AL,01H                ; SETUP ICW4 - MASTER,8086 MODE
889 00BC E6 21      OUT     INTA01,AL
890 00BE EB 00      JMP     $+2                   ; WAIT STATE FOR I/O
891 00C0 B0 FF      MOV     AL,0FFH               ; MASK ALL INTERRUPTS OFF
892 00C2 E6 21      OUT     INTA01,AL             ; (VIDEO ROUTINE ENABLES INTERRUPTS)
893
894 ;----- RE-INITIALIZE THE 8259 INTERRUPT #2 CONTROLLER CHIP :
895
896 00C4 B0 11      MOV     AL,11H                ; ICW1 - EDGE, SLAVE ICW4
897 00C6 E6 A0      OUT     INTB00,AL
898 00C8 EB 00      JMP     $+2                   ; WAIT STATE FOR I/O
899 00CA B0 70      MOV     AL,INT_TYPE           ; SETUP ICW2 - INTERRUPT TYPE 70 (70-7F)
900 00CC E6 A1      OUT     INTB01,AL
901 00CE B0 02      MOV     AL,02H                ; SETUP ICW3 - SLAVE LEVEL 2
902 00D0 EB 00      JMP     $+2                   ; I/O DELAY
903 00D2 E6 A1      OUT     INTB01,AL
904 00D4 EB 00      JMP     $+2                   ; I/O DELAY
905 00D6 B0 01      MOV     AL,01H                ; SETUP ICW4 - 8086 MODE, SLAVE
906 00D8 E6 A1      OUT     INTB01,AL
907 00DA EB 00      JMP     $+2                   ; WAIT STATE FOR I/O
908 00DC B0 FF      MOV     AL,0FFH               ; MASK ALL INTERRUPTS OFF
909 00DE E6 A1      OUT     INTB01,AL
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 BOOT LOADER 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 - (I/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 70H 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 ;----- CHECK FROM WHERE
938
939 00E0
940 00E0 B0 8F      MOV     AL,CMOS_SHUT_DOWN+NMI ; CLEAR CMOS BYTE
941 00E2 E6 70      OUT     CMOS_PORT,AL
942 00E4 90      NOP
943 00E5 2A C0      SUB     AL,AL                  ; I/O DELAY
944 00E7 E6 71      OUT     CMOS_DATA,AL          ; SET BYTE TO 0
945 00E9 86 E0      XCHG   AH,AL
946 00EB 3C 0A      CMP     SHUTO,JA              ; COMPARE WITH MAXIMUM TABLE ENTRIES
947 00ED 77 34      JNC    SHUTO                  ; SKIP TO POST IF GREATER THAN MAXIMUM
948 00EF BE 0103 R  MOV     SI,OFFSET BRANCH      ; POINT TO THE START OF THE BRANCH TABLE
949 00F2 03 F0      ADD     SI,AX
950 00F4 03 F0      ADD     SI,AX
951 00F6 2E 1B 1C  MOV     BX,CS:[SI]           ; POINT TO BRANCH ADDRESS
952 ; MOVE BRANCH TO ADDRESS TO BX REGISTER
953
954 ;----- SET TEMPORARY STACK FOR POST
955
956 00F9 B8 ---- R  MOV     AX,AB50               ; SET STACK SEGMENT TO AB50 SEGMENT
957 00FC 8E D0      MOV     SS,AX
958 00FE BC 0400 R  MOV     SP,OFFSET #TOS       ; SET STACK POINTER TO END OF VECTORS
959 0101 FF E3      JMP     BX                     ; JUMP BACK TO RETURN ROUTINE
960
961 0103 0123 R     BRANCH: DW SHUTO              ; NORMAL POWER UP/UNEXPECTED SHUTDOWN
962 0105 0990 R     DW SHUT1                     ; SHUT DOWN AFTER MEMORY SIZE
963 0107 0000 E     DW SHUT2                     ; SHUT DOWN AFTER MEMORY TEST
964 0109 0000 E     DW SHUT3                     ; SHUT DOWN WITH MEMORY ERROR
965 010B 0000 E     DW SHUT4                     ; SHUT DOWN WITH BOOT LOADER REQUEST
966 010D 0119 R     DW SHUT5                     ; JMP DWORD REQUEST WITH INTERRUPT INIT
967 010F 0000 E     DW SHUT6                     ; PROTECTED MODE TEST3 PASSED
968 0111 0000 E     DW SHUT7                     ; PROTECTED MODE TEST1 FAILED
969 0113 0793 R     DW SHUT8                     ; PROTECTED MODE TEST1 FAILED
970 0115 0000 E     DW SHUT9                     ; BLOCK MOVE SHUTDOWN REQUEST
971 0117 011F R     DW SHUTA                    ; JMP DWORD REQUEST (I/O INTERRUPT INIT)
    
```

SECTION 5


```

1541 03C6 2B F6          SUB     SI,SI
1542 03C8 2B C0          SUB     AX,AX
1543 03CA 8E D8          MOV     DS,AX
1544 03CC 8E C0          MOV     ES,AX
1545 03CE 81 BF 1234     CMP     BX,1234H          ; WARM START?
1546 03D0 75 03          JNZ     E30A 0           ; GO IF NOT
1547 03D4 E9 0582 R     JMP     CLR_STG
1548
1549
1550
1551 03D7 B0 0F          ;----- GET THE INPUT BUFFER (SWITCH SETTINGS)
1552 03D9 E6 80          E30A_0: MOV    AL,0FH           ;
1553
1554 03DB B0 80          ;
1555 03DD E6 87          ;
1556 03DF BC 03EC R     MOV    AL,PARITY_CHECK   ; SET BASE MEMORY PARITY
1557 03E2 E9 0000 E     OUT    DMA_PAGE+6,AL     ; USE AS TEMPORARY SAVE
1558 03E5 8B D8          MOV    SP,OFFSET C2      ; SET RETURN ADDRESS
1559 03E7 75 0F          JMP     STGTST_CNT
1560 03E9 E9 058D R     JNZ    BX,AX             ; SAVE FAILING BIT PATTERN
1561
1562
1563
1564 03EC 03E5 R         JMP    C31                ; STORAGE OK, CONTINUE
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582 03F8
1583 03FA 8A C7          C31:   MOV    AL,BH              ; SAVE HIGH BYTE
1584 03FC E6 81          OUT    MFG_PORT+1,AL     ;
1585 03FE 8A C3          MOV    AL,BL              ; SAVE LOW BYTE
1586 03FF E6 82          OUT    MFG_PORT+2,AL     ;
1587
1588
1589
1590 0400 B9 C000        ;----- CHECK FOR VIDEO ROM
1591 0403 8E D9          MOV    CX,0C000H         ; START OF I/O ROM
1592 0405 2B DB          M1:   SUB    BX,BX              ; POINT TO SEGMENT
1593 0407 8B 07          MOV    AX,[BX]           ; GET THE FIRST 2 LOCATIONS
1594 0409 EB 00          JMP     $+2                ; BUS SETTLE
1595 040B 3D AA55        CMP    AX,0AA55H         ; IS THE VIDEO ROM PRESENT?
1596 040E 6A 03          POP    AX
1597 0410 74 0C          JZ     Z5                  ; GO IF YES
1598 0412 81 C1 0080    ADD    CX,080H           ; POINT TO NEXT 2K BLOCK
1599 0416 81 F7 C800    CMP    CX,0C800H        ; TOP OF VIDEO ROM AREA YET?
1600 041A 7C E1          JZ     M1                  ; TRY AGAIN
1601 041C 23 C9          AND    CX,CX              ; SET NON ZERO FLAG
1602 041E
1603 041E 75 03          Z5:   JNZ    C32                ; GO IF NOT
1604 0420 E9 050F R     JMP    C31_0              ; BYPASS ERROR DISPLAY IF VIDEO ROM
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615 0423 BA 03D8        ;----- INITIALIZE COLOR/MONOCHROME
1616 0426 2A C0          C32:   MOV    DX,3DBH           ; CONTROL REGISTER ADDRESS OF COLOR CARD
1617 0428 EE          SUB    AL,AL              ; MODE SET
1618
1619 0429 BA 03B8        MOV    DX,03B8H          ; CONTROL REGISTER ADDRESS OF B/W CARD
1620 042C B0 01          MOV    AL,1               ; MODE SET FOR CARD
1621 042E EE          OUT    DX,AL              ; RESET VIDEO
1622 042F 83 EA 04          SUB    DX,4               ; BACK TO BASE REGISTER
1623
1624 = 0010
1625
1626 0432 BB 0030 E     M4:   MOV    BX,OFFSET VIDEO_PARAMS+M4*3 ; POINT TO VIDEO PARAMETERS
1627
1628 0435 B9 0010        ASSUME DS:CODE
1629
1630
1631
1632 0438 32 E4          Z_2:  MOV    CX,M4              ; COUNT OF MONOCHROME VIDEO PARAMETERS
1633
1634
1635
1636 043A 8A C4          ;----- BX POINTS TO CORRECT ROW OF INITIALIZATION TABLE
1637 043C EE          XOR    AH,AH              ; AH IS REGISTER NUMBER DURING LOOP
1638 043D 42
1639 043E FE C4          ;----- LOOP THROUGH TABLE, OUTPUTTING REGISTER ADDRESS, THEN VALUE FROM TABLE
1640 0440 2E: 8A 07          M10:  MOV    AL,AH              ; GET 6845 REGISTER NUMBER
1641 0443 EE          OUT    DX,AL              ;
1642 0444 43          INC    DX                  ; POINT TO DATA PORT
1643 0445 4A          INC    AH                  ; NEXT REGISTER VALUE
1644 0446 E2 F2          MOV    AL,CS:[BX]         ; GET TABLE VALUE
1645 0448 8A E2          OUT    DX,AL              ; OUT TO CHIP
1646 044A 80 E4 F0          INC    BX                  ; NEXT IN TABLE
1647 044D 80 FC D0          DEC    DX                  ; BACK TO POINTER REGISTER
1648 0450 74 08          LOOP  M10                 ; DO THE WHOLE TABLE
1649 0452 BB 0000 E     MOV    AH,DL              ; CHECK IF COLOR CARD DONE
1650 0455 BA 03D4        AND    AH,0F0H            ; STRIP UNWANTED BITS
1651 0458 EB DB          CMP    AH,00DH            ; IS IT THE COLOR CARD?
1652
1653
1654
1655 0452 BB 0000 E     JZ     Z_3                 ; CONTINUE IF COLOR
1656 0455 BA 03D4        MOV    BX,OFFSET VIDEO_PARAMS ; POINT TO VIDEO PARAMETERS
1657 0458 EB DB          MOV    DX,3D4H            ; COLOR BASE
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000

```

```

1655 045A 33 FF          Z_3: XOR    DI,D1          ; SET UP POINTER FOR REGEN
1656 045C B8 B000       MOV    AX,0B000H     ; SET UP ES TO VIDEO REGEN
1657 045F 8E C0         MOV    ES,AX
1658
1659 0461 B9 0800       MOV    CX,2048       ; NUMBER OF WORDS IN MONOCHROME CARD
1660 0464 B8 0720       MOV    AX,' '*7*H    ; FILL CHARACTER FOR ALPHA + ATTRIBUTE
1661 0467 F3/ AB        REP    STOSW         ; FILL THE REGEN BUFFER WITH BLANKS
1662
1663 0469 33 FF          XOR    DI,D1         ; CLEAR COLOR VIDEO BUFFER MEMORY
1664 046B BB B800       MOV    BX,0B800H    ; SET UP ES TO COLOR VIDEO MEMORY
1665 046E 8E C3         MOV    ES,BX
1666 0470 B9 2000       MOV    CX,8192
1667 0473 F3/ AB        REP    STOSW         ; FILL WITH BLANKS
1668
1669
1670
1671 0475 BA 03B8        ;----- ENABLE VIDEO AND CORRECT PORT SETTING
1672 0478 B0 29         MOV    DX,3B8H
1673 047A EE           OUT    DX,AL         ; SET VIDEO ENABLE PORT
1674
1675
1676
1677 047B 42             ;----- SET UP OVERSCAN REGISTER
1678 047C B0 30         INC    DX            ; SET OVERSCAN PORT TO A DEFAULT
1679 047E EE           OUT    DX,AL         ; VALUE 30H FOR ALL MODES EXCEPT 640X200
1680
1681
1682
1683 047F BA 03D8        ;----- ENABLE COLOR VIDEO AND CORRECT PORT SETTING
1684 0482 B0 28         MOV    DX,3D8H
1685 0484 EE           OUT    DX,AL         ; SET VIDEO ENABLE PORT
1686
1687
1688
1689 0485 42             ;----- SET UP OVERSCAN REGISTER
1690 0486 B0 30         INC    DX            ; SET OVERSCAN PORT TO A DEFAULT
1691 0488 EE           OUT    DX,AL         ; VALUE 30H FOR ALL MODES EXCEPT 640X200
1692
1693
1694
1695 0489 8C C8         ;----- DISPLAY FAILING CHECKPOINT AND
1696 048B 8E D0         MOV    AX,CS         ; SET STACK SEGMENT TO CODE SEGMENT
1697
1698 048D BB B000       MOV    BX,0B000H    ; SET DS TO B/W DISPLAY BUFFER
1699 0490 8E DB        MOV    DS,BX
1700
1701 0492 B0 30         Z_0: MOV    AL,'0'       ; DISPLAY BANK 000000
1702 0494 B9 0006       MOV    CX,6         ; START AT 0
1703 0497 2B FF        SUB    DI,D1         ; WRITE TO DISPLAY REGEN BUFFER
1704 0499 8B 05        MOV    [DI],AL      ; POINT TO NEXT POSITION
1705 049B 47           INC    DI
1706 049C 47           INC    DI
1707 049D E2 FA        LOOP   Z             ;
1708
1709 049F 80 FF FB      CMP    BH,0B8H      ; CHECK THAT COLOR BUFFER WRITTEN
1710 04A2 74 0C        JZ     Z_1           ;
1711 04A4 2B FF        SUB    DI,DI        ; POINT TO START OF BUFFER
1712
1713 04A6 BT B0         MOV    BH,0B0H
1714 04A8 8E C3         MOV    ES,BX        ; ES = MONOCHROME
1715 04AA BT B8         MOV    BH,0B8H     ; SET SEGMENT TO COLOR
1716 04AC 8E DB        MOV    DS,BX        ; DS = COLOR
1717 04AE EB E2        JMP    Z_0
1718
1719
1720
1721 04B0 80 20         ;----- PRINT FAILING BIT PATTERN
1722 04B2 8B 05        Z_1: MOV    AL,' '       ; DISPLAY A BLANK
1723 04B4 26: 88 05    MOV    [DI],AL      ; WRITE TO COLOR BUFFER
1724 04B7 47           INC    DI            ; WRITE TO MONOCHROME REGEN BUFFER
1725 04B8 47           INC    DI            ; POINT TO NEXT POSITION
1726 04B9 E4 81       IN     AL,MFG_PORT+1 ; GET THE HIGH BYTE OF FAILING PATTERN
1727 04BB B1 04       MOV    CL,4         ; SHIFT COUNT
1728 04BD D2 E8       SHR    AL,CL        ; NIBBLE SWAP
1729 04BF BC 057A R    MOV    SP,OFFSET Z1_0 ; NIBBLE SWAP
1730 04C2 EB 1B       JMP    SHORT PR
1731
1732 04C4 E4 81       Z1:  IN     AL,MFG_PORT+1 ; ISOLATE TO LOW NIBBLE
1733 04C6 24 0F       AND    AL,0FH
1734 04C8 BC 057C R    MOV    SP,OFFSET Z2_0 ;
1735 04CB 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 057E R    MOV    SP,OFFSET Z3_0 ;
1740 04D6 EB 07       JMP    SHORT PR
1741 04D8 E4 82       Z3:  IN     AL,MFG_PORT+2 ; ISOLATE TO LOW NIBBLE
1742 04DA 24 0F       AND    AL,0FH
1743 04DC BC 0580 R    MOV    SP,OFFSET Z4_0 ; RETURN TO Z4:
1744
1745
1746
1747 04DF 04 90        ;----- CONVERT AND PRINT
1748 04E1 27           PR:  ADD    AL,090H     ; CONVERT 00-0F TO ASCII CHARACTER
1749 04E2 14 40        ; ADD FIRST CONVERSION FACTOR
1750 04E4 27           DAA                    ; ADJUST FOR NUMERIC AND ALPHA RANGE
1751
1752 04E5 8B 05        ; ADD CONVERSION AND ADJUST LOW NIBBLE
1753 04E7 26: 88 05    DAA                    ; ADJUST HIGH NIBBLE TO ASCII RANGE
1754 04EA 47           MOV    [DI],AL      ; WRITE TO COLOR BUFFER
1755 04EB 47           INC    DI            ; WRITE TO MONOCHROME BUFFER
1756 04EC C3         INC    DI            ; POINT TO NEXT POSITION
1757
1758
1759
1760 04ED 80 20        ;----- DISPLAY 201 ERROR
1761 04EF 8B 05        Z4:  MOV    AL,' '       ; DISPLAY A BLANK
1762 04F1 26: 88 05    MOV    [DI],AL      ; WRITE TO DISPLAY REGEN BUFFER
1763 04F4 47           MOV    ES:[DI],AL   ; WRITE TO MONOCHROME BUFFER
1764 04F5 47           INC    DI            ; POINT TO NEXT POSITION
1765 04F6 B0 32        INC    DI
1766 04F8 8B 05        MOV    AL,'2'       ; DISPLAY 201 ERROR
1767 04FA 26: 88 05    MOV    [DI],AL      ; WRITE TO DISPLAY REGEN BUFFER
1768 04FD 47           MOV    ES:[DI],AL   ; WRITE TO MONOCHROME BUFFER
1769 04FE 47           INC    DI            ; POINT TO NEXT POSITION
    
```



```

1883
1884
1885
1886 05A3 32 DB
1887 05A5 33 C9
1888 05A7 90
1889 05AB
1890 05AB E4 61
1891 05AA A8 10
1892 05AC E1 FA
1893 05AE
1894 05AE E4 61
1895 05B0 A8 10
1896 05B2 E0 FA
1897
1898 05B4 FE CB
1899 05B6 75 F0
1900
1901 05B8 81 F9 FB A7
1902 05BC 73 07
1903 05BE
1904 05BE BA 0101
1905 05C1 E8 0000 E
1906 05C4 F4
1907 05C5
1908 05C5 81 F9 F9 FD
1909 05C9 77 F3
1910
1911
1912
1913 05CB E4 82
1914 05CD 24 F8
1915 05CF A2 0012 R
1916 05D2 2A C0
1917 05D4 E6 82
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931 05D6 0F 01 E0
1932 05D9 A9 000F
1933 05DC 75 34
1934
1935
1936
1937 05DE B0 12
1938 05E0 E6 80
1939
1940 05E2 1E
1941 05E3 07
1942 05E4 BF D0A0
1943 05E7 B9 0003
1944 05EA B8 AAAA
1945 05ED E8 0615 R
1946 05F0 B8 5555 R
1947 05F3 E8 0615 R
1948 05F6 2B C0
1949 05F8 E8 0615 R
1950
1951
1952
1953 05FB FD
1954 05FC 9C
1955 05FD 58
1956 05FE A9 0200
1957 0601 75 0F
1958 0603 A9 0400
1959 0606 74 0A
1960 0608 FC
1961 0609 9C
1962 060A 58
1963 060B A9 0400
1964 060E 75 02
1965
1966 0610 EB 3D
1967 0612
1968 0612 F4
1969 0613 EB FD
1970
1971
1972
1973 0615 B9 0003
1974 0618 F3/ AB
1975 061A BD D0A0
1976
1977 061D 26
1978
1979 061E 0F
1980 061F
+ ?70001 LABEL BYTE
1981 061F 8B 5E 00
+ ?70002 LABEL BYTE
1982 0622
+ ORG OFFSET CS:770001
1983 061F
+ DB
1984 061F 01
+ ORG OFFSET CS:770002
1985 0622
+ MOV BP,SYS_IDT_LOC
1986 0622 BD D0A0
+ SEGOV ES
1987
+ DB 026H
1988 0625 26
+ LGDT [BP]
1989
+ DB 00FH
1990 0626 0F
+ ?70004 LABEL BYTE
1991 0627
+ MOV DX,WORD PTR [BP]
1992 0627 8B 56 00
+ ?70005 LABEL BYTE
1993 062A
+ ORG OFFSET CS:770004
1994 0627
+ DB 001H
1995 0627 01
+ ORG OFFSET CS:770005
1996 062A
    
```



```

2681 0A0A 7C E3          JL   CHK_VIDEO1      ; TRY AGAIN
2682 0A0C 23 C9          AND  CX,CX           ; SET NON ZERO FLAG
2683 0A0E                CHK_VIDEO2:
2684 0A0E C3            RET                  ; RETURN TO CALLER
2685
2686                ;----- CMOS VIDEO BITS NON ZERO (CHECK FOR PRIMARY DISPLAY AND NO VIDEO ROM)
2687
2688 0A0F                MOS_OK_1:
2689 0A0F E8 09EC R      CALL  CHK_VIDEO      ; IS THE VIDEO ROM INSTALLED?
2690 0A12 74 26          JZ   BAD_MOS        ; WRONG CONFIGURATION IN CONFIG BYTE
2691
2692 0A14 8A C4          MOV  AL,AH           ; RESTORE CONFIGURATION
2693 0A16 F6 06 0012 R 40 TEST  0MFG_TST,DSP_JMP ; CHECK FOR DISPLAY JUMPER
2694 0A1B 74 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                ;----- COLOR CARD
2705
2706 0A27                MOS_OK_2:
2707 0A27 74 30          AND  AL,30H         ; STRIP UNWANTED BITS
2708 0A29 3C 30          CMP  AL,30H         ; MUST NOT BE MONO WITH JUMPER INSTALLED
2709 0A2B 8A C4          MOV  AL,AH           ; RESTORE CONFIGURATION
2710 0A2D 74 0B          JZ   BAD_MOS        ; GO IF YES
2711
2712                ;----- CONFIGURATION MUST HAVE AT LEAST ONE DISKETTE
2713
2714 0A2F                MOS_OK:
2715 0A2F A8 01          AND  AL,01H         ; MUST HAVE AT LEAST ONE DISKETTE
2716 0A31 75 26          JNZ  NORMAL_CONFIG  ; GO SET CONFIGURATION IF OK
2717 0A33 F6 06 0012 R 20 TEST  0MFG_TST,MFG_LOOP ; EXCEPT IF MFG JUMPER IS INSTALLED
2718 0A38 74 1F          JZ   NORMAL_CONFIG  ; GO IF INSTALLED
2719
2720                ;----- MINIMUM CONFIGURATION WITH BAD CMOS OR NON VALID VIDEO
2721
2722 0A3A                BAD_MOS:
2723 0A3A B8 00BE        MOV  AX,CMOS_DIAG+NM1 ; GET THE DIAGNOSTIC STATUS
2724 0A3C E8 0000 E      CALL  CMOS_READ      ; WAS BATTERY DEFECTIVE OR BAD CHECKSUM
2725 0A42 75 03          JNZ  BAD_MOS1       ; GO IF YES
2726
2727 0A44 EB 0000 E      CALL  CONFIG_BAD     ; SET THE MINIMUM CONFIGURATION FLAG
2728 0A47
2729 0A47 E8 09EC R      CALL  CHK_VIDEO      ; CHECK FOR VIDEO ROM
2730 0A4A B0 01          MOV  AL,01H         ; DISKETTE ONLY
2731 0A4C 74 0B          JZ   NORMAL_CONFIG  ; GO IF VIDEO ROM PRESENT
2732
2733 0A4E                BAD_MOS1:
2734 0A4E F6 06 0012 R 40 TEST  0MFG_TST,DSP_JMP ; CHECK FOR DISPLAY JUMPER
2735 0A53 B0 11          MOV  AL,11H         ; DEFAULT TO 40x25 COLOR
2736 0A55 74 02          JZ   NORMAL_CONFIG  ; GO IF JUMPER IS INSTALLED
2737
2738 0A57                ;-----
2739 0A57 B0 31          MOV  AL,31H         ; DISKETTE / B/W DISPLAY 80x25
2740
2741                ;-----
2742                ;----- CONFIGURATION AND MFG MODE
2743
2744 0A59                NORMAL_CONFIG:
2745 0A59 F6 06 0012 R 20 TEST  0MFG_TST,MFG_LOOP ; IS THE MANUFACTURING JUMPER INSTALLED
2746 0A60 24 3E          AND  AL,03EH       ; STRIP DISKETTE FOR MFG TEST
2747
2748 0A62                NORM1:
2749 0A62 2A E4          SUB  AH,AH           ; SAVE SWITCH INFORMATION
2750 0A64 A3 0010 R      MOV  0EQUIP_FLAG,AX ;
2751 0A67 81 3E 0072 R 1234 CMP  0RESET_FLAG,1234H ; BYPASS IF SOFT RESET
2752 0A6D 74 2C          JZ   E6              ;
2753
2754                ;----- GET THE FIRST SELF TEST RESULTS FROM KEYBOARD
2755
2755 0A6F                ;-----
2756 0A6F B0 60          MOV  AL,WRITE_8042_LOC ; ENABLE KEYBOARD
2757 0A71 EB 0396 R      MOV  AL,40H         ; ISSUE WRITE BYTE COMMAND
2758 0A74 B0 4D          MOV  AL,40H         ; ENABLE OUTPUT BUFFER FULL INTERRUPT,
2759 0A76 E6 60          OUT  PORT_A,AL      ; SET SYSTEM FLAG, PC I COMPATIBILITY,
2760                                ; INHIBIT OVERRIDE, ENABLE KEYBOARD
2761 0A78 2B C9          SUB  CX,CX           ; WAIT FOR COMMAND ACCEPTED
2762 0A7A EB 039B R      CALL  C42_1         ;
2763
2764 0A7D                ;-----
2765 0A7D B9 7FFF        MOV  CX,07FFFH      ; SET LOOP COUNT FOR APPROXIMATELY 100MS
2766                                ; TO RESPOND
2767 0A80 E4 64          TST  AL,STATUS_PORT ; WAIT FOR OUTPUT BUFFER FULL
2768 0A82 A8 01          IN   AL,OUT_BUF_FULL ; TRY AGAIN IF NOT
2769 0A84 E1 FA          LOOPZ TST6          ;
2770
2771 0A86                ;-----
2772 0A86 9C            PUSHF                ; SAVE FLAGS
2773 0A87 B0 AD          MOV  AL,DIS_KBD     ; DISABLE KEYBOARD
2774 0A89 EB 0396 R      CALL  C8042         ; ISSUE THE COMMAND
2775 0A8C 9D            POPF                 ; RESTORE FLAGS
2776 0A8D 74 0C          JZ   E6              ; CONTINUE WITHOUT RESULTS
2777
2778 0A8F                ;-----
2779 0A8F E4 60          IN   AL,PORT_A      ; GET INPUT FROM KEYBOARD
2780 0A91 A2 0072 R      MOV  BYTE PTR 0RESET_FLAG,AL ; TEMPORARY SAVE FOR AA RECEIVED
2781
2782                ;----- CHECK FOR TEST REQUEST
2783
2784 0A94                ;-----
2785 0A94 3C 65          CMP  AL,065H        ; LOAD MANUFACTURING TEST REQUEST?
2786 0A96 75 03          JNE  E6              ; CONTINUE IF NOT
2787 0A98 E9 0C27 R      JMP  MFG_BOOT       ; ELSE GO TO MANUFACTURING BOOTSTRAP
2788
2789                ;-----
2790 0A9A                ; TEST_14
2791 0A9A 74 14          JZ   TEST_14        ; INITIALIZE AND START CRT CONTROLLER (6845)
2792 0A9C 74 14          JZ   TEST_14        ; TEST VIDEO READ/WRITE STORAGE.
2793 0A9E 74 14          JZ   TEST_14        ; DESCRIPTION
2794 0AA0 74 14          JZ   TEST_14        ; RESET THE VIDEO ENABLE SIGNAL.
2795 0AA2 74 14          JZ   TEST_14        ; SELECT ALPHANUMERIC MODE, 40 * 25, B & W.
2796 0AA4 74 14          JZ   TEST_14        ; READ/WRITE DATA PATTERNS TO MEMORY. CHECK
2797 0AA6 74 14          JZ   TEST_14        ; STORAGE ADDRESSABILITY.
2798 0AA8 74 14          JZ   TEST_14        ; ERROR = 1 LONG AND 2 SHORT BEEPS
2799 0AAA 74 14          JZ   TEST_14

```

SECTION 5


```
3023 0C11 FE C0          INC     AL          ; INITIALIZE FOR 40X25
3024 0C13              E17_2:  PUSH   AX
3025 0C13 50          E17_4:  JMP    E18
3026 0C14              ;----- BOTH VIDEO CARDS FAILED SET DUMMY RETURN IF RETRACE FAILURE
3027 0C14 E9 0B4A R    ;-----
3028
3029
3030
3031 0C17              E17_3:  PUSH   DS          ; SET DS SEGMENT TO 0
3032 0C17 1E          SUB    AX,AX
3033 0C18 2B C0        MOV    DS,AX
3034 0C1A BE D8        MOV    DI,OFFSET @VIDEO INT ; SET INTERRUPT 10H TO DUMMY
3035 0C1C BF 0040 R    MOV    WORD PTR [DI],OFFSET DUMMY_RETURN ; RETURN IF NO VIDEO CARD
3036 0C1F C7 05 0000 E POP    DS
3037 0C23 1F          JMP    E18_1       ; BYPASS REST OF VIDEO TEST
3038 0C24 E9 0B4F R
```

```

3039          PAGE
3040          ;-----
3041          ; MANUFACTURING BOOT TEST CODE ROUTINE
3042          ; LOAD A BLOCK OF TEST CODE THROUGH THE KEYBOARD PORT FOR MANUFACTURING
3043          ; TESTS.
3044          ; THIS ROUTINE WILL LOAD A TEST (MAX LENGTH=FAFFH) THROUGH THE KEYBOARD
3045          ; PORT. CODE WILL BE LOADED AT LOCATION 0000:0500. AFTER LOADING,
3046          ; CONTROL WILL BE TRANSFERRED TO LOCATION 0000:0500. THE STACK WILL
3047          ; BE LOCATED AT 0000:0400. THIS ROUTINE ASSUMES THAT THE FIRST 2 BYTES
3048          ; TRANSFERRED CONTAIN THE COUNT OF BYTES TO BE LOADED
3049          ; (BYTE 1=COUNT LOW, BYTE 2=COUNT HI.)
3050          ;-----
3051
3052          ;----- DEGATE ADDRESS LINE 20
3053
3054          MFG_BOOT:
3055          MOV     AL,DISABLE_BIT20      ; DEGATE COMMAND FOR ADDRESS LINE 20
3056          CALL    GATE_A20             ; ISSUE TO KEYBOARD ADAPTER AND CLI
3057
3058          ;----- SETUP HARDWARE INTERRUPT VECTOR
3059          ;----- SETUP HARDWARE INTERRUPT VECTORS
3060          PUSH   ABS0                  ; SET ES SEGMENT REGISTER TO ABS0
3061          POP    ES
3062          MOV    CX,24                  ; GET VECTOR COUNT
3063          MOV    AX,C5                  ; GET THE CURRENT CODE SEGMENT VALUE
3064          MOV    DS,AX                  ; SETUP DS SEGMENT REGISTER TO
3065          MOV    SI,OFFSET VECTOR_TABLE ; POINT TO THE ROUTINE ADDRESS TABLE
3066          MOV    DI,OFFSET *INT_PTR    ; SET DESTINATION TO FIRST USED VECTOR
3067          MFG_B1:
3068          MOVSW  STOSW                  ; MOVE ONE ROUTINE OFFSET ADDRESS
3069          STOSW  STOSW                  ; INSERT CODE SEGMENT VALUE
3070          LOOP   MFG_B1                 ; MOVE THE NUMBER OF ENTRIES REQUIRED
3071
3072          ;----- SETUP HARDWARE INTERRUPT VECTORS
3073          ;----- SETUP HARDWARE INTERRUPT VECTORS LEVEL 8-15 (VECTORS START AT INT 70 HI)
3074          MOV    CX,08                  ; GET VECTOR COUNT
3075          MOV    SI,OFFSET SLAVE_VECTOR_TABLE
3076          MOV    DI,OFFSET *SLAVE_INT_PTR
3077          MFG_B2:
3078          MOVSW  STOSW                  ; MOVE ONE ROUTINE OFFSET ADDRESS
3079          STOSW  STOSW                  ; INSERT CODE SEGMENT VALUE
3080          LOOP   MFG_B2
3081
3082          ;----- SET UP OTHER INTERRUPTS AS NECESSARY
3083
3084          ASSUME DS:ABS0,ES:ABS0
3085          PUSH   ES                      ; ES= ABS0
3086          POP    DS                      ; SET DS TO ABS0
3087          MOV    WORD PTR *NMI_PTR,OFFSET NMI_INT ; NMI INTERRUPT
3088          MOV    WORD PTR *INT5_PTR,OFFSET PRINT_SCREEN ; PRINT SCREEN
3089          MOV    WORD PTR *BASIC_PTR+2,OF600H ; CASSETTE BASIC SEGMENT
3090
3091          ;----- ENABLE KEYBOARD PORT
3092
3093          MOV    AL,60H                  ; WRITE 8042 MEMORY LOCATION 0
3094          CALL   CB042                   ; ISSUE THE COMMAND
3095          MOV    AL,00001001B            ; SET INHIBIT OVERRIDE/ENABLE OBF
3096          OUT    PORT_A,AL              ; INTERRUPT AND NOT PC COMPATIBLE
3097
3098          CALL   MFG_B4                  ; GET COUNT LOW
3099          MOV    BH,AL                   ; SAVE IT
3100          CALL   MFG_B4                  ; GET COUNT HI
3101          MOV    CH,AL                   ;
3102          MOV    CL,BH                   ; CX NOW HAS COUNT
3103          CLD                             ; SET DIRECTION FLAG TO INCREMENT
3104          MOV    DI,OFFSET *MFG_TEST_RTN ; SET TARGET OFFSET (DS=0000)
3105          MFG_B3:
3106          IN    AL,STATUS_PORT           ; GET 8042 STATUS PORT
3107          TEST  AL,OUT_BUF_FULL          ; KEYBOARD REQUEST PENDING?
3108          JZ    MFG_B3                   ; LOOP TILL DATA PRESENT
3109          IN    AL,PORT_A                 ; GET DATA
3110          STOSB  STORE IT
3111          OUT   MFG_PORT,AL              ; DISPLAY CHARACTER AT MFG PORT
3112          LOOP  MFG_B3                   ; LOOP TILL ALL BYTES READ
3113
3114          JMP    *MFG_TEST_RTN           ; FAR JUMP TO CODE THAT WAS JUST LOADED
3115
3116          MFG_B4:
3117          IN    AL,STATUS_PORT           ; CHECK FOR OUTPUT BUFFER FULL
3118          TEST  AL,OUT_BUF_FULL          ; HANG HERE IF NO DATA AVAILABLE
3119          LOOPZ MFG_B4
3120
3121          IN    AL,PORT_A                 ; GET THE COUNT
3122          RET
3123
3124          POST1  ENDP
3125          CODE   ENDS
3126          END
    
```



```

229 00D0 8A C1          MOV     AL,CL          ; SET TIMER 0 COUNT REGISTER
230 00D2 E6 40          OUT     TIMER+0,AL    ; WRITE TIMER 0 COUNT REGISTER
231 00D4 FB             STI
232 00D5 F6 06 006B R 01 DB:    TEST     @INTR_FLAG,01H
233
234 00DA 75 0D          JNZ    D9             ; DID TIMER 0 INTERRUPT OCCUR?
235 00DC E2 F7          LOOP   D8             ; CHECK TIMER OPERATION FOR SLOW TIME
236
237 00DE C6 06 0015 R 02 MOV     @MFG_ERR_FLAG,02H ; WAIT FOR INTERRUPT FOR SPECIFIED TIME
238
239
240
241 00E3 BE 0000 E      DB_A1 MOV     SI,OFFSET E102 ; DISPLAY 102 ERROR
242 00E6 E9 0051 R      JMP     D6A           ; TIMER 0 INTERRUPT DID NOT OCCUR= ERROR
243
244
245 00E9 B0 2B          D9:    MOV     AL,2BH      ;
246 00EB E6 80          OUT     MFG_PORT,AL  ;
247
248 00ED FA             CLI
249 00EE B4 0C          MOV     CL,12         ; SET PROGRAM LOOP COUNT
250 00F0 B0 FF          MOV     AL,OFFH       ; WRITE TIMER 0 COUNT REGISTER
251 00F2 E6 40          OUT     TIMER+0,AL
252 00F4 C6 06 006B R 00 MOV     @INTR_FLAG,0  ; RESET INTERRUPT RECEIVED FLAG
253 00F9 B0 FC          MOV     AL,OFFH       ; RE-ENABLE TIMER 0 INTERRUPTS
254 00FB E6 21          OUT     INTA01,AL
255 00FD FB             STI
256 00FE F6 06 006B R 01 D10:   TEST     @INTR_FLAG,01H ; DID TIMER 0 INTERRUPT OCCUR?
257 0103 75 08          JNZ    D12           ; YES - TIMER COUNTING TOO FAST, ERROR
258 0105 E2 F7          LOOP   D10           ; WAIT FOR INTERRUPT FOR SPECIFIED TIME
259
260
261
262 0107 2B C9          SUB     CX,CX         ;
263
264 0109 B0 2C          MOV     AL,2CH       ;
265 010B E6 80          OUT     MFG_PORT,AL  ;
266 010D
267 010D F6 06 006B R 01 D110:   TEST     @INTR_FLAG,01H ; DID TIMER 0 INTERRUPT OCCUR?
268 0112 75 08          JNZ    D11           ; GO IF YES
269 0114 E2 F7          LOOP   D11           ; TRY AGAIN
270
271 0116 BE 0000 E      MOV     SI,OFFSET E103 ; DISPLAY 103 ERROR
272 0119 E9 0051 R      JMP     D6A           ; ERROR IF NOT
273
274
275
276 011C FA             CLI
277 011D B0 FF          MOV     AL,OFFH       ; DISABLE ALL DEVICE INTERRUPTS
278 011F E6 21          OUT     INTA01,AL
279 0121 B0 36          MOV     AL,26H       ; SELECT TIMER 0,LSB,MSB,MODE 3
280 0123 E6 43          OUT     TIMER+3,AL    ; WRITE TIMER MODE REGISTER
281 0125 EB 00          JMP     $+2           ; I/O DELAY
282 0127 B0 00          MOV     AL,0          ; WRITE LSB TO TIMER 0 REGISTER
283 0129 E6 40          OUT     TIMER+0,AL   ; I/O DELAY
284 012B EB 00          JMP     $+2           ; I/O DELAY
285 012D E6 40          OUT     TIMER+0,AL   ; WRITE MSB TO TIMER 0 REGISTER
286
287
288
289 012F 2B C9          SUB     CX,CX         ; SET WAIT TIME
290 0131 B0 2D          MOV     AL,2DH       ;
291 0133 E6 80          OUT     MFG_PORT,AL  ;
292 0135 E4 64          IN     AL,STATUS_PORT ; GET THE 8042 STATUS
293 0137 AB 02          TEST   AL,INPT_BUF_FULL ; HAS THE LAST COMMAND BEEN ACCEPTED?
294 0139 74 08          JZ     E19            ; GO IF YES
295 013B E2 F8          LOOP   D13           ; TRY AGAIN
296
297
298
299 013D BE 0000 E      MOV     SI,OFFSET E105 ; PRINT I05 ERROR
300 0140 E9 0051 R      JMP     D6A           ; GO ERROR HALT
301
302
303
304 -----
305 ; TEST_19
306 ;
307 ; ADDITIONAL READ/WRITE STORAGE TEST
308 ; ***** MUST RUN IN PROTECTED MODE *****
309 ;
310 ; DESCRIPTION
311 ; WRITE/READ DATA PATTERNS TO ANY READ/WRITE STORAGE AFTER THE
312 ; FIRST 64K. STORAGE ADDRESSABILITY IS CHECKED.
313 -----
314
315
316
317 014A 81 3E 0072 R 1234 E19:    ASSUME DS:DATA
318 0150 75 03          CALL   DDS           ; SET DATA SEGMENT
319 0152 E9 0420 R      MOV     AL,2FH       ;
320
321 0154 E6 80          OUT     MFG_PORT,AL  ;
322
323 0156 80 30          MOV     AL,30H       ;
324 0157 E6 80          OUT     MFG_PORT,AL  ;
325
326 0159 B8 028F        CMP     @RESET_FLAG,1234H ; WARM START?
327 015C E8 0000 E      JNE    E19A          ; GO IF NOT
328
329 015E 94 00          JMP     SHUT2         ; GO TO NEXT TEST IF WARM START
330
331
332
333 0155 B0 30          E19A: MOV     AL,30H       ;
334 0157 E6 80          OUT     MFG_PORT,AL  ;
335
336 0159 B8 028F        MOV     AX,2*H+CMOS_SHUT_DOWN+NM1 ; ADDRESS FOR SHUTDOWN BYTE
337 015C E8 0000 E      CALL   CMOS_WRITE    ; SECOND ENTRY IN SHUTDOWN TABLE
338
339
340
341
342
343
344
345 015F BC 0000        MOV     SP,POST_SS   ; SET STACK FOR SYSINITI
346 0162 BE D4          MOV     SS,SP         ;
347 0164 BC 8000        MOV     SP,POST_SP   ;
348
349 0167 E8 0000 E      CALL   SYSINITI      ; GO ENABLE PROTECTED MODE
350
351
352 016A B0 31          MOV     AL,31H       ;
353 016C E6 80          OUT     MFG_PORT,AL  ;
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```



```

343 0170 07 POP ES
344 0171 26: C7 06 005A 0000 MOV ES,SS TEMP_BASE_LO_WORD,0
345 0178 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 BE D6 MOV SI,SI
348 0183 BC FFFD MOV SP,MAX_SEG_LEN-2
349
350
351 ;----- DATA SEGMENT TO SYSTEM DATA AREA
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
359 ;----- PRINT 64 K BYTES OK
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 ;----- GET THE MEMORY SIZE DETERMINED (PREPARE BX AND DX FOR BAD CMOS)
364
365 0193 B8 B0B1 MOV AX,(CMOS_U_M_S_LO+NM1)*H-CMOS_U_M_S_HI+NM1
366 0196 E8 0000 E CALL CMOS_READ ; HIGH BYTE
367 0199 B6 E0 XCHG AH,AL ; MASK OFF THE MANUFACTURING TEST BITS
368 019B E8 0000 E CALL CMOS_READ ; LOW BYTE
369 019E B8 1E 0013 R MOV BX,#MEMORY_SIZE ; LOAD THE 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+NM1 ; 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 E20B0 ; 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:
384 01B1 B8 9596 MOV AX,(CMOS_B_M_S_LO+NM1)*H-CMOS_B_M_S_HI+NM1
385 01B4 E8 0000 E CALL CMOS_READ ; HIGH BYTE
386 01B7 24 3F AND AL,0BFH ; MASK OFF THE MANUFACTURING TEST BITS
387 01B9 B6 E0 XCHG AH,AL ; SAVE HIGH BYTE
388 01BB E8 0000 E CALL CMOS_READ ; LOW BYTE OF BASE MEMORY SIZE
389 01BE 3B D0 CMP DX,AX ; IS MEMORY SIZE GREATER THAN CONFIG?
390 01C0 74 13 JZ E20B1 ; GO IF EQUAL
391
392 ;----- SET MEMORY SIZE DETERMINE NOT EQUAL TO CONFIGURATION
393
394 01C2 50 PUSH AX ; SAVE AX
395 01C3 B8 8E8E MOV AX,X*(CMOS_DIAG+NM1) ; ADDRESS THE STATUS BYTE
396 01C6 E8 0000 E CALL CMOS_READ ; GET THE STATUS
397 01C9 0C 10 OR AL,#MEM_SIZE ; SET CMOS FLAG
398 01CB 86 C4 XCHG AL,AH ; SAVE AL AND GET ADDRESS
399 01CD E8 0000 E CALL CMOS_WRITE ; WRITE UPDATED STATUS
400 01D0 58 POP AX ; RESTORE AX
401 01D1 3B D0 CMP DX,AX ; IS MEMORY SIZE GREATER THAN CONFIG?
402 01D3 77 37 JA E20B1 ; DEFAULT TO MEMORY SIZE DETERMINED?
403 01D5 E20B1:
404 01D5 B8 D8 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+NM1)*H+(CMOS_E_M_S_HI+NM1)
410 01DC E8 0000 E CALL CMOS_READ ; HIGH BYTE
411 01DF B6 E0 XCHG AH,AL ; SAVE HIGH BYTE
412 01E1 E8 0000 E CALL CMOS_READ ; LOW BYTE
413 01E4 B8 C8 MOV CX,AX ; SAVE THE ABOVE 640K MEMORY SIZE
414
415 ;----- ABOVE 640K SIZE FROM MEMORY SIZE DETERMINE
416 CX=CONFIG AX=MEMORY SIZE DETERMINE
417 01E6 B8 B0B1 MOV AX,(CMOS_U_M_S_LO+NM1)*H-(CMOS_U_M_S_HI+NM1)
418 01E9 E8 0000 E CALL CMOS_READ ; HIGH BYTE
419 01EC B6 E0 XCHG AH,AL ; SAVE HIGH BYTE
420 01EE E8 0000 E CALL CMOS_READ ; LOW BYTE
421 ;----- WHICH IS GREATER -- AX = MEMORY SIZE DETERMINE
422 ;----- CX = CONFIGURATION (ABOVE 640) BX = SIZE (BELOW 640)
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 B8 8E8E MOV AX,X*(CMOS_DIAG+NM1) ; ADDRESS THE STATUS BYTE
430 01F9 E8 0000 E CALL CMOS_READ ; GET THE STATUS
431 01FC 0C 10 OR AL,#MEM_SIZE ; SET CMOS FLAG
432 01FE 86 C4 XCHG AL,AH ; 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:
437 0204 3B C8 CMP CX,AX ; IS CONFIG GREATER THAN DETERMINED?
438 0206 77 02 JA SET_MEM ; GO IF YES
439 0208 B8 C8 MOV CX,AX ; USE MEMORY SIZE DETERMINE IF NOT
440 020A SET_MEM:
441 020A 03 D9 ADD BX,CX ; SET TOTAL MEMORY SIZE
442 020C E20C:
443 020C 81 FA 02D1 CMP DX,513 ; CHECK IF BASE MEMORY LESS 512K
444 0210 72 0D JB NO_640 ; GO IF YES
445
446 0212 B8 B2B3 MOV AX,X*(CMOS_INFO128+NM1) ; SET 640K BASE MEMORY BIT
447 0215 E8 0000 E CALL CMOS_READ ; GET THE CURRENT STATUS
448 0218 0C 80 OR AL,M640K ; TURN ON 640K BIT IF NOT ALREADY ON
449 021A B6 C4 XCHG AH,AH ; SAVE THE CURRENT DIAGNOSTIC STATUS
450 021C E8 0000 E CALL CMOS_WRITE ; RESTORE THE STATUS
451 021F NO_640:
452 021F 89 1E 0017 R MOV WORD PTR #KB_FLAG,BX ; SAVE TOTAL SIZE FOR LATER TESTING
453 0223 C1 EB 06 SHR BX,6 ; DIVIDE BY 64
454 0224 4B DEC BX ; IS 64K ALREADY DONE
455 0227 C1 EA 06 SHR DX,6 ; DIVIDE BY 64 FOR BASE
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 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 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 261 C7 06 0048 FFFF             MOV    ES:ES_TEMP_SEG_LIMIT,MAX_SEG_LEN
475 023C 261 C7 06 004A 0000             MOV    ES:ES_TEMP_BASE_LO_WORD,0
476 0243 261 C6 06 004C 00             MOV    BYTE PTR ES:(ES_TEMP_BASE_HI_BYTE),0 ; FIRST 65K
477 0249 261 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 261 C7 06 0060 FFFF             MOV    ES:DS_TEMP_SEG_LIMIT,MAX_SEG_LEN
482 0256 261 C7 06 0062 0000             MOV    ES:DS_TEMP_BASE_LO_WORD,0
483 025D 261 C6 06 0064 00             MOV    BYTE PTR ES:(DS_TEMP_BASE_HI_BYTE),0
484 0263 261 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
495                                     ]----- MEMORY TEST LOOP - POINT TO NEXT BLOCK OF 32K WORDS (64K)
496
497 0273                                     E21:  PUSH  BYTE PTR GDT_PTR               ; MEMORY TEST LOOP
498 0273 6A 08                             POP    DS                             ; POINT TO START OF DESCRIPTOR TABLE
499 0275 1F                                 POP    DS                             ; POINT TO NEXT BLOCK
500 0276 FE 06 0064                       INC    BYTE PTR DS:(DS_TEMP_BASE_HI_BYTE)
501 027A FE 06 004C                       INC    BYTE PTR DS:(ES_TEMP_BASE_HI_BYTE)
502
503                                     ]----- CHECK FOR END OF 256K PLANAR MEMORY
504
505 027E 80 3E 0064 04                   CMP    BYTE PTR DS:(DS_TEMP_BASE_HI_BYTE),04H
506 0283 72 04                             JB     E21_0                          ; GO IF STILL FIRST 256K OF BASE MEMORY
507
508 0285 B0 C0                             MOV    AL,PARITY_CHECK+10_CHECK; CHECK FOR ANY TYPE OF PARITY ERROR
509 0287 E6 87                             OUT    DMA_PAGE+5,AL                  ; AFTER FIRST 256K
510
511                                     ]----- CHECK END OF FIRST 640K OR ABOVE (END OF MAXIMUM BASE MEMORY)
512
513 0289 80 3E 0064 0A                   CMP    BYTE PTR DS:(DS_TEMP_BASE_HI_BYTE),0AH
514 028E 77 16                             JA     NEXT                            ; CONTINUE IF ABOVE 1 MEG
515
516                                     ]----- CHECK FOR END OF BASE MEMORY TO BE TESTED
517
518 0290 59                               POP    CX                              ; GET COUNT
519 0291 5B                               POP    BX                              ; GET COUNT TESTED
520 0292 58                               POP    AX                              ; RECOVER COUNT OF BASE MEMORY BLOCKS
521 0293 50                               PUSH  AX                              ; SAVE BASE COUNT
522 0294 53                               PUSH  BX                              ; SAVE TESTED COUNT
523 0295 51                               PUSH  CX                              ; SAVE TOTAL COUNT
524 0296 38 06 0064                       CMP    BYTE PTR DS:(DS_TEMP_BASE_HI_BYTE),AL ; MAX BASE COUNT
525 029A 72 0A                             JB     NEXT                            ; CONTINUE IF NOT DONE WITH BASE MEMORY
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                                         ; FOR POSSIBLE ERROR
537
538                                     ]----- CHECK FOR TOP OF MEMORY (FE0000) 16 MEG
539
540 02AB 80 3E 004C FE                   CMP    BYTE PTR DS:(ES_TEMP_BASE_HI_BYTE),0FEH ; TOP OF MEMORY?
541 02B0 74 29                             JE     KB_LOOP3                       ; EXIT NEXT TEST IF DONE
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 B0 31                             MOV    AL,31H                         ;
551 02BA E6 80                             OUT    MFG_PORT,AL                    ;
552                                         ;
553                                         ;
554 02BC B9 8000                           MOV    CX,8000H                       ; SET COUNT FOR 32K WORDS
555 02BF E8 0000 E                         CALL  STGTST_CNT                       ;
556 02C2 14 03                             JZ     NI                              ; SKIP IF OK
557 02C4 E9 0367 R                         JMP    E21A                             ; GO PRINT ERROR
558
559 NI:  POP    CX                              ; POP CX TO GET AX
560 02C7 59                               POP    AX                              ; RECOVER TESTED MEMORY
561
562                                     ]----- WRITE THE CURRENT SIZE FOR (ADDRESS LINE 23-17 TEST) USED LATER
563
564 02C9 2B FF                             SUB    DI,DI                           ; POINT TO BEGINNING OR A BLOCK
565 02CB AB                                 STOSW                                  ; WRITE THE CURRENT SIZE
566                                         ; AT THE STARTING ADDRESS
567 02CC 05 0040                           ADD    AX,64                           ; ADVANCE COUNT TO NEXT BLOCK
568 02CF 50                               PUSH  AX
569 02D0 51                               PUSH  CX
570 02D1 E8 099F R                         CALL  PRT_OK                           ; 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:          ;
579 02DB 58          POP     AX          ; CLEAR MAXIMUM BLOCK COUNT
580 02DC 58          POP     AX          ; CLEAR BASE SIZE COUNT FROM STACK
581                    ; ADDRESS TEST VALUES ARE IN STACK
582                    ;
583 02DD              I----- ADDRESS LINE 16-23 TEST
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                    ; ALLOW SIX DISPLAY REFRESH CYCLES
587                    ;
588 02E2              I----- INITIALIZE DS DESCRIPTOR
589 02E3 6A 08      PUSH   BYTE PTR GD_TPTR
590 02E5 07         POP     ES          ;
591 02E6 261 C6 06 0064 00 MOV     BYTE PTR ES:(DS_TEMP.BASE_HI_BYTE),0
592 02EC 261 C7 06 0062 0000 MOV     ES:DS_TEMP.BASE_LO_WORD,0
593                    ;
594 02ED              I----- 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     ; SET HIGH BYTE OF SEGMENT WORD
600 02FB E6 84      OUT    DMA_PAGE+3,AL ; HIGH BYTE OF SEGMENT
601                    ;
602 02FD              I----- 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 ;
607 0301 261 80 06 0064 01 ADD     BYTE PTR ES:(DS_TEMP.BASE_HI_BYTE),01
608                    ;
609 0302              I----- CHECK FOR END OF BASE MEMORY TO BE TESTED
610                    ;
611 0303 261 80 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 5B          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 261 3B 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 031C              I----- DO ADDITIONAL STORAGE ABOVE 1 MEG
624                    ;
625 031C          NEXT_A2:
626 031C 261 C6 06 0064 10 MOV     BYTE PTR ES:(DS_TEMP.BASE_HI_BYTE),10H
627 0322              NEXT_A:
628 0322 261 A0 0064      MOV     AL,BYTE PTR ES:(DS_TEMP.BASE_HI_BYTE)
629                    ;
630 0323              I----- DMA PAGE REGISTERS 3
631                    ;
632 0326 E6 84      OUT    DMA_PAGE+3,AL ; SAVE THE HIGH BYTE OF SEGMENT
633                    ; FOR POSSIBLE ERROR
634                    ;
635 0328              I----- CHECK FOR TOP OF MEMORY (FE0000) 16 MEG
636                    ;
637 032A 3C FE      CMP     AL,0FEH    ; TOP OF MEMORY?
638 032A 74 34      JZ     KB_LOOP_3   ; GO NEXT TEST IF IT IS
639                    ;
640 032C              I----- SET DS REGISTER
641                    ;
642 032C 6A 60      PUSH   BYTE PTR DS_TEMP
643 032E 1F         POP     DS          ;
644 032F 2B FF      SUB     DI,DI      ; POINT TO START OF BLOCK
645 0331 8B 15      MOV     DX,DS:[DI] ; GET THE VALUE OF THIS BLOCK
646 0333 8B F7      MOV     SI,DI      ; SET SI FOR POSSIBLE ERROR
647 0335 2B C0      SUB     AX,AX      ; CLEAR MEMORY LOCATION
648 0337 89 05      MOV     [DI],AX
649                    ;
650 0339              I----- ALLOW DISPLAY TIME TO DISPLAY MESSAGE AND REFRESH TO RUN
651                    ;
652 0339 B9 1A69     MOV     CX,6761    ; COUNT FOR 102 MS FIXED TIME DELAY
653 033C E8 0000 E   CALL    WAITF      ; ALLOW FIVE DISPLAY REFRESH CYCLES
654 033F 59          POP     CX          ; GET THE LOOP COUNT
655 0340 58          POP     AX          ; RECOVER TESTED MEMORY
656 0341 50          PUSH   AX          ; SAVE TESTED MEMORY
657 0342 51          PUSH   CX          ; SAVE LOOP COUNT
658 0343 3B C2      CMP     AX,DX      ; DOES THE BLOCK ID MATCH
659 0345 8B C2      MOV     CX,DX      ; GET THE BLOCK ID FOR POSSIBLE ERROR
660 0347 75 1E      JNZ    E21A        ; GO PRINT ERROR
661                    ;
662 0349              I----- CHECK FOR CHECK PARITY
663                    ;
664 0349 E4 61      IN     AL,PORT_B   ; CHECK FOR I/O OR PARITY CHECK
665 034B 24 C0      AND    AL,PARITY_ERR
666 034D 75 18      JNZ    E21A        ; STRIP UNWANTED BITS
667                    ; EXIT IF PARITY ERROR
668                    ;
669 034F 59          POP     CX          ; POP CX TO GET AX
670 0350 58          POP     AX          ; RECOVER TESTED MEMORY
671 0351 05 0040     ADD    AX,64       ; 64K INCREMENTS
672 0353 50          PUSH   AX          ; SAVE TESTED MEMORY
673 0355 51          PUSH   CX          ; SAVE LOOP COUNT
674 0356 E8 09FF R   CALL    PRN_OK     ; DISPLAY OK MESSAGE
675 0359 59          POP     CX          ; RECOVER 64K BLOCK COUNT
676 035A 49          DEC     CX          ; LOOP TILL ALL MEMORY CHECKED
677 035B E3 03      JCXZ   KB_LOOP_3   ; CONTINUE
678                    ;
679 035D 51          PUSH   CX          ; SAVE LOOP COUNT
680 035E EB 9D      JMP     E21_A      ; CONTINUE TILL DONE
681                    ;
682 0360              I----- BACK TO REAL MODE - MEMORY TESTS DONE
683                    ;
684 0360          KB_LOOP_3:
685 0360 34 B0 34      MOV     AL,34H     ;
686 0362 E6 80      OUT    MFG_PORT,AL ;
687                    ;

```

SECTION 5

```

685                                     ; BACK TO REAL MODE
686 0364 E9 0000 E                       JMP     PROC_SHUTDOWN                ; NEXT TEST VIA JUMP TABLE (SHUT2)
687
688
689 ;----- PRINT FAILING ADDRESS AND XOR'ED PATTERN IF DATA COMPARE ERROR
690 ;----- USE DMA PAGE REGISTERS AS TEMPORARY SAVE AREA FOR ERROR
691 ; SET SHUTDOWN 3
692
693 E21A: OUT     DMA_PAGE+1,AL             ; SAVE FAILING BIT PATTERN (LOW BYTE)
694        MOV     AL,AH                  ; SAVE HIGH BYTE
695        OUT     DMA_PAGE+2,AL
696        MOV     AX,S1                  ; GET THE FAILING OFFSET
697        OUT     DMA_PAGE+5,AL
698        XCHG   AH,AL
699        OUT     DMA_PAGE+4,AL
700
701 ;----- CLEAR I/O CHANNEL CHECK OR R/W PARITY CHECK
702
703 0375 2B F6                             SUB     S1,S1                        ; WRITE TO FAILING BLOCK
704 0377 AB                             STOSW
705 0378 E4 61                             IN      AL,PORT_B                   ; GET PARITY CHECK LATCHES
706 037A E6 88                             OUT     DMA_PAGE+7,AL               ; SAVE FOR ERROR HANDLER
707 037C 0C 0C                             OR      AL,RAM_PAR_OFF              ; TOGGLE I/O-PARITY CHECK ENABLE
708 037E E6 61                             OUT     PORT_B,AL                   ; TO RESET CHECKS
709 0380 24 F3                             AND     AL,RAM_PAR_ON
710 0382 E6 61                             OUT     PORT_B,AL
711
712 ;----- GET THE LAST OF GOOD MEMORY
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                       ; COMPARE LAST GOOD MEMORY WITH BASE
719 038C 73 17                             JAE    E21I                         ; IF ABOVE OR EQUAL, USE REMAINDER IN
720                                     ; CMOS_U_M_S_(H/L)
721
722 ;----- ELSE SET BASE MEMORY SIZE
723 038E 6A 18                             PUSH   BYTE PTR RSDA_PTR           ; SET THE DATA SEGMENT
724 0390 1F                             POP
725                                     ; IN PROTECTED MODE
726 0391 03 C3                             ADD     AX,BX                       ; CONVERT BACK TO LAST WORKING MEMORY
727 0393 A3 0013 R                         MOV     MEMOY_SIZE,AX              ; TO INDICATE HOW MUCH MEMORY WORKING
728
729 ;----- RESET 512K --> 640K OPTION IF SET
730
731 0396 B8 B3B3                          MOV     AX,X*(CMOS_INFO128+NM1)    ; ADDRESS OPTIONS INFORMATION BYTE
732 0399 E8 0000 E                         CALL   CMOS_READ                   ; READ THE MEMORY INFORMATION FLAG
733 039C 24 FF                             AND     AL,NOT M640K                ; SET 640K OPTION OFF
734 039E 86 C4                             XCHG   AL,AH                       ; MOVE TO WORK REGISTER
735 03A0 E8 0000 E                         CALL   CMOS_WRITE                   ; UPDATE STATUS IF IT WAS ON
736 03A3 33 C0                             XOR     AX,AX                       ; CLEAR VALUE FOR EXTENSION MEMORY
737 03A5
738 03A5 8B C8                             MOV     CX,AX                      ; SAVE ADJUSTED MEMORY SIZE
739 03A7 B0 B1                             MOV     AL,CMOS_U_M_S_HI+NM1       ; SAVE THE HIGH BYTE MEMORY SIZE
740 03A9 E8 0000 E                         CALL   CMOS_WRITE                   ; SAVE THE HIGH BYTE MEMORY SIZE
741 03AC 8A E1                             MOV     AH,CL                       ; GET THE LOW BYTE
742 03AE B0 B0                             MOV     AL,CMOS_U_M_S_LO+NM1      ; DO THE LOW BYTE
743 03B0 E8 0000 E                         CALL   CMOS_WRITE                   ; WRITE IT
744
745 ;----- SET SHUTDOWN 3
746
747 03B3 B8 03BF                          MOV     AX,3*H+CMOS_SHUT_DOWN+NM1 ; ADDRESS FOR SHUTDOWN RETURN
748 03B6 E8 0000 E                         CALL   CMOS_WRITE                   ; SET RETURN 3
749
750 ;----- SHUTDOWN
751
752 03B9 E9 0000 E                       JMP     PROC_SHUTDOWN
    
```

```

PAGE
-----
753 ; MEMORY ERROR REPORTING (R/W/ MEMORY OR PARITY ERRORS)
754 ;
755 ;
756 ; DESCRIPTION FOR ERRORS 201 (CMP ERROR OR PARITY)
757 ; OR 202 (ADDRESS LINE 0-15 ERROR)
758 ;
759 ;
760 ; *AABBCC DDEE 201* (OR 202)
761 ; AA=HIGH BYTE OF 24 BIT ADDRESS
762 ; BB=MIDDLE BYTE OF 24 BIT ADDRESS
763 ; CC=LOW BYTE OF 24 BIT ADDRESS
764 ; DD=HIGH BYTE OF XOR FAILING BIT PATTERN
765 ; EE=LOW BYTE OF XOR FAILING BIT PATTERN
766 ;
767 ; DESCRIPTION FOR ERROR 202 (ADDRESS LINE 00-15)
768 ; A WORD OF FFFF IS WRITTEN AT THE FIRST WORD AND LAST WORD
769 ; OF EACH 64K BLOCK WITH ZEROS AT ALL OTHER LOCATIONS OF THE
770 ; BLOCK. A SCAN OF THE BLOCK IS MADE TO INSURE ADDRESS LINE
771 ; 0-15 ARE FUNCTIONING.
772 ;
773 ; DESCRIPTION FOR ERROR 203 (ADDRESS LINE 16-23)
774 ; AT THE LAST PASS OF THE STORAGE TEST, FOR EACH BLOCK OF
775 ; 64K, THE CURRENT STORAGE SIZE (10) IS WRITTEN AT THE FIRST
776 ; WORD OF EACH BLOCK. IT IS USED TO FIND ADDRESSING FAILURES.
777 ;
778 ; *AABBCC DDEE 203* SAME AS ABOVE EXCEPT FOR DDEE
779 ;
780 ;
781 ; GENERAL DESCRIPTION FOR BLOCK ID (DDEE WILL NOW CONTAINED THE ID)
782 ; DD=HIGH BYTE OF BLOCK ID
783 ; EE=LOW BYTE OF BLOCK ID
784 ;
785 ; BLOCK ID ADDRESS RANGE
786 ; 0000 000000 --> 00FFFF
787 ; 0040 010000 --> 01FFFF
788 ; //
789 ; 0200 090000 --> 09FFFF (512->576K) IF 640K BASE
790 ; 100000 --> 10FFFF (1024->1088K) IF 512K BASE
791 ;
792 ; EXAMPLE (640K BASE MEMORY + 512K I/O MEMORY = 1152K TOTAL)
793 ; NOTE: THE CORRECT BLOCK ID FOR THIS FAILURE IS 0280 HEX.
794 ; DUE TO AN ADDRESS FAILURE THE BLOCK ID=128K OVERLAYED
795 ; THE CORRECT BLOCK ID.
796 ;
797 ; 00640K OK <-- LAST OK MEMORY
798 ; 10000 0300 202 <-- ERROR DUE TO ADDRESS FAILURE
799 ;
800 ; IF A PARITY LATCH WAS SET THE CORRESPONDING MESSAGE WILL DISPLAY.
801 ;
802 ; *PARITY CHECK 1* (OR 2)
803 ;
804 ; DMA PAGE REGISTERS ARE USED AS TEMPORARY SAVE AREAS FOR SEGMENT
805 ; DESCRIPTOR VALUES.
806 ;-----
807 ;
808 03BC SHUT3: ; ENTRY FROM PROCESSOR SHUTDOWN 3
809 03BC EB 0000 E ; SET REAL MODE DATA SEGMENT
810 ;
811 ;
812 03BF C6 06 0016 R 01 ; <<<> MEMORY FAILED <<>>
813 03C4 B0 0D ; CLEAR AND SET MANUFACTURING ERROR FLAG
814 03C6 E8 0000 E ; CARRIAGE RETURN
815 03C9 B0 0A ; LINE FEED
816 03CB EB 0000 E ;
817 03CE E4 84 ; IN AL,DMA_PAGE+3 ; GET THE HIGH BYTE OF 24 BIT ADDRESS
818 03D0 EB 0000 E ; CALL XPC_BYTE ; CONVERT AND PRINT CODE
819 03D3 E4 85 ; IN AL,DMA_PAGE+4 ; GET THE MIDDLE BYTE OF 24 BIT ADDRESS
820 03D5 EB 0000 E ; CALL XPC_BYTE
821 03D8 E4 86 ; IN AL,DMA_PAGE+5 ; GET THE LOW BYTE OF 24 BIT ADDRESS
822 03DA EB 0000 E ; CALL XPC_BYTE
823 03DD B0 20 ; MOV AL,' ' ; SPACE TO MESSAGE
824 03DF EB 0000 E ; CALL PRT_HEX
825 03E2 E4 83 ; IN AL,DMA_PAGE+2 ; GET HIGH BYTE FAILING BIT PATTERN
826 03E4 EB 0000 E ; CALL XPC_BYTE ; CONVERT AND PRINT CODE
827 03E7 E4 82 ; IN AL,DMA_PAGE+1 ; GET LOW BYTE FAILING BIT PATTERN
828 03E9 EB 0000 E ; CALL XPC_BYTE ; CONVERT AND PRINT CODE
829 ;
830 ;----- CHECK FOR ADDRESS ERROR
831 ;
832 03EC E4 80 ; IN AL,MFG_PORT ; GET THE CHECKPOINT
833 03EE 3C 33 ; CMP AL,33H ; IS IT AN ADDRESS FAILURE?
834 03F0 FE 0000 E ; MOV SI,OFFSET E203 ; LOAD ADDRESS ERROR 16->23
835 03F3 74 0A ; JZ ERR2 ; GO IF YES
836 ;
837 03F5 FE 0000 E ; MOV SI,OFFSET E202 ; LOAD ADDRESS ERROR 00->15
838 03F8 3C 32 ; CMP AL,32H ; GO IF YES
839 03FA 74 03 ; JZ ERR2
840 ;
841 03FC FE 0000 E ; MOV SI,OFFSET E201 ; SETUP ADDRESS OF ERROR MESSAGE
842 03FF ; ;
843 03FF EB 0000 E ; CALL EMSG ; PRINT ERROR MESSAGE
844 0402 E4 88 ; IN AL,DMA_PAGE+7 ; GET THE PORT_B VALUE
845 ;
846 ;----- DISPLAY "PARITY CHECK ?" ERROR MESSAGES
847 ;
848 0404 A8 80 ; TEST AL,PARITY_CHECK ; CHECK FOR PLANAR ERROR
849 0406 74 0B ; JZ ; SKIP IF NOT
850 ;
851 0408 50 ; PUSH AX ; SAVE STATUS
852 0409 EB 09BF R ; CALL PADING ; INSERT BLANKS
853 040C BE 0000 E ; MOV SI,OFFSET D1 ; PLANAR ERROR, ADDRESS "PARITY CHECK 1"
854 040F EB 0000 E ; CALL P_MSG ; DISPLAY "PARITY CHECK 1" MESSAGE
855 0412 58 ; POP AX ; AND RECOVER STATUS
856 0413 ;
857 0413 A8 40 ; TEST AL,I/O_CHECK ; I/O PARITY CHECK ?
858 0415 74 09 ; JZ ; SKIP IF CORRECT ERROR DISPLAYED
859 ;
860 0417 EB 09BF R ; CALL PADING ; INSERT BLANKS
861 041A BE 0000 E ; MOV SI,OFFSET D2 ; ADDRESS OF "PARITY CHECK 2" MESSAGE
862 041D EB 0000 E ; CALL P_MSG ; DISPLAY "PARITY CHECK 2" ERROR
863 0420 ;
864 ; NMI_M2: ; CONTINUE TESTING SYSTEM ....

```

SECTION 5


```

1093 054A BE 0000 E          MOV     SI,OFFSET SLAVE_VECTOR_TABLE
1094 054D BF 01C0 R          MOVSW  D1,OFFSET @SLAVE_INT_PTR
1095 0550 A5                  FTA1:  INC     DS,AX
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:ABS0
1103 0555 2B C0              SUB     AX,AX
1104 0557 8E D8              MOV     DS,AX
1105 0559 C7 05 000B R 0000 E MOV     WORD PTR @NMI_PTR,OFFSET NMI_INT ; NMI INTERRUPT
1106 055F C7 06 0014 R 0000 E MOV     WORD PTR @INT5_PTR,OFFSET PRINT_SCREEN ; PRINT SCREEN
1107 0565 C7 06 0062 R F600 E MOV     WORD PTR @BASIC_PTR+2,OF600H ; SEGMENT FOR CASSETTE BASIC
1108
1109                          ;----- ZERO RESERVED VECTORS
1110
1111 056B BF 0180              MOV     DI,60H*4
1112 056E B9 0010              MOV     CX,16
1113 0571 C7 05 0000          FTA2:  MOV     WORD PTR DS:[DI],0
1114 0575 83 C7 02              ADD     DI,2
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   D05
1121
1122 057D F6 06 0012 R 20      TEST   @MFG_TST,MFG_LOOP
1123 0582 75 0B              JNZ   F9
1124 0584 26: C7 06 0020 R 0000 E MOV     WORD PTR ES:@INT_PTR,OFFSET BLINK_INT ; SETUP TIMER TO BLINK LED
1125 058B B0 FE              MOV     AL,0FEH
1126 058D E6 21              OUT    INTA01,AL
1127 058F FB              STI
1128
1129                          ;----- ISSUE A RESET TO THE HARD FILE IF SOFT RESET
1130
1131 0590 81 3E 0072 R 1234H  CMP     @RESET_FLAG,1234H
1132 0596 75 0E              JNZ   F9A
1133 0598 B9 00FF              MOV     CX,OFFH
1134 059B BA 03F6              MOV     DX,03F6H
1135 059E B0 04              MOV     AL,04H
1136 05A0 EE              OUT    DX,AL
1137 05A1 E2 FE              F9_A:  LOOP   F9_A
1138 05A3 2A CD              SUB     AL,AL
1139 05A5 EE              OUT    DX,AL
1140
1141                          ;-----
1142                          ; TEST_23
1143                          ; DISKETTE ATTACHMENT TEST
1144                          ; DESCRIPTION
1145                          ; CHECK IF IPL DISKETTE DRIVE IS ATTACHED TO SYSTEM. IF
1146                          ; ATTACHED, VERIFY STATUS OF NEC FDC AFTER A RESET. ISSUE
1147                          ; A RECALIBRATE AND SEEK COMMAND TO FDC AND CHECK STATUS.
1148                          ; COMPLETE SYSTEM INITIALIZATION THEN PASS CONTROL TO THE
1149                          ; BOOT LOADER PROGRAM.
1150                          ;-----
1151
1152 05A6 B0 3C              F9A:   MOV     AL,3CH
1153 05A8 BE 80              OUT    MFG_PORT,AL
1154
1155 05AA B0 02              MOV     AL,02H
1156 05AC BA 03F7              MOV     DX,3F7H
1157 05AF EE              OUT    DX,AL
1158 05B0 F6 06 0010 R 01      TEST   BYTE PTR @EQUIP_FLAG,1H ; DISKETTE PRESENT?
1159 05B5 74 55              JZ    F15
1160 05B7 F6 06 0012 R 20      TEST   @MFG_TST,MFG_LOOP
1161 05BC 74 4E              JZ    F15
1162 05BE
1163 05BE E4 21              F10:   IN     AL,INTA01
1164 05C0 EB 00              JMP    $+2
1165 05C2 24 BF              AND    AL,0BFH
1166 05C4 E6 21              OUT    INTA01,AL
1167 05C6 B4 00              MOV     AH,0
1168 05C8 8A D4              MOV     DL,AH
1169 05CA CD 13              INT    13H
1170 05CC F6 C4 FF              TEST   AH,OFFH
1171 05CF 75 25              JNZ   F13
1172
1173                          ;----- TURN DRIVE 0 MOTOR ON
1174
1175 05D1 BA 03F2              MOV     DX,03F2H
1176 05D4 B0 1C              MOV     AL,0CH
1177 05D6 EE              OUT    DX,AL
1178 05D7 2B C9              SUB     CX,CX
1179 05D9 E8 0000 E          CALL   WAITF
1180
1181 05DC 33 FF              XOR    DI,DI
1182 05DE B5 01              MOV     CH,1
1183 05E0 C6 06 003E R 00      MOV     @SEEK_STATUS,0
1184 05E5 B0 0E 00A0 R 01      OR     @RTC_WAIT_FLAG,01
1185 05EA E8 0000 E          CALL   SEEK
1186 05ED 72 07              JC     F13
1187 05EF B5 22              MOV     CH,34
1188 05F1 E8 0000 E          CALL   SEEK
1189 05F4 73 0B              JNC   F14
1190 05F6
1191 05F6 80 0E 0016 R 40      F13:   OR     @MFG_ERR_FLAG+1,DSK_FAIL
1192
1193 05FB BE 0000 E          MOV     SI,OFFSET E601
1194 05FE E8 0000 E          CALL   E_MSG
1195
1196                          ;----- TURN DRIVE 0 MOTOR OFF
1197
1198 0601
1199 0601 80 26 00A0 R FE      F14:   AND    @RTC_WAIT_FLAG,0FEH
1200 0606 B0 0C              MOV     AL,0CH
1201 0608 BA 03F2              MOV     DX,03F2H
1202 060B EE              OUT    DX,AL
1203
1204                          ;----- SETUP KEYBOARD PARAMETERS
1205
1206 060C C6 06 006B R 00      F15:   MOV     @INTR_FLAG,00H

```



```

1207 0611 BE 001E R      MOV     SI,OFFSET @KB_BUFFER      ; SETUP KEYBOARD PARAMETERS
1208 0614 89 36 001A R  MOV     @BUFFER_HEAD,SI
1209 0618 89 36 001C R  MOV     @BUFFER_TAIL,SI
1210 061C 89 36 0080 R  MOV     @BUFFER_START,SI
1211 0620 83 C6 20      ADD     SI,32                      ; DEFAULT BUFFER OF 32 BYTES
1212 0623 89 36 0082 R  MOV     @BUFFER_END,SI
1213
1214
1215
1216 0627 BF 0078 R      I----- SET PRINTER TIMEOUT DEFAULT
1217 062A IE            MOV     DI,OFFSET @PRINT_TIM_OUT; SET DEFAULT PRINTER TIMEOUT
1218 062B 07           PUSH   DS
1219 062C BB 1414      POP     ES                      ; DEFAULT=20
1220 062F AB           MOV     AX,1414H
1221 0630 AB           STOSW
1222
1223
1224
1225 0631 BB 0101      I----- SET RS232 DEFAULT
1226 0634 AB           MOV     AX,0101H                ; RS232 DEFAULT=01
1227 0635 AB           STOSW
1228
1229
1230
1231 0636 E4 21      I----- ENABLE TIMER INTERRUPTS
1232 0638 24 FE      IN     AL,INTA01                ; ENABLE TIMER INTERRUPTS
1233 063A EB 00      AND   AL,0FEH                  ; I/O DELAY
1234 063C E6 21      JMP   $+2
1235
1236
1237
1238 063E F6 06 0012 R 20 I----- CHECK CMOS BATTERY AND CHECKSUM
1239 0643 75 03      TEST  @MFG_TST,MFG_LOOP        ; MFG_JUMPER?
1240 0645 E9 072E R  JNZ   BI_OK                    ; GO IF NOT
1241 0648             JMP   F15C                      ; BYPASS IF YES
1242 064B B0 BE      BI_OK: MOV   AL,CMOS_DIAG+NMI        ; ADDRESS DIAGNOSTIC STATUS BYTE
1243 064A EB 0000 E  CALL  CMOS_READ                ; READ IT FROM CMOS
1244
1245 064D BE 0000 E    MOV   SI,OFFSET E161          ; LOAD BAD BATTERY MESSAGE 161
1246 0650 AB 80      TEST  AL,BAD_BAT              ; BATTERY BAD?
1247 0652 75 07      JNZ   BI_ER                    ; DISPLAY ERROR IF BAD
1248
1249 0654 BE 0000 E    MOV   SI,OFFSET E162          ; LOAD CHECKSUM BAD MESSAGE 162
1250 0657 AB 60      TEST  AL,BAD_CHKSUM+BAD_CONFIG ; CHECK FOR CHECKSUM OR NO DISKETTE
1251 0659 74 09      JZ    C_OK                     ; SKIP AND CONTINUE TESTING CMOS CLOCK
1252 065B             BI_ER: CALL  E_MSG                     ; ELSE DISPLAY ERROR MESSAGE
1253 065B EB 0000 E  OR    BP,08000H              ; CHECK FOR "SET SYSTEM OPTIONS" DISPLAYED
1254 065E B1 CD 8000 ; SHORT H_OK1A                ; SKIP CLOCK TESTING IF ERROR
1255 0662 EB 45
1256
1257
1258
1259 0664 B3 04      I----- TEST CLOCK UPDATING
1260 0666 2B C9      C_OK: MOV   BL,04H              ; OUTER LOOP COUNT
1261 0668 B0 8A      D_OK: SUB   CX,CX              ; INNER LOOP COUNT
1262 066A EB 0000 E  E_OK: MOV   AL,CMOS_REG_A+NMI  ; GET THE CLOCK UPDATE BYTE
1263 066D AB 80      CALL  CMOS_READ                ; CHECK FOR UPDATE IN PROGRESS
1264 066F 75 1B      TEST  AL,80H                  ; GO IF YES
1265 0671 E2 F5      JNZ   C_OK                     ; TRY AGAIN
1266 0673 FE CB      DEC   BL                      ; DEC OUTER LOOP
1267 0675 75 EF      JNZ   D_OK                     ; TRY AGAIN
1268 0677 BE 0000 E  F_OK: MOV   SI,OFFSET E163      ; PRINT MESSAGE
1269 067A EB 0000 E  CALL  E_MSG
1270
1271
1272
1273 067D B8 0E8E      I----- SET CMOS DIAGNOSTIC STATUS TO 04 (CLOCK ERROR)
1274 0680 EB 0000 E  MOV   AX,X*CMOS_DIAG+NMI      ; SET CLOCK ERROR
1275 0683 0C 04      CALL  CMOS_READ                ; GET THE CURRENT STATUS
1276 0685 B6 C4      OR    AL,CMOS_CLK_FAIL        ; SET NEW STATUS
1277 0687 EB 0000 E  AL,4H                          ; XCHG
1278 068A EB 0E      CALL  CMOS_WRITE              ; MOVE NEW DIAGNOSTIC STATUS TO CMOS
1279
1280
1281
1282 068C B9 0320      I----- CHECK CLOCK UPDATE
1283 068F B0 8A      G_OK: MOV   CX,800              ; LOOP COUNT
1284 0691 EB 0000 E  I_OK: MOV   AL,CMOS_REG_A+NMI  ; CHECK FOR OPPOSITE STATE
1285 0694 AB 80      CALL  CMOS_READ
1286 0696 E0 F7      TEST  AL,80H                  ; TRY AGAIN
1287 0698 E3 DD      LOOPNZ I_OK                    ; PRINT ERROR IF TIMEOUT
1288
1289
1290
1291 069A             I----- CHECK MEMORY SIZE DETERMINED = CONFIGURATION
1292 069A B0 BE      H_OK: MOV   AL,CMOS_DIAG+NMI      ; GET THE STATUS BYTE
1293 069C EB 0000 E  CALL  CMOS_READ
1294 069F AB 10      TEST  AL,W_MEM_SIZE           ; WAS THE CONFIG= MEM_SIZE_DETERMINED?
1295 06A1 74 06      JZ    H_OK                     ; GO IF YES
1296
1297
1298
1299 06A3 BE 0000 E    I----- MEMORY SIZE ERROR
1300 06A6 EB 0000 E  MOV   SI,OFFSET E164          ; PRINT SIZE ERROR
1301
1302
1303
1304 06A9 80 3E 0015 R 30 I----- CHECK FOR CRT ADAPTER ERROR
1305 06AE BE 0000 E  H_OK1A: CMP  @MFG_ERR_FLAG,0CH      ; CHECK FOR MONOCHROME CRT ERROR
1306 06B1 74 0A      MOV   SI,OFFSET E401          ; LOAD MONOCHROME CRT ERROR
1307
1308 06B3 80 3E 0015 R 30 CMP  @MFG_ERR_FLAG,0DH        ; CHECK FOR COLOR CRT ADAPTER ERROR
1309 06B5 75 06      JNZ   J_OK                     ; CONTINUE IF NOT
1310 06B8 BA 0000 E  MOV   ST,OFFSET E501          ; CRT ADAPTER ERROR MESSAGE
1311 06BD             H_OK1B: CALL  E_MSG
1312 06BD EB 0000 E
1313
1314
1315
1316 06C0             I----- CHECK FOR MULTIPLE DATA RATE CAPABILITY
1317 06C0 BA 03F1      J_OK: MOV   DX,03F1H           ; D/S/P DIAGNOSTIC REGISTER
1318 06C3 EC          IN     AL,DX                   ; READ D/S/P TYPE CODE
1319 06C4 24 F8      AND   AL,11111000B           ; KEEP ONLY UNIQUE CODE FOR D/S/P
1320 06C6 3C 50      CMP   AL,01010000B           ; D/S/P CARD - MULTIPLE DATA RATE ?
    
```



```

1          PAGE 118,121
2          TITLE TEST4 ---- 06/10/85 POST AND BIOS UTILITY ROUTINES
3          .LST6C
4          .LIST
5          0000
6          CODE          SEGMENT BYTE PUBLIC
7
8          PUBLIC BEEP
9          PUBLIC BLINK_INT
10         PUBLIC CMOS_READ
11         PUBLIC CMOS_WRITE
12         PUBLIC CONFIG_BAD
13         PUBLIC D11
14         PUBLIC DDIS
15         PUBLIC DUMMY_RETURN_1
16         PUBLIC ERR_BEEP
17         PUBLIC E_MSG
18         PUBLIC INT_287
19         PUBLIC KBD_RESET
20         PUBLIC POST4
21         PUBLIC PROT_PRT_HEX
22         PUBLIC PROC_SHUTDOWN
23         PUBLIC PRT_HEX
24         PUBLIC PRT_SEG
25         PUBLIC P_WAIT
26         PUBLIC RE_DIRECT
27         PUBLIC ROM_CHECK
28         PUBLIC ROM_CHECKSUM
29         PUBLIC SET_TOD
30         PUBLIC WAITF
31         PUBLIC XPC_BYTE
32
33         EXTRN E163:NEAR
34         EXTRN OBF_42:NEAR
35         EXTRN ROM_ERR:NEAR
36         EXTRN XMIT_8042:NEAR
37
38         ASSUME CS:CODE,DS:DATA
39
40         0000
41         POST4:
42         :--- CMOS_READ
43         :-----
44         : READ BYTE FROM CMOS SYSTEM CLOCK CONFIGURATION TABLE
45         :
46         : INPUT: (AL) = CMOS TABLE ADDRESS TO BE READ
47         : BIT 7 = 0 FOR NMI ENABLED AND 1 FOR NMI DISABLED ON EXIT
48         : BITS 6-0 = ADDRESS OF TABLE LOCATION TO READ
49         :
50         : OUTPUT: (AL) VALUE AT LOCATION (AL) MOVED INTO (AL). IF BIT 7 OF (AL) WAS
51         : ON THEN NMI LEFT DISABLED DURING THE CMOS READ BOTH NMI AND
52         : NORMAL INTERRUPTS ARE DISABLED TO PROTECT CMOS DATA INTEGRITY.
53         : THE CMOS ADDRESS REGISTER IS POINTED TO A DEFAULT VALUE AND
54         : THE INTERRUPT FLAG RESTORED TO THE ENTRY STATE ON RETURN.
55         : ONLY THE (AL) REGISTER AND THE NMI STATE IS CHANGED.
56         :-----
57         :
58         CMOS_READ PROC NEAR
59         : SAVE READ LOCATION (AL) INTO (AH)
60         PUSHF
61         ROL AL,1
62         : MOVE NMI BIT TO LOW POSITION
63         STC
64         RCR AL,1
65         : FORCE NMI BIT ON IN CARRY FLAG
66         CLI
67         : DISABLE INTERRUPTS
68         OUT CMOS_PORT,AL
69         : ADDRESS LOCATION AND DISABLE NMI
70         NOP
71         : I/O DELAY
72         IN AL,CMOS_DATA
73         : READ THE REQUESTED CMOS LOCATION
74         PUSH AX
75         : SAVE (AH) REGISTER VALUE AND CMOS BYTE
76         MOV AL,CMOS_REG_D*2
77         : GET ADDRESS OF DEFAULT LOCATION
78         RCR AL,1
79         : PUT ORIGINAL NMI MASK BIT INTO ADDRESS
80         OUT CMOS_PORT,AL
81         : SET DEFAULT TO READ ONLY REGISTER
82         POP AX
83         : RESTORE (AH) AND (AL) = CMOS BYTE
84         PUSH CS
85         : *PLACE CODE SEGMENT IN STACK AND
86         CALL CMOS_POFF
87         : *HANDLE PPOFF FOR B- LEVEL 80286
88         RET
89         : RETURN WITH FLAGS RESTORED
90
91         CMOS_READ ENDP
92
93         CMOS_POFF PROC NEAR
94         : PPOFF FOR LEVEL B- PARTS
95         IRET
96         : RETURN FAR AND RESTORE FLAGS
97
98         CMOS_POFF ENDP
99
100        :--- CMOS_WRITE
101        :-----
102        : WRITE BYTE TO CMOS SYSTEM CLOCK CONFIGURATION TABLE
103        :
104        : INPUT: (AL) = CMOS TABLE ADDRESS TO BE WRITTEN TO
105        : BIT 7 = 0 FOR NMI ENABLED AND 1 FOR NMI DISABLED ON EXIT
106        : BITS 6-0 = ADDRESS OF TABLE LOCATION TO WRITE
107        : (AH) = NEW VALUE TO BE PLACED IN THE ADDRESSED TABLE LOCATION
108        :
109        : OUTPUT: VALUE IN (AH) PLACED IN LOCATION (AL) WITH NMI LEFT DISABLED
110        : IF (BIT 7 OF (AL)) IS ON. DURING THE CMOS UPDATE BOTH NMI AND
111        : NORMAL INTERRUPTS ARE DISABLED TO PROTECT CMOS DATA INTEGRITY.
112        : THE CMOS ADDRESS REGISTER IS POINTED TO A DEFAULT VALUE AND
113        : THE INTERRUPT FLAG RESTORED TO THE ENTRY STATE ON RETURN.
114        : ONLY THE CMOS LOCATION AND THE NMI STATE IS CHANGED.
115        :-----
116        :
117        CMOS_WRITE PROC NEAR
118        : WRITE (AH) TO LOCATION (AL)
119        PUSHF
120        AX
121        : SAVE INTERRUPT ENABLE STATUS AND FLAGS
122        ROL AL,1
123        : SAVE WORK REGISTER VALUES
124        MOV AL,1
125        : MOVE NMI BIT TO LOW POSITION
126        STC
127        RCR AL,1
128        : FORCE NMI BIT ON IN CARRY FLAG
129        CLI
130        : DISABLE INTERRUPTS
131        OUT CMOS_PORT,AL
132        : ADDRESS LOCATION AND DISABLE NMI
133        MOV AL,AH
134        : PLACE IN REQUESTED CMOS LOCATION
135        OUT CMOS_DATA,AL
136        : GET ADDRESS OF DEFAULT LOCATION
137        MOV AL,CMOS_REG_D*2
138        : GET ADDRESS OF DEFAULT LOCATION
139        RCR AL,1
140        : PUT ORIGINAL NMI MASK BIT INTO ADDRESS
141        OUT CMOS_PORT,AL
142        : SET DEFAULT TO READ ONLY REGISTER
143        POP AX
144        : RESTORE WORK REGISTERS
145        PUSH CS
146        : *PLACE CODE SEGMENT IN STACK AND
147        CALL CMOS_POFF
148        : *HANDLE PPOFF FOR B- LEVEL 80286
149        RET
150
151        CMOS_WRITE ENDP
152
153        CMOS_WRITE ENDP

```

SECTION 5

```

114                                     PAGE
115 0034                                DDS   PROC   NEAR                                ; LOAD (DS) TO DATA AREA
116 0034 2E: 8E 1E 003A R              MOV   DS,CS;DDSDATA                ; PUT SEGMENT VALUE OF DATA AREA INTO DS
117 0039 C3                              RET                                     ; RETURN TO USER WITH (DS) = DATA
118
119 003A ---- R                          DDSDATA DW   DATA                    ; SEGMENT SELECTOR VALUE FOR DATA AREA
120
121 003C                                DDS   ENDP
122
123 ----- 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
133 003C                                E_MSG  PROC   NEAR
134 003C F7 C5 3FFF                      TEST   BP,03FFFFH                    ; CHECK FOR NOT FIRST ERROR MESSAGE
135 0040 75 08                            JNZ   E_MSG1                          ; SKIP IF NOT FIRST ERROR MESSAGE
136
137 0042 56                                PUSH  SI                               ; SAVE MESSAGE POINTER
138 0043 81 E6 3FFF                      AND   SI,03FFFFH                    ; USE LOW 14 BITS OF MESSAGE OFFSET
139 0047 0B EE                            OR    BP,SI                            ; AS FIRST ERROR MESSAGE FLAG
140 0049 5E                                POP   SI                               ; (BIT 0 = E161/E162, BIT 1 = BAD_CONFIG)
141 004A                                E_MSG1:
142 004A E8 0063 R                        CALL  P_MSG                            ; PRINT MESSAGE
143 004D 1E                                PUSH  DS                               ; SAVE CALLERS (DS)
144 004E EB 0034 R                        CALL  DDS                              ; POINT TO POST/BIOS DATA SEGMENT
145 0051 F6 06 0010 R 01                 TEST  BYTE PTR @EQUIP_FLAG,01H       ; LOOP/HALT ON ERROR SWITCH ON ?
146 0056 74 02                            JZ    MFG_HALT                        ; YES - THEN GO TO MANUFACTURING HALT
147
148 0058 1F                                POP   DS                               ; RESTORE CALLERS (DS)
149 0059 C3                              RET
150
151 005A                                MFG_HALT:
152 005A FA                                CLI                                     ; DISABLE INTERRUPTS
153 005B A0 0015 R                        MOV   AL,@MFG_ERR_FLAG               ; RECOVER ERROR INDICATOR
154 005E E6 80                            OUT   MFG_PORT,AL                    ; SET INTO MANUFACTURING PORT
155 0060 F4                                HLT                                     ; HALT SYSTEM
156 0061 EB F7                            JMP   MFG_HALT                        ; HOT NMI TRAP
157
158 0063                                E_MSG  ENDP
159
160
161 0063                                P_MSG  PROC   NEAR
162 0063 2E: 8A 04                        MOV   AL,CS:[SI]                     ; DISPLAY STRING FROM (CS:)
163 0066 49                                INC   SI                               ; POINT TO NEXT CHARACTER
164 0067 50                                PUSH  AX                               ; SAVE PRINT CHARACTER
165 0068 EB 0128 R                        CALL  PRT_HEX                         ; CALL VIDEO_IO
166 006B 58                                POP   AX                               ; RECOVER PRINT CHARACTER
167 006C 3C 0A                            CMP   AL,LF                           ; WAS IT LINE FEED?
168 006E 75 F3                            JNE  P_MSG                            ; NO, KEEP PRINTING STRING
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
175 ; MORE SHORT TONES (9/32 SECOND) TO INDICATE A FAILURE ON THE
176 ; PLANAR BOARD, A BAD MEMORY MODULE, OR A PROBLEM WITH THE CRT.
177 ; ENTRY
178 ; PARAMETERS:
179 ; DH = NUMBER OF LONG TONES TO BEEP.
180 ; DL = NUMBER OF SHORT TONES TO BEEP.
181 -----
182 0071                                ERR_BEEP  PROC   NEAR
183 0071 9C                                PUSHF                                  ; SAVE FLAGS
184 0072 FA                                CLI                                     ; DISABLE SYSTEM INTERRUPTS
185 0073 0A F6                            OR    DH,DH                           ; ANY LONG ONES TO BEEP
186 0075 74 1E                            JZ    G3                               ; NO, DO THE SHORT ONES
187 0077                                G1:
188 0077 B3 70                            MOV   BL,112                          ; COUNTER FOR LONG BEEPS (1-3/4 SECONDS)
189 0079 B9 0500                          MOV   CX,1280                          ; DIVISOR FOR 932 HZ
190 007C EB 00AF R                        CALL  BEEP                             ; DO THE BEEP
191 007F B9 C233                          MOV   CX,49715                         ; 2/3 SECOND DELAY AFTER LONG BEEP
192 0082 EB 00F5 R                        CALL  WAITF                            ; DELAY BETWEEN BEEPS
193 0085 FE CE                            DEC   DH                               ; ANY MORE LONG BEEPS TO DO
194 0087 75 EE                            JNZ  G1                               ; LOOP TILL DONE
195
196 0089 1E                                PUSH  DS                               ; SAVE DS REGISTER CONTENTS
197 008A EB 0034 R                        CALL  DDS                              ; MANUFACTURING TEST MODE?
198 008D 80 3E 0012 R 01                 CMP   @MFG_TST,01H                    ; RESTORE ORIGINAL CONTENTS OF (DS)
199 0092 1F                                POP   DS                               ; YES - STOP BLINKING LED
200 0093 74 C5                            JPE  MFG_HALT
201
202 0095                                G3:
203 0095 B3 12                            MOV   BL,18                            ; SHORT BEEPS
204 0097 B9 04B8                          MOV   CX,1208                          ; COUNTER FOR A SHORT BEEP (9/32)
205 009A EB 00AF R                        CALL  BEEP                             ; DO THE SOUND
206 009D B9 8178                          MOV   CX,33144                         ; 1/2 SECOND DELAY AFTER SHORT BEEP
207 00A0 EB 00F5 R                        CALL  WAITF                            ; DELAY BETWEEN BEEPS
208 00A3 FE CA                            DEC   G3                               ; DONE WITH SHORT BEEPS COUNT
209 00A5 75 EE                            JNZ  G3                               ; LOOP TILL DONE
210
211 00A7 B9 8178                          MOV   CX,33144                         ; 1/2 SECOND DELAY AFTER LAST BEEP
212 00AA EB 00F5 R                        CALL  WAITF                            ; MAKE IT ONE SECOND DELAY BEFORE RETURN
213 00AD 9D                                POPF                                  ; RESTORE FLAGS TO ORIGINAL SETTINGS
214 00AE C3                              RET                                     ; RETURN TO CALLER
215
216 00AF                                ERR_BEEP  ENDP

```

```

217 PAGE
218 :--- BEEP -----
219 : ROUTINE TO SOUND THE BEEPER USING TIMER 2 FOR TONE
220 : ENTRY:
221 : (BL) = DURATION COUNTER ( 1 FOR 1/64 SECOND )
222 : (CX) = FREQUENCY DIVISOR (1193180/FREQUENCY) (1331 FOR 886 HZ)
223 : EXIT:
224 : (AX), (BL), (CX) MODIFIED.
225 -----
226
227 00AF BEEP PROC NEAR ; SETUP TIMER 2
228 00AF 9C PUSHF ; SAVE INTERRUPT STATUS
229 00B0 FA CLI ; BLOCK INTERRUPTS DURING UPDATE
230 00B1 80 B6 MOV AL,10110110B ; SELECT TIMER 2, LSB, MSB, BINARY
231 00B3 E6 43 OUT TIMER+3,AL ; WRITE THE TIMER MODE REGISTER
232 00B5 EB 00 JMP $+2 ; 1/0 DELAY
233 00B7 8A C1 MOV AL,CL ; DIVISOR FOR HZ (LOW)
234 00B9 E6 42 OUT TIMER+2,AL ; WRITE TIMER 2 COUNT - LSB
235 00BB EB 00 JMP $+2 ; 1/0 DELAY
236 00BD 8A C5 MOV AL,CH ; DIVISOR FOR HZ (HIGH)
237 00BF E6 42 OUT TIMER+2,AL ; WRITE TIMER 2 COUNT - MSB
238 00C1 E4 61 IN AL,PORT_B ; GET CURRENT SETTING OF PORT
239 00C3 8A E0 MOV AH,AL ; SAVE THAT SETTING
240 00C5 0C 03 OR AL,GATE2+SPK2 ; GATE TIMER 2 AND TURN SPEAKER ON
241 00C7 E6 61 OUT PORT_B,AL ; AND RESTORE INTERRUPT STATUS
242 00C9 9D POPF
243 00CA G7: ; 1/64 SECOND PER COUNT (BL)
244 00CA B9 040B MOV CX,1035 ; DELAY COUNT FOR 1/64 OF A SECOND
245 00CD E8 00F5 R CALL WAITF ; GO TO BEEP DELAY 1/64 COUNT
246 00DD FE CB DEC BL ; (BL) LENGTH COUNT EXPIRED?
247 00DE 75 F6 JNZ G7 ; NO - CONTINUE BEEPING SPEAKER
248
249 00D4 9C PUSHF ; SAVE INTERRUPT STATUS
250 00D5 FA CLI ; BLOCK INTERRUPTS DURING UPDATE
251 00D6 E4 61 IN AL,PORT_B ; GET CURRENT PORT VALUE
252 00D8 0C FC OR AL,NOT (GATE2+SPK2) ; ISOLATE CURRENT SPEAKER BITS IN CASE
253 00DA 22 E0 AND AH,AL ; SOMEONE TURNED THEM OFF DURING BEEP
254 00DC 8A C4 MOV AL,AH ; RECOVER VALUE OF PORT
255 00DE 24 FC AND AL,NOT (GATE2+SPK2) ; FORCE SPEAKER DATA OFF
256 00E0 E6 61 OUT PORT_B,AL ; AND STOP SPEAKER TIMER
257 00E2 9D POPF ; RESTORE INTERRUPT FLAG STATE
258 00E3 B9 040B MOV CX,1035 ; FORCE 1/64 SECOND DELAY (SHORT)
259 00E6 E8 00F5 R CALL WAITF ; MINIMUM DELAY BETWEEN ALL BEEPS
260 00E9 9C PUSHF ; SAVE INTERRUPT STATUS
261 00EA FA CLI ; BLOCK INTERRUPTS DURING UPDATE
262 00EB E4 61 IN AL,PORT_B ; GET CURRENT PORT VALUE IN CASE
263 00ED 24 03 AND AL,GATE2+SPK2 ; SOMEONE TURNED THEM ON
264 00EF 0A C4 OR AL,AH ; RECOVER VALUE OF PORT_B
265 00F1 E6 61 OUT PORT_B,AL ; RESTORE SPEAKER STATUS
266 00F3 9D POPF ; RESTORE INTERRUPT FLAG STATE
267 00F4 C3 RET
268
269 00F5 BEEP ENDP
270
271 :--- WAITF -----
272 : FIXED TIME WAIT ROUTINE (HARDWARE CONTROLLED - NOT PROCESSOR)
273 : ENTRY:
274 : (CX) = COUNT OF 15.085737 MICROSECOND INTERVALS TO WAIT
275 : MEMORY REFRESH TIMER 1 OUTPUT USED AS REFERENCE
276 : EXIT:
277 : AFTER (CX) TIME COUNT (PLUS OR MINUS 16 MICROSECONDS)
278 : (CX) = 0
279 -----
280
281 282 00F5 WAITF PROC NEAR ; DELAY FOR (CX)*15.085737 US
283 00F5 50 PUSH AX ; SAVE WORK REGISTER (AH)
284
285 286 00F6 WAITF1: ; USE TIMER 1 OUTPUT BITS
287 00F6 E4 61 IN AL,PORT_B ; READ CURRENT COUNTER OUTPUT STATUS
288 00F8 24 10 AND AL,REFRESH_BIT ; MASK FOR REFRESH DETERMINE BIT
289 00FA 3A C4 CMP AL,AH ; DID IT JUST CHANGE
290 00FC 74 F8 JE WAITF1 ; WAIT FOR A CHANGE IN OUTPUT LINE
291
292 00FE 8A E0 MOV AH,AL ; SAVE NEW FLAG STATE
293 0100 E2 F4 LOOP WAITF1 ; DECREMENT HALF CYCLES TILL COUNT END
294
295 0102 58 POP AX ; RESTORE (AH)
296 0103 C3 RET ; RETURN (CX)= 0
297
298 299 0104 WAITF ENDP
299
300 :--- CONFIG_BAD -----
301 : SET CMOS_DIAG WITH CONFIG ERROR BIT (WITH NMI DISABLED)
302 : (BP) BIT 14 SET ON TO INDICATE CONFIGURATION ERROR
303 -----
304
305 304 0104 CONFIG_BAD PROC NEAR
306 0104 50 AX ; ADDRESS CMOS DIAGNOSTIC STATUS BYTE
307 0105 BB BEBE MOV AX,X*(CMOS_DIAG+NMI) ; GET CURRENT VALUE
308 0108 EB 0000 R CALL CMOS_READ ; SET BAD CONFIGURATION BIT
309 010D 86 E0 OR AH,AL ; SETUP FOR WRITE
310 010F EB 001A R CALL CMOS_WRITE ; UPDATE CMOS WITH BAD CONFIGURATION
311 0112 58 POP AX
312 0113 B1 CD 4000 OR BP,04000H ; SET CONFIGURATION BAD FLAG IN (BP)
313 0117 C3 OR RET
314
315 0118 CONFIG_BAD ENDP

```

SECTION 5

```

316 PAGE
317 ;--- XPC_BYTE -- XLATE_PR -- PRT_HEX -----
318 ;
319 ; CONVERT AND PRINT ASCII CODE CHARACTERS
320 ;
321 ; AL CONTAINS NUMBER TO BE CONVERTED.
322 ; AX AND BX DESTROYED.
323 ;-----
324
325 0118 XPC_BYTE PROC NEAR ; DISPLAY TWO HEX DIGITS
326 0118 50 PUSH AX ; SAVE FOR LOW NIBBLE DISPLAY
327 0119 C0 E8 04 SHR AL,4 ; NIBBLE SWAP
328 011C EB 0122 R CALL XLAT_PR ; DO THE HIGH NIBBLE DISPLAY
329 011F 58 POP AX ; RECOVER THE NIBBLE
330 0120 24 0F AND AL,0FH ; ISOLATE TO LOW NIBBLE
331 ; FALL INTO LOW NIBBLE CONVERSION
332
333 0122 XLAT_PR PROC NEAR ; CONVERT 00-0F TO ASCII CHARACTER
334 0122 04 90 ADD AL,090H ; ADD FIRST CONVERSION FACTOR
335 0124 27 DAA ; ADJUST FOR NUMERIC AND ALPHA RANGE
336 0125 14 40 ADC AL,040H ; ADD CONVERSION AND ADJUST LOW NIBBLE
337 0127 27 DAA ; ADJUST HIGH NIBBLE TO ASCII RANGE
338
339 0128 PRT_HEX PROC NEAR
340 0128 B4 0E MOV AH,0EH ; DISPLAY CHARACTER IN (AL) COMMAND
341 012A B7 00 MOV BH,0
342 012C CD 10 INT 10H ; CALL VIDEO_IO
343 012E C3 RET
344
345 012F PRT_HEX ENDP
346 012F XLAT_PR ENDP
347 012F XPC_BYTE ENDP
348
349 ;-----
350 ; PRINT A SEGMENT VALUE TO LOOK LIKE A 21 BIT ADDRESS
351 ; DX MUST CONTAIN SEGMENT VALUE TO BE PRINTED
352 ;-----
353
354 012F PRT_SEG PROC NEAR
355 012F 8A C6 MOV AL,DH ; GET MSB
356 0131 EB 0118 R CALL XPC_BYTE ; DISPLAY SEGMENT HIGH BYTE
357 0134 8A C2 MOV AL,DL ; LSB
358 0136 EB 0118 R CALL XPC_BYTE ; DISPLAY SEGMENT LOW BYTE
359 0139 B0 30 MOV AL,'0' ; PRINT A '0'
360 013B EB 0128 R CALL PRT_HEX ; TO MAKE LOOK LIKE ADDRESS
361 013E B0 20 MOV AL,' ' ; ADD ENDING SPACE
362 0140 EB 0128 R CALL PRT_HEX
363 0143 C3 RET
364
365 0144 PRT_SEG ENDP
366
367 ;-----
368 ; PROT_PRT_HEX -----
369 ;
370 ; PUT A CHARACTER TO THE DISPLAY BUFFERS WHEN IN PROTECTED MODE
371 ;
372 ; (AL) = ASCII CHARACTER
373 ; (DI) = DISPLAY REGEN BUFFER POSITION
374 ;-----
375 0144 PROT_PRT_HEX PROC NEAR
376 0144 06 PUSH ES ; SAVE CURRENT SEGMENT REGISTERS
377 0145 57 PUSH DI
378 0146 D1 E7 SAL DI,1 ; MULTIPLY OFFSET BY TWO
379
380 ;----- MONOCHROME VIDEO CARD
381
382 0148 6A 20 PUSH BYTE PTR C_BWCRT_PTR ; GET MONOCHROME BUFFER SEGMENT SELECTOR
383 014A 07 POP ES ; SET (ES) TO B/W DISPLAY BUFFER
384 014B AA STOSB ; PLACE CHARACTER IN BUFFER
385 014C 4F DEC DI ; ADJUST POINTER BACK
386
387 ;----- ENHANCED GRAPHICS ADAPTER
388
389 014D 6A 30 PUSH BYTE PTR E_CCRT_PTR ; ENHANCED COLOR DISPLAY POINTER LOW 64K
390 014F 07 POP ES ; LOAD SEGMENT SELECTOR
391 0150 AA STOSB ; PLACE CHARACTER IN BUFFER
392 0151 4F DEC DI ; ADJUST POINTER BACK
393 0152 6A 38 PUSH BYTE PTR E_CCRT_PTR2 ; ENHANCED COLOR DISPLAY POINTER HI 64K
394 0154 07 POP ES ; LOAD SEGMENT SELECTOR
395 0155 AA STOSB ; PLACE CHARACTER IN BUFFER
396 0156 4F DEC DI ; ADJUST POINTER BACK
397
398 ;----- COMPATIBLE COLOR
399
400 0157 6A 28 PUSH BYTE PTR C_CCRT_PTR ; SET (DS) TO COMPATIBLE COLOR MEMORY
401 0159 07 POP ES
402 015A 53 PUSH BX ; SAVE WORK REGISTERS
403 015B 52 PUSH DX
404 015C 51 PUSH CX
405 015D 33 C9 XOR CX,CX ; TIMEOUT LOOP FOR "BAD" HARDWARE
406 015F BA 03DA MOV DX,03DAH ; STATUS ADDRESS OF COLOR CARD
407 0162 93 XCHG AX,BX ; SAVE IN (BX) REGISTER
408 0163 EC
409 0163 EC
410 0164 A8 09 IN AL,DX ; GET COLOR CARD STATUS
411 0166 E1 0F TEST AL,RVRT+RHRZ ; CHECK FOR VERTICAL RETRACE (OR HORZ)
412 0168 93 XCHG AX,BX ; TIMEOUT LOOP TILL FOUND
413 0169 AA STOSB ; RECOVER CHARACTERS
414 ; PLACE CHARACTER IN BUFFER
415 016A 59 POP CX ; RESTORE REGISTERS
416 016B 5A POP DX
417 016C 5B POP BX
418 016D 5F POP DI
419 016E 07 POP ES
420 016F C3 RET
421
422 0170 PROT_PRT_HEX ENDP

```



```

423 PAGE
424 ;-----
425 ; ROM CHECKSUM SUBROUTINE ;
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: ; ENTRY FOR OPTIONAL ROM TEST
432 0172 32 C0 XOR AL,AL
433 0174 ROM_L:
434 0174 02 07 ADD BX, [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 ; THIS ROUTINE CHECKSUMS OPTIONAL ROM MODULES AND ;
445 ; IF CHECKSUM IS OK, CALLS INITIALIZATION/TEST CODE IN MODULE ;
446 ;-----
447
448 017C ROM_CHECK PROC NEAR
449 017C BB ---- R MOV AX,DATA ; POINT ES TO DATA AREA
450 017F 8E C0 MOV ES,AX
451 0181 2A E4 SUB AH,AH ; ZERO OUT AH
452 0183 8A 47 02 MOV AL,[BX+2] ; GET LENGTH INDICATOR
453 0186 C1 E0 09 SHL AX,9 ; MULTIPLY BY 512
454 0189 8B C8 MOV CX,AX ; SET COUNT
455 018B C1 E8 04 SHR AX,4
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 74 05 JZ ROM_CHECK_1
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_1:
464 019A 52 PUSH DX ; SAVE POINTER
465 019B 26 C7 06 0067 R 0003 MOV ES:010 ROM_INIT,0003H ; LOAD OFFSET
466 01A2 26 8C 1E 0069 R MOV ES:010 ROM_SEG,DS ; LOAD SEGMENT
467 01A7 26 FF 1E 0067 R CALL DWORD PTR ES:010 ROM_INIT; CALL INITIALIZE/TEST ROUTINE
468 01AC 5A POP DX
469
470 01AD ROM_CHECK_END:
471 01AD C3 RET ; RETURN TO CALLER
472
473 ROM_CHECK ENDP
474
475 ;--- KBD RESET ---
476 ; THIS PROCEDURE WILL SEND A SOFTWARE RESET TO THE KEYBOARD. ;
477 ; SCAN CODE 0AAH SHOULD BE RETURNED TO THE PROCESSOR. ;
478 ; SCAN CODE 065H IS DEFINED FOR MANUFACTURING TEST ;
479 ;-----
480
481 01AE KBD_RESET PROC NEAR
482 01AE B0 FF MOV AL,0FFH ; 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 80 FD MOV AL,0FDH ; ENABLE KEYBOARD INTERRUPTS
490 01BB E6 21 OUT INTA01,AL ; WRITE 0259 INTERRUPT MASK REGISTER
491 01BD C6 06 006B R 00 MOV 01INTR_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 01C7 F6 06 006B R 02 G11: TEST 01INTR_FLAG,02H ; DID A KEYBOARD INTERRUPT OCCUR ?
497 01CC 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
501 01D2 75 F3 JNZ G11 ; TRY AGAIN
502 01D4
503 01D4 E4 60 G12: IN AL,PORT_A ; READ KEYBOARD SCAN CODE
504 01D6 8A D8 MOV BL,AL ; SAVE SCAN CODE JUST READ
505 01D8
506 01D8 C3 RET ; RETURN TO CALLER
507
508 01D9 KBD_RESET ENDP
509
510 ;-----
511 ; BLINK LED PROCEDURE FOR MFG RUN-IN TESTS ;
512 ; IF LED IS ON, TURN IT OFF. IF OFF, TURN ON. ;
513 ;-----
514
515 01D9 BLINK_INT PROC NEAR
516 01D9 FB STI
517 01DA 50 PUSH AX ; SAVE AX REGISTER CONTENTS
518 01DB E4 80 IN AL,MFG_PORT ; READ CURRENT VALUE OF MFG_PORT
519 01DD 34 40 XOR AL,01000000B ; FLIP CONTROL BIT
520 01DF E6 80 OUT MFG_PORT,AL
521 01E1 B0 20 MOV AL,E01
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
    
```

SECTION 5

```

527                                     PAGE
528                                     :-----:
529                                     : THIS ROUTINE INITIALIZES THE TIMER DATA AREA IN THE ROM BIOS :
530                                     : DATA AREA. IT IS CALLED BY THE POWER ON ROUTINES. IT CONVERTS :
531                                     : HR:MIN:SEC FROM CMOS TO TIMER TICS. IF CMOS IS INVALID, TIMER :
532                                     : IS SET TO ZERO. :
533                                     :-----:
534 : INPUT NONE PASSED TO ROUTINE BY CALLER :
535 : CMOS LOCATIONS USED FOR TIME :
536 :-----:
537 : OUTPUT *TIMER_LOW :
538 : *TIMER_HIGH :
539 : *TIMER_OFI :
540 : ALL REGISTERS UNCHANGED :
541 :-----:
542 = 0012 COUNTS_SEC EQU 18 ; TIMER DATA CONVERSION EQUATES
543 = 0444 COUNTS_MIN EQU 1092
544 = 0007 COUNTS_HOUR EQU 7 ; 65543 - 65536
545 = 0080 UPDATE_TIMER EQU 10000000B ; RTC UPDATE IN PROCESS BIT MASK
546
547 01E7 SET_TOD PROC NEAR
548 01E7 60 PUSHA
549 01E8 IE PUSH DS
550 01E9 E8 0034 R CALL DDS ; ESTABLISH SEGMENT
551 01EC 2B C0 SUB AX,AX
552 01EE A2 0070 R MOV *TIMER_OFI,AL ; RESET TIMER ROLL OVER INDICATOR
553 01F1 A3 006C R MOV *TIMER_LOW,AX ; AND TIMER COUNT
554 01F4 A3 006E R MOV *TIMER_HIGH,AX
555 01F7 80 8E MOV AL,CMOS_DIAG+NM1 ; CHECK CMOS VALIDITY
556 01F9 E8 0000 R CMOS CALL ; READ DIAGNOSTIC LOCATION IN CMOS
557 01FC 24 C4 AND AL,BAD_BAT+BAD_CKSUM+CMOS_CLK_FAIL
558 01FE 2B C9 SUB CX,CX ; BAD BATTERY, CKSUM ERROR, CLOCK ERROR
559 0200 75 64 JNZ POD_DONE ; CMOS NOT VALID -- TIMER SET TO ZERO
560
561 0202 B0 8A UIP: MOV AL,CMOS_REG_A+NM1 ; ACCESS REGISTER A
562 0204 E8 0000 R CALL CMOS_READ ; READ CMOS CLOCK REGISTER A
563 0207 A8 80 TEST AL,UPDATE_TIMER
564 0209 E1 F7 LOOPZ UIP ; WAIT TILL UPDATE BIT IS ON
565
566 020B E3 59 JCXZ POD_DONE ; CMOS CLOCK STUCK IF TIMEOUT
567
568 020D B0 8A UIPOFF: MOV AL,CMOS_REG_A+NM1 ; ACCESS REGISTER A
569 020F E8 0000 R CALL CMOS_READ ; READ CMOS CLOCK REGISTER A
570 0212 A8 80 TEST AL,UPDATE_TIMER
571 0214 E0 F7 LOOPNZ UIPOFF ; NEXT WAIT TILL END OF UPDATE
572
573 0216 E3 4E JCXZ POD_DONE ; CMOS CLOCK STUCK IF TIMEOUT
574
575 0218 B0 80 MOV AL,CMOS_SECONDS+NM1 ; TIME JUST UPDATED
576 021A E8 0000 R CALL CMOS_READ ; ACCESS SECONDS VALUE IN CMOS
577 021D 3C 59 CMP AL,59H ; ARE THE SECONDS WITHIN LIMITS?
578 021F 77 48 JA TOD_ERROR ; GO IF NOT
579
580 0221 E8 027F R CALL CVT_BINARY ; CONVERT IT TO BINARY
581 0224 8B C8 MOV CX,AX ; MOVE COUNT TO ACCUMULATION REGISTER
582 0226 C1 59 02 SHR CX,2 ; ADJUST FOR SYSTEMATIC SECONDS ERROR
583 0229 B3 12 MOV BL,COUNTS_SEC
584 022B F6 E3 MUL BL ; COUNT FOR SECONDS
585 022D 03 C8 ADD CX,AX
586 022F B0 82 MOV AL,CMOS_MINUTES+NM1
587 0231 E8 0000 R CALL CMOS_READ ; ACCESS MINUTES VALUE IN CMOS
588 0234 3C 59 CMP AL,59H ; ARE THE MINUTES WITHIN LIMITS?
589 0236 77 31 JA TOD_ERROR ; GO IF NOT
590 0238 E8 027F R CALL CVT_BINARY ; CONVERT IT TO BINARY
591 023B 50 PUSH AX ; SAVE MINUTES COUNT
592 023C D1 E8 MOV AX,I ; ADJUST FOR SYSTEMATIC MINUTES ERROR
593 023E 03 C8 ADD CX,AX ; ADD ADJUSTMENT TO COUNT
594 0240 58 POP AX ; RECOVER BCD MINUTES VALUE
595 0241 BB 0444 MOV BX,COUNTS_MIN
596 0244 F7 E3 MUL BX ; COUNT FOR MINUTES
597 0246 03 C8 ADD CX,AX ; ADD TO ACCUMULATED VALUE
598 0248 B0 84 MOV AL,CMOS_HOURS+NM1
599 024A E8 0000 R CALL CMOS_READ ; ACCESS HOURS VALUE IN CMOS
600 024D 3C 23 CMP AL,23H ; ARE THE HOURS WITHIN LIMITS?
601 024F 77 18 JA TOD_ERROR ; GO IF NOT
602
603 0251 E8 027F R CALL CVT_BINARY ; CONVERT IT TO BINARY
604 0254 8B D0 MOV DX,AX
605 0256 B3 07 MOV BL,COUNTS_HOUR
606 0258 F6 E3 MUL BL ; COUNT FOR HOURS
607 025A 03 C1 ADD AX,CX
608 025C 83 D2 00 ADC DX,0000H
609 025F 89 16 006E R MOV *TIMER_HIGH,DX
610 0263 A3 006C R MOV *TIMER_LOW,AX
611 0266
612 0266 IF POD_DONE: POP DS
613 0267 61 POPA
614 0268 C3 RET
615
616 0269 TOD_ERROR: POP DS ; RESTORE SEGMENT
617 0269 IF POPA ; RESTORE REGISTERS
618 026A 61 POPA ; DISPLAY CLOCK ERROR
619 026B BE 0000 E MOV SI,OFFSET E163
620 026E E8 003C R CALL E_MSG ; DISPLAY CLOCK ERROR
621 0271 B8 8E8E MOV AX,*X*(CMOS_DIAG+NM1) ; SET CLOCK ERROR IN STATUS
622 0274 E8 0000 R CALL CMOS_READ ; READ DIAGNOSTIC CMOS LOCATION
623 0277 0C 04 OR AL,CMOS_CLK_FAIL ; SET NEW STATUS WITH CMOS CLOCK ERROR
624 0279 86 C4 AND AL,XCHG ; MOVE NEW STATUS TO WORK REGISTER
625 027B E8 001A R CALL CMOS_WRITE ; UPDATE STATUS LOCATION
626 027E C3 RET
627
628 027F SET_TOD ENDP
629
630 027F CVT_BINARY PROC NEAR
631 027F 8A E0 MOV AH,AL ; UNPACK 2 BCD DIGITS IN AL
632 0281 C0 EC 04 SHR AH,4
633 0284 24 0F AND AL,0FH ; RESULT IS IN AX
634 0286 D5 0A AAD ; CONVERT UNPACKED BCD TO BINARY
635 0288 C3 RET
636
637 0289 CVT_BINARY ENDP

```

```

638 PAGE
639 I--- D11 -- INT ?? H -- ( IRQ LEVEL ?? ) -----
640 ; TEMPORARY INTERRUPT SERVICE ROUTINE FOR POST
641 ;
642 ; THIS ROUTINE IS ALSO LEFT IN PLACE AFTER THE POWER ON DIAGNOSTICS
643 ; TO SERVICE UNUSED INTERRUPT VECTORS. LOCATION *INTR_FLAG* WILL
644 ; CONTAIN EITHER:
645 ; 1) LEVEL OF HARDWARE INTERRUPT THAT CAUSED CODE TO BE EXECUTED, OR
646 ; 2) "FF" FOR A NON-HARDWARE INTERRUPT THAT WAS EXECUTED ACCIDENTALLY.
647 ;-----
648
649 0289          D11 PROC NEAR
650 0289 50      PUSH AX
651 028A 53      PUSH BX
652 028B B0 0B  MOV AL,0BH
653 028D E6 20  OUT INTA00,AL
654 028F EB 00  JMP $+2
655 0291 E4 20  IN AL,INTA00
656 0293 8A E0  MOV AH,AL
657 0295 0A C4  OR AL,AH
658 0297 75 04  JNZ HW_INT
659
660 0299 B4 FF  MOV AH,OFFH
661 029B EB 2F  JMP SHORT SET_INTR_FLAG
662 029D
663 029D B0 0B  HW_INT: MOV AL,0BH
664 029F E6 A0  OUT INTB00,AL
665 02A1 EB 00  JMP $+2
666 02A3 E4 A0  IN AL,INTB00
667 02A5 8A F8  MOV BH,AL
668 02A7 0A FF  OR BH,BH
669 02A9 74 10  JZ NOT_SEC
670
671 02AB E4 A1  IN AL,INTB01
672 02AD 0A C7  OR AL,BH
673 02AF EB 00  JMP $+2
674 02B1 E6 A1  OUT INTB01,AL
675 02B3 B0 20  MOV AL,E01
676 02B5 EB 00  JMP $+2
677 02B7 E6 A0  IN AL,INTB00
678 02B9 EB 0D  JMP SHORT IS_SEC
679 02BB
680 02BB E4 21  NOT_SEC: IN AL,INTA01
681 02BD EB 00  JMP $+2
682 02BF 80 E4 FB AND AH,OFBH
683 02C2 0A C4  OR AL,AH
684 02C4 E6 21  OUT INTA01,AL
685 02C6 EB 00  JMP $+2
686 02C8
687 02C8 B0 20  IS_SEC: MOV AL,E01
688 02CA E6 20  OUT INTA00,AL
689 02CC
690 02CC 5B      SET_INTR_FLAG: POP BX
691 02CD 1E      PUSH DS
692 02CE E8 0034 R CALL D05
693 02D1 88 26 006B R MOV @INTR_FLAG,AH
694 02D5 1F      POP DS
695 02D6 58      POP AX
696 02D7        DUMMY_RETURN_1:
697 02D7 CF      IRET
698
699 02D8        D11 ENDP
700
701
702 ;--- HARDWARE INT 71 H -- ( IRQ LEVEL 9 ) -- TO INT 0A H ---
703 ; REDIRECT SLAVE INTERRUPT 9 TO INTERRUPT LEVEL 2
704 ; THIS ROUTINE FIELDS LEVEL 9 INTERRUPTS AND
705 ; CONTROL IS PASSED TO MASTER INTERRUPT LEVEL 2
706 ;-----
707 02D8        RE_DIRECT PROC NEAR
708 02D8 50      PUSH AX
709 02D9 B0 20  MOV AL,E01
710 02DB E6 A0  OUT INTB00,AL
711 02DD 58      POP AX
712 02DE CD 0A  INT 0AH
713
714 02E0 CF      IRET
715
716 02E1        RE_DIRECT ENDP
717
718 ;--- HARDWARE INT 75 H -- ( IRQ LEVEL 13 ) ---
719 ; SERVICE X287 INTERRUPTS
720 ; THIS ROUTINE FIELDS X287 INTERRUPTS AND CONTROL
721 ; IS PASSED TO THE NMI INTERRUPT HANDLER FOR
722 ; COMPATIBILITY.
723 ;-----
724
725 02E1        INT_287 PROC NEAR
726 02E1 50      PUSH AX
727 02E2 32 C0  XOR AL,AL
728 02E4 E6 F0  OUT X287,AL
729
730 02E6 B0 20  MOV AL,E01
731 02E8 E6 A0  OUT INTB00,AL
732 02EA E6 20  OUT INTA00,AL
733 02EC 58      POP AX
734 02ED CD 02  INT 02H
735
736 02EF CF      IRET
737
738 02F0        INT_287 ENDP
739
740 02F0        PROC_SHUTDOWN PROC
741
742 02F0 B0 FE  MOV AL,SHUT_CMD
743 02F2 E6 64  OUT STATUS_PORT,AL
744 02F4
745 02F4 F4      HLT
746 02F5 EB FD  JMP PROC_5
747
748 02F7        PROC_SHUTDOWN ENDP
749 02F7
750 CODE ENDS
  
```

SECTION 5

```

1          PAGE 118,121
2          TITLE TEST5 ---- 06/10/85 EXCEPTION INTERRUPT TEST HANDLERS
3          .286C
4          .LIST
5          0000      CODE      SEGMENT BYTE PUBLIC
6
7          PUBLIC   POST5
8          PUBLIC   SYSINIT1
9
10         ;-----
11         ;     EXCEPTION INTERRUPT ROUTINE
12         ;-----
13
14         ASSUME   CS:CODE,DS:ABS0
15
16         POST5:
17         EXC_00:  MOV     AL,90H      ; <<<> SET CHECKPOINT <<<>
18                 JMP     TEST_EXC   ; GO TEST IF EXCEPTION WAS EXPECTED
19
20         EXC_01:  MOV     AL,91H      ; <<<> SET CHECKPOINT <<<>
21                 JMP     TEST_EXC   ; GO TEST IF EXCEPTION WAS EXPECTED
22
23         EXC_02:  MOV     AL,92H      ; <<<> SET CHECKPOINT <<<>
24                 JMP     TEST_EXC   ; GO TEST IF EXCEPTION WAS EXPECTED
25
26         EXC_03:  MOV     AL,93H      ; <<<> SET CHECKPOINT <<<>
27                 JMP     TEST_EXC   ; GO TEST IF EXCEPTION WAS EXPECTED
28
29         EXC_04:  MOV     AL,94H      ; <<<> SET CHECKPOINT <<<>
30                 JMP     TEST_EXC   ; GO TEST IF EXCEPTION WAS EXPECTED
31
32         EXC_05:  PUSH    ES          ;
33                 PUSH    BYTE PTR ES,TEMP ; LOAD ES REGISTER WITH SELECTOR
34                 POP     ES
35
36         ;----- FIX BOUND PARAMETERS
37
38         SUB     DI,D1              ; POINT BEGINNING OF THE BLOCK
39         MOV     WORD PTR ES:[D1],0 ; SET FIRST WORD TO ZERO
40         MOV     WORD PTR ES:[D1+2],07FFFH ; SET SECOND TO 07FFFH
41
42         EXC_06:  MOV     AL,95H      ; <<<> SET CHECKPOINT <<<>
43                 JMP     TEST_EXC   ; GO TEST IF EXCEPTION WAS EXPECTED
44
45         EXC_06:  MOV     AL,96H      ; <<<> SET CHECKPOINT <<<>
46                 JMP     SHORT TEST_EXC ; GO TEST IF EXCEPTION WAS EXPECTED
47
48         EXC_07:  MOV     AL,97H      ; <<<> SET CHECKPOINT <<<>
49                 JMP     SHORT TEST_EXC ; GO TEST IF EXCEPTION WAS EXPECTED
50
51         EXC_08:  MOV     AL,98H      ; <<<> SET CHECKPOINT <<<>
52                 JMP     SHORT TEST_EXC ; GO TEST IF EXCEPTION WAS EXPECTED
53
54         EXC_09:  MOV     AL,99H      ; <<<> SET CHECKPOINT <<<>
55                 JMP     SHORT TEST_EXC ; GO TEST IF EXCEPTION WAS EXPECTED
56
57         EXC_10:  MOV     AL,9AH      ; <<<> SET CHECKPOINT <<<>
58                 JMP     SHORT TEST_EXC ; GO TEST IF EXCEPTION WAS EXPECTED
59
60         EXC_11:  MOV     AL,9BH      ; <<<> SET CHECKPOINT <<<>
61                 JMP     SHORT TEST_EXC ; GO TEST IF EXCEPTION WAS EXPECTED
62
63         EXC_12:  MOV     AL,9CH      ; <<<> SET CHECKPOINT <<<>
64                 JMP     SHORT TEST_EXC ; GO TEST IF EXCEPTION WAS EXPECTED
65
66         EXC_13:  MOV     AL,9DH      ; <<<> SET CHECKPOINT <<<>
67                 JMP     SHORT TEST_EXC ; GO TEST IF EXCEPTION WAS EXPECTED
68
69         EXC_14:  MOV     AL,9EH      ; <<<> SET CHECKPOINT <<<>
70                 JMP     SHORT TEST_EXC ; GO TEST IF EXCEPTION WAS EXPECTED
71
72         EXC_15:  MOV     AL,9FH      ; <<<> SET CHECKPOINT <<<>
73                 JMP     SHORT TEST_EXC ; GO TEST IF EXCEPTION WAS EXPECTED
74
75         EXC_16:  MOV     AL,0A0H     ; <<<> SET CHECKPOINT <<<>
76                 JMP     SHORT TEST_EXC ; GO TEST IF EXCEPTION WAS EXPECTED
77
78         EXC_17:  MOV     AL,0A1H     ; <<<> SET CHECKPOINT <<<>
79                 JMP     SHORT TEST_EXC ; GO TEST IF EXCEPTION WAS EXPECTED
80
81         EXC_18:  MOV     AL,0A2H     ; <<<> SET CHECKPOINT <<<>
82                 JMP     SHORT TEST_EXC ; GO TEST IF EXCEPTION WAS EXPECTED
83
84         EXC_19:  MOV     AL,0A3H     ; <<<> SET CHECKPOINT <<<>
85                 JMP     SHORT TEST_EXC ; GO TEST IF EXCEPTION WAS EXPECTED
86
87         EXC_20:  MOV     AL,0A4H     ; <<<> SET CHECKPOINT <<<>
88                 JMP     SHORT TEST_EXC ; GO TEST IF EXCEPTION WAS EXPECTED
89
90         EXC_21:  MOV     AL,0A5H     ; <<<> SET CHECKPOINT <<<>
91                 JMP     SHORT TEST_EXC ; GO TEST IF EXCEPTION WAS EXPECTED
92
93         EXC_22:  MOV     AL,0A6H     ; <<<> SET CHECKPOINT <<<>
94                 JMP     SHORT TEST_EXC ; GO TEST IF EXCEPTION WAS EXPECTED
95
96         EXC_23:  MOV     AL,0A7H     ; <<<> SET CHECKPOINT <<<>
97                 JMP     SHORT TEST_EXC ; GO TEST IF EXCEPTION WAS EXPECTED
98
99         EXC_24:  MOV     AL,0A8H     ; <<<> SET CHECKPOINT <<<>
100                JMP     SHORT TEST_EXC ; GO TEST IF EXCEPTION WAS EXPECTED
101
102         EXC_25:  MOV     AL,0A9H     ; <<<> SET CHECKPOINT <<<>
103                JMP     SHORT TEST_EXC ; GO TEST IF EXCEPTION WAS EXPECTED
104
105         EXC_26:  MOV     AL,0AAH     ; <<<> SET CHECKPOINT <<<>
106                JMP     SHORT TEST_EXC ; GO TEST IF EXCEPTION WAS EXPECTED
107
108         EXC_27:  MOV     AL,0ABH     ; <<<> SET CHECKPOINT <<<>
109                JMP     SHORT TEST_EXC ; GO TEST IF EXCEPTION WAS EXPECTED
110
111         EXC_28:  MOV     AL,0ACH     ; <<<> SET CHECKPOINT <<<>
112                JMP     SHORT TEST_EXC ; GO TEST IF EXCEPTION WAS EXPECTED
113
114         EXC_29:  MOV     AL,0ADH     ; <<<> SET CHECKPOINT <<<>
115                JMP     SHORT TEST_EXC ; GO TEST IF EXCEPTION WAS EXPECTED

```

```

115 008C B0 AD      MOV     AL,0ADH                ; <<>> SET CHECKPOINT <<>>
116 008E EB 22      JMP     SHORT TEST_EXEC       ; GO TEST IF EXCEPTION WAS EXPECTED
117 0090             EXC_30: MOV     AL,0AEH                ; <<>> SET CHECKPOINT <<>>
118 0090 B0 AE      JMP     SHORT TEST_EXEC       ; GO TEST IF EXCEPTION WAS EXPECTED
119 0092 EB 1E      EXC_31: MOV     AL,0AFH                ; <<>> SET CHECKPOINT <<>>
120 0094             JMP     SHORT TEST_EXEC       ; GO TEST IF EXCEPTION WAS EXPECTED
121 0094 B0 AF      SYS_32: MOV     AL,0B0H                ; <<>> SET CHECKPOINT <<>>
122 0096 EB 1A      JMP     SHORT TEST_EXEC       ; GO TEST IF EXCEPTION WAS EXPECTED
123 0098             SYS_33: MOV     AL,0B1H                ; <<>> SET CHECKPOINT <<>>
124 0098 B0 B0      JMP     SHORT TEST_EXEC       ; GO TEST IF INTERRUPT WAS EXPECTED
125 009A EB 16      SYS_34: MOV     AL,0B2H                ; <<>> SET CHECKPOINT <<>>
126 009C             JMP     SHORT TEST_EXEC       ; GO TEST IF INTERRUPT WAS EXPECTED
127 009C B0 B1      SYS_35: MOV     AL,0B3H                ; <<>> SET CHECKPOINT <<>>
128 009E EB 12      JMP     SHORT TEST_EXEC       ; GO TEST IF INTERRUPT WAS EXPECTED
129 00A0             SYS_36: MOV     AL,0B4H                ; <<>> SET CHECKPOINT <<>>
130 00A0 B0 B2      JMP     SHORT TEST_EXEC       ; GO TEST IF INTERRUPT WAS EXPECTED
131 00A2 EB 0E      SYS_37: MOV     AL,0B5H                ; <<>> SET CHECKPOINT <<>>
132 00A4             JMP     SHORT TEST_EXEC       ; GO TEST IF INTERRUPT WAS EXPECTED
133 00A4 B0 B3      SYS_38: MOV     AL,0B6H                ; <<>> SET CHECKPOINT <<>>
134 00A6 EB 0A      JMP     SHORT TEST_EXEC       ; GO TEST IF INTERRUPT WAS EXPECTED
135 00A8             TEST_EXEC: OUT     MFG_PORT,AL           ; OUTPUT THE CHECKPOINT
136 00A8 B0 B4      CMP     AL,0AFH              ; CHECK FOR EXCEPTION
137 00AA EB 06      JA     TEST_EXEC0            ; GO IF A SYSTEM INTERRUPT
138 00AC             PUSH   DS                    ; SAVE THE CURRENT DATA SEGMENT
139 00AC B0 B5      PUSH   DS                    ; DS PTR GDT_PTR
140 00AE EB 02      POP    DS                    ; POP
141 00B0             MOV    DS:ES_TEMP,SEG_LIMIT,MAX_SEG_LEN
142 00B0 B0 B6      MOV    BYTE PTR DS:(ES_TEMP,DATA_ACC_RIGHTS),CPL0_DATA_ACCESS
143 00B2             PUSH   PTR ES_TEMP
144 00B2             POP    ES
145 00B2             POP    DS
146 00B2 E6 80      POP    DX                    ; RESTORE REGISTERS
147 00B4 3C AF      POP    CX                    ; CHECK IF CODE SEGMENT SECOND ON STACK
148 00B6 77 1C      POP    CX
149 00B8             PUSH   CX
150 00B8 1E         CMP    CX,SYS_ROM_CS        ; CONTINUE IF ERROR CODE
151 00B9 6A 08      JNZ   TEST_EXEC0            ; PUT SEGMENT BACK ON STACK
152 00BB 1F         DX
153 00BC C7 06 0048 FFFF
154 00CC C6 06 0048 93
155 00CC 6A 48      PUSH   PTR ES_TEMP
156 00C9 07         POP    ES
157 00CA 1F         POP    DS
158 00CB 5A         POP    DX
159 00CC 59         POP    CX
160 00CD 51         PUSH   CX
161 00CE 83 F9 40   CMP    CX,SYS_ROM_CS        ; CHECK THE CURRENT CHECKPOINT
162 00D1 75 01      JNZ   TEST_EXEC0            ; HALT IF CHECKPOINT BELOW 3BH
163 00D3 52         DX
164 00D4             TEST_EXEC0: XCHG    AH,AL                ; SAVE THE CHECKPOINT
165 00D4             IN     AL,DMA_PAGE+0AH      ; WAS THE EXCEPTION EXPECTED?
166 00D4 86 E0      CMP    AL,AH                ; GO IF YES
167 00D6 E4 8B      JZ     TEST_EXEC1
168 00D8 3A C4      IN     AL,MFG_PORT          ; CHECK THE CURRENT CHECKPOINT
169 00DA 74 0E      CMP    AL,03BH             ; HALT IF CHECKPOINT BELOW 3BH
170 00DC             IN     AL,03BH
171 00DC E4 80      JZ     TEST_EXEC2
172 00DE 3C 3B      JB     TEST_EXEC2
173 00E0 72 01      IRET
174 00E2 CF
175 00E3             TEST_EXEC2: XCHG    AH,AL                ; OUTPUT THE CURRENT CHECKPOINT
176 00E3 86 E0      OUT    MFG_PORT,AL         ; <<>> CHECKPOINT 90 THRU B5 <<>>
177 00E5 E6 80      HLT
178 00E5 E6 80      JMP    TEST_EXEC2          ; INSURE SYSTEM HALT
179 00E7 F4
180 00E8 EB F9      TEST_EXEC3: SUB     AL,AL                ; CLEAR DMA PAGE
181 00EA             OUT    DMA_PAGE+0AH,AL     ; FOR BOUND INSTRUCTION EXPECTED (INT 5)
182 00EA 2A C0      MOV    AX,0100H            ; RETURN
183 00EA 2A C0
184 00EC E6 8B      IRET
185 00EE B8 0100
186 00F1 CF
187
188
189
190
191
192
193
194 00F2             ASSUME CS:CODE,DS:NOTHING,ES:NOTHING,SS:NOTHING
195 00F2 FA
196 00F3 55
197 00F4 B0 81      SYSINIT1 PROC NEAR
198 00F6 E6 80      CL1  BP                      ; NO INTERRUPTS ALLOWED
199 00F8 E8 0149 R  PUSH   BP                    ; SAVE BP
200 00FB 8B EF      MOV    AL,81H               ; <<>><<>><<>><<>><<>><<>><<>>
201 00FD             OUT    MFG_PORT,AL          ; <<>> CHECKPOINT 81 <<>>
202 00FE             CALL  SIDT_BLD
203 00FF 8B EF      MOV    BP,DI                ; SAVE THE POINTER TO JUST PAST THE IDT
204 00FD B8 0800   MOV    AX,SYS_IDT_LEN       ; AS WE HAVE NO SDA, USE THE SIX BYTES
205 0100 AB      STOSW                        ; HERE TO LOAD THE IDTR. WE WILL SIDT
206 0101 B8 0DA0   MOV    AX,SYS_IDT_LOC       ; WHEN WE GET TO SDA INITIALIZATION.
207 0104 AB      STOSW                        ; SEGMENT LIMIT = LENGTH OF IDT
208 0105 B8 0000   MOV    AX,0                 ; STORE THAT AS IDT LIMIT
209 0108 AB      STOSW                        ; IDT ADDRESS
210 0109 26         MOV    ES,0                 ; AND ACCESS RIGHTS BYTE (UNDEFINED)
211 0109 26         SEGOW
212 010A 0F         DB    026H                  ; LOAD THE IDT
213 010A 0F         LIT [BP]                    ; REGISTER FROM THIS AREA
214 010B         DB    00FH
215 010B         + ??70001 LABEL BYTE
216 010E         MOV    BX,WORD PTR [BP]
217 010B         + ??70002 LABEL BYTE
218 010B 01         ORG    OFFSET CS:??70001
219 010E         + DB    001H
220 010E 8B FD     + ORG    OFFSET CS:??70002
221 010E 8B FD     MOV    DI,BP                ; ES:DI NOW --> END OF IDT AGAIN
222
223
224 0110 BF D8A0   +---- BUILD THE GDT.
225 0113 E8 0140 R  MOV    DI,GDT_LOC
226 0116 B8 EF     CALL  GDT_BLD
227 0118 B8 0088   MOV    BP,DI
228 011B AB     MOV    AX,GDT_LEN
229 011B AB     STOSW
230 011B AB     ; SAVE THE ES:DI POINTER
231 011B AB     ; AX = LENGTH OF THE GDT
232 011B AB     ; PUT THAT IN THE LIMIT FIELD

```



```

324 PAGE
325 ; THE FOLLOWING DATA DEFINES THE PRE-INITIALIZED GDT FOR POST TESTS.
326 ; THESE MUST BE INITIALIZED IN THE ORDER IN WHICH THEY APPEAR IN THE
327 ; GDT_DEF STRUCTURE DEFINITION AS IT IS IN "SYSDATA.INC".
328
329 = 01AF
330 GDT_DATA_START EQU $
331 ;----- FIRST ENTRY UNUSABLE - (UNUSED_ENTRY)
332
333 DW 0 ; SEGMENT LIMIT
334 DW 01B1 0000 ; SEGMENT BASE ADDRESS - LOW WORD
335 DW 01B3 00 ; SEGMENT BASE ADDRESS - HIGH BYTE
336 DW 01B4 00 ; ACCESS RIGHTS BYTE
337 DW 01B5 0000 ; RESERVED - MUST BE ZERO
338
339 ;----- THE GDT ITSELF - (GDT_PTR)
340
341 DW GDT_LEN ; SEGMENT LIMIT
342 DW GDT_LOC ; SEGMENT BASE ADDRESS - LOW WORD
343 DW 0 ; SEGMENT BASE ADDRESS - HIGH BYTE
344 DW CPL0_DATA_ACCESS ; ACCESS RIGHTS BYTE
345 DW 01B0 0000 ; RESERVED - MUST BE ZERO
346
347 ;----- THE SYSTEM IDT DESCRIPTOR - (SYS_IDT_PTR)
348
349 DW SYS_IDT_LEN ; SEGMENT LIMIT
350 DW SYS_IDT_LOC ; SEGMENT BASE ADDRESS - LOW WORD
351 DW 0 ; SEGMENT BASE ADDRESS - HIGH BYTE
352 DW CPL0_DATA_ACCESS ; ACCESS RIGHTS BYTE
353 DW 01C5 0000 ; RESERVED - MUST BE ZERO
354
355 ;----- THE SYSTEM DATA AREA DESCRIPTOR - (RSDA_PTR)
356
357 DW SDA_LEN ; SEGMENT LIMIT
358 DW SDA_LOC ; SEGMENT BASE ADDRESS - LOW WORD
359 DW 0 ; SEGMENT BASE ADDRESS - HIGH BYTE
360 DW CPL0_DATA_ACCESS ; ACCESS RIGHTS BYTE
361 DW 01CD 0000 ; RESERVED - MUST BE ZERO
362
363 ;----- COMPATIBLE MONOCHROME DISPLAY REGEN BUFFER - (C_BWCRT_PTR)
364
365 DW MCRT_SIZE ; SEGMENT LIMIT
366 DW MCRT_LO ; SEGMENT BASE ADDRESS - LOW WORD
367 DW MCRT_HI ; SEGMENT BASE ADDRESS - HIGH BYTE
368 DW CPL0_DATA_ACCESS ; ACCESS RIGHTS BYTE
369 DW 01D5 0000 ; RESERVED - MUST BE ZERO
370
371 ;----- COMPATIBLE COLOR DISPLAY REGEN BUFFER - (C_CCRT_PTR)
372
373 DW CCRT_SIZE ; SEGMENT LIMIT
374 DW CCRT_LO ; SEGMENT BASE ADDRESS - LOW WORD
375 DW CCRT_HI ; SEGMENT BASE ADDRESS - HIGH BYTE
376 DW CPL0_DATA_ACCESS ; ACCESS RIGHTS BYTE
377 DW 01DD 0000 ; RESERVED - MUST BE ZERO
378
379 ;----- ENHANCED GRAPHIC ADAPTER REGEN BUFFER - (E_CCRT_PTR)
380
381 DW ECRT_SIZE ; SEGMENT LIMIT
382 DW ECRT_LO ; SEGMENT BASE ADDRESS - LOW WORD
383 DW ECRT_HI ; SEGMENT BASE ADDRESS - HIGH BYTE
384 DW CPL0_DATA_ACCESS ; ACCESS RIGHTS BYTE
385 DW 01E5 0000 ; RESERVED - MUST BE ZERO
386
387 ;----- SECOND PART OF EGA - (E_CCRT_PTR2)
388
389 DW ECRT_SIZE ; SEGMENT LIMIT
390 DW ECRT_LO ; SEGMENT BASE ADDRESS - LOW WORD
391 DW ECRT_HI ; SEGMENT BASE ADDRESS - HIGH BYTE
392 DW CPL0_DATA_ACCESS ; ACCESS RIGHTS BYTE
393 DW 01ED 0000 ; RESERVED - MUST BE ZERO
394
395 ;----- CODE SEGMENT FOR POST CODE, SYSTEM IDT - (SYS_ROM_CS)
396
397 DW MAX_SEG_LEN ; SEGMENT LIMIT
398 DW CSEG_LO ; SEGMENT BASE ADDRESS - LOW WORD
399 DW CSEG_HI ; SEGMENT BASE ADDRESS - HIGH BYTE
400 DW CPL0_CODE_ACCESS ; ACCESS RIGHTS BYTE
401 DW 01F5 0000 ; RESERVED - MUST BE ZERO
402
403 ;----- TEMPORARY DESCRIPTOR FOR ES - (ES_TEMP)
404
405 DW MAX_SEG_LEN ; SEGMENT LIMIT
406 DW NSEG_LO ; SEGMENT BASE ADDRESS - LOW WORD
407 DW NSEG_HI ; SEGMENT BASE ADDRESS - HIGH BYTE
408 DW CPL0_DATA_ACCESS ; ACCESS RIGHTS BYTE
409 DW 01FD 0000 ; RESERVED - MUST BE ZERO
410
411 ;----- TEMPORARY DESCRIPTOR FOR CS AS A DATA SEGMENT - (CS_TEMP)
412
413 DW MAX_SEG_LEN ; SEGMENT LIMIT
414 DW NSEG_LO ; SEGMENT BASE ADDRESS - LOW WORD
415 DW NSEG_HI ; SEGMENT BASE ADDRESS - HIGH BYTE
416 DW CPL0_DATA_ACCESS ; ACCESS RIGHTS BYTE
417 DW 0205 0000 ; RESERVED - MUST BE ZERO
418
419 ;----- TEMPORARY DESCRIPTOR FOR SS - (SS_TEMP)
420
421 DW MAX_SEG_LEN ; SEGMENT LIMIT
422 DW NSEG_LO ; SEGMENT BASE ADDRESS - LOW WORD
423 DW NSEG_HI ; SEGMENT BASE ADDRESS - HIGH BYTE
424 DW CPL0_DATA_ACCESS ; ACCESS RIGHTS BYTE
425 DW 020D 0000 ; RESERVED - MUST BE ZERO
426
427 ;----- TEMPORARY DESCRIPTOR FOR DS - (DS_TEMP)
428
429 DW MAX_SEG_LEN ; SEGMENT LIMIT
430 DW NSEG_LO ; SEGMENT BASE ADDRESS - LOW WORD
431 DW NSEG_HI ; SEGMENT BASE ADDRESS - HIGH BYTE
432 DW CPL0_DATA_ACCESS ; ACCESS RIGHTS BYTE
433 DW 0215 0000 ; RESERVED - MUST BE ZERO
    
```

```

434                                     PAGE
435                                     |----- (POST_TR)
436 0217                                TR_LOC:
437 0217 0800                            DW 00800H          ; SEGMENT LIMIT
438 0219 C000                            DW 0C000H        ; SEGMENT BASE ADDRESS - LOW WORD
439 021B 00                               DB 0             ; SEGMENT BASE ADDRESS - HIGH BYTE
440 021C 81                               DB FREE_TSS     ; ACCESS RIGHTS BYTE
441 021D 0000                            DW 0             ; RESERVED - MUST BE ZERO
442
443                                     |----- (POST_TSS_PTR)
444
445 021F 0800                            DW 00800H        ; SEGMENT LIMIT
446 0221 0217 R                          DW TR_LOC       ; SEGMENT BASE ADDRESS - LOW WORD
447 0223 00                               DB 0             ; SEGMENT BASE ADDRESS - HIGH BYTE
448 0224 93                               DB CPLD_DATA_ACCESS ; ACCESS RIGHTS BYTE
449 0225 0000                            DW 0             ; RESERVED - MUST BE ZERO
450
451                                     |----- (POST_LDTR)
452 0227                                LDT_LOC:
453 0227 0088                            DW GDT_LEN      ; SEGMENT LIMIT
454 0229 D000                            DW 0D000H       ; SEGMENT BASE ADDRESS - LOW WORD
455 022B 00                               DB 0             ; SEGMENT BASE ADDRESS - HIGH BYTE
456 022C E2                               DB LDT_DESC     ; ACCESS RIGHTS BYTE
457 022D 0000                            DW 0             ; RESERVED - MUST BE ZERO
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 CPLD_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                                     |----- ENTRY POINTS FOR THE FIRST 32 SYSTEM INTERRUPTS
472
473                                     SYS_IDT_OFFSETS LABEL WORD
474 0237
475                                     ; INTERRUPTS AS DEFINED
476 0237 0000 R                          DW OFFSET EXC_00 ; EXCPT 00 - DIVIDE ERROR
477 0239 0005 R                          DW OFFSET EXC_01 ; EXCPT 01 - SINGLE STEP
478 023B 000A R                          DW OFFSET EXC_02 ; EXCPT 02 - NMI, SYSTEM REQUEST FOR DI
479 023D 000F R                          DW OFFSET EXC_03 ; EXCPT 03 - BREAKPOINT
480 023F 0014 R                          DW OFFSET EXC_04 ; EXCPT 04 - INTO DETECT
481 0241 0019 R                          DW OFFSET EXC_05 ; EXCPT 05 - BOUND
482 0243 0030 R                          DW OFFSET EXC_06 ; EXCPT 06 - INVALID OPCODE
483 0245 0034 R                          DW OFFSET EXC_07 ; EXCPT 07 - PROCESSOR EXT NOT AVAIL
484 0247 0038 R                          DW OFFSET EXC_08 ; EXCPT 08 - DOUBLE EXCEPTION
485 0249 003C R                          DW OFFSET EXC_09 ; EXCPT 09 - PROCESSOR EXT SEGMENT ERR
486 024B 0040 R                          DW OFFSET EXC_10 ; EXCPT 10 - TSS BAD IN GATE TRANSFER
487 024D 0044 R                          DW OFFSET EXC_11 ; EXCPT 11 - SEGMENT NOT PRESENT
488 024F 0048 R                          DW OFFSET EXC_12 ; EXCPT 12 - STACK SEGMENT NOT PRESENT
489 0251 004C R                          DW OFFSET EXC_13 ; EXCPT 13 - GENERAL PROTECTION
490 0253 0050 R                          DW OFFSET EXC_14
491 0255 0054 R                          DW OFFSET EXC_15
492 0257 0058 R                          DW OFFSET EXC_16 ; EXCPT 16 - PROCESSOR EXTENSION ERROR
493 0259 005C R                          DW OFFSET EXC_17
494 025B 0060 R                          DW OFFSET EXC_18
495 025D 0064 R                          DW OFFSET EXC_19
496 025F 0068 R                          DW OFFSET EXC_20
497 0261 006C R                          DW OFFSET EXC_21
498 0263 0070 R                          DW OFFSET EXC_22
499 0265 0074 R                          DW OFFSET EXC_23
500 0267 0078 R                          DW OFFSET EXC_24
501 0269 007C R                          DW OFFSET EXC_25
502 026B 0080 R                          DW OFFSET EXC_26
503 026D 0084 R                          DW OFFSET EXC_27
504 026F 0088 R                          DW OFFSET EXC_28
505 0271 008C R                          DW OFFSET EXC_29
506 0273 0090 R                          DW OFFSET EXC_30
507 0275 0094 R                          DW OFFSET EXC_31
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                                SIDT_BLD ENDP
515
516 027D                                CODE ENDS
517 END
    
```



```

1          PAGE 118,121
2          TITLE TEST6 ---- 06/10/85 POST TESTS AND SYSTEM BOOT STRAP
3          .286C
4          .LIST
5          0000 CODE SEGMENT BYTE PUBLIC
6
7          PUBLIC BOOT_STRAP_1
8          PUBLIC POST6
9          PUBLIC STGTST_CNT
10         PUBLIC ROM_ERR
11         PUBLIC XMIT_8042
12
13         EXTRN CMOS_READ:NEAR
14         EXTRN DDS:NEAR
15         EXTRN DISK_BASE:NEAR
16         EXTRN E602:NEAR
17         EXTRN ERR_BEEP:NEAR
18         EXTRN E_MSG:NEAR
19         EXTRN F3A:NEAR
20         EXTRN PRT_SEG:NEAR
21
22         ASSUME CS:CODE,DS:DATA
23
24         0000 POST6 PROC NEAR
25         ;-----
26         ; THIS SUBROUTINE PERFORMS A READ/WRITE STORAGE TEST ON A BLOCK ;
27         ; OF STORAGE. ;
28         ; ENTRY REQUIREMENTS: ;
29         ; ES = ADDRESS OF STORAGE SEGMENT BEING TESTED ;
30         ; DS = ADDRESS OF STORAGE SEGMENT BEING TESTED ;
31         ; CX = WORD COUNT OF STORAGE BLOCK TO BE TESTED ;
32         ; EXIT PARAMETERS: ;
33         ; ZERO FLAG = 0 IF STORAGE ERROR (DATA COMPARE OR PARITY ;
34         ; CHECK). AL=0 DENOTES A PARITY CHECK. ELSE AL=XOR'ED ;
35         ; BIT PATTERN OF THE EXPECTED DATA PATTERN VS THE ACTUAL ;
36         ; DATA READ. ;
37         ; AX,BX,CX,DX,DI AND SI ARE ALL DESTROYED. ;
38         ;-----
39         STGTST_CNT PROC NEAR
40         MOV BX,CX ; SAVE WORD COUNT OF BLOCK TO TEST
41         IN AL,PORT_B
42         OR AL,RAM_PAR_OFF ; TOGGLE PARITY CHECK LATCHES
43         OUT PORT_B,AL ; TO RESET ANY PENDING ERROR
44         AND AL,RAM_PAR_ON
45         OUT PORT_B,AL
46
47         ;----- ROLL A BIT THROUGH THE FIRST WORD
48
49         000C 33 D2 XOR DX,DX ; CLEAR THE INITIAL DATA PATTERN
50         000E B9 0010 MOV CX,16 ; ROLL 16 BIT POSITIONS
51         0011 2B FF SUB DI,DI ; START AT BEGINNING OF BLOCK
52         0013 2B F6 SUB SI,SI ; INITIALIZE DESTINATION POINTER
53         0015 F9 STC ; SET CARRY FLAG ON FOR FIRST BIT
54
55         C1: RCL DX,1 ; MOVE BIT OVER LEFT TO NEXT POSITION
56         MOV [DI],DX ; STORE DATA PATTERN
57         MOV AX,[DI] ; GET THE DATA WRITTEN
58         XOR AX,DX ; INSURE DATA AS EXPECTED (CLEAR CARRY)
59         LOOPZ C1 ; LOOP TILL DONE OR ERROR
60
61         0020 75 66 JNZ C13 ; EXIT IF ERROR
62
63         ;----- CHECK CAS LINES FOR HIGH BYTE LOW BYTE
64
65         0022 BA FF00 MOV DX,0FF00H ; TEST DATA - AX= 0000H
66         0025 89 05 MOV [DI],AX ; STORE DATA PATTERN = 0000H
67         0027 88 75 01 MOV [DI+1],DH ; WRITE A BYTE OF FFH AT ODD LOCATION
68         002A 8B 05 MOV AX,[DI] ; GET THE DATA - SHOULD BE 0FF00H
69         002C 33 C2 XOR AX,DX ; CHECK THE FIRST WRITTEN
70         002E 75 58 JNZ C13 ; ERROR EXIT IF NOT ZERO
71
72         0030 89 05 MOV [DI],AX ; STORE DATA PATTERN OF 0000H
73         0032 8B 35 MOV [DI],DH ; WRITE A BYTE OF FFH AT EVEN LOCATION
74         0034 86 F2 XCHG DH,DL ; SET DX= 000FFFH AND BUS SETTLE
75         0036 8B 05 MOV AX,[DI] ; GET THE DATA
76         0038 33 C2 XOR AX,DX ; CHECK THE FIRST WRITTEN
77         003A 75 4C JNZ C13 ; EXIT IF NOT
78
79         ;----- CHECK FOR I/O OR BASE MEMORY ERROR
80
81         003C E4 61 IN AL,PORT_B ; CHECK FOR I/O - PARITY CHECK
82         003E 86 C4 XCHG AL,AH ; SAVE ERROR
83         0040 E4 87 IN AL,DMA_PAGE+6 ; CHECK FOR R/W OR I/O ERROR
84         0042 22 E0 AND AH,AL ; MASK FOR ERROR EXPECTED
85
86         ;----- PARITY ERROR EXIT
87
88         0044 B8 0000 MOV AX,0 ; RESTORE AX TO 0000
89         0047 75 3F JNZ C13 ; EXIT IF PARITY ERROR
90
91         0049 BA AA55 MOV DX,0AA55H ; WRITE THE INITIAL DATA PATTERN
92
93         C3: SUB DI,DI ; START AT BEGINNING OF BLOCK
94         SUB SI,SI ; INITIALIZE DESTINATION POINTER
95         MOV CX,BX ; SETUP BYTE COUNT FOR LOOP
96         MOV AX,DX ; GET THE PATTERN
97         REP STOSW ; STORE 64K BYTES (32K WORDS)
98         MOV CX,BX ; SET COUNT
99         SUB SI,SI ; START AT BEGINNING
100
101         C6: LODSW ; GET THE FIRST WRITTEN
102         XOR AX,DX ; INSURE DATA AS EXPECTED
103         LOOPZ C6 ; LOOP TILL DONE OR ERROR
104
105         005F 75 27 JNZ C13 ; EXIT IF NOT EXPECTED (ERROR BITS ON)
106
107         ;----- CHECK FOR I/O OR BASE MEMORY ERROR
108
109         0061 E4 61 IN AL,PORT_B ; CHECK FOR I/O -PARITY CHECK
110         0063 86 C4 XCHG AL,AH ; SAVE ERROR
111         0065 E4 87 IN AL,DMA_PAGE+6 ; CHECK FOR R/W OR I/O ERROR
112         0067 22 E0 AND AH,AL
113
114

```

SECTION 5


```

229                PAGE
230                ;-----
231                ; PRINT ADDRESS AND ERROR MESSAGE FOR ROM CHECKSUM ERRORS
232                ;-----
233                ROM_ERR PROC NEAR
234                PUSH DX
235                PUSH ES
236                PUSH AX
237                MOV AX,DATA
238                MOV ES,AX
239                POP AX
240                PUSH AX
241                MOV DX,DS
242                MOV ES:MFMF_ERR_FLAG,DH
243                JMP DX,0C800H
244                ROM_ERR BEEP
245                CALL PRT_SEG
246                MOV SI,OFFSET F3A
247                CALL E_MSG
248                ROM_ERR END
249                POP AX
250                POP ES
251                POP DX
252                RET
253                ROM_ERR BEEP:
254                MOV DX,0102H
255                CALL ERR_BEEP
256                JMP SHORT ROM_ERR_END
257                ROM_ERR ENDP
258                ;-----
259                ; THIS SUBROUTINE SENDS AN OUTPUT COMMAND TO THE KEYBOARD AND
260                ; RECEIVES THE KEYBOARD RESPONSE.
261                ; ENTRY REQUIREMENTS:
262                ; AL = COMMAND/DATA TO BE SENT
263                ; EXIT PARAMETERS:
264                ; ZERO FLAG = 1 IF ACK RECEIVED FROM THE KEY BOARD
265                ; AL = RESPONSE
266                ;-----
267                XMIT_8042 PROC NEAR
268                ;----- CHECK INPUT BUFFER FULL
269                ;-----
270                XCHG AH,AL
271                SUB CX,CX
272                XMITLOOP:
273                IN AL,STATUS_PORT
274                TEST AL,INPT_BUF_FULL
275                LOOPNZ XMITLOOP
276                JCXZ SHORT XMIT_EXIT
277                XCHG AH,AL
278                ;----- ISSUE THE COMMAND
279                OUT PORT_A,AL
280                SUB CX,CX
281                ;----- CHECK OUTPUT BUFFER FULL
282                ;-----
283                XMIT_1: IN AL,STATUS_PORT
284                MOV AH,AL
285                TEST AL,OUT_BUF_FULL
286                JZ XMIT_2
287                IN AL,PORT_A
288                TEST AH,INPT_BUF_FULL
289                LOOPNZ XMIT_1
290                JNZ SHORT XMIT_EXIT
291                ;----- CHECK OUTPUT BUFFER FULL
292                MOV BL,6
293                SUB CX,CX
294                XMIT_3: IN AL,STATUS_PORT
295                TEST AL,OUT_BUF_FULL
296                LOOPZ XMIT_3
297                JNZ XMIT_4
298                DEC BL
299                JNZ SHORT XMIT_3
300                INC BL
301                JMP SHORT XMIT_EXIT
302                ;----- GET THE DATA
303                XMIT_4: SUB CX,CX
304                XMIT_5: LOOP XMIT_5
305                IN AL,PORT_A
306                SUB CX,01H
307                XMIT_EXIT: RET
308                XMIT_8042 ENDP
309                ;-----
310                ;--- BOOT STRAP -- INT 19 H -----
311                ; BOOT STRAP LOADER
312                TRACK 0, SECTOR 1 IS READ INTO THE
313                BOOT LOCATION (SEGMENT 0 OFFSET 7C00)
314                AND CONTROL IS TRANSFERRED THERE.
315                ;
316                ; IF THERE IS A HARDWARE ERROR CONTROL IS
317                ; TRANSFERRED TO THE ROM BASIC ENTRY POINT
318                ;-----
319                ASSUME CS:CODE,DS:ABS0,ES:ABS0
320                BOOT_STRAP_1 PROC NEAR
321                MOV AX,ABS0
322                MOV DS,AX
323                MOV ES,AX
324                ;----- RESET THE DISK PARAMETER TABLE VECTOR
325                MOV WORD PTR @DISK_POINTER, OFFSET DISK_BASE
326                MOV WORD PTR @DISK_POINTER+2,CS
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342

```

SECTION 5


```

1 PAGE 118,121
2 TITLE DSKETTE -- 06/10/85 DSKETTE BIOS
3 .286C
4 .LIST
5 0000 CODE SEGMENT BYTE PUBLIC
6
7 PUBLIC DISK_INT_1
8 PUBLIC SEEK
9 PUBLIC DSKETTE_SETUP
10 PUBLIC DSKETTE_IO_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 ; DSKETTE PARAMETER TABLE LOCATION
15 EXTRN WAITF:NEAR ; FIXED WAIT ROUTINE - (CX)*15,086 US
16
17 ;--- INT 13H ---
18 ; DSKETTE I/O
19 THIS INTERFACE PROVIDES ACCESS TO THE 5 1/4" DSKETTE DRIVES
20 ; 320/360K DSKETTE DRIVES AND 1.2M DSKETTE DRIVES SUPPORTED
21 INPUT
22 (AH)= 00H RESET DSKETTE SYSTEM
23 HARD RESET TO NEC, PREPARE COMMAND, RECALIBRATE REQUIRED
24 ON ALL DRIVES
25
26 ;---
27 (AH)= 01H READ THE STATUS OF THE SYSTEM INTO (AH)
28 ; *DSKETTE_STATUS FROM LAST OPERATION IS USED
29
30 ; REGISTERS FOR READ/WRITE/VERIFY/FORMAT
31 (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)
32 (DH) - HEAD NUMBER (0-1 ALLOWED, NOT VALUE CHECKED)
33 (CH) - TRACK NUMBER (NOT VALUE CHECKED)
34 MEDIA DRIVE TRACK NUMBER
35 320/360 320/360 0-39
36 320/360 1.2M 0-39
37 1.2M 1.2M 0-79
38 720K 720K 0-79
39 (CL) - SECTOR NUMBER (NOT VALUE CHECKED, NOT USED FOR FORMAT)
40 MEDIA DRIVE SECTOR NUMBER
41 320/360 320/360 1-8/9
42 320/360 1.2M 1-8/9
43 1.2M 1.2M 1-15
44 720K 720K 1-9
45 (AL) - NUMBER OF SECTORS (NOT VALUE CHECKED)
46 MEDIA DRIVE MAX NUMBER OF SECTORS
47 320/360 320/360 8/9
48 320/360 1.2M 8/9
49 1.2M 1.2M 15
50 720K 720K 9
51 (ES:BX) - ADDRESS OF BUFFER (REQUIRED FOR VERIFY)
52
53 ;---
54 (AH)= 02H READ THE DESIRED SECTORS INTO MEMORY
55
56 ;---
57 (AH)= 03H WRITE THE DESIRED SECTORS FROM MEMORY
58
59 ;---
60 (AH)= 04H VERIFY THE DESIRED SECTORS
61
62 ;---
63 (AH)= 05H FORMAT THE DESIRED TRACK
64 FOR THE FORMAT OPERATION, THE BUFFER POINTER (ES,BX) MUST
65 POINT TO THE COLLECTION OF DESIRED ADDRESS FIELDS FOR THE
66 TRACK. EACH FIELD IS COMPOSED OF 4 BYTES, (C,H,R,N), WHERE
67 C = TRACK NUMBER, H=HEAD NUMBER, R = SECTOR NUMBER, N= NUMBER
68 OF BYTES PER SECTOR (0=128, 01=256, 02=512, 03=1024,)
69 THERE MUST BE ONE ENTRY FOR EVERY SECTOR ON THE TRACK.
70 THIS INFORMATION IS USED TO FIND THE REQUESTED SECTOR DURING
71 READ/WRITE ACCESS.
72 PRIOR TO FORMATTING A DISKETTE, IF THERE EXISTS MORE THAN
73 ONE SUPPORTED MEDIA FORMAT TYPE WITHIN THE DRIVE IN QUESTION,
74 THEN "SET DASD TYPE" (INT 13H, AH = 17H) MUST BE CALLED TO
75 SET THE DISKETTE TYPE THAT IS TO BE FORMATTED. IF "SET DASD
76 TYPE" IS NOT CALLED, THE FORMAT ROUTINE WILL ASSUME THE
77 MEDIA FORMAT TO BE THE MAXIMUM CAPACITY OF THE DRIVE.
78 IN ORDER TO FORMAT 320/360K MEDIA IN EITHER A 320/360K OR
79 1.2M DISKETTE DRIVE THE GAP LENGTH FOR FORMAT PARAMETER
80 OF DISK BASE MUST BE CHANGE TO 050H. ALSO THE EOT
81 PARAMETER (LAST SECTOR ON TRACK) MUST BE SET TO THE
82 DESIRED NUMBER OF SECTORS/TRACK - 8 FOR 320K, 9 FOR 360K.
83 DISK_BASE IS POINTED TO BY DISK POINTER LOCATED AT
84 ABSOLUTE ADDRESS 0178.
85 WHEN 320/360K FORMAT OPERATIONS ARE COMPLETE, THE PARAMETERS
86 SHOULD BE RESTORED TO THEIR RESPECTIVE INITIAL VALUES.
87
88 ;---
89 (AH)= 08H READ DRIVE PARAMETERS
90 REGISTERS
91 INPUT
92 (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)
93 OUTPUT
94 (ES:DI) POINTS TO DISK BASE
95 (CH) - LOW ORDER 8 OF 10 BITS MAXIMUM NUMBER OF TRACKS
96 (CL) - BITS 7 & 6 - HIGH ORDER TWO BITS OF MAXIMUM TRACKS
97 - BITS 5 THRU 0 - MAXIMUM SECTORS PER TRACK
98 (DH) - MAXIMUM HEAD NUMBER
99 (DL) - NUMBER OF DISKETTE DRIVES INSTALLED
100 (BH) = 0
101 (BL) - BITS 7 THRU 4 - 0
102 (AX) - 0
103 (AX) - BITS 3 THRU 0 - VALID DRIVE TYPE VALUE IN CMOS
104 UNDER THE FOLLOWING CIRCUMSTANCES:
105 (1) THE DRIVE NUMBER IS INVALID,
106 (2) THE DRIVE TYPE IS UNKNOWN AND CMOS IS NOT PRESENT,
107 (3) THE DRIVE TYPE IS UNKNOWN AND CMOS IS BAD,
108 (4) OR THE DRIVE TYPE IS UNKNOWN AND THE CMOS DRIVE TYPE IS INVALID
109 THEN ES,AX,BX,CX,DH,D1=0 ; DL=NUMBER OF DRIVES.
110 IF NO DRIVES ARE PRESENT THEN: ES,AX,BX,CX,DX,D1=0.
111 *DSKETTE_STATUS = 0 AND CY IS RESET.
    
```

SECTION 5

```

108 PAGE
109 -----
110 (AH) = 15H READ DASD TYPE
111 REGISTERS
112 (AH) - ON RETURN IF CARRY FLAG NOT SET, OTHERWISE ERROR
113         00 - DRIVE NOT PRESENT
114         01 - DISKETTE, NO CHANGE LINE AVAILABLE
115         02 - DISKETTE, CHANGE LINE AVAILABLE
116         03 - FIXED DISK
117 (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)
118 -----
119 (AH) = 16H DISK CHANGE LINE STATUS
120 REGISTERS
121 (AH)=00 - DISK CHANGE LINE NOT ACTIVE
122         06 - DISK CHANGE LINE ACTIVE & CARRY BIT ON
123 (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)
124 -----
125 (AH) = 17H SET DASD TYPE FOR FORMAT
126 REGISTERS
127 (AL) - 00 - NOT USED
128         01 - DISKETTE 320/360K IN 360K DRIVE
129         02 - DISKETTE 360K IN 1.2M DRIVE
130         03 - DISKETTE 1.2M IN 1.2M DRIVE
131         04 - DISKETTE 720K IN 720K DRIVE
132 (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED);
133         DD NOT USE WHEN DISKETTE ATTACH CARD USED)
134 -----
135 DISK CHANGE STATUS IS ONLY CHECKED WHEN A 1.2M BYTE DISKETTE
136 DRIVE IS SPECIFIED. IF THE DISK CHANGE LINE IS FOUND TO BE
137 ACTIVE THE FOLLOWING ACTIONS TAKE PLACE:
138 ATTEMPT TO RESET DISK CHANGE LINE TO INACTIVE STATE.
139 IF ATTEMPT SUCCEEDS SET DASD TYPE FOR FORMAT AND RETURN DISK
140 CHANGE ERROR CODE
141 IF ATTEMPT FAILS RETURN TIMEOUT ERROR CODE AND SET DASD TYPE
142 TO A PREDETERMINED STATE INDICATING MEDIA TYPE UNKNOWN.
143 IF THE DISK CHANGE LINE IN INACTIVE PERFORM SET DASD TYPE FOR FORMAT.
144 -----
145 DATA VARIABLE -- #DISK POINTER
146 DOUBLE WORD POINTER TO THE CURRENT SET OF DISKETTE PARAMETERS
147 -----
148 OUTPUT FOR ALL FUNCTIONS
149 AH = STATUS OF OPERATION
150 STATUS BITS ARE DEFINED IN THE EQUATES FOR #DISKETTE_STATUS
151 VARIABLE IN THE DATA SEGMENT OF THIS MODULE
152 CY = 0 SUCCESSFUL OPERATION (AH=0 ON RETURN, EXCEPT FOR READ DASD
153 TYPE AH=(15)).
154 CY = 1 FAILED OPERATION (AH HAS ERROR REASON)
155 FOR READ/WRITE/VERIFY
156 DS,BX,DX,CX PRESERVED
157 NOTE: IF AN ERROR IS REPORTED BY THE DISKETTE CODE, THE APPROPRIATE
158 ACTION IS TO RESET THE DISKETTE, THEN RETRY THE OPERATION.
159 ON READ ACCESSES, NO MOTOR START DELAY IS TAKEN, SO THAT
160 THREE RETRIES ARE REQUIRED ON READS TO ENSURE THAT THE
161 PROBLEM IS NOT DUE TO MOTOR START-UP.
162 -----
163 .LIST
164 DISKETTE STATE MACHINE - ABSOLUTE ADDRESS 40:90 (DRIVE A) & 91 (DRIVE B)
165 .LIST
166
167 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
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 | | | | | | | | |
    
```

```

205 PAGE
206 ASSUME CS:CODE,DS:DATA,ES:DATA
207
208 .LIST DISKETTE_10_1 PROC FAR
209 0000 FB STI ;>>> ENTRY POINT FOR ORG 0EC59H
210 0001 55 PUSH BP ; INTERRUPTS BACK ON
211 0002 57 PUSH D1 ; USER REGISTER
212 0003 52 PUSH DX ; HEAD #, DRIVE # OR USER REGISTER
213 0004 53 PUSH BX ; BUFFER OFFSET PARAMETER OR REGISTER
214 0005 51 PUSH CX ; TRACK #=SECTOR # OR USER REGISTER
215 0006 8B EC MOV BP,SP ; BP => PARAMETER LIST DEP. ON AH
216 ; [BP] = SECTOR #
217 ; [BP+1] = TRACK #
218 ; [BP+2] = BUFFER OFFSET
219 ; FOR RETURN OF DRIVE PARAMETERS:
220 ; CL/[BP+1] = BITS 7&6 HI BITS OF MAX CYL
221 ; CH/[BP+1] = LOW 8 BITS OF MAX CYL.
222 ; BL/[BP+2] = BITS 7-4 = 0
223 ; ;
224 ; BITS 3-0 = VALID CMOS TYPE
225 ; BH/[BP+3] = 0
226 ; DH/[BP+5] = # DRIVES INSTALLED
227 ; DI/[BP+6] = OFFSET TO DISK BASE
228 ; BUFFER SEGMENT PARM OR USER REGISTER
229 0008 1E PUSH DS ; USER REGISTERS
230 0009 56 PUSH SI ; SEGMENT OF BIOS DATA AREA TO DS
231 000A E8 0000 E CALL DDS ; CHECK FOR > LARGEST FUNCTION
232 000D 80 FC 18 CMP AH,(FNC_TAE-FNC_TAB)/2 ; FUNCTION OK
233 0010 72 02 JB OK_FUNC
234
235 0012 B4 14 MOV AH,14H ; REPLACE WITH KNOWN INVALID FUNCTION
236 0014
237 0014 80 FC 01 OK_FUNC: ; RESET OR STATUS ?
238 0017 76 0C JBE OK_DRV ; IF RESET OR STATUS DRIVE ALWAYS OK
239 0019 80 FC 08 CMP AH,8 ; READ DRIVE PARMS ?
240 001C 74 07 JZ OK_DRV ; IF SO DRIVE CHECKED LATER
241 001E 80 FA 01 CMP DL,1 ; DRIVES 0 AND 1 OK
242 0021 76 02 JBE OK_DRV ; IF 0 AND 1 THEN JUMP
243 0023 B4 14 MOV AH,14H ; REPLACE WITH KNOWN INVALID FUNCTION
244 0025
245 0025 84 CC OK_DRV: ; CL = FUNCTION
246 0027 32 ED XOR CH,CH ; CX = FUNCTION
247 0029 D0 E1 SHL CL,1 ; FUNCTION TIMES 2
248 002B BB 004E R MOV BX,OFFSET FNC_TAB ; LOAD START OF FUNCTION TABLE
249 002E 03 D9 ADD ; ADD OFFSET INTO TABLE => ROUTINE
250 0030 8A E6 MOV AH,DH ; AX = HEAD #, # OF SECTORS OR DASD TYPE
251 0032 32 F6 XOR DH,DH ; DX = DRIVE #
252 0034 8B F0 MOV SI,AX ; SI = HEAD #, # OF SECTORS OR DASD TYPE
253 0036 8B FA MOV DI,DX ; DI = DRIVE #
254 0038 8A 26 0041 R MOV AH,#DISKETTE_STATUS ; LOAD STATUS TO AH FOR STATUS FUNCTION
255 003C C6 06 0041 R 00 MOV #DISKETTE_STATUS,0 ; INITIALIZE FOR ALL OTHERS
256
257 ; THROUGHOUT THE DISKETTE BIOS, THE FOLLOWING INFORMATION IS CONTAINED IN
258 ; THE FOLLOWING MEMORY LOCATIONS AND REGISTERS. NOT ALL DISKETTE BIOS
259 ; FUNCTIONS REQUIRE ALL OF THESE PARAMETERS.
260
261 ; DI : DRIVE #
262 ; SI-HI : HEAD #
263 ; SI-LOW : # OF SECTORS OR DASD TYPE FOR FORMAT
264 ; ES : BUFFER SEGMENT
265 ; [BP] : SECTOR #
266 ; [BP+1] : TRACK #
267 ; [BP+2] : BUFFER OFFSET
268
269 ; ACROSS CALLS TO SUBROUTINES THE CARRY FLAG (CY=1), WHERE INDICATED IN
270 ; SUBROUTINE PROLOGUES, REPRESENTS AN EXCEPTION RETURN (NORMALLY AN ERROR
271 ; CONDITION). IN MOST CASES, WHEN CY = 1, #DISKETTE_STATUS CONTAINS THE
272 ; SPECIFIC ERROR CODE.
273
274 0041 2E1 FF 17 CALL WORD PTR CS:[BX] ; (AH) = #DISKETTE_STATUS
275 0044 5E POP SI ; CALL THE REQUESTED FUNCTION
276 0045 1F POP DS ; RESTORE ALL REGISTERS
277 0046 59 POP CX
278 0047 5B POP BX
279 0048 5A POP DX
280 0049 5F POP DI
281 004A 5D POP BP
282 004B CA 0002 RET 2 ; THROW AWAY SAVED FLAGS
283
284
285 004E 007E R FNC_TAB DW DISK RESET ; AH = 00; RESET
286 0050 00E8 R DW DISK_STATUS ; AH = 01; STATUS
287 0052 00F4 R DW DISK_READ ; AH = 02; READ
288 0054 0100 R DW DISK_WRITE ; AH = 03; WRITE
289 0056 010C R DW DISK_VERIFY ; AH = 04; VERIFY
290 0058 0118 R DW DISK_FORMAT ; AH = 05; FORMAT
291 005A 016A R DW FNC_ERR ; AH = 06; INVALID
292 005C 016A R DW FNC_ERR ; AH = 07; INVALID
293 005E 0174 R DW DISK_PARMS ; AH = 08; READ DRIVE PARAMETERS
294 0060 016A R DW FNC_ERR ; AH = 09; INVALID
295 0062 016A R DW FNC_ERR ; AH = 0A; INVALID
296 0064 016A R DW FNC_ERR ; AH = 0B; INVALID
297 0066 016A R DW FNC_ERR ; AH = 0C; INVALID
298 0068 016A R DW FNC_ERR ; AH = 0D; INVALID
299 006A 016A R DW FNC_ERR ; AH = 0E; INVALID
300 006C 016A R DW FNC_ERR ; AH = 0F; INVALID
301 006E 016A R DW FNC_ERR ; AH = 10; INVALID
302 0070 016A R DW FNC_ERR ; AH = 11; INVALID
303 0072 016A R DW FNC_ERR ; AH = 12; INVALID
304 0074 016A R DW FNC_ERR ; AH = 13; INVALID
305 0076 016A R DW FNC_ERR ; AH = 14; INVALID
306 0078 021A R DW DISK_TYPE ; AH = 15; READ DASD TYPE
307 007A 023C R DW DISK_CHANGE ; AH = 16; CHANGE STATUS
308 007C 0267 R DW FORMAT_SET ; AH = 17; SET DASD TYPE
309 = 007E EQU $ ; END
310
311 007E DISKETTE_10_1 ENDP
    
```

```

312                                     PAGE
313 -----
314 ; DISK_RESET:  RESET THE DISKETTE SYSTEM.
315 ;
316 ; ON EXIT:      *DISKETTE_STATUS, CY REFLECT STATUS OF OPERATION
317 -----
318 00TE          PROC NEAR
319 00TE BA 03F2  MOV     DX,03F2H          ; ADAPTER CONTROL PORT
320 00B1 FA          CLI                    ; NO INTERRUPTS
321 00B2 A0 003F R  MOV     AL,*MOTOR_STATUS    ; GET DIGITAL OUTPUT REGISTER REFLECTION
322 00B5 24 3F      AND     AL,00111111B   ; KEEP SELECTED AND MOTOR ON BITS
323 00B7 C0 C0 04  ROL     AL,4          ; MOTOR VALUE TO HIGH NIBBLE
324                                     ; DRIVE SELECT TO LOW NIBBLE
325 00BA 0C 08      OR      AL,00001000B       ; TURN ON INTERRUPT ENABLE
326 00BC EE          OUT     DX,AL          ; RESET THE ADAPTER
327 00BD C6 06 003E R 00  MOV     MOV     *SEEK_STATUS,0    ; SET RECALIBRATE REQUIRED ON ALL DRIVES
328 0097 EB 00      JMB     *+2          ; WAIT FOR I/O
329 0094 0C 04      OR      AL,00000100B       ; TURN OFF RESET BIT
330 0096 EE          OUT     DX,AL          ; RESET THE ADAPTER
331 0097 FB          STI                    ; ENABLE THE INTERRUPTS
332 0098 E8 087C R  CALL    WAIT_INT          ; WAIT FOR THE INTERRUPT
333 009B 72 44      JC      DR_ERR          ; IF ERROR, RETURN IT
334 009D B9 00C0    MOV     CX,11000000B       ; CL = EXPECTED *NEC_STATUS
335 -----
336 NXT_DRV:
337 00A0 51          PUSH   CX          ; SAVE FOR CALL
338 00A1 B8 00E0 R  MOV     AX,OFFSET DR_POP_ERR ; LOAD NEC_OUTPUT ERROR ADDRESS
339 00A4 50          PUSH   AX          ;
340 00A5 B4 08      MOV     AH,08H        ; SENSE INTERRUPT STATUS COMMAND
341 00A7 E8 07BD R  CALL    NEC_OUTPUT     ; THROW AWAY ERROR RETURN
342 00AA 58          POP     AX          ; READ IN THE RESULTS
343 00AB E8 08A4 R  CALL    RESULTS        ; RESTORE AFTER CALL
344 00AE 59          POP     CX          ;
345 00AF 72 30      JC      DR_ERR          ; ERROR RETURN
346 00B1 3A 0E 0042 R  CMP     CL,*NEC_STATUS    ; TEST FOR DRIVE READY TRANSITION
347 00B5 75 2A      JNZ     DR_ERR          ; EVERYTHING OK
348 00B7 FE C1      INC     CL          ; NEXT EXPECTED *NEC_STATUS
349 00B9 80 F9 C3  CMP     CL,11000011B     ; ALL POSSIBLE DRIVES CLEARED
350 00BC 76 E2      JBE     NXT_DRV        ; FALL THRU IF 11000100B OR >
351 -----
352 I----- SEND SPECIFY COMMAND TO NEC
353 -----
354 00BE B8 00D8 R  MOV     AX,OFFSET RESBAC   ; LOAD ERROR ADDRESS
355 00C1 50          PUSH   AX          ; PUSH NEC_OUT ERROR RETURN
356 00C2 B4 03      MOV     AH,03H          ; SPECIFY COMMAND
357 00C4 E8 07BD R  CALL    NEC_OUTPUT     ; OUTPUT THE COMMAND
358 00C7 2A D2      DL     DL            ; FIRST SPECIFY BYTE
359 00C9 E8 06CC R  CALL    GET_PARM        ; GET PARAMETER TO AH
360 00CC E8 07BD R  CALL    NEC_OUTPUT     ; OUTPUT THE COMMAND
361 00CF B2 01      MOV     SUB            ; SECOND SPECIFY BYTE
362 00D1 E8 06CC R  CALL    GET_PARM        ; GET PARAMETER TO AH
363 00D4 E8 07BD R  CALL    NEC_OUTPUT     ; OUTPUT THE COMMAND
364 00D7 58          POP     AX          ; POP ERROR RETURN
365 00D8          RESBAC:
366 00D8 E8 0620 R  CALL    SETUP_END       ; VARIOUS CLEANUPS
367 00DB 8B DE      MOV     BX,S1          ; GET SAVED AL TO BL
368 00DD 8A C3      MOV     AL,BL          ; PUT BACK FOR RETURN
369 00DF C9          RET
370 -----
371 00E0          DR_POP_ERR:
372 00E0 59          POP     CX          ; CLEAR STACK
373 00E1          DR_ERR:
374 00E1 80 0E 0041 R 20  OR     *DISKETTE_STATUS,BAD_NEC ; SET ERROR CODE
375 00E6 EB F0      JMP     SHORT RESBAC   ; RETURN FROM RESET
376 00E8          DISK_RESET
377          ENDP
378 -----
379 ; DISK_STATUS:  DISKETTE STATUS.
380 ;
381 ; ON ENTRY:      AH : STATUS OF PREVIOUS OPERATION
382 ;
383 ; ON EXIT:      AH, *DISKETTE_STATUS, CY REFLECT STATUS OF PREVIOUS OPERATION.
384 -----
385 00E8          DISK_STATUS
386 00E8 88 26 0041 R  MOV     *DISKETTE_STATUS,AH ; PUT BACK FOR SETUP_END
387 00EC E8 0620 R  CALL    SETUP_END       ; VARIOUS CLEANUPS
388 00EF 8B DE      MOV     BX,S1          ; GET SAVED AL TO BL
389 00F1 8A C3      MOV     AL,BL          ; PUT BACK FOR RETURN
390 00F3 C3          RET
391 00F4          DISK_STATUS
392          ENDP
393 -----
394 ; DISK_READ:    DISKETTE READ.
395 ;
396 ; ON ENTRY:      DI : DRIVE #
397 ;                SI-HI : HEAD #
398 ;                SI-LOW : # OF SECTORS
399 ;                ES : BUFFER SEGMENT
400 ;                [BP] : SECTOR #
401 ;                [BP+1] : TRACK #
402 ;                [BP+2] : BUFFER OFFSET
403 ;
404 ; ON EXIT:      *DISKETTE_STATUS, CY REFLECT STATUS OF OPERATION
405 -----
406 00F4          DISK_READ
407 00F4 80 26 003F R 7F  AND     *MOTOR_STATUS,01111111B ; INDICATE A READ OPERATION
408 00F9 B8 E646    MOV     AX,0E646H        ; AX = NEC COMMAND, DMA COMMAND
409 00FC E8 035C R  CALL    RD_WR_VF        ; COMMON READ/WRITE/VERIFY
410 0100          RET
411 -----
412 ; DISK_WRITE:   DISKETTE WRITE.
413 ;
414 ; ON ENTRY:      DI : DRIVE #
415 ;                SI-HI : HEAD #
416 ;                SI-LOW : # OF SECTORS
417 ;                ES : BUFFER SEGMENT
418 ;                [BP] : SECTOR #
419 ;                [BP+1] : TRACK #
420 ;                [BP+2] : BUFFER OFFSET
421 ;
422 ; ON EXIT:      *DISKETTE_STATUS, CY REFLECT STATUS OF OPERATION
423 -----
424 0100          DISK_WRITE
425 0100 B8 C54A    MOV     AX,0C54AH        ; AX = NEC COMMAND, DMA COMMAND
    
```



```

426 0103 80 0E 003F R 80          OR      #MOTOR_STATUS,10000000B ; INDICATE WRITE OPERATION
427 0108 E8 035C R              CALL    RD_WR_VF          ; COMMON READ/WRITE/VERIFY
428 010B C3                    DISK_WRITE
429 010C                        ENDP
430
431 ;-----
431 ; DISK_VERIFY: DISKETTE VERIFY.
432 ;
433 ;
434 ; ON ENTRY:  DI      : DRIVE #
435 ;           SI-HI  : HEAD #
436 ;           SI-LOW : # OF SECTORS
437 ;           ES     : BUFFER SEGMENT
438 ;           [BP]   : SECTOR #
439 ;           [BP+1] : TRACK #
440 ;           [BP+2] : BUFFER OFFSET
441 ;
442 ; ON EXIT:   #DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
443 ;-----
443 010C                        DISK_VERIFY  PROC  NEAR
444 010C 80 26 003F R 7F          AND     #MOTOR_STATUS,01111111B ; INDICATE A READ OPERATION
445 0111 B8 E642                MOV     AX,0E642H           ; AX = NEC COMMAND, DMA COMMAND
446 0114 E8 035C R              CALL    RD_WR_VF          ; COMMON READ/WRITE/VERIFY
447 0117 C3                    DISK_VERIFY
448 0118                        ENDP
449
450 ;-----
450 ; DISK_FORMAT: DISKETTE FORMAT.
451 ;
452 ;
453 ; ON ENTRY:  DI      : DRIVE #
454 ;           SI-HI  : HEAD #
455 ;           SI-LOW : # OF SECTORS
456 ;           ES     : BUFFER SEGMENT
457 ;           [BP]   : SECTOR #
458 ;           [BP+1] : TRACK #
459 ;           [BP+2] : BUFFER OFFSET
460 ;
461 ; ON EXIT:   #DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
462 ;-----
462 0118                        DISK_FORMAT  PROC  NEAR
463 0118 E8 02C8 R              CALL    XLAT_NEW          ; TRANSLATE STATE TO PRESENT ARCH.
464 011B E8 03CE R              CALL    FMT_INIT         ; ESTABLISH STATE IF UNESTABLISHED.
465 011E 80 0E 003F R 80      OR      #MOTOR_STATUS,10000000B ; INDICATE WRITE OPERATION
466 0123 E8 0416 R              CALL    MED_CHANGE       ; CHECK MEDIA CHANGE AND RESET IF SO
467 0126 72 37                JC     FM_DON             ; MEDIA CHANGED, SKIP
468 0128 E8 0451 R              CALL    SEND_RATE        ; SEND DATA RATE TO CONTROLLER
469 012B 80 4A                MOV     AL,04AH           ; WILL WRITE TO THE DISKETTE
470 012D E8 0471 R              CALL    DMA_SETUP        ; SET UP THE DMA
471 0130 72 2D                JC     FM_DON             ; RETURN WITH ERROR
472 0132 B4 4D                MOV     AH,04DH          ; ESTABLISH THE FORMAT COMMAND
473 0134 E8 04C7 R              CALL    NEC_INIT         ; INITIALIZE THE NEC
474 0137 B8 015F R              MOV     AX,OFFSET_FM_DON ; LOAD ERROR ADDRESS
475 013A 50                    PUSH    AX                ; PUSH NEC OUT ERROR RETURN
476 013B B2 03                MOV     DL,3             ; BYTES/SECTOR VALUE TO NEC
477 013D E8 06CC R              CALL    GET_PARM         ; GET PARM
478 0140 E8 07BD R              CALL    NEC_OUTPUT       ; NEC OUTPUT
479 0143 B2 04                MOV     DL,4             ; SECTORS/TRACK VALUE TO NEC
480 0145 E8 06CC R              CALL    GET_PARM         ; GET PARM
481 0148 E8 07BD R              CALL    NEC_OUTPUT       ; NEC OUTPUT
482 014B B2 07                MOV     DL,7             ; GAP LENGTH VALUE TO NEC
483 014D E8 06CC R              CALL    GET_PARM         ; GET PARM
484 0150 E8 07BD R              CALL    NEC_OUTPUT       ; NEC OUTPUT
485 0153 B2 08                MOV     DL,8             ; FILLER BYTE TO NEC
486 0155 E8 06CC R              CALL    GET_PARM         ; GET PARM
487 0158 E8 07BD R              CALL    NEC_OUTPUT       ; NEC OUTPUT
488 015B 58                    POP     AX                ; THROW AWAY ERROR
489 015C E8 0530 R              CALL    AX               ; TERMINATE, RECEIVE STATUS, ETC.
490 015F                        FM_DON:
491 015F E8 02EE R              CALL    XLAT_OLD         ; TRANSLATE STATE TO COMPATIBLE MODE
492 0162 E8 0620 R              CALL    SETUP_END        ; VARIOUS CLEANUPS
493 0165 B8 01                MOV     BX,SI            ; GET SAVED AL TO BL
494 0167 8A C3                MOV     AL,BL            ; PUT BACK FOR RETURN
495 0169 C3                    RET
496 016A                        DISK_FORMAT  ENDP
497
498 ;-----
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                        FNC_ERR  PROC  NEAR
504 016A 8B C6                MOV     AL,BX            ; INVALID FUNCTION REQUEST
505 016C B4 01                MOV     AH,BAD_CMD      ; RESTORE AL
506 016E 88 26 0041 R          MOV     #DSKETTE_STATUS,AH ; SET BAD COMMAND ERROR
507 0172 F9                STC                     ; STORE IN DATA AREA
508 0173 C3                    RET                       ; SET CARRY INDICATING ERROR
509 0174                        FNC_ERR  ENDP
510
511 ;-----
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 ;           CH/[BP+1] = LOW 8 BITS OF MAX CYLINDER
517 ;           BL/[BP+2] = BITS 7-4 = 0
518 ;           BH/[BP+3] = BITS 3-0 = VALID CMOS DRIVE TYPE
519 ;           DL/[BP+4] = # DRIVES INSTALLED (VALUE CHECKED)
520 ;           DH/[BP+5] = MAX HEAD #
521 ;           DI/[BP+6] = OFFSET OF DISK_BASE
522 ;           ES       = SEGMENT OF DISK_BASE
523 ;           AX       = 0
524 ;
525 ; NOTE : THE ABOVE INFORMATION IS STORED IN THE USERS STACK AT
526 ; THE LOCATIONS WHERE THE MAIN ROUTINE WILL POP THEM
527 ; INTO THE APPROPRIATE REGISTERS BEFORE RETURNING TO THE
528 ; CALLER.
529 ;
530 ;-----
531 0174                        DISK_PARAMS  PROC  NEAR
532 0174 81 FF 0080            CMP     DI,80H           ; CHECK FOR FIXED MEDIA TYPE REQUEST
533 0178 72 06                JB     DISK_P2           ; CONTINUE IF NOT REQUEST FALL THROUGH
534
535 ;-----
535 ; FIXED DISK REQUEST FALL THROUGH ERROR
536 ;
537 ; ON EXIT:   AX,SI : RESTORE AL WITH CALLERS VALUE
538 ;           AH,BAD_CMD : SET BAD COMMAND ERROR IN (AH)
539 017C B4 01                MOV     AH,BAD_CMD
540

```

```

540 017E F9          STC
541 017F C3          RET
542
543 0180
544 0180 EB 02C8 R   DISK_P2:
545 0183 C7 46 02 0000 CALL XLAT_NEW          ; TRANSLATE STATE TO PRESENT ARCH.
546 0188 A1 0010 R   MOV WORD_PTR [BP+2],0 ; DRIVE TYPE = 0
547 018B 24 C1       MOV AX,=EQUIP_FLAG   ; LOAD EQUIPMENT FLAG FOR # DISKETTES
548 018D B2 02       AND AL,1000001B     ; KEEP DISKETTE DRIVE BITS
549 018F 3C 41       CMP DL,2             ; DISKETTE DRIVES = 2
550 0191 74 06       JZ DISK_P3          ; 2 DRIVES INSTALLED ?
551
552 0193 FE CA       DEC DL               ; IF YES JUMP
553 0195 3C 01       CMP AL,00000001B   ; DISKETTE DRIVES = 1
554 0197 75 6A       JNZ DISK_P8        ; 1 DRIVE INSTALLED ?
555 0199
556 0199 88 56 04    DISK_P3:
557 019C 83 FF 01    MOV [BP+4],DL
558 019F 77 66       CMP D1,1
559 01A1 C6 46 05 01 MOV DISK_P9
560 01A5 EB 06B3 R   CALL BYTE_PTR[BP+5],1 ; CHECK FOR VALID DRIVE
561 01A8 72 18       JC DISK_P4          ; DRIVE INVALID
562 01AA 0A 0A       OR AL,AL            ; MAXIMUM HEAD NUMBER = 1
563 01AC 74 14       JZ DISK_P4          ; RETURN DRIVE TYPE IN AL
564 01AE 3C 03       CMP AL,[DR_PTE-DR_PT]/2 ; IF CMOS BAD CHECKSUM ESTABLISHED
565 01B0 77 10       JA DISK_P4          ; TEST FOR NO DRIVE TYPE
566
567 01B2 88 46 02    MOV [BP+2],AL
568 01B5 FE C8       DEC AL              ; JUMP IF SO
569 01B7 D0 E0       SHL AL,1           ; STORE VALID CMOS DRIVE TYPE
570 01B9 8A DA       MOV BL,AL           ; MAKE 0 ORIGIN
571 01BB 32 F7       XOR BH,BH           ; ACCOUNT FOR FIELD WIDTH
572 01BD 2E: 8B 8F 0214 R MOV CX,WORD_PTR DR_PT[2] ; FINISH MAKING INDEX POINTER
573
574 01C2
575 01C2 8A 85 0090 R DISK_P4:
576 01C6 A8 10       MOV AL,=DSK_STATE[D1] ; LOAD STATE FOR THIS DRIVE
577 01C8 75 08       TEST AL,=MED_DET   ; CHECK FOR ESTABLISHED STATE
578 01CA 80 7C 02 00 JNZ DISK_P5         ; GO TO CMOS FOR DRIVE CHECK
579 01CC E2 73 07    MOV [BP+2],1
580 01DD EB 1C       JMP DISK_P9         ; CHECK FOR CMOS BAD/INVALID
581
582 01D2
583 01D2 24 C0       AND AL,RATE_MSK    ; CMOS BAD/INVALID AND UNESTABLISHED
584 01D4 2E: 8B 0E 0216 R MOV CX,WORD_PTR CS:DR_PT+2 ; CMOS GOOD AND UNESTABLISHED; USE CMOS
585 01D9 3C 80       CMP AL,RATE_250    ; ISOLATE STATE
586 01DB 75 11       JNE DISK_P6        ; GET DRIVE PARAMETERS FOR 1.2 M DRIVE
587
588 01DD 2E: 8B 0E 0214 R MOV CX,WORD_PTR CS:DR_PT ; GET DRIVE PARAMETERS 360K DRIVE
589 01E2 F6 85 0090 R 01 TEST #DSK_STATE[D1],TRK_CAPA ; 80 TRACK ?
590 01E7 74 05       JZ DISK_P6         ; MUST BE 360
591
592 01E9 2E: 8B 0E 0218 R MOV CX,WORD_PTR CS:DR_PT+4 ; GET DRIVE PARAMETERS
593 01EE
594 01EE 89 4E 00     DISK_P6:
595 01F1 8D 06 0000 E MOV [BP],CX
596 01F5 89 46 06    LEA AX,DISK_BASE   ; SAVE POINTER IN STACK FOR RETURN
597 01F8 8C C8       MOV [BP+6],AX      ; ADDRESS OF DISK_BASE
598 01FA
599 01FA 8E C0       MOV AX,CS          ; SAVE IN STACK
600 01FC EB 02EE R   DISK_P7:
601 01FF 33 C0       MOV AX,CS          ; SEGMENT DISK_BASE (SAME AS THIS ONE)
602 0201 F8
603 0202 C3
604
605
606
607 0203
608 0203 C6 46 04 00 ;----- NO DRIVE PRESENT HANDLER
609 0207
610 0207 33 C0
611 0209 89 46 00
612 020C 88 66 05
613 020F 89 46 06
614 0212 EB E6
615 0214
616
617
618
619 0214 09 27
620 0216 0F 4F
621 0218 09 4F
622 = 021A
623
624
625
626
627
628
629
630 021A
631 021A EB 02C8 R   ;----- DRIVE PARAMETER TABLE
632 021D 8A 85 0090 R DR_PT DB 09H,027H ; MAX. TRACKS, SECTORS/TRACK 360K
633 0221 0A C0       DB 0FH,04FH ; " " " " 1.2M
634 0223 74 13       DB 09H,04FH ; " " " " 720K
635 0225 B4 01
636 0227 A8 01
637 0229 74 02
638 022B B4 02
639
640 022D
641 022D 50
642 022E EB 02EE R DR_PTE EQU $
643 0231 58
644 0232 F8
645 0233 8B DE
646 0235 BA C3
647 0237 C4
648 0238
649 0238 32 E4
650 023A EB F1
651 023C
652
653
DISK_TYPE ENDP
; DISK_CHANGE : THIS ROUTINE RETURNS THE STATE OF THE DISK CHANGE LINE.

```

```

654 ;
655 ; ON ENTRY: DI : DRIVE #
656 ;
657 ; ON EXIT: AH : 0DSKETTE_STATUS
658 ; 00 - DISK CHANGE LINE INACTIVE, CY = 0
659 ; 06 - DISK CHANGE LINE ACTIVE, CY = 1
660 ;
-----
661 023C PROC NEAR
662 023C EB 02C8 R CALL XLAT_NEW ; TRANSLATE STATE TO PRESENT ARCH.
663 023F 8A 85 0090 R MOV AL,0DSK_STATE[DI] ; GET MEDIA STATE INFORMATION
664 0243 0A C0 OR AL,AL ; DRIVE PRESENT ?
665 0245 74 19 JZ DC_NON ; JUMP IF NO DRIVE
666 0247 A8 01 AL,TRK_CAPA ; 80 TRACK DRIVE ?
667 0249 74 05 JZ SETIT ; IF 50, CHECK CHANGE LINE
668 ;
669 024B EB 08E3 R CALL READ_DSKCHNG ; READ CHECK STATE OF DISK CHANGE LINE
670 024E 74 05 JZ FINIS ; CHANGE LINE NOT ACTIVE
671 ;
672 0250 C6 06 0041 R 06 SETIT: MOV 0DSKETTE_STATUS,MEDIA_CHANGE ; INDICATE MEDIA REMOVED
673 ;
674 0255 EB 02EE R FINIS: CALL XLAT_OLD ; TRANSLATE STATE TO COMPATIBLE MODE
675 0258 EB 0620 R CALL SETUP_END ; VARIOUS CLEANUPS
676 025B 8B DE MOV BX,SI ; GET SAVED AL TO BL
677 025D 8A C3 MOV AL,BL ; PUT BACK FOR RETURN
678 025F C3 RET
679 ;
680 0260 DC_NON:
681 0260 80 0E 0041 R 80 JMP ORP 0DSKETTE_STATUS,TIME_OUT ; SET TIMEOUT, NO DRIVE
682 0265 EB EE
683 0267 DISK_CHANGE ENDP
-----
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: 0DSKETTE_STATUS REFLECTS STATUS
692 ; AH : 0DSKETTE_STATUS
693 ; CY : 1 IF ERROR
694 ;
-----
695 0267 PROC NEAR
696 0267 EB 02C8 R CALL XLAT_NEW ; TRANSLATE STATE TO PRESENT ARCH.
697 026A 56 PUSH SI ; SAVE DASD TYPE
698 026B 8B C6 MOV AX,SI ; AH = ?, AL = DASD TYPE
699 026D 32 E4 XOR AH,AH ; AH = 0, AL = DASD TYPE
700 026F 8B F0 MOV SI,AX ; SI = DASD TYPE
701 0271 80 A5 0090 R 0F AND 0DSK_STATE[DI],NOT MED_DET+DBL_STEP+RATE_MSK ; CLEAR STATE
702 0276 4E DEC SI ; CHECK FOR 320/360K MEDIA & DRIVE
703 0277 75 07 JNZ NOT_320 ; BYPASS IF NOT
704 0279 80 8D 0090 R 90 OR 0DSK_STATE[DI],MED_DET+RATE_250 ; SET TO 320/360
705 027E EB 37 JMP SHORT S0
706 ;
707 0280 NOT_320:
708 0280 EA 0416 R CALL MED_CHANGE ; CHECK FOR TIME_OUT
709 0283 80 3E 0041 R 80 CMP 0DSKETTE_STATUS,TIME_OUT
710 0288 74 2D JZ S0 ; IF TIME OUT TELL CALLER
711 ;
712 028A 4E DEC SI ; CHECK FOR 320/360K IN 1.2M DRIVE
713 028B 75 07 JNZ NOT_320_12 ; BYPASS IF NOT
714 028D 80 8D 0090 R 70 OR 0DSK_STATE[DI],MED_DET+DBL_STEP+RATE_300 ; SET STATE
715 0292 EB 23 JMP SHORT S0
716 ;
717 0294 NOT_320_12:
718 0294 4E DEC SI ; CHECK FOR 1.2M MEDIA IN 1.2M DRIVE
719 0295 75 07 JNZ NOT_12 ; BYPASS IF NOT
720 0297 80 8D 0090 R 10 OR 0DSK_STATE[DI],MED_DET+RATE_500 ; SET STATE VARIABLE
721 029C EB 19 JMP SHORT S0 ; RETURN TO CALLER
722 ;
723 029E NOT_12:
724 029E 4E DEC SI ; CHECK FOR SET DASD TYPE 04
725 029F 75 20 JNZ FS_ERR ; BAD COMMAND EXIT IF NOT VALID TYPE
726 ;
727 02A1 F6 85 0090 R 04 TEST 0DSK_STATE[DI],DRV_DET ; DRIVE DETERMINED ?
728 02A6 74 09 JZ ASSUME ; IF STILL NOT DETERMINED ASSUME
729 02A8 B0 50 MOV AL,MED_DET+RATE_300
730 02AA F6 85 0090 R 02 TEST 0DSK_STATE[DI],FMT_CAPA ; MULTIPLE FORMAT CAPABILITY ?
731 02AF 75 02 JNZ OR_IT_IN ; OR IF 1.2 M THEN DATA RATE 300
732 ;
733 02B1 ASSUME:
734 02B1 B0 90 MOV AL,MED_DET+RATE_250 ; SET UP
735 ;
736 02B3 OR_IT_IN:
737 02B3 08 85 0090 R OR 0DSK_STATE[DI],AL ; OR IN THE CORRECT STATE
738 ;
739 02B7 S0:
740 02B7 EB 02EE R CALL XLAT_OLD ; TRANSLATE STATE TO COMPATIBLE MODE
741 02BA EB 0620 R CALL SETUP_END ; VARIOUS CLEANUPS
742 02BD 5B MOV BX,SI ; GET SAVED AL TO BL
743 02BE 8A C3 MOV AL,BL ; PUT BACK FOR RETURN
744 02C0 C3 RET
745 ;
746 02C1 FS_ERR:
747 02C1 C6 06 0041 R 01 MOV 0DSKETTE_STATUS,BAD_CMD ; UNKNOWN STATE,BAD COMMAND
748 02C6 EB EF JMP SHORT S0
749 ;
750 02C8 FORMAT_SET ENDP
-----
751 ;-----
752 ; XLAT_NEW: TRANSLATES DISKETTE STATE LOCATIONS FROM COMPATIBLE MODE TO
753 ; NEW ARCHITECTURE.
754 ;
755 ; ON ENTRY: DI : DRIVE
756 ;
-----
757 02C8 PROC NEAR
758 02C8 83 FF 01 CMP DI,1 ; VALID DRIVE ?
759 02CB 77 1C JNA XN_OUT ; IF INVALID BACK
760 02CD 80 8D 0090 R 00 CMP 0DSK_STATE[DI],0 ; NO DRIVE ?
761 02D2 74 16 JZ DO_DET ; IF NO DRIVE ATTEMPT DETERMINE
762 02D4 8B CF MOV CX,DI ; CX = DRIVE NUMBER
763 02D6 C0 E1 02 SHL CL,2 ; CL = SHIFT COUNT, A=0, B=4
764 02D9 A0 008F R MOV AL,RHF_CNTRL ; DRIVE INFORMATION
765 02DC D2 C8 ROR AL,CL ; TO LOW NIBBLE
766 02DE 24 07 AND AL,DRV_DET+FMT_CAPA+TRK_CAPA ; KEEP DRIVE BITS
767 02E0 80 A5 0090 R F8 AND 0DSK_STATE[DI],NOT DRV_DET+FMT_CAPA+TRK_CAPA

```

```

768 02E5 08 85 0090 R      OR      #DSK_STATE[D1],AL      ; UPDATE DRIVE STATE
769 02E9                    XN_OUT:  RET
770 02E9 C3                RET
771
772 02EA                    DO_DET:  CALL      DRIVE_DET      ; TRY TO DETERMINE
773 02EA E8 08ED R        CALL
774 02ED C3                RET
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:  D1 : DRIVE
782 -----
783 02EE                    XLAT_OLD  PROC   NEAR
784 02EE 83 FF 01          CMP     D1,1          ; VALID DRIVE ?
785 02F1 77 68            JNA    XO_OUT        ; IF INVALID BACK
786 02F3 80 8D 0090 R 00  CMP     #DSK_STATE[D1],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 BB CF            MOV     CX,D1        ; CX = DRIVE NUMBER
792 02FC C0 E1 02        SHL    CL,2         ; CL = SHIFT COUNT, A=0, B=4
793 02FF 84 02          MOV     AH,FMT_CAPA ; LOAD MULTIPLE-DATA RATE BIT MASK
794 0301 D2 CC          ROR    AH,CL        ; ROTATE BY MASK
795 0303 84 26 008F R   TEST   #HF_CNTRL,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_CNTRL FOR THIS DRIVE
799
800 0309 84 07          MOV     AH,DRV_DET+FMNT_CAPA+TRK_CAPA ; MASK TO KEEP
801 030B D2 CC          ROR    AH,CL        ; FIX MASK TO KEEP
802 030D F6 D4          NOT    AH           ; TRANSLATE MASK
803 030F 20 26 008F R   AND    #HF_CNTRL,AH ; KEEP BITS FROM OTHER DRIVE INTACT
804
805 ;----- ACCESS CURRENT DRIVE BITS AND STORE IN #HF_CNTRL
806
807 0313 8A 85 0090 R   MOV     AL,#DSK_STATE[D1] ; ACCESS STATE
808 0317 24 07          AND    AL,DRV_DET+FMNT_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_CNTRL,AL ; UPDATE SAVED DRIVE STATE
811
812 ;----- TRANSLATE TO COMPATIBILITY MODE
813
814 031F                    SAVE_SET:
815 031F 8A A5 0090 R   MOV     AH,#DSK_STATE[D1] ; ACCESS STATE
816 0323 8A FC          MOV     BH,AH       ; TO BH FOR LATER
817 0325 80 E4 C0        AND    AH,RATE_MSK ; KEEP ONLY RATE
818 0328 B0 02          MOV     AL,MID1U    ; AL = 1:2 IN 1:2 UNESTABLISHED
819 032A 80 FC 00       CMP     AH,RATE_500 ; RATE 500 ?
820 032D 74 1C          JZ     TST_DET     ; JUMP IF 1:2 IN 1:2
821 032F B0 01          MOV     AL,M3D1U    ; 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                    UNKN0:  MOV     AL,MED_UNK  ; NONE OF THE ABOVE
828 033B B0 07          MOV     BH,07       ; TO BH FOR LATER
829 033D EB 13          JMP    SHORT AL_SET ; PROCESS COMPLETE
830
831 033F                    CHK_250:  MOV     AL,M3D3U    ; AL = 360 IN 360 UNESTABLISHED
832 033F B0 00          MOV     BH,00       ; TO BH FOR LATER
833 0341 80 FC 80       CMP     AH,RATE_250 ; RATE 250 ?
834 0344 75 05          JNZ    UNKN0       ; IF SO FALL THRU
835 0346 F6 C7 01       TEST   BH,TRK_CAPA ; NO TRACK FALL THRU ?
836 0349 75 F0          JNZ    UNKN0       ; IF SO JUMP, FALL THRU TEST DET
837
838 034B                    TST_DET:  TEST   BH,MED_DET   ; DETERMINED ?
839 034B F6 C7 10       TEST   JZ,AL_SET   ; IF NOT THEN SET
840 034E 74 02          JZ     AL_SET       ; MAKE DETERMINED/ESTABLISHED
841 0350 04 03          ADD    AL,3
842
843 0352                    AL_SET:  AND    #DSK_STATE[D1],NOT DRV_DET+FMNT_CAPA+TRK_CAPA ; CLEAR DRIVE
844 0352 B0 A5 0090 R F8  OR     #DSK_STATE[D1],AL ; REPLACE WITH COMPATIBLE MODE
845 0357 08 85 0090 R   OR
846 035B                    XO_OUT:  RET
847 035B C3
848 035C                    XLAT_OLD  ENDP
849 -----
850 ; RD_WR_VF : COMMON READ, WRITE AND VERIFY; MAIN LOOP FOR STATE RETRIES.
851 ;
852 ; ON ENTRY:  AH : READ/WRITE/VERIFY DMA PARAMETER
853 ;           AL : READ/WRITE/VERIFY NEC PARAMETER
854 ;
855 ; ON EXIT:  #DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
856 -----
857 035C                    RD_WR_VF  PROC   NEAR
858 035C 50                PUSH   AX           ; SAVE DMA, NEC PARAMETERS
859 035D E8 02C8 R      CALL  XLAT_NEW     ; TRANSLATE STATE TO PRESENT ARCH.
860 0360 50 039B R      CALL  SETUP_STATE  ; STATE INITIALIZATIONS
861 0363 58                POP    AX           ; RESTORE DMA,NEC PARAMETERS
862
863 0364                    DO_AGAIN:  PUSH   AX           ; SAVE READ/WRITE/VERIFY PARAMETER
864 0364 50                CALL  MED_CHANGE   ; MEDIA CHANGE AND RESET IF CHANGED
865 0365 E8 0416 R      CALL  SEND_READ   ; RESTORE READ/WRITE/VERIFY
866 0368 58                POP    AX           ; MEDIA CHANGE ERROR OR TIME-OUT
867 0369 76 72 21       JC     RWV_END     ; SEND READ/WRITE/VERIFY PARAMETER
868 036B 50                PUSH   AX           ; SEND DATA RATE TO NEC
869 036C E8 0451 R      CALL  SEND_RATE   ; CHECK FOR DOUBLE STEP
870 036F E8 063A R      CALL  SETUP_DBL   ; ERROR FROM READ ID, POSSIBLE RETRY
871 0372 72 12          JC     CHK_RET     ; RESTORE NEC,DMA COMMAND
872 0374 58                POP    AX           ; SAVE NEC COMMAND
873 0375 50                PUSH   AX           ; SET UP THE DMA
874 0376 E8 0471 R      CALL  DMA_SETUP   ; RESTORE NEC COMMAND
875 0379 58                POP    AX           ; CHECK FOR DMA BOUNDARY ERROR
876 037A 72 16          JC     RWV_BAC    ; SAVE NEC COMMAND
877 037C 50                PUSH   AX           ; INITIALIZE NEC
878 037D E8 04C7 R      CALL  NEC_INIT    ; OP CODE COMMON TO READ/WRITE/VERIFY
879 0380 E8 04EC R      CALL  RWV_COM     ; TERMINATE, GET STATUS, ETC.
880 0383 E8 0530 R      CALL  NEC_TERM
881

```

```

882 0366                CHK_RET:
883 0366 E8 05B1 R      CALL RETRY                ; CHECK FOR, SETUP RETRY
884 0369 58            POP AX                    ; RESTORE READ/WRITE/VERIFY PARAMETER
885 036A 72 D8         JC DO_AGAIN              ; CY = 1 MEANS RETRY
886
887 036C                RWV_END:
888 036C E8 0582 R      CALL DSTATE              ; ESTABLISH STATE IF SUCCESSFUL
889 036F E8 05F3 R      CALL NUM_TRANS          ; AL = NUMBER TRANSFERRED
890
891 0392                RWV_BAC:
892 0392 50            PUSH AX                    ; BAD DMA ERROR ENTRY
893 0393 E8 02EE R      CALL XLAT_OLD           ; SAVE NUMBER TRANSFERRED
894 0396 58            POP AX                    ; TRANSLATE STATE TO COMPATIBLE MODE
895 0397 E8 0620 R      CALL SETUP_END         ; RESTORE NUMBER TRANSFERRED
896 039A C3            RET                      ; VARIOUS CLEANUPS
897 039B
898
899                RD_WR_VF  ENDP
900                ;-----
901                ; SETUP_STATE: INITIALIZES START AND END RATES.
902                ;-----
903                SETUP_STATE PROC NEAR
904                TEST  #DSK_STATE[D1],MED_DET ; MEDIA DETERMINED ?
905                JNZ  JIC                    ; NO STATES IF DETERMINED
906                MOV  AX,RATE_300*H+RATE_250 ; AH = START RATE, AL = END RATE
907                TEST  #DSK_STATE[D1],DRV_DET ; DRIVE ?
908                JZ   AX_SET                ; DO NOT KNOW DRIVE
909                MOV  AL,RATE_500          ; SET UP FOR 1.2 M END RATE
910                TEST  #DSK_STATE[D1],FMT_CAPA ; 1.2 M ?
911                JNZ  AX_SET                ; JUMP WITH FIXED END RATE
912                MOV  AX,RATE_250*X       ; START & END RATE = 250 FOR 360 DRIVE
913                OR   AND #DSK_STATE[D1],NOT_RATE_MSK+DBL_STEP ; TURN OFF THE RATE
914                AND  OR #LAstrate,NOT STRT_MSK ; ERASE LAST TO TRY RATE BITS
915                AND  OR AL,4 ; TO OPERATION LAST RATE LOCATION
916                OR   OR #LAstrate,AL ; LAST RATE
917                JIC:
918                RET
919                SETUP_STATE ENDP
920                ;-----
921                ; FMT_INIT: ESTABLISH STATE IF UNESTABLISHED AT FORMAT TIME.
922                ;-----
923                FMT_INIT PROC NEAR
924                TEST  #DSK_STATE[D1],MED_DET ; IS MEDIA ESTABLISHED
925                JNZ  FI_OUT                ; IF SO RETURN
926                CALL CMDS_TYPE            ; RETURN DRIVE TYPE IN AL
927                JC   CL_DRV              ; ERROR IN CMOS ASSUME NO DRIVE
928                DEC  AL                   ; MAKE ZERO ORIGIN
929                AL  DEC                   ; NO DRIVE IF AL 0
930                JS   CL_DRV              ; AH = CURRENT STATE
931                MOV  AH,#DSK_STATE[D1] ; AH = CURRENT STATE
932                AND  AND AH,NOT MED_DET+DBL_STEP+RATE_MSK ; CLEAR
933                OR   OR AL,AL            ; CHECK FOR 360
934                JNZ  N_360              ; IF 360 WILL BE 0
935                OR   OR AH,MED_DET+RATE_250 ; ESTABLISH MEDIA
936                JMP  SHORT SKP_STATE     ; SKIP OTHER STATE PROCESSING
937
938                N_360:
939                DEC  AL                   ; 1.2 M DRIVE
940                JNZ  N_12                ; JUMP IF NOT
941                OR   OR AH,MED_DET+RATE_500 ; DEFAULT TO 1.2M FORMAT
942                JMP  SHORT SKP_STATE     ; SKIP OTHER STATE PROCESSING
943
944                N_12:
945                DEC  AL                   ; CHECK FOR TYPE 3
946                JNZ  CL_DRV              ; NO DRIVE, CMOS BAD
947                TEST AH,DRV_DET          ; IS DRIVE DETERMINED
948                JZ   ISNT_12            ; TREAT AS NON 1.2 DRIVE
949                TEST AH,FMT_CAPA        ; IS 1.2M
950                JZ   ISNT_12            ; JUMP IF NOT
951                OR   OR AH,MED_DET+RATE_300 ; RATE 300
952                JMP  SHORT SKP_STATE     ; CONTINUE
953
954                ISNT_12:
955                OR   OR AH,MED_DET+RATE_250 ; MUST BE RATE 250
956
957                SKP_STATE:
958                MOV  #DSK_STATE[D1],AH ; STORE AWAY
959
960                FI_OUT:
961                RET
962
963                CL_DRV:
964                XOR  AH,AH                ; CLEAR STATE
965                JMP  SHORT SKP_STATE     ; SAVE IT
966
967                FMT_INIT ENDP
968                ;-----
969                ; MED_CHANGE: CHECKS FOR MEDIA CHANGE, RESETS MEDIA CHANGE, CHECKS MEDIA
970                ; CHANGE AGAIN.
971                ;
972                ; ON EXIT: CY = 1 MEANS MEDIA CHANGE OR TIMEDOUT
973                ;
974                ;-----
975                MED_CHANGE PROC NEAR
976                CALL READ_DSKCHNG        ; READ DISK CHANGE LINE STATE
977                JZ   MED_C9              ; BYPASS HANDLING DISK CHANGE LINE
978                AND  #DSK_STATE[D1],NOT MED_DET ; CLEAR STATE FOR THIS DRIVE
979
980                ; THIS SEQUENCE ENSURES WHENEVER A DISKETTE IS CHANGED THAT
981                ; ON THE NEXT OPERATION THE REQUIRED MOTOR START UP TIME WILL
982                ; BE WAITED. (DRIVE MOTOR MAY GO OFF UPON DOOR OPENING).
983                MOV  CX,D1                ; CL = DRIVE #
984                MOV  AL,1                ; MOTOR ON BIT MASK
985                SHL  AL,CL                ; TO APPROPRIATE POSITION
986                NOT  AL                   ; KEEP ALL BUT MOTOR ON
987                CLI ; NO INTERRUPTS
988                AND #MOTOR_STATUS,AL ; TURN MOTOR OFF INDICATOR
989                STI ; INTERRUPTS ENABLED
990                CALL MOTOR_ON            ; TURN MOTOR ON
991
992                ;----- THIS SEQUENCE OF SEKS IS USED TO RESET DISKETTE CHANGE SIGNAL
993
994                CALL DISK_RESET          ; RESET NEC
995                MOV  CH,0TH              ; MOVE TO CYLINDER 1
    
```

SECTION 5

```

996 0436 E8 07DE R      CALL    SEEK          ; ISSUE SEEK
997 0439 32 ED         XOR     CH,CH          ; MOVE TO CYLINDER 0
998 043B E8 07DE 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 74 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
1006 044D F9         STC     RET           ; MEDIA CHANGED, SET CY
1007 044E C3         RET
1008 044F
1009 044F F8         CLC     RET           ; NO MEDIA CHANGED, CLEAR CY
1010 0450 C3         RET
1011 0451
1012
1013
1014
1015 0451
1016 0451 8A 26 008B R   AND     @LAstrate,NOT SEND_MSK ; ELSE CLEAR LAST RATE ATTEMPTED
1017 0455 8A 85 0090 R   OR      @LAstrate,AL      ; SAVE NEW RATE FOR NEXT CHECK
1018 0459 25 C0C0       ROL     AL,2            ; MOVE TO BIT OUTPUT POSITIONS
1019 045C 3A C4         MOV     DX,03F7H       ; OUTPUT NEW DATA RATE
1020 045E 74 10         JE      C_S_OUT        ; IF SAME, NO NEW TRANSFER RATE
1021
1022 0460 80 26 008B R 3F AND     @LAstrate,NOT SEND_MSK ; ELSE CLEAR LAST RATE ATTEMPTED
1023 0465 08 06 008B R   OR      @LAstrate,AL      ; SAVE NEW RATE FOR NEXT CHECK
1024 0469 C0 C0 02       ROL     AL,2            ; MOVE TO BIT OUTPUT POSITIONS
1025 046C BA 03F7H     MOV     DX,03F7H       ; OUTPUT NEW DATA RATE
1026 046F EE         OUT    DX,AL
1027
1028 0470
1029 0470 C3         RET
1030 0471
1031
1032
1033
1034
1035
1036
1037
1038 0471
1039 0471 FA         CL1    DMA+12,AL       ; DISABLE INTERRUPTS DURING DMA SET-UP
1040 0472 E6 0C         OUT    $+2            ; SET THE FIRST/LAST F/F
1041 0474 EB 00         JMP    DMA+11,AL      ; WAIT FOR I/O
1042 0476 E6 0B         OUT    DMA+11,AL     ; OUTPUT THE MODE BYTE
1043 0478 8C C0       MOV    AX,ES         ; GET THE ES VALUE
1044 047A C1 C0 04     ROL    AX,4          ; ROTATE LEFT
1045 047D 8A E8         MOV    CH,AL         ; GET HIGHEST NIBBLE OF ES TO CH
1046 047F 24 F0       AND    AL,11110000B ; 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           ; CARRY MEANS HIGH 4 BITS MUST BE INC
1049 0486 FE C5       INC    CH
1050 0488
1051 0488 50         PUSH   AX            ; SAVE START ADDRESS
1052 0489 E6 04         OUT    DMA+4,AL     ; OUTPUT LOW ADDRESS
1053 048B EB 00         JMP    DMA+10,AL    ; WAIT FOR I/O
1054 048D 8A C4       MOV    AL,AH         ; OUTPUT HIGH ADDRESS
1055 048F E6 04         OUT    AL,CH        ; GET HIGH 4 BITS
1056 0491 8A C5       MOV    AL,CH        ; I/O WAIT STATE
1057 0493 EB 00         JMP    DMA+5,AL    ; I/O WAIT STATE
1058 0495 24 0F       AND    AL,00001111B ; I/O WAIT STATE
1059 0497 E6 81         OUT    081H,AL     ; OUTPUT HIGH 4 BITS TO PAGE REGISTER
1060
1061
1062
1063 0499 8B C6       MOV    AX,S1        ; AL = # OF SECTORS
1064 049B 86 C4       XCHG  AX,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 B2 03       MOV    DL,3        ; GET BYTES/SECTOR PARAMETER
1069 04A4 E8 06CC R   CALL  GET_PARM     ;
1070 04A7 8A CC       MOV    CL,AH        ; SHIFT COUNT (0=128, 1=256 ETC)
1071 04A9 58 E8       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 04AD 50         PUSH  AX           ; SAVE COUNT VALUE
1075 04AE E4 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         STI
1080 04B7 59         POP   CX           ; RECOVER COUNT VALUE
1081 04B8 58         POP   AX           ; RECOVER ADDRESS VALUE
1082 04B9 03 C1       ADD   AX,CX        ; ADD, TEST FOR 64K OVERFLOW
1083 04BB B0 02       MOV   AL,2         ; MODE FOR 8237
1084 04BD E6 0A       OUT   DMA+10,AL   ; INITIALIZE THE DISKETTE CHANNEL
1085 04BF 73 05       JNC   NO_BAD       ; CHECK FOR ERROR
1086 04C1 C6 06 0041 R 09 MOV     @DSKETTE_STATUS,DMA_BOUNDARY ; SET ERROR
1087 04C6
1088 04C6 C3         RET
1089 04C7
1090
1091
1092
1093
1094
1095
1096
1097
1098 04C7
1099 04C7 50         PUSH  AX           ; SAVE NEC COMMAND
1100 04C8 E8 06E1 R     CALL  MOTOR_ON     ; TURN MOTOR ON FOR SPECIFIC DRIVE
1101
1102
1103
1104 04CB 8A 6E 01     MOV    CH,[BP+1]    ; CH = TRACK #
1105 04CE E8 07DE R     CALL  SEEK         ; MOVE TO CORRECT TRACK
1106 04D1 58         POP   AX           ; RECOVER COMMAND
1107 04D2 72 17       JC    ER_1         ; ERROR ON SEEK
1108 04D4 BB 04EB R     MOV    BX,OFFSET ER_1 ; LOAD ERROR ADDRESS
1109 04D7 53         PUSH  BX           ; PUSH NEC_OUT ERROR RETURN
    
```

```

1110
1111 ;----- SEND OUT THE PARAMETERS TO THE CONTROLLER
1112
1113 CALL NEC_OUTPUT ; OUTPUT THE OPERATION COMMAND
1114 MOV AX,S1 ; AH = HEAD #
1115 MOV BX,D1 ; BL = DRIVE #
1116 MOV AH,2 ; MOVE IT TO BIT 2
1117 AND AH,00001000B ; ISOLATE THAT BIT
1118 OR AH,BL ; OR IN THE DRIVE NUMBER
1119 CALL NEC_OUTPUT ; FALL THRU CY SET IF ERROR
1120 POP BX ; THROW AWAY ERROR RETURN
1121 ER_1:
1122 RET
1123 NEC_INIT ENDP
1124
1125 ;-----
1126 ; RWV_COM: THIS ROUTINE SENDS PARAMETERS TO THE NEC SPECIFIC TO THE
1127 ; READ/WRITE/VERIFY OPERATIONS.
1128 ;
1129 ; ON EXIT: #DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
1130
1130 RWV_COM PROC NEAR
1131 MOV AX,OFFSET ER_2 ; LOAD ERROR ADDRESS
1132 PUSH AX ; PUSH NEC OUT ERROR RETURN
1133 MOV AH,[BP+1] ; OUTPUT TRACK #
1134 CALL NEC_OUTPUT
1135 MOV AX,S1 ; OUTPUT HEAD #
1136 CALL NEC_OUTPUT ; OUTPUT SECTOR #
1137 MOV AH,[BP]
1138 CALL NEC_OUTPUT ; BYTES/SECTOR PARAMETER FROM BLOCK
1139 MOV DL,3 ; TO THE NEC
1140 CALL GET_PARM ; OUTPUT TO CONTROLLER
1141 CALL NEC_OUTPUT ; EOT PARAMETER FROM BLOCK
1142 MOV DL,4 ; GET PARM
1143 CALL GET_PARM ; OUTPUT TO CONTROLLER
1144 CALL NEC_OUTPUT ; GET DRIVE STATE VALUE
1145 MOV AL,#DSK_STATE[D1] ; 1,2,1,2 DRIVE GAP LENGTH
1146 MOV AH,01BH ; STRIP OFF HIGH BITS
1147 AND AL,RATE_MSK ; IF SO JUMP
1148 JZ R15
1149 MOV AH,023H ; 320,360/1,2 DRIVE GAP LENGTH
1150 DEC AL ; CHECK FOR 320 MEDIA IN 1.2 DRIVE
1151 R15 ; IF SO JUMP
1152
1153 MOV AH,02AH ; 360/360 DRIVE GAP LENGTH
1154 R15:
1155 CALL NEC_OUTPUT
1156 MOV DL,6 ; DTL PARAMETER FROM BLOCK
1157 CALL GET_PARM ; TO THE NEC
1158 CALL NEC_OUTPUT ; OUTPUT TO CONTROLLER
1159 POP AX ; THROW AWAY ERROR EXIT
1160 ER_2:
1161 RET
1162 RWV_COM ENDP
1163
1164 ;-----
1165 ; NEC_TERM: THIS ROUTINE WAITS FOR THE OPERATION THEN ACCEPTS THE STATUS
1166 ; FROM THE NEC FOR THE READ/WRITE/VERIFY/FORMAT OPERATION.
1167 ;
1168 ; ON EXIT: #DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
1169
1169 NEC_TERM PROC NEAR
1170
1171 ;----- LET THE OPERATION HAPPEN
1172
1173 PUSH SI ; SAVE HEAD #, # OF SECTORS
1174 CALL WAIT_INT ; WAIT FOR THE INTERRUPT
1175 PUSHF
1176 CALL RESULTS ; GET THE NEC STATUS
1177 SET_END_POP
1178 POPF
1179 JC SET_END ; LOOK FOR ERROR
1180
1181 ;----- CHECK THE RESULTS RETURNED BY THE CONTROLLER
1182
1183 CLD ; SET THE CORRECT DIRECTION
1184 MOV SI,OFFSET #NEC_STATUS ; POINT TO STATUS FIELD
1185 LODS #NEC_STATUS ; GET ST0
1186 AND AL,10000000B ; TEST FOR NORMAL TERMINATION
1187 JZ SET_END ; TEST FOR ABNORMAL TERMINATION
1188 CMP AL,01000000B ; NOT ABNORMAL, BAD NEC
1189 JNZ J18
1190
1191 ;----- ABNORMAL TERMINATION, FIND OUT WHY
1192
1193 LODS #NEC_STATUS ; GET ST1
1194 SAL AL,1 ; TEST FOR EOT FOUND
1195 MOV AH,RECORD_NOT_FND
1196 JC J19
1197 SAL AL,2
1198 MOV AH,BAD_CRC
1199 JC J19
1200 SAL AL,1 ; TEST FOR DMA OVERRUN
1201 MOV AH,BAD_DMA
1202 JC J19
1203 SAL AL,2 ; TEST FOR RECORD NOT FOUND
1204 MOV AH,RECORD_NOT_FND
1205 JC J19
1206 SAL AL,1
1207 MOV AH,WRITE_PROTECT ; TEST FOR WRITE_PROTECT
1208 JC J19
1209 SAL AL,1 ; TEST MISSING ADDRESS MARK
1210 MOV AH,BAD_ADDR_MARK
1211 JC J19
1212
1213 ;----- NEC MUST HAVE FAILED
1214 J18:
1215 MOV AH,BAD_NEG
1216 J19:
1217 OR #DSKETTE_STATUS,AH
1218 SET_END:
1219 CMP #DSKETTE_STATUS,1 ; SET ERROR CONDITION
1220 CMC
1221 POP SI ; RESTORE HEAD #, # OF SECTORS
1222 RET
1223

```

```

1224 057F          SET_END_POP:
1225 057F 9D          JMPF          SHORT SET_END
1226 0580 EB F5      NEC_TERM     JMP          ENDP
1227 0582
1228
1229 ;-----
1229 ; DSTATE:     ESTABLISH STATE UPON SUCCESSFUL OPERATION.
1230 ;-----
1231 0582
1232 0582 80 3E 0041 R 00 DSTATE PROC NEAR
1233 0587 75 27          CMP          #DSKETTE_STATUS,0 ; CHECK FOR ERROR
1234 0580 8D 0900 R 10 OR           #DSK_STATE[D1],MED_DET ; IF ERROR JUMP
1235 058E F6 85 0090 R 04 TEST        #DSK_STATE[D1],DRV_DET ; NO ERROR, MARK MEDIA AS DETERMINED
1236 0593 75 1B          JNZ         ; DRIVE DETERMINED ?
1237 0595 8A 85 0090 R 04 MOV         AL,#DSK_STATE[D1] ; IF DETERMINED NO TRY TO DETERMINE
1238 0599 24 C0          AND         AL,RATE_MSK ; LOAD STATE
1239 059B 3C 80          CMP        AL,RATE_250 ; KEEP ONLY RATE
1240 059D 75 0C          JNE        M 12 ; CHECK FOR 1.2M
1241 059F 80 A5 0090 R FD JNC        #DSK_STATE[D1],NOT_FMT_CAPA ; MUST BE 1.2
1242 05A4 80 8D 0090 R 04 OR          #DSK_STATE[D1],DRV_DET ; CAPA ; TURN OFF FORMAT CAPABILITY
1243 05A9 EB 05          JMP        SHORT SETBAC ; MARK DRIVE DETERMINED
1244
1245 05AB
1246 05AB 80 8D 0090 R 06 M_12:      OR          #DSK_STATE[D1],DRV_DET+_FMT_CAPA ; TURN ON DETERMINED & FMT CAPA
1247
1248 05B0
1249 05B0 C3          SETBAC:    RET
1250 05B1
1251 ;-----
1251 ; DSTATE ENDP
1252 ;-----
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          RETRY     PROC NEAR
1258 05B1 80 3E 0041 R 00 CMP          #DSKETTE_STATUS,0 ; GET STATUS OF OPERATION
1259 05B6 74 39          JZ          NO_RETRY ; SUCCESSFUL OPERATION
1260 05B8 80 8D 0041 R 80 CMP          #DSKETTE_STATUS,TIME_OUT ; IF TIME OUT NO RETRY
1261 05BD 74 32          JZ          NO_RETRY ;
1262 05BF 8A A5 0090 R 04 MOV         AH,#DSK_STATE[D1] ; GET MEDIA STATE OF DRIVE
1263 05C3 F6 C4 10      TEST        AH,MED_DET ; ESTABLISHED/DETERMINED ?
1264 05C7 75 29          JNZ         ; IF ESTABLISHED STATE THEN TRUE ERROR
1265 05C8 80 E4 C0      AND         AH,RATE_MSK ; ISOLATE RATE
1266 05CB 8A 2E 00BB R 04 MOV         CH,#LAstrate ; GET START OPERATION STATE
1267 05CF C0 C5 04      ROL        CH,4 ; TO CORRESPONDING BITS
1268 05D2 80 E5 C0      AND         CH,LAstrate ; ISOLATE RATE BITS
1269 05D5 3A EC          CMP        CH,AH ; ALL RATES TRIED
1270 05D7 74 18          JE          NO_RETRY ; IF YES, THEN TRUE ERROR
1271
1272 ;-----
1272 ; SETUP STATE INDICATOR FOR RETRY ATTEMPT TO NEXT RATE
1273 ; 00000000B (500) -> 10000000B (250)
1274 ; 10000000B (250) -> 01000000B (300)
1275 ; 01000000B (300) -> 00000000B (500)
1276
1277 05D9 80 FC 01          CMP        AH,RATE_500+1 ; SET CY FOR RATE 500
1278 05DC DD DC          RCR        AH,1 ; TO NEXT STATE
1279 05DE 80 E4 C0      AND         AH,RATE_MSK ; KEEP ONLY RATE BITS
1280 05E1 80 A5 0090 R 1F AND         #DSK_STATE[D1],NOT RATE_MSK+DBL_STEP ; RATE, DBL STEP OFF
1281 05E6 08 A5 0090 R 04 OR          #DSK_STATE[D1],AH ; TURN ON NEW RATE
1282 05EA C6 06 0041 R 00 MOV         #DSKETTE_STATUS,0 ; RESET STATUS FOR RETRY
1283 05EF F9 C3          STC        RETRY ; SET CARRY FOR RETRY
1284 05F0 C3          RETRY     RETRY ; RETRY RETURN
1285
1286 05F1
1287 05F1 F8          NO_RETRY:  CLC        ; CLEAR CARRY NO RETRY
1288 05F2 C3          RETRY     RETRY ; NO RETRY RETURN
1289 05F3
1290 ;-----
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
1300 05F3          NUM_TRANS PROC NEAR
1301 05F3 32 C0          XOR        AL,AL ; CLEAR FOR ERROR
1302 05F5 80 3E 0041 R 00 CMP          #DSKETTE_STATUS,0 ; CHECK FOR ERROR
1303 05FA 75 23          JNZ        NT_OUT ; IF ERROR 0 TRANSFERRED
1304 05FC B2 04          MOV        DL,4 ; SECTORS/TRACK OFFSET TO DL
1305 05FE 05 E8 06CC R 04 CALL       GET_FARM ; AH = SECTORS/TRACK
1306 0601 8A 1E 0047 R 04 MOV        BL,#NEC_STATUS+5 ; GET ENDING SECTOR
1307 0605 8B EC          MOV        CX,SI ; CH = HEAD # STARTED
1308 0607 3A 2E 0046 R 04 CMP        CH,#NEC_STATUS+4 ; GET HEAD ENDED UP ON
1309 060B 75 0B          JNZ        DIF_HD ; IF ON SAME HEAD, THEN NO ADJUST
1310
1311 060D 8A 2E 0045 R 04 MOV        CH,#NEC_STATUS+3 ; GET TRACK ENDED UP ON
1312 0611 3A 6E 01          CMP        CH,[BP+1] ; IS IT ASKED FOR TRACK
1313 0614 74 04          JZ         SAME_TRK ; IF SAME TRACK NO INCREASE
1314
1315 0616 02 DC          ADD        BL,AH ; ADD SECTORS/TRACK
1316 0618
1317 0618 02 DC          DIF_HD:   ADD        BL,AH ; ADD SECTORS/TRACK
1318 061A
1319 061A 2A 5E 00          SUB        BL,[BP] ; SUBTRACT START FROM END
1320 061D 8A C3          MOV        AL,BL ; TO AL
1321
1322 061F          NT_OUT:   RET
1323 061F C3          RET
1324 0620
1325 ;-----
1325 ; SETUP_END: RESTORES #MOTOR COUNT TO PARAMETER PROVIDED IN TABLE AND LOADS
1326 ;            #DSKETTE_STATUS TO AH, AND SETS CY.
1327 ;
1328 ; ON EXIT:    AH, #DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
1329 ;-----
1330
1331 0620          SETUP_END PROC NEAR
1332 0620 B2 02          MOV        DL,2 ; GET THE MOTOR WAIT PARAMETER
1333 0622 50          PUSH       AX ; SAVE NUMBER TRANSFERRED
1334 0623 E8 06CC R 04 CALL       GET_FARM ;
1335 0626 B8 26 0040 R 04 MOV        #MOTOR_COUNT,AH ; STORE UPON RETURN
1336 062A 58          POP        AX ; RESTORE NUMBER TRANSFERRED
1337 062B 8A 26 0041 R 04 MOV        AH,#DSKETTE_STATUS ; GET STATUS OF OPERATION

```



```

1338 062F 0A E4 OR AH,AH ; CHECK FOR ERROR
1339 0631 78 02 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 CMC ; SUCCESS OR FAILURE
1345 0639 C3 RET
1346 063A SETUP_END ENDP
-----
1347 ;
1348 ; SETUP_DBL: CHECK DOUBLE STEP.
1349 ;
1350 ; ON ENTRY: AH = RATE; DI = DRIVE
1351 ;
1352 ; ON EXIT: CY = 1 MEANS ERROR
1353 ;
-----
1354 063A SETUP_DBL PROC NEAR
1355 063A 8A 5A 0090 R MOV AH,DSK_STATE[DI] ; ACCESS STATE
1356 063E F6 C4 10 TEST AH,MED_DET ; ESTABLISHED STATE ?
1357 0641 75 59 JNZ NO_DBL ; IF ESTABLISHED THEN DOUBLE DONE
1358
1359 ;----- CHECK FOR TRACK 0 TO SPEED UP ACKNOWLEDGE OF UNFORMATTED DISKETTE
1360
1361 0643 C6 06 003E R 00 MOV DSSEK_STATUS,0 ; SET RECALIBRATE REQUIRED ON ALL DRIVES
1362 0648 E8 06E1 R CALL MOTOR_ON ; ENSURE MOTOR STAY ON
1363 064B B5 00 MOV CH,0 ; LOAD TRACK 0
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 ;----- INITIALIZE START AND MAX TRACKS (TIMES 2 FOR BOTH HEADS)
1369
1370 0655 B9 0450 MOV CX,0450H ; START, MAX TRACKS
1371 0658 F6 B5 0090 R 01 TEST DSK_STATE[DI],TRK_CAPA ; TEST FOR 80 TRACK CAPABILITY
1372 065D 74 02 JZ CNT_OK ; IF NOT COUNT IS SETUP
1373 065F B1 A0 MOV CL,500H ; MAXIMUM TRACK 1.2 MB
1374
1375 ;
1376 ; ATTEMPT READ ID OF ALL TRACKS, ALL HEADS UNTIL SUCCESS; UPON SUCCESS,
1377 ; MUST SEE IF ASKED FOR TRACK IN SINGLE STEP MODE = TRACK ID READ; IF NOT
1378 ; THEN SET DOUBLE STEP ON.
1379 0661 CNT_OK:
1380 0661 51 PUSH CX ; SAVE TRACK COUNT
1381 0662 C6 06 0041 R 00 MOV DSSETTE_STATUS,0 ; CLEAR STATUS, EXPECT ERRORS
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 PUSH AX ; SAVE HEAD
1386 066F E8 07DE R CALL SEEK ; SEEK TO TRACK
1387 0672 58 POP AX ; RESTORE HEAD
1388 0673 0B F8 OR DI,AX ; DI = HEAD OR *ED DRIVE
1389 0675 E8 069E R CALL READ_ID ; READ ID HEAD 0
1390 0678 9C PUSHF ; SAVE RETURN FROM READ_ID
1391 0679 81 E7 00FB AND DI,11111011B ; TURN OFF HEAD 1 BIT
1392 067D 9D POPF ; RESTORE ERROR RETURN
1393 067E 59 POP CX ; RESTORE COUNT
1394 067F 73 08 JNC DD_CHK ; IF OK, ASKED = RETURNED TRACK ?
1395 0681 FE C5 INC CH ; INC FOR NEXT TRACK
1396 0683 3A E9 CMP CH,CL ; REACHED MAXIMUM YET
1397 0685 75 DA JNZ CNT_OK ; CONTINUE TILL ALL TRIED
1398
1399 ;----- FALL THRU, READ ID FAILED FOR ALL TRACKS
1400
1401 0687 SD_ERR:
1402 0687 F9 STC ; SET CARRY FOR ERROR
1403 0688 C3 RET ; SETUP_DBL ERROR EXIT
1404
1405 0689 DO_CHK:
1406 0689 8A E8 0045 R MOV CL,DSK_STATUS+3 ; LOAD RETURNED TRACK
1407 068D 88 E8 0094 R MOV DSSTK_TRK[DI],CL ; DSK_TRACK NUMBER
1408 0691 D0 ED SHR CH,1 ; HALVE TRACK
1409 0693 3A E9 CMP CH,CL ; IS IT THE SAME AS ASKED FOR TRACK
1410 0695 74 05 JZ NO_DBL ; IF SAME THEN NO DOUBLE STEP
1411 0697 80 8D 0090 R 20 OR DSK_STATE[DI],DBL_STEP ; TURN ON DOUBLE STEP REQUIRED
1412
1413 069C NO_DBL:
1414 069C F8 CLC ; CLEAR ERROR FLAG
1415 069D C3 RET
1416 069E SETUP_DBL ENDP
-----
1417 ;
1418 ; READ_ID: READ ID FUNCTION.
1419 ;
1420 ; ON ENTRY: DI : BIT 2 = HEAD; BITS 1,0 = DRIVE
1421 ;
1422 ; ON EXIT: DI : BIT 2 IS RESET, BITS 1,0 = DRIVE
1423 ; DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
1424 ;
-----
1425 069E READ_ID PROC NEAR
1426 069E B8 06B2 R MOV AX,OFFSET_ER_3 ; MOVE NEC OUTPUT ERROR ADDRESS
1427 06A1 50 PUSH AX
1428 06A2 B4 4A MOV AH,4AH ; READ ID COMMAND
1429 06A4 E8 07BD R CALL NEC_OUTPUT ; TO CONTROLLER
1430 06A7 8B C7 MOV AX,DI ; DRIVE # TO AH, HEAD 0
1431 06A9 8A E0 MOV AH,AL
1432 06AB E8 07BD R CALL NEC_OUTPUT ; TO CONTROLLER
1433 06AE E8 0530 R CALL NEC_TERM ; WAIT FOR OPERATION, GET STATUS
1434 06B1 58 POP AX ; THROW AWAY ERROR ADDRESS
1435 06B2 ER_3:
1436 06B2 C3 RET
1437 06B3 READ_ID ENDP
-----
1438 ;
1439 ; CMOS_TYPE: RETURNS DISKETTE TYPE FROM CMOS
1440 ;
1441 ; ON ENTRY: DI : DRIVE #
1442 ;
1443 ; ON EXIT: AL = TYPE (IF VALID) ; CY REFLECTS STATUS
1444 ;
-----
1445 06B3 CMOS_TYPE PROC NEAR
1446 06B3 B0 0E MOV AL,CMOS_DIAG ; CMOS DIAGNOSTIC STATUS BYTE ADDRESS
1447 06B5 E8 0000 E CALL CMOS_READ ; GET CMOS STATUS
1448 06B8 A8 C0 TEST AL,BAD_BAT+BAD_CKSUM ; BATTERY GOOD AND CHECKSUM VALID ?
1449 06BA F9 STC ; SET CY = 1 INDICATING ERROR FOR RETURN
1450 06BB 75 0E JNZ CMOS_T9 ; ERROR EXIT IF EITHER ERROR BIT WAS ON
1451

```

```

1452 06D0 B0 10          MOV    AL,CMOS_DISKETTE    ; ADDRESS OF DISKETTE BYTE IN CMOS
1453 06BF E8 0000 E     CALL   CMOS_READ          ; GET DISKETTE BYTE
1454 06C2 0B FF         OR     D1,DT              ; SEE WHICH DRIVE IN QUESTION
1455 06C4 75 03        JNZ   CMOS_T5            ; IF DRIVE 1, DATA IN LOW NIBBLE
1456
1457 06C6 C0 C8 04      ROR    AL,4              ; EXCHANGE NIBBLES IF SECOND DRIVE
1458 06C9
1459 06C9 24 0F        AND    AL,00FH           ; KEEP ONLY DRIVE DATA, RESET CY = 0
1460 06CB              CMOS_T9:
1461 06CB C3            RET                     ; CY = STATUS OF READ
1462 06CC              CMOS_TYPE ENDP
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 ;
1470 ; ON ENTRY: DL = INDEX OF BYTE TO BE FETCHED
1471 ;
1472 ; ON EXIT: AH = THAT BYTE FROM BLOCK
1473 ;          AL,DH DESTROYED
-----
1474 06CC              GET_PARM PROC NEAR
1475 06CC 1E          PUSH  DS
1476 06CD 56          PUSH  SI
1477 06CE 2B C0      SUB   AX,AX              ; DS = 0 , BIOS DATA AREA
1478 06D0 8E D8      MOV   DS,AX
1479 06D2 87 D3      XCHG DX,BX              ; BL = INDEX
1480 06D4 2A FF      SUB   DS,BH              ; BX = INDEX
1481
1482 06D6 C5 36 0078 R  LDS   SI,%DISK_POINTER  ; POINT TO BLOCK
1483 06DA 8A 20      MOV   AH,[SI+BX]        ; GET THE WORD
1484 06DC 87 D3      XCHG DX,BX              ; RESTORE BX
1485 06DE 5E        POP   SI
1486 06DF 1F        POP   DS
1487 06E0 C3        RET
1488
1489 06E1              GET_PARM DS:DATA ENDP
-----
1491 ; MOTOR_ON : TURN MOTOR ON AND WAIT FOR MOTOR START UP TIME. THE %MOTOR_COUNT
1492 ; IS REPLACED WITH A SUFFICIENTLY HIGH NUMBER (%FFH) TO ENSURE
1493 ; THAT THE MOTOR DOES NOT GO OFF DURING THE OPERATION. IF THE
1494 ; MOTOR NEEDED TO BE TURNED ON, THE MULTI-TASKING HOOK FUNCTION
1495 ; (%X=90FDH, INT 15H) IS CALLED TELLING THE OPERATING SYSTEM
1496 ; THAT THE BIOS IS ABOUT TO WAIT FOR MOTOR START UP. IF THIS
1497 ; FUNCTION RETURNS WITH CY = 1, IT MEANS THAT THE MINIMUM WAIT
1498 ; HAS BEEN COMPLETED. AT THIS POINT A CHECK IS MADE TO ENSURE
1499 ; THAT THE MOTOR WASN'T TURNED OFF BY THE TIMER. IF THE HOOK DID
1500 ; NOT WAIT, THE WAIT FUNCTION (%AH=066H) IS CALLED TO WAIT THE
1501 ; PRESCRIBED AMOUNT OF TIME. IF THE CARRY FLAG IS SET ON RETURN,
1502 ; IT MEANS THAT THE FUNCTION IS IN USE AND DID NOT PERFORM THE
1503 ; WAIT. A TIMER I WAIT LOOP WILL THEN DO THE WAIT.
1504
1505 ; ON ENTRY: DI = DRIVE #
1506 ;
1507 ; ON EXIT: AX,BX,CX,DX DESTROYED
-----
1509 06E1              MOTOR_ON PROC NEAR
1510 06E1 E8 072A R     CALL   TURN_ON           ; TURN ON MOTOR
1511 06E4 72 43        JNC   AX,T5_ON           ; IF CY=1 NO WAIT
1512 06E6 E8 02EE R     CALL   XLAT_OLD          ; TRANSLATE STATE TO COMPATIBLE MODE
1513 06E9 B8 90FD     MOV   AX,090FDH         ; LOAD WAIT CODE & TYPE
1514 06EC CD 15        INT   15H               ; TELL OPERATING SYSTEM ABOUT TO DO WAIT
1515 06EE 9C          PUSHF
1516 06EF E8 02C8 R     CALL   XLAT_NEW         ; TRANSLATE STATE TO PRESENT ARCH.
1517 06F2 9D          POPF                      ; RESTORE CY FOR TEST
1518 06F3 73 05        JNC   M_WAIT            ; BYPASS LOOP IF OP SYSTEM HANDLED WAIT
1519 06F5 E8 072A R     CALL   TURN_ON           ; CHECK AGAIN IF MOTOR ON
1520 06F8 72 2F        JCC   MOT_TS_ON         ; IF NO WAIT MEANS IT IS ON
1521
1522 06FA              M_WAIT:
1523 06FA B2 0A        MOV   DL,10             ; GET THE MOTOR WAIT PARAMETER
1524 06FC E8 06CC R     CALL   GET_PARM         ; AL = MOTOR WAIT PARAMETER
1525 06FF 8A C4        MOV   AL,AH             ; AX = MOTOR WAIT PARAMETER
1526 0701 32 E4        XOR   AH,AH             ; SEE IF AT LEAST A SECOND IS SPECIFIED
1527 0703 3C 08        CMP   AL,8              ; IF YES, CONTINUE
1528 0705 73 02        JAE   GP2               ; IF NO WAIT WAIT FOR MOTOR START UP
1529 0707 B0 08        MOV   AL,8
1530
1531 ;----- AX CONTAINS NUMBER OF 1/8 SECONDS (125000 MICROSECONDS) TO WAIT
1532
1533 0709 50          GP2:  PUSH  AX              ; SAVE WAIT PARAMETER
1534 070A BA F424     MOV   DX,62500          ; LOAD LARGEST POSSIBLE MULTIPLIER
1535 070D F7 E2      MUL   DX                 ; MULTIPLY BY HALF OF WHAT'S NECESSARY
1536 070F 8B CA      MOV   CX,DX             ; CX = HIGH WORD
1537 0711 8B D0      MOV   DX,AX             ; CX,DX = 1/2 * (# OF MICROSECONDS)
1538 0713 F8        CLC                     ; CLEAR CARRY FOR ROTATE
1539 0714 D1 D2      RCL   DX,1              ; DOUBLE LOW WORD, CY CONTAINS OVERFLOW
1540 0716 D1 D1      RCL   CX,1              ; DOUBLE HI, INCLUDING LOW WORD OVERFLOW
1541 0718 B4 B6      MOV   AH,86H            ; LOAD WAIT CODE
1542 071A CD 15      INT   15H               ; PERFORM WAIT
1543 071C 58        POP   AX                 ; RESTORE WAIT PARAMETER
1544 071D 73 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              J13:
1549 071F B9 205E     MOV   CX,8286           ; WAIT FOR 1/8 SECOND (AL)
1550 0722 E8 0000 E     CALL  WAITF             ; COUNT FOR 1/8 SECOND AT 15.085737 US
1551 0725 FE C8      DEC   AL                ; DECREMENT TIME VALUE
1552 0727 F6 F6      JNZ   J13               ; ARE WE DONE YET
1553
1554 0729              MOT_IS_ON:
1555 0729 C3          MOTOR_ON RET
1556 072A              MOTOR_ON ENDP
-----
1557 ; TURN_ON : TURN MOTOR ON AND RETURN WAIT STATE.
1558 ;
1559 ; ON ENTRY: DI = DRIVE #
1560 ;
1561 ; ON EXIT: CY = 0 MEANS WAIT REQUIRED
1562 ;          CY 1 MEANS NO WAIT REQUIRED
1563 ;          AX,BX,CX,DX DESTROYED
-----

```

```

1566 072A          TURN_ON PROC          NEAR
1567 072A          MOV          BX,DI
1568 072C          MOV          CL,BL
1569 072E          ROL          BL,4
1570 0731          CLI
1571 0732          MOV          AL,MOTOR_STATUS
1572 0737          AND          AL,003F R
1573 073A          AND          AL,0010000B
1574 073C          MOV          AH,1
1575 073E          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          CMP          AL,BL
1582 0742          JNZ          TEST
1583 0744          MOV          AH,MOTOR_STATUS
1584 0748          JNZ          NO_MOT_WAIT
1585
1586 074A          TURN_IT_ON:
1587 074A          MOV          AH,BL
1588 074C          SHV,MOTOR_STATUS
1589 0750          AND          BH,00001111B
1590 0753          OR          AL,MOTOR_STATUS,11001111B
1591 0758          OR          *MOTOR_STATUS,AH
1592 075C          MOV          AL,*MOTOR_STATUS
1593 075F          MOV          BL,*MOTOR_STATUS
1594 0761          AND          BL,00001111B
1595 0764          STI
1596 0765          AND          AL,00111111B
1597 0767          ROL          AL,4
1598 076A          OR          AL,00001100B
1599 076C          MOV          DX,03F2H
1600 076F          OUT          DX,AL
1601 0770          CMP          BL,BH
1602 0772          JZ          NO_MOT_WAIT
1603 0774          CLC
1604 0775          RET
1605
1606 0776          NO_MOT_WAIT:
1607 0776          STC
1608 0777          STI
1609 0778          RET
1610 0779          TURN_ON ENDP
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          MOV          DL,9
1620 077B          CALL          GET_PARM
1621 077E          TEST         *MOTOR_STATUS,10000000B
1622 0783          JZ          ISNT_WRITE
1623 0785          OR          AH,AH
1624 0787          JNZ          DO_WAIT
1625 0789          MOV          AH,HD12_SETTLE
1626 078B          MOV          AL,*MSK_STATE[DI]
1627 078F          AND          AL,RATE_MSK
1628 0791          CMP          AL,2 M DRIVE ?
1629 0793          JNZ          DO_WAIT
1630
1631 0795          MOV          AH,HD320_SETTLE
1632 0797          JMP          SHORT DO_WAIT
1633
1634 0799          ISNT_WRITE:
1635 0799          OR          AH,AH
1636 079B          JZ          HW_DONE
1637
1638          ;----- AH CONTAINS NUMBER OF MILLISECONDS TO WAIT
1639
1640 079D          DO_WAIT:
1641 079D          MOV          AL,AH
1642 079F          XOR          AH,AH
1643 07A1          PUSH         AX
1644 07A2          MOV          DX,1000
1645 07A5          MUL          DX
1646 07A7          MOV          CX,DX
1647 07A9          MOV          DX,AX
1648 07AB          MOV          AH,86H
1649 07AD          INT          15H
1650 07AF          POP          AX
1651 07B0          JNC          HW_DONE
1652
1653 07B2          JZ9:
1654 07B2          MOV          CX,66
1655 07B5          CALL          WAITF
1656 07B8          DEC          AL
1657 07BA          JNZ          JZ9
1658
1659 07BC          HW_DONE: 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
1669          ; ON EXIT: CY = 0 SUCCESS
1670          ; CY = 1 FAILURE -- DISKETTE STATUS UPDATED
1671          ; IF A FAILURE HAS OCCURRED, THE RETURN IS MADE ONE LEVEL
1672          ; HIGHER THAN THE CALLER OF NEC_OUTPUT. THIS REMOVES THE
1673          ; REQUIREMENT OF TESTING AFTER EVERY CALL OF NEC_OUTPUT.
1674          ; AX,BX,CX,DX DESTROYED
1675
1676 07BD          NEC_OUTPUT PROC          NEAR
1677 07BD          MOV          DX,03F4H
1678 07C0          MOV          BL,2
1679 07C2          XOR          CX,CX
    
```

```

1680
1681 07C4 EC          J23:  IN    AL,DX          ; GET STATUS
1682 07C5 24 C0      AND    AL,11000000B      ; KEEP STATUS AND DIRECTION
1683 07C7 3C 80      CMP    AL,10000000B      ; 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                ;----- FALL THRU TO ERROR RETURN
1691
1692 07D1 80 0E 0041 R 80  OR    #DSKETTE_STATUS,TIME_OUT
1693 07D6 58          POP    AX              ; DISCARD THE RETURN ADDRESS
1694 07D7 F9          STC                    ; INDICATE ERROR TO CALLER
1695 07D8 C3          RET
1696
1697                ;----- DIRECTION AND STATUS OK; OUTPUT BYTE
1698
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                    ; CY = 0 FROM TEST INSTRUCTION
1703 07DE          NEC_OUTPUT  ENDP
1704
1705                ;-----
1706                ; SEEK: 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                ;-----
1716 07DE          SEEK    PROC    NEAR
1717 07DE 0B DF      MOV    BX,DI          ; BX = DRIVE #
1718 07E0 BA 083D R  MOV    DX,OFFSET NEC_ERR ; LOAD RETURN ADDRESS
1719 07E4 80 01      PUSH  MOV    AL,1          ; ON STACK FOR NEC_OUTPUT ERROR
1720 07E6 86 CB      XCHG  CL,BL          ; ESTABLISH MASK FOR RECALIBRATE TEST
1721 07E8 D2 C0      ROL   CL,BL          ; GET DRIVE VALUE INTO CL
1722 07EA 86 CB      XCHG  CL,BL          ; SHIFT MASK BY THE DRIVE VALUE
1723 07EC 84 06 003E R TEST  AL,#SEEK_STATUS ; RECOVER TRACK VALUE
1724 07F0 75 1C      JNZ   J26A           ; TEST FOR RECALIBRATE REQUIRED
1725                ; JUMP IF RECALIBRATE NOT REQUIRED
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 73 0A      JNC  AFT_RECAL       ; RECALIBRATE DONE
1729
1730                ;----- ISSUE RECALIBRATE FOR 80 TRACK DISKETTES
1731
1732 07FB C6 06 0041 R 00 MOV  #DSKETTE_STATUS,0 ; CLEAR OUT INVALID STATUS
1733 0800 E8 083E R  CALL  RECAL          ; RECALIBRATE DRIVE
1734 0803 72 37      JC   RB              ; IF RECALIBRATE FAILS TWICE THEN ERROR
1735
1736                ;
1737 0805          AFT_RECAL:
1738 0805 C6 85 0094 R 00 MOV  #DSK_TRK[DI],0    ; SAVE NEW CYLINDER AS PRESENT POSITION
1739 080A 0A ED      OR    CH,CH          ; CHECK FOR SEEK TO TRACK 0
1740 080C 74 29      JZ   DO_WAIT         ; HAD SETTLE, CY = 0 IF JUMP
1741
1742                ;----- DRIVE IS IN SYNCHRONIZATION WITH CONTROLLER, SEEK TO TRACK
1743
1744 080E F6 85 0090 R 20 J28A: TEST #DSK_STATE[DI],DBL_STEP ; CHECK FOR DOUBLE STEP REQUIRED
1745 0813 74 02      JZ   SHL             ; SINGLE STEP REQUIRED BYPASS DOUBLE
1746 0815 D0 E5      SHL  CH,1           ; DOUBLE NUMBER OF STEP TO TAKE
1747
1748 0817 3A AD 0094 R RT:  CMP    CH,#DSK_TRK[DI] ; SEE IF ALREADY AT THE DESIRED TRACK
1749 0818 B8 74 1F      JB   RB              ; IF YES, DO NOT NEED TO SEEK
1750 081D 88 AD 0094 R MOV    #DSK_TRK[DI],CH ; SAVE NEW CYLINDER AS PRESENT POSITION
1751 0821 B4 0F      MOV    AH,0FH        ; SEEK COMMAND TO NEC
1752 0823 E8 07BD R  CALL  NEC_OUTPUT
1753 0826 0B DF      MOV    BX,DI          ; BX = DRIVE #
1754 0828 8A E3      MOV    AH,BL          ; OUTPUT DRIVE NUMBER
1755 082A E8 07BD R  CALL  NEC_OUTPUT
1756 082D 8A A5 0094 R MOV    AH,#DSK_TRK[DI] ; GET CYLINDER NUMBER
1757 0831 E8 07BD R  CALL  NEC_OUTPUT
1758 0834 E8 0855 R  CALL  CHK_STAT_2     ; ENDING INTERRUPT AND SENSE STATUS
1759
1760                ;----- WAIT FOR HEAD SETTLE
1761
1762 0837          DO_WAIT:
1763 0837 9C          PUSHF
1764 0838 E8 0779 R  CALL  HD_WAIT        ; SAVE STATUS
1765 083B 9D          POPFD                ; WAIT FOR HEAD SETTLE TIME
1766 083C          CALL  RESTORE_STATUS ; RESTORE STATUS
1767 083C 58          POP    AX            ; CLEAR ERROR RETURN FROM NEC_OUTPUT
1768 083D          NEC_ERR:
1769 083D C3          RET                    ; RETURN TO CALLER
1770 083E          SEEK    ENDP
1771
1772                ;-----
1773                ; RECAL: RECALIBRATE DRIVE
1774                ;
1775                ; ON ENTRY:  DI = DRIVE #
1776                ;
1777                ; ON EXIT:  CY REFLECTS STATUS OF OPERATION.
1778                ;-----
1779 083E          RECAL  PROC    NEAR
1780 083F B8 0853 R  MOV    AX,OFFSET RC_BACK ; LOAD NEC_OUTPUT ERROR
1781 0842 50          PUSH  AX              ; ON STACK FOR NEC_OUTPUT ERROR
1782 0843 B4 07      MOV    AH,07H        ; RECALIBRATE COMMAND
1783 0845 E8 07BD R  CALL  NEC_OUTPUT
1784 0848 0B DF      MOV    BX,DI          ; BX = DRIVE #
1785 084A 8A E3      MOV    AH,BL          ; OUTPUT DRIVE NUMBER
1786 084C E8 07BD R  CALL  NEC_OUTPUT
1787 084F E8 0855 R  CALL  CHK_STAT_2     ; GET THE INTERRUPT AND SENSE INT STATUS
1788 0852 58          POP    AX            ; THROW AWAY ERROR
1789 0853          RC_BACK:
1790 0853 59          POP    CX
1791 0854 C3          RET
1792 0855          RECAL  ENDP
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      |
1798      | ON EXIT:  *DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION.
1799      |-----|
1800 0855      CHK_STAT_2  PROC NEAR
1801 0855 B8 0873 R    MOV AX,OFFSET CS_BACK      | LOAD NEC_OUTPUT ERROR ADDRESS
1802 0858 50          AX PUSH                    |
1803 0859 E8 087C R    CALL WAIT_INT              | WAIT FOR THE INTERRUPT
1804 085C 72 14      JC J34                | IF ERROR, RETURN IT
1805 085E B4 08      MOV AH,08H                 | SENSE INTERRUPT STATUS COMMAND
1806 0860 E8 07BD R    CALL NEC_OUTPUT            |
1807 0863 E8 08A4 R    CALL RESULTS              | READ IN THE RESULTS
1808 0866 72 0A      JC J34                |
1809 0868 A0 0042 R    MOV AL,*NEC_STATUS        | GET THE FIRST STATUS BYTE
1810 086B 24 60      AND AL,01100000B         | ISOLATE THE BITS
1811 086D 3C 60      CMP AL,01100000B         | TEST FOR CORRECT VALUE
1812 086F 74 03      JZ J35                | IF ERROR, GO MARK IT
1813 0871 F8        CLC                          | GOOD RETURN
1814 0872
1815 0872 58          J34: POP AX                | THROW AWAY ERROR RETURN
1816 0873
1817 0873 C3          CS_BACK: RET
1818
1819 0874
1820 0874 80 0E 0041 R 40 J35: OR *DSKETTE_STATUS,BAD_SEEK | ERROR RETURN CODE
1821 0879 F9          STC
1822 087A EB F6      JMP SHORT J34
1823 087C
1824      CHK_STAT_2  ENDP
1825      |-----|
1826      | WAIT_INT: THIS ROUTINE WAITS FOR AN INTERRUPT TO OCCUR A TIME OUT ROUTINE
1827      |          TAKES PLACE DURING THE WAIT, SO THAT AN ERROR MAY BE RETURNED
1828      |          IF THE DRIVE IS NOT READY.
1829      |
1830      | ON EXIT:  *DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION.
1831      |-----|
1832 087C          WAIT_INT  PROC NEAR
1833 087C F8        CLC                          | CLEAR TIMEOUT INDICATOR
1834 087E B8 9001    MOV AX,09001H            | LOAD WAIT CODE AND TYPE
1835 0881 CD 15      INT 15H                  | PERFORM OTHER FUNCTION
1836 0883 72 11      JC J36A                 | BYPASS TIMING LOOP IF TIMEOUT DONE
1837
1838 0885 B3 04      MOV BL,4                 | CLEAR THE COUNTERS
1839 0887 33 C9      XOR CX,CX                | FOR 2 SECOND WAIT
1840 0889
1841 0889 F6 06 003E R 80 J36: TEST *SEEK_STATUS,INT_FLAG | TEST FOR INTERRUPT OCCURRING
1842 088E 75 0C      JNZ J37                 |
1843 0890 E2 F7      LOOP J36                | COUNT DOWN WHILE WAITING
1844 0892 FE CB      DEC BL                   | SECOND LEVEL COUNTER
1845 0894 75 F3      JNZ J36                |
1846
1847 0896 80 0E 0041 R 80 J36A: OR *DSKETTE_STATUS,TIME_OUT | NOTHING HAPPENED
1848 089B F9        STC                          | ERROR RETURN
1849 089C
1850 089C 9C        J37: PUSHF                       | SAVE CURRENT CARRY
1851 089D 80 26 003E R 7F AND *SEEK_STATUS,NOT_INT_FLAG | TURN OFF INTERRUPT FLAG
1852 08A2 9D        POPF                      | RECOVER CARRY
1853 08A3 C3        RET                          | GOOD RETURN CODE
1854 08A4
1855      WAIT_INT  ENDP
1856      |-----|
1857      | RESULTS: THIS ROUTINE WILL READ ANYTHING THAT THE NEC CONTROLLER RETURNS
1858      |          FOLLOWING AN INTERRUPT.
1859      |
1860      | ON EXIT:  *DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION.
1861      |          AX,BX,CX,DX DESTROYED
1862      |-----|
1863 08A4          RESULTS  PROC NEAR
1864 08A4 57        PUSH DI
1865 08A5 BF 0042 R    MOV DI,OFFSET *NEC_STATUS | POINTER TO DATA AREA
1866 08A8 B3 07      MOV BL,7                 | MAX STATUS BYTES
1867 08AA BA 03F4     MOV DX,03F4H             | STATUS PORT
1868
1869      |-----| WAIT FOR REQUEST FOR MASTER
1870 08AD B7 02      MOV BH,2                 | HIGH ORDER COUNTER
1871 08AF 33 C9      XOR CX,CX                | COUNTER
1872 08B1
1873 08B1 EC        J39: IN AL,DX              | GET STATUS
1874 08B2 24 C0      AND AL,11000000B        | KEEP ONLY STATUS AND DIRECTION
1875 08B4 3C 00      CMP AL,11000000B        | STATUS 1 AND DIRECTION 0 ?
1876 08B6 74 0E      JZ J42                  | STATUS AND DIRECTION OK
1877 08B8 E2 F7      LOOP J39                | LOOP TILL TIMEOUT
1878
1879 08BA FE CF      DEC BH                   | DECREMENT HIGH ORDER COUNTER
1880 08BC 75 F3      JNZ J39                 | REPEAT TILL DELAY DONE
1881
1882 08BE 80 0E 0041 R 80 OR *DSKETTE_STATUS,TIME_OUT |
1883 08C3 F9        STC                          | SET ERROR RETURN
1884 08C4 EB 1B      JMP SHORT POPRES        | POP REGISTERS AND RETURN
1885
1886      |-----| READ IN THE STATUS
1887
1888 08C6          J42: INC DX                    | POINT AT DATA PORT
1889 08C6 42        IN AL,DX                 | GET THE DATA
1890 08C7 EC        DEC [DI],AL              | STORE THE BYTE
1891 08C8 88 05      MOV DI,DI                | INCREMENT THE POINTER
1892 08CA 47        INC DI
1893
1894 08CB B9 0002     MOV CX,2                 | MINIMUM 12 MICROSECONDS FOR NEC
1895 08CE E8 0000 E    CALL WAITF              | WAIT 15 TO 30 MICROSECONDS
1896 08D1 4A        DEC DX                   | POINT AT STATUS PORT
1897 08D2 EC        IN AL,DX                 | GET STATUS
1898 08D3 A8 10     TEST AL,00010000B       | TEST FOR NEC STILL BUSY
1899 08D5 74 0A     JZ POPRES                | RESULTS DONE ?
1900
1901 08D7 FE CB      DEC BL                   | DECREMENT THE STATUS COUNTER
1902 08D9 75 D2     JNZ R10                  | GO BACK FOR MORE
1903 08DB 80 0E 0041 R 20 OR *DSKETTE_STATUS,BAD_NEC | TOO MANY STATUS BYTES
1904 08E0 F9        STC                          | SET ERROR FLAG
1905
1906      |-----| RESULT OPERATION IS DONE
1907
    
```

SECTION 5

```

1908 08E1          POPRES:
1909 08E1 5F      POP        DI
1910 08E2 C3      RET
1911 08E3          ; RETURN WITH CARRY SET
1912          -----
1913          ; READ_DSKCHNG: READS THE STATE OF THE DISK CHANGE LINE.
1914          ;
1915          ; ON ENTRY:   DI = DRIVE #
1916          ;
1917          ; ON EXIT:   DI = DRIVE #
1918          ;             ZERO FLAG = 0 : DISK CHANGE LINE INACTIVE
1919          ;             ZERO FLAG = 1 : DISK CHANGE LINE ACTIVE
1920          ;             AX,CX,DX DESTROYED
1921          ;
1922 08E3          READ_DSKCHNG  PROC  NEAR
1923 08E3 E8 06E1 R CALL  MOTOR_ON          ; TURN ON THE MOTOR IF OFF
1924 08E6 BA 03F7 F 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          ; DRIVE_DET:   DETERMINES WHETHER DRIVE IS 80 OR 40 TRACKS AND UPDATES STATE
1931          ;             INFORMATION ACCORDINGLY.
1932          ;
1933          ; ON ENTRY:   DI = DRIVE #
1934          ;
1935 08ED          DRIVE_DET    PROC  NEAR
1936 08ED E8 06E1 R CALL  MOTOR_ON          ; TURN ON MOTOR IF NOT ALREADY ON
1937 08F0 E8 083E R CALL  RECAL              ; RECALIBRATE DRIVE
1938 08F3 72 3C    JC     DD_BAC           ; ASSUME NO DRIVE PRESENT
1939 08F5 B5 30    MOV   CH,TRK_SLAP      ; SEEK TO TRACK 48
1940 08F7 E8 07DE R CALL  SEEK              ; SEEK TO TRACK 48
1941 08FA 72 35    JC     DD_BAC           ; ?
1942 08FC B5 0B    MOV   CH,QUIET_SEEK+1 ; ERROR NO DRIVE
1943 08FE          ; SEEK TO TRACK 10
1944 08FE FE CD   DEC   CH                ; DECREMENT TO NEXT TRACK
1945 0900 51      PUSH  CX                ; SAVE TRACK
1946 0901 E8 07DE R CALL  SEEK              ;
1947 0904 72 2C    JC     POP_BAC         ; POP AND RETURN
1948 0906 B8 0931 R MOV   AX,OFFSET DD_BAC ; LOAD NEC OUTPUT ERROR ADDRESS
1949 0909 50      PUSH  AX                ;
1950 090A B4 04    MOV   AH,SENSE_DRV_ST  ; SENSE DRIVE STATUS COMMAND BYTE
1951 090C E8 07BD R CALL  NEC_OUTPUT       ; OUTPUT TO NEC
1952 090F 8B C7    MOV   AX,DI            ; AL = DRIVE
1953 0911 8A E0    MOV   AH,AL           ; AH = DRIVE
1954 0913 E8 07BD R CALL  NEC_OUTPUT       ; OUTPUT TO NEC
1955 0916 E8 08A4 R CALL  RESULTS          ; GO GET STATUS
1956 0919 58      POP   AX                ; THROW AWAY ERROR ADDRESS
1957 091A 59      POP   CX                ; RESTORE TRACK
1958 091B F6 06 0042 R 10 TEST  0NEC_STATUS,HOME ; TRACK 0 ?
1959 0920 74 DC    JZ     SK_GTN          ; GO TILL TRACK 0
1960 0922 0A ED    OR     CH,CH           ; IS HOME AT TRACK 0 ?
1961 0924 74 06    JZ     IS_80          ; MUST BE 80 TRACK DRIVE
1962          ;
1963          ; DRIVE IS A 360; SET DRIVE TO DETERMINED;
1964          ; SET MEDIA TO DETERMINED AT RATE 250.
1965          ;
1966 0926 80 8D 0090 R 94 OR     0DSK_STATE[DI],DRV_DET+MED_DET+RATE_250
1967 092B C3      RET                ; ALL INFORMATION SET
1968          ;
1969 092C          IS_80:
1970 092C 80 8D 0090 R 01 OR     0DSK_STATE[DI],TRK_CAPA ; SETUP 80 TRACK CAPABILITY
1971 0931          DD_BAC:
1972 0931 C3      RET
1973          ;
1974 0932          POP_BAC:
1975 0932 59      POP   CX                ; THROW AWAY
1976 0933 C3      RET
1977          ;
1978 0934          DRIVE_DET    ENDP
    
```

```

1979          PAGE
1980          ;--- HARDWARE INT 08 H -- ( IRQ LEVEL 6 ) -----
1981          ;
1982          ; DISK_INT          THIS ROUTINE HANDLES THE DISKETTE INTERRUPT.
1983          ;
1984          ; ON EXIT:          THE INTERRUPT FLAG IS SET IN #SEEK_STATUS.
1985          ;
1986          ;-----
1987
1988 0934          DISK_INT_1          PROC          FAR          ; ENTRY POINT FOR ORG 0EF57H
1989 0934 FB          STI          ; RE-ENABLE INTERRUPTS
1990 0935 50          PUSH         AX          ; SAVE WORK REGISTER
1991 0936 1E          PUSH         DS          ; SAVE REGISTERS
1992 0937 E8 0000 E          CALL         DDS          ; SETUP DATA ADDRESSING
1993 093A 80 0E 003E R 00          OR          #SEEK_STATUS,INT_FLAG ; TURN ON INTERRUPT OCCURRED
1994 093F 1F          POP          DS          ; RESTORE USER (DS)
1995 0940 20          MOV          AL,EDI          ; END OF INTERRUPT MARKER
1996 0942 E6 20          OUT          INTA00,AL ; INTERRUPT CONTROL PORT
1997 0944 B8 9101          MOV          AX,09101H ; INTERRUPT POST CODE AND TYPE
1998 0947 CD 15          INT          15H          ; GO PERFORM OTHER TASK
1999 0949 58          POP          AX          ; RECOVER REGISTER
2000 094A CF          IRET         ; RETURN FROM INTERRUPT
2001
2002 094B          DISK_INT_1          ENDP
2003
2004          ;-----
2005          ; DISKETTE_SETUP: THIS ROUTINE DOES A PRELIMINARY CHECK TO SEE WHAT TYPE OF
2006          ; DISKETTE DRIVES ARE ATTACH TO THE SYSTEM.
2007          ;-----
2008
2009
2010 094B          DSKETTE_SETUP PROC          NEAR          ; SAVE REGISTERS
2011 094B 50          PUSH         AX
2012 094C 53          PUSH         BX
2013 094D 51          PUSH         CX
2014 094E 52          PUSH         DX
2015 094F 57          PUSH         DI
2016 0950 1E          PUSH         DS
2017 0951 E8 0000 E          CALL         DDS          ; POINT DATA SEGMENT TO BIOS DATA AREA
2018 0954 80 0E 00A0 R 01          OR          #RTC_WAIT_FLAG,01 ; NO RTC WAIT, FORCE USE OF LOOP
2019 0959 33 FF          XOR          DI,DI          ; INITIALIZE DRIVE POINTER
2020 095B C7 06 0090 R 0000          MOV          WORD PTR #DSK_STATE,0 ; INITIALIZE STATES
2021 0961 80 26 008B R 33          AND          #LAstrate,NOT #STRT_MSK+SEND_MSK ; CLEAR START & SEND
2022 0966 80 0E 008B R 00          OR          #LAstrate,SEND_MSK ; INITIALIZE SENT TO IMPOSSIBLE
2023 096B C6 06 003E R 00          MOV          #SEEK_STATUS,0 ; INDICATE RECALIBRATE NEEDED
2024 0970 C6 06 0040 R 00          MOV          #MOTOR_COUNT,0 ; INITIALIZE MOTOR COUNT
2025 0975 C6 06 003F R 00          MOV          #MOTOR_STATUS,0 ; INITIALIZE DRIVES TO OFF STATE
2026 097A C6 06 0041 R 00          MOV          #DSKETTE_STATUS,0 ; NO ERRORS
2027
2028 097F          SUP0:
2029 097F E8 08ED R          CALL         DRIVE_DET ; DETERMINE DRIVE
2030 0982 E8 02EE R          CALL         XLAT_OLD ; TRANSLATE STATE TO COMPATIBLE MODE
2031 0985 47          INC          DI          ; POINT TO NEXT DRIVE
2032 0986 83 FF 02          CMP          DI,MAX_DRV ; SEE IF DONE
2033 0989 75 F4          JNZ         SUP0 ; REPEAT FOR EACH DRIVE
2034 098B C6 06 003E R 00          MOV          #SEEK_STATUS,0 ; FORCE RECALIBRATE
2035 0990 80 26 00A0 R FE          AND          #RTC_WAIT_FLAG,0FEH ; ALLOW FOR RTC WAIT
2036 0995 E8 0620 R          CALL         SETUP_END ; VARIOUS CLEANUPS
2037 0998 1F          POP          DS          ; RESTORE CALLERS REGISTERS
2038 0999 5F          POP          DI
2039 099A 5A          POP          DX
2040 099B 59          POP          CX
2041 099C 5B          POP          BX
2042 099D 58          POP          AX
2043 099E C3          RET
2044
2045 099F          DSKETTE_SETUP ENDP
2046
2047 099F          CODE          ENDS
2048          END
    
```

1
 2
 3
 4
 5 0000
 6
 7
 8
 9
 10
 11
 12
 13
 14
 15
 16
 17
 18
 19
 20
 21
 22
 23
 24
 25
 26
 27
 28
 29
 30
 31
 32
 33
 34
 35
 36
 37
 38
 39
 40
 41
 42
 43
 44
 45
 46
 47
 48
 49
 50
 51
 52
 53
 54
 55
 56
 57
 58
 59
 60
 61
 62
 63
 64
 65
 66
 67
 68
 69
 70
 71
 72
 73
 74
 75
 76
 77
 78
 79
 80
 81
 82
 83
 84
 85
 86
 87
 88
 89
 90
 91
 92
 93
 94
 95
 96
 97
 98

```

PAGE 118,121
TITLE DISK ----- 06/10/85 FIXED DISK BIOS
.286C
.LIST
CODE SEGMENT BYTE PUBLIC

PUBLIC DISK_IO
PUBLIC DISK_SETUP
PUBLIC HD_INT

EXTRN CMOS_READ:NEAR
EXTRN CMOS_WRITE:NEAR
EXTRN DDS:NEAR
EXTRN E_MSG:NEAR
EXTRN F1780:NEAR
EXTRN F1781:NEAR
EXTRN F1782:NEAR
EXTRN F1790:NEAR
EXTRN F1791:NEAR
EXTRN FD_TBL:NEAR
    
```

```

-----
INT 13H
-----
FIXED DISK I/O INTERFACE
:
:
: THIS INTERFACE PROVIDES ACCESS TO 5 1/4" FIXED DISKS THROUGH
: THE IBM FIXED DISK CONTROLLER.
:
: THE BIOS ROUTINES ARE MEANT TO BE ACCESSED THROUGH
: SOFTWARE INTERRUPTS ONLY. ANY ADDRESSES PRESENT IN
: THESE LISTINGS ARE INCLUDED ONLY FOR COMPLETENESS,
: NOT FOR REFERENCE. APPLICATIONS WHICH REFERENCE ANY
: ABSOLUTE ADDRESSES WITHIN THE CODE SEGMENTS OF BIOS
: VIOLATE THE STRUCTURE AND DESIGN OF BIOS.
:
:-----
    
```

```

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

```

REGISTERS USED FOR FIXED DISK OPERATIONS
:
: (DL) - DRIVE NUMBER (80H-81H FOR DISK, VALUE CHECKED)
: (DH) - HEAD NUMBER (0-15 ALLOWED, NOT VALUE CHECKED)
: (CH) - CYLINDER NUMBER (0-1023, NOT VALUE CHECKED) (SEE CL)
: (CL) - SECTOR NUMBER (1-17, NOT VALUE CHECKED)
:
: NOTE: HIGH 2 BITS OF CYLINDER NUMBER ARE PLACED
: IN THE HIGH 2 BITS OF THE CL REGISTER
: (10 BITS TOTAL)
:
: (AL) - NUMBER OF SECTORS (MAXIMUM POSSIBLE RANGE 1-80H,
: FOR READ/WRITE LONG 1-79H)
:
: (ES:BX) - ADDRESS OF BUFFER FOR READS AND WRITES,
: (NOT REQUIRED FOR VERIFY)
:
: FORMAT (AH=5) ES:BX POINTS TO A 512 BYTE BUFFER. THE FIRST
: 2*(SECTORS/TRACK) BYTES CONTAIN F,N FOR EACH SECTOR.
: F = 00H FOR A GOOD SECTOR
: 80H FOR A BAD SECTOR
: N = SECTOR NUMBER
: FOR AN INTERLEAVE OF 2 AND 17 SECTORS/TRACK
: THE TABLE SHOULD BE:
:
: DB 00H,01H,00H,0AH,00H,02H,00H,0BH,00H,03H,00H,0CH
: DB 00H,04H,00H,0DH,00H,05H,00H,0EH,00H,06H,00H,0FH
: DB 00H,07H,00H,10H,00H,08H,00H,11H,00H,09H
:
:-----
    
```



```

99          PAGE
100         ;-----
101         ; OUTPUT
102         ; AH = STATUS OF CURRENT OPERATION
103         ; STATUS BITS ARE DEFINED IN THE EQUATES BELOW
104         ; CY = 0 SUCCESSFUL OPERATION (AH=0 ON RETURN)
105         ; CY = 1 FAILED OPERATION (AH HAS ERROR REASON)
106         ;
107         ; NOTE: ERROR 11H INDICATES THAT THE DATA READ HAD A RECOVERABLE
108         ; ERROR WHICH WAS CORRECTED BY THE ECC ALGORITHM. THE DATA
109         ; IS PROBABLY GOOD, HOWEVER THE BIOS ROUTINE INDICATES AN
110         ; ERROR TO ALLOW THE CONTROLLING PROGRAM A CHANCE TO DECIDE
111         ; FOR ITSELF. THE ERROR MAY NOT RECUR IF THE DATA IS
112         ; REWRITTEN.
113         ;
114         ; IF DRIVE PARAMETERS WERE REQUESTED (DL >= 80H),
115         ; INPUT:
116         ; (DL) = DRIVE NUMBER
117         ; OUTPUT:
118         ; (DL) = NUMBER OF CONSECUTIVE ACKNOWLEDGING DRIVES ATTACHED (1-2)
119         ; (CONTROLLER CARD ZERO TALLY ONLY)
120         ; (DH) = MAXIMUM USEABLE VALUE FOR HEAD NUMBER
121         ; (CH) = MAXIMUM USEABLE VALUE FOR CYLINDER NUMBER
122         ; (CL) = MAXIMUM USEABLE VALUE FOR SECTOR NUMBER
123         ; AND CYLINDER NUMBER HIGH BITS
124         ;
125         ; IF READ DASH TYPE WAS REQUESTED,
126         ;
127         ; AH = 0 - NOT PRESENT
128         ; 1 - DISKETTE - NO CHANGE LINE AVAILABLE
129         ; 2 - DISKETTE - CHANGE LINE AVAILABLE
130         ; 3 - FIXED DISK
131         ; CX,DX = NUMBER OF 512 BYTE BLOCKS WHEN AH = 3
132         ;
133         ; REGISTERS WILL BE PRESERVED EXCEPT WHEN THEY ARE USED TO RETURN
134         ; INFORMATION.
135         ;
136         ; NOTE: IF AN ERROR IS REPORTED BY THE DISK CODE, THE APPROPRIATE
137         ; ACTION IS TO RESET THE DISK, THEN RETRY THE OPERATION.
138         ;-----

```

```

141
142 = 00FF          SENSE FAIL          EQU 0FFH          ; NOT IMPLEMENTED
143 = 00E0          NO_ERR              EQU 0E0H          ; STATUS ERROR/ERROR REGISTER=0
144 = 00CC          WRITE_FAULT        EQU 0CCH          ; WRITE FAULT ON SELECTED DRIVE
145 = 00BB          UNDEF_ERR          EQU 0BBH          ; UNDEFINED ERROR OCCURRED
146 = 00AA          NOT_RDY            EQU 0AAH          ; DRIVE NOT READY
147 = 0080          TIME_OUT           EQU 80H           ; ATTACHMENT FAILED TO RESPOND
148 = 0040          BAD_SEEK           EQU 40H           ; SEEK OPERATION FAILED
149 = 0020          BAD_CNTRLR         EQU 20H           ; CONTROLLER HAS FAILED
150 = 0011          DATA_CORRECTED   EQU 11H           ; ECC CORRECTED DATA ERROR
151 = 0010          BAD_ECC            EQU 10H           ; BAD ECC ON DISK READ
152 = 000B          BAD_TRACK          EQU 0BH           ; NOT IMPLEMENTED
153 = 000A          BAD_SECTOR         EQU 0AH           ; BAD SECTOR FLAG DETECTED
154 = 0009          DMA_BOUNDARY       EQU 09H           ; DATA EXTENDS TOO FAR
155 = 0007          INIT_FAIL          EQU 07H           ; DRIVE PARAMETER ACTIVITY FAILED
156 = 0005          BAD_RESET          EQU 05H           ; RESET FAILED
157 = 0004          RECRDR_NOT_FND     EQU 04H           ; REQUESTED SECTOR NOT FOUND
158 = 0002          BAD_ADDR_MARK      EQU 02H           ; ADDRESS MARK NOT FOUND
159 = 0001          BAD_CMD            EQU 01H           ; BAD COMMAND PASSED TO DISK I/O
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

```

```

-----
;
; FIXED DISK PARAMETER TABLE
;
; - THE TABLE IS COMPOSED OF A BLOCK DEFINED AS:
;
; +0 (1 WORD) - MAXIMUM NUMBER OF CYLINDERS
; +2 (1 BYTE) - MAXIMUM NUMBER OF HEADS
; +3 (1 WORD) - NOT USED/SEE PC-XT
; +5 (1 WORD) - STARTING WRITE PRECOMPENSATION CYL
; +7 (1 BYTE) - MAXIMUM ECC DATA BURST LENGTH
; +8 (1 BYTE) - CONTROL BYTE
; BIT 7 DISABLE RETRIES -OR-
; BIT 6 DISABLE RETRIES
; BIT 5 MORE THAN 8 HEADS
; +9 (3 BYTES) - NOT USED/SEE PC-XT
; +12 (1 WORD) - LANDING ZONE
; +14 (1 BYTE) - NUMBER OF SECTORS/TRACK
; +15 (1 BYTE) - RESERVED FOR FUTURE USE
;
; - TO DYNAMICALLY DEFINE A SET OF PARAMETERS
; BUILD A TABLE FOR UP TO 15 TYPES AND PLACE
; THE CORRESPONDING VECTOR INTO INTERRUPT 41
; FOR DRIVE 0 AND INTERRUPT 46 FOR DRIVE 1.
;
-----

```

```

188                                     PAGE
189                                     -----
190                                     ;
191                                     ;   HARDWARE SPECIFIC VALUES
192                                     ;
193                                     ;   - CONTROLLER I/O PORT
194                                     ;
195                                     ;
196                                     ;   > WHEN READ FROM:
197                                     ;   HF_PORT+0 - READ DATA (FROM CONTROLLER TO CPU)
198                                     ;   HF_PORT+1 - GET ERROR REGISTER
199                                     ;   HF_PORT+2 - GET SECTOR COUNT
200                                     ;   HF_PORT+3 - GET SECTOR NUMBER
201                                     ;   HF_PORT+4 - GET CYLINDER LOW
202                                     ;   HF_PORT+5 - GET CYLINDER HIGH (2 BITS)
203                                     ;   HF_PORT+6 - GET SIZE/DRIVE/HEAD
204                                     ;   HF_PORT+7 - GET STATUS REGISTER
205                                     ;
206                                     ;   > WHEN WRITTEN TO:
207                                     ;   HF_PORT+0 - WRITE DATA (FROM CPU TO CONTROLLER)
208                                     ;   HF_PORT+1 - SET PRECOMPENSATION CYLINDER
209                                     ;   HF_PORT+2 - SET SECTOR COUNT
210                                     ;   HF_PORT+3 - SET SECTOR NUMBER
211                                     ;   HF_PORT+4 - SET CYLINDER LOW
212                                     ;   HF_PORT+5 - SET CYLINDER HIGH (2 BITS)
213                                     ;   HF_PORT+6 - SET SIZE/DRIVE/HEAD
214                                     ;   HF_PORT+7 - SET COMMAND REGISTER
215                                     ;
216                                     -----
217 = 01F0      HF_PORT      EQU      01F0H      ; DISK PORT
218 = 03F6      HF_REG_PORT  EQU      03F6H
219
220                                     ;-----
221                                     ;   STATUS REGISTER
222 = 0001      ST_ERROR     EQU      00000001B  ;
223 = 0002      ST_INDEX     EQU      00000010B  ;
224 = 0004      ST_CORRCTD   EQU      00000100B  ; ECC CORRECTION SUCCESSFUL
225 = 0008      ST_DRQ       EQU      00001000B  ;
226 = 0010      ST_SEEK_COMPL EQU      00010000B  ; SEEK COMPLETE
227 = 0020      ST_WRT_FLT   EQU      00100000B  ; WRITE FAULT
228 = 0040      ST_READY     EQU      01000000B  ;
229 = 0080      ST_BUSY      EQU      10000000B  ;
230
231                                     ;-----
232                                     ;   ERROR REGISTER
233 = 0001      ERR_DAM       EQU      00000001B  ; DATA ADDRESS MARK NOT FOUND
234 = 0002      ERR_TRK_0    EQU      00000010B  ; TRACK 0 NOT FOUND ON RECAL
235 = 0004      ERR_ABORT    EQU      00000100B  ; ABORTED COMMAND
236 = 0008      I            EQU      00001000B  ; NOT USED
237 = 0010      ERR_ID       EQU      00010000B  ; ID NOT FOUND
238 = 0020      I            EQU      00100000B  ; NOT USED
239 = 0040      ERR_DATA_ECC EQU      01000000B  ;
240 = 0080      ERR_BAD_BLOCK EQU      10000000B  ;
241
242
243 = 0010      RECAL_CMD     EQU      00010000B  ; DRIVE RECAL (10H)
244 = 0020      READ_CMD     EQU      00100000B  ; READ (20H)
245 = 0030      WRITE_CMD    EQU      00110000B  ; WRITE (30H)
246 = 0040      VERIFY_CMD   EQU      01000000B  ; VERIFY (40H)
247 = 0050      FMTRK_CMD    EQU      01010000B  ; FORMAT TRACK (50H)
248 = 0060      INIT_CMD     EQU      01100000B  ; INITIALIZE (60H)
249 = 0070      SEEK_CMD     EQU      01110000B  ; SEEK (70H)
250 = 0090      DIAG_CMD     EQU      10010000B  ; DIAGNOSTIC (90H)
251 = 0091      SET_PARM_CMD  EQU      10010001B  ; DRIVE PARMS (91H)
252 = 0001      ND_RETRIEVE  EQU      00000001B  ; CMD MODIFIER (01H)
253 = 0002      ECC_MODE     EQU      00000010B  ; CMD MODIFIER (02H)
254 = 0008      BUFFER_MODE  EQU      00001000B  ; CMD MODIFIER (08H)
255
256 = 0002      MAX_FILE      EQU      2
257 = 0002      S_MAX_FILE   EQU      2
258
259 = 0025      DELAY_1       EQU      25H      ; DELAY FOR OPERATION COMPLETE
260 = 0600      DELAY_2       EQU      0600H    ; DELAY FOR READY
261 = 0100      DELAY_3       EQU      0100H    ; DELAY FOR DATA REQUEST
262
263 = 0008      HF_FAIL       EQU      08H      ; CMOS FLAG IN BYTE 0EH
264
265                                     ;-----
266                                     ;   COMMAND BLOCK REFERENCE
267 =          *CMD_BLOCK     EQU      BYTE PTR [BP]-8 ; *CMD BLOCK REFERENCES BLOCK HEAD IN $$
268                                     ; [BP] POINTS TO COMMAND BLOCK TAIL
269                                     ; AS DEFINED BY THE "ENTER" PARMS
    
```

```

270                                     PAGE
271                                     |-----|
272                                     | FIXED DISK I/O SETUP                               |
273                                     |-----|
274                                     | - ESTABLISH TRANSFER VECTORS FOR THE FIXED DISK      |
275                                     | - PERFORM POWER ON DIAGNOSTICS                          |
276                                     | - SHOULD AN ERROR OCCUR A "1701" MESSAGE IS DISPLAYED |
277                                     |-----|
278                                     | ASSUME CS:CODE,DS:ABS0                               |
279                                     |-----|
280                                     | WORK OFF DS REGISTER
281 0000                                DISK_SETUP PROC NEAR
282 0000 FA                                CLI
283 0001 B8 ---- R                        MOV AX,ABS0                                | GET ABSOLUTE SEGMENT
284 0004 8E D8                            MOV DS,AX                                | SET SEGMENT REGISTER
285 0006 A1 004C R                        MOV AX,WORD PTR @ORG_VECTOR              | GET DISKETTE VECTOR
286 0009 A3 0108 R                        WORD PTR @DISK_VECTOR,AX                | INTO INT 40H
287 000C A1 004E R                        MOV AX,WORD PTR @ORG_VECTOR+2
288 000F A3 0102 R                        MOV WORD PTR @DISK_VECTOR+2,AX
289 0012 C7 06 004C R 01A9 R             MOV WORD PTR @ORG_VECTOR,OFFSET DISK_10 | FIXED DISK HANDLER
290 0018 8C 0E 004E R                     MOV WORD PTR @ORG_VECTOR+2,CS
291 001C C7 06 01D8 R 06CD R             MOV WORD PTR @HDISK_INT,OFFSET HD_INT   | FIXED DISK INTERRUPT
292 0022 8C 0E 01DA R                     MOV WORD PTR @HDISK_INT+2,CS
293 0026 C7 06 0104 R 0000 E            MOV WORD PTR @HF_TBL_VEC,OFFSET FD_TBL  | PARM TABLE DRIVE 80
294 002C C7 06 0106 R                     MOV WORD PTR @HF_TBL_VEC+2,CS
295 0030 C7 06 0118 R 0000 E            MOV WORD PTR @HF1_TBL_VEC,OFFSET FD_TBL | PARM TABLE DRIVE 81
296 0036 8C 0E 011A R                     MOV WORD PTR @HF1_TBL_VEC+2,CS
297 003A E4 A1                             IN AL,INTB01                             | TURN ON SECOND INTERRUPT CHIP
298 003C 24 BF                             AND AL,0BFH
299 003E EB 00                             JMP $+2
300 0040 E6 A1                             OUT INTB01,AL
301 0042 E4 21                             IN AL,INTA01                             | LET INTERRUPTS PASS THRU TO
302 0044 24 FB                             AND AL,0FBH                             | SECOND CHIP
303 0046 EB 00                             JMP $+2
304 0048 E6 21                             OUT INTA01,AL
305
306 004A FB                                STI
307                                     ASSUME DS:DATA,ES:ABS0
308 004B 1E                                PUSH DS
309 004C 07                                POP ES
310 004D E8 0000 E                        CALL DDS
311 0050 C6 06 0074 R 00                 MOV @DISK_STATUS1,0
312 0055 C6 01 0075 R 00                 MOV @HF_NUM,0
313 005A C6 06 0076 R 00                 MOV @CONTROL_BYTE,0
314 005F B0 8E                            MOV AL,CMOS_DTAG+NM1
315 0061 E8 0000 E                        CALL CMOS_READ
316 0064 8A F0                            MOV AH,AL
317 0066 24 C0                            AND AL,BAD_BAT+BAD_CKSUM
318 0068 74 03                            JZ L1
319 006A E9 00FB R                        JMP L1
320 006D                                     L1:
321 006D 80 E4 F7                         AND AH,NOT HF_FAIL
322 0070 B0 8E                            MOV AL,CMOS_DTAG+NM1
323 0072 E8 0000 E                        CALL CMOS_WRITE
324 0075 B0 92                            MOV AL,CMOS_DISK+NM1
325 0077 E8 0000 E                        CALL CMOS_READ
326 007A C6 06 0077 R 00                 MOV @PORT_OFF,0
327 007F 8A D8                            MOV BL,AL
328 0081 25 00F0 F                        AND AX,00F0H
329 0084 74 72                            JZ L2
330
331 0086 3C F0                             CMP AL,0F0H
332 0088 75 10                             JNE L2
333
334 008A B0 99                             MOV AL,CMOS_DISK_1+NM1
335 008C E8 0000 E                        CALL CMOS_READ
336 008F 3C 00                             CMP AL,0
337 0091 74 6F                             JE L3
338 0093 3C 2F                             CMP AL,4
339 0095 71 11                             JA L4
340 0097 C1 E0 04                         SHL AX,4
341 009A                                     L2:
342 009A 05 FFF0 E                        ADD AX,OFFSET FD_TBL-16D
343 009D 26 A3 0104 R                     MOV WORD PTR @HF_TBL_VEC,AX
344 00A1 C6 06 0075 R 01                 MOV @HF_NUM,1
345 00A6 8A C3                             MOV AL,BL
346 00AB C0 E0 04                         SHL AL,4
347 00AB 74 2A                             JZ SHORT L4
348 00AD B4 00                             MOV AH,0
349
350 00AF 3C F0                             CMP AL,0F0H
351 00B1 75 10                             JNE L3
352
353 00B3 B0 9A                             MOV AL,CMOS_DISK_2+NM1
354 00B5 E8 0000 E                        CALL CMOS_READ
355 00B8 3C 00                             CMP AL,0
356 00BA 74 1B                             JE L4
357 00BC 3C 2F                             CMP AL,4
358 00BE 71 17                             JA L4
359 00C0 C1 E0 04                         SHL AX,4
360 00C3                                     L3:
361 00C3 05 FFF0 E                        ADD AX,OFFSET FD_TBL-16D
362 00C6 8B D8                            MOV BX,AX
363 00C8 2E : 83 3F 00                   CMP WORD PTR CS:[BX],0
364 00CC 74 09                             JE L4
365 00CE 26 A3 0118 R                     MOV WORD PTR @HF1_TBL_VEC,AX
366 00D2 C6 06 0075 R 02                 MOV @HF_NUM,2
367 00D7                                     L4:
368 00D7 B2 80                             MOV DL,80H
369 00D9 B4 14                             MOV AH,14H
370 00DB CD 13                             INT 13H
371 00DD 72 1A                             JC CTL_ERRX
372 00DF A1 006C R                        MOV AX,RTIMER_LOW
373 00E2 8B D8                            MOV BX,AX
374 00E4 05 0444                         ADD AX,6*182
375 00E7 8B C8                            MOV CX,AX
376 00E9 E8 1104 R                       MOV EDI,1104H
377 00EC B0 3E 0075 R 01                 CMP @HF_NUM,T
378 00F1 76 05                             JBE L4
379 00F3 B2 81                             MOV DL,81H
380 00F5 E8 0104 R                       MOV EDI,0104H
381 00F8                                     POD_DONE:
382 00F8 C3                                RET
383

```

SECTION 5

```

384 ;---- POD ERROR
385
386 00F9 CTL_ERRX:
387 00F9 BE 0000 E MOV SI,OFFSET F17&2 ; CONTROLLER ERROR
388 00FC E8 017C R CALL SET_FAIL ; DO NOT IPL FROM DISK
389 00FF E8 0000 E E_MSG ; DISPLAY ERROR AND SET (BP) ERROR FLAG
390 0102 EB 014 R JMP POD_DONE
391
392
393 0104 HD_RESET_I PROC NEAR
394 0104 53 PUSH BX ; 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 T2 06 JC RES_2
399 010C B4 11 MOV AH,T1H ; RECALIBRATE DRIVE
400 010E CD 13 INT 13H
401 0110 73 19 JNC RES_OK ; DRIVE OK
402 0112 E8 018A R RES_2: CALL POD_TCHK ; CHECK TIME OUT
403 0115 73 EF JNC RES_1
404 0117 BE 0000 E MOV SI,OFFSET F17&1 ; INDICATE DISK 1 FAILURE
405 011A F6 C2 01 TEST DL,1
406 011D 75 57 JNZ RES_E1
407 011F BE 0000 E MOV SI,OFFSET F17&0 ; INDICATE DISK 0 FAILURE
408 0122 E8 017C R CALL SET_FAIL ; DO NOT TRY TO IPL DISK 0
409 0125 EB 4F JMP SHORT_RES_E1
410 0127 B4 00 MOV AH,00H ; RESET THE DRIVE
411 0129 CD 13 INT 13H
412 012B B4 08 MOV AH,08H ; GET MAX CYLINDER,HEAD,SECTOR
413 012D 8A DA MOV BL,DL ; SAVE DRIVE CODE
414 012F CD 13 INT 13H
415 0131 72 38 JC RES_ER
416 0133 89 0E 0042 R MOV WORD PTR @NEC_STATUS,CX ; SAVE MAX CYLINDER, SECTOR
417 0137 8A D3 MOV DL,BL ; RESTORE DRIVE CODE
418 0139 B8 0401 MOV AX,0401H ; VERIFY THE LAST SECTOR
419 013C CD 13 INT 13H
420 013E 73 39 JNC RES_OK ; VERIFY OK
421 0140 80 FC 0A CMP AH,BAD_SECTOR ; OK ALSO IF JUST ID READ
422 0143 74 34 JE RES_OK
423 0145 80 FC 11 CMP AH,DATA_CORRECTED
424 0148 74 2F JE RES_OK
425 014A 80 FC 10 CMP AH,BAD_ECC
426 014D 74 2A JE RES_OK
427 014F E8 018A R CALL POD_TCHK ; CHECK FOR TIME OUT
428 0152 72 17 JC RES_ER ; FAILED
429 0154 8B 0E 0042 R CX,WORD PTR @NEC_STATUS ; GET SECTOR ADDRESS, AND CYLINDER
430 0158 8A C1 MOV AL,CL ; SEPARATE OUT SECTOR NUMBER
431 015A 24 3F AND AL,3FH
432 015C FE C8 DEC AL ; TRY PREVIOUS ONE
433 015E 74 C7 JZ RES_RS ; WE'VE TRIED ALL SECTORS ON TRACK
434 0160 80 E1 C0 AND CL,0C0H ; KEEP CYLINDER BITS
435 0163 0A C8 CL_AL ; MERGE SECTOR WITH CYLINDER BITS
436 0165 89 0E 0042 R MOV WORD PTR @NEC_STATUS,CX ; SAVE CYLINDER, NEW SECTOR NUMBER
437 0169 EB CE JMP RES_3 ; TRY AGAIN
438 016B BE 0000 E RES_ER: MOV SI,OFFSET F1791 ; INDICATE DISK 1 ERROR
439 016E F6 C2 01 TEST DL,1
440 0171 75 03 JNZ RES_E1
441 0173 BE 0000 E MOV SI,OFFSET F1790 ; INDICATE DISK 0 ERROR
442 0176 74 16 JC RES_ER
443 0178 E8 0000 E CALL E_MSG ; DISPLAY ERROR AND SET (BP) ERROR FLAG
444 0179
445 0179 59 POP CX ; RESTORE TIMER LIMITS
446 017A 5B POP BX
447 017B C3 RET
448 017C
449
450 017C HD_RESET_I ENDP
451 017C B8 BE8E SET_FAIL PROC NEAR
452 017F E8 0000 E MOV AX,X*(CMOS_DIAG+NM1) ; GET CMOS ERROR BYTE
453 0182 0C 08 CALL CMOS_READ ; GET CMOS ERROR BYTE
454 0184 86 E0 OR AL,HF_FAIL ; SET DO NOT IPL FROM DISK FLAG
455 0186 E8 0000 E CALL XCHG ; SAVE IT
456 0189 C3 RET ; PUT IT OUT
457 018A SET_FAIL ENDP
458
459 018A POD_TCHK PROC NEAR
460 018A 58 POP AX ; CHECK FOR 30 SECOND TIME OUT
461 018B 59 POP CX ; SAVE RETURN
462 018C 5B POP BX ; GET TIME OUT LIMITS
463 018D 53 PUSH BX ; AND SAVE THEM AGAIN
464 018E 51 PUSH CX
465 018F 50 PUSH AX
466 0190 A1 006C R MOV AX,@TIMER_LOW ; RESTORE RETURN
467 ; AX = CURRENT TIME
468 ; BX = START TIME
469 ; CX = END TIME
470 0195 T2 06 JB TCHK1 ; START < END
471 0197 3B D8 CMP BX,AX ; END < START < CURRENT
472 0199 72 0C JB TCHK2 ; END, CURRENT < START
473 019B EB 04 JMP SHORT_TCHK2
474 019D 3B C3 TCHK1: CMP AX,BX
475 019F 72 04 JB TCHKNG ; CURRENT < START < END
476 01A1 3B C1 TCHK2: CMP AX,CX
477 01A3 72 02 JB TCHKG ; START < CURRENT < END
478 ; OR CURRENT < END < START
479 01A5 F9 TCHKNG: STC ; CARRY SET INDICATES TIME OUT
480 01A6 C3 RET
481 01A7 F8 TCHKG: CLC ; INDICATE STILL TIME
482 01A8 C3 RET
483 01A9
484
485 01A9 POD_TCHK ENDP
486 01A9 DISK_SETUP ENDP

```

```

486 PAGE
487 |-----|
488 | FIXED DISK BIOS ENTRY POINT |
489 |-----|
490
491 01A9 DISK_10 PROC FAR
492 01A9 80 FA 80 ASSUME DS:DATA,ES:NOTHING ; TEST FOR FIXED DISK DRIVE
494 01AC 73 05 JAE A1 ; YES, HANDLE HERE
495 01AE CD 40 INT 40H ; DISKETTE HANDLER
496 01B0 RET_2: RET 2 ; BACK TO CALLER
497 01B0 CA 0002
498
499 01B3 A1: ; ENABLE INTERRUPTS
500 01B3 FB STI
501 01B4 0A E4 OR AH,AH
502 01B6 75 09 JNZ A2
503 01B8 CD 40 INT 40H ; RESET NEC WHEN AH=0
504 01BA 2A E4 SUB AH,AH
505 01BC 80 FA 81 CMP DL,(80H + S_MAX_FILE - 1)
506 01BF 77 EF JA RET_2
507 01C1
508 01C1 80 FC 08 A2: CMP AH,08H ; GET PARAMETERS IS A SPECIAL CASE
509 01C4 75 03 JNZ A3
510 01C6 E9 0393 R JMP GET_PARM_N
511 01C9 80 FC 15 A3: CMP AH,T5H ; READ DASD TYPE IS ALSO
512 01CC 75 03 JNZ
513 01CE E9 0353 R JMP READ_DASD_TYPE
514
515 01D1 A4: ; SAVE REGISTERS DURING OPERATION
516 01D1 C8 0008 00 ENTER 8,0 ; SAVE (BP) AND MAKE ROOM FOR %CMD_BLOCK
517 01D5 53 PUSH BX ; IN THE STACK. THE COMMAND_BLOCK IS:
518 01D6 51 PUSH CX ; %CMD_BLOCK == BYTE PTR [BP]-8
519 01D7 52 PUSH DX
520 01D8 1E PUSH DS
521 01D9 06 PUSH ES
522 01DA 56 PUSH SI
523 01DB 57 PUSH DI
524 01DC 0A E4 OR AH,AH ; CHECK FOR RESET
525 01DE 75 02 JNZ A5
526 01E0 B2 80 MOV DL,80H ; FORCE DRIVE 80 FOR RESET
527 01E2 EB 0225 R CALL DISK_IO_CONT ; PERFORM THE OPERATION
528 01E5 E8 0000 E CALL DDS ; ESTABLISH SEGMENT
529 01E8 8A 26 0074 R MOV AH,%DISK_STATUS1 ; GET STATUS FROM OPERATION
530 01EC 80 FC 01 CMP AH,1 ; SET THE CARRY FLAG TO INDICATE
531 01EF F5 CMC ; SUCCESS OR FAILURE
532 01F0 5F POP D1 ; RESTORE REGISTERS
533 01F1 5E POP SI
534 01F2 07 POP ES
535 01F3 1F POP DS
536 01F4 5A POP DX
537 01F5 59 POP CX
538 01F6 58 POP BX
539 01F7 C9 LEAVE ; 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 ; FUNCTION TRANSFER TABLE
544 01FB 02C1 R DW DISK_RESET ; 000H
545 01FD 0315 R DW RETURN_STATUS ; 001H
546 01FF 031E R DW DISK_READ ; 002H
547 0201 0325 R DW DISK_WRITE ; 003H
548 0203 032C R DW DISK_VERIFY ; 004H
549 0205 033E R DW FMT_TRK ; 005H
550 0207 02B9 R DW BAD_COMMAND ; 006H FORMAT BAD SECTORS
551 0209 02B9 R DW BAD_COMMAND ; 007H FORMAT DRIVE
552 020B 02B9 R DW BAD_COMMAND ; 008H RETURN PARAMETERS
553 020D 043F R DW INIT_DRY ; 009H
554 020F 0423 R DW RD_LONG ; 00AH
555 0211 042A R DW WR_LONG ; 00BH
556 0213 0431 R DW DISK_SEEK ; 00CH
557 0215 02C1 R DW DISK_RESET ; 00DH
558 0217 02B9 R DW BAD_COMMAND ; 00EH READ BUFFER
559 0219 02B9 R DW BAD_COMMAND ; 00FH WRITE BUFFER
560 021B 044F R DW TST_RDY ; 010H
561 021D 0466 R DW HDISK_RECAL ; 011H
562 021F 02B9 R DW BAD_COMMAND ; 012H MEMORY DIAGNOSTIC
563 0221 02B9 R DW BAD_COMMAND ; 013H DRIVE DIAGNOSTIC
564 0223 048E R DW CTRL_DIAGNOSTIC ; 014H CONTROLLER DIAGNOSTIC
565 = 002A MIL EQU $-M1
566
567 0225 DISK_IO_CONT PROC NEAR
568 0225 E8 0000 E CALL DDS ; ESTABLISH SEGMENT
569 0228 80 FC 01 CMP AH,01H ; RETURN STATUS
570 022B 75 03 JNZ SU0
571 022D E9 0315 R JMP RETURN_STATUS
572 0230
573 0230 C6 06 0074 R 00 SU0: MOV %DISK_STATUS1,0 ; RESET THE STATUS INDICATOR
574 0235 53 PUSH BX ; SAVE DATA ADDRESS
575 0236 8A 1E 0075 R MOV BL,%HF_NUM ; GET NUMBER OF DRIVES
576 023A 50 PUSH AX
577 023B 80 E2 7F AND DL,7FH ; GET DRIVE AS 0 OR 1
578 023C 3A DA CMP BL,DL
579 0240 76 75 JBE BAD_COMMAND_POP ; INVALID DRIVE
580 0242 06 PUSH ES
581 0243 E8 06B7 R CALL GET_VEC ; GET DISK PARAMETERS
582 0246 26 1 8B 47 05 MOV AX,WORD PTR ES:[BX][5] ; GET WRITE PRE-COMPENSATION CYLINDER
583 024A C1 E8 02 SHR AX,2 ; %CMD_BLOCK,AL
584 024D 88 46 F8 MOV AL,%CMD_BLOCK,AL ; GET CONTROL BYTE MODIFIER
585 0250 26 1 8A 47 08 MOV AL,%CONTROL_BYTE,AX ; GET CONTROL BYTE MODIFIER
586 0254 52 PUSH DX
587 0255 BA 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_BYTE ; SET EXTRA HEAD OPTION IN
592 025F 80 E4 C0 AND AH,0COH ; CONTROL BYTE
593 0262 0A E0 OR AH,AL
594 0264 88 26 0076 R MOV %CONTROL_BYTE,AX
595 0268 58 POP AX
596 0269 88 46 F9 MOV %CMD_BLOCK+1,AL ; SECTOR COUNT
597 026C 50 PUSH AX
598 026D 5A C1 MOV AL,CL ; GET SECTOR NUMBER
599 026F 24 3F AND AL,3FH
    
```

SECTION 5

```

600 0271 88 46 FA      MOV     @CMD_BLOCK+2,AL
601 0274 88 6E FB      MOV     @CMD_BLOCK+3,CH      ; GET CYLINDER NUMBER
602 0277 8A C1        MOV     AL,CL
603 0279 C0 E8 06      SHR     AL,6
604 027C 88 46 FC      MOV     @CMD_BLOCK+4,AL    ; CYLINDER HIGH ORDER 2 BITS
605 027F 8A C2        MOV     AL,DL              ; DRIVE NUMBER
606 0281 C0 E0 04      SHL     AL,4
607 0284 80 E6 0F      AND     DH,0FH            ; HEAD NUMBER
608 0287 0A C6        OR      AL,DH
609 0289 0C A0        OR      AL,80H OR 20H     ; ECC AND 512 BYTE SECTORS
610 028B 88 46 FD      MOV     @CMD_BLOCK+5,AL    ; ECC/SIZE/DRIVE/HEAD
611 028E 58           POP     AX
612 0291 50           PUSH   AX
613 0290 8A C4        MOV     AL,AH
614 0292 32 E4        XOR     AH,AH              ; GET INTO LOW BYTE
615 0294 D1 E0        SAL     AX,1              ; ZERO HIGH BYTE
616 0296 8B F0        MOV     SI,AX              ; *2 FOR TABLE LOOKUP
617 0298 3D 002A      CMP     AX,MIL             ; PUT INTO SI FOR BRANCH
618 029B 73 1A        JNB    BAD_COMMAND_POP    ; TEST WITHIN RANGE
619 029D 58           POP     AX
620 029E 5B           POP     BX
621 029F 51           PUSH   CX
622 02A0 50           PUSH   AX
623 02A1 8B CB        MOV     CX,BX              ; ADJUST ES:BX
624 02A3 C1 E9 04      SHR     CX,4              ; GET 3 HIGH ORDER NIBBLES OF BX
625 02A6 8C C0        MOV     AX,ES
626 02A8 03 C1        ADD     AX,CX
627 02AA 8E C0        MOV     ES,AX
628 02AC 81 E3 000F   AND    BX,000FH           ; ES:BX CHANGED TO ES:000X
629 02B0 58           POP     AX
630 02B1 59           POP     CX
631 02B2 2E1 FF A4 01FB R JMP     WORD PTR CS:[SI + OFFSET MI]
632 02B7
633 02B7 58           POP     AX
634 02B8 5B           POP     BX
635 02B9
636 02B9 C6 06 0074 R 01 MOV     @DISK_STATUS1,BAD_CMD ; COMMAND ERROR
637 02BE 80 00        MOV     AL,0
638 02C0 C3           RET
639 02C1
640
641
642
643
644
645 02C1
646 02C1 FA           CLI
647 02C2 E4 A1        IN     AL,INTB01          ; GET THE MASK REGISTER
648 02C4 EB 00        JMP     $+2
649 02C6 24 8F        AND    AL,0BFH           ; ENABLE FIXED DISK INTERRUPT
650 02C8 E4 A1        OUT    INTB01,AL
651 02CA FB           STI
652 02CB 80 04        MOV     AL,04H
653 02CD BA 03F6      MOV     DX,HF_REG_PORT   ; START INTERRUPTS
654 02D0 EE           OUT    DX,AL
655 02D1 B9 000A      MOV     CX,10            ; RESET
656 02D4 49           DEC     CX                ; DELAY COUNT
657 02D5 75 FD        JNZ    DRD
658 02D7 A0 0076 R   MOV     AL,@CONTROL_BYTE ; WAIT 4.8 MICRO-SEC
659 02DA 24 0F        AND    AL,0FH           ; SET HEAD OPTION
660 02DC EE           OUT    DX,AL
661 02DD E8 05E6 R   CALL   NOT_BUSY          ; TURN RESET OFF
662 02E0 75 2D        JNZ    DRERR             ; TIME OUT ON RESET
663 02E2 BA 01F1     MOV     DX,HF_PORT+1
664 02E5 EC           IN     AL,DX
665 02E6 3C 01        CMP     AL,1
666 02E8 75 25        JNZ    DRERR             ; GET RESET STATUS
667 02EA 80 66 FD EF   AND    @CMD_BLOCK+5,0EFH ; BAD RESET STATUS
668 02EE 2A D2        SUB    DL,DL             ; SET TO DRIVE 0
669 02F0 EA 03F1 R   CALL   INIT_DRY          ; SET MAX HEADS
670 02F3 E8 0466 R   CALL   HDISK_RECAL       ; RECAL TO RESET SEEK SPEED
671 02F6 80 3E 0075 R 01 CMP     @HF_NUM,1        ; CHECK FOR DRIVE 1
672 02F9 76 0C        JBE    DRE
673 02FD 80 4E FD 10 AND    @CMD_BLOCK+5,010H ; SET TO DRIVE 1
674 0301 B2 01        MOV     DL,1
675 0303 E8 03F1 R   CALL   INIT_DRY          ; SET MAX HEADS
676 0306 E8 0466 R   CALL   HDISK_RECAL       ; RECAL TO RESET SEEK SPEED
677 0309 C6 06 0074 R 00 MOV     @DISK_STATUS1,0  ; IGNORE ANY SET UP ERRORS
678 030E C3           RET
679 030F C6 06 0074 R 05 MOV     @DISK_STATUS1,BAD_RESET ; CARD FAILED
680 0314 C3           RET
681 0315
682
683
684
685
686
687
688 0315
689 0315 A0 0074 R   MOV     @DISK_STATUS1,0  ; OBTAIN PREVIOUS STATUS
690 031D C3           RET
691 031E
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
    
```

```

692 PAGE
693 ;-----
694 ; DISK READ ROUTINE (AH = 02H) ;
695 ;-----
696
697 031E DISK_READ PROC NEAR
698 031E C6 46 FE 20 MOV @CMD_BLOCK+6,READ_CMD
699 0322 E9 04C6 R JMP COMMAND1
700 0325 DISK_READ ENDP
701
702 ;-----
703 ; DISK WRITE ROUTINE (AH = 03H) ;
704 ;-----
705
706 0325 DISK_WRITE PROC NEAR
707 0325 C6 46 FE 30 MOV @CMD_BLOCK+6,WRITE_CMD
708 0329 E9 0505 R JMP COMMAND0
709 032C DISK_WRITE ENDP
710
711 ;-----
712 ; DISK VERIFY (AH = 04H) ;
713 ;-----
714
715 032C DISK_VERIFY PROC NEAR
716 032C C6 46 FE 40 MOV @CMD_BLOCK+6,VERIFY_CMD
717 0330 E8 054F R CALL COMMAND
718 0333 75 08 JNZ VERIFY_EXIT ; CONTROLLER STILL BUSY
719 0335 E8 05B5 R CALL WAIT ; TIME OUT
720 0338 75 03 JNZ VERIFY_EXIT
721 033A E8 0623 R CALL CHECK_STATUS
722 033D VERIFY_EXIT: RET
723 033D C3 DISK_VERIFY ENDP
724 033E
725
726 ;-----
727 ; FORMATTING (AH = 05H) ;
728 ;-----
729
730 033E FMT_TRK PROC NEAR ; FORMAT TRACK (AH = 005H)
731 033E C6 46 FE 50 MOV @CMD_BLOCK+6,FMTTRK_CMD
732 0342 06 PUSH ES
733 0343 53 PUSH BX
734 0344 E8 06B7 R CALL GET_VEC ; GET DISK PARAMETERS ADDRESS
735 0347 26 14 MOV AL,ES:[BX][14] ; GET SECTORS/TRACK
736 0348 88 46 F9 MOV @CMD_BLOCK+1,AL ; SET SECTOR COUNT IN COMMAND
737 034E 5B POP BX
738 034F 07 POP ES
739 0350 E9 050A R JMP CMD_DF ; GO EXECUTE THE COMMAND
740 0353 FMT_TRK ENDP
741
742 ;-----
743 ; READ DASD TYPE (AH = 15H) ;
744 ;-----
745
746
747 0353 READ_DASD_TYPE LABEL NEAR
748 0353 READ_D_T PROC FAR ; GET DRIVE PARAMETERS
749 0353 1E PUSH DS ; SAVE REGISTERS
750 0354 06 PUSH ES
751 0355 53 PUSH BX
752 ASSUME DS:DATA
753 0356 E8 0000 E CALL DDS ; ESTABLISH ADDRESSING
754 0359 C6 06 0074 R 00 MOV @DISK_STATUS1,0 ; GET NUMBER OF DRIVES
755 035E 8A 1E 0075 R MOV BL,@HF_NUM ; GET DRIVE NUMBER
756 0362 80 E2 7F AND DL,7FH ; GET DRIVE NUMBER
757 0365 3A DA CMP BL,DL
758 0367 76 22 JBE RDT_NOT_PRESENT ; RETURN DRIVE NOT PRESENT
759 0369 E8 06B7 R CALL GET_VEC ; GET DISK PARAMETER ADDRESS
760 036C 26 14 MOV AL,ES:[BX][2] ; HEADS
761 0370 26 14 MOV CL,ES:[BX][14]
762 0374 F6 E9 CL ; * NUMBER OF SECTORS
763 0376 26 14 MOV CX,ES:[BX] ; MAX NUMBER OF CYLINDERS
764 0379 49 DEC CX ; LEAVE ONE FOR DIAGNOSTICS
765 037A F7 E9 IMUL CX ; NUMBER OF SECTORS
766 037C 8B CA MOV CX,DX ; HIGH ORDER HALF
767 037E 8B D0 MOV DX,AX ; LOW ORDER HALF
768 0380 2B C0 SUB AX,AX
769 0382 84 03 MOV AH,03H ; INDICATE FIXED DISK
770 0384 5B POP BX ; RESTORE REGISTERS
771 0385 07 POP ES
772 0386 1F POP DS
773 0387 F8 CLC ; CLEAR CARRY
774 0388 CA 0002 RET 2
775 038B
776 038B RDT_NOT_PRESENT: SUB AX,AX ; DRIVE NOT PRESENT RETURN
777 038D 2B C0 MOV CX,AX ; ZERO BLOCK COUNT
778 038F 8B D0 MOV DX,AX
779 0391 EB F1 RDT2: POP ES
780 0393 READ_D_T ENDP
    
```

```

781                                     PAGE
782                                     |-----|
783                                     |   GET PARAMETERS   (AH = 08H) :
784                                     |-----|
785
786 0393                                GET_PARM_N LABEL NEAR
787 0393                                GET_PARM_N PROC FAR           ; GET DRIVE PARAMETERS
788 0393 1E                                PUSH DS           ; SAVE REGISTERS
789 0394 06                                PUSH ES
790 0395 53                                PUSH BX
791 0396 BB ---- R                        ASSUME DS:ABS0
792 0399 BE D8 ---- R                    MOV AX,ABS0           ; ESTABLISH ADDRESSING
793 0399 BE D8 ---- R                    MOV DS,AX
794 039B F6 C2 01                        TEST DL,1             ; CHECK FOR DRIVE 1
795 039E 74 01 0E                        JZ G1
796 03A0 C4 1E 0118 R                    LES BX,#HF1_TBL_VEC
797 03A4 EB 04                            JMP SHORT GT
798 03A6 C4 1E 0104 R                    LES BX,#HF_TBL_VEC
799
800 03AA                                G1:
801 03AA E8 0000 E                        CALL DDS             ; ESTABLISH SEGMENT
802 03AD 80 EA 80                        SUB DL,80H
803 03B0 80 0A 02                        CMP DL,MAX_FILE     ; TEST WITHIN RANGE
804 03B3 73 2C                            JAE G4
805 03B5 C6 06 0074 R 00                MOV #DISK_STATUS1,0
806 03BA 26 1B 07                        MOV AX,EST[BX]
807 03BD 2D 0002                        SUB AX,2             ; MAX NUMBER OF CYLINDERS
808 03C0 8A E8                            MOV CH,AL           ; ADJUST FOR 0-N
809 03C2 25 0300                        AND AX,0300H        ; HIGH TWO BITS OF CYLINDER
810 03C5 D1 E8                            SHR AX,1
811 03C7 D1 E8                            SHR AX,1
812 03C9 26 1A 47 0E                    OR AL,ES:[BX][14]   ; SECTORS
813 03CD 8A C8                            MOV CL,AL
814 03CF 26 1A 77 02                    MOV DH,ES:[BX][2]   ; HEADS
815 03D3 FE CE                            DEC DH              ; 0-N RANGE
816 03D5 8A 16 0075 R                    MOV DL,#HF_NUM
817 03D9 2B C0                            SUB AX,AX           ; DRIVE COUNT
818
819 03DB 5B                                G5:
820 03DC 07                                POP BX             ; RESTORE REGISTERS
821 03DD 1F                                POP ES
822 03DE CA 0002                        POP DS
823 03E1                                RET 2
824 03E1 C6 06 0074 R 07                MOV #DISK_STATUS1,INIT_FAIL ; OPERATION FAILED
825 03E6 B4 07                            MOV AH,INIT_FAIL
826 03E8 2A C0                            SUB AL,AL
827 03EA 2B D2                            SUB DX,DX
828 03EC 2B C9                            SUB CX,CX
829 03EE F9                                STC                ; SET ERROR FLAG
830 03EF EB EA                            JMP G5
831 03F1                                GET_PARM ENDP
832
833                                     |-----|
834                                     | INITIALIZE DRIVE   (AH = 09H) :
835                                     |-----|
836
837 03F1 C6 46 FE 91                      INIT_DRV PROC NEAR
838 03F5 E8 06B7 R                        MOV #CMD_BLOCK+8,SET_PARM_CMD
839 03FB 26 1A 8A 07 02                  CALL GET_VEC        ; ES:BX -> PARAMETER BLOCK
840 03FC FE C8                            DEC AL              ; GET NUMBER OF HEADS
841 03FE 8A 66 FD                            MOV AH,#CMD_BLOCK+5 ; CONVERT TO 0-INDEX
842 0401 80 E4 F0                            AND AH,0F0H        ; GET SDH REGISTER
843 0404 0A E0                            OR AH,AL            ; CHANGE HEAD NUMBER
844 0406 8B 66 FD                            MOV AH,#CMD_BLOCK+6 ; TO MAX HEAD
845 0409 26 1A 47 0E                    MOV DH,ES:[BX][14] ; MAX SECTOR NUMBER
846 040D 8B 46 F9                            MOV AX,AX
847 0410 2B C0                            SUB AX,AX           ; ZERO FLAGS
848 0412 8B 46 FB                            MOV AH,#CMD_BLOCK+3 ; TELL CONTROLLER
849 0415 EB 054F R                        CALL COMMAND        ; CONTROLLER BUSY ERROR
850 0418 75 08                            JNZ INIT_EXIT      ; CONTROLLER BUSY ERROR
851 041A E8 05E6 R                        CALL NOT_BUSY       ; WAIT FOR IT TO BE DONE
852 041D 75 03                            JNZ INIT_EXIT      ; TIME OUT
853 041F E8 0623 R                        CALL CHECK_STATUS
854 0422                                INIT_EXIT:
855 0422 C3                                RET
856 0423                                INIT_DRV ENDP
857
858                                     |-----|
859                                     |   READ LONG       (AH = 0AH) :
860                                     |-----|
861
862 0423                                RD_LONG PROC NEAR
863 0423 C6 46 FE 22                      MOV #CMD_BLOCK+8,READ_CMD OR ECC_MODE
864 0427 E9 04C6 R                        JMP COMMAND1
865 042A                                RD_LONG ENDP
866
867                                     |-----|
868                                     |   WRITE LONG      (AH = 0BH) :
869                                     |-----|
870
871 042A                                WR_LONG PROC NEAR
872 042A C6 46 FE 32                      MOV #CMD_BLOCK+6,WRITE_CMD OR ECC_MODE
873 042E E9 0505 R                        JMP COMMAND0
874 0431                                WR_LONG ENDP
875
876                                     |-----|
877                                     |   SEEK            (AH = 0CH) :
878                                     |-----|
879
880 0431                                DISK_SEEK PROC NEAR
881 0431 C6 46 FE 70                      MOV #CMD_BLOCK+6,SEEK_CMD
882 0435 E8 054F R                        CALL COMMAND
883 0438 75 14                            JNZ DS_EXIT        ; CONTROLLER BUSY ERROR
884 043A E8 05B5 R                        CALL WAIT           ; TIME OUT ON SEEK
885 043D 75 0F                            JNZ DS_EXIT
886 043F E8 0623 R                        CALL CHECK_STATUS
887 0442 80 3E 0074 R 40                  CMP #DISK_STATUS1,BAD_SEEK
888 0447 75 05                            JNE DS_EXIT
889 0449 C6 06 0074 R 00                  MOV #DISK_STATUS1,0
890 044E                                DS_EXIT:
891 044E C3                                RET
892
893 044F                                DISK_SEEK ENDP
    
```



```

894                                     PAGE
895                                     :-----:
896                                     : TEST DISK READY (AH = 10H) :
897                                     :-----:
898
899 044F TST_RDY PROC NEAR
900 044F EB 05E6 R CALL NOT_BUSY ; WAIT FOR CONTROLLER
901 0452 75 11 JNZ TR_EX ; ERROR
902 0454 BA 46 FD MOV AL,%CMD_BLOCK+5 ; 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 %DISK_STATUS1,0 ; WIPE OUT DATA CORRECTED ERROR
908 0465 C3 TR_EX: RET
909 0466 TST_RDY ENDP
910
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
917 046A EB 054F R CALL COMMAND ; START THE OPERATION
918 046D 75 19 JNZ RECAL_EXIT ; ERROR
919 046F EB 05B5 R CALL WAIT ; WAIT FOR COMPLETION
920 0472 74 05 JZ RECAL_X ; TIME OUT ONE OK ?
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 RECAL_X: CALL CHECK_STATUS
924 0479 E8 0623 R CMP %DISK_STATUS1,BAD_SEEK ; SEEK NOT COMPLETE
925 047C 80 3E 0074 R 40 JNE RECAL_EXIT ; 1 IS OK
926 0481 75 05 MOV %DISK_STATUS1,0
927 0483 C6 06 0074 R 00 MOV %DISK_STATUS1,0
928 0488 RECAL_EXIT: CMP %DISK_STATUS1,0
929 048B 80 3E 0074 R 00 JZ RET
930 048D C3 HDISK_RECAL ENDP
931 048E
932
933                                     :-----:
934                                     : CONTROLLER DIAGNOSTIC (AH = 14H) :
935                                     :-----:
936
937 048E CTLR_DIAGNOSTIC PROC NEAR
938 048E FA CLI ; DISABLE INTERRUPTS WHILE CHANGING MASK
939 048F E4 A1 IN AL,INTB01 ; TURN ON SECOND INTERRUPT CHIP
940 0491 24 BF AND AL,0BFH
941 0493 EB 00 JMP $+2
942 0495 E6 A1 OUT INTB01,AL
943 0497 E4 21 IN AL,INTA01 ; LET INTERRUPTS PASS THRU TO
944 0499 24 FB AND AL,0FBH ; SECOND CHIP
945 049B EB 00 JMP $+2
946 049D E6 21 OUT INTA01,AL
947 049F FB STI
948 04A0 EB 05E6 R CALL NOT_BUSY ; WAIT FOR CARD
949 04A3 75 1A JNZ CD_ERR ; BAD CARD
950 04A5 BA 01F7 MOV DX,BA 01F7
951 04A8 B0 90 MOV AL,DIAG_CMD ; START DIAGNOSE
952 04AB EA EE OUT DX,AL
953 04AB E8 05E6 R CALL NOT_BUSY ; WAIT FOR IT TO COMPLETE
954 04AE B4 80 MOV AH,TIME_OUT
955 04B0 75 0F JNZ CD_EXIT ; TIME OUT ON DIAGNOSTIC
956 04B2 BA 01F1 MOV DX,HF_PORT+1 ; GET ERROR REGISTER
957 04B5 EC IN AL,DX
958 04B6 A2 00BD 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 04BD 74 02 JE SHORT CD_EXIT
962 04BF B4 20 MOV AH,BAD_CNTRL
963 04C1 CD_ERR: MOV AH,BAD_CNTRL
964 04C1 88 26 0074 R MOV %DISK_STATUS1,AH
965 04C5 C3 RET
966 04C6 CTLR_DIAGNOSTIC ENDP
967
968                                     :-----:
969                                     : COMMAND1 :
970                                     : REPEATEDLY INPUTS DATA TILL :
971                                     : NSECTOR RETURNS ZERO :
972                                     :-----:
973 04C6 COMMAND1: CALL CHECK_DMA ; CHECK 64K BOUNDARY ERROR
974 04C6 EB 0694 R CALL CMD_ABORT
975 04C9 72 39 JNC DI BX
976 04CB 8B FB MOV DI,BX
977 04CD EB 054F R CALL COMMAND ; OUTPUT COMMAND
978 04D0 75 32 JNZ CMD_ABORT
979 04D2 CMD_11: CALL WAIT ; WAIT FOR DATA REQUEST INTERRUPT
980 04D2 E8 05B5 R CALL TM_OUT ; TIME OUT
981 04D5 75 2D JNZ CX,256D ; SECTOR SIZE IN WORDS
982 04D7 B9 0100 MOV MOV DX,256D
983 04DA BA 01F0 MOV DX,HF_PORT
984 04DD FA CLI
985 04DE FC CLD
986 04DF F3/6D REP ; GET THE SECTOR
987 04E1 FB STI
988 04E2 F6 46 FE 02 TEST %CMD_BLOCK+6,ECC_MODE ; CHECK FOR NORMAL INPUT
989 04E6 74 12 JZ WAIT_DRQ ; WAIT DRQ
990 04E8 EB 06D0 R CALL WAIT ; WAIT FOR DATA REQUEST
991 04EB 72 17 JNC TM_OUT
992 04ED BA 01F0 MOV DX,HF_PORT
993 04F0 B9 0004 MOV CX,4
994 04F3 EC IN AL,DX
995 04F4 26: 88 05 MOV ES:BYTE PTR [DI],AL ; GO SLOW FOR BOARD
996 04F7 41 INC DI
997 04F8 E2 F9 LOOP CMD_12
998 04FA EB 0623 R CALL CHECK_STATUS
999 04FD 75 05 JNZ CMD_ABORT ; ERROR RETURNED
1000 04FF FE 4E F9 DEC %CMD_BLOCK+1 ; CHECK FOR MORE
1001 0502 75 CE JNZ SHORT CMD_11
1002 0504 CMD_ABORT:
1003 0504 TM_OUT:
1004 0504 C3 RET
    
```

```

1005 PAGE
1006 |-----|
1007 | COMMAND |
1008 | REPEATEDLY OUTPUTS DATA TILL |
1009 | NSECTOR RETURNS ZERO |
1010 |-----|
1011 0505 COMMAND0:
1012 0505 E8 0694 R CALL CHECK_DMA ; CHECK 64K BOUNDARY ERROR
1013 0508 72 FA JC CMD_ABORT
1014 050A 8B F3 JZ S1,BX
1015 050C E8 054F R CALL COMMAND ; OUTPUT COMMAND
1016 050F 75 F3 JNZ CMD_ABORT
1017 0511 E8 060D R CALL WAIT_DRQ ; WAIT FOR DATA REQUEST
1018 0514 72 EE JC TM_OUT ; TOO LONG
1019 0516 1E JZ DS
1020 0517 06 PUSH ES ; MOVE ES TO DS
1021 0518 1F POP DS
1022 0519 B9 0100 MOV CX,256D ; PUT THE DATA OUT TO THE CARD
1023 051C BA 01F0 MOV DX,HF_PORT
1024 051F FA CLI
1025 0520 FC CLD
1026 0521 F3/ 6F REP OUTSW
1027 0523 FB STI
1028 0524 1F POP DS ; RESTORE DS
1029 0525 F6 46 FE 02 TEST *CMD_BLOCK+6,ECC_MODE ; CHECK FOR NORMAL OUTPUT
1030 0529 74 12 JZ CMD_03
1031 052B E8 060D R CALL WAIT_DRQ ; WAIT FOR DATA REQUEST
1032 052E 72 D4 JC TM_OUT
1033 0530 BA 01F0 MOV DX,HF_PORT
1034 0533 B9 0004 MOV CX,4 ; OUTPUT THE ECC BYTES
1035 0536 26 8A 04 CMD_02: MOV AL,ES:BYTE PTR [SI]
1036 0539 EE OUT DX,AL
1037 053A 46 INC SI
1038 053B E2 F9 LOOP CMD_02
1039 053D CMD_03:
1040 053D CALL WAIT ; WAIT FOR SECTOR COMPLETE INTERRUPT
1041 0540 75 C2 JNZ TM_OUT ; ERROR RETURNED
1042 0542 E8 0623 R CALL CHECK_STATUS
1043 0545 75 BD JNZ CMD_ABORT
1044 0547 F6 06 008C R 08 TEST *STATUS,ST_DRQ ; CHECK FOR MORE
1045 054C 75 C8 JNZ SHORT_CMD_01
1046 054E C3 RET
1047
1048 |-----|
1049 | COMMAND |
1050 | THIS ROUTINE OUTPUTS THE COMMAND BLOCK |
1051 | OUTPUT |
1052 | BL = STATUS |
1053 | BH = ERROR REGISTER |
1054 |-----|
1055
1056 054F COMMAND PROC NEAR
1057 054F 53 PUSH BX
1058 0550 B9 0600 MOV CX,DELAY_2 ; WAIT FOR SEEK COMPLETE AND READY
1059 0553 ; ; SET INITIAL DELAY BEFORE TEST
1060 0553 51 COMMAND1:
1061 0554 E8 044F R CX ; SAVE LOOP COUNT
1062 0557 59 CALL TST_RDY ; CHECK DRIVE READY
1063 0558 74 0B JC CX ; DRIVE IS READY
1064 055A 80 3E 0074 R 80 CMP *DISK_STATUS1,TIME_OUT ; TST_RDY TIMED OUT--GIVE UP
1065 055F 74 48 JZ CMD_TIMEOUT
1066 0561 E2 F0 LOOP COMMAND1 ; KEEP TRYING FOR A WHILE
1067 0563 EB 49 JMP SHORT_COMMAND4 ; ITS NOT GOING TO GET READY
1068 0565 COMMAND2:
1069 0565 5B POP BX
1070 0566 57 PUSH D1
1071 0567 C6 06 008E R 00 MOV *HF_INT_FLAG,0 ; RESET INTERRUPT FLAG
1072 056C FA CLI ; INHIBIT INTERRUPTS WHILE CHANGING MASK
1073 056D E4 A1 IN AL,INTB01 ; TURN ON SECOND INTERRUPT CHIP
1074 056F 24 BF AND AL,0BFH
1075 0571 EB 00 JMP $+2
1076 0573 E6 A1 OUT INTB01,AL
1077 0575 E4 21 IN AL,INTA01 ; LET INTERRUPTS PASS THRU TO
1078 0577 24 FB AND AL,0FBH ; SECOND CHIP
1079 0579 EB 00 JMP $+2
1080 057B E6 21 OUT INTA01,AL
1081 057D FB STI
1082 057E 33 FF XOR D1,D1 ; INDEX THE COMMAND TABLE
1083 0580 BA 01F1 MOV DX,HF_PORT+1 ; DISK ADDRESS
1084 0583 F6 06 0076 R C0 TEST *CONTROL_BYTE,0C0H ; CHECK FOR RETRY SUPPRESSION
1085 0588 74 11 JZ COMMAND3
1086 058A 8A 46 FE MOV AL,*CMD_BLOCK+6 ; YES--GET OPERATION CODE
1087 058D 24 F0 AND AL,0F0H ; GET RID OF MODIFIERS
1088 058F 3C 20 CMP AL,20H ; 20H-40H IS READ, WRITE, VERIFY
1089 0591 72 08 JB COMMAND3
1090 0593 3C 40 CMP AL,40H
1091 0595 77 04 JA COMMAND3
1092 0597 80 4E FE 01 OR *CMD_BLOCK+6,NO_RETRIES ; VALID OPERATION FOR RETRY SUPPRESS
1093 059B COMMAND3:
1094 059B 8A 43 F8 MOV AL,[*CMD_BLOCK+D1] ; GET THE COMMAND STRING BYTE
1095 059E EE OUT DX,AL ; GIVE IT TO CONTROLLER
1096 059F 47 INC D1 ; NEXT BYTE IN COMMAND BLOCK
1097 05A0 42 INC DX ; NEXT DISK ADAPTER REGISTER
1098 05A1 81 FA 01F8 CMP DX,HF_PORT+8 ; ALL DONE?
1099 05A5 75 F4 JNZ COMMAND3 ; NO--GO DO NEXT ONE
1100 05A7 5F POP D1
1101 05A8 C3 RET ; ZERO FLAG IS SET
1102 05A9
1103 05A9 C6 06 0074 R 20 CMD_TIMEOUT: MOV *DISK_STATUS1,BAD_CNTRL
1104 05AE
1105 05AE 5B POP BX
1106 05AF 80 3E 0074 R 00 CMP *DISK_STATUS1,0 ; SET CONDITION CODE FOR CALLER
1107 05B4 C3 RET
1108 05B5 COMMAND ENDP
    
```

```

1109 PAGE
1110 |-----|
1111 | WAIT FOR INTERRUPT |
1112 |-----|
1113 05B5 WAIT PROC NEAR
1114 05B5 FB STI ; MAKE SURE INTERRUPTS ARE ON
1115 05B6 2B C9 SUB ; SET INITIAL DELAY BEFORE TEST
1116 05B8 FB CLC
1117 05B9 88 9000 MOV AX,9000H ; DEVICE WAIT INTERRUPT
1118 05BC CD 15 INT 15H
1119 05BE 72 0F JC ; DEVICE TIMED OUT
1120
1121 05C0 B3 25 MOV BL,DELAY_1 ; SET DELAY COUNT
1122
1123 |-----|
1124 | WAIT LOOP |
1125 05C2 F6 06 008E R 80 WT1: TEST #HF_INT_FLAG,80H ; TEST FOR INTERRUPT
1126 05C7 E1 F9 LOOPZ WT1 ;
1127 05C9 75 0B JNZ WT3 ; INTERRUPT--LETS GO
1128 05CB FE CB DEC BL
1129 05CD 75 F3 JNZ WT1 ; KEEP TRYING FOR A WHILE
1130
1131 05CF C6 06 0074 R 80 WT2: MOV #DISK_STATUS1,TIME_OUT ; REPORT TIME OUT ERROR
1132 05D4 EB 0A JMP SHORT WT4
1133 05D6 C6 06 0074 R 80 WT3: MOV #DISK_STATUS1,0
1134 05DB C6 06 008E R 80 MOV #HF_INT_FLAG,0
1135 05E0 80 3E 0074 R 80 WT4: CMP #DISK_STATUS1,0 ; SET CONDITION CODE FOR CALLER
1136 05E5 C3 RET
1137 05E6 WAIT ENDP
1138
1139 |-----|
1140 | WAIT FOR CONTROLLER NOT BUSY |
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
1146 05EA 2B C9 SUB ; SET INITIAL DELAY BEFORE TEST
1147 05EC BA 01F7 MOV DX,HF_PORT+7
1148 05EF EC IN AL,DX ; CHECK STATUS
1149 05F0 A8 80 TEST AL,ST_BUSY
1150 05F2 E0 FB LOOPNZ NB1
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 NB2: MOV #DISK_STATUS1,TIME_OUT ; REPORT TIME OUT ERROR
1156 05FF EB 05 JMP SHORT NB3
1157 0601 C6 06 0074 R 80 NB3: MOV #DISK_STATUS1,0
1158 0606 5B POP BX
1159 0607 80 3E 0074 R 80 CMP #DISK_STATUS1,0 ; SET CONDITION CODE FOR CALLER
1160 060C C3 RET
1161 060D NOT_BUSY ENDP
1162
1163 |-----|
1164 | WAIT FOR DATA REQUEST |
1165 |-----|
1166 060D WAIT_DRQ PROC NEAR
1167 060D B9 0100 MOV CX,DELAY_3
1168 0610 BA 01F7 MOV DX,HF_PORT+7
1169 0613 EC IN AL,DX ; GET STATUS
1170 0614 A8 0B TEST AL,ST_DRQ ; WAIT FOR DRQ
1171 0616 75 09 JNZ WQ_OK
1172 0618 E2 F9 LOOP WQ_1 ; KEEP TRYING FOR A SHORT WHILE
1173 061A C6 06 0074 R 80 MOV #DISK_STATUS1,TIME_OUT ; ERROR
1174 061F F9 JMP SHORT WQ_1
1175 0620 C3 RET
1176 0621 FB CLC
1177 0622 C3 RET
1178 0623 WAIT_DRQ ENDP
1179
1180 |-----|
1181 | CHECK FIXED DISK STATUS |
1182 0623 CHECK_STATUS PROC NEAR
1183 0623 E8 0635 R CALL ; CHECK THE STATUS BYTE
1184 0626 75 07 JNZ CHECK_S1 ; AN ERROR WAS FOUND
1185 0628 A8 01 TEST AL,ST_ERROR ; WERE THERE ANY OTHER ERRORS
1186 062A 74 03 JZ CHECK_S1 ; NO ERROR REPORTED
1187 062C E8 0669 R CALL ; ERROR REPORTED
1188 062F CHECK_S1:
1189 062F 80 3E 0074 R 80 CMP #DISK_STATUS1,0 ; SET STATUS FOR CALLER
1190 0634 C3 RET
1191 0635 CHECK_STATUS ENDP
1192
1193 |-----|
1194 | CHECK FIXED DISK STATUS BYTE |
1195 0635 CHECK_ST PROC NEAR
1196 0635 BA 01F7 MOV DX,HF_PORT+7 ; GET THE STATUS
1197 0638 E8 0100 CALL IN AL,DX
1198 0639 A2 008C R MOV #HF_STATUS,AL
1199 063C B4 00 MOV AH,0
1200 063E A8 80 TEST AL,ST_BUSY ; IF STILL BUSY
1201 0640 75 1A JNZ CKST_EXIT ; REPORT OK
1202 0642 B4 CC MOV AH,WRITE_FAULT
1203 0644 A8 20 TEST AL,ST_WRT_FLT ; 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 B4 40 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_CORCTD ; CHECK FOR CORRECTED ECC
1213 0658 75 02 JNZ CKST_EXIT
1214 065A B4 00 MOV AH,0
1215 065C CKST_EXIT:
1216 065C 88 26 0074 R MOV #DISK_STATUS1,AH ; SET ERROR FLAG
1217 0660 80 FC 11 CMP AH,DATA_CORRECTED ; KEEP GOING WITH DATA CORRECTED
1218 0663 74 03 JZ CKST_EXT
1219 0665 80 FC 00 CMP AH,0
1220 0668 CKST_EXT:
1221 0668 C3 RET
1222 0669 CHECK_ST ENDP
    
```

```

1223 PAGE
1224 ;
1225 ; CHECK FIXED DISK ERROR REGISTER ;
1226 ;-----
1227 0669 CHECK_ER PROC NEAR
1228 0669 BA 01F1 MOV DX,HF_PORT+1 ; GET THE ERROR REGISTER
1229 066C EC 01 IN AL,DX
1230 066D A2 008D R MOV #HF_ERROR,AL
1231 0670 53 PUSH BX
1232 0671 B9 0008 MOV CX,8 ; TEST ALL 8 BITS
1233 0674 D0 E0 SHL AL,1 ; MOVE NEXT ERROR BIT TO CARRY
1234 0676 72 02 JC K2 ; FOUND THE ERROR
1235 0678 E2 FA LOOP CK1 ; KEEP TRYING
1236 067A BB 068B R CK2: MOV BX,OFFSET_ERR_TBL ; COMPUTE ADDRESS OF
1237 067D 03 D9 ADD BX,DX ; ERROR CODE
1238 067F 2E: BA 27 MOV AH,BYTE PTR CS:[BX] ; GET ERROR CODE
1239 0682 88 26 0074 R CKEX: MOV #DISK_STATUS1,AH ; SAVE ERROR CODE
1240 0686 5B POP BX
1241 0687 80 FC 00 CMP AH,0
1242 068A C3 RET
1243 068B E0 ERR_TBL DB NO_ERR
1244 068C 02 40 01 BB DB BAD_ADDR_MARK,BAD_SEEK,BAD_CMD,UNDEF_ERR
1245 0690 04 BB 10 0A DB REC'D_NOT_FND,UNDEF_ERR,BAD_ECC,BAD_SECTOR
1246 0694 CHECK_ER ENDP
1247 ;
1248 ;-----
1249 ; CHECK DMA ;
1250 ; -CHECK ES:BX AND # SECTORS TO MAKE SURE THAT IT WILL ;
1251 ; FIT WITHOUT SEGMENT OVERFLOW. ;
1252 ; -ES:BX HAS BEEN REVISED TO THE FORMAT S55S:000X ;
1253 ; -OK IF # SECTORS < 80H (7FH IF LONG READ OR WRITE) ;
1254 ; -OK IF # SECTORS = 80H (7FH) AND BX <= 00H (04H) ;
1255 ; -ERROR OTHERWISE ;
1256 ;-----
1257 0694 CHECK_DMA PROC NEAR
1258 0694 50 PUSH AX ; SAVE REGISTERS
1259 0695 B8 8000 MOV AX,8000H ; AH = MAX # SECTORS AL = MAX OFFSET
1260 0698 F6 40 FE 02 TEST #CM_BLOCK+6,ECC_MODE
1261 069C 74 03 JZ CKD1
1262 069E B8 7F04 MOV AX,7F04H ; ECC IS 4 MORE BYTES
1263 06A1 3A 66 F9 CMP AH,#CM_BLOCK+1 ; NUMBER OF SECTORS
1264 06A4 77 06 JA CKDK ; IT WILL FIT
1265 06A6 72 07 JB CKDERR ; TOO MANY
1266 06A8 3A C3 CMP AL,BL ; CHECK OFFSET ON MAX SECTORS
1267 06AA 72 03 JB CKDERR ; ERROR
1268 06AC F8 CLC ; CLEAR CARRY
1269 06AD 58 POP AX
1270 06AE C3 RET ; NORMAL RETURN
1271 06AF F9 CKDERR: STC ; INDICATE ERROR
1272 06B0 C6 06 0074 R 09 MOV #DISK_STATUS1,DMA_BOUNDARY
1273 06B5 58 POP AX
1274 06B6 C3 RET
1275 06B7 CHECK_DMA ENDP
1276 ;
1277 ;-----
1278 ; SET UP ES:BX-> DISK PARMS ;
1279 ;-----
1280 06B7 GET_VEC PROC NEAR
1281 06B7 2B C0 SUB AX,AX ; GET DISK PARAMETER ADDRESS
1282 06B9 BE C0 MOV ES,AX
1283 06BB F6 C2 01 TEST DL,1
1284 06BE 74 07 JZ GV_0
1285 06C0 26: C4 IE 0118 R LES BX,#HF1_TBL_VEC ; ES:BX -> DRIVE PARAMETERS
1286 06C5 EB 05 JMP SHORT GV_EXT1
1287 06C7 GV_0: LES BX,#HF_TBL_VEC ; ES:BX -> DRIVE PARAMETERS
1288 06C7 LES BX,#HF_TBL_VEC ; ES:BX -> DRIVE PARAMETERS
1289 06C7 26: C4 IE 0104 R
1290 06CC GV_EXIT: RET
1291 06CC C3 GET_VEC ENDP
1292 06CD ;
1293 ;-----
1294 ;--- HARDWARE INT 76H --- ( IRQ LEVEL 14 ) -----
1295 ;
1296 ; FIXED DISK INTERRUPT ROUTINE ;
1297 ;-----
1298 ;
1299 ;
1300 06CD HD_INT PROC NEAR
1301 06CD 50 PUSH AX
1302 06CE 1E PUSH DS
1303 06CF E8 0000 E CALL DDS
1304 06D2 C6 06 008E R FF MOV #HF_INT_FLAG,OFFH ; ALL DONE
1305 06D7 B0 20 MOV AL,E01 ; NON-SPECIFIC END OF INTERRUPT
1306 06D9 E6 A0 OUT INTB00,AL ; FOR CONTROLLER #2
1307 06DB EB 00 JMP $+2 ; WAIT
1308 06DD E6 20 OUT INTA00,AL ; FOR CONTROLLER #1
1309 06DF 1F POP DS
1310 06E0 FB STI ; RE-ENABLE INTERRUPTS
1311 06E1 B8 9100 MOV AX,9100H ; DEVICE POST
1312 06E4 CD 15 INT 15H ; INTERRUPT
1313 06E6 5B POP AX
1314 06E7 CF IRET ; RETURN FROM INTERRUPT
1315 ;
1316 06EB HD_INT ENDP
1317 ;
1318 06EB 30 36 2F 31 30 2F DB '06/10/85' ; RELEASE MARKER
1319 06F0 38 35
1320 06F0 CODE ENDS
1321 END

```

```

1 PAGE 118,121
2 TITLE KYBD ----- 06/10/85 KEYBOARD BIOS
3 LIST
4 CODE SEGMENT BYTE PUBLIC
5
6 PUBLIC K16
7 PUBLIC KEYBOARD_IO_1
8 PUBLIC KB_INT_1
9 PUBLIC SND_DATA
10
11 EXTRN BEEP:NEAR
12 EXTRN DDS:NEAR
13 EXTRN START_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 K6LIABS
22 EXTRN K7:BYTE
23 EXTRN K8:BYTE
24 EXTRN K9:BYTE
25
26 ;----- INT 16 H -----
27 ; KEYBOARD I/O
28 ; THESE ROUTINES PROVIDE READ KEYBOARD SUPPORT
29 ; INPUT
30 ; (AH)= 00H READ THE NEXT ASCII CHARACTER ENTERED FROM THE KEYBOARD,
31 ; RETURN THE RESULT IN (AL), SCAN CODE IN (AH).
32 ; (AH)= 01H SET THE ZERO FLAG TO INDICATE IF AN ASCII CHARACTER IS
33 ; AVAILABLE TO BE READ FROM THE KEYBOARD BUFFER.
34 ; (ZF)= 1 -- NO CODE AVAILABLE
35 ; (ZF)= 0 -- CODE IS AVAILABLE (AX)= CHARACTER
36 ; IF (ZF)= 0, THE NEXT CHARACTER IN THE BUFFER TO BE READ IS
37 ; IN (AX), AND THE ENTRY REMAINS IN THE BUFFER.
38 ; (AH)= 02H RETURN THE CURRENT SHIFT STATUS IN (AL) REGISTER
39 ; THE BIT SETTINGS FOR THIS CODE ARE INDICATED IN THE
40 ; THE EQUATES FOR *KB_FLAG
41 ; OUTPUT
42 ; AS NOTED ABOVE, ONLY (AX) AND FLAGS CHANGED
43 ; ALL REGISTERS RETAINED
44 ;-----
45 ASSUME CS:CODE,DS:DATA
46
47 0000 KEYBOARD_IO_1 PROC FAR ;>>> ENTRY POINT FOR ORG 0E82EH
48 0000 FB STI ; INTERRUPTS BACK ON
49 0001 IE PUSH DS ; SAVE CURRENT DS
50 0002 53 PUSH BX ; SAVE BX TEMPORARILY
51 0003 E8 0000 E CALL DDS ; ESTABLISH POINTER TO DATA REGION
52 0006 0A E4 OR AH,AH ; CHECK FOR (AH)= 00H
53 0008 74 0B JZ K1B ; ASCII READ
54 000A FE CC DEC AH ; CHECK FOR (AH)= 01H
55 000C 74 45 JZ K2 ; ASCII STATUS
56 000E FE CC DEC AH ; CHECK FOR (AH)= 02H
57 0010 74 67 JZ K3 ; SHIFT STATUS
58 0012 5B POP BX ; RECOVER REGISTER
59 0013 1F POP DS
60 0014 CF IRET ; INVALID COMMAND EXIT
61
62 ;----- READ THE KEY TO FIGURE OUT WHAT TO DO
63
64 0015 8B IE 001A R K1B: MOV BX,*BUFFER_HEAD ; GET POINTER TO HEAD OF BUFFER
65 0019 3B 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 B8 9002 MOV AX,09002H ; MOVE IN WAIT CODE & TYPE
69 0022 CD 15 INT 15H ; PERFORM OTHER FUNCTION
70 0024
71 0024 FB STI ; ASCII READ
72 0025 90 NOP ; INTERRUPTS BACK ON DURING LOOP
73 0026 FA CLI ; ALLOW AN INTERRUPT TO OCCUR
74 0027 8B IE 001A R K1C: MOV BX,*BUFFER_HEAD ; INTERRUPTS BACK OFF
75 0028 3B IE 001C R CMP BX,*BUFFER_TAIL ; TEST END OF BUFFER
76 002F 53 PUSH BX ; SAVE ADDRESS
77 0030 9C PUSHF ; SAVE FLAG
78 0031 E8 0587 R CALL MAKE_LED ; GO GET MODE INDICATOR DATA BYTE
79 0034 8A IE 0097 R MOV BL,*KB_FLAG_2 ; GET PREVIOUS BITS
80 0038 32 D8 XOR BL,AL ; SEE IF ANY DIFFERENT
81 003A 80 E3 07 AND BL,KB_LEDS ; ISOLATE INDICATOR BITS
82 003D 74 04 JZ K1A ; IF NO CHANGE BYPASS UPDATE
83
84 003F E8 0549 R K1A: CALL SND_LED1 ; GO TURN ON MODE INDICATORS
85 0042 FA CLI ; DISABLE INTERRUPTS
86 0043 9D POPF ; RESTORE FLAGS
87 0044 5B POP BX ; RESTORE ADDRESS
88 0045 74 DD JZ K1 ; LOOP UNTIL SOMETHING IN BUFFER
89
90 0047 8B 07 MOV AX,[BX] ; GET SCAN CODE AND ASCII CODE
91 0049 E8 007F R CALL K4 ; MOVE POINTER TO NEXT POSITION
92 004C 89 IE 001A R MOV *BUFFER_HEAD,BX ; STORE VALUE IN VARIABLE
93
94 0050 5B POP BX ; RECOVER REGISTER
95 0051 1F POP DS ; RECOVER SEGMENT
96 0052 CF IRET ; RETURN TO CALLER
97
98 ;----- ASCII STATUS
99
100 0053 K2: CLI ; INTERRUPTS OFF
101 0053 FA CLI ; DISABLE INTERRUPTS
102 0054 8B IE 001A R MOV BX,*BUFFER_HEAD ; GET HEAD POINTER
103 0058 3B IE 001C R CMP BX,*BUFFER_TAIL ; IF EQUAL (Z=1) THEN NOTHING THERE
104 005C 8B 07 MOV AX,[BX]
105 005E 9C PUSHF ; SAVE FLAGS
106 005F 50 PUSH AX ; SAVE CODE
107 0060 E8 0587 R CALL MAKE_LED ; GO GET MODE INDICATOR DATA BYTE
108 0063 8A IE 0097 R MOV BL,*KB_FLAG_2 ; GET PREVIOUS BITS
109 0067 32 D8 XOR BL,AL ; SEE IF ANY DIFFERENT
110 0069 80 E3 07 AND BL,KB_LEDS ; ISOLATE INDICATOR BITS
111 006C 74 03 JZ SK2 ; IF NO CHANGE BYPASS UPDATE
112
113 006E E8 0549 R SK2: CALL SND_LED1 ; GO TURN ON MODE INDICATORS
114 0071 5B POP AX ; RESTORE CODE

```

SECTION 5

```

115 0072 9D          POPF                ; RESTORE FLAGS
116 0073 FB          STI                 ; INTERRUPTS BACK ON
117 0074 5B          POP                ; RECOVER REGISTER
118 0075 1F          POP                ; RECOVER SEGMENT
119 0076 CA 0002     RET                ; THROW AWAY FLAGS
120
121
122
123 0079             ;----- SHIFT STATUS
124 0079 A0 0017 R   K3:      MOV     AL,0KB_FLAG      ; GET THE SHIFT STATUS FLAGS
125 007C 5B          POP                ; RECOVER REGISTER
126 007D 1F          POP                ; RECOVER REGISTERS
127 007E CF          IRET               ; RETURN TO CALLER
128 007F             KEYBOARD_10_1  ENDP
129
130
131
132 007F             ;----- INCREMENT A BUFFER POINTER
133 007F 43          K4:      PROC    NEAR
134 0080 43          INC     BX                ; MOVE TO NEXT WORD IN LIST
135 0081 3B 1E 0082 R CMP     BX,0BUFFER_END   ; AT END OF BUFFER?
136 0085 75 04          JNE    K5                 ; NO, CONTINUE
137 0087 8B 1E 0080 R MOV     BX,0BUFFER_START ; YES, RESET TO BUFFER BEGINNING
138 008B             K5:
139 008B C3          RET
140 008C             K4      ENDP
141
142
143
144             ;--- HARDWARE INT 09H -- ( IRQ LEVEL 1 ) -----
145             ;
146             ; KEYBOARD INTERRUPT ROUTINE
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                ; FORWARD DIRECTION
160 0097 E8 0000 E   CALL    DDS           ; SET UP ADDRESSING
161
162
163
164 009A B0 AD          MOV     AL,DIS_KBD      ; DISABLE THE KEYBOARD COMMAND
165 009C E8 0595 R   CALL    SHIP_IT        ; EXECUTE DISABLE
166 009F FA          CLI                 ; DISABLE INTERRUPTS
167 00A0 2B C9          SUB     CX,CX          ; SET MAXIMUM TIMEOUT
168 00A2             KB_INT_01:
169 00A2 E4 64          IN     AL,STATUS_PORT  ; READ ADAPTER STATUS
170 00A4 A8 02          TEST   AL,INPT_BUF_FULL ; CHECK INPUT BUFFER FULL STATUS BIT
171 00A6 E0 FA          LOOPNZ KB_INT_01      ; WAIT FOR COMMAND TO BE ACCEPTED
172
173
174
175 00AB E4 60          ;----- READ CHARACTER FROM KEYBOARD INTERFACE
176             IN     AL,PORT_A          ; READ IN THE CHARACTER
177
178
179 00AA B4 4F          ;----- SYSTEM HOOK INT 15H - FUNCTION 4FH (ON HARDWARE INTERRUPT LEVEL 9H)
180 00AC F9          MOV     AH,04FH        ; SYSTEM INTERCEPT - KEY CODE FUNCTION
181 00AD CD 15          STC                 ; SET CY=1 (IN CASE OF IRET)
182             INT     15H                ; CASSETTE CALL (AL)=KEY SCAN CODE
183 00AF 72 03          JC     KB_INT_02      ; RETURNS CY=1 FOR INVALID FUNCTION
184             JC     KB_INT_02      ; CONTINUE IF CARRY FLAG SET ((AL)=CODE)
185 00B1 E9 02EE R   JMP     K26           ; EXIT IF SYSTEM HANDLED SCAN CODE
186             ; EXIT HANDLES HARDWARE EOI AND ENABLE
187
188
189
190             ;----- CHECK FOR A RESEND COMMAND TO KEYBOARD
191 00B4             KB_INT_02:
192 00B4 FB          STI                 ; (AL)=SCAN CODE
193 00B5 3C FE          CMP     BL,KB_RESEND   ; ENABLE INTERRUPTS AGAIN
194 00B7 74 0D          JE     KB_INT_4        ; IS THE INPUT A RESEND
195             ; GO IF RESEND
196
197             ;----- CHECK FOR RESPONSE TO A COMMAND TO KEYBOARD
198 00B9 3C FA          CMP     AL,KB_ACK      ; IS THE INPUT AN ACKNOWLEDGE
199 00BB 75 12          JNZ    KB_INT_2        ; GO IF NOT
200
201
202             ;----- A COMMAND TO THE KEYBOARD WAS ISSUED
203 00BD FA          CLD                 ; DISABLE INTERRUPTS
204 00BE 80 0E 0097 R OR     0KB_FLAG_2,KB_FA ; INDICATE ACK RECEIVED
205 00C3 E9 02EE R   JMP     K26           ; RETURN IF NOT (ACK RETURNED FOR DATA)
206
207
208             ;----- RESEND THE LAST BYTE
209 00C6             KB_INT_4:
210 00C6 FA          CLD                 ; DISABLE INTERRUPTS
211 00C7 80 0E 0097 R OR     0KB_FLAG_2,KB_FE ; INDICATE RESEND RECEIVED
212 00CC E9 02EE R   JMP     K26           ; RETURN IF NOT (ACK RETURNED FOR DATA)
213
214
215             KB_INT_2:
216
217             ;----- UPDATE MODE INDICATORS IF CHANGE IN STATE
218 00CF 50          PUSH    AX                ; SAVE DATA IN
219 00D0 E8 0587 R   CALL    MAKE_LED       ; GO GET MODE INDICATOR DATA BYTE
220 00D3 8A 1E 0097 R MOV     BL,0KB_FLAG_2   ; GET PREVIOUS BITS
221 00D7 32 D8          XOR    BL,AL            ; SEE IF ANY DIFFERENT
222 00D9 80 E3 07          AND    BL,KB_LEDS      ; ISOLATE INDICATOR BITS
223 00DC 74 03          JZ     UP0              ; IF NO CHANGE BYPASS UPDATE
224 00DE E8 0536 R   CALL    SND_LED        ; GO TURN ON MODE INDICATORS
225 00E1 58          POP     AX                ; RESTORE DATA IN
226 00E2 8A E0          MOV     AH,AL           ; SAVE SCAN CODE IN AH ALSO
227
228
229             ;----- TEST FOR OVERRUN SCAN CODE FROM KEYBOARD

```

```

229                                CMP     AL,KB_OVER_RUN           ; IS THIS AN OVERRUN CHAR
230                                JNZ     K16                       ; NO, TEST FOR SHIFT KEY
231                                JMP     K62                      ; BUFFER_FULL_BEEP
232    00E8 E9 04EB R
233
234                                ;----- THIS CODE CONTAINS THE KBX SUPPORT FOR INT 09H
235
236                                ;
237                                ; EQUATES
238    237 = 00D9                    F11_M EQU 217           ; FUNC 11 MAKE
239    238 = 00D7                    F11_B EQU 215           ; FUNC 11 BREAK
240    239 = 00DA                    F12_M EQU 218           ; FUNC 12 MAKE
241    240 = 00D8                    F12_B EQU 216           ; FUNC 12 BREAK
242    241 = 0056                    K102_M EQU 86          ; KEY 102 MAKE
243    242 = 00D6                    K102_B EQU 214          ; KEY 102 BREAK
244
245    244 = 0052                    INS_M EQU 82          ; INSERT KEY MAKE
246    245 = 0053                    DEL_M EQU 83          ; DELETE KEY MAKE
247    246 = 004B                    LEFT_M EQU 75         ; CURSOR LEFT MAKE
248    247 = 004D                    RIGHT_M EQU 77        ; CURSOR RIGHT MAKE
249    248 = 0048                    UP_M EQU 72           ; CURSOR UP MAKE
250    249 = 0050                    DN_M EQU 80           ; CURSOR DOWN MAKE
251    250 = 0049                    PGDP_M EQU 73        ; PG UP MAKE
252    251 = 0051                    PGDN_M EQU 81        ; PG DN MAKE
253    252 = 0047                    HOME_M EQU 71        ; HOME MAKE
254    253 = 004F                    END_M EQU 79         ; END MAKE
255
256    255 = 0085                    FUNC11 EQU 133       ; FUNCTION 11 KEY
257    256 = 00E0                    HC EQU 224           ; HIDDEN CODE
258
259                                ;----- TABLE OF KEYPAD CURSOR & CONTROL KEYS
260                                ;
261    260 00EB 48 50 52 53 4B 4D    K_TAB1 DB UP_M,DN_M,INS_M,DEL_M,LEFT_M,RIGHT_M
262    261 00F1 49 51 47 4F          DB PGDP_M,PGDN_M,HOME_M,END_M
263    262 = 000A                    L_TAB1 EQU          $-K_TAB1
264
265    264 00F5                    K16:
266    265 00F5 24 7F              AND     AL,07FH      ; REMOVE BREAK BIT
267    266 00F1 0E                PUSH   CS             ;
268    267 00F8 07                POP     ES            ; ESTABLISH ADDRESS OF TABLES
269
270    269 00F9 F6 06 0096 R C0     TEST   %KB_FLAG_3,RD_ID<LC_AB ; ARE WE DOING A READ ID?
271    270 00FE 74 33              JZ     NOT_ID         ; CONTINUE IF NOT
272    271 0100 79 11              JNS   TST_ID_2       ; IS THE RD_ID FLAG ON?
273    272 0102 80 FC AB          CMP    AH,TD_1        ; IS THIS THE 1ST ID CHARACTER?
274    273 0105 75 05              JNE   RST_RD_ID      ;
275    274 0107 80 0E 0096 R 40    OR     %KB_FLAG_3,LC_AB ; INDICATE 1ST ID WAS OK
276    275 010C                    RST_RD_ID:
277    276 010C 80 26 0096 R 7F    AND    %KB_FLAG_3,NOT_RD_ID ; RESET THE READ ID FLAG
278    277 0111 EB 4B              JMP    SHORT_DO_EXT
279
280    279 0113                    TST_ID_2:
281    280 0113 80 26 0096 R BF    AND    %KB_FLAG_3,NOT_LC_AB ; RESET FLAG
282    281 0118 80 FC 41          CMP    AH,TD_2        ; IS THIS THE 2ND ID CHARACTER?
283    282 011B 75 41              JNE   DO_EXT         ; LEAVE IF NOT
284
285                                ;----- A READ ID SAID THAT IT WAS KBX
286
287    286 011D 80 0E 0096 R 01    OR     %KB_FLAG_3,KBX ; INDICATE KBX WAS FOUND
288    287 0122 F6 06 0096 R 20    TEST  %KB_FLAG_3,SET_NUM_LK ; SHOULD WE SET NUM LOCK?
289    288 0127 74 35              JZ     DO_EXT         ; EXIT IF NOT
290    289 0129 80 0E 0017 R 20    OR     %KB_FLAG_NUM_STATE ; FORCE NUM LOCK ON
291    290 012E EB 0536 R          CALL   SND_LED        ; GO SET THE NUM LOCK INDICATOR
292    291 0131 EB 70              JMP    SHORT_EXIT
293
294    293 0133 F6 06 0096 R 02    TEST  %KB_FLAG_3,LC_HC ; WAS THE LAST CHARACTER A HIDDEN CODE
295    294 0138 74 5F              JZ     NOT_LC_HC     ; JUMP IF NOT
296
297                                ;----- THE LAST CHARACTER WAS A HIDDEN CODE
298
299    298 013A 80 26 0096 R FD    AND    %KB_FLAG_3,NOT_LC_HC ; RESET LAST CHAR HIDDEN CODE FLAG
300    299 013F 3C 52              CMP    AL,TNS_M       ; WAS IT THE INSERT KEY?
301    300 0141 74 05              JZ     NOT_I         ;
302    301 0143 F6 C4 80          TEST  %AH,80H        ; IS THIS A BREAK CODE
303    302 0146 75 5B              JNZ   EXIT           ; IGNORE BREAK ON REST OF THESE KEYS
304
305    304 0148 BF 00EB R          MOV    DI,OFFSET K_TAB1 ; TEST FOR ONE OF THE KEYPAD CURSOR FUNC
306    305 014B B9 000A          MOV    CX,L_TAB1      ;
307    306 014E F2/ AE             REPNE SCASB          ; SCAN FOR THE KEY
308    307 0150 75 54              JNE   NOT_CUR        ; GO ON IF NOT FOUND
309    308 0152 F6 06 0018 R 08    TEST  %KB_FLAG_1,HOLD_STATE ; ARE WE IN HOLD STATE?
310    309 0157 74 07              JZ     N_HLD         ;
311    310 0159 80 26 0018 R F7    AND    %KB_FLAG_1,NOT_HOLD_STATE ; EXIT HOLD STATE
312
313    312 015E EB 43              JMP    SHORT_EXIT    ; IGNORE THIS KEY
314
315    314 0160 F6 06 0017 R 08    TEST  %KB_FLAG,ALT_SHIFT ; IS ALT DOWN?
316    315 0165 74 0E              JZ     NOT_ALT       ;
317    316 0167 F6 06 0017 R 04    TEST  %KB_FLAG,CTL_SHIFT ; HOW ABOUT CTRL?
318    317 016C 74 35              JZ     EXIT          ; IGNORE ALL IF ONLY ALT DOWN
319    318 016E 3C 53              CMP    AL,DEL_M       ; WAS IT THE DELETE KEY?
320    319 0170 75 31              JNE   EXIT           ; IGNORE IF NOT
321    320 0172 E9 030D R          JMP    K29           ; GO DO THE CTL, ALT, DEL RESET
322
323    322 0175                    NOT_ALT:
324    323 0175 F6 06 0017 R 04    TEST  %KB_FLAG,CTL_SHIFT ; IS CTL DOWN?
325    324 017A 75 15              JNZ   CTL_ON         ; SPECIAL CASE IF SO
326    325 017C 3C 52              CMP    AL,TNS_M       ; IS THIS THE INSERT KEY?
327    326 017E 75 0E              JNE   N_INS         ;
328
329                                ;----- SPECIAL HANDLING FOR INSERT KEY
330
331    330 0180 8A C4              MOV    AL,AH          ; RECOVER SCAN CODE
332    331 0182 84 80              MOV    AH,INS_SHIFT  ; AH = MASK FOR INSERT
333    332 0184 A8 80              TEST  AL,80H        ; WAS THIS A BREAK CODE?
334    333 0186 75 03              JZ     B_C           ;
335    334 0188 E9 028F R          JMP    R22           ; GO HANDLE INSERT SHIFT
336
337    335 018B                    B_C:
338    336 018B E9 02D2 R          JMP    K24           ; HANDLE BREAK
339
340    337 018E                    N_INS:
341    338 018E E9 0453 R          JMP    K49           ; HANDLE & IGNORE NUMLOCK
342
343    339 0191                    CTL_ON:
344    340 0191 80 F9 05          CMP    CL,5          ; WAS IT INS, DEL, UP OR DOWN?
345    341 0194 77 0D              JA     EXIT          ; IGNORE IF SO
346    342 0196 E9 0401 R          JMP    K42           ; GO HANDLE CTRL CASE

```

SECTION 5

```

343
344 0199          NOT_LC_HC:          ; LAST CHARACTER WAS NOT A HIDDEN CODE
345 0199 80 FC E0      CMP          AH,HC          ; IS THIS CHARACTER A HIDDEN CODE?
346 019C 75 08          JNE          NOT_CUR
347 019E 80 0E 0096 R 03 OR          0KB_FLAG_3,LC_HC+KBX
348 01A3          EXIT:          ; SET LAST CHAR WAS A HIDDEN CODE & KBX
349 01A3 E9 02EE R      JMP          K26          ; THROW AWAY THIS CODE
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 01AD EB 07          JMP          SHORT DO_FN
356 01AF          T_F12:          ;
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 01B6          DO_FN:          ;
361 01B6 80 FC D7      CMP          AH,F11_B      ; IS THIS A BREAK CODE
362 01B9 74 E8          JE          EXIT          ; IGNORE BREAK CODES
363 01BB 80 FC D8      CMP          AH,F12_B      ; IS THIS A BREAK CODE
364 01BE 74 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 66          JZ          N_HLD1        ;
367 01C7 80 26 0018 R F7 AND          0KB_FLAG_1,NOT_HOLD_STATE ; EXIT HOLD STATE
368 01CC EB D5          JMP          SHORT EXTT    ; IGNORE THIS KEY
369 01CE          N_HLD1:          ;
370 01CE 8A E1          MOV          AH,CL
371
372 01D0 F6 06 0017 R 08 TEST         0KB_FLAG,ALT_SHIFT ; ARE WE IN ALT
373 01D5 74 05          JZ          T_CTL        ;
374 01D7 80 C4 06      ADD          AH,6         ; CNVT TO ALT FN 11-12
375 01DA EB 16          JMP          SHORT SET_FN
376 01DC          T_CTL:          ;
377 01DC F6 06 0017 R 04 TEST         0KB_FLAG,CTL_SHIFT ; ARE WE IN CTRL
378 01E1 74 05          JZ          T_SHF        ;
379 01E3 80 C4 04      ADD          AH,4         ; CNVT TO CTRL FN 11-12
380 01E6 EB 0A          JMP          SHORT SET_FN
381 01E8          T_SHF:          ;
382 01E8 F6 06 0017 R 03 TEST         0KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT ; IS EITHER SHIFT ON?
383 01ED 74 03          JZ          SET_FN       ;
384 01EF 80 C4 02      ADD          AH,2         ; CNVT TO SHIFT FN 11-12
385 01F1          SET_FN:          ;
386 01F2 2A C0          SUB          AL,AL        ; FORCE PSEUDO SCAN CODE
387 01F4 E9 04BA R      JMP          K61         ; PUT IT INTO BUFFER
388
389
390
391 01F7          I----- TEST FOR SYSTEM KEY
392 01F7 3C 54          T_SYS_KEY:          ;
393 01F9 75 3D          CMP          AL,SYS_KEY   ; IS IT THE SYSTEM KEY?
394                                JNC          K16A        ; CONTINUE IF NOT
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                                ; INTERRUPT-RETURN-NO-E01
405 0210 B0 AE          MOV          AL,ENA_KBD   ; INSURE KEYBOARD IS ENABLED
406 0212 EB 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 021B E9 02F8 R      JMP          K27A        ; END PROCESSING
411 021E          K16B:          ;
412 021E E9 02EE R      JMP          K26         ; IGNORE SYSTEM KEY
413 0221          K16C:          ;
414 0221 80 26 0018 R FB 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                                ; INTERRUPT-RETURN-NO-E01
418 022A B0 AE          MOV          AL,ENA_KBD   ; INSURE KEYBOARD IS ENABLED
419 022C EB 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 02F8 R      JMP          K27A        ; IGNORE SYSTEM KEY
424 0238          K16A:          ;
425 0238 BF 0000 E      MOV          DI,OFFSET K6 ; SHIFT KEY TABLE
426 023B B9 0000 E      MOV          CX,OFFSET K6L ; LENGTH
427 023E F2 AE          REPNE       SCASB        ; LOOK THROUGH THE TABLE FOR A MATCH
428 0240 8A CA          MOV          AL,AH        ; RECOVER SCAN CODE
429 0242 74 03          JE          K17          ; JUMP IF MATCH FOUND
430 0244 E9 02DA R      JMP          K25         ; IF NO MATCH, THEN SHIFT NOT FOUND
431
432
433 0247          I----- SHIFT KEY FOUND
434 0247 81 EF 0001 E      K17:          ;
435 024B 2E1: 8A A5 0000 E MOV          AH,CS:[K7][DI] ; ADJUST PTR TO SCAN CODE MATCH
436 0250 A8 80          TEST         AL,80H      ; GET MASK INTO AH
437 0252 74 02          JZ          K17C        ; TEST FOR BREAK KEY
438 0254 EB 5D          JMP          SHORT K23    ; BREAK SHIFT_FOUND
439                                ; CONTINUE
440
441 0256          I----- DETERMINE SET OR TOGGLE
442 0256 80 FC 10      K17C:          ;
443 0259 73 07          CMP          AH,SCROLL_SHIFT ; IF SCROLL SHIFT OR ABOVE, TOGGLE KEY
444                                JAE          K18
445
446
447 025B 08 26 0017 R      I----- PLAIN SHIFT KEY, SET SHIFT ON
448 025F E9 02EE R      OR          0KB_FLAG,AH   ; TURN ON SHIFT BIT
449                                JMP          K26         ; INTERRUPT_RETURN
450
451 0262          I----- TOGGLED SHIFT KEY, TEST FOR 1ST MAKE OR NOT
452 0262 F6 06 0017 R 04 K18:          ;
453 0265 74 01          TEST         0KB_FLAG,CTL_SHIFT ; SHIFT-TOGGLE
454 0267 75 71          JNZ          K25         ; CHECK CTL_SHIFT STATE
455                                ; JUMP IF CTL STATE
456 0269 3C 52          CMP          AL,INS_KEY   ; CHECK FOR INSERT KEY

```



```

457 026B 75 22          JNZ    K22              ; JUMP IF NOT INSERT KEY
458 026D F6 06 0017 R 0B 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 ; CHECK FOR LEFT/RIGHT SHIFT
464 0280 74 0D          JZ     K22              ; JUMP IF BASE STATE
465
466 0282                ; NUMERIC ZERO, NOT INSERT KEY
467 0282 B8 5230        MOV    AX,5230H         ; PUT OUT AN ASCII ZERO
468 0285 E9 048A R      JMP    K57              ; BUFFER FILL
469 0288                ; MIGHT BE NUMERIC
470 0288 F6 06 0017 R 03 K21:   TEST   *KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT
471 028D 74 F3          JZ     K20              ; JUMP NUMERIC, NOT INSERT
472
473 028F                ; SHIFT TOGGLE KEY HIT; PROCESS IT
474 028F 84 26 0018 R  TEST   AH,*KB_FLAG_I    ; IS KEY ALREADY DEPRESSED
475 0293 74 02          JZ     K22A0            ; GO IF NOT
476 0295 EB 57          JMP    SHORT K26         ; JUMP IF KEY ALREADY DEPRESSED
477 0297
478 0297 08 26 0018 R  DR     *KB_FLAG_I,AH    ; INDICATE THAT THE KEY IS DEPRESSED
479 029B 30 26 0017 R  XOR    *KB_FLAG,AH      ; TOGGLE THE SHIFT STATE
480
481
482 ;----- TOGGLE LED IF CAPS OR NUM KEY DEPRESSED
483 029F F6 C4 70        TEST   AH,CAPS_SHIFT+NUM_SHIFT+SCROLL_SHIFT ; SHIFT TOGGLE?
484 02A2 74 05          JZ     K22B             ; GO IF NOT
485
486 02A4 50              PUSH   AX                ; SAVE SCAN CODE AND SHIFT MASK
487 02A5 E8 0536 R      CALL  SND_LED           ; GO TURN MODE INDICATORS ON
488 02A8 58              POP    AX                ; RESTORE SCAN CODE
489 02A9
490 02A9 3C 52          K22B:  CMP    AL,INS_KEY       ; TEST FOR 1ST MAKE OF INSERT KEY
491 02AB 75 41          JNE    K26              ; JUMP IF NOT INSERT KEY
492 02AD B8 5200        MOV    AX,INS_KEY*H     ; SET SCAN CODE INTO AH, 0 INTO AL
493 02B0 E9 048A R      JMP    K57              ; PUT INTO OUTPUT BUFFER
494
495 ;----- BREAK SHIFT FOUND
496
497 02B3                ; BREAK-SHIFT-FOUND
498 02B3 80 FC 10        K23:   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 26 06 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              ; INTERRUPT_RETURN
504
505 ;----- ALTERNATE SHIFT KEY RELEASED, GET THE VALUE INTO BUFFER
506
507 02C2 A0 0019 R      MOV    AL,*ALT_INPUT    ; SCAN CODE OF 0
508 02C5 B4 00          MOV    AH,0              ; ZERO OUT THE FIELD
509 02C7 88 26 0019 R  MOV    *ALT_INPUT,AH    ; ARE WE IN HOLD STATE
510 02CB 3C 00          CMP    AL,0              ; WAS THE INPUT=0
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                ; BREAK-TOGGLE
515 02D2 F6 D4          K24:   NOT    AH                ; INVERT MASK
516 02D4 26 06 0018 R  AND    *KB_FLAG_I,AH    ; INDICATE NO LONGER DEPRESSED
517 02D8 EB 14          JMP    SHORT K26         ; INTERRUPT_RETURN
518
519 ;----- TEST FOR HOLD STATE
520
521 02DA                ; NO-SHIFT-FOUND
522 02DA 3C 80          K25:   CMP    AL,80H           ; TEST FOR BREAK KEY
523 02DC 73 10          JAE    K26              ; NOTHING FOR BREAK CHARS FROM HERE ON
524 02DE F6 06 0018 R 0B AND    *KB_FLAG_I,HOLD_STATE ; ARE WE IN HOLD STATE
525 02E3 74 1E          TEST   JZ, K28           ; BRANCH AROUND TEST IF NOT
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                ; INTERRUPT_RETURN
531 02EE FA              CLI     INTERRUPTS      ; TURN OFF INTERRUPTS
532 02EF B0 20          MOV    AL,E0I           ; END OF INTERRUPT COMMAND
533 02F1 E6 20          OUT    INTA00,AL        ; SEND COMMAND TO INTERRUPT CONTROL PORT
534 02F3                ; INTERRUPT_RETURN-NO-E0I
535 02F3 B0 AE          K27:   MOV    AL,ENA_KBD       ; INSURE KEYBOARD IS ENABLED
536 02F5 EB 05 95 R    CALL  SHIP_IT          ; EXECUTE ENABLE
537 02F8
538 02F8 FA              K27A:  CLI     INTERRUPTS     ; DISABLE INTERRUPTS
539 02F9 07              POP    ES                ; RESTORE REGISTERS
540 02FA 1F              POP    DS
541 02FB 5F              POP    DI
542 02FC 5E              POP    SI
543 02FD 5A              POP    DX
544 02FE 59              POP    CX
545 02FF 5B              POP    BX
546 0300 58              POP    AX
547 0301 5D              POP    BP
548 0302 CF              IRET                    ; RETURN, INTERRUPTS ON WITH FLAG CHANGE
549
550 ;----- NOT IN HOLD STATE
551
552 0303                ; NO-HOLD-STATE
553 0303 F6 06 0017 R 0B K28:   TEST   *KB_FLAG,ALT_SHIFT ; ARE WE IN ALTERNATE SHIFT
554 0308 75 03          JNZ    K29              ; JUMP IF ALTERNATE SHIFT
555 030A E9 03A5 R      JMP    K38              ; JUMP IF NOT ALTERNATE
556
557 ;----- TEST FOR CONTROL KEY AND RESET KEY SEQUENCE (CTL ALT DEL)
558
559 030D                ; TEST-RESET
560 030D F6 06 0017 R 04 K29:   TEST   *KB_FLAG,CTL_SHIFT ; ARE WE IN CONTROL SHIFT ALSO
561 0312 75 03          JNZ    K31              ; NO-RESET
562 0314 3C 45          CMP    AL,NUM_KEY       ; CHECK FOR INVALID NUM LOCK KEY
563 0316 74 D6          JE     K26              ; THROW AWAY IF (ALT-CTL)+NUM LOCK
564 0318 3C 46          CMP    AL,SCROLL_KEY    ; CHECK FOR INVALID SCROLL_LOCK KEY
565 031A 74 D2          JE     K26              ; THROW AWAY IF (ALT-CTL)+SCROLL_LOCK
566 031C 3C 53          CMP    AL,DEL_KEY       ; CTL-ALT STATE, TEST FOR DELETE KEY
567 031E 75 2D          JNE    K31              ; NO-RESET
568
569 ;----- CTL-ALT-DEL HAS BEEN FOUND
570

```

```

571 0320 C7 06 0022 R 1234      MOV    0RESET_FLAG,1234H      ; SET FLAG FOR RESET FUNCTION
572 0326 E9 0000 E              JMP    START_T                ; JUMP TO POWER ON DIAGNOSTICS
573
574
575 0329                          ;----- ALT-INPUT-TABLE
576 0329 52 4F 50 51 4B 4C      K30   LABEL    BYTE
577 032F 4D 47 48 49            DB    82,79,80,81,75,76      ; 10 NUMBERS ON KEYPAD
578                                DB    77,71,72,73
579                                ;----- SUPER-SHIFT-TABLE
580 0333 10 11 12 13 14 15      DB    16,17,18,19,20,21    ; A-Z TYPEWRITER CHARS
581 033F 20 21 22 23 24 25      DB    22,23,24,25,30,31
582 0345 26 2C 2D 2E 2F 30      DB    32,33,34,35,36,37
583 034B 31 32                  DB    38,44,45,46,47,48
584                                DB    49,50
585
586                                ;----- IN ALTERNATE SHIFT, RESET NOT FOUND
587 034D                          K31:
588 034D 3C 39                  CMP    AL,57                  ; NO-RESET
589 034F 75 05                  JNE   K32                     ; TEST FOR SPACE KEY
590 0351 B0 20                  MOV    AL,' '                 ; NOT THERE
591 0353 E9 048A R              JMP    K57                     ; SET SPACE CHAR
592                                ; BUFFER_FILL
593
594                                ;----- LOOK FOR KEY PAD ENTRY
595 0356                          K32:
596 0356 BF 0329 R              MOV    DI,OFFSET K30         ; ALT-KEY-PAD
597 0359 59 000A                MOV    CX,10                 ; ALT-INPUT-TABLE
598 035C F2/ AE                REPNE SCASB                  ; LOOK FOR ENTRY USING KEYPAD
599 035E 75 13                  JNE   K33                     ; LOOK FOR MATCH
600 0360 81 EF 032A R          SUB    DI,OFFSET K30+1       ; NO ALT_KEYPAD
601 0364 A0 0019 R              MOV    AL,ALT_INPUT          ; DI NOW HAS ENTRY VALUE
602 0367 B4 0A                  MOV    AH,10                 ; GET THE CURRENT BYTE
603 0369 F6 E4                  MUL   AH                      ; MULTIPLY BY 10
604 036B 03 C7                  ADD   AX,DI                   ; ADD IN THE LATEST ENTRY
605 036D A2 0019 R              MOV    #ALT_INPUT,AL        ; STORE IT AWAY
606 0370 E9 02EE R              JMP    K26                     ; THROW AWAY THAT KEYSTROKE
607
608                                ;----- LOOK FOR SUPERSHIFT ENTRY
609
610 0373                          K33:
611 0373 C6 06 0019 R 00        MOV    0ALT_INPUT,0         ; NO-ALT-KEYPAD
612 0378 B9 001A                MOV    CX,26                 ; ZERO ANY PREVIOUS ENTRY INTO INPUT
613 037B F2/ AE                REPNE SCASB                  ; (DI) (ES) ALREADY POINTING
614 037D 75 05                  JNE   K34                     ; LOOK FOR MATCH IN ALPHABET
615 037F B0 00                  MOV    AL,0                  ; NOT FOUND, FUNCTION KEY OR OTHER
616 0381 E9 048A R              JMP    K57                     ; ASCII CODE OF ZERO
617                                ; PUT IT IN THE BUFFER
618
619                                ;----- LOOK FOR TOP ROW OF ALTERNATE SHIFT
620 0384                          K34:
621 0384 3C 02                  CMP    AL,2                   ; ALT-TOP-ROW
622 0386 72 0C                  JB    K35                     ; KEY WITH '!' ON IT
623 0388 3C 0E                  CMP    AL,14                 ; NOT ONE OF INTERESTING KEYS
624 038A 73 08                  JAE   K35                     ; IS IT IN THE REGION
625 038C 80 C4 76              ADD   AH,118                 ; ALT-FUNCTION
626 038F B0 00                  MOV    AL,0                  ; CONVERT PSEUDO SCAN CODE TO RANGE
627 0391 E9 048A R              JMP    K57                     ; INDICATE AS SUCH
628                                ; BUFFER_FILL
629
630                                ;----- TRANSLATE ALTERNATE SHIFT PSEUDO SCAN CODES
631 0394                          K35:
632 0394 3C 3B                  CMP    AL,59                 ; ALT-FUNCTION
633 0396 73 03                  JAE   K37                     ; TEST FOR IN TABLE
634 0398                          K36:
635 0398 E9 02EE R              JMP    K26                     ; ALT-CONTINUE
636                                ; CLOSE-RETURN
637 039B                          K37:
638 039B 3C 47                  CMP    AL,71                 ; IGNORE THE KEY
639 039D 73 F9                  JAE   K36                     ; ALT-CONTINUE
640 03A2 E9 04E1 R              JMP    K63                     ; IN KEYPAD REGION
641                                ; IF SO, IGNORE
642                                ; BX, OFFSET K13
643                                ; TRANSLATE THAT
644
645                                ;----- NOT IN ALTERNATE SHIFT
646 03A5                          K38:
647 03A5 F6 06 0017 R 04        TEST   0KB_FLAG,CTL_SHIFT    ; NOT-ALT-SHIFT
648 03AA 74 62                  JZ    K44                     ; ARE WE IN CONTROL SHIFT
649                                ; NOT-CTL-SHIFT
650
651                                ;----- CONTROL SHIFT, TEST SPECIAL CHARACTERS
652                                ;----- TEST FOR BREAK AND PAUSE KEYS
653 03AC 3C 46                  CMP    AL,SCROLL_KEY         ; TEST FOR BREAK
654 03AE 75 1D                  JNE   K39                     ; NO-BREAK
655 03B0 8B 1E 0080 R          MOV    BX,0BUFFER_START      ; RESET BUFFER TO EMPTY
656 03B4 89 1E 001A R          MOV    0BUFFER_HEAD,BX
657 03B8 89 1E 001C R          MOV    0BUFFER_TAIL,BX
658 03BC C6 06 0071 R 80        MOV    0BIOS_BREAK,80H      ; TURN ON 0BIOS_BREAK BIT
659
660                                ;----- ENABLE KEYBOARD
661 03C1 B0 AE                  MOV    AL,ENA_KBD            ; ENABLE KEYBOARD
662 03C3 E8 0595 R              CALL  SHIP_IT                 ; EXECUTE ENABLE
663 03C6 CD 1B                  INT    1BH                   ; BREAK INTERRUPT VECTOR
664 03C8 2B C0                  SUB    AX,AX                  ; PUT OUT DUMMY CHARACTER
665 03CA E9 048A R              JMP    K57                     ; BUFFER_FILL
666
667 03CD                          K39:
668 03CD 3C 45                  CMP    AL,NUM_KEY            ; NO-BREAK
669 03CF 75 26                  JNE   K41                     ; LOOK FOR PAUSE KEY
670 03D1 B0 0E 0018 R 08        OR     0KB_FLAG_1,HOLD_STATE ; NO-PAUSE
671                                ; TURN ON THE HOLD FLAG
672
673                                ;----- ENABLE KEYBOARD
674 03D6 B0 AE                  MOV    AL,ENA_KBD            ; ENABLE KEYBOARD
675 03D8 B0 20                  CALL  SHIP_IT                 ; EXECUTE ENABLE
676 03DB DD 20                  MOV    AL,EDI                 ; END OF INTERRUPT TO CONTROL PORT
677 03DD E6 20                  OUT    INTA00,AL              ; ALLOW FURTHER KEYSTROKE INTERRUPTS
678
679                                ;----- DURING PAUSE INTERVAL, TURN COLOR CRT BACK ON
680 03DF B0 3E 0049 R 07        CMP    0CRT_MODE,7           ; IS THIS THE MONOCHROME CARD
681 03E4 74 07                  JE    K40                     ; YES, NOTHING TO DO
682 03E6 BA 03DB                MOV    DX,03DBH              ; PORT FOR COLOR CARD
683 03E9 A0 0065 R              MOV    AL,0CRT_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 K40: ;----- SUSPEND SYSTEM OPERATION (LOOP) TILL NEXT KEY CLEARS HOLD STATE FLAG
689 03ED F6 06 0018 R 08 TEST #KB_FLAG_1,HOLD_STATE ; PAUSE-LOOP
690 03F2 75 F9 JNZ K40 ; CHECK HOLD STATE FLAG
691 ; LOOP UNTIL FLAG TURNED OFF
692 03F4 E9 02F8 R JMP K27A ; INTERRUPT_RETURN_NO_E01
693
694 ;----- TEST SPECIAL CASE KEY 55
695
696 03F7 K41: ; NO-PAUSE
697 03F7 3C 37 CMP AL,55 ; NOT-KEY-55
698 03F9 75 06 JNE K42 ; IS IT IN TABLE
699 03FB BB 7200 MOV AX,114*H ; START/STOP PRINTING SWITCH
700 03FE E9 048A R JMP K57 ; BUFFER_FILL
701
702 ;----- SET UP TO TRANSLATE CONTROL SHIFT
703
704 0401 K42: ; NOT-KEY-55
705 0401 BB 0000 E MOV BX,OFFSET K8 ; SET UP TO TRANSLATE CTL
706 0404 3C 3E CMP AL,59 ; IS IT IN TABLE
707 0406 72 7E JB K56 ; YES, GO TRANSLATE CHAR
708 ; CTL-TABLE-TRANSLATE
709 0408 BB 0000 E MOV BX,OFFSET K9 ; CTL TABLE SCAN
710 040B E9 04E1 R JMP K63 ; TRANSLATE_SCAN
711
712 ;----- NOT IN CONTROL SHIFT
713
714 040E K44: ; NOT-CTL-SHIFT
715 040E 3C 47 CMP AL,71 ; TEST FOR KEYPAD REGION
716 0410 73 33 JAE K48 ; HANDLE KEYPAD REGION
717 0412 F6 06 0017 R 03 TEST #KB_FLAG_LEFT_SHIFT+RIGHT ; SHIFT
718 0417 74 62 JZ K54 ; TEST FOR SHIFT STATE
719
720 ;----- UPPER CASE, HANDLE SPECIAL CASES
721
722 0419 3C 0F CMP AL,15 ; BACK TAB KEY
723 041B 75 05 JNE K45 ; NOT-BACK-TAB
724 041D BB 06 00 MOV AX,15*H ; SET PSEUDO SCAN CODE
725 0420 EB 68 JMP SHORT K57 ; BUFFER_FILL
726
727 0422 K45: ; NOT-BACK-TAB
728 0422 3C 37 CMP AL,55 ; PRINT SCREEN KEY
729 0424 75 10 JNE K46 ; NOT-PRINT-SCREEN
730
731 ;----- ISSUE INTERRUPT TO INDICATE PRINT SCREEN FUNCTION
732
733 0426 B0 AE MOV AL,ENA_KBD ; INSURE KEYBOARD IS ENABLED
734 0428 EB 0595 R CALL SHIP_IT ; EXECUTE ENABLE
735 042B B0 20 MOV AL,E01 ; END OF CURRENT INTERRUPT
736 042D E6 20 OUT INTA00,AL ; SO FURTHER THINGS CAN HAPPEN
737 042F 55 PUSH BP ; SAVE POINTER
738 0430 CD 05 INT 05H ; ISSUE PRINT SCREEN INTERRUPT
739 0432 5D POP BP ; RESTORE POINTER
740 0433 E9 02F3 R JMP K27 ; GO BACK WITHOUT E01 OCCURRING
741
742 0436 K46: ; NOT-PRINT-SCREEN
743 0436 3C 3B CMP AL,59 ; FUNCTION KEYS
744 0438 72 06 JB K47 ; NOT-UPPER-FUNCTION
745 043A BB 0000 E MOV BX,OFFSET K12 ; UPPER CASE PSEUDO SCAN CODES
746 043D E9 04E1 R JMP K63 ; TRANSLATE_SCAN
747
748 0440 K47: ; NOT-UPPER-FUNCTION
749 0440 BB 0000 E MOV BX,OFFSET K11 ; POINT TO UPPER CASE TABLE
750 0443 EB 41 JMP SHORT K56 ; OK, TRANSLATE THE CHAR
751
752 ;----- KEYPAD KEYS, MUST TEST NUM LOCK FOR DETERMINATION
753
754 0445 K48: ; KEYPAD-REGION
755 0445 F6 06 0017 R 20 TEST #KB_FLAG_NUM_STATE ; ARE WE IN NUM_LOCK
756 044A 75 21 JNZ K52 ; TEST FOR SURE_
757 044C F6 06 0017 R 03 TEST #KB_FLAG_LEFT_SHIFT+RIGHT ; SHIFT ; ARE WE IN SHIFT STATE
758 0451 75 21 JNZ K53 ; IF SHIFTED, REALLY NUM STATE
759
760 ;----- BASE CASE FOR KEYPAD
761
762 0453 K49: ; BASE-CASE
763
764 0453 3C 4A CMP AL,74 ; SPECIAL CASE FOR A COUPLE OF KEYS
765 0455 74 0C JE K50 ; MINUS
766 0457 3C 4E CMP AL,78
767 0459 74 0D JE K51
768 045B 2C 47 SUB AL,71 ; CONVERT ORIGIN
769 045D BB 0000 E MOV BX,OFFSET K15 ; BASE CASE TABLE
770 0460 E9 04E3 R JMP K64 ; CONVERT TO PSEUDO SCAN
771 0463 K50: ; MINUS
772 0463 BB 4A2D MOV AX,74*H+'' ; BUFFER_FILL
773 0466 EB 22 JMP SHORT K57
774 0468 K51: ; PLUS
775 0468 BB 4E2B MOV AX,78*H+'' ; BUFFER_FILL
776 046B EB 1D JMP SHORT K57
777
778 ;----- MIGHT BE NUM LOCK, TEST SHIFT STATUS
779
780 046D K52: ; ALMOST-NUM-STATE
781 046D F6 06 0017 R 03 TEST #KB_FLAG_LEFT_SHIFT+RIGHT ; SHIFTED TEMP OUT OF NUM STATE
782 0472 75 DF JNZ K49
783
784 0474 K53: ; REALLY NUM STATE
785 0474 2C 46 SUB AL,70 ; CONVERT ORIGIN
786 0476 BB 0000 E MOV BX,OFFSET K14 ; NUM STATE TABLE
787 0479 EB 0B JMP SHORT K56 ; TRANSLATE_CHAR
788
789 ;----- PLAIN OLD LOWER CASE
790
791 047B K54: ; NOT-SHIFT
792 047B 3C 3B CMP AL,59 ; TEST FOR FUNCTION KEYS
793 047D 72 04 JB K55 ; NOT-LOWER-FUNCTION
794 047F B0 00 MOV AL,0 ; SCAN CODE IN AH ALREADY
795 0481 EB 07 JMP SHORT K57 ; BUFFER_FILL
796
797 0483 K55: ; NOT-LOWER-FUNCTION
798 0483 BB 0000 E MOV BX,OFFSET K10 ; LC TABLE

```

```

799
800
801
802 0486          ;----- TRANSLATE THE CHARACTER
803 0486 FE C8   K56:      DEC AL          ; TRANSLATE-CHAR
804 048B 2E: D7  XLAT     CS:K11    ; CONVERT ORIGIN
805                                     ; CONVERT THE SCAN CODE TO ASCII
806
807
808 048A          ;----- PUT CHARACTER INTO BUFFER
809 048A 3C FF   K57:      CMP AL,-1      ; BUFFER-FILL
810 048C 74 1F   JE K59          ; IS THIS AN IGNORE CHAR
811 048E 80 FC FF CMP AH,-1      ; YES, DO NOTHING WITH IT
812 0491 74 1A   JE K59          ; LOOK FOR -1 PSEUDO SCAN
813                                     ; NEAR_INTERRUPT_RETURN
814
815
816 0493          ;----- HANDLE THE CAPS LOCK PROBLEM
817 0493 F6 06 0017 R 40 K58:      TEST 0KB_FLAG,CAPS_STATE ; BUFFER-FILL-NOTEST
818 0498 74 20   JZ K61          ; ARE WE IN CAPS LOCK STATE
819                                     ; SKIP IF NOT
820
821
822 049A F6 06 0017 R 03 ;----- IN CAPS LOCK STATE
823 049F 74 0F   JZ K60          ; TEST 0KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT ; TEST FOR SHIFT STATE
824                                     ; IF NOT SHIFT, CONVERT LOWER TO UPPER
825
826
827 04A1 3C 41   ;----- CONVERT ANY UPPER CASE TO LOWER CASE
828 04A3 72 15   CMP AL,'A'      ; LOWER-TO-UPPER
829 04A5 3C 5A   JB K61          ; FIND OUT IF ALPHABETIC
830 04A7 77 11   CMP AL,'Z'      ; NOT_CAPS_STATE
831 04A9 04 20   JA K61          ; NOT_CAPS_STATE
832 04AB EB 0D   ADD AL,'A'-'A'  ; CONVERT TO LOWER CASE
833 JMP SHORT K61 ; NOT_CAPS_STATE
834
835 04AD E9 02EE R K59:      JMP K26          ; NEAR_INTERRUPT_RETURN
836 ; INTERRUPT_RETURN
837
838
839 04B0          ;----- CONVERT ANY LOWER CASE TO UPPER CASE
840 04B0 3C 61   K60:      CMP AL,'a'      ; LOWER-TO-UPPER
841 04B2 72 06   JB K61          ; FIND OUT IF ALPHABETIC
842 04B4 3C 7A   CMP AL,'z'      ; NOT_CAPS_STATE
843 04B6 77 02   JA K61          ; NOT_CAPS_STATE
844 04B8 2C 20   SUB AL,'a'-'A'  ; CONVERT TO UPPER CASE
845
846 04BA          ;-----
847 04BA BB 1E 001C R K61:      MOV BX,0BUFFER_TAIL ; NOT-CAPS-STATE
848 04BE 8B FF R  MOV SI,BX      ; GET THE END POINTER TO THE BUFFER
849 04C0 E8 007F R CALL K4         ; SAVE THE VALUE
850 04C3 3B 1E 001A R CMP BX,0BUFFER_HEAD ; ADVANCE THE TAIL
851 04C7 74 22   JE K62          ; HAS THE BUFFER WRAPPED AROUND
852 04C9 89 04   MOV [SI],AX     ; BUFFER FULL BEEP
853 04CB 89 1E 001C R MOV 0BUFFER_TAIL,BX ; STORE THE VALUE
854 04CF FA      CLI           ; MOVE THE POINTER UP
855 04D0 B0 20   MOV AL,E01      ; TURN OFF INTERRUPTS
856 04D2 E6 20   OUT INTA00,AL   ; END OF INTERRUPT COMMAND
857 04D4 B0 AE   MOV AL,ENA_KBD  ; SEND COMMAND TO INTERRUPT CONTROL PORT
858 04D6 EB 0595 R CALL SHIP IT     ; INSURE KEYBOARD IS ENABLED
859 04D9 B8 9102 MOV AX,09102H   ; MOVE IN POST CODE & TYPE
860 04DC CD 15   INT 15H         ; PERFORM OTHER FUNCTION
861 04DE E9 02F8 R JMP K27A        ; INTERRUPT_RETURN
862
863
864
865
866 04E1          ;----- TRANSLATE SCAN FOR PSEUDO SCAN CODES
867 04E1 2C 3B   K63:      SUB AL,59        ; TRANSLATE-SCAN
868 04E3          ; CONVERT ORIGIN TO FUNCTION KEYS
869 04E3 2E: D7  K64:      XLAT CS:K9      ; TRANSLATE-SCAN-ORGD
870 04E5 BA E0   MOV AH,AL       ; CTL TABLE SCAN
871 04E7 B0 00   MOV AL,0        ; PUT VALUE INTO AH
872 04E9 EB 9F   JMP K57         ; ZERO ASCII CODE
873                                     ; PUT IT INTO THE BUFFER
874
875
876 04EB          ;-----
877 04EB B0 20   K62:      MOV AL,E01      ; ENABLE INTERRUPT CONTROLLER CHIP
878 04ED E6 20   OUT INTA00,AL   ;
879 04EF B9 02A6 MOV CX,678      ; DIVISOR FOR 1760 HZ
880 04F2 B3 04   MOV BL,4        ; SHORT BEEP COUNT (1/16 + 1/64 DELAY)
881 04F4 EB 0000 E CALL BEEP       ; GO TO COMMON BEEP HANDLER
882 04F7 E9 02F3 R JMP K27         ; EXIT
883
884
885
886
887
888
889
890
891
892
893 04FA          ;-----
894 04FA 50      SND_DATA PROC NEAR ;
895 04FB 53      PUSH AX          ; SAVE REGISTERS
896 04FC 51      PUSH CX          ;
897 04FD BA F8   MOV BH,AL        ; SAVE TRANSMITTED BYTE FOR RETRIES
898 04FF B3 03   MOV BL,3         ; LOAD RETRY COUNT
899 0501 FA      S0D0:          ;
900 0501 FA      CLI           ; DISABLE INTERRUPTS
901 0502 B0 26 0097 R CF AND 0KB_FLAG_2,NOT (KB_FE+KB_FA) ; CLEAR ACK AND RESEND FLAGS
902
903
904
905 0507 2B C9   ;----- WAIT FOR ANY PENDING COMMAND TO BE ACCEPTED
906 0509          ;
907 0509 E4 64   SD1:          ;
908 050B AB 02   IN AL,STATUS_PORT ; READ KEYBOARD PROCESSOR STATUS PORT
909 050D E0 FA   TEST AL,INPT_BUF_FULL ; CHECK FOR ANY PENDING COMMAND
910 ; WAIT FOR COMMAND TO BE ACCEPTED
911 050F BA C7   MOV AH,BH        ; REESTABLISH BYTE TO TRANSMIT
912 0511 E6 60   OUT PORT_A,AL    ; SEND BYTE

```

```

913 0513 FB          STI          ; ENABLE INTERRUPTS
914 0514 B9 1A00    MOV          CX,01A00H    ; 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 FE CB          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 052B            SD7:        TEST     *KB_FLAG_2,KB_FA    ; SEE IF THIS IS AN ACKNOWLEDGE
927 052B F6 06 0097 R 10 JZ     SD5          ; IF NOT, GO RESEND
928 0530 74 EE          ;
929 0532            SD9:        POP     CX          ; RESTORE REGISTERS
930 0532 59          POP     BX          ;
931 0533 5B          POP     AX          ;
932 0534 58          RET          ; RETURN, GOOD TRANSMISSION
933 0535 C3          ;
934 0536            SND_DATA ENDP
935                ;
936                ;
937                ;
938                ;
939                ;
940                ;
941                ;
942                ;
943 0536            SND_LED PROC NEAR
944 0536 FA          CLI          ; TURN OFF INTERRUPTS
945 0537 F6 06 0097 R 40 TEST    *KB_FLAG_2,KB_PR_LED ; CHECK FOR MODE INDICATOR UPDATE
946 053C 75 47          JNZ          SL9          ; DON'T UPDATE AGAIN IF UPDATE UNDERWAY
947                ;
948 053E 80 0E 0097 R 40 OR      *KB_FLAG_2,KB_PR_LED ; TURN ON UPDATE IN PROCESS
949 0543 80 20          MOV     AL,_EOI          ; END OF INTERRUPT COMMAND
950 0545 E6 20          OUT    INTAGO,AL        ; SEND COMMAND TO INTERRUPT CONTROL PORT
951 0547 EB 0D          JMP     SHORT SL3        ; GO SEND MODE INDICATOR COMMAND
952                ;
953 0549            SND_LED1:
954 0549 FA          CLI          ; TURN OFF INTERRUPTS
955 054A F6 06 0097 R 40 TEST    *KB_FLAG_2,KB_PR_LED ; CHECK FOR MODE INDICATOR UPDATE
956 054F 75 34          JNZ          SL9          ; DON'T UPDATE AGAIN IF UPDATE UNDERWAY
957                ;
958 0551 80 0E 0097 R 40 OR      *KB_FLAG_2,KB_PR_LED ; TURN ON UPDATE IN PROCESS
959 0556            ;
960 0556 80 ED          MOV     AL,_LED_CMD     ; LED CMD BYTE
961 0558 EB 04FA R     CALL    SND_DATA      ; SEND DATA TO KEYBOARD
962 0558 FA          CLI          ;
963 055C E8 0587 R     CALL    MAKE_LED       ; GO FORM INDICATOR DATA BYTE
964 055F 80 26 0097 R F8 AND     *KB_FLAG_2,NOT KB_LEDS ; CLEAR MODE INDICATOR BITS
965 0564 08 06 0097 R 80 OR      *KB_FLAG_2,AL        ; SAVE INDICATORS STATES FOR NEXT TIME
966 0568 F6 06 0097 R 80 TEST    *KB_FLAG_2,KB_ERR ; TRANSMIT ERROR DETECTED
967 056D 75 0B          JNZ          SL5          ; IF SO, BYPASS SECOND BYTE TRANSMISSION
968                ;
969 056F EB 04FA R     CALL    SND_DATA      ; SEND DATA TO KEYBOARD
970 0572 FA          CLI          ; TURN OFF INTERRUPTS
971 0573 F6 06 0097 R 80 TEST    *KB_FLAG_2,KB_ERR ; TRANSMIT ERROR DETECTED
972 0578 74 06          JZ     SL7          ; IF NOT, DON'T SEND AN ENABLE COMMAND
973 057A            SL5:        MOV     AL,_KB_ENABLE    ; GET KEYBOARD CSA ENABLE COMMAND
974 057A 80 F4          CALL    SND_DATA      ; SEND DATA TO KEYBOARD
975 057C EB 04FA R     CLI          ; TURN OFF INTERRUPTS
976 057F FA          ;
977 0580            SL7:        AND     *KB_FLAG_2,NOT(KB_PR_LED+KB_ERR) ; TURN OFF MODE INDICATOR
978 0580 80 26 0097 R 3F ; UPDATE AND TRANSMIT ERROR FLAG
979 0585 FB          STI          ; ENABLE INTERRUPTS
980 0585 FB          RET          ; RETURN TO CALLER
981 0586 C3          ;
982 0587            SND_LED ENDP
983                ;
984                ;
985                ;
986                ;
987                ;
988                ;
989                ;
990                ;
991 0587            MAKE_LED PROC NEAR
992 0587 51          PUSH   CX          ; SAVE CX
993 0588 A0 0017 R    MOV     AL,*KB_FLAG    ; GET CAPS & NUM LOCK INDICATORS
994 058B 24 70          AND     AL,_CAPS_STATE+NUM_STATE+SCROLL_STATE ; ISOLATE INDICATORS
995 058D B1 04          MOV     CL,_4          ; SHIFT COUNT
996 058F 02 C0          ROL    AL,_CL          ; SHIFT BITS OVER TO TURN ON INDICATORS
997 0591 24 07          AND     AL,_07H        ; MAKE SURE ONLY MODE BITS ON
998 0593 59          POP     CX          ;
999 0594 C3          RET          ; RETURN TO CALLER
1000 0595            MAKE_LED ENDP
1001                ;
1002                ;
1003                ;
1004                ;
1005                ;
1006                ;
1007                ;
1008 0595            SHIP_IT PROC NEAR
1009 0595 50          PUSH   AX          ; SAVE DATA TO SEND
1010                ;
1011                ;
1012 0596 FA          CLI          ; DISABLE INTERRUPTS TILL DATA SENT
1013 0597 2B C9          SUB     CX,CX          ; CLEAR TIMEOUT COUNTER
1014 0599            S10:        ;
1015 0599 E4 64          IN     AL,_STATUS_PORT ; READ KEYBOARD CONTROLLER STATUS
1016 059B AB 02          TEST    AL,_INPT_BUF_FULL ; CHECK FOR ITS INPUT BUFFER BUSY
1017 059D E0 FA          LOOPNZ S10          ; WAIT FOR COMMAND TO BE ACCEPTED
1018                ;
1019 059F 58          POP     AX          ; GET DATA TO SEND
1020 05A0 E6 64          OUT    STATUS_PORT,AL ; SEND TO KEYBOARD CONTROLLER
1021 05A2 FB          STI          ; ENABLE INTERRUPTS AGAIN
1022 05A3 C3          RET          ; RETURN TO CALLER
1023 05A4            SHIP_IT ENDP
1024 05A4            CODE ENDS
1025                ;

```

```

1
2 PAGE 118,121
3 TITLE PRT ----- 06/10/85 PRINTER ADAPTER BIOS
4 -286C
5 .LIST
6 0000 CODE SEGMENT BYTE PUBLIC
7 PUBLIC PRINTER_IO_1
8 EXTRN DDS:NEAR
9
10
11 :--- INT 17 H -----
12 : PRINTER_IO :
13 : THIS ROUTINE PROVIDES COMMUNICATION WITH THE PRINTER :
14 : INPUT :
15 : (AH)= 00H PRINT THE CHARACTER IN (AL) :
16 : ON RETURN, (AH)= 1 IF CHARACTER NOT BE PRINTED (TIME OUT) :
17 : OTHER BITS SET AS ON NORMAL STATUS CALL :
18 : (AH)= 01H INITIALIZE THE PRINTER PORT :
19 : RETURNS WITH (AH) SET WITH PRINTER STATUS :
20 : (AH)= 02H READ THE PRINTER STATUS INTO (AH) :
21 : :
22 : :
23 : :
24 : :
25 : :
26 : :
27 : :
28 : :
29 : :
30 : :
31 : :
32 : :
33 : :
34 : :
35 : :
36 : (DX) = PRINTER TO BE USED (0,1,2) CORRESPONDING TO ACTUAL VALUES :
37 : IN PRINTER_BASE AREA :
38 : DATA AREA @PRINTER_BASE CONTAINS THE BASE ADDRESS OF THE PRINTER CARD(S) :
39 : AVAILABLE (LOCATED AT BEGINNING OF DATA SEGMENT, 400H ABSOLUTE, 3 WORDS) :
40 : :
41 : DATA AREA @PRINT_TIM_OUT (BYTE) MAY BE CHANGE TO CAUSE DIFFERENT :
42 : TIME OUT WAITS. DEFAULT=20 * 4 :
43 : :
44 : REGISTERS (AH) IS MODIFIED WITH STATUS INFORMATION :
45 : ALL OTHERS UNCHANGED :
46 :-----
47 ASSUME CS:CODE,DS:DATA
48 0000 PRINTER_IO_1 PROC FAR ; ENTRY POINT FOR ORG 0EFD2H
49 0000 FB STI ; INTERRUPTS BACK ON
50 0001 IE PUSH DS ; SAVE SEGMENT
51 0002 56 PUSH SI
52 0003 52 PUSH DX
53 0004 51 PUSH CX
54 0005 53 PUSH BX
55 0006 EB 0000 E CALL DDS ; ADDRESS DATA SEGMENT
56 0009 BB F2 MOV SI,DX ; GET PRINTER PARAMETER
57 000B 8A 9C 007B R MOV BL,@PRINT_TIM_OUT[SI] ; LOAD TIMEOUT VALUE
58 000F D1 E6 SHL SI,1 ; WORD OFFSET INTO TABLE INTO (SI)
59 0011 BB 94 000B R MOV DX,@PRINTER_BASE[SI] ; GET BASE ADDRESS FOR PRINTER CARD
60 0015 0B D2 OR DX,DX ; TEST DX = ZERO, INDICATING NO PRINTER
61 0017 74 0C JZ B10 ; EXIT, NO PRINTER ADAPTER AT OFFSET
62 0019 0A E4 OR AH,AH ; TEST FOR (AH)= 00H
63 001B 74 0E JZ B20 ; PRINT CHARACTER IN (AL)
64 001D FE CC DEC AH ; TEST FOR (AH)= 01H
65 001F 74 58 JZ B80 ; INITIALIZE PRINTER
66 0021 FE CC DEC AH ; TEST FOR (AH)= 02H
67 0023 74 3F JZ B50 ; GET PRINTER STATUS
68 0025 B10:
69 0025 5B POP BX ; RETURN
70 0026 59 POP CX
71 0027 5A POP DX
72 0028 5E POP SI ; RECOVER REGISTERS
73 0029 IF POP DS
74 002A CF IRET ; RETURN TO CALLING PROGRAM
75
76
77 :----- PRINT THE CHARACTER IN (AL)
78
79 002B B20:
80 002B 50 PUSH AX ; SAVE VALUE TO PRINT
81 002C EE OUT DX,AL ; OUTPUT CHARACTER TO DATA PORT
82 002D 42 INC DX ; POINT TO STATUS PORT
83
84 :----- CHECK FOR PRINTER BUSY
85
86 002E 53 PUSH BX ; SAVE TIMEOUT BASE COUNT
87 002F EC IN AL,DX ; GET STATUS PORT VALUE
88 0030 A8 80 TEST AL,80H ; IS THE PRINTER CURRENTLY BUSY
89 0032 75 05 JNZ B25 ; SKIP SYSTEM DEVICE BUSY CALL IF NOT
90
91 :----- INT 15 H -- DEVICE BUSY
92
93 0034 BB 90FE MOV AX,90FEH ; FUNCTION 90 PRINTER ID
94 0037 CD 15 INT 15H ; SYSTEM CALL
95
96 :----- WAIT BUSY
97
98 0039 B25:
99 0039 2A FF SUB BH,BH ; ADJUST OUTER LOOP COUNT
100 003B C1 D3 02 RCL BX,2 ; CLEAR (BH)
101 003E 2B C9 SUB CX,CX ; MULTIPLY BY 4
102 0040 B30:
103 0040 EC B30:
104 0041 8A E0 IN AL,DX ; GET STATUS
105 0043 A8 80 TEST AL,80H ; IS THE PRINTER CURRENTLY BUSY
106 0045 75 0E JNZ B40 ; GO TO OUTPUT STROBE
107 0047 E2 F7 LOOP B35 ; LOOP IF NOT
108 0049 4B DEC BX ; DECREMENT OUTER LOOP COUNT
109 004A 75 F2 JNZ B30 ; MAKE ANOTHER PASS IF NOT ZERO
110
111 004C 5B POP BX ; CLEAR (BX) FROM STACK
112 004D 80 CC 01 OR AH,1 ; SET ERROR FLAG
113 0050 80 E4 F9 AND AH,0F9H ; TURN OFF THE UNUSED BITS
114 0053 EB 1C JMP SHORT B70 ; RETURN WITH ERROR FLAG SET

```

```

115 0055          B40:          ; SEND STROBE PULSE
116 0055 5B      POP          BX          ; RESTORE (BX) WITH TIMEOUT COUNT
117 0056 B0 0D   MOV          AL,0DH        ; SET THE STROBE LOW (BIT ON)
118 0058 42     INC          DX          ; OUTPUT STROBE TO CONTROL PORT
119 0059 FA     CLI          DX          ; PREVENT INTERRUPT PULSE STRETCHING
120 005A EE     OUT          DX,AL        ; OUTPUT STROBE BIT > 1µs < 5µs
121 005B EB 00   JMP          $+2            ; I/O DELAY TO ALLOW FOR LINE LOADING
122 005D EB 00   JMP          $+2            ; AND FOR CORRECT PULSE WIDTH
123 005F B0 0C   MOV          AL,0CH        ; SET THE -STROBE HIGH
124 0061 EE     OUT          DX,AL
125 0062 FB     STI          AX          ; INTERRUPTS BACK ON
126 0063 58     POP          AX          ; RECOVER THE OUTPUT CHAR
127
128             ;----- PRINTER STATUS
129
130 0064          B50:          ; SAVE (AL) REGISTER
131 0064 50     PUSH         AX
132 0065          B60:          ; GET PRINTER ATTACHMENT BASE ADDRESS
133 0065 8B 94 00B R MOV          DX,0PRINTER_BASE[S1] ; POINT TO CONTROL PORT
134 0069 42     INC          DX          ; PRE-CHARGE BUSY LINE IF FLOATING
135 006A EC     IN          AL,DX         ; GET PRINTER STATUS HARDWARE BITS
136 006B EC     IN          AL,DX
137 006C 8A E0   MOV          AH,AL        ; SAVE
138 006E 80 E4 F8 AND          AH,0F8H      ; TURN OFF UNUSED BITS
139 0071          B70:          ; RECOVER (AL) REGISTER
140 0071 5A     POP          DX          ; MOVE CHARACTER INTO (AL)
141 0072 8A C2   MOV          AL,DL        ; FLIP A COUPLE OF BITS
142 0074 80 F4 48 XOR          AH,48H
143 0077 EB AC   JMP          B10         ; RETURN FROM ROUTINE WITH STATUS IN AH
144
145             ;----- INITIALIZE THE PRINTER PORT
146
147 0079          B80:          ; SAVE (AL)
148 0079 50     PUSH         AX          ; POINT TO OUTPUT PORT
149 007A 42     INC          DX
150 007B 42     INC          DX
151 007C B0 08   MOV          AL,8         ; SET INIT LINE LOW
152 007E EE     OUT          DX,AL
153 007F B8 0FA0 MOV          AX,1000*4    ; ADJUST FOR INITIALIZATION DELAY LOOP
154 0082          B90:          ; INIT_LOOP
155 0082 48     DEC          AX          ; LOOP FOR RESET TO TAKE
156 0083 75 FD   JNZ         B90         ; INIT_LOOP
157 0085 B0 0C   MOV          AL,0CH      ; NO INTERRUPTS, NON AUTO LF, INIT HIGH
158 0087 EE     OUT          DX,AL
159 0088 EB DB   JMP          B60         ; EXIT THROUGH STATUS ROUTINE
160
161 008A          PRINTER_IO_1 ENDP
162
163 008A          CODE ENDS
164

```

```

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_IO_1
7      EXTRN A1:NEAR
8      EXTRN DDS:NEAR
9
10     ;----- INT 14 H -----
11     RS232_IO
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     ;          7          6          5          4          3          2          1          0
19     ;          ---- BAUD RATE -- - - - - PARITY - - - - STOPBIT - - - - WORD LENGTH - -
20
21     ;          000 - 110          X0 - NONE          0 - 1          10 - 7 BITS
22     ;          001 - 150          01 - ODD          1 - 2          11 - 8 BITS
23     ;          010 - 300          11 - EVEN
24     ;          011 - 600
25     ;          100 - 1200
26     ;          101 - 2400
27     ;          110 - 4800
28     ;          111 - 9600
29     ; ON RETURN, CONDITIONS SET AS IN CALL TO COMMO STATUS (AH=03H)
30
31     ; (AH) = 01H SEND THE CHARACTER IN (AL) OVER THE COMMO 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     ; (AH) = 02H RECEIVE A CHARACTER IN (AL) FROM COMMO LINE BEFORE
39     ; RETURNING TO CALLER
40     ; ON EXIT, (AH) HAS THE CURRENT LINE STATUS, AS SET BY THE
41     ; THE STATUS ROUTINE, EXCEPT THAT THE ONLY BITS
42     ; LEFT ON ARE THE ERROR BITS (7,4,3,2,1)
43     ; IF (AH) HAS BIT 7 ON (TIME OUT) THE REMAINING
44     ; BITS ARE NOT PREDICTABLE.
45     ; THUS, (AH) IS NON ZERO ONLY WHEN AN ERROR OCCURRED.
46     ; (AH) = 03H RETURN THE COMMO PORT STATUS IN (AX)
47     ; (AH) CONTAINS THE LINE CONTROL STATUS
48     ; BIT 7 = TIME OUT
49     ; BIT 6 = TRANSMIT SHIFT REGISTER EMPTY
50     ; BIT 5 = TRANSMIT HOLDING REGISTER EMPTY
51     ; BIT 4 = BREAK DETECT
52     ; BIT 3 = FRAMING ERROR
53     ; BIT 2 = PARITY ERROR
54     ; BIT 1 = OVERRUN ERROR
55     ; BIT 0 DATA READY
56     ; (AL) CONTAINS THE MODEM STATUS
57     ; BIT 7 = RECEIVE LINE SIGNAL DETECT
58     ; BIT 6 = RING INDICATOR
59     ; BIT 5 = DATA SET READY
60     ; BIT 4 = CLEAR TO SEND
61     ; BIT 3 = DELTA RECEIVE LINE SIGNAL DETECT
62     ; BIT 2 = TRAILING EDGE RING DETECTOR
63     ; BIT 1 = DELTA DATA SET READY
64     ; BIT 0 = DELTA CLEAR TO SEND
65
66     ; (DX) = PARAMETER INDICATING WHICH RS232 CARD (0,1 ALLOWED)
67
68     ; DATA AREA #RS232_BASE CONTAINS THE BASE ADDRESS OF THE 8250 ON THE CARD
69     ; LOCATION 400H CONTAINS UP TO 4 RS232 ADDRESSES POSSIBLE
70     ; DATA AREA LABEL #RS232_TIM_OUT (BYTE) CONTAINS OUTER LOOP COUNT
71     ; VALUE FOR TIMEOUT (DEFAULT=1)
72
73     ; OUTPUT
74     ; AX MODIFIED ACCORDING TO PARAMETERS OF CALL
75     ; ALL OTHERS UNCHANGED
76
77     ;-----
78     ASSUME CS:CODE,DS:DATA
79
80     0000 RS232_IO_1 PROC FAR
81     ;----- VECTOR TO APPROPRIATE ROUTINE -----
82     STI ; INTERRUPTS BACK ON
83     PUSH DS ; SAVE SEGMENT
84     PUSH DX
85     PUSH SI
86     PUSH DI
87     PUSH CX
88     PUSH BX
89     MOV SI,DX ; RS232 VALUE TO (SI)
90     MOV DI,DX ; AND TO (DI) (FOR TIMEOUTS)
91     SHL SI,1 ; WORD OFFSET
92     CALL DDS
93     MOV DX,#RS232_BASE[SI] ; GET BASE ADDRESS
94     OR DX,DX ; TEST FOR 0 BASE ADDRESS
95     JZ A3 ; RETURN
96     OR AH,AH ; TEST FOR (AH)=00H
97     JZ A4 ; COMMO INITIALIZATION
98     DEC AH ; TEST FOR (AH)=01H
99     JZ A5 ; SEND (AL)
100    DEC AH ; TEST FOR (AH)=02H
101    JZ A12 ; RECEIVE INTO (AL)
102    DEC AH
103    DEC AH ; TEST FOR (AH)=03H
104    JNZ A3
105    JMP A18 ; COMMUNICATION STATUS
106    ; RETURN FROM RS232
107    POP BX
108    POP CX
109    POP DI
110    POP SI
111    POP DX
112    POP DS
113    IRET ; RETURN TO CALLER, NO ACTION

```



```

114                                     PAGE
115                                     ;----- INITIALIZE THE COMMUNICATIONS PORT
116
117 0032                                     A4:
118 0032 BA E0                                     MOV AH,AL ; SAVE INITIALIZATION PARAMETERS IN (AH)
119 0034 B3 C2 03                               ADD DX,3 ; POINT TO 8250 CONTROL REGISTER
120 0037 B0 80                                     MOV AL,80H
121 0039 EE                                     OUT DX,AL ; SET DLAB=1
122
123                                     ;----- DETERMINE BAUD RATE DIVISOR
124
125 003A BA D4                                     MOV DL,AH ; GET PARAMETERS TO (DL)
126 003C B1 04                                     MOV CL,4
127 003E D2 C2                                     ROL DL,CL
128 0040 B1 E2 000E                             AND DX,DEH ; ISOLATE THEM
129 0044 BF 0000 E                               D1,OFFSET A1 ; BASE OF TABLE
130 0047 03 FA                                     ADD DI,DX ; PUT INTO INDEX REGISTER
131 0049 BB 94 0000 R                             MOV DX,®RS232_BASE[S1] ; POINT TO HIGH ORDER OF DIVISOR
132 004D 42                                     INC DX
133 004E 2E: 8A 45 01                             MOV AL,CS:[DI]+1 ; GET HIGH ORDER OF DIVISOR
134 0052 EE                                     OUT DX,AL ; SET ms OF DIVISOR TO 0
135 0053 4A                                     DEC DX
136 0054 EB 00                                     JMP $+2 ; I/O DELAY
137 0056 2E: 8A 05                             MOV AL,CS:[DI] ; GET LOW ORDER OF DIVISOR
138 0059 EE                                     OUT DX,AL ; SET LOW OF DIVISOR
139 005A B3 C2 03                               ADD DX,3 ; GET PARAMETERS BACK
140 005D BA C4                                     MOV AL,AH ; STRIP OFF THE BAUD BITS
141 005F 24 1F                             AND AL,01FH ; LINE CONTROL TO 8 BITS
142 0061 EE                                     OUT DX,AL
143 0062 4A                                     DEC DX
144 0063 4A                                     DEC DX
145 0064 EB 00                                     JMP $+2 ; I/O DELAY
146 0066 B0 00                                     MOV AL,0 ; INTERRUPT ENABLES ALL OFF
147 0068 EE                                     OUT DX,AL ; COM STATUS
148 0069 EB 4B                                     JMP SHORT A1B
149
150                                     ;----- SEND CHARACTER IN (AL) OVER COMMO LINE
151
152 006B 50                                     A5:                                     ; SAVE CHAR TO SEND
153 006C B3 C2 04                               ADD DX,4 ; MODEM CONTROL REGISTER
154 006F B0 03                                     MOV AL,3 ; DTR AND RTS
155 0071 EE                                     OUT DX,AL ; DATA TERMINAL READY, REQUEST TO SEND
156 0072 42                                     INC DX ; MODEM STATUS REGISTER
157 0073 42                                     INC DX
158 0074 B7 30                                     MOV BH,30H ; DATA SET READY & CLEAR TO SEND
159 0076 EB 00C5 R                             CALL WAIT_FOR_STATUS ; ARE BOTH TRUE
160 0079 74 08                                     JE A9 ; YES, READY TO TRANSMIT CHAR
161
162 007B 59                                     A7:                                     ; RELOAD DATA BYTE
163 007B 59                                     POP CX
164 007C BA C1                                     MOV AL,CL
165 007E
166 007E 80 CC 80                               A8:                                     ; INDICATE TIME OUT
167 0081 EB A8                                     OR AH,80H ; RETURN
168 0081 EB A8                                     JMP A3
169
169 0083 59                                     A9:                                     ; CLEAR TO SEND
170 0083 4A                                     DEC DX ; LINE STATUS REGISTER
171 0084                                     WAIT SEND
172 0084 B7 20                                     MOV BH,20H ; IS TRANSMITTER READY
173 0086 EB 00C5 R                             CALL WAIT_FOR_STATUS ; TEST FOR TRANSMITTER READY
174 0089 75 F0                                     JNZ A7 ; RETURN WITH TIME OUT SET
175 008B
176 008B B3 EA 05                               A11:                                     ; OUT CHAR
177 008E 59                                     SUB DX,5 ; DATA PORT
178 008F 8A C1                                     POP CX ; RECOVER IN CX TEMPORARILY
179 0091 EE                                     MOV AL,CL ; MOVE CHAR TO AL FOR OUT, STATUS IN AH
180 0092 EB 97                                     OUT DX,AL ; OUTPUT CHARACTER
181 0092 EB 97                                     JMP A3 ; RETURN
182
183                                     ;----- RECEIVE CHARACTER FROM COMMO LINE
184
184 0094                                     A12:                                     ; MODEM CONTROL REGISTER
185 0094 B3 C2 04                               ADD DX,4 ; DATA TERMINAL READY
186 0097 B0 01                                     MOV AL,1 ; MODEM STATUS REGISTER
187 0099 EE                                     OUT DX,AL
188 009A 42                                     INC DX
189 009B 42                                     INC DX ; WAIT_DSR
190 009C                                     MOV BH,20H ; DATA SET READY
191 009C B7 20                                     CALL WAIT_FOR_STATUS ; TEST FOR DSR
192 009E EB 00C5 R                             CALL WAIT_FOR_STATUS ; RETURN WITH ERROR
193 00A1 75 D8                                     JNZ A8 ; WAIT_DSR END
194 00A3
195 00A3 4A                                     A15:                                     ; LINE STATUS REGISTER
196 00A4                                     DEC DX ; WAIT RECV
197 00A4 B7 01                                     MOV BH,1 ; RECEIVE BUFFER FULL
198 00A6 EB 00C5 R                             CALL WAIT_FOR_STATUS ; TEST FOR RECEIVE BUFFER FULL
199 00A9 75 D3                                     JNZ A8 ; SET TIME OUT ERROR
200 00AB
201 00AB 80 E4 1E                               A17:                                     ; GET CHAR
202 00AB 80 E4 1E                               AND AH,00011110B ; TEST FOR ERROR CONDITIONS ON RECEIVE
203
203 00AE BB 94 0000 R                             MOV DX,®RS232_BASE[S1] ; DATA PORT
204 00B2 EC                                     IN AL,DX ; GET CHARACTER FROM LINE
205 00B3 E9 002B R                             JMP A3 ; RETURN
206
207                                     ;----- COMMO PORT STATUS ROUTINE
208
209 00B6                                     A18:                                     ; CONTROL PORT
210 00B6 BB 94 0000 R                             MOV DX,®RS232_BASE[S1] ; GET LINE CONTROL STATUS
211 00BA B3 C2 05                               ADD DX,5 ; PUT IN (AH) FOR RETURN
212 00BD EC                                     IN AL,DX ; POINT TO MODEM STATUS REGISTER
213 00BE 8A E0                                     MOV AH,AL ; GET MODEM CONTROL STATUS
214 00C0 42                                     INC DX ; RETURN
215 00C1 EC                                     IN AL,DX
216 00C2 E9 002B R                             JMP A3

```

```

217                                     PAGE
218                                     |-----|
219                                     | WAIT FOR STATUS ROUTINE          |
220 |ENTRY: (BH)= STATUS BIT(S) TO LOOK FOR |
221 |      (DX)= ADDRESS OF STATUS REG      |
222 |EXIT:  ZERO FLAG ON = STATUS FOUND     |
223 |      ZERO FLAG OFF = TIMEOUT.        |
224 |      (AH)= LAST STATUS READ          |
225 |-----|
226
227 00C5                                WAIT_FOR STATUS PROC NEAR
228 00C5 8A 9D 00TC R                   MOV     BL, #RS232_TIM_OUT[D1] ; LOAD OUTER LOOP COUNT
229
230                                     |-----| ADJUST OUTER LOOP COUNT
231
232 00C9 55                                PUSH    BP                    ; SAVE (BP)
233 00CA 53                                PUSH    BX                    ; SAVE (BX)
234 00CB 5D                                POP     BP                    ; USE BP FOR OUTER LOOP COUNT
235 00CC 81 E5 00FF                       AND     BP, 00FFH            ; STRIP HIGH BITS
236 00DD D1 D5                             RCL    BP, 1                  ; MULTIPLY OUTER COUNT BY 4
237 00DE D1 D5                             RCL    BP, 1
238 00D4                                WFS0: SUB     CX, CX
239 00D4 2B C9                             WFS1: IN     AL, DX            ; GET STATUS
240 00D6                                MOV     AH, AL                ; MOVE TO (AH)
241 00D6 EC                                AND     AL, BH                ; ISOLATE BITS TO TEST
242 00D7 8A E0                             CMP     AL, BH                ; EXACTLY = TO MASK
243 00D9 22 C7                             JE     WFS_END                ; RETURN WITH ZERO FLAG ON
244 00DB 3A C7
245 00DD 74 07
246
247 00DF E2 F5                             LOOP   WFS1                    ; TRY AGAIN
248
249 00E1 4D                                DEC     BP
250 00E2 75 F0                             JNZ    WFS0
251
252 00E4 0A FF                             OR     BH, BH                  ; SET ZERO FLAG OFF
253 00E6                                WFS_END: POP    BP             ; RESTORE (BP)
254 00E6 5D
255 00E7 C3
256
257 00E8                                WAIT_FOR_STATUS ENDP
258
259 00E8                                RS232_IO_1 ENDP
260
261 00E8                                CODE ENDS
262                                     END

```

```

1 PAGE 118,121
2 TITLE VIDEO1 --- 06/10/85 VIDEO DISPLAY BIOS
3 .286C
4 .LIST
5 0000 CODE SEGMENT BYTE PUBLIC
6
7 PUBLIC ACT_DISP_PAGE
8 PUBLIC READ_AC_CURRENT
9 PUBLIC READ_CURSOR
10 PUBLIC READ_DOT
11 PUBLIC READ_LPEN
12 PUBLIC SCROLL_DOWN
13 PUBLIC SCROLL_UP
14 PUBLIC SET_COLOR
15 PUBLIC SET_CPOS
16 PUBLIC SET_CTYPE
17 PUBLIC SET_MODE
18 PUBLIC WRITE_C_CURRENT
19 PUBLIC WRITE_C_CURRENT
20 PUBLIC WRITE_DOT
21 PUBLIC WRITE_TTY
22 PUBLIC VIDEO_IO_1
23 PUBLIC VIDEO_STATE
24
25 EXTRN BEEP:NEAR ; SPEAKER BEEP ROUTINE
26 EXTRN CRT_CHAR_GEN:NEAR ; CHARACTER GENERATOR GRAPHICS TABLE
27 EXTRN DDS:NEAR ; LOAD (DS) WITH DATA SEGMENT SELECTOR
28 EXTRN M5:WORD ; REGEN BUFFER LENGTH TABLE
29 EXTRN M6:BYTE ; COLUMNS PER MODE TABLE
30 EXTRN M7:BYTE ; MODE SET VALUE PER MODE TABLE
31
32
33 ----- INT 10 H -----
34 VIDEO_IO ;
35 ; THESE ROUTINES PROVIDE THE CRT DISPLAY INTERFACE ;
36 ; THE FOLLOWING FUNCTIONS ARE PROVIDED: ;
37 ;
38 (AH) = 00H SET MODE (AL) CONTAINS MODE VALUE ;
39 (AL) = 00H 40X25 BW MODE (POWER ON DEFAULT) ;
40 (AL) = 01H 40X25 COLOR ;
41 (AL) = 02H 80X25 BW ;
42 (AL) = 03H 80X25 COLOR ;
43 GRAPHICS MODES ;
44 (AL) = 04H 320X200 COLOR ;
45 (AL) = 05H 320X200 BW MODE ;
46 (AL) = 06H 640X200 BW MODE ;
47 (AL) = 07H 80X25 MONOCHROME (USED INTERNAL TO VIDEO ONLY) ;
48 *** NOTES -BW MODES OPERATE SAME AS COLOR MODES, BUT COLOR ;
49 -CURSOR IS NOT DISPLAYED IN GRAPHICS MODE ;
50 (AH) = 01H SET CURSOR TYPE ;
51 (CH) = BITS #+0 = START LINE FOR CURSOR ;
52 ** HARDWARE WILL ALWAYS CAUSE BLINK ;
53 ** SETTING BIT 5 OR 6 WILL CAUSE ERRATIC BLINKING ;
54 OR NO CURSOR AT ALL ;
55 (CL) = BITS #+0 = END LINE FOR CURSOR ;
56 (AH) = 02H SET CURSOR POSITION ;
57 (DH,DL) = ROW,COLUMN (00H,00H) IS UPPER LEFT ;
58 (BH) = PAGE NUMBER (MUST BE 00H FOR GRAPHICS MODES) ;
59 (AH) = 03H READ CURSOR POSITION ;
60 (BH) = PAGE NUMBER (MUST BE 00H FOR GRAPHICS MODES) ;
61 ON EXIT (DH,DL) = ROW,COLUMN OF CURRENT CURSOR ;
62 (CH,CL) = CURSOR MODE CURRENTLY SET ;
63 (AH) = 04H READ LIGHT PEN POSITION ;
64 ON EXIT: ;
65 (AH) = 00H -- LIGHT PEN SWITCH NOT DOWN/NOT TRIGGERED ;
66 (AH) = 01H -- VALID LIGHT PEN VALUE IN REGISTERS ;
67 (DH,DL) = ROW,COLUMN OF CHARACTER LP POSITION ;
68 (CX) = RASTER LINE (0-199) ;
69 (BX) = PIXEL COLUMN (0-319,639) ;
70 (AH) = 05H SELECT ACTIVE DISPLAY PAGE (VALID ONLY FOR ALPHA MODES) ;
71 (AL) = NEW PAGE VALUE (0-7 FOR MODES 0A1, 0-3 FOR MODES 2A3) ;
72 (AH) = 06H SCROLL ACTIVE PAGE UP ;
73 (AL) = NUMBER OF LINES, (LINES BLANKED AT BOTTOM OF WINDOW) ;
74 (CH,CL) = ROW,COLUMN OF UPPER LEFT CORNER OF SCROLL ;
75 (DH,DL) = ROW,COLUMN OF LOWER RIGHT CORNER OF SCROLL ;
76 (BH) = ATTRIBUTE TO BE USED ON BLANK LINE ;
77 (AH) = 07H SCROLL ACTIVE PAGE DOWN ;
78 (AL) = NUMBER OF LINES, INPUT LINES BLANKED AT TOP OF WINDOW ;
79 (AL) = 00H MEANS BLANK ENTIRE WINDOW ;
80 (CH,CL) = ROW,COLUMN OF UPPER LEFT CORNER OF SCROLL ;
81 (DH,DL) = ROW,COLUMN OF LOWER RIGHT CORNER OF SCROLL ;
82 (BH) = ATTRIBUTE TO BE USED ON BLANK LINE ;
83
84 CHARACTER HANDLING ROUTINES
85
86
87 (AH) = 08H READ ATTRIBUTE/CHARACTER AT CURRENT CURSOR POSITION ;
88 (BH) = DISPLAY PAGE (VALID FOR ALPHA MODES ONLY) ;
89 ON EXIT ;
90 (AL) = CHAR READ ;
91 (AH) = ATTRIBUTE OF CHARACTER READ (ALPHA MODES ONLY) ;
92 (AH) = 09H WRITE ATTRIBUTE/CHARACTER AT CURRENT CURSOR POSITION ;
93 (BH) = DISPLAY PAGE (VALID FOR ALPHA MODES ONLY) ;
94 (CX) = COUNT OF CHARACTERS TO WRITE ;
95 (AL) = CHAR TO WRITE ;
96 (BL) = ATTRIBUTE OF CHARACTER (ALPHA)/COLOR OF CHAR (GRAPHICS) ;
97 ** SEE NOTE ON WRITE DOT FOR BIT 7 OF BL = 1. ;
98 (AH) = 0AH WRITE CHARACTER ONLY AT CURRENT CURSOR POSITION ;
99 (BH) = DISPLAY PAGE (VALID FOR ALPHA MODES ONLY) ;
100 (CX) = COUNT OF CHARACTERS TO WRITE ;
101 (AL) = CHAR TO WRITE ;
102 NOTE: USE FUNCTION (AH) = 09H IN GRAPHICS MODES ;
103 FOR READ/WRITE CHARACTER INTERFACE WHILE IN GRAPHICS MODE, THE ;
104 CHARACTERS ARE FORMED FROM A CHARACTER GENERATOR IMAGE ;
105 MAINTAINED IN THE SYSTEM ROM. ONLY THE 1ST 128 CHARS ;
106 ARE CONTAINED THERE. TO READ/WRITE THE SECOND 128 CHARS, ;
107 THE USER MUST INITIALIZE THE POINTER AT INTERRUPT 1FH ;
108 (LOCATION 007CH) TO POINT TO THE 1K BYTE TABLE CONTAINING ;
109 THE CODE POINTS FOR THE SECOND 128 CHARS (128-255). ;
110 FOR WRITE CHARACTER INTERFACE IN GRAPHICS MODE, THE REPLICATION FACTOR ;
111 CONTAINED IN (CX) ON ENTRY WILL PRODUCE VALID RESULTS ONLY. ;
112 FOR CHARACTERS CONTAINED ON THE SAME ROW, CONTINUATION TO ;
113 SUCCEEDING LINES WILL NOT PRODUCE CORRECTLY. ;
114

```

SECTION 5

```

115      ; GRAPHICS INTERFACE
116      ;
117      ;
118      (AH) = 0BH SET COLOR PALETTE
119      ; (BH) = PALETTE COLOR ID BEING SET (0-127)
120      ; (BL) = COLOR VALUE TO BE USED WITH THAT COLOR ID
121      ; NOTE: FOR THE CURRENT COLOR CARD, THIS ENTRY POINT HAS
122      ; MEANING ONLY FOR 320X200 GRAPHICS.
123      ; COLOR ID = 0 SELECTS THE BACKGROUND COLOR (0-15)
124      ; COLOR ID = 1 SELECTS THE PALETTE TO BE USED:
125      ; 0 = GREEN(1)/RED(2)/YELLOW(3)
126      ; 1 = CYAN(1)/MAGENTA(2)/WHITE(3)
127      ; IN 40X25 OR 80X25 ALPHA MODES, THE VALUE SET FOR
128      ; PALETTE COLOR 0 INDICATES THE BORDER COLOR
129      ; TO BE USED (VALUES 0-31, WHERE 16-31 SELECT
130      ; THE HIGH INTENSITY BACKGROUND SET.
131      ;
132      (AH) = 0CH WRITE DOT
133      ; (DX) = ROW NUMBER
134      ; (CX) = COLUMN NUMBER
135      ; (AL) = COLOR VALUE
136      ; IF BIT 7 OF AL = 1, THEN THE COLOR VALUE IS EXCLUSIVE
137      ; Ored WITH THE CURRENT CONTENTS OF THE DOT
138      ;
139      (AH) = 0DH READ DOT
140      ; (DX) = ROW NUMBER
141      ; (CX) = COLUMN NUMBER
142      ; (AL) RETURNS THE DOT READ
143      ;
144      ASCII TELETYPE ROUTINE FOR OUTPUT
145      ;
146      (AH) = 0EH WRITE TELETYPE TO ACTIVE PAGE
147      ; (AL) = CHAR TO WRITE
148      ; (BL) = FOREGROUND COLOR IN GRAPHICS MODE
149      ; NOTE -- SCREEN WIDTH IS CONTROLLED BY PREVIOUS MODE SET
150      ;
151      (AH) = 0FH CURRENT VIDEO STATE
152      ; RETURNS THE CURRENT VIDEO STATE
153      ; (AL) = MODE CURRENTLY SET ( SEE (AH) = 00H FOR EXPLANATION )
154      ; (AH) = NUMBER OF CHARACTER COLUMNS ON SCREEN
155      ; (BH) = CURRENT ACTIVE DISPLAY PAGE
156      ;
157      (AH) = 10H RESERVED
158      ;
159      (AH) = 11H RESERVED
160      ;
161      (AH) = 12H RESERVED
162      ;
163      (AH) = 13H WRITE STRING
164      ; ES:BP - POINTER TO STRING TO BE WRITTEN
165      ; CX - LENGTH OF CHARACTER STRING TO WRITTEN
166      ; DX - CURSOR POSITION FOR STRING TO BE WRITTEN
167      ; BH - PAGE NUMBER
168      ; (AL) = 00H WRITE CHARACTER STRING
169      ; BL - ATTRIBUTE
170      ; STRING IS <CHAR,CHAR, ... ,CHAR>
171      ; CURSOR IS NOT MOVED
172      ; (AL) = 01H WRITE CHARACTER STRING AND MOVE CURSOR
173      ; BL - ATTRIBUTE
174      ; STRING IS <CHAR,CHAR, ... ,CHAR>
175      ; CURSOR IS MOVED
176      ; (AL) = 02H WRITE CHARACTER AND ATTRIBUTE STRING
177      ; (VALID FOR ALPHA MODES ONLY)
178      ; STRING IS <CHAR,ATTR,CHAR,ATTR ... ,CHAR,ATTR>
179      ; CURSOR IS NOT MOVED
180      ; (AL) = 03H WRITE CHARACTER AND ATTRIBUTE STRING AND MOVE CURSOR
181      ; (VALID FOR ALPHA MODES ONLY)
182      ; STRING IS <CHAR,ATTR,CHAR,ATTR ... ,CHAR,ATTR>
183      ; CURSOR IS MOVED
184      ; NOTE: CARRIAGE RETURN, LINE FEED, BACKSPACE, AND BELL ARE
185      ; TREATED AS COMMANDS RATHER THAN PRINTABLE CHARACTERS.
186      ;
187      ; BX,CX,DX,S1,D1,BP,SP,DS,ES,SS PRESERVED DURING CALLS EXCEPT FOR
188      ; BX,CX,DX RETURN VALUES ON FUNCTIONS 03H,04H,0DH AND 0DH. ON ALL CALLS
189      ; AX IS MODIFIED.
190      ;
191      -----
192      ASSUME CS:CODE,DS:DATA,ES:NOTHING
193      ;
194      M1 OFFSET SET_MODE ; TABLE OF ROUTINES WITHIN VIDEO 1/0
195      DW OFFSET SET_CTYPE
196      DW OFFSET SET_CPOS
197      DW OFFSET READ_CURSOR
198      DW OFFSET ACT_DISP_PAGE
199      DW OFFSET SCROLL_UP
200      DW OFFSET SCROLL_DOWN
201      DW OFFSET READ_AC_CURRENT
202      DW OFFSET WRITE_AC_CURRENT
203      DW OFFSET WRITE_C_CURRENT
204      DW OFFSET SET_COLOR
205      DW OFFSET WRITE_DOT
206      DW OFFSET READ_DOT
207      DW OFFSET WRITE_TTY
208      DW OFFSET VIDEO_STATE
209      DW OFFSET VIDEO_RETURN ; RESERVED
210      DW OFFSET VIDEO_RETURN ; RESERVED
211      DW OFFSET VIDEO_RETURN ; RESERVED
212      DW OFFSET WRITE_STRING ; CASE 19H, WRITE STRING
213      EQU $-M1
214      ;
215      VIDEO_IO_1 PROC NEAR
216      ; ENTRY POINT FOR ORG 0F065H
217      ; INTERRUPTS BACK ON
218      CLD ; SET DIRECTION FORWARD
219      PUSH ES ; SAVE WORK AND PARAMETER REGISTERS
220      PUSH DX
221      PUSH CX
222      PUSH BX
223      PUSH SI
224      PUSH DI
225      PUSH BP
226      CALL DDS ; POINT DS: TO DATA SEGMENT
227      MOV SI,0B800H ; GET SEGMENT FOR COLOR CARD
228      MOV DI,0EQUIP_FLAG ; GET EQUIPMENT FLAGS SETTING
229      AND DI,30H ; ISOLATE CRT SWITCHES
230      CMP DI,30H ; IS SETTING FOR BW CARD?
231      JNE M2 ; SKIP IF NOT BW CARD
232      MOV SI,0B000H ; ELSE GET SEGMENT FOR BW CARD
233      ;
234      M2: CMP AH,13H ; TEST FOR WRITE STRING OPERATION
235      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,B          ; 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          MOV     AH,%CRT_MODE  ; MOVE CURRENT MODE INTO AH
236          JMP     WORD PTR CS:[SI-OFFSET M1] ; GO TO SELECTED FUNCTION
237 005B 8A 26 0049 R    MOV     AH,%CRT_MODE  ; MOVE CURRENT MODE INTO AH
238 005F 2E1 FF A4 0000 R  JMP     WORD PTR CS:[SI-OFFSET M1] ; GO TO SELECTED FUNCTION
239
240 0064          M4:   JMP     VIDEO_RETURN  ; COMMAND NOT VALID
241 0064 E9 012E R      JMP     VIDEO_RETURN  ; DO NOTHING IF NOT IN VALID RANGE
242 0067
243
244 ; SET_MODE
245 ; THIS ROUTINE INITIALIZES THE ATTACHMENT TO
246 ; THE SELECTED MODE. THE SCREEN IS BLANKED.
247 ; INPUT
248 ; (AL) = MODE SELECTED (RANGE 0-7)
249 ; OUTPUT
250 ; NONE
251 ;-----
252 0067          SET_MODE PROC NEAR
253 0067 8A 03D4          MOV     DX,03D4H      ; ADDRESS OF COLOR CARD
254 006A 83 FF 30          CMP     DI,30H        ; IS BW CARD INSTALLED
255 006D 75 04          JNE     M8            ; OK WITH COLOR
256 006F 80 07          MOV     AL,7          ; INDICATE INTERNAL BW CARD MODE
257 0071 82 84          MOV     DL,0B4H       ; ADDRESS OF BW (MONOCHROME) CARD
258 0073          M8:   MOV     %CRT_MODE,AL  ; SAVE MODE IN GLOBAL VARIABLE
259 0073 A2 0049 R      MOV     %ADDR_6845,DX ; SAVE ADDRESS OF BASE
260 0076 8B 16 0063 R    MOV     %ROWS,25-1    ; INITIALIZE DEFAULT ROW COUNT OF 25
261 007A C6 06 0084 R 16  PUSH    DS            ; SAVE POINTER TO DATA SEGMENT
262 007F 1E          DS     ; SAVE MODE NUMBER (AL)
263 0080 50          CDB    AX             ; CLEAR HIGH BYTE OF MODE
264 0081 98          MOV     SI,AX         ; SET TABLE POINTER, INDEXED BY MODE
265 0082 8B F0          MOV     AL,CS:[SI + OFFSET M7] ; GET THE MODE SET VALUE FROM TABLE
266 0085 A2 0065 R      MOV     %CRT_MODE_SET,AL ; SAVE THE MODE SET VALUE
267 008C 24 37          AND    AL,%037H      ; VIDEO OFF. SAVE HIGH RESOLUTION BIT
268 008E 52          PUSH   DX             ; SAVE OUTPUT PORT VALUE
269 008F 83 C2 04       ADD    DX,4           ; POINT TO CONTROL REGISTER
270 0092 EE          OUT    DX,AL         ; RESET VIDEO TO OFF TO SUPPRESS ROLLING
271 0093 5A          POP    DX             ; BACK TO BASE REGISTER
272 0093 5A          POP    DS:IBS0
273          SUB    BX,BX         ; SET UP FOR ABS0 SEGMENT
274 0094 2B DB          MOV     DS,BX        ; ESTABLISH VECTOR TABLE ADDRESSING
275 0096 8E DB          LDS    DS,%PARAM_PTR ; GET POINTER TO VIDEO PARAMS
276 0098 C5 1E 0074 R  ASSUME DS:CODE
277          POP    AX          ; RECOVER MODE NUMBER IN (AL)
278 009C 58          MOV     CX,16        ; LENGTH OF EACH ROW OF TABLE
279 009D 89 0010        CMP     AL,2         ; DETERMINE WHICH ONE TO USE
280 00A0 3C 02          JC     M9            ; MODE IS 0 OR 1
281 00A2 72 0E          ADD    BX,CX        ; NEXT ROW OF INITIALIZATION TABLE
282 00A4 03 D9          CMP     AL,4         ; MODE IS 2 OR 3
283 00A6 3C 04          JC     M9            ; MOVE TO GRAPHICS ROW OF INIT_TABLE
284 00A8 72 08          ADD    BX,CX        ; MODE IS 4,5, OR 6
285 00AA 03 D9          CMP     AL,7         ; MOVE TO BW CARD ROW OF INIT_TABLE
286 00AC 3C 07          JC     M9            ; MODE IS 4,5, OR 6
287 00AE 72 02          ADD    BX,CX        ; MOVE TO BW CARD ROW OF INIT_TABLE
288 00B0 03 D9          ADD    BX,CX
289
290          ;----- BX POINTS TO CORRECT ROW OF INITIALIZATION TABLE
291
292 00B2          M9:   PUSH   AX           ; OUT INIT
293 00B2 50          MOV     AX,[BX+10]   ; SAVE MODE IN (AL)
294 00B3 8B 47 0A          XCHG   AH,AL        ; GET THE CURSOR MODE FROM THE TABLE
295 00B6 8E 0E          PUSH   DS           ; PUT CURSOR MODE IN CORRECT POSITION
296 00B8 1E          DS     ; SAVE TABLE SEGMENT POINTER
297          ASSUME DS:DATA
298 00B9 E8 0000 E      CALL   DDS           ; POINT DS TO DATA SEGMENT
299 00BC A3 0060 R      MOV     %CURSOR_MODE,AX ; PLACE INTO BIOS DATA SAVE AREA
300          ASSUME DS:CODE
301 00BF 1F          POP    DS           ; RESTORE THE TABLE SEGMENT POINTER
302 00C0 32 E4          XOR    AH,AH        ; AH IS REGISTER NUMBER DURING LOOP
303
304          ;----- LOOP THROUGH TABLE, OUTPUTTING REGISTER ADDRESS, THEN VALUE FROM TABLE
305
306 00C2          M10:  MOV     AL,AH        ; INITIALIZATION LOOP
307 00C2 8A C4          OUT    DX,AL        ; GET 6845 REGISTER NUMBER
308 00C4 EE          INC    DX           ; POINT TO DATA PORT
309 00C5 42 04          INC    AH           ; NEXT REGISTER VALUE
310 00C6 FE C4          MOV     AL,[BX]     ; GET TABLE VALUE
311 00C8 8A 07          OUT    DX,AL        ; GET TABLE VALUE
312 00CA EE          INC    BX           ; OUT TO CHIP
313 00CB 43 02          INC    BX           ; NEXT IN TABLE
314 00CC 4A          DEC    DX           ; BACK TO POINTER REGISTER
315 00CD E2 F3          LOOP  M10          ; DO THE WHOLE TABLE
316 00CF 58          POP    AX           ; GET MODE BACK INTO (AL)
317 00D0 1F          POP    DS           ; RECOVER SEGMENT VALUE
318          ASSUME DS:DATA
319
320          ;----- FILL REGEN AREA WITH BLANK
321
322 00D1 33 FF          XOR    DI,D1        ; SET UP POINTER FOR REGEN
323 00D3 89 3E 004E R   MOV     %CRT_START,DI ; START ADDRESS SAVED IN GLOBAL
324 00D7 C6 06 0062 R 00 MOV     %ACTIVE_PAGE,DI ; SET PAGE VALUE
325 00DC B9 2000        MOV     CX,8192      ; NUMBER OF WORDS IN COLOR CARD
326 00DF 3C 04          CMP     AL,4         ; TEST FOR GRAPHICS
327 00E1 72 0A          JC     M12          ; NO GRAPHICS INIT
328 00E3 3C 07          CMP     AL,7         ; TEST FOR BW CARD
329 00E5 74 04          JE     M11          ; BW CARD INIT
330 00E7 33 C0          XOR    AX,AX        ; FILL FOR GRAPHICS MODE
331 00E9 EB 05          JMP     SHORT M13    ; CLEAR BUFFER
332 00EB          M11:  MOV     CH,08H      ; BW CARD INIT
333 00EB B5 08          MOV     CH,08H      ; BUFFER SIZE ON BW CARD (2048)
334 00ED          M12:  MOV     AX,' '*7'H  ; NO GRAPHICS INIT
335 00ED 8B 0720        MOV     AX,' '*7'H  ; FILL CHAR FOR ALPHA + ATTRIBUTE
336 00F0          M13:  REP     STOSW       ; CLEAR BUFFER
337 00F0 F3/ AB        REP     STOSW       ; FILL THE REGEN BUFFER WITH BLANKS
338
339          ;----- ENABLE VIDEO AND CORRECT PORT SETTING
340
341 00F2 8B 16 0063 R  MOV     DX,%ADDR_6845 ; PREPARE TO OUTPUT TO VIDEO ENABLE PORT
342 00F6 83 C2 04       ADD    DX,4           ; POINT TO THE MODE CONTROL REGISTER

```

```

343 00F9 A0 0065 R      MOV     AL,@CRT_MODE_SET      ; GET THE MODE SET VALUE
344 00FC EE             OUT     DX,AL                 ; SET VIDEO ENABLE PORT
345
346 ;----- DETERMINE NUMBER OF COLUMNS, BOTH FOR ENTIRE DISPLAY
347 ;----- AND THE NUMBER TO BE USED FOR TTY INTERFACE
348
349 00FD 2E: 8A 84 0000 E  MOV     AL,CS:[SI + OFFSET M6] ; GET NUMBER OF COLUMNS ON THIS SCREEN
350 0102 98             CBW     ; CLEAR HIGH BYTE
351 0103 A3 004A R      MOV     @CRT_COLS,AX          ; INITIALIZE NUMBER OF COLUMNS COUNT
352
353 ;----- SET CURSOR POSITIONS
354
355 0106 81 E6 000E     AND     SI,000EH             ; WORD OFFSET INTO CLEAR LENGTH TABLE
356 010A 2E: 8B 84 0000 E  MOV     AX,CS:[SI + OFFSET M5] ; LENGTH TO CLEAR
357 010F A3 004C R      MOV     @CRT_LEN,AX          ; SAVE LENGTH OF CRT -- NOT USED FOR BW
358 0112 B9 0008       MOV     CX,8                 ; CLEAR ALL CURSOR POSITIONS
359 0115 BF 0050 R      MOV     DI,OFFSET @CURSOR_POSN
360 0118 1E             PUSH    DS                   ; ESTABLISH SEGMENT
361 0119 07             POP     ES                   ; ADDRESSING
362 011A 33 C0         XOR     AX,AX                ; FILL WITH ZEROES
363 011C F3/ AB        REP     STOSW
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     @CRT_MODE,6         ; SEE IF THE MODE IS 640X200 BW
370 0126 75 02         JNZ     M14                  ; IF NOT 640X200, THEN GO TO REGULAR
371 0128 B0 3F         MOV     AL,3FH              ; IF IT IS 640X200, THEN PUT IN 3FH
372 012A E2             M14:  OUT     DX,AL               ; OUTPUT THE CORRECT VALUE TO 3D9 PORT
373 012A E2             MOV     @CRT_PALETTE,AL     ; SAVE THE VALUE FOR FUTURE USE
374 012B A2 0066 R
375
376 ;----- NORMAL RETURN FROM ALL VIDEO RETURNS
377
378 012E             VIDEO_RETURN:
379 012E 5D             POP     BP
380 012F 5F             POP     DI
381 0130 5E             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 5F             POP     DS
387 0135 07             POP     ES                   ; RECOVER SEGMENTS
388 0136 CF             IRET    ; ALL DONE
389 0137
390 SET_MODE          ENDP
391 ;-----
392 ; SET_CTYPE
393 ; THIS ROUTINE SETS THE CURSOR VALUE
394 ; INPUT
395 ; (CX) HAS CURSOR VALUE CH-START LINE, CL-STOP LINE
396 ; OUTPUT
397 ; NONE
398 ;-----
399 0137             PROC    NEAR
400 0139 89 0E 0060 R   MOV     @CURSOR_MODE,CX     ; 6845 REGISTER FOR CURSOR SET
401 013D EB 0142 R     CALL    M16                  ; SAVE IN DATA AREA
402 0140 EB EC         JMP     VIDEO_RETURN        ; OUTPUT CX REGISTER
403
404 ;----- THIS ROUTINE OUTPUTS THE CX REGISTER TO THE 6845 REGISTERS NAMED IN (AH)
405
406 M16:
407 0142             MOV     DX,@ADDR_6845      ; ADDRESS REGISTER
408 0142 8B 16 0063 R   MOV     AL,AH               ; GET VALUE
409 0146 8A C4         OUT     DX,AL               ; REGISTER SET
410 0148 EE           INC     DX                   ; DATA REGISTER
411 0149 42           INC     DX                   ; I/O DELAY
412 014A EB 00         JMP     $+2                  ; DATA
413 014C 8B C5         MOV     AL,CH               ; DATA
414 014E EE           OUT     DX,AL               ; DATA
415 014F 4A           DEC     DX                   ; DATA
416 0150 8A C4         MOV     AL,AH               ; POINT TO OTHER DATA REGISTER
417 0152 FE C0         INC     AL                   ; SET FOR SECOND REGISTER
418 0154 EE           OUT     DX,AL               ; REGISTER SET
419 0155 42           INC     DX                   ; I/O DELAY
420 0156 EB 00         JMP     $+2                  ; SECOND DATA VALUE
421 0158 8A C1         MOV     AL,CL               ; SECOND DATA VALUE
422 015A EE           OUT     DX,AL               ; REGISTER SET
423 015B C3           RET     ; ALL DONE
424 015C
425 SET_CTYPE        ENDP
426 ;-----
427 ; SET_CPOS
428 ; THIS ROUTINE SETS THE CURRENT CURSOR POSITION TO THE
429 ; NEW X-Y VALUES PASSED
430 ; INPUT
431 ; DX - ROW,COLUMN OF NEW CURSOR
432 ; BH - DISPLAY PAGE OF CURSOR
433 ; OUTPUT
434 ; CURSOR IS SET AT 6845 IF DISPLAY PAGE IS CURRENT DISPLAY
435 ;-----
436 015C             SET_CPOS:  PROC    NEAR
437 015C 8A C7         MOV     AL,BH               ; MOVE PAGE NUMBER TO WORK REGISTER
438 015E 98           CBW     ; CONVERT PAGE TO WORD VALUE
439 015F D1 E0         SAL     AX,1                ; WORD OFFSET
440 0161 96           XCHG   AX,SI                ; USE INDEX REGISTER
441 0162 89 94 0050 R   MOV     [SI+OFFSET @CURSOR_POSN],DX ; SAVE THE POINTER
442 0166 38 3E 0062 R   CMP     @ACTIVE_PAGE,BH     ; SET_CPOS RETURN
443 016A 75 05         JNZ     M17                  ; GET ROW/COLUMN TO AX
444 016C 8B C2         MOV     AX,DX               ; CURSOR SET
445 016E EB 0173 R     CALL    M18                  ; CURSOR SET
446 0171             M17:  JMP     VIDEO_RETURN        ; SET_CPOS_RETURN
447 0171 EB BB
448 0173             SET_CPOS:  ENDP
449
450 ;----- SET CURSOR POSITION, AX HAS ROW/COLUMN FOR CURSOR
451
452 0173             M18:  PROC    NEAR
453 0173 E8 01F7 R     CALL    POSITION             ; DETERMINE LOCATION IN REGEN BUFFER
454 0176 8B C8         MOV     CX,AX               ; GET ROW/COLUMN TO AX
455 0178 03 0E 004E R   ADD     CX,@CRT_START      ; ADD IN THE START ADDRESS FOR THIS PAGE
456 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 E8 0142 R      CALL   M16           ; OUTPUT THE VALUE TO THE 6845
459 0183 C3             RET
460 0184                M18
461                ENDP
462                -----
463                ; READ_CURSOR
464                ; THIS ROUTINE READS THE CURRENT CURSOR VALUE FROM THE
465                ; 6845, FORMATS IT, AND SENDS IT BACK TO THE CALLER
466                ; INPUT
467                ; BH - PAGE OF CURSOR
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    BL,BH
473 0186 32 FF        XOR    BH,BH
474 0188 D1 E3       SAL    BX,1          ; WORD OFFSET
475 018A 8B 97 0050 R MOV    DX,[BX+OFFSET @CURSOR_POSN]
476 018E 8B 0E 0060 R MOV    CX,@CURSOR_MODE
477 0192 5D          POP    BP
478 0193 5F          POP    D1
479 0194 5E          POP    SI
480 0195 5B          POP    BX
481 0196 58          POP    AX          ; DISCARD SAVED CX AND DX
482 0197 58          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    RACTIVE_PAGE,AL ; SAVE ACTIVE PAGE VALUE
498 019E 8B 0E 004C R MOV    CX,@CRT_LEN     ; GET SAVED LENGTH OF REGEN BUFFER
499 01A2 98          PUSHW AX              ; CONVERT AL TO WORD
500 01A3 50          PUSH  AX              ; SAVE PAGE VALUE
501 01A4 F7 E1      MUL    CX              ; DISPLAY PAGE TIMES REGEN LENGTH
502 01A6 A3 004E R   MOV    @CRT_START,AX  ; SAVE START ADDRESS FOR LATER
503 01A9 8B C8      MOV    CX,AX          ; START ADDRESS TO CX
504 01AB D1 F9      SAR    CX,1           ; DIVIDE BY 2 FOR 6845 HANDLING
505 01AD B4 0C      MOV    AH,12          ; 6845 REGISTER FOR START ADDRESS
506 01AF E8 0142 R   CALL   M16
507 01B2 5B          POP    BX              ; RECOVER PAGE VALUE
508 01B3 D1 E3       SAL    BX,1          ; *2 FOR WORD OFFSET
509 01B5 8B 87 0050 R MOV    AX,[BX + OFFSET @CURSOR_POSN] ; GET CURSOR FOR THIS PAGE
510 01B9 E8 0173 R   CALL   M18           ; SET THE CURSOR POSITION
511 01BC E9 012E R   JMP    VIDEO_RETURN
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 (0-31)
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 8B 8B 16 0063 R MOV    DX,@ADDR_6845  ; I/O PORT FOR PALETTE
531 01C3 83 C2 05     ADD    DX,5           ; OVERSCAN PORT
532 01C6 A0 0066 R   MOV    AL,@CRT_PALETTE ; GET THE CURRENT PALETTE VALUE
533 01C9 0A FF       OR     BH,BH          ; IS THIS COLOR 0?
534 01CB 75 0E       JNZ   M20            ; OUTPUT COLOR 1
535                -----
536                ;----- HANDLE COLOR 0 BY SETTING THE BACKGROUND COLOR
537                AND    AL,0E0H          ; TURN OFF LOW 5 BITS OF CURRENT
538 01CD 24 E0       AND    BL,01FH       ; TEST THE LOW ORDER BIT OF INPUT VALUE
539 01CF 80 E3 IF    OR    AL,BL         ; PUT VALUE INTO REGISTER
540 01D2 0A C3       OR    AL,BL         ; OUTPUT THE PALETTE
541 01D4             M19: OUT    DX,AL         ; OUTPUT COLOR SELECTION TO 3D9 PORT
542 01D4 EE         MOV    @CRT_PALETTE,AL ; SAVE THE COLOR VALUE
543 01D5 A2 0066 R   JMP    VIDEO_RETURN
544 01D8 E9 012E R   ;----- HANDLE COLOR 1 BY SELECTING THE PALETTE TO BE USED
545                -----
546                M20: AND    AL,0DFH       ; TURN OFF PALETTE SELECT BIT
547                SHR    BL,1           ; TEST THE LOW ORDER BIT OF BL
548 01DB 24 DF       JNC   M19           ; ALREADY DONE
549 01DB 24 DF       OR    AL,20H        ; TURN ON PALETTE SELECT BIT
550 01DD D0 EB       JMP   M19           ; GO DO IT
551 01DF 73 F3       SET_COLOR  ENDP
552                -----
553                ; VIDEO STATE
554                ; RETURNS THE CURRENT VIDEO STATE IN AX
555                ; AH = NUMBER OF COLUMNS ON THE SCREEN
556                ; AL = CURRENT VIDEO MODE
557                ; BH = CURRENT ACTIVE PAGE
558                -----
559 01E5                VIDEO_STATE  PROC   NEAR
560 01E5 8A 26 004A R MOV    AH,BYTE PTR @CRT_COLS ; GET NUMBER OF COLUMNS
561 01E9 A0 04 9 R   MOV    AL,@CRT_MODE     ; CURRENT MODE
562 01E5             MOV    BH,@ACTIVE_PAGE  ; GET CURRENT ACTIVE PAGE
563 01EC 8A 3E 0062 R MOV    BH,@ACTIVE_PAGE
564 01F0 5D          POP    BP              ; RECOVER REGISTERS
565 01F1 5F          POP    D1
566 01F2 5E          POP    SI
567 01F3 59          POP    CX              ; DISCARD SAVED BX
568 01F4 E9 0132 R   JMP    M15            ; RETURN TO CALLER

```

```

571 01F7          VIDEO_STATE  ENDP
572             -----
573             : POSITION
574             : THIS SERVICE ROUTINE CALCULATES THE REGEN BUFFER ADDRESS
575             : OF A CHARACTER IN THE ALPHA MODE
576             : INPUT
577             : AX = ROW, COLUMN POSITION
578             : OUTPUT
579             : AX = OFFSET OF CHAR POSITION IN REGEN BUFFER
580             -----
581 01F7          POSITION        PROC   NEAR
582 01F7 53      PUSH          BX          ; SAVE REGISTER
583 01F8 8B D8   MOV           BX,AX
584 01FA 8A C4   MOV           AL,AH          ; ROWS TO AL
585 01FC F6 26 004A R  MUL          BYTE PTR @CRT_COLS ; DETERMINE BYTES TO ROW
586 0200 32 FF   XOR           BH,BH
587 0202 03 C3   ADD          AX,BX          ; ADD IN COLUMN VALUE
588 0204 D1 E0   SAL          AX,1          ; * 2 FOR ATTRIBUTE BYTES
589 0206 5B     POP           BX
590 0207 C3     RET
591 0208          POSITION        ENDP
592             -----
593             : SCROLL_UP
594             : THIS ROUTINE MOVES A BLOCK OF CHARACTERS UP
595             : ON THE SCREEN
596             : INPUT
597             : (AH) = CURRENT CRT MODE
598             : (AL) = NUMBER OF ROWS TO SCROLL
599             : (CX) = ROW/COLUMN OF UPPER LEFT CORNER
600             : (DX) = ROW/COLUMN OF LOWER RIGHT CORNER
601             : (BH) = ATTRIBUTE TO BE USED ON BLANKED LINE
602             : (DS) = DATA SEGMENT
603             : (ES) = REGEN BUFFER SEGMENT
604             : OUTPUT
605             : NONE -- THE REGEN BUFFER IS MODIFIED
606             -----
607             ASSUME  DS:DATA,ES:DATA
608 0208          SCROLL_UP     PROC   NEAR
609
610 0208 E8 02E4 R CALL    TEST_LINE_COUNT
611 020B 80 FC 04  CMP    AH,4          ; TEST FOR GRAPHICS MODE
612 020E 72 08     JC     N1            ; HANDLE SEPARATELY
613 0210 80 FC 07  CMP    AH,7          ; TEST FOR BW CARD
614 0213 74 03     JE     N1
615 0215 E9 04A3 R JMP    GRAPHICS_UP
616 0218
617 0218 53      PUSH          BX          ; UP CONTINUE
618 0219 8B C1     MOV    AX,CX          ; SAVE FILL ATTRIBUTE IN BH
619 021B EB E255 R CALL    SCROLL_POSITION
620 021E 74 31     JC     N1            ; UPPER LEFT POSITION
621 0220 03 F0     ADD    SI,AX          ; DO SETUP FOR SCROLL
622 0222 8A E6     MOV    AH,DH          ; BLANK FIELD
623 0224 2A E3     SUB    AH,BL          ; FROM ADDRESS
624 0226
625 0226 E8 0297 R CALL    N10
626 0229 03 F5     ADD    SI,BP          ; # ROWS TO BE MOVED
627 022B 03 FD     ADD    DI,BP          ; ROW LOOP
628 022D FE CC     DEC    AH            ; MOVE ONE ROW
629 022F 75 F5     JNZ   N2            ; POINT TO NEXT LINE IN BLOCK
630 0231
631 0231 58      POP          AX          ; COUNT OF LINES TO MOVE
632 0232 80 20     CMP    AL,' '        ; ROW_LOOP
633 0234          CALL    N11          ; CLEAR ENTRY
634 0234 E8 0240 R CALL    N11          ; RECOVER ATTRIBUTE IN AH
635 0237 03 FD     ADD    DI,BP          ; FILL WITH BLANKS
636 0239 FE CB     DEC    BL            ; CLEAR_LOOP
637 023B 75 F7     JNZ   N4            ; CLEAR THE ROW
638 023D
639 023D E8 0000 E CALL    DDS
640 0240 80 3E 0049 R 07 CMP    @CRT_MODE,7   ; IS THIS THE BLACK AND WHITE CARD
641 0245 74 07     JE     N6            ; IF SO, SKIP THE MODE RESET
642 0247 A0 0065 R MOV    AL,@CRT_MODE_SET ; GET THE VALUE OF THE MODE SET
643 024A BA 03D8 R MOV    DX,03D8H      ; ALWAYS SET COLOR CARD PORT
644 024D EE     OUT    DX,AL
645 024E
646 024E E9 012E R JMP    VIDEO_RETURN ; VIDEO_RET_HERE
647 0251
648 0251 8A DE     MOV    BL,DH          ; BLANK FIELD
649 0253 EB DC     JMP    N3            ; GET ROW COUNT
650 0255          SCROLL_UP     ENDP
651
652             :----- HANDLE COMMON SCROLL SET UP HERE
653
654 0255          SCROLL_POSITION PROC   NEAR
655 0255 E8 01F7 R CALL    POSITION
656 0258 03 06 004E R MOV    AX,@CRT_START ; CONVERT TO REGEN POINTER
657 025C 8B F8     MOV    DI,AX          ; OFFSET OF ACTIVE PAGE
658 025E 8B F0     MOV    SI,AX          ; TO ADDRESS FOR SCROLL
659 0260 2B D1     SUB    DX,CX          ; FROM ADDRESS FOR SCROLL
660 0262 FE C6     INC    DH            ; DX = #ROWS, #COLS IN BLOCK
661 0264 FE C2     DEC    DL            ; INCREMENT FOR 0 ORIGIN
662 0266 32 ED     XOR    CH,CH          ; SET HIGH BYTE OF COUNT TO ZERO
663 0268 8B 2E 004A R MOV    BP,@CRT_COLS  ; GET NUMBER OF COLUMNS IN DISPLAY
664 026C 03 ED     ADD    BP,BP          ; GET BYTE OF ATTRIBUTE
665 026E 8A C3     MOV    AL,BL          ; TIMES 2 FOR ATTRIBUTE BYTE
666 0270 F6 26 004A R MUL   BYTE PTR @CRT_COLS ; DETERMINE OFFSET TO FROM ADDRESS
667 0274 03 C0     ADD    AX,AX          ; *2 FOR ATTRIBUTE BYTE
668 0276 50     PUSH    AX            ; SAVE LINE COUNT
669 0277 A0 0049 R MOV    AL,@CRT_MODE  ; GET CURRENT MODE
670 027A 06     PUSH    ES            ; ESTABLISH ADDRESSING TO REGEN BUFFER
671 027B 1F     POP     DS            ; FOR BOTH POINTERS
672 027C 3C 02     CMP    AL,2          ; TEST FOR COLOR CARD SPECIAL CASES HERE
673 027E 72 13     JB     N9            ; HAVE TO HANDLE 80X25 SEPARATELY
674 0280 3C 03     CMP    AL,3
675 0282 77 0F     JA     N9
676
677 0284 52     PUSH    DX            ; 80X25 COLOR CARD SCROLL
678 0285 BA 03DA H MOV    DX,3DAH
679 0288
680 0288 EC     MOV    DI,3DAH      ; GUARANTEED TO BE COLOR CARD HERE
681 0289 A8 08     TEST   AL,DX         ; WAIT DISP_ENABLE
682 028B 74 FB     JZ     N8            ; GET PORT
683 028D B0 25     MOV    AL,25H        ; WAIT FOR VERTICAL RETRACE
684 028F B2 D8     MOV    DL,0D8H      ; WAIT_DISP_ENABLE
685
686             : ADDRESS CONTROL PORT

```



```

685 0291 EE          OUT  DX,AL          ; TURN OFF VIDEO DURING VERTICAL RETRACE
686 0292 5A          POP  DX
N9:
687 0293             POP  AX          ; RESTORE LINE COUNT
688 0293 58          OR   BL,BL          ; 0 SCROLL MEANS BLANK FIELD
689 0294 0A DB       RET             ; RETURN WITH FLAGS SET
690 0296 C3          SCROLL_POSITION ENDP
691 0297
692
693
694 0297             I----- MOVE_ROW
N10:
695 0297 8A CA       PROC  NEAR
696 0299 56          MOV  CL,DL          ; GET # OF COLS TO MOVE
697 029A 57          PUSH SI
698 029B F3/ A5     REP  MOVSW         ; SAVE START ADDRESS
699 029D 5F          POP  DI          ; MOVE THAT LINE ON SCREEN
700 029E 5E          POP  SI          ; RECOVER ADDRESSES
701 029F C3          RET
702 02A0
703
704
705 02A0             I----- CLEAR_ROW
N11:
706 02A0 8A CA       PROC  NEAR
707 02A2 57          MOV  DI          ; GET # COLUMNS TO CLEAR
708 02A3 F3/ AB     REP  STOSW        ; STORE THE FILL CHARACTER
709 02A5 5F          POP  DI
710 02A6 C3          RET
711 02A7
N11:
712 ENDP
712
713             I----- SCROLL_DOWN
714             ; THIS ROUTINE MOVES THE CHARACTERS WITHIN A DEFINED
715             ; BLOCK DOWN ON THE SCREEN, FILLING THE TOP LINES
716             ; WITH A DEFINED CHARACTER
717             ; INPUT
718             ; (AH) = CURRENT CRT MODE
719             ; (AL) = NUMBER OF LINES TO SCROLL
720             ; (CX) = UPPER LEFT CORNER OF REGION
721             ; (DX) = LOWER RIGHT CORNER OF REGION
722             ; (BH) = FILL CHARACTER
723             ; (DS) = DATA SEGMENT
724             ; (ES) = REGEN SEGMENT
725             ; OUTPUT
726             ; NONE -- SCREEN IS SCROLLED
727             I-----
728 02A7             SCROLL_DOWN  PROC  NEAR
729 02A7 FD          STD
730 02A8 E8 02E4 R    CALL TEST_LINE_COUNT ; DIRECTION FOR SCROLL DOWN
731 02AB 80 FC 04     CMP  AH,4          ; TEST FOR GRAPHICS
732 02AC 72 08       JC   N12           ; TEST FOR BW CARD
733 02B0 80 FC 07     CMP  AH,7
734 02B3 74 03       JE   N12
735 02B5 E9 04FA R   JMP  GRAPHICS_DOWN
N12:
736 02B8             ; CONTINUE_DOWN
737 02B8 53          PUSH BX          ; SAVE ATTRIBUTE IN BH
738 02B9 8B C2       MOV  AX,DX        ; LOWER RIGHT CORNER
739 02BB E8 0255 R   CALL SCROLL_POSITION ; GET REGEN LOCATION
740 02BE 74 20       JC   N16
741 02C0 2B F0       SUB  SI,AX        ; SI IS FROM ADDRESS
742 02C2 8A E6       MOV  AH,DX        ; GET TOTAL # ROWS
743 02C4 2A E3       SUB  AH,BL        ; GET TO MOVE IN SCROLL
N13:
744 02C6             ; MOVE ONE ROW
745 02C6 E8 0297 R   CALL N10
746 02C9 2B F5       SUB  SI,BP
747 02CB 2B FD       SUB  DI,BP
748 02CD FE CC       DEC  AH
749 02CF 75 F5       JNZ  N13
N14:
750 02D1             ; RECOVER ATTRIBUTE IN AH
751 02D1 58          POP  AX
752 02D2 80 20       MOV  AL,' '
N15:
753 02D4             ; CLEAR ONE ROW
754 02D4 E8 02A0 R   CALL N11          ; GO TO NEXT ROW
755 02D7 2B FD       SUB  DI,BP
756 02D9 FE CB       DEC  BL
757 02DB 75 F7       JNZ  N15
N16:
758 02DD E9 023D R   JMP  N5          ; SCROLL_END
759 02E0
760 02E0 8A DE       MOV  BL,DH
761 02E2 EB ED       JMP  N14
762 02E4             SCROLL_DOWN  ENDP
763
764             I----- IF AMOUNT OF LINES TO BE SCROLLED = AMOUNT OF LINES IN WINDOW
765             I----- THEN ADJUST AL; ELSE RETURN;
766
767 02E4             TEST_LINE_COUNT PROC  NEAR
768
769 02E4 8A DB       MOV  BL,AL        ; SAVE LINE COUNT IN BL
770 02E6 0A C0       OR   AL,AL        ; TEST IF AL IS ALREADY ZERO
771 02E8 74 0E       JC   BL_SET       ; IF IT IS THEN RETURN...
772 02EA 50          PUSH AX          ; SAVE AX
773 02EB 8A C6       MOV  AL,CH        ; SUBTRACT LOWER ROW FROM UPPER ROW
774 02ED 2A C5       SUB  AL,CH
775 02EF FE C0       INC  AL          ; ADJUST DIFFERENCE BY 1
776 02F1 3A C3       CMP  AL,BL        ; LINE COUNT = AMOUNT OF ROWS IN WINDOW?
777 02F3 58          POP  AX          ; RESTORE AX
778 02F4 75 02       JNE  BL_SET       ; IF NOT THEN WE'RE ALL SET
779 02F6 2A DB       SUB  BL,BL        ; OTHERWISE SET BL TO ZERO
BL_SET:
780 02F8             RET
781 02F8 C3          TEST_LINE_COUNT ENDP
782 02F9
    
```

```

783 PAGE
784 ; READ_AC_CURRENT -----
785 ; THIS ROUTINE READS THE ATTRIBUTE AND CHARACTER AT THE CURRENT
786 ; CURSOR POSITION AND RETURNS THEM TO THE CALLER
787 ; INPUT
788 ; (AH) = CURRENT CRT MODE
789 ; (BH) = DISPLAY PAGE ( ALPHA MODES ONLY )
790 ; (DS) = DATA SEGMENT
791 ; (ES) = REGEN SEGMENT
792 ; OUTPUT
793 ; (AL) = CHARACTER READ
794 ; (AH) = ATTRIBUTE READ
795 ;-----
796 ASSUME DS:DATA,ES:DATA
797
798
799 02F9 READ_AC_CURRENT PROC NEAR
800 02F9 80 FC 04 CMP AH,4 ; IS THIS GRAPHICS
801 02FC T2 08 JC P10
802
803 02FE 80 FC 07 CMP AH,7 ; IS THIS BW CARD
804 0301 T4 03 JE P10
805
806 0303 E9 062A R JMP GRAPHICS_READ
807 0306
808 0306 EB 0322 R P10: CALL FIND_POSITION ; READ AC_CONTINUE
809 0309 BB F7 MOV SI,DI ; GET REGEN LOCATION AND PORT ADDRESS
810 030B 06 PUSH ES ; ESTABLISH ADDRESSING IN SI
811 030C IF POP DS ; GET REGEN SEGMENT FOR QUICK ACCESS
812
813 ;----- WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
814
815 030D 0A DB OR BL,BL ; CHECK MODE FLAG FOR COLOR CARD IN 80
816 030F T5 0D JNZ P11 ; ELSE SKIP RETRACE WAIT -- DO FAST READ
817 0311 ; WAIT FOR HORIZ RETRACE LOW OR VERTICAL
818 0311 FB STI ; ENABLE INTERRUPTS FIRST
819 0312 90 NOP ; ALLOW FOR SMALL INTERRUPT WINDOW
820 0313 FA CLI ; BLOCK INTERRUPTS FOR SINGLE LOOP
821 0314 EC IN AL,DX ; GET STATUS FROM THE ADAPTER
822 0315 AB 01 TEST AL,RHRZ ; IS HORIZONTAL RETRACE LOW
823 0317 T5 F8 JNZ P11 ; WAIT UNTIL IT IS
824 0319 ; NOW WAIT FOR EITHER RETRACE HIGH
825 0319 EC IN AL,DX ; GET STATUS
826 031A AB 09 TEST AL,RVRT+RHRZ ; IS HORIZONTAL OR VERTICAL RETRACE HIGH
827 031C T4 FB JZ P12 ; WAIT UNTIL EITHER IS ACTIVE
828 031E
829 031E AD P13: LODSW ; GET THE CHARACTER AND ATTRIBUTE
830 031F E9 012E R JMP VIDEO_RETURN ; EXIT WITH (AX)
831
832 0322 READ_AC_CURRENT ENDP
833
834
835 0322 FIND_POSITION PROC NEAR
836 0322 86 E3 XCHG AH,BL ; SETUP FOR BUFFER READ OR WRITE
837 0324 8B E8 MOV BP,AX ; SWAP MODE TYPE WITH ATTRIBUTE
838 0326 80 EB 02 SUB BL,2 ; SAVE CHARACTER/ATTR IN (BP) REGISTER
839 0329 D0 EB 02 SHR BL,1 ; CONVERT DISPLAY MODE TYPE TO A
840 032B BB F3 MOV SI,BX ; ZERO VALUE FOR COLOR IN 80 COLUMN
841 032D 8A DF MOV BL,BH ; AND SAVE (2 OR 3 --> ZERO)
842 032F 32 FF XOR BH,BH ; MOVE DISPLAY PAGE TO LOW BYTE
843 0331 8B FB MOV DI,BX ; CLEAR HIGH BYTE OF COUNT/BYTE OFFSET
844 0333 D1 E7 SAL DI,1 ; MOVE DISPLAY PAGE (COUNT) TO WORK REG
845 0335 8B 85 0050 R MOV AX,[DI+OFFSET*CURSOR_POSN] ; TIMES 2 FOR WORD OFFSET
846 0339 T4 09 JZ P21 ; GET ROW/COLUMN OF THAT PAGE
847 ; SKIP BUFFER ADJUSTMENT IF PAGE ZERO
848 033B 33 FF XOR DI,DI ; ELSE SET BUFFER START ADDRESS TO ZERO
849 033D
850 033D 03 3E 004C R P20: ADD DI,@CRT_LEN ; ADD LENGTH OF BUFFER FOR ONE PAGE
851 0341 4B DEC BX ; DECREMENT PAGE COUNT
852 0342 T5 F9 JNZ P20 ; LOOP TILL PAGE COUNT EXHAUSTED
853 0344
854 0344 EB 01F7 R P21: CALL POSITION ; DETERMINE LOCATION IN REGEN IN PAGE
855 0347 03 F8 ADD DI,AX ; ADD LOCATION TO START OF REGEN PAGE
856 0349 8B 16 0063 R MOV DX,@ADDR_6845 ; GET BASE ADDRESS OF ACTIVE DISPLAY
857 034D 83 C6 ADD DX,6 ; POINT AT STATUS PORT
858 0350 8B DE MOV BX,SI ; RECOVER CONVERTED MODE TYPE IN (BL)
859 0352 C3 RET ; BP= ATTRIBUTE/CHARACTER (FROM BL/AL)
860 ; DI= POSITION (OFFSET IN REGEN BUFFER)
861 ; DX= STATUS PORT ADDRESS OF ADAPTER
862 0353 FIND_POSITION ENDP ; BL= MODE FLAG (ZERO FOR 80X25 COLOR)

```

```

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 WRITE_AC_CURRENT PROC NEAR
882 CMP AH,4 ; IS THIS GRAPHICS
883 JC P30
884 CMP AH,7 ; IS THIS BW CARD
885 JMP GRAPHICS_WRITE
886
887 P30: CALL FIND_POSITION ; WRITE AC CONTINUE
888 ; GET REGEN LOCATION AND PORT ADDRESS
889 ; ADDRESS IN (DI) REGISTER
890 OR BL,BL ; CHECK MODE FLAG FOR COLOR CARD AT 80
891 JZ ; SKIP TO RETRACE WAIT IF COLOR AT 80
892 XCHG AX,BP ; GET THE ATTR/CHAR SAVED FOR FAST WRITE
893 REP STOSW ; STRING WRITE THE ATTRIBUTE & CHARACTER
894 JMP SHORT P35 ; EXIT FAST WRITE ROUTINE
895
896 |-----|
897 |-----|
898 | WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
899 |
900 P31: XCHG BP,AX ; LOOP FOR EACH ATTR/CHAR WRITE
901 ; PLACE ATTR/CHAR BACK IN SAVE REGISTER
902 P32: STI ; ENABLE INTERRUPTS FIRST
903 ; ALLOW FOR INTERRUPT WINDOW
904 CL1 ; BLOCK INTERRUPTS FOR SINGLE LOOP
905 IN AL,DX ; GET STATUS FROM THE ADAPTER
906 TEST AL,RVRT ; CHECK FOR VERTICAL RETRACE FIRST
907 JNZ P34 ; DO FAST WRITE NOW IF VERTICAL RETRACE
908 TEST AL,RHRZ ; IS HORIZONTAL RETRACE LOW THEN
909 JNZ P32 ; WAIT UNTIL IT IS
910 ; WAIT FOR EITHER RETRACE HIGH
911 IN AL,DX ; GET STATUS AGAIN
912 TEST AL,RVRT+RHRZ ; IS HORIZONTAL OR VERTICAL RETRACE HIGH
913 JZ P33 ; WAIT UNTIL EITHER IS ACTIVE
914
915 P34: XCHG AX,BP ; GET THE ATTR/CHAR SAVED IN (BP)
916 STOSW ; WRITE THE ATTRIBUTE AND CHARACTER
917 LOOP P31 ; AS MANY TIMES AS REQUESTED - TILL CX=0
918
919 P35: JMP VIDEO_RETURN ; EXIT
920
921 WRITE_AC_CURRENT ENDP
922
923 |-----|
924 | WRITE_C_CURRENT
925 | THIS ROUTINE WRITES THE CHARACTER AT
926 | THE CURRENT CURSOR POSITION, ATTRIBUTE UNCHANGED
927 |
928 | INPUT
929 | (AH) = CURRENT CRT MODE
930 | (BH) = DISPLAY PAGE
931 | (CX) = COUNT OF CHARACTERS TO WRITE
932 | (AL) = CHAR TO WRITE
933 | (DS) = DATA SEGMENT
934 | (ES) = REGEN SEGMENT
935 |
936 | OUTPUT
937 | DISPLAY REGEN BUFFER UPDATED
938 |-----|
939 WRITE_C_CURRENT PROC NEAR
940 CMP AH,4 ; IS THIS GRAPHICS
941 JC P40
942 CMP AH,7 ; IS THIS BW CARD
943 JMP GRAPHICS_WRITE
944
945 P40: CALL FIND_POSITION ; GET REGEN LOCATION AND PORT ADDRESS
946 ; ADDRESS OF LOCATION IN (DI)
947
948 |-----|
949 |-----|
950 | WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
951 |
952 P41: STI ; WAIT FOR HORZ RETRACE LOW OR VERTICAL
953 ; ENABLE INTERRUPTS FIRST
954 OR BL,BL ; CHECK MODE FLAG FOR COLOR CARD IN 80
955 JNZ P43 ; ELSE SKIP RETRACE WAIT - DO FAST WRITE
956 CLI ; BLOCK INTERRUPTS FOR SINGLE LOOP
957 IN AL,DX ; GET STATUS FROM THE ADAPTER
958 TEST AL,RVRT ; CHECK FOR VERTICAL RETRACE FIRST
959 JNZ P42 ; DO FAST WRITE NOW IF VERTICAL RETRACE
960 TEST AL,RHRZ ; IS HORIZONTAL RETRACE LOW THEN
961 JNZ P41 ; WAIT UNTIL IT IS
962 ; WAIT FOR EITHER RETRACE HIGH
963 IN AL,DX ; GET STATUS AGAIN
964 TEST AL,RVRT+RHRZ ; IS HORIZONTAL OR VERTICAL RETRACE HIGH
965 JZ P43 ; WAIT UNTIL EITHER RETRACE ACTIVE
966
967 P43: MOV AX,BP ; GET THE CHARACTER SAVE IN (BP)
968 STOSB ; PUT THE CHARACTER INTO REGEN BUFFER
969 INC DI ; BUMP POINTER PAST ATTRIBUTE
970 LOOP P41 ; AS MANY TIMES AS REQUESTED
971
972 JMP VIDEO_RETURN
973
974 WRITE_C_CURRENT ENDP

```

```

972                                     PAGE
973 -----
974 | WRITE_STRING                        |
975 | THIS ROUTINE WRITES A STRING OF   |
976 | CHARACTERS TO THE CRT.            |
977 | INPUT                               |
978 | (AL) = WRITE STRING COMMAND 0 - 3 |
979 | (BH) = DISPLAY PAGE                |
980 | (CX) = COUNT OF CHARACTERS TO    |
981 | WRITE, IF (CX) = 0 THEN RETURN    |
982 | (DX) = CURSOR POSITION FOR START   |
983 | OF STRING WRITE                   |
984 | (BL) = ATTRIBUTE OF CHARACTER    |
985 | TO WRITE IF (AL) = 0 OR (AL) = 1 |
986 | (ESI) = SOURCE STRING SEGMENT    |
987 | (BP) = SOURCE STRING OFFSET      |
988 | OUTPUT                             |
989 | NONE                              |
990 -----
991
992 03B2 WRITE_STRING PROC NEAR
993 03B2 3C 04 CMP AL,04 ; TEST FOR INVALID WRITE STRING OPTION
994 03B4 73 7C JNB P59 ; IF OPTION INVALID THEN RETURN
995 03B6 E3 7A JCXZ P59 ; IF ZERO LENGTH STRING THEN RETURN
996
997 03B8 8B F3 MOV SI,BX ; GET CURRENT CURSOR PAGE
998 03BA C1 EE 08 SHR SI,8 ; CLEAR HIGH BYTE
999 03BC D1 E8 08 SAL SI,1 ; CONVERT TO PAGE OFFSET (SI= PAGE)
1000 03BE FF B4 0050 R PUSH [SI+OFFSET *CURSOR_POSN]; SAVE CURRENT CURSOR POSITION IN STACK
1001 03C0 50 00 PUSH AX ; SAVE WRITE STRING OPTION
1002 03C2 B8 0200 MOV AX,0200H ; SET NEW CURSOR POSITION
1003 03C4 CD 10 INT 10H ; RESTORE WRITE STRING OPTION
1004 03C6 58 POP AX
1005
1006 03CA P50: PUSH CX
1007 03CB 51 PUSH BX
1008 03CD 53 PUSH AX
1009 03CE 50 XCHG AH,AL ; PUT THE WRITE STRING OPTION INTO (AH)
1010 03D0 86 E0 MOV AL,ES:[BP] ; GET CHARACTER FROM INPUT STRING
1011 03D2 26 8A 46 00 INC BP ; BUMP POINTER TO CHARACTER
1012 03D4 45
1013 0007 ;----- TEST FOR SPECIAL CHARACTER'S
1014 0008
1015 0009
1016 03DA 3C 08 CMP AL,08H ; IS IT A BACKSPACE
1017 03DC 74 0C JE P51 ; BACK SPACE
1018 03DE 3C 0D CMP AL,CR ; IS IT CARRIAGE RETURN
1019 03E0 74 08 JE P51 ; CAR_RET
1020 03E2 3C 0A CMP AL,LF ; IS IT A LINE FEED
1021 03E4 74 04 JE P51 ; LINE FEED
1022 03E6 3C 07 CMP AL,07H ; IS IT A BELL
1023 03E8 74 0D JNE P52 ; IF NOT THEN DO WRITE CHARACTER
1024 03EA
1025 03EC B4 0E MOV AH,0EH ; TTY_CHARACTER WRITE
1026 03EE 10 INT 10H ; WRITE TTY CHARACTER TO THE CRT
1027 03F0 B9 0001 MOV DX,[SI+OFFSET *CURSOR_POSN] ; GET CURRENT CURSOR POSITION
1028 03F2 58 POP AX ; RESTORE REGISTERS
1029 03F4 58 POP BX
1030 03F6 59 POP CX
1031 03F8 EB 2E JMP SHORT P54 ; GO SET CURSOR POSITION AND CONTINUE
1032 03FA
1033 03FC P52: MOV CX,1 ; SET CHARACTER WRITE AMOUNT TO ONE
1034 03FE 0A CMP AH,2 ; IS THE ATTRIBUTE IN THE STRING
1035 0400 72 05 JB P53 ; IF NOT THEN SKIP
1036 0402 26 8A 5E 00 MOV BL,ES:[BP] ; ELSE GET NEW ATTRIBUTE
1037 0404 45 INC BP ; BUMP STRING POINTER
1038 0406
1039 0408 P53: MOV AH,09H ; GOT CHARACTER
1040 040A CD 10 INT 10H ; WRITE CHARACTER TO THE CRT
1041 040C 58 POP AX ; RESTORE REGISTERS
1042 040E 5B POP BX
1043 0410 58 POP CX
1044 0412 FE C2 INC DL ; INCREMENT COLUMN COUNTER
1045 0414 3A 16 004A R CMP DL,BYTE PTR *CRT_COLS ; IF COLS ARE WITHIN RANGE FOR THIS MODE
1046 0416 72 12 JB P54 ; THEN GO TO COLUMNS SET
1047 0418 FE C6 INC DH ; BUMP ROW COUNTER BY ONE
1048 041A 2A D2 SUB DL,DL ; SET COLUMN COUNTER TO ZERO
1049 041C 80 FE 19 CMP DH,25 ; IF ROWS ARE LESS THAN 25 THEN
1050 041E 72 09 JB P54 ; GO TO ROWS_COLUMNS_SET
1051 0420
1052 0422 P54: PUSH AX ; ELSE SCROLL SCREEN
1053 0424 50 MOV AX,0E0AH ; DO SCROLL ONE LINE
1054 0426 CD 10 INT 10H ; RESET ROW COUNTER TO 24
1055 0428 FE CE DEC DH
1056 042A 58 POP AX ; RESTORE REGISTERS
1057 042C P54: POP AX ; ROW COLUMNS SET
1058 042E 50 MOV AX,0200H ; SAVE WRITE STRING OPTION
1059 0430 CD 10 INT 10H ; SET NEW CURSOR POSITION COMMAND
1060 0432 58 POP AX ; ESTABLISH NEW CURSOR POSITION
1061 0434 E2 A2 LOOP P50 ; DO IT ONCE MORE UNTIL (CX) = ZERO
1062 0436
1063 0438 P59: POP DX ; RESTORE OLD CURSOR COORDINATES
1064 043A 5A TEST AL,01H ; IF CURSOR WAS NOT TO BE MOVED THEN
1065 043C 75 05 JNZ P59 ; THEN EXIT WITHOUT RESETTING OLD VALUE
1066 043E B8 0200 MOV AX,0200H ; ELSE RESTORE OLD CURSOR POSITION
1067 0440 CD 10 INT 10H
1068 0442 432 ; DONE - EXIT WRITE STRING
1069 0444 E9 012E R JMP VIDEO_RETURN ; RETURN TO CALLER
1070 0446
1071 0448 WRITE_STRING ENDP

```

```

1067 PAGE
1068 ;-----
1069 ; READ DOT -- WRITE DOT
1070 ; THESE ROUTINES WILL WRITE A DOT, OR READ THE
1071 ; DOT AT THE INDICATED LOCATION
1072 ; ENTRY --
1073 ; DX = ROW (0-199) (THE ACTUAL VALUE DEPENDS ON THE MODE)
1074 ; CX = COLUMN (0-639) (THE VALUES ARE IN RANGE CHECKED)
1075 ; AL = DOT VALUE TO WRITE (1,2 OR 4 BITS DEPENDING ON MODE,
1076 ; REQUIRED FOR WRITE DOT ONLY, RIGHT JUSTIFIED)
1077 ; BIT 7 OF AL = 1 INDICATES XOR THE VALUE INTO THE LOCATION
1078 ; DS = DATA SEGMENT
1079 ; ES = REGEN SEGMENT
1080 ; EXIT
1081 ; AL = DOT VALUE READ, RIGHT JUSTIFIED, READ ONLY
1082 ;-----
1083 ASSUME DS:DATA,ES:DATA
1084
1085 0435 READ_DOT PROC NEAR
1086 0435 E8 0469 R CALL R3 ; DETERMINE BYTE POSITION OF DOT
1087 043B 26: 8A 04 MOV AL,ES:[SI] ; GET THE BYTE
1088 043B 22 C4 AND AL,AH ; MASK OFF THE OTHER BITS IN THE BYTE
1089 043D D2 E0 SHL AL,CL ; LEFT JUSTIFY THE VALUE
1090 043F 8A CE MOV CL,DL ; GET NUMBER OF BITS IN RESULT
1091 0441 D2 C0 ROL AL,CH ; RIGHT JUSTIFY THE RESULT
1092 0443 E9 012E R JMP VIDEO_RETURN ; RETURN FROM VIDEO I/O
1093 0446 ENDP
1094
1095 0446 WRITE_DOT PROC NEAR
1096 0446 50 PUSH AX ; SAVE DOT VALUE
1097 0447 50 PUSH AX ; TWICE
1098 0448 E8 0469 R CALL R3 ; DETERMINE BYTE POSITION OF THE DOT
1099 044B D2 EB SHR AL,CL ; SHIFT TO SET UP THE BITS FOR OUTPUT
1100 044D 22 C4 AND AL,AH ; STRIP OFF THE OTHER BITS
1101 044F 26: 8A 0C MOV CL,ES:[SI] ; GET THE CURRENT BYTE
1102 0452 5B POP BX ; RECOVER XOR FLAG
1103 0453 F6 C3 80 TEST BL,80H ; IS IT ON
1104 0456 75 0D JNZ R2 ; YES, XOR THE DOT
1105 0458 F6 D4 NOT AH ; SET MASK TO REMOVE THE INDICATED BITS
1106 045A 22 CC AND CL,AH ; OR IN THE NEW VALUE OF THOSE BITS
1107 045C 0A C1 OR AL,CL
1108 045E R1: MOV ES:[SI],AL ; RESTORE THE BYTE IN MEMORY
1109 045E 26: 8B 04 POP AX
1110 0461 58 JMP VIDEO_RETURN ; RETURN FROM VIDEO I/O
1111 0462 E9 012E R R2: XOR AL,CL ; EXCLUSIVE OR THE DOTS
1112 0465 32 C1 AND AL,CL ; FINISH UP THE WRITING
1113 0465 32 C1 XOR AL,CL
1114 0467 EB F5 JMP R1
1115 0469 ENDP
1116 ;-----
1117 ; THIS SUBROUTINE DETERMINES THE REGEN BYTE LOCATION OF THE
1118 ; INDICATED ROW COLUMN VALUE IN GRAPHICS MODE.
1119 ; ENTRY --
1120 ; DX = ROW VALUE (0-199)
1121 ; CX = COLUMN VALUE (0-639)
1122 ; EXIT --
1123 ; SI = OFFSET INTO REGEN BUFFER FOR BYTE OF INTEREST
1124 ; AH = MASK TO STRIP OFF THE BITS OF INTEREST
1125 ; CL = BITS TO SHIFT TO RIGHT JUSTIFY THE MASK IN AH
1126 ; DH = # BITS IN RESULT
1127 ; BX = MODIFIED
1128 ;-----
1129 0469 R3 PROC NEAR
1130 ;-----
1131 ;----- DETERMINE 1ST BYTE IN INDICATED ROW BY MULTIPLYING ROW VALUE BY 40
1132 ;----- ( LOW BIT OF ROW DETERMINES EVEN/ODD, 80 BYTES/ROW )
1133
1134 0469 93 XCHG AX,BX ; WILL SAVE AL AND AH DURING OPERATION
1135 046A B0 28 MOV AL,40
1136 046C F6 E2 MUL DL ; AX= ADDRESS OF START OF INDICATED ROW
1137 046E A8 08 TEST AL,008H ; TEST FOR EVEN/ODD ROW CALCULATED
1138 0470 74 03 JZ R4 ; JUMP IF EVEN ROW
1139 0472 05 1FDB ADD AX,2000H-40 ; OFFSET TO LOCATION OF ODD ROWS ADJUST
1140 0475 R4: EVEN ROW
1141 0475 96 XCHG SI,AX ; MOVE POINTER TO SI
1142 0476 93 XCHG AX,BX ; RECOVER AL AND AH VALUES
1143 0477 8B D1 MOV DX,CX ; COLUMN VALUE TO DX
1144
1145 ;----- DETERMINE GRAPHICS MODE CURRENTLY IN EFFECT
1146
1147 ; SET UP THE REGISTERS ACCORDING TO THE MODE
1148 ; CH = MASK FOR LOW OF COLUMN ADDRESS ( 7/3 FOR HIGH/MED RES )
1149 ; CL = # OF ADDRESS BITS IN COLUMN VALUE ( 3/2 FOR H/M )
1150 ; BL = MASK TO SELECT BITS FROM POINTED BYTE ( 80H/COH FOR H/M )
1151 ; BH = NUMBER OF VALID BITS IN POINTED BYTE ( 1/2 FOR H/M )
1152
1153 0479 BB 02C0 MOV BX,2C0H
1154 047C B9 0302 MOV CX,302H ; SET PARMS FOR MED RES
1155 047F 80 3E 0049 R 06 CMP @CRT_MODE,6
1156 0484 72 06 JC R5 ; HANDLE IF MED RES
1157 0486 BB 0180 MOV BX,180H
1158 0489 B9 0703 MOV CX,703H ; SET PARMS FOR HIGH RES
1159
1160 ;----- DETERMINE BIT OFFSET IN BYTE FROM COLUMN MASK
1161 048C R5: AND CH,DL ; ADDRESS OF PEL WITHIN BYTE TO CH
1162 048C 22 EA
1163
1164 ;----- DETERMINE BYTE OFFSET FOR THIS LOCATION IN COLUMN
1165
1166 048E D3 EA SHR DX,CL ; SHIFT BY CORRECT AMOUNT
1167 0490 03 F2 ADD SI,DX ; INCREMENT THE POINTER
1168 0492 8A F7 MOV DH,BH ; GET THE # OF BITS IN RESULT TO DH
1169
1170 ;----- MULTIPLY BH (VALID BITS IN BYTE) BY CH (BIT OFFSET)
1171
1172 0494 2A C9 SUB CL,CL ; ZERO INTO STORAGE LOCATION
1173 0496
1174 0496 D0 C8 ROR AL,1 ; LEFT JUSTIFY VALUE IN AL (FOR WRITE)
1175 0498 02 CD ADD CL,CH ; ADD IN THE BIT OFFSET VALUE
1176 049A FE CF DEC BH ; LOOP CONTROL
1177 049C 75 F8 JNZ R6 ; ON EXIT, CL HAS COUNT TO RESTORE BITS
1178 049E 8A E3 MOV AH,BL ; GET MASK TO AH
1179 04A0 D2 EC SHR AH,CL ; MOVE THE MASK TO CORRECT LOCATION
1180 04A2 C3 RET ; RETURN WITH EVERYTHING SET UP

```

SECTION 5

```

1181 04A3          R3      ENDP
1182          -----
1183          | SCROLL UP
1184          | THIS ROUTINE SCROLLS UP THE INFORMATION ON THE CRT
1185          | ENTRY --
1186          | CH,CL = UPPER LEFT CORNER OF REGION TO SCROLL
1187          | DH,DL = LOWER RIGHT CORNER OF REGION TO SCROLL
1188          | BOTH OF THE ABOVE ARE IN CHARACTER POSITIONS
1189          | BH = FILL VALUE FOR BLANKED LINES
1190          | AL = # LINES TO SCROLL (AL=0 MEANS BLANK THE ENTIRE FIELD)
1191          | DS = DATA SEGMENT
1192          | ES = REGEN SEGMENT
1193          | EXIT --
1194          | NOTHING, THE SCREEN IS SCROLLED
1195          -----
1196 04A3          GRAPHICS_UP  PROC      NEAR
1197 04A3 8A D8      MOV     BL,AL          ; SAVE LINE COUNT IN BL
1198 04A5 8B C1      MOV     AX,CX          ; GET UPPER LEFT POSITION INTO AX REG
1199
1200          |-----
1201          |----- USE CHARACTER SUBROUTINE FOR POSITIONING
1202          |----- ADDRESS RETURNED IS MULTIPLIED BY 2 FROM CORRECT VALUE
1203 04A7 E8 06D8 R  CALL   GRAPH_POSN
1204 04AA 8B F8      MOV     DI,AX          ; SAVE RESULT AS DESTINATION ADDRESS
1205
1206          |----- DETERMINE SIZE OF WINDOW
1207
1208 04AC 2B D1      SUB     DX,CX
1209 04AE 81 C2 0101 ADD    DX,101H        ; ADJUST VALUES
1210 04B2 C0 E6 02  SAL    DH,2          ; MULTIPLY ROWS BY 4 AT 8 VERT DOTS/CHAR
1211                                     ; AND EVEN/ODD ROWS
1212          |----- DETERMINE CRT MODE
1213
1214 04B5 80 3E 0049 R 06 CMP    CRT_MODE,6    ; TEST FOR MEDIUM RES
1215 04BA 73 04      JNC    RT             ; FIND_SOURCE
1216
1217          |----- MEDIUM RES UP
1218 04BC D0 E2      SAL    DL,1          ; # COLUMNS * 2, SINCE 2 BYTES/CHAR
1219 04BE D1 E7      SAL    DI,1          ; OFFSET *2 SINCE 2 BYTES/CHAR
1220
1221          |----- DETERMINE THE SOURCE ADDRESS IN THE BUFFER
1222 04C0          RT:          ; FIND_SOURCE
1223 04C0 06          PUSH   ES            ; GET SEGMENTS BOTH POINTING TO REGEN
1224 04C1 1F          POP    DS
1225 04C2 2A ED      SUB    CH,CH         ; ZERO TO HIGH OF COUNT REGISTER
1226 04C4 C0 E3 02  SAL    BL,2          ; MULTIPLY NUMBER OF LINES BY 4
1227 04C7 74 2D      JZ     R11           ; IF ZERO, THEN BLANK ENTIRE FIELD
1228 04C9 8A C3      MOV    AL,BL         ; GET NUMBER OF LINES IN AL
1229 04CB B4 50      MOV    AH,80         ; 80 BYTES/ROW
1230 04CD F6 E4      MUL    DI             ; DETERMINE OFFSET TO SOURCE
1231 04CF 8B F7      MOV    SI,DI        ; SET UP SOURCE
1232 04D1 03 F0      ADD    SI,AX         ; ADD IN OFFSET TO IT
1233 04D3 8A E6      MOV    AH,DH        ; NUMBER OF ROWS IN FIELD
1234 04D5 2A E3      SUB    AH,BL        ; DETERMINE NUMBER TO MOVE
1235
1236          |----- LOOP THROUGH, MOVING ONE ROW AT A TIME, BOTH EVEN AND ODD FIELDS
1237 04D7          R8:          ; ROW LOOP
1238 04D7 E8 0558 R  CALL   R17           ; MOVE ONE ROW
1239 04DA 81 EE 1FB0 SUB    SI,2000H-80   ; MOVE TO NEXT ROW
1240 04DE 81 EF 1FB0 SUB    DI,2000H-80
1241 04E2 FE CC      DEC    AH            ; NUMBER OF ROWS TO MOVE
1242 04E4 75 F1      JNZ   R8            ; CONTINUE TILL ALL MOVED
1243
1244          |----- FILL IN THE VACATED LINE(S)
1245 04E6          R9:          ; CLEAR ENTRY
1246 04E6 8A C7      MOV    AL,BH        ; ATTRIBUTE TO FILL WITH
1247 04E8          R10:         ; CLEAR THAT ROW
1248 04E8 E8 0571 R  CALL   R18           ; POINT TO NEXT LINE
1249 04EB 81 EF 1FB0 SUB    DI,2000H-80   ; NUMBER OF LINES TO FILL
1250 04EF FE CB      DEC    BL           ; CLEAR LOOP
1251 04F1 75 F5      JNZ   R10          ; EVERYTHING DONE
1252 04F3 E9 012E R  JMP    VIDEO_RETURN
1253
1254 04F6          R11:         ; BLANK FIELD
1255 04F6 8A DE      MOV    BL,DH        ; SET BLANK COUNT TO EVERYTHING IN FIELD
1256 04F8 EB EC      JMP    R9           ; CLEAR THE FIELD
1257 04FA          GRAPHICS_UP  ENDP
1258          -----
1259          | SCROLL DOWN
1260          | THIS ROUTINE SCROLLS DOWN THE INFORMATION ON THE CRT
1261          | ENTRY --
1262          | CH,CL = UPPER LEFT CORNER OF REGION TO SCROLL
1263          | DH,DL = LOWER RIGHT CORNER OF REGION TO SCROLL
1264          | BOTH OF THE ABOVE ARE IN CHARACTER POSITIONS
1265          | BH = FILL VALUE FOR BLANKED LINES
1266          | AL = # LINES TO SCROLL (AL=0 MEANS BLANK THE ENTIRE FIELD)
1267          | DS = DATA SEGMENT
1268          | ES = REGEN SEGMENT
1269          | EXIT --
1270          | NOTHING, THE SCREEN IS SCROLLED
1271          -----
1272
1273 04FA          GRAPHICS_DOWN PROC      NEAR
1274 04FA FD          STD
1275 04FB 8A D8      MOV    BL,AL        ; SET DIRECTION
1276 04FD 8B C2      MOV    AX,DX        ; GET LOWER RIGHT POSITION INTO AX REG
1277
1278          |-----
1279          |----- USE CHARACTER SUBROUTINE FOR POSITIONING
1280          |----- ADDRESS RETURNED IS MULTIPLIED BY 2 FROM CORRECT VALUE
1281 04FF E8 06D8 R  CALL   GRAPH_POSN
1282 0502 8B F8      MOV    DI,AX        ; SAVE RESULT AS DESTINATION ADDRESS
1283
1284          |----- DETERMINE SIZE OF WINDOW
1285
1286 0504 2B D1      SUB    DX,CX
1287 0506 81 C2 0101 ADD    DX,101H        ; ADJUST VALUES
1288 050A C0 E6 02  SAL    DH,2          ; MULTIPLY ROWS BY 4 AT 8 VERT DOTS/CHAR
1289                                     ; AND EVEN/ODD ROWS
1290          |----- DETERMINE CRT MODE
1291
1292 050D 80 3E 0049 R 06 CMP    CRT_MODE,6    ; TEST FOR MEDIUM RES
1293 0512 73 05      JNC    R12          ; FIND_SOURCE_DOWN
1294

```

```

1295          ;----- MEDIUM RES DOWN
1296 0514 D0 E2      SAL    DL,1          ; # COLUMNS * 2, SINCE 2 BYTES/CHAR
1297 0516 D1 E7      SAL    DI,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:          ; FIND_SOURCE_DOWN
1302 0519 06         PUSH   ES          ; BOTH SEGMENTS TO REGEN
1303 051A 1F         POP    DS          ;
1304 051B 2A ED      SUB    CH,CH          ; ZERO TO HIGH OF COUNT REGISTER
1305 051D 81 C7 00F0 ADD    DI,240         ; POINT TO LAST ROW OF PIXELS
1306 0521 C0 E3 02  SAL    BL,2          ; MULTIPLY NUMBER OF LINES BY 4
1307 0524 74 2E     JZ    R16          ; IF ZERO, THEN BLANK ENTIRE FIELD
1308 0526 8A C3     MOV    AL,BL          ; GET NUMBER OF LINES IN AL
1309 0528 B4 50     MOV    AH,80         ; 80 BYTES/ROW
1310 052A F6 E4     MUL    AH            ; DETERMINE OFFSET TO SOURCE
1311 052C 8B F7     MOV    SI,DI         ; SET UP SOURCE
1312 052E 2B F0     SUB    SI,AX         ; SUBTRACT THE OFFSET
1313 0530 8A E6     MOV    AH,DI         ; NUMBER OF ROWS IN FIELD
1314 0532 2A E3     SUB    AH,BL         ; DETERMINE NUMBER TO MOVE
1315
1316          ;----- LOOP THROUGH, MOVING ONE ROW AT A TIME, BOTH EVEN AND ODD FIELDS
1317 0534          R13:          ; ROW_LOOP_DOWN
1318 0534 E8 0558 R   CALL   R17          ; MOVE ONE ROW
1319 0537 81 EF 2050 SUB    SI,2000H+80   ; MOVE TO NEXT ROW
1320 053B 81 EF 2050 SUB    DI,2000H+80   ;
1321 053F FE CC     DEC    AH            ; NUMBER OF ROWS TO MOVE
1322 0541 75 F1     JNZ   R13          ; CONTINUE TILL ALL MOVED
1323
1324          ;----- FILL IN THE VACATED LINE(S)
1325 0543          R14:          ; CLEAR_ENTRY_DOWN
1326 0543 8A C7     MOV    AL,BH         ; ATTRIBUTE TO FILL WITH
1327 0545          R15:          ; CLEAR_LOOP_DOWN
1328 0545 E8 0571 R   CALL   R18          ; CLEAR_A_ROW
1329 0548 81 EF 2050 SUB    DI,2000H+80   ; POINT TO NEXT LINE
1330 054C FE CB     DEC    BL            ; NUMBER OF LINES TO FILL
1331 054E 75 F5     JNZ   R15          ; CLEAR_LOOP_DOWN
1332 0550 FC 02     CLD    ; RESET THE DIRECTION FLAG
1333 0551 E9 012E R   JMP    VIDEO_RETURN ; EVERYTHING DONE
1334
1335 0554          R16:          ; BLANK_FIELD_DOWN
1336 0554 8A DE     MOV    BL,DH         ; SET BLANK COUNT TO EVERYTHING IN FIELD
1337 0556 EB EB     JMP    R14          ; CLEAR THE FIELD
1338 0558          GRAPHICS_DOWN ENDP
1339
1340          ;----- ROUTINE TO MOVE ONE ROW OF INFORMATION
1341
1342 0558          R17          PROC NEAR
1343 0558 8A CA     MOV    CL,DL         ; NUMBER OF BYTES IN THE ROW
1344 055A 56         PUSH   SI            ;
1345 055B 57         PUSH   DI            ; SAVE POINTERS
1346 055C F3/ A4    REP    MOVSB         ; MOVE THE EVEN FIELD
1347 055E 5F         POP    DI            ;
1348 055F 5E         POP    SI            ;
1349 0560 81 C6 2000 ADD    SI,2000H      ; POINT TO THE ODD FIELD
1350 0564 81 C7 2000 ADD    DI,2000H      ;
1351 0566 56         PUSH   SI            ;
1352 0569 57         PUSH   DI            ; SAVE THE POINTERS
1353 056A 8A CA     MOV    CL,DL         ; COUNT BACK
1354 056C F3/ A4    REP    MOVSB         ; MOVE THE ODD FIELD
1355 056E 5F         POP    DI            ;
1356 056F 5E         POP    SI            ; POINTERS BACK
1357 0570 C3         RET                ; RETURN TO CALLER
1358 0571          R17          ENDP
1359
1360          ;----- CLEAR A SINGLE ROW
1361
1362 0571          R18          PROC NEAR
1363 0571 8A CA     MOV    CL,DL         ; NUMBER OF BYTES IN FIELD
1364 0573 57         PUSH   DI            ; SAVE POINTER
1365 0574 F3/ AA    REP    STOSB        ; STORE THE NEW VALUE
1366 0576 5F         POP    DI            ; POINTER BACK
1367 0577 81 C7 2000 ADD    DI,2000H      ; POINT TO ODD FIELD
1368 057B 57         PUSH   DI            ;
1369 057C 8A CA     MOV    CL,DL         ; FILL THE ODD FIELD
1370 057E F3/ AA    REP    STOSB        ;
1371 0580 5F         POP    DI            ;
1372 0581 C3         RET                ; RETURN TO CALLER
1373 0582          R18          ENDP
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          ; IF BIT 7 IS SET, THE CHAR IS XOR'D INTO THE REGEN BUFFER
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 128 CHARS. TO ACCESS CHARS IN THE SECOND HALF, THE USER
1400          ; MUST INITIALIZE THE VECTOR AT INTERRUPT# 19 (LOCATION 00D0H) TO
1401          ; POINT TO THE USER SUPPLIED TABLE OF GRAPHIC IMAGES (8X8 BOXES).
1402          ; FAILURE TO DO SO WILL CAUSE UNSTRANGE RESULTS
1403          ;-----
1404          ASSUME DS:DATA,ES:DATA
1405 0582          GRAPHICS_WRITE PROC NEAR
1406 0582 B4 00     MOV    AH,0          ; ZERO TO HIGH OF CODE POINT
1407 0584 50         PUSH   AX            ; SAVE CODE POINT VALUE
1408

```

```

1409          ;----- DETERMINE POSITION IN REGEN BUFFER TO PUT CODE POINTS
1410
1411 0585 E8 06D5 R      CALL S26          ; FIND LOCATION IN REGEN BUFFER
1412 0588 B8 F8         MOV D1,AX          ; REGEN POINTER IN D1
1413
1414          ;----- DETERMINE REGION TO GET CODE POINTS FROM
1415
1416 058A 58           POP AX            ; RECOVER CODE POINT
1417 058B 3C 80       CMP AL,80H       ; IS IT IN SECOND HALF
1418 058D 73 06       JAE S1           ; YES
1419
1420          ;----- IMAGE IS IN FIRST HALF, CONTAINED IN ROM
1421
1422 058F BE 0000 E     MOV S1,OFFSET CRT_CHAR_GEN ; OFFSET OF IMAGES
1423 0592 0E           PUSH CS          ; SAVE SEGMENT ON STACK
1424 0593 EB 0F       JMP SHORT S2     ; DETERMINE_MODE
1425
1426          ;----- IMAGE IS IN SECOND HALF, IN USER MEMORY
1427
1428 0595          S1:          ; EXTEND CHAR
1429 0595 2C 80       SUB AL,80H      ; ZERO ORIGIN FOR SECOND HALF
1430 0597 1E           PUSH DS         ; SAVE DATA POINTER
1431 0598 2B F6       SUB SI,S1       ;
1432 059A BE DE       MOV D5,S1      ; ESTABLISH VECTOR ADDRESSING
1433              ASSUME DS:ABS0
1434 059C C5 36 007C R LDS S1,TEXT_PTR ; GET THE OFFSET OF THE TABLE
1435 05A0 8C DA       MOV DX,DS      ; GET THE SEGMENT OF THE TABLE
1436              ASSUME DS:DATA
1437 05A2 1F         POP D5         ; RECOVER DATA SEGMENT
1438 05A3 52         PUSH DX        ; SAVE TABLE SEGMENT ON STACK
1439
1440          ;----- DETERMINE GRAPHICS MODE IN OPERATION
1441
1442 05A4          S2:          ; DETERMINE_MODE
1443 05A4 C1 E0 03     SAL AX,3        ; MULTIPLY CODE POINT VALUE BY 8
1444 05A7 03 F0       ADD S1,AX       ; S1 HAS OFFSET OF DESIRED CODES
1445 05A9 80 3E 0049 R CMP CRT_MODE,6
1446 05AE 1F         POP DS         ;
1447 05AF 7C 2C       JC S7          ; RECOVER TABLE POINTER SEGMENT
1448              ; TEST FOR MEDIUM RESOLUTION MODE
1449
1450          ;----- HIGH RESOLUTION MODE
1451          S3:          ; HIGH_CHAR
1452 05B1 57         PUSH D1        ; SAVE REGEN POINTER
1453 05B2 56         PUSH S1        ; SAVE CODE POINTER
1454 05B3 B6 04       MOV DH,4       ; NUMBER OF TIMES THROUGH LOOP
1455          S4:          ; GET BYTE FROM CODE POINTS
1456 05B5 AC         LODSB         ; SHOULD WE USE THE FUNCTION
1457 05B6 F6 C3 80   TEST BL,80H    ; TO PUT CHAR IN
1458 05B8 AA         JNZ S5        ; STORE IN REGEN BUFFER
1459 05B9 56         STOSB
1460 05BA AC         LODSB
1461          S5:          ; STORE IN SECOND HALF
1462 05BD 26 88 85 IFF MOV ES:[D1+2000H-1],AL
1463 05C2 83 CT 4F   ADD D1,79      ; MOVE TO NEXT ROW IN REGEN
1464 05C5 FE CE     DEC DH         ; DONE WITH LOOP
1465 05C7 75 EC     JNZ S4        ;
1466 05C9 9E       POP SI        ;
1467 05CA 5F       POP D1        ; RECOVER REGEN POINTER
1468 05CB 47       INC D1        ; POINT TO NEXT CHAR POSITION
1469 05CC E2 E3     LODP S3       ; MORE CHARS TO WRITE
1470 05CE E9 012E R  JMP VIDEO_RETURN
1471 05D1
1472 05D1 26 32 05   XOR AL,ES:[D1] ; EXCLUSIVE OR WITH CURRENT
1473 05D4 AA         STOSB         ; STORE THE CODE POINT
1474 05D5 AC         LODSB        ; AGAIN FOR ODD FIELD
1475 05D6 26 32 85 IFF XOR AL,ES:[D1+2000H-1]
1476 05DB EB E0     JMP S5        ; BACK TO MAINSTREAM
1477
1478          ;----- MEDIUM RESOLUTION WRITE
1479          S7:          ; MED_RES_WRITE
1480 05DD 8A D3       MOV DL,BL     ; SAVE HIGH COLOR BIT
1481 05DF D1 E7     SAL D1,1     ; OFFSET*2 SINCE 2 BYTES/CHAR
1482              ; EXPAND BL TO FULL WORD OF COLOR
1483 05E1 80 E3 03   AND BL,3     ; ISOLATE THE COLOR BITS ( LOW 2 BITS )
1484 05E4 80 55     MOV AL,055H  ; GET BIT CONVERSION MULTIPLIER
1485 05E6 F6 E3     MUL BL,AL    ; EXPAND 2 COLOR BITS TO 4 REPLICATIONS
1486 05E8 8A D8     MOV BL,AL    ; PLACE BACK IN WORK REGISTER
1487 05EA 8A F8     MOV BH,AL    ; EXPAND 2 COLOR BITS TO 4 REPLICATIONS OF COLOR BITS
1488 05EC          S8:          ; MED_CHAR
1489 05EC 57         PUSH D1        ; SAVE REGEN POINTER
1490 05ED 56         PUSH S1        ; SAVE THE CODE POINTER
1491 05EE B6 04       MOV DH,4       ; NUMBER OF LOOPS
1492 05F0          S9:          ; GET CODE POINT
1493 05F0 AC         LODSB         ; GET CODE POINT
1494 05F1 EB 06AD R  CALL S21      ; DOUBLE UP ALL THE BITS
1495 05F4 23 C3     AND AX,BX    ; CONVERT TO FOREGROUND COLOR ( 0 BACK )
1496 05F6 86 E0     XCHG AH,AL   ; SWAP HIGH/LOW BYTES FOR WORD MOVE
1497 05F8 F6 C2 80   TEST DL,80H  ; IS THIS XOR FUNCTION
1498 05FB 74 03     JZ S10       ; NO, JUST STORE IT IN AS IT IS
1499 05FD 26 33 05   XOR AX,ES:[D1] ; DO FUNCTION WITH LOW/HIGH
1500 0600          ; STORE FIRST BYTE HIGH, SECOND LOW
1501 0600 26 89 05   MOV ES:[D1],AX ; GET CODE POINT
1502 0603 AC         LODSB
1503 0604 EB 06AD R  CALL S21
1504 0607 23 C3     AND AX,BX    ; CONVERT TO COLOR
1505 0609 86 E0     XCHG AH,AL   ; SWAP HIGH/LOW BYTES FOR WORD MOVE
1506 060B F6 C2 80   TEST DL,80H  ; AGAIN, IS THIS XOR FUNCTION
1507 060E 74 05     JZ S11       ; NO, JUST STORE THE VALUES
1508 0610 26 33 85 2000 XOR AX,ES:[D1+2000H] ; FUNCTION WITH FIRST HALF LOW
1509 0615          S11:         ; STORE SECOND PORTION HIGH
1510 0615 26 89 85 2000 MOV ES:[D1+2000H],AX ; POINT TO NEXT LOCATION
1511 061A 83 CT 50   ADD D1,80    ;
1512 061D FE CE     DEC DH       ;
1513 061F 75 CF     JNZ S9       ; KEEP GOING
1514 0621 9E       POP SI       ; RECOVER CODE POINTER
1515 0622 5F       POP D1       ; RECOVER REGEN POINTER
1516 0623 47       INC D1       ; POINT TO NEXT CHAR POSITION
1517 0624 47       INC D1       ;
1518 0625 E2 C5     LOOP SB      ; MORE TO WRITE
1519 0627 E9 012E R  JMP VIDEO_RETURN
1520 062A          GRAPHICS_WRITE ENDP
1521          ;-----
1522          ; GRAPHICS READ
    
```



```

1523                                     ;-----
1524 062A                                     GRAPHICS_READ PROC NEAR
1525 062A E8 06D5 R                          CALL S26
1526 062D 8B F0                             MOV SI,AX
1527 062F 83 EC 08                          SUB SP,8
1528 0632 8B EC                             MOV BP,SP
1529
1530 ;----- DETERMINE GRAPHICS MODES
1531
1532 0634 80 3E 0049 R 06                    CMP #CRT_MODE,6
1533 0639 06                                 PUSH ES
1534 063A IF                                 POP DS
1535 063B 72 19                             JC S13
1536
1537 ;----- HIGH RESOLUTION READ
1538
1539 ;----- GET VALUES FROM REGEN BUFFER AND CONVERT TO CODE POINT
1540 063D 86 04                             MOV DH,4
1541 063F S12:
1542 063F 8A 04                             MOV AL,[SI]
1543 0641 88 46 00                          MOV [BP],AL
1544 0644 45                                 INC BP
1545 0645 8A 84 2000                         MOV AL,[SI+2000H]
1546 0649 88 46 00                          MOV [BP],AL
1547 064C 45                                 INC BP
1548 064D 83 C6 50                         ADD SI,80
1549 0650 FE CE                             DEC DH
1550 0652 75 EB                             JNZ S12
1551 0654 EB 16                             JMP SHORT S15
1552
1553 ;----- MEDIUM RESOLUTION READ
1554 0656 S13:
1555 0656 D1 E6                             SAL SI,1
1556 0658 B6 04                             MOV DH,4
1557 065A S14:
1558 065A E8 06BC R                          CALL S23
1559 065D B1 C6 1FFE                         ADD SI,2000H-2
1560 0661 E8 06BC R                          CALL S23
1561 0664 81 EE 1FB2                        SUB SI,2000H-80+2
1562 0668 FE CE                             DEC DH
1563 066A 75 EE                             JNZ S14
1564
1565 ;----- SAVE AREA HAS CHARACTER IN IT, MATCH IT
1566 0666 S15:
1567 0666 BF 0000 E                          MOV DI,OFFSET CRT_CHAR_GEN
1568 066F 0E                                 PUSH CS
1569 0670 07                                 POP ES
1570 0671 83 ED 08                          SUB BP,8
1571 0674 8B F5                             MOV SI,BP
1572 0676 FC                                 CLD
1573 0677 80 00                             MOV AL,0
1574 0679 S16:
1575 0679 16                                 PUSH SI
1576 067A IF                                 POP DS
1577 067B 8A 0080                          MOV DX,128
1578 067E S17:
1579 067E 56                                 PUSH SI
1580 067F 57                                 PUSH DI
1581 0680 89 0004                          MOV CX,4
1582 0683 F3 A7                             REPE CMPSW
1583 0685 5F                                 POP DI
1584 0686 5E                                 POP SI
1585 0687 74 1E                             JZ S18
1586 0689 FE C0                             INC AL
1587 068B 83 C7 08                          ADD DI,8
1588 068E 4A                                DEC DX
1589 068F 75 ED                             JNZ S17
1590
1591 ;----- CHAR NOT MATCHED, MIGHT BE IN USER SUPPLIED SECOND HALF
1592
1593 0691 3C 00                             CMP AL,0
1594 0693 74 12                             JNE S18
1595 0695 2B C0                             SUB AX,AX
1596 0697 8E D8                             MOV DS,AX
1597
1598 0699 C4 3E 007C R                       ASSUME DS:ABS0
1599 069D 8C C0                             LES DI,0EXT_PTR
1600 069F 0B C7                             OR AX,ES
1601 06A1 74 04                             MOV AX,DI
1602 06A3 80 80                             JZ S18
1603 06A5 EB D2                             MOV AL,128
1604
1605 ASSUME DS:DATA
1606
1607 06A7 S18:
1608 06A7 83 C4 08                          ADD SP,8
1609 06AA E9 012E R                          JMP VIDEO_RETURN
1610 06AD
1611 GRAPHICS_READ ENDP
1612 ;-----
1613 ; EXPAND BYTE
1614 ; THIS ROUTINE TAKES THE BYTE IN AL AND DOUBLES ALL
1615 ; OF THE BITS, TURNING THE 8 BITS INTO 16 BITS.
1616 ; THE RESULT IS LEFT IN AX
1617 06AD S21:
1618 06AD 51                             PROC NEAR
1619 06AE 89 0008                          PUSH CX
1620 06B1 MOV CX,8
1621 06B1 D0 C8                             ROR AL,1
1622 06B3 D1 DD                             RCR BP,1
1623 06B5 D1 FD                             SAR BP,1
1624 06B7 E2 F8                             LOOP S22
1625
1626 06B9 95                             XCHG AX,BP
1627 06BA 59                             POP CX
1628 06BB C3                             RET
1629 06BC S21: ENDP
1630 ;-----
1631 ; MED_READ_BYTE
1632 ; THIS ROUTINE WILL TAKE 2 BYTES FROM THE REGEN BUFFER,
1633 ; COMPARE AGAINST THE CURRENT FOREGROUND COLOR, AND PLACE
1634 ; THE CORRESPONDING ON/OFF BIT PATTERN INTO THE CURRENT
1635 ; POSITION IN THE SAVE AREA
1636 ; ENTRY --
    
```

SECTION 5

```

1637      ; SI,DS = POINTER TO REGEN AREA OF INTEREST
1638      ; BX = EXPANDED FOREGROUND COLOR
1639      ; BP = POINTER TO SAVE AREA
1640      ; EXIT --
1641      ; SI AND BP ARE INCREMENTED
1642
1643 06BC      S23      PROC      NEAR
1644 06BC AD      LODSW     XCHG      AL,AH      ; GET FIRST BYTE AND SECOND BYTES
1645 06BD B6 C4    MOV      CX,0C000H ; SWAP FOR COMPARE
1646 06BF B9 C000  MOV      DX,0      ; 2 BIT MASK TO TEST THE ENTRIES
1647 06C2 B2 00    MOV      DX,0      ; RESULT REGISTER
1648 06C4          S24:     TEST      AX,CX      ; IS THIS SECTION BACKGROUND?
1649 06C5 85 C1    JZ       S25        ; IF ZERO, IT IS BACKGROUND (CARRY=0)
1650 06C6 74 01    STC      S25        ; WASN'T, SO SET CARRY
1651 06C8 F9          S25:     RCL      DL,1      ; MOVE THAT BIT INTO THE RESULT
1652 06C9          SHR      CX,2      ; MOVE THE MASK TO THE RIGHT BY 2 BITS
1653 06CB C1 E9 02 JNC      S24        ; DO IT AGAIN IF MASK DIDN'T FALL OUT
1654 06CC 8B 56 00 MOV      [BP],DL    ; STORE RESULT IN SAVE AREA
1655 06CE 73 F4    INC      BP        ; ADJUST POINTER
1656 06D0 88 56 00 INC      BP        ; ALL DONE
1657 06D3 45          S23      ENDP
1658 06D4 C3
1659 06D5
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          S26      PROC      NEAR
1672 06D5 A1 0050 R  MOV      AX,CURSOR_POSN ; GET CURRENT CURSOR
1673 06D8          GRAPH_POSN LABEL NEAR
1674 06D8 53      PUSH     BX            ; SAVE REGISTER
1675 06D9 B8 D8    MOV      BX,AX        ; SAVE COPY OF CURRENT CURSOR
1676 06DB BA C4    MOV      AL,AH        ; GET ROWS TO AL
1677 06DD F6 26 004A R MUL     BYTE PTR #CRT_COLS ; MULTIPLY BY BYTES/COLUMN
1678 06E1 C1 ED 02 SHL     AX,2          ; MULTIPLY * 4 SINCE 4 ROWS/BYTE
1679 06E4 2A FF    SUB     BH,BH        ; ISOLATE COLUMN VALUE
1680 06E6 03 C3    ADD     AX,BX        ; DETERMINE OFFSET
1681 06E8 5B      POP     BX            ; RECOVER POINTER
1682 06E9 C3      RET
1683 06EA          S26      ENDP
1684
1685      ;--- WRITE_TTY
1686
1687      ; THIS INTERFACE PROVIDES A TELETYPE LIKE INTERFACE TO THE
1688      ; VIDEO CARDS. THE INPUT CHARACTER IS WRITTEN TO THE CURRENT
1689      ; CURSOR POSITION, AND THE CURSOR IS MOVED TO THE NEXT POSITION.
1690      ; IF THE CURSOR LEAVES THE LAST COLUMN OF THE FIELD, THE COLUMN
1691      ; IS SET TO ZERO, AND THE ROW VALUE IS INCREMENTED. IF THE ROW
1692      ; ROW VALUE LEAVES THE FIELD, THE CURSOR IS PLACED ON THE LAST ROW,
1693      ; FIRST COLUMN, AND THE ENTIRE SCREEN IS SCROLLED UP ONE LINE.
1694      ; WHEN THE SCREEN IS SCROLLED UP, THE ATTRIBUTE FOR FILLING THE
1695      ; NEWLY BLANKED LINE IS READ FROM THE CURSOR POSITION ON THE PREVIOUS
1696      ; LINE BEFORE THE SCROLL, IN CHARACTER MODE. IN GRAPHICS MODE,
1697      ; THE 0 COLOR IS USED.
1698      ; ENTRY --
1699      ; (AH) = CURRENT CRT MODE
1700      ; (AL) = CHARACTER TO BE WRITTEN
1701      ; NOTE THAT BACK SPACE, CARRIAGE RETURN, BELL AND LINE FEED ARE
1702      ; HANDLED AS COMMANDS RATHER THAN AS DISPLAY GRAPHICS CHARACTERS
1703      ; (BL) = FOREGROUND COLOR FOR CHAR WRITE IF CURRENTLY IN A GRAPHICS MODE
1704      ; EXIT --
1705      ; ALL REGISTERS SAVED
1706
1707          ASSUME DS:DATA
1708 06EA          WRITE_TTY PROC NEAR
1709 06EB 50      PUSH     AX            ; SAVE REGISTERS
1710 06ED 50      PUSH     AX            ; SAVE CHARACTER TO WRITE
1711 06EE B4 03    MOV      AH,03H        ; GET CURRENT PAGE SETTING
1712 06F2 CD 10    INT     10H           ; READ THE CURRENT CURSOR POSITION
1713 06F4 58      POP     AX            ; RECOVER CHARACTER
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 U0:
1722 06F9 B4 0A      MOV      AH,0AH        ; WRITE CHARACTER ONLY COMMAND
1723 06FB B9 0001 MOV      CX,1          ; ONLY ONE CHARACTER
1724 06FE CD 10    INT     10H           ; WRITE THE CHARACTER
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,BYTE PTR #CRT_COLS ; SET_CURSOR
1730 0706 75 33    JNZ     U7            ; CHECK FOR CURSOR
1731 0708 B2 00    MOV      DX,0          ; CHECK FOR LAST ROW
1732 070A 80 FE 18 CMP     DH,25-1       ; SET_CURSOR_INC
1733 070D 75 2A    JNZ     U6            ; SET_CURSOR_INC
1734
1735      ;---- SCROLL REQUIRED
1736 U1:
1737 070F B4 02      MOV      AH,02H        ; SET THE CURSOR
1738 0711 CD 10    INT     10H           ; READ THE CURRENT CURSOR POSITION
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            ; READ-CURSOR
1745 071A 3C 07    CMP     AL,7          ; FILL WITH BACKGROUND
1746 071C B7 00    MOV     BH,0          ; SCROLL-UP
1747 071E 75 06    JNE     U3            ; READ-CURSOR
1748 0720          U2:
1749 0720 B4 08      MOV      AH,08H        ; GET READ CURSOR COMMAND
1750 0722 CD 10    INT     10H           ; READ CHAR/ATTR AT CURRENT CURSOR
    
```

```

1751 0724 8A FC          MOV    BH,AH          ; STORE IN BH
1752 0726                ; SCROLL-UP
1753 0726 BB 0601       MOV    AX,0601H       ; SCROLL ONE LINE
1754 0729 2B C9         SUB    CX,CX          ; UPPER LEFT CORNER
1755 072B B6 18         MOV    DH,25-1       ; LOWER RIGHT ROW
1756 072D 8A 16 004A R  MOV    DL,BYTE PTR @CRT_COLS ; LOWER RIGHT COLUMN
1757 0731 FE CA         DEC    DL
1758 0733                ; VIDEO-CALL-RETURN
1759 0733 CD 10         INT    10H           ; SCROLL UP THE SCREEN
1760 0735                ; TTY-RETURN
1761 0735 58            POP    AX             ; RESTORE THE CHARACTER
1762 0736 E9 012E R     JMP    VIDEO_RETURN  ; RETURN TO CALLER
1763
1764 0739                U6:    ; SET-CURSOR-INC
1765 0739 FE C6         INC    DH             ; NEXT ROW
1766 073B                ; SET-CURSOR
1767 073B B4 02         MOV    AH,02H        ; ESTABLISH THE NEW CURSOR
1768 073D EB F4         JMP    U4
1769
1770                ;----- CHECK FOR CONTROL CHARACTERS
1771 073F                U8:    ; WAS IT A CARRIAGE RETURN
1772 073F 74 13         JE     U9             ; IS IT A LINE FEED
1773 0741 3C 0A         CMP    AL,LF         ; GO TO LINE FEED
1774 0743 74 13         JE     U10           ; IS IT A BELL
1775 0745 3C 07         CMP    AL,07H       ; GO TO BELL
1776 0747 74 16         JE     U11           ; IS IT A BACKSPACE
1777 0749 3C 08         CMP    AL,08H       ; IF NOT A CONTROL, DISPLAY IT
1778 074B 75 AC         JNE   U0
1779
1780                ;----- BACK SPACE FOUND
1781 074D                ;
1782 074D 0A D2         OR     DL,DL         ; IS IT ALREADY AT START OF LINE
1783 074F 74 EA         JE     U7            ; SET_CURSOR
1784 0751 4A            DEC    DX            ; NO -- JUST MOVE IT BACK
1785 0752 EB E7         JMP    U7            ; SET_CURSOR
1786
1787                ;----- CARRIAGE RETURN FOUND
1788 0754                ;
1789 0754                U9:    ; MOVE TO FIRST COLUMN
1790 0754 B2 00         MOV    DL,0          ; SET_CURSOR
1791 0756 EB E3         JMP    U7
1792
1793                ;----- LINE FEED FOUND
1794 0758                ;
1795 0758                U10:   ;
1796 0758 80 FE 18     CMP    DH,25-1       ; BOTTOM OF SCREEN
1797 075B 75 DC         JNE   U6             ; YES, SCROLL THE SCREEN
1798 075D EB B0         JMP    U1            ; NO, JUST SET THE CURSOR
1799
1800                ;----- BELL FOUND
1801 075F                ;
1802 075F                U11:   ;
1803 075F B9 0533       MOV    CX,1331       ; DIVISOR FOR 896 HZ TONE
1804 0762 B3 1F         MOV    BL,31         ; SET COUNT FOR 31/64 SECOND FOR BEEP
1805 0764 EB 0000 E     CALL  BEEP           ; SOUND THE POD BELL
1806 0767 EB CC         JMP    U5            ; TTY_RETURN
1807 0769                ;
1808                ;----- WRITE TTY
1809                ; LIGHT PEN
1810                ; THIS ROUTINE TESTS THE LIGHT PEN SWITCH AND THE LIGHT
1811                ; PEN TRIGGER. IF BOTH ARE SET, THE LOCATION OF THE LIGHT
1812                ; PEN IS DETERMINED. OTHERWISE, A RETURN WITH NO INFORMATION
1813                ; IS MADE.
1814                ; ON EXIT:
1815                ; (AH) = 0 IF NO LIGHT PEN INFORMATION IS AVAILABLE
1816                ; BX,CX,DX ARE DESTROYED
1817                ; (AH) = 1 IF LIGHT PEN IS AVAILABLE
1818                ; (DH,DL) = ROW,COLUMN OF CURRENT LIGHT PEN POSITION
1819                ; (CH) = RASTER POSITION
1820                ; (BX) = BEST GUESS AT PIXEL HORIZONTAL POSITION
1821                ;-----
1822                ; ASSUME DS:DATA
1823 0769 03 03 05 05 03 03 VI  DS:DATA          ; SUBTRACT_TABLE
1824 03 04                DB    3,3,5,5,3,3,3,4
1825
1826                ;----- WAIT FOR LIGHT PEN TO BE DEPRESSED
1827 0771                ;
1828 0771 B4 00                PROC  NEAR
1829 0773 BB 16 0063 R    MOV    DX,@ADDR_6845 ; SET NO LIGHT PEN RETURN CODE
1830 0777 83 C2 06       ADD    DX,6          ; GET BASE ADDRESS OF 6845
1831 077A EC             IN     AL,DX         ; POINT TO STATUS REGISTER
1832 077B A8 04         TEST   AL,004H       ; GET STATUS REGISTER
1833 077D 74 03         JZ     V6_A          ; TEST LIGHT PEN SWITCH
1834 077F E9 0803 R     JMP    V6            ; GO IF YES
1835                ; NOT SET, RETURN
1836
1837                ;----- NOW TEST FOR LIGHT PEN TRIGGER
1838 0782 A8 02                V6_A:  TEST   AL,2          ; TEST LIGHT PEN TRIGGER
1839 0784 75 03                JNZ   V7A           ; RETURN WITHOUT RESETTING TRIGGER
1840 0786 E9 080D R     JMP    V7
1841
1842                ;----- TRIGGER HAS BEEN SET, READ THE VALUE IN
1843 0789                ;
1844 0789                V7A:   ;
1845 0789 B4 10                MOV    AH,16         ; LIGHT PEN REGISTERS ON 6845
1846
1847                ;----- INPUT REGISTERS POINTED TO BY AH, AND CONVERT TO ROW COLUMN IN (DX)
1848 078B                ;
1849 078B 8B 16 0063 R    MOV    DX,@ADDR_6845 ; ADDRESS REGISTER FOR 6845
1850 078F 8A C4          MOV    AL,AH         ; REGISTER TO READ
1851 0791 EE            OUT    DX,AL         ; SET IT UP
1852 0793 EB 00         JMP    $+2           ; I/O DELAY
1853 0794 42          INC    DX            ; DATA REGISTER
1854 0795 EC          IN     AL,DX         ; GET THE VALUE
1855 0796 8A E8         MOV    CH,AL         ; SAVE IN CH
1856 0798 4A          DEC    DX            ; ADDRESS REGISTER
1857 0799 FE C4         INC    AH            ;
1858 079B 8A C4         MOV    AL,AH         ; SECOND DATA REGISTER
1859 079D EE            OUT    DX,AL         ;
1860 079E 42          INC    DX            ; POINT TO DATA REGISTER
1861 079F EB 00         JMP    $+2           ; I/O DELAY
1862 07A1 EC          IN     AL,DX         ; GET SECOND DATA VALUE
1863 07A2 8A E5         MOV    AH,CH         ; AX HAS INPUT VALUE
1864
    
```

SECTION 5

```

1865          1----- 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                ; MODE VALUE TO BX
1869 07AA 2E: 8A 9F 0769 R  MOV    BL,CS:V1[BX]        ; DETERMINE AMOUNT TO SUBTRACT
1870 07AF 2B C3            SUB    AX,BX                ; TAKE IT AWAY
1871 07B1 8B 1E 004E R      MOV    BX,#CRT_START
1872 07B5 D1 EB            SHR   BX,1
1873 07B7 2B C3            SUB    AX,BX                ; CONVERT TO CORRECT PAGE ORIGIN
1874 07B9 79 02            JNS   V2                    ; IF POSITIVE, DETERMINE MODE
1875 07BB 2B C0            SUB    AX,AX                ; <0 PLAYS AS 0
1876
1877          1----- DETERMINE MODE OF OPERATION
1878
1879 07BD            V2:
1880 07BD B1 03            MOV    CL,3                ; DETERMINE MODE
1881 07BF 80 3E 0049 R 04  CMP    #CRT_MODE,4        ; SET *8 SHIFT COUNT
1882 07C4 72 29            JB    V4                    ; DETERMINE IF GRAPHICS OR ALPHA
1883 07C6 80 3E 0049 R 07  CMP    #CRT_MODE,7        ; ALPHA_PEN
1884 07CB 74 22            JE    V4                    ; ALPHA_PEN
1885
1886          1----- 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          1----- DETERMINE GRAPHIC ROW POSITION
1892
1893 07D1 8A E8            MOV    CH,AL               ; SAVE ROW VALUE IN CH
1894 07D3 02 D0            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 8 FOR MEDIUM RES
1897 07D9 80 3E 0049 R 06  CMP    #CRT_MODE,6        ; DETERMINE MEDIUM OR HIGH RES
1898 07DB 75 04            JNE   V3                    ; NOT HIGH RES
1899 07DD B1 04            MOV    CL,4                ; SHIFT VALUE FOR HIGH RES
1900 07E2 D0 E4            SAL   AH,1                 ; COLUMN VALUE TIMES 2 FOR HIGH RES
1901 07E4
1902 07E4 D3 E3            V3:  SHL   BX,CL              ; NOT HIGH RES
1903
1904          1----- 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          1----- ALPHA MODE ON LIGHT PEN
1912
1913 07EF            V4:
1914 07EF F6 36 004A R      DIV   BYTE PTR #CRT_COLS  ; ALPHA PEN
1915 07F3 8A F0            MOV    DH,AL               ; DETERMINE ROW,COLUMN VALUE
1916 07F5 8A D4            MOV    DL,AH               ; ROWS TO DH
1917 07F7 D2 E0            SAL   AL,CL                ; COLS TO DL
1918 07F9 8A EC            MOV    CH,AL               ; MULTIPLY ROWS * 8
1919 07FB 8A DC            MOV    BL,AH               ; GET RASTER VALUE TO RETURN REGISTER
1920 07FD 32 FF            XOR   BH,BH                ; COLUMN VALUE
1921 07FF D3 E3            SAL   BX,CL                ; TO BX
1922 0801
1923 0801 B4 01            V5:  MOV    AH,1                 ; LIGHT_PEN_RETURN_SET
1924 0803
1925 0803 52            V6:  PUSH  DX                    ; INDICATE EVERY THING SET
1926 0804 8B 16 0063 R    MOV    DX,#ADDR_6845     ; LIGHT_PEN_RETURN
1927 0808 82 C2 07        ADD   DX,7                 ; SAVE RETURN VALUE (IN CASE)
1928 080B EE            OUT   DX,AL                ; GET BASE ADDRESS
1929 080C 5A            POP   DX                    ; POINT TO RESET FIRM
1930 080D            V7:  POP   BP                    ; ADDRESS, NOT DATA, IS IMPORTANT
1931 080D 5D            POP   DI                    ; RECOVER VALUE
1932 080E 5F            POP   SI                    ; RETURN_NO_RESET
1933 080F 5E            POP   DI
1934 0810 1F            POP   DS                    ; DISCARD SAVED BX,CX,DX
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_1
8      PUBLIC MEMORY_SIZE_DET_1
9      PUBLIC NMI_INT_1
10
11     EXTRN C8042:NEAR          ; POST SEND 8042 COMMAND ROUTINE
12     EXTRN CMOS_READ:NEAR     ; READ CMOS LOCATION ROUTINE
13     EXTRN D1:NEAR           ; "PARITY CHECK 1" MESSAGE
14     EXTRN D2:NEAR           ; "PARITY CHECK 2" MESSAGE
15     EXTRN 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_HEX:NEAR      ; DISPLAY CHARACTER ROUTINE
19     EXTRN PRT_SEG:NEAR      ; DISPLAY FIVE CHARACTER ADDRESS ROUTINE
20     EXTRN P_MSG:NEAR        ; DISPLAY MESSAGE STRING ROUTINE
21
22     ;----- INT 12 H -----
23     ; MEMORY SIZE DETERMINATION
24     ; THIS ROUTINE RETURNS THE AMOUNT OF MEMORY IN THE SYSTEM AS
25     ; DETERMINED BY THE POST ROUTINES. (UP TO 640K)
26     ; NOTE THAT THE SYSTEM MAY NOT BE ABLE TO USE I/O MEMORY UNLESS
27     ; THERE IS A FULL COMPLEMENT OF 512K BYTES ON THE PLANAR.
28     ; INPUT
29     ; NO REGISTERS
30     ; THE @MEMORY_SIZE VARIABLE IS SET DURING POWER ON DIAGNOSTICS
31     ; ACCORDING TO THE FOLLOWING ASSUMPTIONS:
32     ;
33     ; 1. CONFIGURATION RECORD IN NON-VOLATILE MEMORY EQUALS THE ACTUAL
34     ;    MEMORY SIZE INSTALLED.
35     ;
36     ; 2. ALL INSTALLED MEMORY IS FUNCTIONAL. IF THE MEMORY TEST DURING
37     ;    POST INDICATES LESS, THEN THIS VALUE BECOMES THE DEFAULT.
38     ;    IF NON-VOLATILE MEMORY IS NOT VALID (NOT INITIALIZED OR BATTERY
39     ;    FAILURE) THEN ACTUAL MEMORY DETERMINED BECOMES THE DEFAULT.
40     ;
41     ; 3. ALL MEMORY FROM 0 TO 640K MUST BE CONTIGUOUS.
42     ;
43     ; OUTPUT
44     ; (AX) = NUMBER OF CONTIGUOUS 1K BLOCKS OF MEMORY
45     ;-----
46     ASSUME CS:CODE,DS:DATA
47
48     0000 MEMORY_SIZE_DET_1 PROC FAR
49     0000 FB STI ; INTERRUPTS BACK ON
50     0001 IE PUSH DS ; SAVE SEGMENT
51     0002 EB 0000 E CALL DDS ; ESTABLISH ADDRESSING
52     0005 A1 0013 R MOV AX,@MEMORY_SIZE ; GET VALUE
53     0008 1F POP DS ; RECOVER SEGMENT
54     0009 CF IRET ; RETURN TO CALLER
55     000A MEMORY_SIZE_DET_1 ENDP
56
57     ;----- INT 11 H -----
58     ; EQUIPMENT DETERMINATION
59     ; THIS ROUTINE ATTEMPTS TO DETERMINE WHAT OPTIONAL
60     ; DEVICES ARE ATTACHED TO THE SYSTEM.
61     ; INPUT
62     ; NO REGISTERS
63     ; THE @EQUIP_FLAG VARIABLE IS SET DURING THE POWER ON
64     ; DIAGNOSTICS USING THE FOLLOWING HARDWARE ASSUMPTIONS:
65     ; PORT 03FA = INTERRUPT ID REGISTER OF 8250 (PRIMARY)
66     ; 02FA = INTERRUPT ID REGISTER OF 8250 (SECONDARY)
67     ; BITS 7-3 ARE ALWAYS 0
68     ; PORT 0378 = OUTPUT PORT OF PRINTER (PRIMARY)
69     ; 0278 = OUTPUT PORT OF PRINTER (SECONDARY)
70     ; 03BC = OUTPUT PORT OF PRINTER (MONOCHROME-PRINTER)
71     ; OUTPUT
72     ; (AX) IS SET, BIT SIGNIFICANT, TO INDICATE ATTACHED I/O
73     ; BIT 15,14 = NUMBER OF PRINTERS ATTACHED
74     ; BIT 13 = INTERNAL MODEM INSTALLED
75     ; BIT 12 NOT USED
76     ; BIT 11,10,9 = NUMBER OF RS232 CARDS ATTACHED
77     ; BIT 8 = NOT USED
78     ; BIT 7,6 = NUMBER OF DISKETTE DRIVES
79     ; 00=1, 01=2 ONLY IF BIT 0 = 1
80     ; BIT 5,4 = INITIAL VIDEO MODE
81     ; 00 - UNUSED
82     ; 01 - 40X25 BW USING COLOR CARD
83     ; 10 - 80X25 BW USING COLOR CARD
84     ; 11 - 80X25 BW USING BW CARD
85     ;
86     ; BIT 3 = NOT USED
87     ; BIT 2 = NOT USED
88     ; BIT 1 = MATH COPROCESSOR
89     ; BIT 0 = 1 (IPL DISKETTE INSTALLED)
90     ; NO OTHER REGISTERS AFFECTED
91     ;-----
92
93     000A EQUIPMENT_1 PROC FAR ; ENTRY POINT FOR ORG 0F84DH
94     000A FB STI ; INTERRUPTS BACK ON
95     000B IE PUSH DS ; SAVE SEGMENT REGISTER
96     000C EB 0000 E CALL DDS ; ESTABLISH ADDRESSING
97     000F A1 0010 R MOV AX,@EQUIP_FLAG ; GET THE CURRENT SETTINGS
98     0012 1F POP DS ; RECOVER SEGMENT
99     0013 CF IRET ; RETURN TO CALLER
100    0014 EQUIPMENT_1 ENDP

```

SECTION 5

```

101 PAGE
102 ;-- HARDWARE INT 02H -- ( NMI LEVEL )-----
103 ; NON-MASKABLE INTERRUPT ROUTINE (REAL MODE)
104 ; THIS ROUTINE WILL PRINT A "PARITY CHECK 1 OR 2" MESSAGE AND ATTEMPT
105 ; TO FIND THE STORAGE LOCATION IN BASE 640K CONTAINING THE BAD PARITY.
106 ; IF FOUND, THE SEGMENT ADDRESS WILL BE PRINTED. IF NO PARITY ERROR
107 ; CAN BE FOUND (INTERMITTENT READ PROBLEM) ?????? WILL BE DISPLAYED
108 ; WHERE THE ADDRESS WOULD NORMALLY GO.
109 ;
110 ; PARITY CHECK 1 = PLANAR BOARD MEMORY FAILURE.
111 ; PARITY CHECK 2 = OFF BOARD BOARD MEMORY FAILURE.
112 -----
113
114 NMI_INT_1 PROC NEAR
115     PUSH    AX                ; SAVE ORIGINAL CONTENTS OF (AX)
116
117     IN      AL,PORT_B        ; READ STATUS PORT
118     TEST   AL,PARITY_ERR    ; PARITY CHECK OR I/O CHECK ?
119     JNZ    NMI_1            ; GO TO ERROR HALTS IF HARDWARE ERROR
120
121     MOV    AL,CMOS_REG_D    ; ELSE ?? - LEAVE NMI ON
122     CALL  CMOS_READ        ; TOGGLE NMI USING COMMON READ ROUTINE
123     POP    AX              ; RESTORE ORIGINAL CONTENTS OF (AX)
124     IRET                   ; EXIT NMI HANDLER BACK TO PROGRAM
125
126
127 NMI_1:
128     PUSH    AX                ; SAVE INITIAL CHECK MASK IN (AL)
129     MOV    AL,CMOS_REG_D+NMI ; MASK TRAP (NMI) INTERRUPTS OFF
130     OUT   CMOS_PORT,AL
131     MOV    AL,DTS_KBD
132     CALL  C8042             ; DISABLE THE KEYBOARD
133     CALL  DDS              ; SEND COMMAND TO ADAPTER
134     CALL  DDS              ; ADDRESS DATA SEGMENT
135     MOV    AH,0            ; INITIALIZE AND SET MODE FOR VIDEO
136     MOV    AL,PCRT_MODE    ; GET CURRENT MODE
137     INT   10H             ; CALL VIDEO_IO TO CLEAR SCREEN
138
139 ;----- DISPLAY "PARITY CHECK ?" ERROR MESSAGES
140     POP    AX              ; RECOVER INITIAL CHECK STATUS
141     MOV    SI,OFFSET D1     ; PLANAR ERROR, ADDRESS "PARITY CHECK 1"
142     TEST  AL,PARITY_CHECK ; CHECK FOR PLANAR ERROR
143     JZ    NMI_2            ; SKIP IF NOT
144
145     PUSH    AX              ; SAVE STATUS
146     CALL  P_MSG           ; DISPLAY "PARITY CHECK 1" MESSAGE
147     POP    AX              ; AND RECOVER STATUS
148
149 NMI_2:
150     MOV    SI,OFFSET D2     ; ADDRESS OF "PARITY CHECK 2" MESSAGE
151     TEST  AL,IO_CHECK     ; I/O PARITY CHECK ?
152     JZ    NMI_3            ; SKIP IF CORRECT ERROR DISPLAYED
153     CALL  P_MSG           ; DISPLAY "PARITY CHECK 2" ERROR
154
155 ;----- TEST FOR HOT NMI ON PLANAR PARITY LINE
156
157 NMI_3:
158     IN      AL,PORT_B
159     OR     AL,RAM_PAR_OFF  ; TOGGLE PARITY CHECK ENABLES
160     OUT   PORT_B,AL
161     AND   AL,RAM_PAR_ON   ; TO CLEAR THE PENDING CHECK
162     OUT   PORT_B,AL
163
164     CLD                    ; SET DIRECTION FLAG TO INCREMENT
165     SUB    DX,DX           ; POINT (DX) AT START OF REAL MEMORY
166     SI,SI                 ; SET (SI) TO START OF (DS:1)
167     IN      AL,PORT_B
168     TEST  AL,PARITY_ERR   ; READ CURRENT PARITY CHECK LATCH
169     JNZ    NMI_5          ; CHECK FOR HOT NMI SOURCE
170
171 ;----- SEE IF LOCATION THAT CAUSED PARITY CHECK CAN BE FOUND IN BASE MEMORY
172     MOV    BX,#MEMORY_SIZE ; GET BASE MEMORY SIZE WORD
173
174 NMI_4:
175     MOV    DS,DX           ; POINT TO 64K SEGMENT
176     MOV    CX,4000H*2     ; SET WORD COUNT FOR 64 KB SCAN
177     REP   LODSB           ; READ 64 KB OF MEMORY
178     IN      AL,PORT_B
179     TEST  AL,PARITY_ERR   ; READ PARITY CHECK LATCHES
180     JNZ    NMI_6          ; CHECK FOR ANY PARITY ERROR PENDING
181
182     ADD    DH,010H        ; GO PRINT SEGMENT ADDRESS IF ERROR
183     SUB    BX,16D*4
184     JVA   NMI_4
185
186 NMI_5:
187     MOV    SI,OFFSET D2A   ; POINT TO NEXT 64K BLOCK
188     CALL  P_MSG           ; DECREMENT COUNT OF 1024 BYTE SEGMENTS
189     HLT                    ; LOOP TILL ALL 64K SEGMENTS DONE
190
191 NMI_6:
192     CALL  PRT_SEG         ; PRINT SEGMENT VALUE (IN DX)
193     MOV    AL,'('
194     CALL  PRT_HEX        ; PRINT (S)
195     MOV    AL,','
196     CALL  PRT_HEX
197     MOV    AL,')'
198     CALL  PRT_HEX
199     CALL  CLT
200     HLT                    ; HALT SYSTEM
201
202 NMI_INT_1 ENDP
203
204 CODE ENDS
205 END

```

```

1 PAGE 118,121
2 TITLE BIOS1 ---- 06/10/85 INTERRUPT 15H BIOS ROUTINES
3 .286C
4 .LIST
5 0000 CODE SEGMENT BYTE PUBLIC
6
7 PUBLIC CASSETTE_10_1
8 PUBLIC GATE_A20
9 PUBLIC SHUT9
10
11 EXTRN CMOS_READ:NEAR ; READ CMOS LOCATION ROUTINE
12 EXTRN CMOS_WRITE:NEAR ; WRITE CMOS LOCATION ROUTINE
13 EXTRN CONF_TBL:NEAR ; SYSTEM/BIOS CONFIGURATION TABLE
14 EXTRN DDS:NEAR ; LOAD (DS) WITH DATA SEGMENT SELECTOR
15 EXTRN PROC_SHUTDOWN:NEAR ; 80286 HARDWARE RESET ROUTINE
16
17 -----INT 15H-----
18 INPUT - CASSETTE I/O FUNCTIONS
19
20 ; (AH) = 00H
21 ; (AH) = 01H
22 ; (AH) = 02H
23 ; (AH) = 03H
24 ; RETURNS FOR THESE FUNCTIONS ALWAYS (AH) = 86H, CY = 1)
25 ; IF CASSETTE PORT NOT PRESENT
26 -----
27 INPUT - UNUSED FUNCTIONS
28 ; (AH) = 04H THROUGH 7FH
29 ; RETURNS FOR THESE FUNCTIONS ALWAYS (AH) = 86H, CY = 1)
30 ; (UNLESS INTERCEPTED BY SYSTEM HANDLERS)
31 ; NOTE: THE KEYBOARD INTERRUPT HANDLER INTERRUPTS WITH AH=4FH
32 -----
33 EXTENSIONS
34 ; (AH) = 80H DEVICE OPEN
35 ; (BX) = DEVICE ID
36 ; (CX) = PROCESS ID
37
38 ; (AH) = 81H DEVICE CLOSE
39 ; (BX) = DEVICE ID
40 ; (CX) = PROCESS ID
41
42 ; (AH) = 82H PROGRAM TERMINATION
43 ; (BX) = DEVICE ID
44
45 ; (AH) = 83H EVENT WAIT
46
47 ; (AL) = 00H SET INTERVAL
48 ; (ES:BX) POINTER TO A BYTE IN CALLERS MEMORY
49 ; THAT WILL HAVE THE HIGH ORDER BIT SET
50 ; AS SOON AS POSSIBLE AFTER THE INTERVAL
51 ; EXPIRES.
52 ; (CX,DX) NUMBER OF MICROSECONDS TO ELAPSE BEFORE
53 ; POSTING.
54 ; (AL) = 01H CANCEL
55
56 ; RETURNS: CARRY IF AL NOT = 00H OR 01H
57 ; OR IF FUNCTION AL=0 ALREADY BUSY
58
59 ; (AH) = 84H JOYSTICK SUPPORT
60 ; (DX) = 00H - READ THE CURRENT SWITCH SETTINGS
61 ; RETURNS AL = SWITCH SETTINGS (BITS 7-4)
62 ; (DX) = 01H - READ THE RESISTIVE INPUTS
63 ; RETURNS AX = A(x) VALUE
64 ; BX = A(y) VALUE
65 ; CX = B(x) VALUE
66 ; DX = B(y) VALUE
67
68 ; (AH) = 85H SYSTEM REQUEST KEY PRESSED
69 ; (AL) = 00H MAKE OF KEY
70 ; (AL) = 01H BREAK OF KEY
71
72 ; (AH) = 86H WAIT
73 ; (CX,DX) NUMBER OF MICROSECONDS TO ELAPSE BEFORE
74 ; RETURN TO CALLER
75
76 ; (AH) = 87H MOVE_BLOCK
77 ; (CX) NUMBER OF WORDS TO MOVE
78 ; (ES:SI) POINTER TO DESCRIPTOR TABLE
79
80 ; (AH) = 88H EXTENDED MEMORY SIZE DETERMINE
81
82 ; (AH) = 89H PROCESSOR TO VIRTUAL MODE
83
84 ; (AH) = 90H DEVICE BUSY LOOP
85 ; (AL) SEE TYPE CODE
86
87 ; (AH) = 91H INTERRUPT COMPLETE FLAG SET
88 ; (AL) TYPE CODE
89 ; 00H -> TFX
90 ; SERIALLY REUSABLE DEVICES
91 ; OPERATING SYSTEM MUST SERIALIZE 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 ; COH -> FPH
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
104 00H = DISK YES
105 01H = DISKETTE YES
106 02H = KEYBOARD NO
107 80H = NETWORK NO
108 ; ES:BX -> NCB
109 FDH = DISKETTE MOTOR START YES
110 FEH = PRINTER YES
111

```

```

112 PAGE
113 ; (AH) = COH RETURN CONFIGURATION PARAMETERS POINTER
114 ; RETURNS
115 ; (AH) = 00H AND CY= 0 (IF PRESENT ELSE 86 AND CY= 1)
116 ; (ES:BX) = PARAMETER TABLE ADDRESS POINTER
117 ; WHERE:
118 ;
119 ;
120 ; DW 8 LENGTH OF FOLLOWING TABLE
121 ; DB MODEL_BYTE SYSTEM MODEL_BYTE
122 ; DB TYPE_BYTE SYSTEM MODEL_TYPE_BYTE
123 ; DB BIOS_LEVEL BIOS_REVISION_LEVEL
124 ; DB ? 1000000 = DMA CHANNEL 3 USE BY BIOS
125 ; 0100000 = CASCADED INTERRUPT LEVEL 2
126 ; 0010000 = REAL TIME CLOCK AVAILABLE
127 ; 0001000 = KEYBOARD SCAN CODE HOOK 1AH
128 ; DB 0 RESERVED
129 ; DB 0 RESERVED
130 ; DB 0 RESERVED
131 ; DB 0 RESERVED
132 ;
133 -----

```

ASSUME CS:CODE

```

136 0000 CASSETTE_IO_1 PROC FAR
137 0000 FB STI ; ENABLE INTERRUPTS
138 0001 80 FC 80 CMP AH,080H ; CHECK FOR RANGE
139 0004 72 4E JB C1 ; RETURN IF 00-7FH
140 0006 80 FC C0 CMP AH,0C0H ; CHECK FOR CONFIGURATION PARAMETERS
141 0009 74 51 JE CONF_PARMS
142 000B 80 EC 80 SUB AH,080H ; BASE ON 0
143 000E 0A E4 OR AH,AH
144 0010 74 48 JB DEV_OPEN ; DEVICE OPEN
145 0012 FE CC DEC AH
146 0014 74 44 JB DEV_CLOSE ; DEVICE CLOSE
147 0016 FE CC DEC AH
148 0018 74 40 JB PROG_TERM ; PROGRAM TERMINATION
149 001A FE CC DEC AH
150 001C 74 47 JB EVENT_WAIT ; EVENT WAIT
151 001E FE CC DEC AH
152 0020 75 03 JNZ NOT_JOYSTICK
153 0022 E9 00D0 R JMP JOY_STICK ; JOYSTICK BIOS
154 0025 NOT_JOYSTICK:
155 0025 FE CC DEC AH
156 0027 74 31 JB SYS_REQ ; SYSTEM REQUEST KEY
157 0029 FE CC DEC AH
158 002B 74 07 JB C1_A ; WAIT
159 002D FE CC DEC AH
160 002F 75 06 JNZ C1_B
161 0031 E9 01CA R JMP BLOCKMOVE ; MOVE BLOCK
162 0034 E9 016A R C1_A: JMP WAIT ; WAIT
163 0037 FE CC DEC AH
164 0039 75 03 JNZ C1_C
165 003B E9 03EE R JMP EXT_MEMORY ; GO GET THE EXTENDED MEMORY
166 003E FE CC DEC AH
167 0040 75 03 JNZ C1_D ; CHECK FOR FUNCTION 89H
168 0042 E9 03FA R JMP SET_VMODE ; SWAP TO VIRTUAL MODE
169 0045 FE CC DEC AH
170 0047 75 03 JNZ C1_D ; CHECK FOR FUNCTION 90H
171 0049 E9 0483 R JMP DEVICE_BUSY ; GO IF NOT
172 004C FE CC DEC AH
173 004E 75 03 JNZ C1_E ; CHECK FOR FUNCTION 8BH
174 0050 E9 0487 R JMP INT_COMPLETE ; GO IF NOT
175 0053 FE CC MOV AH,86H ; SET BAD COMMAND
176 0055 B4 86 STC ; SET CARRY FLAG ON
177 0057 CA 0002 C1_F: RET 2 ; FAR RETURN EXIT FROM ROUTINES
178 005A DEV_OPEN: ; NULL HANDLERS
179 005B DEV_CLOSE:
180 005C PROG_TERM:
181 005D SYS_REQ:
182 005E CASSETTE_IO_1 JMP C1_F ; RETURN
183 005F ENDP
184 0060 CONF_PARMS PROC NEAR
185 0060 0E CS ; GET CODE SEGMENT
186 0061 07 POP ES ; PLACE IN SELECTOR POINTER
187 0062 08 MOV BX,OFFSET CONF_TBL ; GET OFFSET OF PARAMETER TABLE
188 0063 04 XOR AH,AH ; CLEAR AH AND SET CARRY OFF
189 0064 02 JMP C1_F ; EXIT THROUGH COMMON RETURN
190 0065 ENDP
191 0066 EVENT_WAIT PROC NEAR
192 0066 0E ASSUME DS:DATA ; SAVE
193 0067 07 PUSH DS
194 0068 0A OR AL,AL
195 0069 0A OR AL,AL
196 006A 08 JB EVENT_WAIT_2 ; GO IF ZERO
197 006B 08 DEC AL ; CHECK IF 1
198 006C 08 JB EVENT_WAIT_3
199 006D 08 POP DS ; RESTORE DATA SEGMENT
200 006E 08 STC ; SET CARRY
201 006F 08 JMP C1_F ; EXIT
202 0070 EVENT_WAIT_2:
203 0070 08 CLI ; NO INTERRUPTS ALLOWED
204 0071 08 TEST #RTC_WAIT_FLAG,01 ; CHECK FOR FUNCTION ACTIVE
205 0072 08 JZ EVENT_WAIT_1
206 0073 08 STI ; ENABLE INTERRUPTS
207 0074 08 POP DS
208 0075 08 STC ; SET ERROR
209 0076 08 JMP C1_F ; RETURN
210 0077 08
211 0078 08
212 0079 08
213 007A 08
214 007B 08
215 007C 08
216 007D 08
217 007E 08
218 007F 08
219 0080 08
220 0081 08
221 0082 08
222 0083 08
223 0084 08
224 0085 08
225 0086 08

```



```

226
227 0082          EVENT_WAIT_1:  IN      AL,INTB01          ; ENSURE INTERRUPT UNMASKED
228 0082 E4 A1    IN          ; $+2
229 0084 EB 00    JMP          AL,OFEH
230 0086 24 FE    AND          INTB01,AL
231 0088 E6 A1    OUT          #USER_FLAG_SEG,ES ; SET UP TRANSFER TABLE
232 008A 8C 06 009A R MOV     #USER_FLAG,BX
233 008E 89 1E 009B R MOV     #RTC_HIGH,CX
234 0092 89 0E 009E R MOV     #RTC_LOW,DX
235 0096 89 16 009C R MOV     #RTC_WAIT_FLAG,01
236 009A C6 06 00A0 R 01 MOV     AL,CMOS_REG_B ; SET ON FUNCTION ACTIVE SWITCH
237 009F B0 0B    MOV     AL,CMOS_REG_B ; ENABLE PIE
238 00A1 E8 00D0 E CALL    CMOS_READ      ; READ CMOS LOCATION
239 00A4 24 7F    AND     AL,0FFH        ; CLEAR SET
240 00A6 OC 40    OR      AL,040H        ; ENABLE PIE
241 00A8 50       PUSH   AX              ; SAVE AH
242 00A9 8A E0    MOV     AH,AL          ; PLACE DATA INTO DATA REGISTER
243 00AB B0 0B    MOV     AH,CMOS_REG_B ; ADDRESS ALARM REGISTER
244 00AD EB 00D0 E CALL    CMOS_WRITE     ; PLACE DATA IN AH INTO ALARM REGISTER
245 00B0 58       POP     AX             ; RESTORE AH
246 00B1 1F       POP     DS            ;
247 00B2 FB       STI          ; ENABLE INTERRUPTS
248 00B3 F8       CLC          ; CLEAR CARRY
249 00B4 EB A1    JMP     C1_F          ;
250
251
252
253 00B6          EVENT_WAIT_3:  PUSH   AX              ; SAVE
254 00B6 50       CLD          ; DISABLE INTERRUPTS
255 00B7 FA       MOV     AX,X*CMOS_REG_B ; TURN OFF PIE
256 00B8 B8 0B0B MOV     CMOS_READ      ; GET ALARM REGISTER
257 00BB EB 00D0 E AND     AL,0BFH        ; CLEAR PIE
258 00BE 24 BF    XCHG   AH,AX          ; PLACE INTO WRITE REGISTER
259 00C0 86 E0    CALL   CMOS_WRITE     ; WRITE BACK TO ALARM REGISTER
260 00C2 EB 00D0 E POP     AX             ; RESTORE AH
261 00C5 58       MOV     #RTC_WAIT_FLAG,0 ; SET FUNCTION ACTIVE FLAG OFF
262 00C6 C4 06 00A0 R 00 STI          ; ENABLE INTERRUPTS
263 00CB FB       POP     DS            ; RESTORE DATA SEGMENT
264 00CC 1F       POP     DS            ; SET CARRY OFF
265 00CD F8       CLC          ; RETURN
266 00CE EB 87    JMP     C1_F          ;
267
268 00D0          EVENT_WAIT  ENDP
269
270 00D0          ;--- JOY STICK
271 00D0          ; THIS ROUTINE WILL READ THE JOYSTICK PORT
272 00D0          ;
273 00D0          ; INPUT
274 00D0          ; (DX)=0 READ THE CURRENT SWITCHES
275 00D0          ; RETURNS (AL)= SWITCH SETTINGS IN BITS 7-4
276 00D0          ;
277 00D0          ; (DX)=1 READ THE RESISTIVE INPUTS
278 00D0          ; RETURNS (AX)=A(x) VALUE
279 00D0          ; (BX)=A(y) VALUE
280 00D0          ; (CX)=B(x) VALUE
281 00D0          ; (DX)=B(y) VALUE
282 00D0          ;
283 00D0          ; CY FLAG ON IF NO ADAPTER CARD OR INVALID CALL
284 00D0          ;-----
285 00D0          JOY_STICK  PROC  NEAR
286 00D0 FB       STI          ; INTERRUPTS BACK ON
287 00D1 8B C2    MOV     AX,DX          ; GET SUB FUNCTION CODE
288 00D3 BA 0201 MOV     DX,201H        ; ADDRESS OF PORT
289 00D6 0A C0    MOV     AL,AH
290 00D8 74 0B    JZ     JOY_2          ; READ SWITCHES
291 00DA FE C8    DEC     AL
292 00DC 74 0C    JZ     JOY_3          ; READ RESISTIVE INPUTS
293 00DE E9 0054 R JMP     C1          ; GO TO ERROR RETURN
294 00E1          JOY_1:  STI          ; GO TO COMMON RETURN
295 00E1 FB       JMP     C1_F          ;
296 00E2 E9 0057 R
297
298 00E5          JOY_2:  IN      AL,DX          ; STRIP UNWANTED BITS OFF
299 00E5 EC       AND     AL,0F0H
300 00E6 24 F0    JMP     JOY_1          ; FINISHED
301 00E8 EB F7
302
303 00EA          JOY_3:  MOV     BL,1
304 00EA B3 01    CALL   TEST_CORD     ; SAVE A(X) VALUE
305 00EC EB 0108 R PUSH   CX
306 00EE 51       CALL   TEST_CORD     ; SAVE A(Y) VALUE
307 00F0 B3 02    PUSH  BL,2
308 00F2 EB 0108 R CALL   TEST_CORD     ; SAVE B(X) VALUE
309 00F5 51       PUSH  CX
310 00F6 B3 04    MOV     BL,4
311 00F8 EB 0108 R CALL   TEST_CORD     ; SAVE B(Y) VALUE
312 00FB 51       PUSH  CX
313 00FC B3 08    MOV     BL,8
314 00FE EB 0108 R CALL   TEST_CORD     ; SAVE B(X) VALUE
315 0101 8B D1    MOV     DX,CX
316 0103 59       POP     CX            ; GET B(Y) VALUE
317 0104 5B       POP     BX            ; GET B(X) VALUE
318 0105 5D       POP     AX            ; GET A(Y) VALUE
319 0106 EB D9    JMP     JOY_1          ; FINISHED - RETURN
320
321 0108          TEST_CORD PROC  NEAR
322 0108 52       PUSH   DX
323 0109 FA       CLD          ; SAVE
324 010A B0 00    MOV     AL,0          ; BLOCK INTERRUPTS WHILE READING
325 010C E6 43    OUT    TIMER+3,AL    ; SET UP TO LATCH TIMER 0
326 010E EB 00    JMP     $+2
327 0110 EA 40    IN     AL,TIMER      ; READ LOW BYTE OF TIMER 0
328 0112 EB 00    JMP     $+2
329 0114 8A E0    IN     AL,TIMER      ; READ HIGH BYTE OF TIMER 0
330 0116 EA 40    IN     AL,TIMER
331 0118 86 E0    XCHG   AH,AL        ; REARRANGE TO HIGH,LOW
332 011A 50       PUSH   AX            ; SAVE
333 011B B9 04FF MOV     CX,4FFH      ; SET COUNT
334 011E EE       OUT    DX,AL        ; FIRE TIMER
335 011F EB 00    JMP     $+2
336 0121          TEST_CORD_1: IN      AL,DX          ; READ VALUES
337 0121 EC       IN     AL,BL         ; HAS PULSE ENDED?
338 0122 84 C3    TEST  AL,BL
339 0124 E0 FB    LOOPNZ TEST_CORD_1
    
```

```

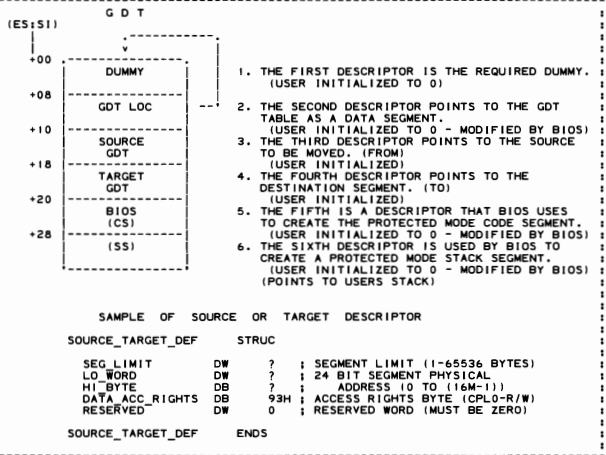
340 0126 83 F9 00          CMP     CX,0
341 0129 59              POP     CX
342 012A 75 04          JNZ     SHORT TEST_CORD_2      ; ORIGINAL COUNT
343 012C 2B C9          SUB     CX,CX
344 012E EB 28          JMP     SHORT TEST_CORD_3      ; SET 0 COUNT FOR RETURN
345 0130              TEST_CORD_2:                ; EXIT WITH COUNT = 0
346 0130 B0 00          MOV     AL,0
347 0132 E6 43          OUT     TIMER+3,AL            ; SET UP TO LATCH TIMER 0
348 0134 EB 00          JMP     $+2
349 0136 E4 40          IN     AL,TIMER              ; READ LOW BYTE OF TIMER 0
350 0138 8A E0          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
356 0142 73 0B          JAE     TEST_CORD_4          ; CHECK FOR COUNTER WRAP
357 0144 52          PUSH   DX
358 0145 BA FFFF        MOV     DX,-1                ; GO IF NO
359
360 0148 2B D0          SUB     DX,AX
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,1FF0H            ; 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 015C 51          PUSH   CX                    ; FLUSH OTHER INPUTS
375 015D 50          PUSH   AX
376 015E B9 04FF        MOV     CX,4FFH            ; COUNT
377 0161              TEST_CORD_6:                ;
378 0161 EC          IN     AL,DX
379 0162 A8 0F          TEST   AL,0FH              ;
380 0164 E0 FB          LOOPNZ TEST_CORD_6
381
382 0166 58          POP     AX
383 0167 59          POP     CX
384 0168 5A          POP     DX
385
386 0169 C3          RET
387
388 016A              TEST_CORD
389 016A              JOY_STICK
390
391 016A              WAIT
392 016A IE          PROC   NEAR
393 016B E8 0000 E      CALL   DS
394 016E F6 06 00A0 R 01 TEST   @RTC_WAIT_FLAG,01    ; SAVE
395 0173 74 05          JZ     WAIT_1              ; TEST FOR FUNCTION ACTIVE
396 0175 IF          POP     DS
397 0176 F9          STC
398 0177 E9 0057 R     JMP     C1_F                ; SET ERROR
399
400 017A FA          WAIT_1:                ; RETURN
401 017B E4 A1          CLI
402 017D EB 00          IN     AL,INTB01           ; NO INTERRUPTS ALLOWED
403 017F 24 FE          JMP     $+2                ; ENSURE INTERRUPT UNMASKED
404 0181 E6 A1          AND   AL,0FEH
405 0183 8C 1E 009A R  MOV   @USER_FLAG_SEG,DS    ; SET UP TRANSFER TABLE
406 0187 C7 06 009B R 00A0 R MOV   @USER_FLAG_OFFSET,@RTC_WAIT_FLAG
407 018D 89 0E 009E R  MOV   @RTC_HIGH,CX
408 0191 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 B8 0B0B        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 517 BIT
415 01A5 86 E0          XCHG   AH,AL               ; ENABLE PIE BIT
416 01A7 E8 0000 E     CALL  CMOS_WRITE           ; DATA TO WORK REGISTER
417 01AA 58          POP     AX                  ; WRITE NEW ALARM BYTE
418
419
420
421 01AB FB          ;----- WAIT TILL RTC TIMEOUT POSTED (WITH ERROR TIMEOUT)
422 01AC 51          STI
423 01AD 52          PUSH   CX
424 01AE 87 D1          PUSH   DX
425 01B0          XCHG   DX,CX
426 01B5 F6 06 00A0 R 00 TEST   @RTC_WAIT_FLAG,0B0H ; ENABLE INTERRUPTS
427 01B5 E1 F9          LOOPZ  WAIT_2              ; SAVE CALLERS PARAMETERS
428 01B7 75 05          JNZ   WAIT_9              ; SWAP COUNT WORK REGISTERS
429 01B9 83 EA 01      SUB     DX,1
430 01BC 73 F2          JNC   WAIT_2              ; CHECK FOR END OF WAIT - CLEAR CARRY
431 01BE          WAIT_2:                ; DECREMENT TIMEOUT DELAY TILL WAIT END
432 01BE C6 06 00A0 R 00 TEST   @RTC_WAIT_FLAG,0    ; EXIT IF RTC TIMER WAIT ENDED FLAG SET
433 01C3 5A          POP     DX
434 01C4 59          POP     CX
435 01C5 IF          POP     DS
436 01C6 F8          CLC
437 01C7 E9 0057 R     JMP     C1_F                ; DECREMENT ERROR TIMEOUT COUNTER
438
439 01CA              WAIT_9:                ; LOOP TILL COUNTERS TIMEOUT
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

```

440 PAGE
441 ---- INT 15 H -- ( FUNCTION 87 H - BLOCK MOVE ) -----
442 ;
443 ; THIS BIOS FUNCTION PROVIDES A MEANS FOR A REAL MODE PROGRAM OR SYSTEM
444 ; TO TRANSFER A BLOCK OF STORAGE TO AND FROM STORAGE ABOVE THE 1 MEG
445 ; ADDRESS RANGE IN PROTECTED MODE SPACE BY SWITCHING TO PROTECTED MODE.
446 ;
447 ; ENTRY:
448 ; (AH) = 87H (FUNCTION CALL) - BLOCK MOVE.
449 ; (CX) = WORD COUNT OF STORAGE BLOCK TO BE MOVED.
450 ; NOTE: MAX COUNT = 8000H FOR 32K WORDS (65K BYTES)
451 ; ES:SI = LOCATION OF A GDT TABLE BUILT BY ROUTINE USING THIS FUNCTION.
452 ;
453 ; (ES:SI) POINTS TO A DESCRIPTOR TABLE (GDT) BUILT BEFORE INTERRUPTING
454 ; TO THIS FUNCTION. THE DESCRIPTORS ARE USED TO PERFORM THE BLOCK
455 ; MOVE IN THE PROTECTED MODE. THE SOURCE AND TARGET DESCRIPTORS
456 ; BUILT BY THE USER MUST HAVE A SEGMENT LENGTH = 2 * CX-1 OR GREATER.
457 ; THE DATA ACCESS RIGHTS BYTE MUST BE SET TO CPL0-R/W (93H). THE
458 ; 24 BIT ADDRESS (BYTE HI, WORD LOW) MUST BE SET TO THE TARGET/SOURCE.
459 ;
460 ; *** NO INTERRUPTS ARE ALLOWED DURING TRANSFER. LARGE BLOCK MOVES
461 ; MAY CAUSE LOST INTERRUPTS.
462 ;
463 ; EXIT:
464 ; (AH) = 00H IF SUCCESSFUL
465 ; (AH) = 01H IF MEMORY PARITY (PARITY ERROR REGISTERS ARE CLEARED)
466 ; (AH) = 02H IF ANY OTHER EXCEPTION INTERRUPT ERROR OCCURRED
467 ; (AH) = 03H IF GATE ADDRESS LINE 20 FAILED
468 ; ALL REGISTERS ARE RESTORED EXCEPT (AH).
469 ;
470 ; IF SUCCESSFUL - CARRY FLAG = 0
471 ; IF ERROR ----- CARRY FLAG = 1
472 ;
473 ; DESCRIPTION:
474 ;
475 ; 1. SAVE ENTRY REGISTERS AND SETUP FOR SHUTDOWN EXIT.
476 ; 2. THE REQUIRED ENTRIES ARE BUILT IN THE GDT AT (ES:SI).
477 ; 3. GATE ADDRESS LINE 20 ACTIVE, CLI AND SET SHUTDOWN CODES.
478 ; 4. THE IDTR IS LOADED AND POINTS TO A ROM RESIDENT TABLE.
479 ; 5. THE GDTR IS LOADED FROM THE OFFSET POINTER (ES:SI).
480 ; 6. THE PROCESSOR IS PUT INTO PROTECTED MODE.
481 ; 7. LOAD (DS) AND (ES) WITH SELECTORS FOR THE SOURCE AND TARGET.
482 ; 8. DS:SI (SOURCE) (ES:DI) (TARGET) REP MOVSW IS EXECUTED.
483 ; 9. CHECK MADE FOR PARITY ERRORS.
484 ; 10. REAL MODE RESTORED WHEN SHUTDOWN 09H IS EXECUTED.
485 ; 11. ERRORS ARE CHECKED FOR AND RETURN CODES ARE SET FOR (AH).
486 ; 12. ADDRESS LINE 20 GATE IS DISABLED.
487 ; 13. RETURN WITH REGISTERS RESTORED AND STATUS RETURN CODE.
488 ; (FOR PC-AT COMPATIBILITY ZF=1 IF SUCCESSFUL, ZF=0 IF ERROR.)
489 ;
490 ;
491 ; THE FOLLOWING DIAGRAM DEPICTS THE ORGANIZATION OF A BLOCK MOVE GDT.
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 ;

```



```

535 ; THE GLOBAL DESCRIPTOR TABLE (ACTUAL LOCATION POINTED TO BY ES:SI)
536 ;
537 ;
538 ;
539 ;
540 ;
541 ;
542 ;
543 ;
544 ;
545 ;
546 ;
547 ;
548 ;
549 ;
550 ;
551 ;
552 ;
553 ;

```

SECTION 5

```

554 01D1 8C 16 0069 R      MOV     @ID_ROM_SEG,SS      ; SAVE USERS STACK SEGMENT
555 01D5 89 26 0067 R      MOV     @ID_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 01D9 8C C0              MOV     AX,ES              ; GET THE GDT DATA SEGMENT
563 01DB BE D8              MOV     DS,AX              ; MOVE THE GDT SEGMENT POINTER TO (DS)
564 01DD BA F4              MOV     DH,AH              ; BUILD HIGH BYTE OF THE 24 BIT ADDRESS
565 01DF C0 EE 04           SHR     DH,4               ; USE ONLY HIGH NIBBLE SHIFT - RIGHT 4
566 01E2 C1 E0 04           SHL     AX,4               ; STRIP HIGH NIBBLE FROM (AX)
567 01E5 03 C6              ADD     AX,SI              ; ADD THE GDT OFFSET TO DEVELOP LOW WORD
568 01E7 80 D6 00           ADC     DH,0               ; ADJUST HIGH BYTE IF CARRY FROM LOW
569
570 ;----- SET THE GDT_LOC
571
572 01EA C7 44 08 FFFF      MOV     [SI].CGDT_LOC,SEG_LIMIT,MAX_SEG_LEN ; POINT (DS) TO USERS CONTROL BLOCK
573 01EF 89 44 0A           MOV     [SI].CGDT_LOC,BASE_LO_WORD,AX      ; SET THE LOW WORD
574 01F2 88 74 0A           MOV     [SI].CGDT_LOC,BASE_HI_BYTE,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 ; POINT (DS) TO USERS CONTROL BLOCK
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_BYTE,CSEG_HI  ; HIGH BYTE OF (CS) = 0FH
582 0208 C6 44 25 9B      MOV     [SI].BIOS_CS,DATA_ACC_RIGHTS,CPLD_CODE_ACCESS ; CPLD_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) - 1 (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 0218 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_BYTE,DH      ; SET THE HIGH BYTE
597 0226 C6 44 2D 93      MOV     [SI].TEMP_SS,DATA_ACC_RIGHTS,CPLD_CODE_ACCESS ; SET CPLD 0
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 80 03              MOV     AL,03H             ; SET THE ERROR FLAG IF NOT
607 0235 E6 80              OUT     MFG_PORT,AL        ; SET THE ERROR FLAG IF NOT
608 0237 EB 51              JMP     SHORT SHUT9        ; EARLY ERROR EXIT
609
610 ;----- SET SHUTDOWN RETURN ADDRESS AND DISABLE NMI
611 0239
612 0239 BB 09BF          BL4:  MOV     AX,9*H+C805_SHUT_DOWN+NMI ; SET THE SHUTDOWN BYTE LOCATION
613 023C E8 0000 E        CALL    CMOS_WRITE         ; TO SHUT DOWN 9 AND DISABLE NMI
614
615 ;----- CLEAR EXCEPTION ERROR FLAG
616
617 023F 2A C0              SUB     AL,AL              ; CLEAR EXCEPTION ERROR FLAG
618 0241 E6 80              OUT     MFG_PORT,AL        ; SET ERROR FLAG LOCATION TO 0
619
620 ;----- LOAD THE IDT AND GDT
621
622 0243 BD 02C6 R        MOV     BP,OFFSET ROM_IDT_LOC ; LOAD THE IDT
623
624 0246 2E                +     DB     02EH           ; REGISTER FROM THIS AREA
625
626 0247 0F                +     LDT     [BP]           ; REGISTER FROM THIS AREA
627 0248                +     DB     00FH           ; REGISTER FROM THIS AREA
628 0248 BB 5E 00          + ?70001 LABEL BYTE
629 024B                +     MOV     BX,WORD PTR [BP] ; REGISTER FROM THIS AREA
630 0248                +     LABEL BYTE
631 0248 01                +     ORG   OFFSET CS:??70001 ; REGISTER FROM THIS AREA
632 024B                +     DB     001H           ; REGISTER FROM THIS AREA
633
634 024B 0F                +     ORG   OFFSET CS:??70002 ; REGISTER FROM THIS AREA
635
636 024B 0F                +     LGDT  [SI].CGDT_LOC    ; LOAD GLOBAL DESCRIPTOR TABLE REGISTER
637 024C                +     DB     00FH           ; REGISTER FROM THIS AREA
638 024C BB 54 08          + ?70003 LABEL BYTE
639 024F                +     MOV     DX,WORD PTR [SI].CGDT_LOC ; REGISTER FROM THIS AREA
640 024C 01                +     LABEL BYTE
641 024F                +     ORG   OFFSET CS:??70003 ; REGISTER FROM THIS AREA
642
643 024C 01                +     DB     001H           ; REGISTER FROM THIS AREA
644
645 ;----- SWITCH TO VIRTUAL MODE
646
647 024F BB 0001          MOV     AX,VIRTUAL_ENABLE ; MACHINE STATUS WORD NEEDED TO
648
649 0252 0F 01 F0          +     LMSW  AX              ; SWITCH TO VIRTUAL MODE
650
651 0255 EA                +     DB     00FH,001H,0F0H ; PURGE PRE-FETCH QUEUE WITH FAR JUMP
652 0256 025A R          +     DW     0EAH           ; TO OFFSET
653 0258 0020             +     DW     BIOS_CS       ; - IN SEGMENT -PROTECTED MODE SELECTOR
654
655 VIRT:
656
657 ;----- IN PROTECTED MODE - SETUP STACK SELECTOR AND SOURCE/TARGET SELECTORS
658
659 025A BB 0028          MOV     AX,TEMP_SS        ; USER'S SS+SP IS NOT A DESCRIPTOR
660 025D BE C0             MOV     SS,AX              ; LOAD STACK SELECTOR
661 025F BB 0010          MOV     AX,SOURCE         ; GET THE SOURCE ENTRY
662 0262 BE D8             MOV     DS,AX              ; LOAD SOURCE SELECTOR
663 0264 BB 0018          MOV     AX,TARGET         ; GET THE TARGET ENTRY
664 0267 BE C0             MOV     ES,AX              ; LOAD TARGET SELECTOR
665 0269 2B F6             SUB     SI,SI              ; SET SOURCE INDEX REGISTER TO ZERO
666 026B 2B FF             SUB     DI,DI              ; SET TARGET INDEX REGISTER TO ZERO
667
668 026D F3/ A5           REP     MOVSB              ; MOVE THE BLOCK COUNT PASSED IN (CX)
669
670 ;----- 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 74 12      JZ      DONE1         ; GO IF NO PARITY ERROR
672
673 ;----- CLEAR PARITY BEFORE SHUTDOWN
674
675 0275 8B 05      MOV     AX,DS:[DI]    ; FETCH CURRENT SOURCE DATA
676 0277 89 05      MOV     DI,[DI],AX    ; WRITE IT BACK
677 0279 B0 01      MOV     AL,01         ; SET PARITY CHECK ERROR = 01
678 027B E6 80      OUT     MFG_PORT,AL   ;
679 027D E4 61      IN      AL,PORT_B     ;
680 027F 0C 0C      OR      AL,RAM_PAR_OFF; TOGGLE PARITY CHECK LATCHES
681 0281 E6 61      OUT     PORT_B,AL     ; 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 ;----- CAUSE A SHUTDOWN
686
687 0287          DONE1:
688 0287 E9 0000 E   JMP     PROC_SHUTDOWN ; GO RESET PROCESSOR AND SHUTDOWN
689
690 ;-----
691 ;=====
692 ;=====
693 028A          SHUT9:
694 ;=====
695 ;=====
696 ;=====
697 ;=====
698 ;=====
699 ;=====
700
701 0291 B4 DD      MOV     AH,DISABLE_BIT20 ; DISABLE MASK
702 0293 E8 03CC R  CALL    GATE_A20        ; GATE ADDRESS 20 LINE OFF
703 0295 C3 00 00  CMP     AL,0           ; COMMAND ACCEPTED?
704 0297 74 0A      JZ      DONE3         ; GO IF YES
705
706 02A0 E4 80      IN      AL,MFG_PORT   ; CHECK FOR ANY OTHER ERROR FIRST
707 02A2 C3 00 00  CMP     AL,0           ; WAS THERE AN ERROR?
708 02A4 75 04      JNZ     DONE3         ; REPORT FIRST ERROR IF YES
709 02A6 B0 03      MOV     AL,03H        ; ELSE SET GATE A20 ERROR FLAG
710 02A8 E6 80      OUT     MFG_PORT,AL   ;
711
712 ;----- RESTORE THE USERS REGISTERS AND SET RETURN CODES
713
714 02AA          DONE3:
715 02AA B8 000D    MOV     CX,CMOS_REG_D  ; CLEAR (AH) TO ZERO AND (AL) TO DEFAULT
716 02AC E6 70      OUT     CMOS_PORT,AL  ; ENABLE NMI INTERRUPTS
717
718 02AF 1F         POP     DS             ; RESTORE USER DATA SEGMENT
719 02B0 07         POP     ES             ; RESTORE USER EXTRA SEGMENT
720 02B1 E4 80      IN      AL,MFG_PORT   ; GET THE ENDING STATUS RETURN CODE
721 02B3 8B EC      MOV     BP,SP          ; POINT TO REGISTERS IN THE STACK
722 02B5 88 46 0F  MOV     [BP+15],AL    ; PLACE ERROR CODE INTO STACK AT (AH)
723 02B7 3A E0      CMP     AH,AL         ; SET THE ZF & CY FLAGS WITH RETURN CODE
724 02B9 61 03      POPA   AH,AL         ; RESTORE THE GENERAL PURPOSE REGISTERS
725 02BB FA         POPA   ST1           ; TURN INTERRUPTS ON
726 02BD C3 00 00  DNE4:  PROC     FAR        ;
727 02BF RET         RET     2           ; RETURN WITH FLAGS SET -- (AH)=CODE
728 02C1          DNE4:  ENDP        ; (CY=0,ZF=1)=OK (CY=1,ZF=0)=ERROR
729 02C3
730
731 ;----- BLOCK MOVE EXCEPTION INTERRUPT HANDLER
732
733 02BF          EX_INT:
734 02BF B0 02      MOV     AL,02H        ; GET EXCEPTION ERROR CODE
735 02C1 E6 80      OUT     MFG_PORT,AL  ; SET EXCEPTION INTERRUPT OCCURRED FLAG
736 02C3 E9 0000 E JMP     PROC_SHUTDOWN ; CAUSE A EARLY SHUTDOWN
737
738 ;----- ROM IDT LOCATION
739
740 02C6          ROM_IDT_LOC:
741 02C6 0100      DW     ROM_IDT_END-ROM_IDT ; LENGTH OF ROM IDT TABLE
742 02C8 C2CC R   DW     ROM_IDT         ; LOW WORD OF BASE ADDRESS
743 02CA 0F       DB     CSEG@_HI       ; HIGH BYTE OF BASE ADDRESS
744 02CB 00       DB     0              ; RESERVED
745
746 ;----- THE ROM EXCEPTION INTERRUPT VECTOR GATES FOR BLOCK MOVE
747
748 02CC          ROM_IDT:
749 02CC 02BF R   DW     EX_INT         ; EXCEPTION 00
750 02CE 0020    DW     BIOS_CS        ; DESTINATION OFFSET
751 02D0 00      DB     0              ; DESTINATION SEGMENT SELECTOR
752 02D1 87      DB     TRAP_GATE     ; WORD COPY COUNT
753 02D2 0000    DW     0              ; GATE TYPE - ACCESS RIGHTS BYTE
754 ;-----
755 02D4 02BF R   DW     EX_INT         ; EXCEPTION 01
756 02D6 0020    DW     BIOS_CS        ; DESTINATION OFFSET
757 02D8 00      DB     0              ; DESTINATION SEGMENT SELECTOR
758 02D9 87      DB     TRAP_GATE     ; WORD COPY COUNT
759 02DA 0000    DW     0              ; GATE TYPE - ACCESS RIGHTS BYTE
760 ;-----
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 87      DB     TRAP_GATE     ; WORD COPY COUNT
765 02E2 0000    DW     0              ; GATE TYPE - ACCESS RIGHTS BYTE
766 ;-----
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 87      DB     TRAP_GATE     ; WORD COPY COUNT
771 02EA 0000    DW     0              ; GATE TYPE - ACCESS RIGHTS BYTE
772 ;-----
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 02F1 87      DB     TRAP_GATE     ; WORD COPY COUNT
777 02F2 0000    DW     0              ; 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              ; DESTINATION SEGMENT SELECTOR

```

SECTION 5

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

```

896 0391 87          DB   TRAP_GATE          ; GATE TYPE - ACCESS RIGHTS BYTE
897 0392 0000       DW   0                  ; RESERVED
898                ; EXCEPTION 25
899 0394 02BF R     DW   EX_INT             ; DESTINATION OFFSET
900 0396 0020       DW   BIOS_CS           ; DESTINATION SEGMENT SELECTOR
901 0398 00         DB   0                 ; WORD COPY COUNT
902 0399 87        DB   TRAP_GATE         ; GATE TYPE - ACCESS RIGHTS BYTE
903 039A 0000       DW   0                 ; RESERVED
904                ; EXCEPTION 26
905 039C 02BF R     DW   EX_INT             ; DESTINATION OFFSET
906 039E 0020       DW   BIOS_CS           ; DESTINATION SEGMENT SELECTOR
907 03A0 00         DB   0                 ; WORD COPY COUNT
908 03A1 87        DB   TRAP_GATE         ; GATE TYPE - ACCESS RIGHTS BYTE
909 03A2 0000       DW   0                 ; RESERVED
910                ; EXCEPTION 27
911 03A4 02BF R     DW   EX_INT             ; DESTINATION OFFSET
912 03A6 0020       DW   BIOS_CS           ; DESTINATION SEGMENT SELECTOR
913 03A8 00         DB   0                 ; WORD COPY COUNT
914 03A9 87        DB   TRAP_GATE         ; GATE TYPE - ACCESS RIGHTS BYTE
915 03AA 0000       DW   0                 ; RESERVED
916                ; EXCEPTION 28
917 03AC 02BF R     DW   EX_INT             ; DESTINATION OFFSET
918 03AE 0020       DW   BIOS_CS           ; DESTINATION SEGMENT SELECTOR
919 03B0 00         DB   0                 ; WORD COPY COUNT
920 03B1 87        DB   TRAP_GATE         ; GATE TYPE - ACCESS RIGHTS BYTE
921 03B2 0000       DW   0                 ; RESERVED
922                ; EXCEPTION 29
923 03B4 02BF R     DW   EX_INT             ; DESTINATION OFFSET
924 03B6 0020       DW   BIOS_CS           ; DESTINATION SEGMENT SELECTOR
925 03B8 00         DB   0                 ; WORD COPY COUNT
926 03B9 87        DB   TRAP_GATE         ; GATE TYPE - ACCESS RIGHTS BYTE
927 03BA 0000       DW   0                 ; RESERVED
928                ; EXCEPTION 30
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                ; EXCEPTION 31
935 03C4 02BF R     DW   EX_INT             ; DESTINATION OFFSET
936 03C6 0020       DW   BIOS_CS           ; DESTINATION SEGMENT SELECTOR
937 03C8 00         DB   0                 ; WORD COPY COUNT
938 03C9 87        DB   TRAP_GATE         ; GATE TYPE - ACCESS RIGHTS BYTE
939 03CA 0000       DW   0                 ; RESERVED
940 03CC                ROM_IDT_END:
941
942 03CC                BLOCKMOVE  ENDP
  
```

```

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 MODE.
949 ; IT SHOULD BE GATED OFF AFTER ENTERING REAL MODE FROM PROTECTED
950 ; MODE. INTERRUPTS ARE LEFT DISABLED ON EXIT.
951 ; INPUT
952 ; (AH) = 00H ADDRESS BIT 20 GATE OFF. (A20 ALWAYS ZERO)
953 ; (AH) = 0FH 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 PROC
959 03CC 51 PUSH CX ; SAVE USERS (CX)
960 03CD FA CLI ; 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 B0 D1 MOV AL,0D1H ; 8042 COMMAND TO WRITE OUTPUT PORT
964 03D5 E6 64 OUT STATUS_PORT,AL ; OUTPUT COMMAND TO 8042
965 03D7 E8 03E5 R CALL EMPTY_8042 ; WAIT FOR 8042 TO ACCEPT COMMAND
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 03DE 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:
974 03E3 59 POP CX ; RESTORE USERS (CX)
975 03E4 C3 RET
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:
987 03E5 2B C9 SUB CX,CX ; (CX)=0, WILL BE USED AS TIME OUT VALUE
988 03E7
989 03E7 E4 64 IN AL,STATUS_PORT ; READ 8042 STATUS PORT
990 03E9 24 02 AND AL,INPT_BUF_FULL ; TEST INPUT BUFFER FULL FLAG (BIT 1)
991 03EB E0 FA LOOPNZ EMPTY_L ; LOOP UNTIL BUFFER EMPTY OR TIME OUT
992 03ED C3 RET
993 03EE
994
995 GATE_A20 RET ENDP
996
997 ;--- INT 15 H -- ( FUNCTION 88 H - I/O MEMORY SIZE DETERMINE ) -----
998 ; EXT_MEMORY
999 ; THIS ROUTINE RETURNS THE AMOUNT OF MEMORY IN THE SYSTEM THAT IS
1000 ; LOCATED STARTING AT THE 1024K ADDRESSING RANGE, AS DETERMINED BY
1001 ; THE POST ROUTINES.
1002 ; NOTE THAT THE SYSTEM MAY NOT BE ABLE TO USE I/O MEMORY UNLESS THERE
1003 ; IS A FULL COMPLEMENT OF 512K OR 640 BYTES ON THE PLANAR. THIS SIZE
1004 ; SIZE IS STORED IN CMOS AT ADDRESS LOCATIONS 30H AND 31H.
1005 ; INPUT
1006 ; AH = 88H
1007 ;
1008 ; THE I/O MEMORY SIZE VARIABLE IS SET DURING POWER ON
1009 ; DIAGNOSTICS ACCORDING TO THE FOLLOWING ASSUMPTIONS:
1010 ;
1011 ; 1. ALL INSTALLED MEMORY IS FUNCTIONAL.
1012 ; 2. ALL MEMORY FROM 0 TO 640K MUST BE CONTIGUOUS.
1013 ;
1014 ; OUTPUT
1015 ; (AX) = NUMBER OF CONTIGUOUS 1K BLOCKS OF MEMORY A
1016 ; AVAILABLE STARTING AT ADDRESS 1024K.
1017 -----
1018 1019 03EE EXT_MEMORY PROC
1020
1021 03EE B8 3031 MOV AX,CMOS_U_M_S_LO*H+CMOS_U_M_S_HI ; ADDRESS HIGH/LOW BYTES
1022 03F1 E8 0000 E CALL CMOS_READ ; GET THE HIGH BYTE OF I/O MEMORY
1023 03F4 86 C4 XCHG AL,AH ; PUT HIGH BYTE IN POSITION (AH)
1024 03F6 E8 0000 E CALL CMOS_READ ; GET THE LOW BYTE OF I/O MEMORY
1025 03F9 CF IRET ; RETURN TO USER
1026
1027 03FA EXT_MEMORY ENDP

```

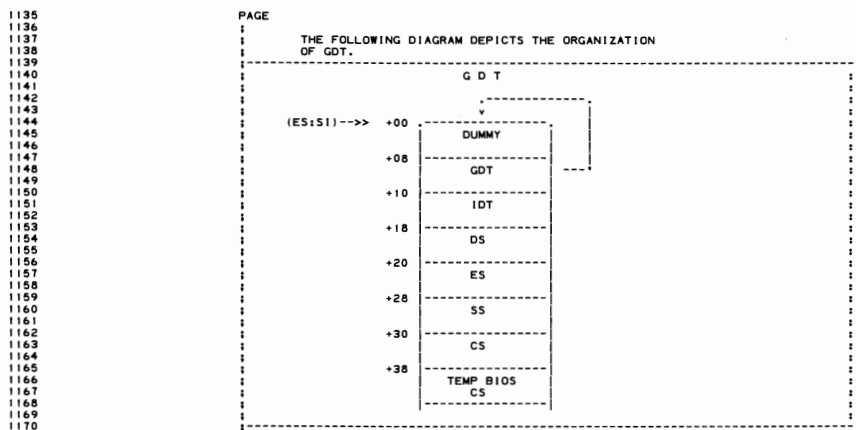


```

1028 PAGE
1029 ;--- INT 15 H ( FUNCTION 89 H ) -----
1030 ; PURPOSE:
1031 ; THIS BIOS FUNCTION PROVIDES A MEANS TO THE USER TO SWITCH INTO
1032 ; VIRTUAL (PROTECTED) MODE. UPON COMPLETION OF THIS FUNCTION THE
1033 ; PROCESSOR WILL BE IN VIRTUAL (PROTECTED) MODE AND CONTROL WILL
1034 ; BE TRANSFERRED TO THE CODE SEGMENT THAT WAS SPECIFIED BY THE USER.
1035 ;
1036 ; ENTRY REQUIREMENTS:
1037 ;
1038 ; (ES:SI) POINTS TO A DESCRIPTOR TABLE (GDT) BUILT BEFORE INTERRUPTING
1039 ; TO THIS FUNCTION. THESE DESCRIPTORS ARE USED BY THIS FUNCTION TO
1040 ; INITIALIZE THE IDTR, THE GDTR AND THE STACK SEGMENT SELECTOR, THE
1041 ; DATA SEGMENT (DS) SELECTOR AND THE EXTRA SEGMENT (ES) SELECTOR WILL
1042 ; BE INITIALIZED TO DESCRIPTORS BUILT BY THE ROUTINE USING THIS FUNCTION.
1043 ; BH - OFFSET INTO THE INTERRUPT DESCRIPTOR TABLE STATING WHERE THE
1044 ; FIRST EIGHT HARDWARE INTERRUPTS WILL BEGIN. ( INTERRUPT LEVEL 1 )
1045 ; BL - OFFSET INTO THE INTERRUPT DESCRIPTOR TABLE STATING WHERE THE
1046 ; SECOND EIGHT HARDWARE INTERRUPTS BEGIN. ( INTERRUPT LEVEL 2 )
1047 ;
1048 ; THE DESCRIPTORS ARE DEFINED AS FOLLOWS:
1049 ;
1050 ; 1. THE FIRST DESCRIPTOR IS THE REQUIRED DUMMY.
1051 ; (USER INITIALIZED TO 0)
1052 ; 2. THE SECOND DESCRIPTOR POINTS TO THE GDT TABLE AS
1053 ; A DATA SEGMENT.
1054 ; (USER INITIALIZED)
1055 ; 3. THE THIRD DESCRIPTOR POINTS TO THE USER DEFINED
1056 ; INTERRUPT DESCRIPTOR TABLE (IDT).
1057 ; (USER INITIALIZED)
1058 ; 4. THE FORTH DESCRIPTOR POINTS TO THE USER'S DATA
1059 ; SEGMENT (DS).
1060 ; (USER INITIALIZED)
1061 ; 5. THE FIFTH DESCRIPTOR POINTS TO THE USER'S EXTRA
1062 ; SEGMENT (ES).
1063 ; (USER INITIALIZED)
1064 ; 6. THE SIXTH DESCRIPTOR POINTS TO THE USER'S STACK
1065 ; SEGMENT (SS).
1066 ; (USER INITIALIZED)
1067 ; 7. THE SEVENTH DESCRIPTOR POINTS TO THE CODE SEGMENT
1068 ; THAT THIS FUNCTION WILL RETURN TO.
1069 ; (USER INITIALIZED TO THE USER'S CODE SEGMENT.)
1070 ; 8. THE EIGHTH DESCRIPTOR IS USED BY THIS FUNCTION TO
1071 ; ESTABLISH A CODE SEGMENT FOR ITSELF. THIS IS
1072 ; NEEDED SO THAT THIS FUNCTION CAN COMPLETE IT'S
1073 ; EXECUTION WHILE IN PROTECTED MODE. WHEN CONTROL
1074 ; GETS PASSED TO THE USER'S CODE THIS DESCRIPTOR CAN
1075 ; BE USED BY HIM IN ANY WAY HE CHOOSES.
1076 ;
1077 ; NOTE - EACH DESCRIPTOR MUST CONTAIN ALL THE NECESSARY DATA
1078 ; I.E. THE LIMIT, BASE ADDRESS AND THE ACCESS RIGHTS BYTE.
1079 ;
1080 ; AH= 89H (FUNCTION CALL)
1081 ; ES:SI = LOCATION OF THE GDT TABLE BUILT BY ROUTINE
1082 ; USING THIS FUNCTION.
1083 ;
1084 ; EXIT PARAMETERS:
1085 ;
1086 ; AH = 0 IF SUCCESSFUL
1087 ; ALL SEGMENT REGISTERS ARE CHANGED, (AX) AND (BP) DESTROYED
1088 ;
1089 ; CONSIDERATIONS:
1090 ;
1091 ; 1. NO BIOS AVAILABLE TO USER. USER MUST HANDLE ALL
1092 ; I/O COMMANDS.
1093 ; 2. INTERRUPTS - INTERRUPT VECTOR LOCATIONS MUST BE
1094 ; MOVED, DUE TO THE 286 RESERVED AREAS. THE
1095 ; HARDWARE INTERRUPT CONTROLLERS MUST BE REINITIALIZED
1096 ; TO DEFINE LOCATIONS THAT DO NOT RESIDE IN THE 286
1097 ; RESERVED AREAS.
1098 ; 3. EXCEPTION INTERRUPT TABLE AND HANDLER MUST BE
1099 ; INITIALIZED BY THE USER.
1100 ; 4. THE INTERRUPT DESCRIPTOR TABLE MUST NOT OVERLAP
1101 ; THE REAL MODE BIOS INTERRUPT DESCRIPTOR TABLE.
1102 ; 5. THE FOLLOWING GIVES AN IDEA OF WHAT THE USER CODE
1103 ; SHOULD LOOK LIKE WHEN INVOKING THIS FUNCTION.
1104 ;
1105 ; REAL MODE ----> "USER CODE"
1106 ; " MOV AX,GDT SEGMENT
1107 ; " MOV ES,AX
1108 ; " MOV SI,GDT OFFSET
1109 ; " MOV BH,HARDWARE INT LEVEL 1 OFFSET
1110 ; " MOV BL,HARDWARE INT LEVEL 2 OFFSET
1111 ; " MOV AH,89H
1112 ; " INT 15H
1113 ; VIRTUAL MODE ----> "USER CODE"
1114 ;
1115 ; DESCRIPTION:
1116 ;
1117 ; 1. CLI (NO INTERRUPTS ALLOWED) WHILE THIS FUNCTION IS EXECUTING.
1118 ; 2. ADDRESS LINE 20 IS GATED ACTIVE.
1119 ; 3. THE CURRENT USER STACK SEGMENT DESCRIPTOR IS INITIALIZED.
1120 ; THE GDTR IS LOADED WITH THE GDT BASE ADDRESS.
1121 ; 5. THE IDTR IS LOADED WITH THE IDT BASE ADDRESS.
1122 ; 6. THE 8259 IS REINITIALIZED WITH THE NEW INTERRUPT OFFSETS.
1123 ; 7. THE PROCESSOR IS PUT IN VIRTUAL MODE WITH THE CODE
1124 ; SEGMENT DESIGNATED FOR THIS FUNCTION.
1125 ; 8. DATA SEGMENT IS LOADED WITH THE USER DEFINED
1126 ; SELECTOR FOR THE DS REGISTER.
1127 ; 9. EXTRA SEGMENT IS LOADED WITH THE USER DEFINED
1128 ; SELECTOR FOR THE ES REGISTER.
1129 ; 10. STACK SEGMENT IS LOADED WITH THE USER DEFINED
1130 ; SELECTOR FOR THE SS REGISTER.
1131 ; 11. CODE SEGMENT DESCRIPTOR SELECTOR VALUE IS
1132 ; SUBSTITUTED ON THE STACK FOR RETURN TO USER.
1133 ; 12. WE TRANSFER CONTROL TO THE USER WITH INTERRUPTS DISABLED.
1134 ;

```

SECTION 5



1171 |
1172 | THE GLOBAL DESCRIPTOR TABLE (ACTUAL LOCATION POINTED TO BY ES:SI) |
1173 |-----|-----|-----|-----|-----|-----|-----|-----|-----|
1174 |
1175 |

```

1176 VIRTUAL_ENABLE_GDT_DEF STRUC
1177         DQ ?
1178 GDTPTR  DQ ?
1179 IDTPTR  DQ ?
1180 USER_DS DQ ?
1181 USER_ES DQ ?
1182 USER_SS DQ ?
1183 USER_CS DQ ?
1184 BIO_CS  DQ ?
1185 VIRTUAL_ENABLE_GDT_DEF ENDS

```

```

1186         ASSUME DS:DATA
1187
1188 X_VIRTUAL PROC FAR
1189     SET_VMODE:
1190
1191     I----- ENABLE ADDRESS LATCH BIT 20
1192
1193     1194 03FA FA          CLI          ; NO INTERRUPTS ALLOWED
1195     1195 03FB B4 DF      MOV     AH,ENABLE_BIT20 ; ENABLE BIT 20 FOR ADDRESS GATE
1196     1196 03FD E8 03CC R  CALL    GATE_A20
1197     1197 0400 3C 00      CMP     AL,0
1198     1198 0402 74 04      JZ     BIT20_ON ; WAS THE COMMAND ACCEPTED?
1199     1199 0404 B4 FF      MOV     AH,OFFH ; GO IF YES
1200     1200 0406 F9        STC     ; SET THE ERROR FLAG
1201     1201 0407 CF        IRET    ; SET CARRY
1202     ; EARLY EXIT

```

```

1203
1204     BIT20_ON:
1205     1205 0408 06        PUSH    ES ; MOVE SEGMENT POINTER
1206     1206 0409 1F        POP     DS ; TO THE DATA SEGMENT

```

1207 |-----|-----|-----|-----|-----|-----|-----|-----|-----|
1208 | REINITIALIZE THE 8259 INTERRUPT CONTROLLER #1 TO THE USER SPECIFIED OFFSET |
1209 |-----|-----|-----|-----|-----|-----|-----|-----|-----|
1210 |
1211 |

```

1212     1212 040A B0 11      MOV     AL,11H ; START INITIALIZATION SEQUENCE-ICW1
1213     1213 040C E6 20      OUT    INTA00,AL ; EDGE, INTERVAL-8,MASTER,ICW4 NEEDED
1214     1214 040E EB 00      JMP     $+2
1215     1215 0410 8A C7      MOV     AL,BH ; HARDWARE INT'S START AT INT # (BH)
1216     1216 0412 E6 21      OUT    INTA01,AL ; SEND ICW2
1217     1217 0414 EB 00      JMP     $+2
1218     1218 0416 B0 04      MOV     AL,04H ; SEND ICW3 - MASTER LEVEL 2
1219     1219 0418 E6 21      OUT    INTA01,AL
1220     1220 041A EB 00      JMP     $+2
1221     1221 041C B0 01      MOV     AL,01H ; SEND ICW4 - MASTER,8086 MODE
1222     1222 041E E6 21      OUT    INTA01,AL
1223     1223 0420 EB 00      JMP     $+2
1224     1224 0422 B0 FF      MOV     AL,OFFH ; MASK OFF ALL INTERRUPTS
1225     1225 0424 E6 21      OUT    INTA01,AL

```

1226 |-----|-----|-----|-----|-----|-----|-----|-----|-----|
1227 | REINITIALIZE THE 8259 INTERRUPT CONTROLLER #2 TO THE USER SPECIFIED OFFSET |
1228 |-----|-----|-----|-----|-----|-----|-----|-----|-----|
1229 |
1230 |

```

1231     1231 0426 B0 11      MOV     AL,11H ; INITIALIZE SEQUENCE-ICW1 FOR SLAVE
1232     1232 0428 E6 A0      OUT    INTB00,AL ; EDGE, INTERVAL-8,MASTER,ICW4 NEEDED
1233     1233 042A EB 00      JMP     $+2
1234     1234 042C 8A C3      MOV     AL,BL ; HARDWARE INT'S START AT INT # (BL)
1235     1235 042E E6 A1      OUT    INTB01,AL ; SEND ICW2
1236     1236 0430 B0 02      MOV     AL,02H
1237     1237 0432 EB 00      JMP     $+2
1238     1238 0434 E6 A1      OUT    INTB01,AL ; SEND ICW3 - SLAVE LEVEL 2
1239     1239 0436 EB 00      JMP     $+2
1240     1240 0438 B0 01      MOV     AL,01H ; SEND ICW4 - SLAVE,8086 MODE
1241     1241 043A E6 A1      OUT    INTB01,AL
1242     1242 043C EB 00      JMP     $+2
1243     1243 043E B0 FF      MOV     AL,OFFH ; MASK OFF ALL INTERRUPTS
1244     1244 0440 E6 A1      OUT    INTB01,AL

```

1245 |-----|-----|-----|-----|-----|-----|-----|-----|-----|
1246 | SETUP BIOS CODE SEGMENT DESCRIPTOR |
1247 |-----|-----|-----|-----|-----|-----|-----|-----|-----|
1248 |

```

1249
1250 0442 C7 44 38 FFFF      MOV     [SI],BIO_CS.SEG_LIMIT,MAX_SEG_LEN      ; SET LENGTH
1251 0447 C6 44 3C 0F      MOV     [SI],BIO_CS.BASE_HI_BYTE,CSEG00_HI    ; SET HIGH BYTE OF CS=0F
1252 0448 C7 44 3A 0000    MOV     [SI],BIO_CS.BASE_LO_WORD,CSEG00_LO    ; SET LOW WORD OF CS=0
1253 0450 C6 44 3D 9B      MOV     [SI],BIO_CS.DATA_ACC_RIGHTS,CPL0_CODE_ACCESS
1254 0454 C7 44 3E 0000    MOV     [SI],BIO_CS.DATA_RESERVED,0          ; ZERO RESERVED AREA
1255
1256
1257      ; -----
1258      ; ENABLE PROTECTED MODE
1259      ; -----
1259          LGDT     [SI],GDTPTR                ; LOAD GLOBAL DESCRIPTOR TABLE REGISTER
1260          DB      00FH
1261 045A          + ????05 LABEL BYTE
1262 045A 8B 54 08      MOV     DX,WORD PTR [SI],GDTPTR
1263 045D          + ????06 LABEL BYTE
1264 045A          + ORG     OFFSET CS:????05
1265 045A 01          + DB      001H
1266 045D          + ORG     OFFSET CS:????06
1267          LIDT     [SI],IDTPTR                ; INTERRUPT DESCRIPTOR TABLE REGISTER
1268          DB      00FH
1269 045E          + ????07 LABEL BYTE
1270 045E 8B 54 10      MOV     BX,WORD PTR [SI],IDTPTR
1271 0461          + ????08 LABEL BYTE
1272 045E          + ORG     OFFSET CS:????07
1273 045E 01          + DB      001H
1274 0461          + ORG     OFFSET CS:????08
1275
1276 0461 8B 0001      MOV     AX,VIRTUAL_ENABLE                    ; MACHINE STATUS WORD NEEDED TO
1277          LMSW   AX                               ; SWITCH TO VIRTUAL MODE
1278 0464 0F 01 F0      + DB      00FH,001H,0F0H
1279 0467 EA          DB      0E4H                                ; PURGE PRE-FETCH QUEUE WITH FAR JUMP
1280 0468 046C R      DW      OFFSET VMODE                        ; - TO OFFSET
1281 046A 0038          DW      BIO_CS                               ; - IN SEGMENT -PROTECTED MODE SELECTOR
1282
1283 046C          VMODE:
1284          ; -----
1285          ; SETUP USER SEGMENT REGISTERS
1286          ; -----
1287 046C 8B 0018      MOV     AX,USER_DS                          ; SETUP USER'S DATA SEGMENT
1288 046F 8E D8          MOV     DS,AX                               ; TO PROTECTED MODE SELECTORS
1289 0471 8B 0020      MOV     AX,USER_ES                          ; SETUP USER'S EXTRA SEGMENT
1290 0474 8E C0          MOV     ES,AX
1291 0476 8B 0028      MOV     AX,USER_SS                          ; SETUP USER'S STACK SEGMENT
1292 0479 8E D0          MOV     SS,AX
1293
1294          ; -----
1295          ; PUT TRANSFER ADDRESS ON STACK
1296          ; AND RETURN TO THE USER
1297          ; -----
1297 047B 8B          POP     BX                                  ; GET RETURN IP FROM THE STACK
1298 047C 83 C4 04      ADD     SP,4                                ; NORMALIZE STACK POINTER
1299 047F 6A 30          PUSH   USER_CS                             ; SET STACK FOR A RETURN FAR
1300 0481 53          PUSH   BX
1301 0482 CB          RET                                         ; RETURN TO USER IN VIRTUAL MODE
1302
1303 0483          X_VIRTUAL      ENDP
1304
1305          ; --- DEVICE BUSY AND INTERRUPT COMPLETE ---
1306          ;
1307          ; THIS ROUTINE IS A TEMPORARY HANDLER FOR DEVICE BUSY
1308          ; AND INTERRUPT COMPLETE
1309          ;
1310          ; INPUT - SEE PROLOGUE
1311          ; -----
1312
1313 0483          DEVICE_BUSY PROC NEAR
1314 0483 F8          CLC                                       ; TURN CARRY OFF
1315 0484 E9 0057 R    JMP     C1_F                               ; RETURN WITH CARRY FLAG
1316 0487          DEVICE_BUSY ENDP
1317
1318 0487          INT_COMPLETE PROC NEAR
1319 0487 CF          IRET                                    ; RETURN
1320 0488          INT_COMPLETE ENDP
1321
1322 0488          CODE      ENDS
1323          END
  
```

```

1 PAGE 118,121
2 TITLE BIOS2 ---- 06/10/85 BIOS INTERRUPT ROUTINES
3 .286C
4 .LIST
5 0000 CODE SEGMENT BYTE PUBLIC
6
7 PUBLIC PRINT_SCREEN_1
8 PUBLIC RTC_INT
9 PUBLIC TIME_OF_DAY_1
10 PUBLIC TIMER_INT_1
11
12 EXTRN CMOS_READ:NEAR ; READ CMOS LOCATION ROUTINE
13 EXTRN CMOS_WRITE:NEAR ; WRITE CMOS LOCATION ROUTINE
14 EXTRN DDS:NEAR ; LOAD (DS) WITH DATA SEGMENT SELECTOR
15
16
17 ;--- INT 1A H --- (TIME_OF_DAY) -----
18 ; THIS BIOS ROUTINE ALLOWS THE CLOCKS TO BE SET OR READ
19 ;
20 ; PARAMETERS:
21 ; (AH) = 00H READ THE CURRENT CLOCK SETTING AND RETURN WITH,
22 ; (CX) = HIGH PORTION OF COUNT
23 ; (DX) = LOW PORTION OF COUNT
24 ; (AL) = 0 TIMER HAS NOT PASSED 24 HOURS SINCE LAST READ
25 ; 1 IF ON ANOTHER DAY. (RESET TO ZERO AFTER READ)
26 ;
27 ; (AH) = 01H SET THE CURRENT CLOCK USING,
28 ; (CX) = HIGH PORTION OF COUNT
29 ; (DX) = LOW PORTION OF COUNT.
30 ;
31 ; NOTE: COUNTS OCCUR AT THE RATE OF 1193180/65536 COUNTS/SECOND
32 ; (OR ABOUT 18.2 PER SECOND -- SEE EQUATES)
33 ;
34 ; (AH) = 02H READ THE REAL TIME CLOCK AND RETURN WITH,
35 ; (CH) = HOURS IN BCD (00-23)
36 ; (CL) = MINUTES IN BCD (00-59)
37 ; (DH) = SECONDS IN BCD (00-59)
38 ; (DL) = DAYLIGHT SAVINGS ENABLE (00-01).
39 ;
40 ; (AH) = 03H SET THE REAL TIME CLOCK USING,
41 ; (CH) = HOURS IN BCD (00-23)
42 ; (CL) = MINUTES IN BCD (00-59)
43 ; (DH) = SECONDS IN BCD (00-59)
44 ; (DL) = 01 IF DAYLIGHT SAVINGS ENABLE OPTION, ELSE 00.
45 ;
46 ; NOTE: (DL) = 00 IF DAYLIGHT SAVINGS TIME ENABLE IS NOT ENABLED.
47 ; (DL) = 01 ENABLES TWO SPECIAL UPDATES THE LAST SUNDAY IN
48 ; APRIL (1:59:59 --> 3:00:00 AM) AND THE LAST SUNDAY IN
49 ; OCTOBER (1:59:59 --> 1:00:00 AM) THE FIRST TIME.
50 ;
51 ; (AH) = 04H READ THE DATE FROM THE REAL TIME CLOCK AND RETURN WITH,
52 ; (CH) = CENTURY IN BCD (19 OR 20)
53 ; (CL) = YEAR IN BCD (00-99)
54 ; (DH) = MONTH IN BCD (01-12)
55 ; (DL) = DAY IN BCD (01-31).
56 ;
57 ; (AH) = 05H SET THE DATE INTO THE REAL TIME CLOCK USING,
58 ; (CH) = CENTURY IN BCD (19 OR 20)
59 ; (CL) = YEAR IN BCD (00 - 99)
60 ; (DH) = MONTH IN BCD (01 - 12)
61 ; (DL) = DAY IN BCD (01-31).
62 ;
63 ; (AH) = 06H SET THE ALARM TO INTERRUPT AT SPECIFIED TIME,
64 ; (CX) = HOURS IN BCD (00-23 (OR FFH))
65 ; (CL) = MINUTES IN BCD (00-59 (OR FFH))
66 ; (DH) = SECONDS IN BCD (00-59 (OR FFH)).
67 ;
68 ; (AH) = 07H RESET THE ALARM INTERRUPT FUNCTION.
69 ;
70 ; NOTES: FOR ALL RETURNS CY = 0 FOR SUCCESSFUL OPERATION.
71 ; FOR (AH) = 2, 4, 6 - CARRY FLAG SET IF REAL TIME CLOCK NOT OPERATING.
72 ; FOR (AH) = 6 - CARRY FLAG SET IF ALARM ALREADY ENABLED.
73 ; FOR THE ALARM FUNCTION (AH = 6) THE USER MUST SUPPLY A ROUTINE AND
74 ; INTERCEPT THE CORRECT ADDRESS IN THE VECTOR TABLE FOR INTERRUPT 4AH.
75 ; USE OFFH FOR ANY "DO NOT CARE" POSITION FOR INTERVAL INTERRUPTS.
76 ; INTERRUPTS ARE DISABLED DURING DATA MODIFICATION.
77 ; AH & AL ARE RETURNED MODIFIED AND NOT DEFINED EXCEPT WHERE INDICATED.
78 ;-----
79 ASSUME CS:CODE,DS:DATA
80
81 0000 FB "STI" TIME_OF_DAY_1 PROC FAR
82 0001 80 FC 08 CMP AH,(RTC_TBE-RTC_TB)/2 ; INTERRUPTS BACK ON
83 0004 F5 CMC ; CHECK IF COMMAND IN VALID RANGE (0-7)
84 0005 F2 17 JC TIME_9 ; COMPLEMENT CARRY FOR ERROR EXIT
85 ; EXIT WITH CARRY = 1 IF NOT VALID
86 0007 IE PUSH DS ; SAVE USERS (DS) SEGMENT
87 0008 E8 0000 E CALL DDS ; GET DATA SEGMENT SELECTOR
88 000B 56 PUSH SI ; SAVE WORK REGISTER
89 000C C1 E8 08 SHR AX,8 ; CONVERT FUNCTION TO BYTE OFFSET
90 000F 03 CD ADD AX,AX ; CONVERT FUNCTION TO WORD OFFSET (CY=0)
91 0011 8B F0 MOV SI,AX ; PLACE INTO ADDRESSING REGISTER
92 0013 FA CLI ; NO INTERRUPTS DURING TIME FUNCTIONS
93 0014 2E: FF 94 0021 R CALL CS:[SI]+OFFSET RTC_TB ; VECTOR TO FUNCTION REQUESTED WITH CY=0
94 ; RETURN WITH CARRY FLAG SET FOR RESULT
95 0019 FB STI ; INTERRUPTS BACK ON
96 001A B4 00 MOV AH,0 ; CLEAR (AH) TO ZERO
97 001C 5F POP SI ; RECOVER USERS REGISTER
98 001D 1F POP DS ; RECOVER USERS SEGMENT SELECTOR
99 001E CA 0002 TIME_9: RET ; RETURN WITH CY = 0 IF NO ERROR
100
101
102
103 0021 0031 R RTC_TB DW RTC_00 ; ROUTINE VECTOR TABLE (AH) =
104 0023 0042 R DW RTC_10 ; 0 = READ CURRENT CLOCK COUNT
105 0025 0050 R DW RTC_20 ; 1 = SET CLOCK COUNT
106 0027 0075 R DW RTC_30 ; 2 = READ THE REAL TIME CLOCK TIME
107 0029 00A8 R DW RTC_40 ; 3 = SET REAL TIME CLOCK TIME
108 002B 00CB R DW RTC_50 ; 4 = READ THE REAL TIME CLOCK DATE
109 002D 0104 R DW RTC_60 ; 5 = SET REAL TIME CLOCK DATE
110 002F 0145 R DW RTC_70 ; 6 = SET THE REAL TIME CLOCK ALARM
111 = 0031 EQU $ ; 7 = RESET ALARM
112
113 0031 TIME_OF_DAY_1 ENDP

```

```

114                                     PAGE
115 0031                                RTC_00 PROC    NEAR
116 0031 A0 0070 R                      MOV     AL,0TIMER_OFL          ; GET THE OVERFLOW FLAG
117 0034 C6 06 0070 R 00                MOV     0TIMER_OFL,0        ; AND THEN RESET THE OVERFLOW FLAG
118 0039 8B 0E 006E R                    MOV     CX,0TIMER_HIGH      ; GET COUNT OF TIME HIGH WORD
119 003D 8B 16 006C R                    MOV     DX,0TIMER_LOW       ; GET COUNT OF TIME LOW WORD
120 0041 C3                               RET                             ; RETURN WITH NO CARRY
121
122 0042                                RTC_10:
123 0042 89 16 006C R                    MOV     0TIMER_LOW,DX      ; SET TIME COUNT LOW WORD
124 0046 89 0E 006E R                    MOV     0TIMER_HIGH,CX     ; SET THE TIME COUNT HIGH WORD
125 004A C6 06 0070 R 00                MOV     0TIMER_OFL,0        ; RESET OVERFLOW FLAG
126 004F C3                               RET                             ; RETURN WITH NO CARRY
127
128 0050                                RTC_20:
129 0050 E8 016B R                        CALL    UPD_IPR             ; GET RTC TIME
130 0053 72 1F                            JC      RTC_29              ; CHECK FOR UPDATE IN PROCESS
131                                     ; EXIT IF ERROR (CY= 1)
132 0055 80 00                            MOV     AL,CMOS_SECONDS     ; SET ADDRESS OF SECONDS
133 0057 E8 0000 E                        CALL    CMOS_READ          ; GET SECONDS
134 005A 8A F0                            MOV     DH,AL               ; SAVE
135 005C 80 0B                            MOV     AL,CMOS_REG_B       ; ADDRESS ALARM REGISTER
136 005E E8 0000 E                        CALL    CMOS_READ          ; READ CURRENT VALUE OF DSE BIT
137 0061 24 00                            AND     AL,00000001B        ; MASK FOR VALID DSE BIT
138 0063 8A D0                            MOV     DL,AL               ; SET (DL) TO ZERO FOR NO DSE BIT
139 0065 80 02                            MOV     AL,CMOS_MINUTES     ; SET ADDRESS OF MINUTES
140 0067 E8 0000 E                        CALL    CMOS_READ          ; GET MINUTES
141 006A 8A C8                            MOV     CL,AL               ; SAVE
142 006C 80 04                            MOV     AL,CMOS_HOURS       ; SET ADDRESS OF HOURS
143 006E E8 0000 E                        CALL    CMOS_READ          ; GET HOURS
144 0071 8A E8                            MOV     CH,AL               ; SAVE
145 0073 F8                               CLC                           ; SET CY= 0
146 0074                                     ;
147 0074 C3                               RET                             ; RETURN WITH RESULT IN CARRY FLAG
148
149 0075                                RTC_30:
150 0075 E8 016B R                        CALL    UPD_IPR             ; SET RTC TIME
151 0078 73 03                            JNC    RTC_35              ; CHECK FOR UPDATE IN PROCESS
152 007A E8 0154 R                        CALL    RTC_STA             ; GO AROUND IF CLOCK OPERATING
153 007D                                     ; ELSE TRY INITIALIZING CLOCK
154 007D 8A E6                            MOV     AH,DH               ; GET TIME BYTE - SECONDS
155 007F 80 00                            MOV     AL,CMOS_SECONDS     ; ADDRESS SECONDS
156 0081 E8 0000 E                        CALL    CMOS_WRITE         ; UPDATE SECONDS
157 0084 8A E1                            MOV     AH,CL               ; GET TIME BYTE - MINUTES
158 0086 80 02                            MOV     AL,CMOS_MINUTES     ; ADDRESS MINUTES
159 0088 E8 0000 E                        CALL    CMOS_WRITE         ; UPDATE MINUTES
160 008B 8A D0                            MOV     AH,CL               ; GET TIME BYTE - HOURS
161 008D 80 04                            MOV     AL,CMOS_HOURS       ; ADDRESS HOURS
162 008F E8 0000 E                        CALL    CMOS_WRITE         ; UPDATE ADDRESS
163 0092 8B 0B 0000 E                    MOV     AX,XCMOS_REG_B     ; ADDRESS ALARM REGISTER
164 0095 E8 0000 E                        CALL    CMOS_READ          ; READ CURRENT VALUE
165 0098 24 62                            AND     AL,0100010B        ; MASK FOR VALID BIT POSITIONS
166 009A 0A 02                            OR     AL,00000010B        ; TURN ON 24 HOUR MODE
167 009C 80 E2 01                        AND     DL,00000001B        ; USE ONLY THE DSE BIT
168 009F 0A C2                            OR     AL,DL                ; GET DAY LIGHT SAVING TIME BIT (DSE)
169 00A1 86 E0                            XCHG   AH,AL               ; PLACE IN WORK REGISTER AND GET ADDRESS
170 00A3 E8 0000 E                        CALL    CMOS_WRITE         ; SET NEW ALARM BITS
171 00A5 F8                               CLC                           ; SET CY= 0
172 00A7 C3                               RET                             ; RETURN WITH CY= 0
173
174 00A8                                RTC_40:
175 00A8 E8 016B R                        CALL    UPD_IPR             ; GET RTC DATE
176 00AB 72 1D                            JC      RTC_49              ; CHECK FOR UPDATE IN PROCESS
177                                     ; EXIT IF ERROR (CY= 1)
178 00AD 80 07                            MOV     AL,CMOS_DAY_MONTH   ; ADDRESS DAY OF MONTH
179 00AF E8 0000 E                        CALL    CMOS_READ          ; READ DAY OF MONTH
180 00B2 8A D0                            MOV     DL,AL               ; SAVE
181 00B4 80 0B                            MOV     AL,CMOS_MONTH       ; ADDRESS MONTH
182 00B6 E8 0000 E                        CALL    CMOS_READ          ; READ MONTH
183 00B9 8A F0                            MOV     DH,AL               ; SAVE
184 00BB 80 09                            MOV     AL,CMOS_YEAR        ; ADDRESS YEAR
185 00BD E8 0000 E                        CALL    CMOS_READ          ; READ YEAR
186 00C0 8A C8                            MOV     CL,AL               ; SAVE
187 00C2 80 32                            MOV     AL,CMOS_CENTURY     ; ADDRESS CENTURY LOCATION
188 00C4 E8 0000 E                        CALL    CMOS_READ          ; GET CENTURY BYTE
189 00C7 8A E8                            MOV     CH,AL               ; SAVE
190 00C9 F8                               CLC                           ; SET CY=0
191 00CA                                     ;
192 00CA C3                               RET                             ; RETURN WITH RESULTS IN CARRY FLAG
193
194 00CB                                RTC_50:
195 00CB E8 016B R                        CALL    UPD_IPR             ; SET RTC DATE
196 00CE 73 03                            JNC    RTC_55              ; CHECK FOR UPDATE IN PROCESS
197 00D0 E8 0154 R                        CALL    RTC_STA             ; GO AROUND IF NO ERROR
198 00D3                                     ; ELSE INITIALIZE CLOCK
199 00D3 8B 006E R                        MOV     AX,CMOS_DAY_WEEK    ; ADDRESS OF DAY OF WEEK BYTE
200 00D6 E8 0000 E                        CALL    CMOS_WRITE         ; LOAD ZEROS TO DAY OF WEEK
201 00D9 8A 0E                            MOV     AH,DH               ; GET DAY OF MONTH BYTE
202 00DB 80 07                            MOV     AL,CMOS_DAY_MONTH   ; ADDRESS DAY OF MONTH BYTE
203 00DD E8 0000 E                        CALL    CMOS_WRITE         ; WRITE DAY OF MONTH REGISTER
204 00E0 8A E6                            MOV     AH,DH               ; GET MONTH
205 00E2 80 0B                            MOV     AL,CMOS_MONTH       ; ADDRESS MONTH BYTE
206 00E4 E8 0000 E                        CALL    CMOS_WRITE         ; WRITE MONTH REGISTER
207 00E7 8A E1                            MOV     AH,CL               ; GET YEAR BYTE
208 00E9 80 09                            MOV     AL,CMOS_YEAR        ; ADDRESS YEAR REGISTER
209 00EB E8 0000 E                        CALL    CMOS_WRITE         ; WRITE YEAR REGISTER
210 00EE 8A E5                            MOV     AH,CH               ; GET CENTURY BYTE
211 00F0 80 32                            MOV     AL,CMOS_CENTURY     ; ADDRESS CENTURY BYTE
212 00F2 E8 0000 E                        CALL    CMOS_WRITE         ; WRITE CENTURY LOCATION
213 00F5 8B 0B 00B0 R                    MOV     AX,XCMOS_REG_B     ; ADDRESS ALARM REGISTER
214 00F8 E8 0000 E                        CALL    CMOS_READ          ; READ CURRENT SETTINGS
215 00FB 24 7F                            AND     AL,07FH            ; CLEAR - SET BIT
216 00FD 86 E0                            XCHG   AH,AL               ; MOVE TO WORK REGISTER
217 00FF E8 0000 E                        CALL    CMOS_WRITE         ; AND START CLOCK UPDATING
218 0102 F8                               CLC                           ; SET CY= 0
219 0103 C3                               RET                             ; RETURN CY=0
220
221 0104                                RTC_60:
222 0104 80 0B                            MOV     AL,CMOS_REG_B       ; SET RTC ALARM
223 0106 E8 0000 E                        CALL    CMOS_READ          ; ADDRESS ALARM REGISTER
224 0109 A8 20                            TEST   AL,&20H              ; READ ALARM REGISTER
225 010B F9                               STC                             ; CHECK FOR ALARM ALREADY ENABLED
226 010C 75 33                            JNZ    RTC_69              ; SET CARRY IN CASE OF ERROR
227                                     ; ERROR EXIT IF ALARM SET

```

SECTION 5

```

228 010E E8 016B R      CALL   UPD_IPR          ; CHECK FOR UPDATE IN PROCESS
229 0111 73 03          JNC    RTC_65          ; SKIP INITIALIZATION IF NO ERROR
230 0113 E8 0154 R      CALL   RTC_STA         ; ELSE INITIALIZE CLOCK
231 0116
RTC_65:
232 0116 8A E6         MOV    AH,0H           ; GET SECONDS BYTE
233 0118 B0 01         MOV    AL,CMOS_SEC_ALARM ; ADDRESS THE SECONDS ALARM REGISTER
234 011A E8 0000 E      CALL   CMOS_WRITE     ; INSERT SECONDS
235 011D 8A E1         MOV    AH,CL          ; GET MINUTES PARAMETER
236 011F B0 03         MOV    AL,CMOS_MIN_ALARM ; ADDRESS MINUTES ALARM REGISTER
237 0121 E8 0000 E      CALL   CMOS_WRITE     ; INSERT MINUTES
238 0124 8A E5         MOV    AH,CH          ; GET HOURS PARAMETER
239 0126 B0 05         MOV    AL,CMOS_HR_ALARM  ; ADDRESS HOUR ALARM REGISTER
240 0128 E8 0000 E      CALL   CMOS_WRITE     ; INSERT HOURS
241 012B E4 A1         IN    AL,INTB01       ; READ SECOND INTERRUPT MASK REGISTER
242 012D 24 FE         OUT   AL,0F0EH        ; ENABLE ALARM TIMER BIT (CY= 0)
243 012F E6 A1         MOV    INTB01,AL       ; WRITE UPDATED MASK
244 0131 B8 080B E      MOV    AX,*CMOS_REG_B  ; ADDRESS ALARM REGISTER
245 0134 E8 0000 E      CALL   CMOS_READ       ; READ CURRENT ALARM REGISTER
246 0137 24 7F         AND   AL,0F0FH        ; ENSURE SET BIT TURNED OFF
247 0139 C0 20         OR    AL,20H          ; TURN ON ALARM ENABLE
248 013B B6 E0         XCHG  AH,AL           ; MOVE MASK TO OUTPUT REGISTER
249 013D E8 0000 E      CALL   CMOS_WRITE     ; WRITE NEW ALARM MASK
250 0140 F8           CLC                    ; SET CY= 0
251 0141
RTC_69:
252 0141 B8 0000 E      MOV    AX,0           ; CLEAR AX REGISTER
253 0144 C3           RET                    ; RETURN WITH RESULTS IN CARRY FLAG
254
RTC_70:
255 0145
256 0145 B8 080B E      MOV    AX,*CMOS_REG_B  ; ADDRESS ALARM REGISTER (TO BOTH AH,AL)
257 0148 E8 0000 E      CALL   CMOS_READ       ; READ ALARM REGISTER
258 014B 24 57         AND   AL,57H          ; TURN OFF ALARM ENABLE
259 014D B6 E0         XCHG  AH,AL           ; SAVE DATA AND RECOVER ADDRESS
260 014F E8 0000 E      CALL   CMOS_WRITE     ; RESTORE NEW VALUE
261 0152 F8           CLC                    ; SET CY= 0
262 0153 C3           RET                    ; RETURN WITH NO CARRY
263
RTC_00 ENDP
264 0154
265
RTC_STA PROC
266 0154          NEAR
267 0154 BB 260A        MOV    AX,26H*H+CMOS_REG_A ; ADDRESS REGISTER A AND LOAD DATA MASK
268 0157 E8 0000 E      CALL   CMOS_WRITE     ; INITIALIZE STATUS REGISTER A
269 015A BB 820B        MOV    AX,82H*H+CMOS_REG_B ; SET "SET BIT" FOR CLOCK INITIALIZATION
270 015D E8 0000 E      CALL   CMOS_WRITE     ; AND 24 HOUR MODE TO REGISTER B
271 0160 B0 0C         MOV    AL,CMOS_REG_C   ; ADDRESS REGISTER C
272 0162 E8 0000 E      CALL   CMOS_READ       ; READ REGISTER C TO INITIALIZE
273 0165 B0 0D         MOV    AL,CMOS_REG_D   ; ADDRESS REGISTER D
274 0167 E8 0000 E      CALL   CMOS_READ       ; READ REGISTER D TO INITIALIZE
275 016A C3           RET
276
RTC_STA ENDP
277 016B
278
UPD_IPR PROC
279 016B          NEAR
280 016B 51           CX
281 016C B9 0320        PUSH  CX,800          ; WAIT TILL UPDATE NOT IN PROGRESS
282 016F          MOV    CX,800         ; SAVE CALLERS REGISTER
283 016F B0 0A         UPD_10:              ; SET TIMEOUT LOOP COUNT
284 0171 FA           MOV    AL,CMOS_REG_A   ; ADDRESS STATUS REGISTER A
285 0172 E8 0000 E      CALL   CMOS_READ       ; NO TIMER INTERRUPTS DURING UPDATES
286 0175 AB 80         TEST  AL,80H           ; READ UPDATE IN PROCESS FLAG
287 0177 74 06         JZ    UPD_90           ; IF UP BIT IS ON I CANNOT READ TIME )
288 0179 FB           STI                    ; EXIT WITH CY= 0 IF CAN READ CLOCK NOW
289 017A E2 F3        LOOP UPD_10           ; ALLOW INTERRUPTS WHILE WAITING
290 017C 33 C0        XOR   AX,AX           ; LOOP TILL READY OR TIMEOUT
291 017E F9           STC                    ; CLEAR RESULTS IF ERROR
292 017F          UPD_90:              ; SET CARRY FOR ERROR
293 017F 59           POP    CX              ; RESTORE CALLERS REGISTER
294 0180 FA           CLI                    ; INTERRUPTS OFF DURING SET
295 0181 C3           RET                    ; RETURN WITH CY FLAG SET
296
UPD_IPR ENDP
297 0182

```

```

298                                     PAGE
299 ;--- HARDWARE INT 70 H -- ( IRQ LEVEL 8 ) -----
300 ; ALARM INTERRUPT HANDLER (RTC)
301 ; THIS ROUTINE HANDLES THE PERIODIC AND ALARM INTERRUPTS FROM THE CMOS
302 ; TIMER. INPUT FREQUENCY IS 1.024 KHZ OR APPROXIMATELY 1024 INTERRUPTS
303 ; EVERY SECOND FOR THE PERIODIC INTERRUPT. FOR THE ALARM FUNCTION,
304 ; THE INTERRUPT WILL OCCUR AT THE DESIGNATED TIME.
305 ;
306 ; INTERRUPTS ARE ENABLED WHEN THE EVENT OR ALARM FUNCTION IS ACTIVATED.
307 ; FOR THE EVENT INTERRUPT, THE HANDLER WILL DECREMENT THE WAIT COUNTER
308 ; AND WHEN IT EXPIRES WILL SET THE DESIGNATED LOCATION TO 80H. FOR
309 ; THE ALARM INTERRUPT, THE USER MUST PROVIDE A ROUTINE TO INTERCEPT
310 ; THE CORRECT ADDRESS FROM THE VECTOR TABLE INVOKED BY INTERRUPT 4AH
311 ; PRIOR TO SETTING THE REAL TIME CLOCK ALARM (INT 1AH, AH= 06H).
312 -----
313
314 0182                                     RTC_INT PROC     FAR
315 0182 1E                                 PUSH     DS
316 0183 50                                 PUSH     AX
317 0184 57                                 PUSH     DI
318
319 0185                                     ; ALARM INTERRUPT
320 0185 B8 8B8C                           ; LEAVE INTERRUPTS DISABLED
321 0188 E6 70                             ; SAVE REGISTERS
322 018A 90
323 018B E4 71                             ; CHECK FOR SECOND INTERRUPT
324 018D A8 60                             ; REG_C+NM1 ; ALARM AND STATUS
325 018F 74 4D                             ; WRITE ALARM FLAG MASK ADDRESS
326                                     ; CMOS_PORT,AL
327 0191 86 E0                             ; I/O DELAY
328 0193 E6 70                             ; READ AND RESET INTERRUPT REQUEST FLAGS
329 0195 90                             ; CHECK FOR EITHER INTERRUPT PENDING
330 0196 E4 71                             ; EXIT IF NOT A VALID RTC INTERRUPT
331 0198 22 C4                             ; SAVE FLAGS AND SET ENABLE ADDRESS
332 019A A8 40                             ; WRITE ALARM ENABLE MASK ADDRESS
333 019C 74 30                             ; CMOS_PORT,AL
334                                     ; NOP
335                                     ; IN AL,CMOS_DATA
336                                     ; AND AL,AH
337 019E E8 0000 E ; DECREMENT WAIT COUNT BY INTERRUPT INTERVAL
338 01A1 81 2E 009C R 03D0                ; CALL DDS ; ESTABLISH DATA SEGMENT ADDRESSABILITY
339 01A7 83 1E 009E R 00                  ; SUB 0RTC_LOW,0976 ; DECREMENT COUNT LOW BY 1/1024
340 01AC 73 20                             ; SBB 0RTC_HIGH,0 ; ADJUST HIGH WORD FOR LOW WORD BORROW
341                                     ; JNC RTC_T_5 ; SKIP TILL 32 BIT WORD LESS THAN ZERO
342
343 -----
344 01AE 50                                     ; TURN OFF PERIODIC INTERRUPT ENABLE
345 01AF B8 8B8B                           ; PUSH AX
346 01B2 E6 70                             ; MOV AX,X*(CMOS_REG_B+NM1) ; SAVE INTERRUPT FLAG MASK
347 01B4 90                             ; OUT CMOS_PORT,AL ; INTERRUPT ENABLE REGISTER
348 01B5 E4 71                             ; NOP ; WRITE ADDRESS TO CMOS CLOCK
349 01B7 24 BF                             ; IN AL,CMOS_DATA ; I/O DELAY
350 01B9 86 C4                             ; AND AL,0BFH ; READ CURRENT ENABLES
351 01BB E6 70                             ; XCHG AL,AH ; TURN OFF PIE
352 01BD 86 C4                             ; OUT CMOS_PORT,AL ; GET CMOS ADDRESS AND SAVE VALUE
353 01BF E6 71                             ; XCHG AL,AH ; ADDRESS REGISTER B
354 01C1 C6 06 00AD R 00                  ; MOV 0RTC_WAIT_FLAG,0 ; GET NEW INTERRUPT ENABLE MASK
355 01C6 C5 3E 0098 R                      ; LDS DI,WORD PTR USER_FLAG ; SET MASK IN INTERRUPT ENABLE REGISTER
356 01CA C6 05 80                          ; MOV BYTE PTR [DI],80H ; SET FUNCTION ACTIVE FLAG OFF
357 01CC 58                               ; POP AX ; SET UP (DS:DI) TO POINT TO USER FLAG
358 01CE                                     ; ; TURN ON USERS FLAG
359 01CE A8 20                             ; RTC_I_5: ; GET INTERRUPT SOURCE BACK
360 01D0 74 0A                             ; TEST AL,00100000B ; TEST FOR ALARM INTERRUPT
361                                     ; JZ RTC_I_7 ; SKIP USER INTERRUPT CALL IF NOT ALARM
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 01D7 52                               ; PUSH DX
366 01DB CD 4A                             ; INT 4AH ; TRANSFER TO USER ROUTINE
367 01DA 5A                               ; POP DX
368 01DB FA                               ; CLI ; BLOCK INTERRUPT FOR RETRY
369 01DC                                     ; ; RESTART ROUTINE TO HANDLE DELAYED
370 01DE EB A7                             ; RTC_I_7: ; ENTRY AND SECOND EVENT BEFORE DONE
371                                     ; JMP RTC_I_1
372
373 01DE B0 0D                             ; RTC_I_9: ; EXIT - NO PENDING INTERRUPTS
374 01E0 E6 70                             ; MOV AL,CMOS_REG_D ; POINT TO DEFAULT READ ONLY REGISTER
375 01E2 B0 20                             ; OUT CMOS_PORT,AL ; ENABLE NMI AND CMOS ADDRESS TO DEFAULT
376 01E4 E6 A0                             ; MOV AL,EDI ; END OF INTERRUPT MASK TO 8259 - 2
377 01E6 E6 20                             ; OUT INTB00,AL ; TO 8259 - 2
378 01E8 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 PAGE
386 ;--- INT 05H
387 | PRINT_SCREEN
388 | THIS LOGIC WILL BE INVOKED BY INTERRUPT 05H TO PRINT THE SCREEN.
389 | THE CURSOR POSITION AT THE TIME THIS ROUTINE IS INVOKED WILL BE
390 | SAVED AND RESTORED UPON COMPLETION. THE ROUTINE IS INTENDED TO
391 | RUN WITH INTERRUPTS ENABLED. IF A SUBSEQUENT PRINT_SCREEN KEY
392 | IS DEPRESSED WHILE THIS ROUTINE IS PRINTING IT WILL BE IGNORED.
393 | THE BASE PRINTERS STATUS IS CHECKED FOR NOT BUSY AND NOT OUT OF
394 | PAPER. AN INITIAL STATUS ERROR WILL AVOID THE PRINT REQUEST.
395 | ADDRESS 00501000 CONTAINS THE STATUS OF THE PRINT SCREEN:
396 |
397 | 50:0 = 0 PRINT_SCREEN HAS NOT BEEN CALLED OR UPON RETURN
398 | FROM A CALL THIS INDICATES A SUCCESSFUL OPERATION.
399 | = 1 PRINT_SCREEN IS IN PROGRESS - IGNORE THIS REQUEST.
400 | = 255 ERROR ENCOUNTERED DURING PRINTING.
401 |
402 |-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
403 01EC PRINT_SCREEN_1 PROC FAR
404 |
405 01EC IE 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 DOS
411 01F4 80 3E 0100 R 01 CMP *STATUS_BYTE,1
412 01F9 74 74 JE *PR190
413 01FB C6 06 0100 R 01 MOV *STATUS_BYTE,1
414 0200 F0 STI
415 0201 B4 0F MOV AH,0FH
416 0203 CD 10 INT 10H
417 |
418 |
419 0205 8A CC MOV CL,AH
420 0207 BA 2E 0084 R MOV CH,ROWS
421 020B FE C5 INC CH
422 |
423 |-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
424 |
425 | AT THIS POINT WE KNOW THE COLUMNS/LINE COUNT IS IN (CL) ;
426 | AND THE NUMBER OF ROWS ON THE DISPLAY IS IN (CH). ;
427 | THE PAGE IF APPLICABLE IS IN (BH). THE STACK HAS ;
428 | (DS), (AX), (BX), (CX), (DX) PUSHED. ;
429 |-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
430 020D 33 D2 XOR DX,DX ; FIRST PRINTER
431 020F B4 02 MOV AH,02H ; SET PRINTER STATUS REQUEST COMMAND
432 0211 CD 17 INT 17H ; REQUEST CURRENT PRINTER STATUS
433 0213 80 F4 80 XOR AH,080H ; CHECK FOR PRINTER BUSY (NOT CONNECTED)
434 0216 F6 C4 A0 TEST AH,0A0H ; OR OUT OF PAPER
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 POP CX ; RECALL 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 | THIS LOOP IS TO READ EACH CURSOR POSITION FROM THE ;
447 | SCREEN AND PRINT IT. (BH) = VISUAL PAGE (CH) = ROWS ;
448 |-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
449 0227 PR110: MOV AH,02H ; INDICATE CURSOR SET REQUEST
450 0227 B4 02 INT 02H ; NEW CURSOR POSITION ESTABLISHED
451 0229 CD 10 MOV AH,08H ; INDICATE READ CHARACTER FROM DISPLAY
452 022B B4 08 INT 08H ; CHARACTER NOW IN (AL)
453 022D CD 10 OR AL,AL ; SEE IF VALID CHAR
454 022F 0A C0 JNZ PR120 ; JUMP IF VALID CHAR
455 0231 75 02 MOV AL,' ' ; ELSE MAKE IT A BLANK
456 0233 B0 20 PR120: PUSH DX ; SAVE CURSOR POSITION
457 0235 DX XOR DX,DX ; INDICATE FIRST PRINTER (DX= 0)
458 0235 52 XOR AH,AH ; INDICATE PRINT CHARACTER IN (AL)
459 0236 33 D2 INT 17H ; PRINT THE CHARACTER
460 0238 32 E4 POP DX ; RECALL CURSOR POSITION
461 023A CD 17 JNZ PR170 ; TEST FOR PRINTER ERROR
462 023C 5A POP DX ; EXIT IF ERROR DETECTED
463 023D F6 C4 29 JNZ PR170 ; ADVANCE TO NEXT COLUMN
464 0240 75 22 SEE IF AT END OF LINE
465 0242 FE C2 EXIT IF ERROR DETECTED
466 0244 3A C1 CMP CL,DL ; ADVANCE TO NEXT COLUMN
467 0246 75 DF JNZ PR110 ; IF NOT LOOP FOR NEXT COLUMN
468 0248 32 D2 XOR DL,DL ; BACK TO COLUMN 0
469 024A 8A E2 MOV AH,DL ; (AH)=0
470 024C 52 PUSH DX ; SAVE NEW CURSOR POSITION
471 024D E8 0275 R CALL CRLF ; LINE FEED CARRIAGE RETURN
472 0250 5A POP DX ; RECALL CURSOR POSITION
473 0251 FE C6 INC DH ; ADVANCE TO NEXT LINE
474 0253 3A EE CMP CH,DH ; FINISHED?
475 0255 75 D0 JNZ PR110 ; IF NOT LOOP FOR NEXT LINE
476 |
477 0257 5A POP DX ; GET CURSOR POSITION
478 0258 B4 02 MOV AH,02H ; INDICATE REQUEST CURSOR SET
479 025A CD 10 INT 10H ; CURSOR POSITION RESTORED
480 025C FA CLI ; BLOCK INTERRUPTS TILL STACK CLEARED
481 025D C6 06 0100 R 00 MOV *STATUS_BYTE,0 ; MOVE OK RESULTS FLAG TO STATUS_BYTE
482 0262 EB 0B JMP SHORT PR190 ; EXIT PRINTER ROUTINE
483 |
484 0264 PR170: POP DX ; ERROR EXIT
485 0264 5A POP DX ; GET CURSOR POSITION
486 0265 B4 02 MOV AH,02H ; INDICATE REQUEST CURSOR SET
487 0267 CD 10 INT 10H ; CURSOR POSITION RESTORED
488 |
489 0269 FA CLI ; BLOCK INTERRUPTS TILL STACK CLEARED
490 026A C6 06 0100 R FF MOV *STATUS_BYTE,OFFH ; SET ERROR FLAG
491 026F PR190: POP DX ; EXIT ROUTINE
492 026F 5A POP CX ; RESTORE ALL THE REGISTERS USED
493 0270 59 POP BX
494 0271 5B POP AX
495 0272 58 POP DS
496 0273 IF IRET
497 0274 CF IRET ; RETURN WITH INITIAL INTERRUPT MASK
498 0275 PRINT_SCREEN_1 ENDP

```



```

499
500 ;----- CARRIAGE RETURN, LINE FEED SUBROUTINE
501
502 0275 CRLF PROC NEAR
503 ; SEND CR,LF TO FIRST PRINTER
504 0275 33 D2 XOR DX,DX ; ASSUME FIRST PRINTER (DX= 0)
505 0277 B8 000D MOV AX,CR ; GET THE PRINT CHARACTER COMMAND AND
506 027A CD 17 INT 17H ; THE CARRIAGE RETURN CHARACTER
507 027C B8 000A MOV AX,LF ; NOW GET THE LINE FEED AND
508 027F CD 17 INT 17H ; SEND IT TO THE BIOS PRINTER ROUTINE
509 0281 C3 RET
510 0282 CRLF ENDP
511
512 ;--- HARDWARE INT 08 H -- ( IRQ LEVEL 0 ) -----
513 ;
514 ; THIS ROUTINE HANDLES THE TIMER INTERRUPT FROM FROM CHANNEL 0 OF
515 ; THE 8254 TIMER. INPUT FREQUENCY IS 1.19318 MHZ AND THE DIVISOR
516 ; IS 65536, RESULTING IN APPROXIMATELY 18.2 INTERRUPTS EVERY SECOND.
517 ;
518 ; THE INTERRUPT HANDLER MAINTAINS A COUNT (40:6C) OF INTERRUPTS SINCE
519 ; POWER ON TIME, WHICH MAY BE USED TO ESTABLISH TIME OF DAY.
520 ; THE INTERRUPT HANDLER ALSO DECREASES THE MOTOR CONTROL COUNT (40:40)
521 ; OF THE DISKETTE, AND WHEN IT EXPIRES, WILL TURN OFF THE
522 ; DISKETTE MOTOR(s), AND RESET THE MOTOR RUNNING FLAGS.
523 ; THE INTERRUPT HANDLER WILL ALSO INVOKE A USER ROUTINE THROUGH
524 ; INTERRUPT 1CH AT EVERY TIME TICK. THE USER MUST CODE A
525 ; ROUTINE AND PLACE THE CORRECT ADDRESS IN THE VECTOR TABLE.
526 ;-----
527
528
529
530 0282 TIMER_INT I PROC FAR
531 0282 FB STI ; INTERRUPTS BACK ON
532 0283 1E PUSH DS
533 0284 50 PUSH AX
534 0285 52 PUSH DX ; SAVE MACHINE STATE
535 0286 E8 0000 E CALL DDS ; ESTABLISH ADDRESSABILITY
536 0289 FF 06 006C R INC @TIMER_LOW ; INCREMENT TIME
537 028D 75 04 JNZ T4 ; GO TO TEST DAY
538 028F FF 06 006E R INC @TIMER_HIGH ; INCREMENT HIGH WORD OF TIME
539 0293 T4: ; TEST DAY
540 0293 83 3E 006E R 18 CMP @TIMER_HIGH,018H ; TEST FOR COUNT EQUALING 24 HOURS
541 0298 75 15 JNZ T5 ; GO TO DISKETTE_CTL
542 029A 81 3E 006C R 00B0 CMP @TIMER_LOW,00B0H
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 02AA C6 06 0070 R 01 MOV @TIMER_OFL,I
551
552 ;----- TEST FOR DISKETTE TIME OUT
553
554 02AF T5:
555 02AF FE 0E 0040 R DEC @MOTOR_COUNT ; DECREMENT DISKETTE MOTOR CONTROL
556 02B3 75 0B JNZ T6 ; RETURN IF COUNT NOT OUT
557 02B5 80 26 003F R F0 AND @MOTOR_STATUS,0F0H ; TURN OFF MOTOR RUNNING BITS
558 02BA B0 0C MOV AL,0CH
559 02BC BA 03F2 MOV DX,03F2H ; FDC CTL PORT
560 02BF EE OUT DX,AL ; TURN OFF THE MOTOR
561
562 02C0 T6: ; TIMER TICK INTERRUPT
563 02C0 CD 1C INT 1CH ; TRANSFER CONTROL TO A USER ROUTINE
564
565 02C2 5A POP DX ; RESTORE (DX)
566 02C3 B0 20 MOV AL,E0I ; GET END OF INTERRUPT MASK
567 02C5 FA CLI ; DISABLE INTERRUPTS TILL STACK CLEARED
568 02C6 E6 20 OUT INTA00,AL ; END OF INTERRUPT TO 8259 - I
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 I ENDP
574
575 02CB CODE ENDS
576 END
    
```

```

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 DUMM7_RETURN
14 PUBLIC E101
15 PUBLIC E102
16 PUBLIC E103
17 PUBLIC E104
18 PUBLIC E105
19 PUBLIC E106
20 PUBLIC E107
21 PUBLIC E108
22 PUBLIC E109
23 PUBLIC E161
24 PUBLIC E162
25 PUBLIC E163
26 PUBLIC E164
27 PUBLIC E201
28 PUBLIC E202
29 PUBLIC E203
30 PUBLIC E301
31 PUBLIC E302
32 PUBLIC E303
33 PUBLIC E304
34 PUBLIC E401
35 PUBLIC E501
36 PUBLIC E601
37 PUBLIC E602
38 PUBLIC F1780
39 PUBLIC F1781
40 PUBLIC F1782
41 PUBLIC F1790
42 PUBLIC F1791
43 PUBLIC F3A
44 PUBLIC F3D
45 PUBLIC F3D1
46 PUBLIC FD_TBL
47 PUBLIC FLOPPY
48 PUBLIC HRD
49 PUBLIC K10
50 PUBLIC K11
51 PUBLIC K12
52 PUBLIC K13
53 PUBLIC K14
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 NM1_INT
65 PUBLIC PRINT_SCREEN
66 PUBLIC P_O_R
67 PUBLIC SEEK5_I
68 PUBLIC SLAVE_VECTOR_TABLE
69 PUBLIC TUTOR
70 PUBLIC VECTOR_TABLE
71 PUBLIC VIDEO_FARMS
72
73 EXTRN BOOT_STRAP_I:NEAR
74 EXTRN CASSETTE_IO_I:NEAR
75 EXTRN D11:NEAR
76 EXTRN DISK_INT_I:NEAR
77 EXTRN DISK_SETUP:NEAR
78 EXTRN DISKETTE_IO_I:NEAR
79 EXTRN DSKETTE_SETUP:NEAR
80 EXTRN EQUIPMENT_I:NEAR
81 EXTRN INT_287:NEAR
82 EXTRN K16:NEAR
83 EXTRN KEYBOARD_IO_I:NEAR
84 EXTRN KB_INT_I:NEAR
85 EXTRN MEMORY_SIZE_DET_I:NEAR
86 EXTRN NM1_INT_I:NEAR
87 EXTRN PRINT_SCREEN_I:NEAR
88 EXTRN PRINTER_IO_I:NEAR
89 EXTRN RE_DIRECT:NEAR
90 EXTRN RS232_IO_I:NEAR
91 EXTRN RTC_INT:NEAR
92 EXTRN SEEK:NEAR
93 EXTRN START_I:NEAR
94 EXTRN TIME_OF_DAY_I:NEAR
95 EXTRN TIMER_INT_I:NEAR
96 EXTRN VIDEO_IO_T:NEAR
97
98 ASSUME CS:CODE,DS:DATA
99
100

```

```

-----
: 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 0000                                |  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 4B
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 4B
133      20 32 0D 0A
134 0040 3F 3F 3F 3F 3F 0D             D2A     DB      '?????',CR,LF
135      0A
136 = 0047                                IP      =      $
137                                     |  ORG  0E05BH  |
138 005B                                |  ORG  0005BH  |
139 005B                                RESET:   JUMP   START_I ; RESET START
140 005B E9 0000 E                       ; VECTOR ON TO THE MOVED POST CODE
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 00A9 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 00DB 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             E161   DB      ' 161-System Options Not Set-(Run SETUP)',CR,LF ; DEAD BATTERY
192      79 73 74 65 6D 20
193      4F 70 74 69 6F 6E
194      73 20 4E 6F 74 20
195      53 65 74 2D 28 52
196      75 6E 20 53 45 54
197      55 50 29 0D 0A
198 0168 20 31 36 32 2D 53             E162   DB      ' 162-System Options Not Set-(Run SETUP)',CR,LF ; CHECKSUM/CONFIG
199      79 73 74 65 6D 20
200      4F 70 74 69 6F 6E
201      73 20 4E 6F 74 20
202      53 65 74 2D 28 52
203      75 6E 20 53 45 54
204      55 50 29 0D 0A
205 0191 20 31 36 33 2D 53             E163   DB      ' 163-Time & Date Not Set-(Run SETUP)',CR,LF ; CLOCK NOT UPDATING
206      69 6D 65 20 26 20
207      44 61 74 65 20 4E
208      6F 74 20 53 65 74
209      2D 28 52 75 6E 20
210      53 45 54 55 50 29
211      0D 0A
212 01B7 20 31 36 34 2D 4D             E164   DB      ' 164-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 50 29 0D 0A
218 01D8 20 32 30 31 2D 4D             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 ; LINE ERROR 00->15
223      65 6D 6F 72 79 20

```

SECTION 5

```

224      41 64 64 72 65 73
225      73 20 45 72 72 6F
226      72 0D 0A
0209 20 32 30 33 2D 4D  E203  DB      ' 203-Memory Address Error',CR,LF      ; LINE ERROR 16->23
228      65 6D 6F 72 79 20
229      41 64 64 72 65 73
230      73 20 45 72 72 6F
231      72 0D 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 0D 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      55 6E 69 74 20 4B
239      65 79 6C 6F 63 6B
240      20 69 73 20 4C 6F
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 2D 4B 45
245      59 29 0D 0A
246
247      ;----- NMI ENTRY
248
249      = 0273
250      ;IP      =      $
251      ;i:-    ORG      0E2C3H
252      ;       ORG      002C3H
253      ;NMI_INT EQU     $
254      ;02C3 E9 0000 E      NMI_INT_I      ; VECTOR ON TO MOVED NMI CODE
255
256 02C6 20 33 30 33 2D 4B  E303  DB      ' 303-Keyboard Or System Unit Error',CR,LF
257      65 79 62 6F 61 72
258      64 20 45 72 20 4B
259      79 73 74 65 6D 20
260      55 6E 69 74 20 45
261      72 72 6F 72 0D 0A
262 02EA 20 33 30 34 2D 4B  ;----- KEYBOARD/SYSTEM ERROR
263      65 79 62 6F 61 72  E304  DB      ' 304-Keyboard Or System Unit Error',CR,LF ; KEYBOARD CLOCK HIGH
264      64 20 4F 72 20 53
265      79 73 74 65 6D 20
266      55 6E 69 74 20 45
267      72 72 6F 72 0D 0A
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 0D 0A
271 031E 20 35 30 31 2D 43  E501  DB      ' 501-CRT Error',CR,LF      ; COLOR
272      52 54 20 45 72 72
273      6F 72 0D 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 0D 0A
278
279      ;----- DISKETTE BOOT RECORD IS NOT VALID
280 0343 20 36 30 32 2D 44  E602  DB      ' 602-Diskette Boot Record Error',CR,LF
281      69 73 6B 65 74 74
282      65 20 42 6F 6F 74
283      20 52 65 63 6F 72
284      64 20 45 72 72 6F
285      72 0D 0A
286
287      ;----- HARD FILE ERROR MESSAGE
288 0364 31 37 38 30 2D 44  F1780 DB      '1780-Disk 0 Failure',CR,LF
289      69 73 6B 20 30 20
290      46 61 69 6C 75 72
291      65 0D 0A
292 0379 31 37 38 31 2D 44  F1781 DB      '1781-Disk 1 Failure',CR,LF
293      69 73 6B 20 31 20
294      46 61 69 6C 75 72
295      65 0D 0A
296 038E 31 37 38 32 2D 44  F1782 DB      '1782-Disk Controller Failure',CR,LF
297      69 73 6B 20 43 6F
298      6E 74 72 6F 6C 6C
299      65 72 20 46 61 69
300      6C 75 72 65 0D 0A
301 03AC 31 37 39 30 2D 44  F1790 DB      '1790-Disk 0 Error',CR,LF
302      69 73 6B 20 30 20
303      45 72 72 6F 72 0D
304      0A
305 03BF 31 37 39 31 2D 44  F1791 DB      '1791-Disk 1 Error',CR,LF
306      69 73 6B 20 31 20
307      45 72 72 6F 72 0D
308      0A
309 03D2 52 4F 4D 20 20 45  F3A   DB      'ROM Error ',CR,LF      ; ROM CHECKSUM
310      72 72 6F 72 20 0D
311      0A
312 03DF 20 20 20 20 20 55  F3D1  DB      ' -Unlock System Unit Keylock ',CR,LF
313      6E 6C 6F 63 6B 20
314      53 79 73 74 65 6D
315      20 55 6E 69 74 20
316      4B 65 79 6C 6F 63
317      6B 20 0D 0A

```

```

317 PAGE
318 -----
319 ; INITIALIZE DRIVE CHARACTERISTICS ;
320 ; ;
321 ; FIXED DISK PARAMETER TABLE ;
322 ; ;
323 ; - THE TABLE IS COMPOSED OF A BLOCK DEFINED AS: ;
324 ; ;
325 ; +0 (1 WORD) - MAXIMUM NUMBER OF CYLINDERS ;
326 ; +2 (1 BYTE) - MAXIMUM NUMBER OF HEADS ;
327 ; +3 (1 WORD) - NOT USED/SEE PC-XT ;
328 ; +5 (1 WORD) - STARTING WRITE PRECOMPENSATION CYL. ;
329 ; +7 (1 BYTE) - NOT USED/SEE PC-XT ;
330 ; +8 (1 BYTE) - CONTROL BYTE ;
331 ; BIT 7 DISABLE RETRIES -OR- ;
332 ; BIT 6 DISABLE RETRIES ;
333 ; BIT 3 MORE THAN 8 HEADS ;
334 ; +9 (3 BYTES) - NOT USED/SEE PC-XT ;
335 ; +12 (1 WORD) - LANDING ZONE ;
336 ; +14 (1 BYTE) - NUMBER OF SECTORS/TRACK ;
337 ; +15 (1 BYTE) - RESERVED FOR FUTURE USE ;
338 ; ;
339 ; - TO DYNAMICALLY DEFINE A SET OF PARAMETERS ;
340 ; BUILD A TABLE FOR UP TO 15 TYPES AND PLACE ;
341 ; THE CORRESPONDING VECTOR INTO INTERRUPT 41 ;
342 ; FOR DRIVE 0 AND INTERRUPT 46 FOR DRIVE 1. ;
343 ; ;
344 -----

```

```

346 0401 FD_TBL:
347
348 1----- DRIVE TYPE 01
349
350 0401 0132 DW 0306D ; CYLINDERS
351 0403 04 DB 04D ; HEADS
352 0404 0000 DW 0 ;
353 0406 0080 DW 0128D ; WRITE PRE-COMPENSATION CYLINDER
354 0408 00 DB 0 ;
355 0409 00 DB 0 ; CONTROL BYTE
356 040A 00 00 00 DB 0,0,0 ;
357 040D 0131 DW 0305D ; LANDING ZONE
358 040F 11 DB 17D ; SECTORS/TRACK
359 0410 00 DB 0 ;
360
361 1----- DRIVE TYPE 02
362
363 0411 0267 DW 0615D ; CYLINDERS
364 0413 04 DB 04D ; HEADS
365 0414 0000 DW 0 ;
366 0416 012C DW 0300D ; WRITE PRE-COMPENSATION CYLINDER
367 0418 00 DB 0 ;
368 0419 00 DB 0 ; CONTROL BYTE
369 041A 00 00 00 DB 0,0,0 ;
370 041D 0267 DW 0615D ; LANDING ZONE
371 041F 11 DB 17D ; SECTORS/TRACK
372 0420 00 DB 0 ;
373
374 1----- DRIVE TYPE 03
375
376 0421 0267 DW 0615D ; CYLINDERS
377 0423 06 DB 06D ; HEADS
378 0424 0000 DW 0 ;
379 0426 012C DW 0300D ; WRITE PRE-COMPENSATION CYLINDER
380 0428 00 DB 0 ;
381 0429 00 DB 0 ; CONTROL BYTE
382 042A 00 00 00 DB 0,0,0 ;
383 042D 0267 DW 0615D ; LANDING ZONE
384 042F 11 DB 17D ; SECTORS/TRACK
385 0430 00 DB 0 ;
386
387 1----- DRIVE TYPE 04
388
389 0431 03AC DW 0940D ; CYLINDERS
390 0433 08 DB 08D ; HEADS
391 0434 0000 DW 0 ;
392 0436 0200 DW 0512D ; WRITE PRE-COMPENSATION CYLINDER
393 0438 00 DB 0 ;
394 0439 00 DB 0 ; CONTROL BYTE
395 043A 00 00 00 DB 0,0,0 ;
396 043D 03AC DW 0940D ; LANDING ZONE
397 043F 11 DB 17D ; SECTORS/TRACK
398 0440 00 DB 0 ;
399
400 1----- DRIVE TYPE 05
401
402 0441 03AC DW 0940D ; CYLINDERS
403 0443 06 DB 06D ; HEADS
404 0444 0000 DW 0 ;
405 0446 0200 DW 0512D ; WRITE PRE-COMPENSATION CYLINDER
406 0448 00 DB 0 ;
407 0449 00 DB 0 ; CONTROL BYTE
408 044A 00 00 00 DB 0,0,0 ;
409 044D 03AC DW 0940D ; LANDING ZONE
410 044F 11 DB 17D ; SECTORS/TRACK
411 0450 00 DB 0 ;
412
413 1----- DRIVE TYPE 06
414
415 0451 0267 DW 0615D ; CYLINDERS
416 0453 04 DB 04D ; HEADS
417 0454 0000 DW 0 ;
418 0456 FFFF DW 0FFFFH ; NO WRITE PRE-COMPENSATION
419 0458 00 DB 0 ;
420 0459 00 DB 0 ; CONTROL BYTE
421 045A 00 00 00 DB 0,0,0 ;
422 045D 0267 DW 0615D ; LANDING ZONE
423 045F 11 DB 17D ; SECTORS/TRACK
424 0460 00 DB 0 ;
425
426 1----- DRIVE TYPE 07
427
428 0461 01CE DW 0462D ; CYLINDERS
429 0463 08 DB 08D ; HEADS
430 0464 0000 DW 0 ;

```

```

431 0466 0100          DW      0256D          ; WRITE PRE-COMPENSATION CYLINDER
432 0468 00           DB      0
433 0469 00           DB      0
434 046A 00 00 00    DB      0,0,0          ; CONTROL BYTE
435 046D 01FF        DW      0511D          ; LANDING ZONE
436 046F 11          DB      17D           ; SECTORS/TRACK
437 0470 00           DB      0
438
439
440 ;----- DRIVE TYPE 08
441 0471 02DD        DW      0733D          ; CYLINDERS
442 0473 05          DB      05D           ; HEADS
443 0474 0000        DW      0
444 0476 FFFF        DW      0FFFFH         ; NO WRITE PRE-COMPENSATION
445 0478 00           DB      0
446 0479 00           DB      0
447 047A 00 00 00    DB      0,0,0          ; CONTROL BYTE
448 047D 02DD        DW      0733D          ; LANDING ZONE
449 047F 11          DB      17D           ; SECTORS/TRACK
450 0480 00           DB      0
451
452 ;----- DRIVE TYPE 09
453
454 0481 0384        DW      0900D          ; CYLINDERS
455 0483 0F          DB      15D           ; HEADS
456 0484 0000        DW      0
457 0486 FFFF        DW      0FFFFH         ; NO WRITE PRE-COMPENSATION
458 0488 00           DB      0
459 0489 08           DB      008H          ; CONTROL BYTE
460 048A 00 00 00    DB      0,0,0          ; CONTROL BYTE
461 048D 0385        DW      0901D          ; LANDING ZONE
462 048F 11          DB      17D           ; SECTORS/TRACK
463 0490 00           DB      0
464
465 ;----- DRIVE TYPE 10
466
467 0491 0334        DW      0820D          ; CYLINDERS
468 0493 03          DB      03D           ; HEADS
469 0494 0000        DW      0
470 0496 FFFF        DW      0FFFFH         ; NO WRITE PRE-COMPENSATION
471 0498 00           DB      0
472 0499 00           DB      0
473 049A 00 00 00    DB      0,0,0          ; CONTROL BYTE
474 049D 0334        DW      0820D          ; LANDING ZONE
475 049F 11          DB      17D           ; SECTORS/TRACK
476 04A0 00           DB      0
477
478 ;----- DRIVE TYPE 11
479
480 04A1 0357        DW      0855D          ; CYLINDERS
481 04A3 05          DB      05D           ; HEADS
482 04A4 0000        DW      0
483 04A6 FFFF        DW      0FFFFH         ; NO WRITE PRE-COMPENSATION
484 04A8 00           DB      0
485 04A9 00           DB      0
486 04AA 00 00 00    DB      0,0,0          ; CONTROL BYTE
487 04AD 0357        DW      0855D          ; LANDING ZONE
488 04AF 11          DB      17D           ; SECTORS/TRACK
489 04B0 00           DB      0
490
491 ;----- DRIVE TYPE 12
492
493 04B1 0357        DW      0855D          ; CYLINDERS
494 04B3 07          DB      07D           ; HEADS
495 04B4 0000        DW      0
496 04B6 FFFF        DW      0FFFFH         ; NO WRITE PRE-COMPENSATION
497 04B8 00           DB      0
498 04B9 00           DB      0
499 04BA 00 00 00    DB      0,0,0          ; CONTROL BYTE
500 04BD 0357        DW      0855D          ; LANDING ZONE
501 04BF 11          DB      17D           ; SECTORS/TRACK
502 04C0 00           DB      0
503
504 ;----- DRIVE TYPE 13
505
506 04C1 0132        DW      0306D          ; CYLINDERS
507 04C3 08          DB      08D           ; HEADS
508 04C4 0000        DW      0
509 04C6 0080        DW      0128D          ; WRITE PRE-COMPENSATION CYLINDER
510 04C8 00           DB      0
511 04C9 00           DB      0
512 04CA 00 00 00    DB      0,0,0          ; CONTROL BYTE
513 04CD 013F        DW      0319D          ; LANDING ZONE
514 04CF 11          DB      17D           ; SECTORS/TRACK
515 04D0 00           DB      0
516
517 ;----- DRIVE TYPE 14
518
519 04D1 02DD        DW      0733D          ; CYLINDERS
520 04D3 07          DB      07D           ; HEADS
521 04D4 0000        DW      0
522 04D6 FFFF        DW      0FFFFH         ; NO WRITE PRE-COMPENSATION
523 04D8 00           DB      0
524 04D9 00           DB      0
525 04DA 00 00 00    DB      0,0,0          ; CONTROL BYTE
526 04DD 02DD        DW      0733D          ; LANDING ZONE
527 04DF 11          DB      17D           ; SECTORS/TRACK
528 04E0 00           DB      0
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
535 04E6 0000        DW      0000D          ; WRITE PRE-COMPENSATION CYLINDER
536 04E8 00           DB      0
537 04E9 00           DB      0
538 04EA 00 00 00    DB      0,0,0          ; CONTROL BYTE
539 04ED 0000        DW      0000D          ; LANDING ZONE
540 04EF 00           DB      00D           ; SECTORS/TRACK
541 04F0 00           DB      0
542
543 ;----- DRIVE TYPE 16
544

```

```

545 04F1 0264          DW 0612D          ; CYLINDERS
546 04F3 04           DB 04D           ; HEADS
547 04FA 0000         DW 0             ;
548 04F6 0000         DW 0000D        ; WRITE PRE-COMPENSATION ALL CYLINDER
549 04F8 00           DB 0             ;
550 04F9 00           DB 0             ;
551 04FA 00 00 00     DB 0,0,0       ; CONTROL BYTE
552 04FD 0297         DW 0663D        ; LANDING ZONE
553 04FF 11           DB 17D          ; SECTORS/TRACK
554 0500 00           DB 0             ;
555
556 ;----- DRIVE TYPE 17
557
558 0501 03D1         DW 0977D        ; CYLINDERS
559 0503 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             ; CONTROL BYTE
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 0FFFFFH      ; NO WRITE PRE-COMPENSATION
575 0518 00           DB 0             ;
576 0519 00           DB 0             ; CONTROL BYTE
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             ; CONTROL BYTE
590 052A 00 00 00     DB 0,0,0       ;
591 052D 03FF         DW 1023D        ; 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             ; CONTROL BYTE
603 053A 00 00 00     DB 0,0,0       ;
604 053D 032D         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 0548 00           DB 0             ;
615 0549 00           DB 0             ; CONTROL BYTE
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             ; CONTROL BYTE
629 055A 00 00 00     DB 0,0,0       ;
630 055D 02DC         DW 0732D        ; 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 0564 0000         DW 0             ;
639 0566 0000         DW 0000D        ; WRITE PRE-COMPENSATION ALL CYL
640 0568 00           DB 0             ;
641 0569 00           DB 0             ; CONTROL BYTE
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 0             ;
652 0576 0000         DW 0000D        ; WRITE PRE-COMPENSATION CYL
653 0578 00           DB 0             ;
654 0579 00           DB 0             ; CONTROL BYTE
655 057A 00 00 00     DB 0,0,0       ;
656 057D 0000         DW 0000D        ; LANDING ZONE
657 057F 00           DB 00D          ; SECTORS/TRACK
658 0580 00           DB 0             ;

```

```

659
660
661          ;----- DRIVE TYPE 25      *** RESERVED***
662      0581 0000      DW      0000D      ; CYLINDERS
663      0583 00      DB      00D         ; HEADS
664      0584 0000      DW      0         ;
665      0586 0000      DW      0000D      ; WRITE PRE-COMPENSATION CYL
666      0588 00      DB      0         ;
667      0589 00      DB      0         ; CONTROL BYTE
668      058A 00 00 00  DB      0,0,0     ;
669      058D 0000      DW      0000D      ; LANDING ZONE
670      058F 00      DB      00D         ; SECTORS/TRACK
671      0590 00      DB      0         ;
672
673          ;----- DRIVE TYPE 26      *** RESERVED***
674
675      0591 0000      DW      0000D      ; CYLINDERS
676      0593 00      DB      00D         ; HEADS
677      0594 0000      DW      0         ;
678      0596 0000      DW      0000D      ; WRITE PRE-COMPENSATION CYL
679      0598 00      DB      0         ;
680      0599 00      DB      0         ; CONTROL BYTE
681      059A 00 00 00  DB      0,0,0     ;
682      059D 0000      DW      0000D      ; LANDING ZONE
683      059F 00      DB      00D         ; SECTORS/TRACK
684      05A0 00      DB      0         ;
685
686          ;----- DRIVE TYPE 27      *** RESERVED***
687
688      05A1 0000      DW      0000D      ; CYLINDERS
689      05A3 00      DB      00D         ; HEADS
690      05A4 0000      DW      0         ;
691      05A6 0000      DW      0000D      ; WRITE PRE-COMPENSATION CYL
692      05A8 00      DB      0         ;
693      05A9 00      DB      0         ; CONTROL BYTE
694      05AA 00 00 00  DB      0,0,0     ;
695      05AD 0000      DW      0000D      ; LANDING ZONE
696      05AF 00      DB      00D         ; SECTORS/TRACK
697      05B0 00      DB      0         ;
698
699          ;----- DRIVE TYPE 28      *** RESERVED***
700
701      05B1 0000      DW      0000D      ; CYLINDERS
702      05B3 00      DB      00D         ; HEADS
703      05B4 0000      DW      0         ;
704      05B6 0000      DW      0000D      ; WRITE PRE-COMPENSATION CYL
705      05B8 00      DB      0         ;
706      05B9 00      DB      0         ; CONTROL BYTE
707      05BA 00 00 00  DB      0,0,0     ;
708      05BD 0000      DW      0000D      ; LANDING ZONE
709      05BF 00      DB      00D         ; SECTORS/TRACK
710      05C0 00      DB      0         ;
711
712          ;----- DRIVE TYPE 29      *** RESERVED***
713
714      05C1 0000      DW      0000D      ; CYLINDERS
715      05C3 00      DB      00D         ; HEADS
716      05C4 0000      DW      0         ;
717      05C6 0000      DW      0000D      ; WRITE PRE-COMPENSATION CYL
718      05C8 00      DB      0         ;
719      05C9 00      DB      0         ; CONTROL BYTE
720      05CA 00 00 00  DB      0,0,0     ;
721      05CD 0000      DW      0000D      ; LANDING ZONE
722      05CF 00      DB      00D         ; SECTORS/TRACK
723      05D0 00      DB      0         ;
724
725          ;----- DRIVE TYPE 30      *** RESERVED***
726
727      05D1 0000      DW      0000D      ; CYLINDERS
728      05D3 00      DB      00D         ; HEADS
729      05D4 0000      DW      0         ;
730      05D6 0000      DW      0000D      ; WRITE PRE-COMPENSATION CYL
731      05D8 00      DB      0         ;
732      05D9 00      DB      0         ; CONTROL BYTE
733      05DA 00 00 00  DB      0,0,0     ;
734      05DD 0000      DW      0000D      ; LANDING ZONE
735      05DF 00      DB      00D         ; SECTORS/TRACK
736      05E0 00      DB      0         ;
737
738          ;----- DRIVE TYPE 31      *** RESERVED***
739
740      05E1 0000      DW      0000D      ; CYLINDERS
741      05E3 00      DB      00D         ; HEADS
742      05E4 0000      DW      0         ;
743      05E6 0000      DW      0000D      ; WRITE PRE-COMPENSATION CYL
744      05E8 00      DB      0         ;
745      05E9 00      DB      0         ; CONTROL BYTE
746      05EA 00 00 00  DB      0,0,0     ;
747      05ED 0000      DW      0000D      ; LANDING ZONE
748      05EF 00      DB      00D         ; SECTORS/TRACK
749      05F0 00      DB      0         ;
750
751          ;----- DRIVE TYPE 32      *** RESERVED***
752
753      05F1 0000      DW      0000D      ; CYLINDERS
754      05F3 00      DB      00D         ; HEADS
755      05F4 0000      DW      0         ;
756      05F6 0000      DW      0000D      ; WRITE PRE-COMPENSATION CYL
757      05F8 00      DB      0         ;
758      05F9 00      DB      0         ; CONTROL BYTE
759      05FA 00 00 00  DB      0,0,0     ;
760      05FD 0000      DW      0000D      ; LANDING ZONE
761      05FF 00      DB      00D         ; SECTORS/TRACK
762      0600 00      DB      0         ;
763
764          ;----- DRIVE TYPE 33      *** RESERVED***
765
766      0601 0000      DW      0000D      ; CYLINDERS
767      0603 00      DB      00D         ; HEADS
768      0604 0000      DW      0         ;
769      0606 0000      DW      0000D      ; WRITE PRE-COMPENSATION CYL
770      0608 00      DB      0         ;
771      0609 00      DB      0         ; CONTROL BYTE
772      060A 00 00 00  DB      0,0,0     ;

```



```

773 060D 0000      DW 0000D      ; LANDING ZONE
774 060F 0000      DB 00D        ; SECTORS/TRACK
775 0610 0000      DB 0
776
777 ;----- DRIVE TYPE 34 *** RESERVED***
778
779 0611 0000      DW 0000D      ; CYLINDERS
780 0613 0000      DB 00D        ; HEADS
781 0614 0000      DW 0
782 0616 0000      DW 0000D      ; WRITE PRE-COMPENSATION CYL
783 0618 0000      DB 0
784 0619 0000      DB 0
785 061A 00 00 00 DW 0,0,0
786 061D 0000      DW 0000D      ; LANDING ZONE
787 061F 0000      DB 00D        ; SECTORS/TRACK
788 0620 0000      DB 0
789
790 ;----- DRIVE TYPE 35 *** RESERVED***
791
792 0621 0000      DW 0000D      ; CYLINDERS
793 0623 0000      DB 00D        ; HEADS
794 0624 0000      DW 0
795 0626 0000      DW 0000D      ; WRITE PRE-COMPENSATION CYL
796 0628 0000      DB 0
797 0629 0000      DB 0
798 062A 00 00 00 DW 0,0,0
799 062D 0000      DW 0000D      ; LANDING ZONE
800 062F 0000      DB 00D        ; SECTORS/TRACK
801 0630 0000      DB 0
802
803 ;----- DRIVE TYPE 36 *** RESERVED***
804
805 0631 0000      DW 0000D      ; CYLINDERS
806 0633 0000      DB 00D        ; HEADS
807 0634 0000      DW 0
808 0636 0000      DW 0000D      ; WRITE PRE-COMPENSATION CYL
809 0638 0000      DB 0
810 0639 0000      DB 0
811 063A 00 00 00 DW 0,0,0
812 063D 0000      DW 0000D      ; LANDING ZONE
813 063F 0000      DB 00D        ; SECTORS/TRACK
814 0640 0000      DB 0
815
816 ;----- DRIVE TYPE 37 *** RESERVED***
817
818 0641 0000      DW 0000D      ; CYLINDERS
819 0643 0000      DB 00D        ; HEADS
820 0644 0000      DW 0
821 0646 0000      DW 0000D      ; WRITE PRE-COMPENSATION CYL
822 0648 0000      DB 0
823 0649 0000      DB 0
824 064A 00 00 00 DW 0,0,0
825 064D 0000      DW 0000D      ; LANDING ZONE
826 064F 0000      DB 00D        ; SECTORS/TRACK
827 0650 0000      DB 0
828
829 ;----- DRIVE TYPE 38 *** RESERVED***
830
831 0651 0000      DW 0000D      ; CYLINDERS
832 0653 0000      DB 00D        ; HEADS
833 0654 0000      DW 0
834 0656 0000      DW 0000D      ; WRITE PRE-COMPENSATION CYL
835 0658 0000      DB 0
836 0659 0000      DB 0
837 065A 00 00 00 DW 0,0,0
838 065D 0000      DW 0000D      ; LANDING ZONE
839 065F 0000      DB 00D        ; SECTORS/TRACK
840 0660 0000      DB 0
841
842 ;----- DRIVE TYPE 39 *** RESERVED***
843
844 0661 0000      DW 0000D      ; CYLINDERS
845 0663 0000      DB 00D        ; HEADS
846 0664 0000      DW 0
847 0666 0000      DW 0000D      ; WRITE PRE-COMPENSATION CYL
848 0668 0000      DB 0
849 0669 0000      DB 0
850 066A 00 00 00 DW 0,0,0
851 066D 0000      DW 0000D      ; LANDING ZONE
852 066F 0000      DB 00D        ; SECTORS/TRACK
853 0670 0000      DB 0
854
855 ;----- DRIVE TYPE 40 *** RESERVED***
856
857 0671 0000      DW 0000D      ; CYLINDERS
858 0673 0000      DB 00D        ; HEADS
859 0674 0000      DW 0
860 0676 0000      DW 0000D      ; WRITE PRE-COMPENSATION CYL
861 0678 0000      DB 0
862 0679 0000      DB 0
863 067A 00 00 00 DW 0,0,0
864 067D 0000      DW 0000D      ; LANDING ZONE
865 067F 0000      DB 00D        ; SECTORS/TRACK
866 0680 0000      DB 0
867
868 ;----- DRIVE TYPE 41 *** RESERVED***
869
870 0681 0000      DW 0000D      ; CYLINDERS
871 0683 0000      DB 00D        ; HEADS
872 0684 0000      DW 0
873 0686 0000      DW 0000D      ; WRITE PRE-COMPENSATION CYL
874 0688 0000      DB 0
875 0689 0000      DB 0
876 068A 00 00 00 DW 0,0,0
877 068D 0000      DW 0000D      ; LANDING ZONE
878 068F 0000      DB 00D        ; SECTORS/TRACK
879 0690 0000      DB 0
880
881 ;----- DRIVE TYPE 42 *** RESERVED***
882
883 0691 0000      DW 0000D      ; CYLINDERS
884 0693 0000      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          ; CONTROL BYTE
889 069A 00 00 00   DW      0,0,0
890 069D 0000       DW      0000      ; 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 06A6 0000       DW      0000D     ; WRITE PRE-COMPENSATION CYL
900 06A8 00          DB      0
901 06A9 00          DB      0          ; CONTROL BYTE
902 06AA 00 00 00   DW      0,0,0
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 06B6 0000       DW      0000D     ; WRITE PRE-COMPENSATION CYL
913 06B8 00          DB      0
914 06B9 00          DB      0          ; CONTROL BYTE
915 06BA 00 00 00   DW      0,0,0
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          ; CONTROL BYTE
928 06CA 00 00 00   DW      0,0,0
929 06CD 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          ; CONTROL BYTE
941 06DA 00 00 00   DW      0,0,0
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          ; CONTROL BYTE
954 06EA 00 00 00   DW      0,0,0
955 06ED 0000       DW      0000D     ; LANDING ZONE
956 06EF 00          DB      00D      ; SECTORS/TRACK
957 06F0 00          DB      0
958
959          ;----- BOOT LOADER INTERRUPT
960
961          IP      =      $
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          ;          USE INT 15 H AH= 0C0H
969          06F5     CONF_TBL_1 ; CONFIGURATION TABLE FOR THIS SYSTEM
970          06F5 0008 DW      CONF E-CONF_TBL-2 ; LENGTH OF FOLLOWING TABLE
971          06F7 FC   DB      MODEL_BYTE ; SYSTEM MODEL BYTE
972          06F8 00   DB      SUB_MODEL_BYTE ; SYSTEM SUB MODEL TYPE BYTE
973          06F9 01   DB      BIOS_LEVEL ; BIOS REVISION LEVEL
974          06FA 70   DB      01110000B ; 10000000 = DMA CHANNEL 3 USE BY BIOS
975          ; 01000000 = CASCADED INTERRUPT LEVEL 2
976          ; 00100000 = REAL TIME CLOCK AVAILABLE
977          ; 00010000 = KEYBOARD SCAN CODE HOOK 1AH
978          06FB 00   DB      0 ; RESERVED
979          06FC 00   DB      0 ; RESERVED
980          06FD 00   DB      0 ; RESERVED
981          06FE 00   DB      0 ; RESERVED
982          = 06FF     CONF_E EQU      $ ; RESERVED FOR EXPANSION
983
984          ;----- BAUD RATE INITIALIZATION TABLE
985
986          06FF     IP      =      $
987          0729     ;:-      ORG      0E729H
988          0729     ORG      00729H
989          0729 0417 AI      DW      1047 ; 110 BAUD ; TABLE OF VALUES
990          072B 0300 AI      DW      768 ; 150 ; FOR INITIALIZATION
991          072D 0180 AI      DW      384 ; 300
992          072F 00C0 AI      DW      192 ; 600
993          0731 0060 AI      DW      96 ; 1200
994          0733 0030 AI      DW      48 ; 2400
995          0735 0018 AI      DW      24 ; 4800
996          0737 000C AI      DW      12 ; 9600
997
998          ;----- RS232
999
1000         ;:-      ORG      0E739H

```

```

1001 0739          ORG      00739H
1002 = 0739      R5232_10 EQU     $
1003 0739 E9 0000 E      JMP     R5232_10_1      ; VECTOR ON TO MOVED R5232 CODE
1004
1005          ;----- KEYBOARD
1006
1007          ;:- ORG      0E82EH
1008 082E        ORG      0082EH
1009 = 082E      KEYBOARD_10 EQU    $
1010 082E E9 0000 E      JMP     KEYBOARD_10_1    ; VECTOR ON TO MOVED KEYBOARD CODE
1011
1012          ;----- TABLE OF SHIFT KEYS AND MASK VALUES (EARLY PC)
1013
1014          ;:- ORG      0E87EH
1015 087E        ORG      0087EH
1016 087E 52     K6      DB      INS_KEY      ; INSERT KEY
1017 087F 3A 45 46 38 1D  DB      CAPS_KEY,NUM_KEY,SCROLL_KEY,ALT_KEY,CTL_KEY
1018 088A 2A 36         DB      LEFT_KEY,RIGRT_KEY
1019 = 0008      K6L     EQU     $-K6
1020
1021          ;----- SHIFT_MASK_TABLE
1022
1023 0886 80     K7      DB      INS_SHIFT      ; INSERT MODE SHIFT
1024 0887 40 20 10 08 04  DB      CAPS_SHIFT,NUM_SHIFT,SCROLL_SHIFT,ALT_SHIFT,CTL_SHIFT
1025 088C 02 01         DB      LEFT_SHIFT,RIGRT_SHIFT
1026
1027          ;----- SCAN CODE TABLES
1028
1029 088E 1B FF 00 FF FF FF FF  K8      DB      27,-1,0,-1,-1,-1,30,-1,-1,-1,-1,31
1030          1E FF FF FF FF FF  DB
1031 089A FF 7F FF 11 17 05     DB      -1,127,-1,17,23,5,18,20,25,21,9,15
1032          12 14 19 15 09 0F  DB
1033 08A6 10 1B 1D 0A FF 01     DB      16,27,29,10,-1,1,19,4,6,7,8,10
1034          13 04 06 07 08 0A     DB
1035 08B2 08 0C FF FF FF FF FF  DB      11,12,-1,-1,-1,-1,28,26,24,3,22,2
1036          1C 1A 18 03 16 02     DB
1037 08BE 0E 0D FF FF FF FF FF  DB      14,13,-1,-1,-1,-1,-1,-1,' ','-1
1038          FF FF 20 FF
1039
1040          ;----- CTL TABLE SCAN
1041
1042 08C8 5E 5F 60 61 62 63     K9      DB      94,95,96,97,98,99,100,101,102,103,-1,-1
1043          64 65 66 67 FF FF     DB
1044 08D4 77 FF 84 FF 73 FF     DB      119,-1,132,-1,115,-1,116,-1,117,-1,118,-1
1045          74 FF 75 FF 76 FF     DB      -1
1046
1047          ;----- LC TABLE
1048
1049 08E1 1B 31 32 33 34 35     K10     DB      01BH,'1234567890-='',08H,09H
1050          36 37 38 39 30 2D     DB
1051          30 08 09
1052 08F0 71 77 65 72 74 79     DB      'qwertyuiop[]',0DH,-1,'asdfghjkl;',027H
1053          75 69 6F 70 5B 5D     DB
1054          0D FF 61 73 64 66     DB
1055          67 68 6A 6B 6C 3B     DB
1056          27
1057 0909 60 FF 5C 7A 78 63     DB      60H,-1,5CH,'zxcvbnm,./',-1,'*',-1,' '
1058          76 62 6E 6D 2C 2E     DB
1059          2F FF 2A FF 20
1060 091A FF
1061          DB      -1
1062
1063          ;----- UC TABLE
1064
1065 091B 1B 21 40 23 24 25     K11     DB      27,'!@#$%',37,05EH,'&*()_+',08H,0
1066          5E 26 2A 2B 29 5F     DB
1067          2B 08 00
1068 092A 51 57 45 52 54 59     DB      'QWERTYUIOP[]',0DH,-1,'ASDFGHJKL;'
1069          55 49 4F 50 7B 7D     DB
1070          0D FF 41 53 44 46     DB
1071          47 48 4A 4B 4C 3A     DB
1072          22
1073 0943 7E FF 7C 5A 58 43     DB      07EH,-1,'|ZXCVCBNM<?>,-1,0,-1,' ','-1
1074          56 42 4E 4D 3C 3E     DB
1075          3F FF 00 FF 20 FF
1076
1077          ;----- UC TABLE SCAN
1078 0955 54 55 56 57 58 59     K12     DB      84,85,86,87,88,89
1079 095B 5A 5B 5C 5D         DB      90,91,92,93
1080
1081          ;----- ALT TABLE SCAN
1082 095F 68 69 6A 6B 6C     K13     DB      104,105,106,107,108
1083          6D 6E 6F 70 71         DB      109,110,111,112,113
1084
1085          ;----- NUM STATE TABLE
1086 0969 37 38 39 2D 34 35     K14     DB      '789-456+1230.'
1087          36 2B 31 32 33 30     DB
1088          2E
1089
1090          ;----- BASE CASE TABLE
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
1095          ;----- KEYBOARD INTERRUPT
1096
1097          ;:- ORG      0E987H
1098 0987        ORG      00987H
1099 = 0987      KB_INT   EQU     $
1100 0987 E9 0000 E      JMP     KB_INT_1        ; VECTOR ON TO MOVED KEYBOARD HANDLER

```

SECTION 5

```

1101 PAGE
1102 ;----- DISKETTE I/O
1103
1104 ;1- ORG 0EC59H
1105 0C59 ORG 00C59H
1106 = 0C59 DISKETTE_10 EQU $
1107 0C59 E9 0000 E JMP DISKETTE_10_1 ; VECTOR ON TO MOVED DISKETTE CODE
1108
1109 ;----- DISKETTE INTERRUPT
1110
1111 ;1- ORG 0EF57H
1112 0F57 ORG 00F57H
1113 = 0F57 DISK_INT EQU $
1114 0F57 E9 0000 E JMP DISK_INT_1 ; VECTOR ON TO MOVED DISKETTE HANDLER
1115
1116 ;----- DISKETTE PARAMETERS
1117
1118 ;1- ORG 0EF77H
1119 0F77 ORG 00F77H
1120
1121 -----
1122 ; DISK_BASE ;
1123 ; THIS IS THE SET OF PARAMETERS REQUIRED FOR ;
1124 ; DISKETTE OPERATION, THEY ARE POINTED AT BY THE ;
1125 ; DATA VARIABLE DISK_POINTER, TO MODIFY THE PARAMETERS, ;
1126 ; BUILD ANOTHER PARAMETER BLOCK AND POINT AT IT ;
1127 -----
1128
1129 DISK_BASE LABEL BYTE
1130
1131 0F77 DF DB 11011111B ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
1132 0FC8 02 DB 2 ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
1133 0FC9 25 DB MOTOR_WAIT ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
1134 0FCA 02 DB 2 ; 512 BYTES/SECTOR
1135 0FCB 0F DB 15 ; EOT ( LAST SECTOR ON TRACK)
1136 0FCC 1B DB 01BH ; GAP LENGTH
1137 0FCD FF DB 0F7H ; DTL
1138 0FCE 54 DB 054H ; GAP LENGTH FOR FORMAT
1139 0FCF F6 DB 0F6H ; FILL BYTE FOR FORMAT
1140 0FD0 0F DB 15 ; HEAD SETTLE TIME (MILLISECONDS)
1141 0FD1 06 DB 8 ; MOTOR START TIME (1/8 SECONDS)
1142
1143 ;----- PRINTER I/O
1144
1145 ;1- ORG 0EFD2H
1146 0FD2 ORG 00FD2H
1147 = 0FD2 PRINTER_10 EQU $
1148 0FD2 E9 0000 E JMP PRINTER_10_1 ; VECTOR ON TO MOVED PRINTER CODE
1149
1150 ;----- FOR POSSIBLE COMPATIBILITY ENTRY POINTS
1151
1152 ;1- ORG 0F045H
1153 1045 ORG 01045H
1154 ASSUME CS:CODE,DS:DATA
1155
1156 EXTRN SET_MODE:NEAR
1157 EXTRN SET_CTYPE:NEAR
1158 EXTRN SET_CPOS:NEAR
1159 EXTRN READ_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_CURRENT: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 MI DW OFFSET SET_MODE ; TABLE OF ROUTINES WITHIN VIDEO I/O
1174 1047 0000 E DW OFFSET SET_CTYPE ; EXIT STACK VALUES MAY BE
1175 1049 0000 E DW OFFSET SET_CPOS ; DIFFERENT DEPENDING ON THE
1176 104B 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 MIL EQU $-M1
1190
1191 ;1- ORG 0F065H
1192 1065 ORG 01065H
1193 = 1065 VIDEO_10 EQU $
1194 1065 E9 0000 E JMP VIDEO_10_1 ; VECTOR ON TO MOVED VIDEO CODE
1195
1196 ;----- VIDEO PARAMETERS --- INIT_TABLE
1197
1198 ;1- ORG 0F0A4H
1199 10A4 ORG 010A4H
1200
1201 10A4 VIDEO_PARAMS LABEL BYTE
1202 10A4 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 10B0 00 00 00 00 DB 0,0,0,0
1206 = 0010 M4 EQU $-VIDEO_PARAMS
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 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 10DD 00 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 00      DB      0,0,0,0
1222
1223 10E4 0800                  M5      DW      2048      ; TABLE OF REGEN LENGTHS
1224 10E4 1000                  DW      4096      ; 40X25
1225 10E8 4000                  DW      16384     ; 80X25
1226 10EA 4000                  DW      16384     ; GRAPHICS
1227
1228 I----- COLUMNS
1229
1230 10EC 28 28 50 50 28 28    M6      DB      40,40,80,80,40,40,80,80
1231 50 50
1232 I----- 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 I----- MEMORY SIZE
1237
1238 I-- 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 I----- EQUIPMENT DETERMINE
1244
1245 I-- ORG 0F84DH
1246 184D ORG 0184DH
1247 = 184D EQUIPMENT_1 EQU $
1248 184D E9 0000 E          JMP     EQUIPMENT_1 ; VECTOR ON TO MOVED BIOS CODE
1249
1250 I----- CASSETTE (NO BIOS SUPPORT)
1251
1252 I-- 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 I-----
1258 ; CHARACTER GENERATOR GRAPHICS FOR 320X200 AND 640X200 GRAPHICS ;
1259 I-----
1260
1261 1A6E I-- ORG 0FA6EH
1262 1A6E ORG 01A6EH
1263 1A6E 00 00 00 00 00 00 00 00 DB      00H,000H,000H,000H,000H,000H,000H,000H ; D_00 BLANK
1264 00 00
1265 1A76 7E 81 A5 81 BD 99    DB      07EH,081H,0A5H,081H,0BDH,099H,081H,07EH ; D_01 SMILING FACE
1266 81 7E
1267 1A7E 7E FF DB FF C3 E7    DB      07EH,0FFH,0DBH,0FFH,0C3H,0E7H,0FFH,07EH ; D_02 SMILING FACE N
1268 FF 7E
1269 1A86 6C FE FE FE 7C 38    DB      06CH,0FEH,0FEH,0FEH,07CH,038H,010H,000H ; D_03 HEART
1270 10 00
1271 1A8E 10 38 7C FE 7C 38    DB      010H,038H,07CH,0FEH,07CH,038H,010H,000H ; D_04 DIAMOND
1272 10 00
1273 1A96 3C 7C 38 FE 7C 7C    DB      038H,07CH,038H,0FEH,0FEH,07CH,038H,07CH ; D_05 CLUB
1274 38 7C
1275 1A9E 10 10 38 7C FE 7C    DB      010H,010H,038H,07CH,0FEH,07CH,038H,07CH ; D_06 SPADE
1276 38 7C
1277 1AA6 00 00 18 3C 3C 18    DB      000H,000H,018H,03CH,03CH,018H,000H,000H ; D_07 BULLET
1278 00 00
1279 1AAE FF FF E7 C3 C3 E7    DB      0FFH,0FFH,0E7H,0C3H,0C3H,0E7H,0FFH,0FFH ; D_08 BULLET NEG
1280 FF FF
1281 1AB6 00 3C 66 42 42 66    DB      000H,03CH,066H,042H,042H,066H,03CH,000H ; D_09 CIRCLE
1282 3C 00
1283 1ABE FF C3 99 BD 80 99    DB      0FFH,0C3H,099H,0BDH,0BDH,099H,0C3H,0FFH ; D_0A CIRCLE NEG
1284 C3 FF
1285 1AC6 0F 07 0F 7D CC CC    DB      00FH,007H,00FH,07DH,0CCH,0CCH,0CCH,078H ; D_0B MALE
1286 CC 78
1287 1ACE 3C 66 66 66 3C 18    DB      03CH,066H,066H,066H,03CH,018H,07EH,018H ; D_0C FEMALE
1288 7E 18
1289 1AD6 3F 3F 3F 30 70      DB      03FH,033H,03FH,030H,030H,070H,070H,0E0H ; D_0D EIGHTH NOTE
1290 F0 E0
1291 1ADE 7F 63 7F 63 63 67    DB      07FH,063H,07FH,063H,063H,067H,0E6H,0C0H ; D_0E TWO 1/16 NOTE
1292 E6 00
1293 1AE6 99 5A 3C E7 E7 3C    DB      099H,05AH,03CH,0E7H,0E7H,03CH,05AH,099H ; D_0F SUN
1294 5A 99
1295
1296 1AEE 80 E0 F8 FE F8 E0      DB      080H,0E0H,0F8H,0FEH,0F8H,0E0H,080H,000H ; D_10 R ARROWHEAD
1297 80 00
1298 1AF6 02 0E 3E FE 3E 0E    DB      002H,00EH,03EH,0FEH,03EH,00EH,002H,000H ; D_11 L ARROWHEAD
1299 02 00
1300 1AFE 18 3C 7E 18 18 7E    DB      018H,03CH,07EH,018H,018H,07EH,03CH,018H ; D_12 ARROW 2 VERT
1301 3C 18
1302 1B06 66 66 66 66 66 00    DB      066H,066H,066H,066H,066H,000H,066H,000H ; D_13 2 EXCLAMATIONS
1303 66 FF
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 63 6C 6C 38    DB      03EH,063H,038H,06CH,06CH,038H,0CCH,078H ; D_15 SECTION
1307 C3 7E
1308 1B1E 00 00 00 00 00 7E    DB      000H,000H,000H,000H,07EH,07EH,07EH,000H ; D_16 RECTANGLE
1309 7E 00
1310 1B26 18 3C 7E 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 7E 18 18 18    DB      018H,03CH,07EH,018H,018H,018H,018H,000H ; D_18 ARROW VRT UP
1313 18 00
1314 1B36 18 18 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,0CCH,0FEH,06FH,00CCH,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 C0 C0 FE 00    DB      000H,000H,0C0H,0C0H,0C0H,0FEH,000H,000H ; D_1C NOT INVERTED
1321 00 00
1322 1B56 00 66 FF 66 24      DB      000H,024H,066H,0FFH,066H,024H,000H,000H ; D_1D ARROW 2 HORZ
1323 00 00
1324 1B5E 00 18 3C 7E FF FF    DB      000H,018H,03CH,07EH,0FEH,0FFH,0FFH,000H,000H ; D_1E ARROWHEAD UP
1325 00 00
1326 1B66 00 FF FF 7E 3C 18    DB      000H,0FFH,0FFH,07EH,03CH,018H,000H,000H ; D_1F ARROWHEAD DOWN
1327 00 00
1328

```

SECTION 5

1329	IB6E	00	00	00	00	00	00	DB	000H,000H,000H,000H,000H,000H,000H,000H ;	D_20	SPACE
1330		00	00								
1331	IB7F	30	78	30	30	00	00	DB	030H,078H,078H,030H,030H,000H,030H,000H ;	D_21	! EXCLAMATION
1332		00	00								
1333	IB7E	6C	6C	6C	00	00	00	DB	06CH,06CH,06CH,000H,000H,000H,000H,000H ;	D_22	" QUOTATION
1334		00	00								
1335	IB86	6C	6C	FE	6C	FE	6C	DB	06CH,06CH,0FEH,06CH,0FEH,06CH,06CH,000H ;	D_23	# LB.
1336		6C	00								
1337	IB8E	30	7C	0C	78	0C	F8	DB	030H,07CH,0C0H,078H,00CH,0F8H,030H,000H ;	D_24	\$ DOLLAR SIGN
1338		00	00								
1339	IB96	00	C6	18	30	66		DB	000H,0C6H,0CCH,018H,030H,066H,0C6H,000H ;	D_25	% PERCENT
1340		C6	00								
1341	IB9E	38	6C	38	76	DC	CC	DB	038H,06CH,038H,076H,0DCH,0CCH,076H,000H ;	D_26	& AMPERSAND
1342		76	00								
1343	IBA6	60	C0	60	00	00	00	DB	060H,060H,0C0H,000H,000H,000H,000H,000H ;	D_27	' APOSTROPHE
1344		00	00								
1345	IBA8	18	30	60	60	60	00	DB	018H,030H,060H,060H,060H,030H,018H,000H ;	D_28	(L. PARENTHESIS
1346		18	00								
1347	IBB6	60	30	18	18	18	30	DB	060H,030H,018H,018H,018H,030H,060H,000H ;	D_29) R. PARENTHESIS
1348		60	00								
1349	IBBE	00	66	3C	FF	3C	66	DB	000H,066H,03CH,0FFH,03CH,066H,000H,000H ;	D_2A	* ASTERISK
1350		00	00								
1351	IBC6	00	30	30	FC	30	60	DB	000H,030H,030H,0FCH,030H,0FCH,000H,000H ;	D_2B	+ PLUS
1352		00	00								
1353	IBCE	00	00	00	00	00	30	DB	000H,000H,000H,000H,000H,030H,030H,060H ;	D_2C	, COMMA
1354		30	60								
1355	IBD6	00	00	00	FC	00	00	DB	000H,000H,000H,0FCH,000H,000H,000H,000H ;	D_2D	- DASH
1356		00	00								
1357	IBDE	00	00	00	00	00	30	DB	000H,000H,000H,000H,000H,030H,030H,000H ;	D_2E	_ PERIOD
1358		30	00								
1359	IBE6	06	0C	18	30	60	C0	DB	006H,00CH,018H,030H,060H,0C0H,080H,000H ;	D_2F	/ SLASH
1360		80	00								
1361											
1362	IBEE	7C	C6	CE	DE	F6	E6	DB	07CH,0C6H,0CEH,0DEH,0F6H,0E6H,07CH,000H ;	D_30	0
1363		7C	00								
1364	IBF6	30	70	30	30	30	30	DB	030H,070H,030H,030H,030H,030H,0FCH,000H ;	D_31	1
1365		FC	00								
1366	IBFE	78	CC	0C	38	60	CC	DB	078H,0CCH,00CH,038H,060H,0CCH,0FCH,000H ;	D_32	2
1367		FC	00								
1368	IC0E	78	CC	0C	38	0C	CC	DB	078H,0CCH,00CH,038H,00CH,0CCH,078H,000H ;	D_33	3
1369		78	00								
1370	IC0E	1C	3C	6C	CC	FE	0C	DB	01CH,03CH,06CH,0FCH,030H,0CCH,01CH,000H ;	D_34	4
1371		1E	00								
1372	IC1E	FC	CD	F8	0C	0C	CC	DB	0FCH,0C0H,0F8H,00CH,00CH,0CCH,078H,000H ;	D_35	5
1373		78	00								
1374	IC1E	38	60	C0	F8	CC	CC	DB	038H,060H,0C0H,0F8H,0CCH,0CCH,078H,000H ;	D_36	6
1375		78	00								
1376	IC2E	FC	CC	0C	18	30	30	DB	0FCH,0CCH,00CH,018H,030H,030H,030H,000H ;	D_37	7
1377		30	00								
1378	IC2E	78	CC	7C	CC	CC	CC	DB	078H,0CCH,0CCH,078H,0CCH,0CCH,078H,000H ;	D_38	8
1379		78	00								
1380	IC3E	78	CC	7C	0C	18		DB	078H,0CCH,0CCH,07CH,00CH,018H,070H,000H ;	D_39	9
1381		70	00								
1382	IC3E	00	30	30	00	00	30	DB	000H,030H,030H,000H,000H,030H,030H,000H ;	D_3A	: COLON
1383		30	00								
1384	IC4E	00	30	30	00	00	00	DB	000H,030H,030H,000H,000H,030H,030H,060H ;	D_3B	: SEMICOLON
1385		30	60								
1386	IC4E	18	30	60	C0	60	30	DB	018H,030H,060H,0C0H,0C0H,030H,018H,000H ;	D_3C	< LESS THAN
1387		18	00								
1388	IC5E	00	FC	00	00	FC		DB	000H,000H,0FCH,000H,000H,0FCH,000H,000H ;	D_3D	= EQUAL
1389		00	00								
1390	IC5E	60	30	18	0C	18	30	DB	060H,030H,018H,00CH,018H,030H,060H,000H ;	D_3E	> GREATER THAN
1391		60	00								
1392	IC6E	78	CC	0C	18	30	00	DB	078H,0CCH,00CH,018H,030H,000H,030H,000H ;	D_3F	? QUESTION MARK
1393		30	00								
1394											
1395	IC6E	7C	C6	DE	DE	DE	00	DB	07CH,0C6H,0DEH,0DEH,0DEH,0C0H,078H,000H ;	D_40	@ AT
1396		78	00								
1397	IC7E	30	78	CC	CC	FC	CC	DB	030H,078H,0CCH,0CCH,0FCH,0CCH,0CCH,000H ;	D_41	A
1398		CC	00								
1399	IC7E	FC	66	7C	66	66		DB	0FCH,066H,066H,07CH,066H,066H,0FCH,000H ;	D_42	B
1400		FC	00								
1401	IC8E	3C	66	C0	C0	66		DB	03CH,066H,0C0H,0C0H,0C0H,066H,03CH,000H ;	D_43	C
1402		3C	00								
1403	IC8E	F8	6C	66	66	66	6C	DB	0F8H,06CH,066H,066H,066H,06CH,0F8H,000H ;	D_44	D
1404		F8	00								
1405	IC9E	FE	62	68	78	68	62	DB	0FEH,062H,068H,078H,068H,062H,0FEH,000H ;	D_45	E
1406		FE	00								
1407	IC9E	FE	62	68	78	68	60	DB	0FEH,062H,068H,078H,068H,060H,0F0H,000H ;	D_46	F
1408		F0	00								
1409	ICA6	3C	66	C0	C0	CE	66	DB	03CH,066H,0C0H,0C0H,0CEH,066H,03EH,000H ;	D_47	G
1410		3C	00								
1411	ICAE	CC	CC	FC	CC	CC	CC	DB	0CCH,0CCH,0CCH,0FCH,0CCH,0CCH,0CCH,000H ;	D_48	H
1412		CC	00								
1413	ICB6	78	30	30	30	30	30	DB	078H,030H,030H,030H,030H,030H,078H,000H ;	D_49	I
1414		78	00								
1415	ICBE	1E	0C	0C	0C	CC	CC	DB	01EH,00CH,00CH,00CH,0CCH,0CCH,078H,000H ;	D_4A	J
1416		78	00								
1417	ICC6	E6	66	6C	78	6C	66	DB	0E6H,066H,06CH,078H,06CH,066H,0E6H,000H ;	D_4B	K
1418		E6	00								
1419	ICCE	F0	60	60	60	62	66	DB	0F0H,060H,060H,060H,062H,066H,0FEH,000H ;	D_4C	L
1420		FE	00								
1421	ICD6	C6	EE	FE	FE	D6	C6	DB	0C6H,0EEH,0FEH,0FEH,0D6H,0C6H,0C6H,000H ;	D_4D	M
1422		C6	00								
1423	ICDE	C6	E6	F6	DE	CE	C6	DB	0C6H,0E6H,0F6H,0DEH,0CEH,0C6H,0C6H,000H ;	D_4E	N
1424		C6	00								
1425	ICE6	30	C6	C6	C6	6C		DB	038H,06CH,0C6H,0C6H,0C6H,06CH,038H,000H ;	D_4F	O
1426		38	00								
1427											
1428	ICEE	FC	66	66	7C	60	60	DB	0FCH,066H,066H,07CH,060H,060H,0F0H,000H ;	D_50	P
1429		FC	00								
1430	ICF6	78	CC	CC	CC	DC	78	DB	078H,0CCH,0CCH,0CCH,0CCH,078H,01CH,000H ;	D_51	Q
1431		1C	00								
1432	ICFE	FC	66	66	7C	6C	66	DB	0FCH,066H,066H,07CH,06CH,066H,0E6H,000H ;	D_52	R
1433		FC	00								
1434	ID0E	78	CC	EE	70	1C	CC	DB	078H,0CCH,0E0H,070H,01CH,0CCH,078H,000H ;	D_53	S
1435		78	00								
1436	ID0E	78	CC	30	30	30	30	DB	0FCH,0B4H,030H,030H,030H,030H,078H,000H ;	D_54	T
1437		78	00								
1438	ID1E	CC	CC	CC	CC	CC	CC	DB	0CCH,0CCH,0CCH,0CCH,0CCH,0CCH,0CCH,000H ;	D_55	U
1439		CC	00								
1440	ID1E	CC	CC	CC	CC	78		DB	0CCH,0CCH,0CCH,0CCH,0CCH,078H,030H,000H ;	D_56	V
1441		CC	00								
1442	ID2E	C6	C6	C6	D6	FE	EE	DB	0C6H,0C6H,0C6H,0D6H,0FEH,0EEH,0C6H,000H ;	D_57	W

```

1443      C6 00
1444 ID2E C6 C6 C6 38 38 6C      DB      0C6H,0C6H,06CH,038H,038H,06CH,0C6H,000H ; D_58 X
1445      C6 00
1446 ID36 CC CC CC 78 30 30      DB      0CCH,0CCH,0CCH,078H,030H,030H,078H,000H ; D_59 Y
1447      78 00
1448 ID3E FE C6 8C 18 32 66      DB      0FEH,0C6H,08CH,018H,032H,066H,0FEH,000H ; D_5A Z
1449      FE 00
1450 ID46 78 60 60 60 60 60      DB      078H,060H,060H,060H,060H,060H,078H,000H ; D_5B [ LEFT BRACKET
1451      78 00
1452 ID4C 0C 60 30 18 0C 06      DB      0C0H,060H,030H,018H,0C0H,006H,002H,000H ; D_5C \ BACKSLASH
1453      02 00
1454 ID56 78 18 18 18 18 18      DB      078H,018H,018H,018H,018H,018H,078H,000H ; D_5D ] RIGHT BRACKET
1455      78 00
1456 ID5E 10 38 6C C6 00 00      DB      010H,038H,06CH,0C6H,000H,000H,000H,000H ; D_5E ^ CIRCUMFLEX
1457      00 00
1458 ID66 0F 00 00 00 00 00      DB      000H,000H,000H,000H,000H,000H,000H,0FFH ; D_5F _ UNDERSCORE
1459      00 FF
1460
1461 ID6E 30 30 18 00 00 00      DB      030H,030H,018H,000H,000H,000H,000H,000H ; D_60 ' APOSTROPHE REV
1462      00 00
1463 ID76 00 00 78 0C 7C CC      DB      000H,000H,078H,00CH,07CH,0CCH,076H,000H ; D_61 a
1464      76 00
1465 ID7E E0 60 60 7C 66 66      DB      0E0H,060H,060H,07CH,066H,066H,0DCH,000H ; D_62 b
1466      DC 00
1467 ID86 00 00 78 CC C0 CC      DB      000H,000H,078H,0CCH,0C0H,0CCH,078H,000H ; D_63 c
1468      78 00
1469 ID8E 1C 0C 0C 7C CC CC      DB      01CH,00CH,00CH,07CH,0CCH,0CCH,076H,000H ; D_64 d
1470      76 00
1471 ID96 00 00 78 CC FC C0      DB      000H,000H,078H,0CCH,0FCCH,0C0H,078H,000H ; D_65 e
1472      78 00
1473 ID9E 38 60 F0 60 60 60      DB      038H,06CH,060H,0F0H,060H,060H,0F0H,000H ; D_66 f
1474      F0 00
1475 IDA6 00 00 76 CC CC 7C      DB      000H,000H,076H,0CCH,0CCH,07CH,00CH,0F8H ; D_67 g
1476      0C F8
1477 IDAE E0 60 76 66 66 66      DB      0E0H,060H,06CH,076H,066H,066H,0E6H,000H ; D_68 h
1478      E6 00
1479 IDB6 30 00 70 30 30 30      DB      030H,000H,070H,030H,030H,030H,070H,000H ; D_69 i
1480      78 00
1481 IDBE 0C 00 0C 0C 0C CC      DB      00CH,000H,00CH,00CH,00CH,0CCH,0CCH,078H ; D_6A j
1482      CC 78
1483 IDC6 E0 60 66 6C 78 6C      DB      0E0H,060H,066H,06CH,078H,06CH,0E6H,000H ; D_6B k
1484      E6 00
1485 IDC8 70 30 30 30 30 30      DB      070H,030H,030H,030H,030H,030H,070H,000H ; D_6C l
1486      78 00
1487 IDD6 00 00 CC FE FE D6      DB      000H,000H,0CCH,0FEH,0FEH,0D6H,0C6H,000H ; D_6D m
1488      C6 00
1489 IDDE 00 00 F8 CC CC CC      DB      000H,000H,0F8H,0CCH,0CCH,0CCH,0CCH,000H ; D_6E n
1490      CC 00
1491 IDE6 00 00 78 CC CC CC      DB      000H,000H,078H,0CCH,0CCH,0CCH,078H,000H ; D_6F o
1492      78 00
1493
1494 IDEE 00 00 DC 66 66 7C      DB      000H,000H,0DCH,066H,066H,07CH,060H,0F0H ; D_70 p
1495      60 F0
1496 IDFE 6 00 00 76 CC CC 7C      DB      000H,000H,076H,0CCH,0CCH,07CH,00CH,01EH ; D_71 q
1497      0C 1E
1498 IDFE 00 00 DC 76 66 60      DB      000H,000H,0DCH,076H,066H,060H,0F0H,000H ; D_72 r
1499      F0 00
1500 IE06 00 00 7C C0 78 0C      DB      000H,000H,07CH,0C0H,078H,00CH,0F8H,000H ; D_73 s
1501      F8 00
1502 IE0E 10 30 7C 30 30 34      DB      010H,030H,07CH,030H,030H,034H,018H,000H ; D_74 t
1503      18 00
1504 IE16 00 00 CC CC CC CC      DB      000H,000H,0CCH,0CCH,0CCH,0CCH,030H,076H,000H ; D_75 u
1505      76 00
1506 IE1E 00 00 CC CC CC 78      DB      000H,000H,0CCH,0CCH,0CCH,078H,030H,000H ; D_76 v
1507      30 00
1508 IE26 00 00 C6 D6 FE FE      DB      000H,000H,0C6H,0D6H,0FEH,0FEH,06CH,000H ; D_77 w
1509      6C 00
1510 IE2E 00 00 C6 6C 38 6C      DB      000H,000H,0C6H,06CH,038H,06CH,0C6H,000H ; D_78 x
1511      C6 00
1512 IE36 00 00 CC CC CC 7C      DB      000H,000H,0CCH,0CCH,0CCH,07CH,00CH,0F8H ; D_79 y
1513      0C F8
1514 IE3E 00 00 FC 98 30 64      DB      000H,000H,0FCH,098H,030H,064H,0FCH,000H ; D_7A z
1515      FC 00
1516 IE46 1C 30 30 E0 30 30      DB      01CH,030H,030H,0E0H,030H,030H,01CH,000H ; D_7B | LEFT BRACE
1517      1C 00
1518 IE4E 18 18 18 00 18 18      DB      018H,018H,018H,000H,018H,018H,018H,000H ; D_7C | BROKEN STROKE
1519      18 00
1520 IE56 E0 30 30 1C 30 30      DB      0E0H,030H,030H,01CH,030H,030H,0E0H,000H ; D_7D | RIGHT BRACE
1521      E0 00
1522 IE5E 76 DC 00 00 00 00      DB      076H,0DCH,000H,000H,000H,000H,000H,000H ; D_7E ~ TILDE
1523      00 00
1524 IE66 00 10 38 6C C6 C6      DB      000H,010H,038H,06CH,0C6H,0C6H,0FEH,000H ; D_7F Δ DELTA
1525      FE 00
1526
1527      ;----- TIME OF DAY
1528
1529      ;:- ORG OFE6EH
1530 IE6E      ORG 01E6EH
1531 = IE6E      TIME_OF_DAY EQU $
1532 IE6E E9 0000 E      JMP TIME_OF_DAY_1 ; VECTOR ON TO MOVED BIOS CODE
1533
1534      ;----- TIMER INTERRUPT
1535
1536      ;:- ORG OFEA5H
1537 IEA5      ORG 01EA5H
1538 = IEA5      TIMER_INT EQU $
1539 IEA5 E9 0000 E      JMP TIMER_INT_1 ; VECTOR ON TO MOVED BIOS CODE

```

```

1540 PAGE
1541 |----- VECTOR TABLE
1542
1543 |-- ORG 0FEF3H ; AT LOCATION 0FEF3H
1544 |----- VECTOR TABLE LABEL WORD ; VECTOR TABLE VALUES FOR POST TESTS
1545 |----- ORG 0FEF3H ; INT 08H - HARDWARE TIMER 0 IRQ 0
1546 |----- DW OFFSET TIMER_INT ; INT 09H - KEYBOARD IRQ 1
1547 |----- DW OFFSET KB_INT ; INT 0AH - SLAVE INTERRUPT INPUT IRQ 3
1548 |----- DW OFFSET D11 ; INT 0BH - IRQ 4
1549 |----- DW OFFSET D11 ; INT 0CH - IRQ 5
1550 |----- DW OFFSET D11 ; INT 0DH - IRQ 6
1551 |----- DW OFFSET D11 ; INT 0EH - DISKETTE IRQ 7
1552 |----- DW OFFSET D11 ; INT 0FH -
1553 |----- DW OFFSET D11 ; INT 0FH -
1554
1555 |----- SOFTWARE INTERRUPTS ( BIOS CALLS AND POINTERS )
1556
1557 |----- DW OFFSET VIDEO_IO ; INT 10H -- VIDEO DISPLAY
1558 |----- DW OFFSET EQUIPMENT ; INT 11H -- GET EQUIPMENT FLAG WORD
1559 |----- DW OFFSET MEMORY_SIZE_DET ; INT 12H -- GET REAL MODE MEMORY SIZE
1560 |----- DW OFFSET DISKETTE_IO ; INT 13H -- DISKETTE
1561 |----- DW OFFSET RS232_IO ; INT 14H -- COMMUNICATION ADAPTER
1562 |----- DW OFFSET CASSETTE_IO ; INT 15H -- EXPANDED BIOS FUNCTION CALL
1563 |----- DW OFFSET KEYBOARD_IO ; INT 16H -- KEYBOARD INPUT
1564 |----- DW OFFSET PRINTER_IO ; INT 17H -- PRINTER OUTPUT
1565 |----- DW 00000H ; INT 18H -- 0F600H INSERTED FOR BASIC
1566 |----- DW OFFSET BOOT_STRAP ; INT 19H -- BOOT FROM SYSTEM MEDIA
1567 |----- DW OFFSET TIME_OF_DAY ; INT 1AH -- TIME OF DAY
1568 |----- DW OFFSET DUMMY_RETURN ; INT 1BH -- KEYBOARD BREAK ADDRESS
1569 |----- DW OFFSET DUMMY_RETURN ; INT 1CH -- TIMER BREAK ADDRESS
1570 |----- DW OFFSET VIDEO_PARAMS ; INT 1DH -- VIDEO PARAMETERS
1571 |----- DW OFFSET DISK_BASE ; INT 1EH -- DISKETTE PARAMETERS
1572 |----- DW 00000H ; INT 1FH -- POINTER TO VIDEO EXTENSION
1573
1574 SLAVE_VECTOR_TABLE LABEL WORD ; ( INTERRUPT 70H THRU 7FH )
1575
1576 |----- DW OFFSET RTC_INT ; INT 70H - REAL TIME CLOCK IRQ 8
1577 |----- DW OFFSET RE_DIRECT ; INT 71H - REDIRECT TO INT 0AH IRQ 9
1578 |----- DW OFFSET D11 ; INT 72H - IRQ 10
1579 |----- DW OFFSET D11 ; INT 73H - IRQ 11
1580 |----- DW OFFSET D11 ; INT 74H - IRQ 12
1581 |----- DW OFFSET INT_287 ; INT 75H - -MATH COPROCESSOR IRQ 13
1582 |----- DW OFFSET D11 ; INT 76H - -FIXED DISK IRQ 14
1583 |----- DW OFFSET D11 ; INT 77H - IRQ 15
1584
1585 |----- DUMMY INTERRUPT HANDLER
1586
1587 |-- ORG 0FF53H
1588 |-- ORG 01F53H
1589
1590 DUMMY_RETURN EQU $ ; BIOS DUMMY (NULL) INTERRUPT RETURN
1591 IRET
1592 |----- 1F53 CF
1593
1594 |----- PRINT SCREEN
1595
1596 |-- ORG 0FF54H
1597 |-- ORG 01F54H
1598 PRINT_SCREEN EQU $
1599 |----- JMP PRINT_SCREEN_1 ; VECTOR ON TO MOVED BIOS CODE
1600 .LIST ; TUTOR
1601 |-----
1602 |-----
1603 |----- POWER ON RESET VECTOR
1604 |-----
1605 |-----
1606 |-- ORG 0FFF0H
1607 |-- ORG 01FF0H
1608
1609 |----- POWER ON RESET
1610
1611 P_O_R LABEL FAR ; POWER ON RESTART EXECUTION LOCATION
1612
1613 |----- DB 0EAH ; HARD CODE FAR JUMP TO SET
1614 |----- DW OFFSET RESET ; OFFSET
1615 |----- DW 0F000H ; SEGMENT
1616
1617 |----- DB '06/10/85' ; RELEASE MARKER
1618 |----- 38 35
1619
1620 |----- ORG 01FFE0H
1621 |----- DB 01FFE0H ; THIS PC'S ID ( MODEL BYTE )
1622
1623 |----- CODE ENDS END ; CHECKSUM AT LAST LOCATION
1624

```


SECTION 6. INSTRUCTION SET

Contents

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

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 \neq 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 \neq 11)
----------	----------------------------

JMP = Unconditional Jump

Short/Long

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

Direct within Segment

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

Register/Memory Indirect Within Segment

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

Direct Intersegment

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

Indirect Intersegment

11111111	mod101 r/m (mod \neq 11)
----------	----------------------------

RET = Return from Call

Within Segment

11000011

Within Segment Adding Immediate to SP

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

Intersegment

11001011

Intersegment Adding Immediate to SP

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

JE/JZ = Jump on Equal/Zero

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

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

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

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

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

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

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

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

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

JP/JPE = Jump on Parity/Parity Even

01111010	disp
----------	------

JO = Jump on Overflow

01110000	disp
----------	------

JS = Jump on Sign

01111000	disp
----------	------

JNE/JNZ = Jump on Not Equal/Not Zero

01110101	disp
----------	------

JNL/JGE = Jump on Not Less/Greater, or Equal

01111101	disp
----------	------

JNLE/JG = Jump on Not Less, or Equal/Greater

01111111	disp
----------	------

JNB/JAE = Jump on Not Below/Above, or Equal

01110011	disp
----------	------

JNBE/JA = Jump on Not Below, or Equal/Above

01110111	disp
----------	------

JNP/JPO = Jump on Not Parity/Parity Odd

01111011	disp
----------	------

JNO = Jump on Not Overflow

01110001	disp
----------	------

JNS = Jump on Not Sign

01111011	disp
----------	------

LOOP = Loop CX Times

11100010	disp
----------	------

LOOPZ/LOOPE = Loop while Zero/Equal

11100001	disp
----------	------

LOOPNZ/LOOPNE = Loop while Not Zero/Not Equal

11100000	disp
----------	------

JCXZ = Jump on CX Zero

11100011	disp
----------	------

ENTER = Enter Procedure

11001000	data-low	data-high
----------	----------	-----------

LEAVE = Leave Procedure

11001001

INT = Interrupt

Type Specified

11001101	Type
----------	------

Type 3

11001100

INTO = Interrupt on Overflow

11001110

IRET = Interrupt Return

11001111

BOUND = Detect Value Out of Range

01100010	mod reg r/m
----------	-------------

Processor Control

CLC = Clear Carry

11111000

CMC = Complement Carry

11110101

STC = Set Carry

11111001

CLD = Clear Direction

11111100

STD = Set Direction

11111101

CLI Clear Interrupt

11111010

STI = Set Interrupt

11111011

HLT = Halt

11110100

WAIT = Wait

10011011

LOCK = Bus Lock Prefix

11110000

CTS = Clear Task Switched Flag

00001111	00000110
----------	----------

ESC = Processor Extension Escape

11011TTT	modLLL r/m
----------	------------

Protection Control

LGDT = Load Global Descriptor Table Register

00001111	00000001	mod010 r/m
----------	----------	------------

SGDT = Store Global Descriptor Table Register

00001111	00000001	mod000 r/m
----------	----------	------------

LIDT = Load Interrupt Descriptor Table Register

00001111	00000001	mod011 r/m
----------	----------	------------

SIDT = Store Interrupt Descriptor Table Register

00001111	00000001	mod001 r/m
----------	----------	------------

LLDT = Load Local Descriptor Table Register from Register/Memory

00001111	00000000	mod010 r/m
----------	----------	------------

SLDT = Store Local Descriptor Table Register from Register/Memory

00001111	00000000	mod000 r/m
----------	----------	------------

LTR = Load Task Register from Register/Memory

00001111	00000000	mod011 r/m
----------	----------	------------

STR = Store Task Register to Register/Memory

00001111	00000000	mod001 r/m
----------	----------	------------

LMSW = Load Machine Status Word from Register/Memory

00001111	00000001	mod110 r/m
----------	----------	------------

SMSW = Store Machine Status Word

00001111	00000001	mod100 r/m
----------	----------	------------

LAR = Load Access Rights from Register/Memory

00001111	00000010	mod reg r/m
----------	----------	-------------

LSL = Load Segment Limit from Register/Memory

00001111	00000011	mod reg r/m
----------	----------	-------------

ARPL = Adjust Requested Privilege Level from Register/Memory

	01100011	mod reg r/m
--	----------	-------------

VERR = Verify Read Access; Register/Memory

00001111	00000000	mod100 r/m
----------	----------	------------

VERR = Verify Write Access

00001111	00000000	mod101 r/m
----------	----------	------------

The effective address (EA) of the memory operand is computed according to the mod and r/m fields:

If mod = 11, then r/m is treated as a reg field.

If mod = 00, then disp = 0, disp-low and disp-high are absent.

If mod = 01, then disp = disp-low sign-extended to 16 bits, disp-high is absent.

If mod = 10, then disp = disp-high:disp-low.

If r/m = 000, then $EA = (BX) + (SI) + DISP$

If r/m = 001, then $EA = (BX) + (SI) + DISP$

If r/m = 010, then $EA = (BP) + (SI) + DISP$

If r/m = 011, then $EA = (BP) + (DI) + DISP$

If r/m = 100, then $EA = (SI) + DISP$

If r/m = 101, then $EA = (DI) + DISP$

If r/m = 110, then $EA = (BP) + DISP$

If r/m = 111, then $EA = (BX) + DISP$

DISP follows the second byte of the instruction (before data if required).

Note: An exception to the above statements occurs when mod=00 and r/m=110, in which case EA = disp-high; disp-low.

Segment Override Prefix

001reg001

The 2-bit and 3-bit reg fields are defined as follows:

2-Bit reg Field

reg	Segment Register	reg	Segment Register
00	ES	10	SS
01	CS	11	DS

3-Bit reg Field

16-bit (w = 1)	8-bit (w = 0)
000 AX	000 AL
001 CX	001 CL
010 DX	010 DL
011 BX	011 BL
100 SP	100 AH
101 BP	101 CH
110 SI	110 DH
111 DI	111 BH

The physical addresses of all operands addressed by the BP register are computed using the SS segment register. The physical addresses of the destination operands of the string primitive operations (those addressed by the DI register) are computed using the ES segment, which may not be overridden.

80287 Coprocessor Instruction Set

The following is an instruction set summary for the 80287 coprocessor. In the following, the bit pattern for escape is 11011.

Data Transfer

FLD = Load

Integer/Real Memory to ST(0)

escape MF 1	mod 000 r/m
-------------	-------------

Long Integer Memory to ST(0)

escape 111	mod 101 r/m
------------	-------------

Temporary Real Memory to ST(0)

escape 011	mod 101 r/m
------------	-------------

BCD Memory to ST(0)

escape 111	mod 100 r/m
------------	-------------

ST(i) to ST(0)

escape 001	11000ST(i)
------------	------------

FST = Store

ST(0) to Integer/Real Memory

escape MF 1	mod 010 r/m
-------------	-------------

ST(0) to ST(i)

escape 101	11010 ST(i)
------------	-------------

FSTP = Store and Pop

ST(0) to Integer/Real Memory

escape MF 1	mod 011 r/m
-------------	-------------

ST(0) to Long Integer Memory

escape 111	mod 111 r/m
------------	-------------

ST(0) to Temporary Real Memory

escape 011	mod 111 r/m
------------	-------------

ST(0) to BCD Memory

escape 111	mod 110 r/m
------------	-------------

ST(0) to ST(i)

escape 101	11011 ST(i)
------------	-------------

FXCH = Exchange ST(i) and ST(0)

escape 001	11001 ST(i)
------------	-------------

Comparison

FCOM = Compare

Integer/Real Memory to ST(0)

escape MF 0	mod 010 r/m
-------------	-------------

ST(i) to ST(0)

escape 000	11010 ST(i)
------------	-------------

FCOMP = Compare and Pop

Integer/Real Memory to ST(0)

escape MF 0	mod 011 r/m
-------------	-------------

ST(i) to ST(0)

escape 000	11010 ST(i)
------------	-------------

FCOMPP = Compare ST(i) to ST(0) and Pop Twice

escape 110	11011001
------------	----------

FTST = Test ST(0)

escape 001	11100100
------------	----------

FXAM = Examine ST(0)

escape 001	11100101
------------	----------

Constants

FLDZ = Load + 0.0 into ST(0)

escape 000	11101110
------------	----------

FLD1 = Load + 1.0 into ST(0)

escape 001	11101000
------------	----------

FLDP1 = Load π into ST(0)

escape 001	11101011
------------	----------

FLDL2T = Load $\log_2 10$ into ST(0)

escape 001	11101001
------------	----------

FLDLG2 = Load $\log_{10} 2$ into ST(0)

escape 001	11101100
------------	----------

FLDLN2 = Load $\log_e 2$ into ST(0)

escape 001	11101101
------------	----------

Arithmetic

FADD = Addition

Integer/Real Memory with ST(0)

escape MF 0	mod 000 r/m
-------------	-------------

ST(i) and ST(0)

escape dP0	11000 ST(i)
------------	-------------

FSUB = Subtraction

Integer/Real Memory with ST(0)

escape MF 0	mod 10R r/m
-------------	-------------

ST(i) and ST(0)

escape dP0	1110R r/m
------------	-----------

FMUL = Multiplication

Integer/Real Memory with ST(0)

escape MF 0	mod 001 r/m
-------------	-------------

ST(i) and ST(0)

escape dP0	11001 r/m
------------	-----------

FDIV = Division

Integer/Real Memory with ST(0)

escape MF 0	mod 11R r/m
-------------	-------------

ST(i) and ST(0)

escape dP0	1111R r/m
------------	-----------

FSQRT = Square Root of ST(0)

escape 001	11111010
------------	----------

FSCALE = Scale ST(0) by ST(1)

escape 001	11111101
------------	----------

FPREM = Partial Remainder of ST(0) + ST(1)

escape 001	11111000
------------	----------

FRNDINT = Round ST(0) to Integer

escape 001	11111100
------------	----------

FXTRACT = Extract Components of ST(0)

escape 001	11110100
------------	----------

FABS = Absolute Value of ST(0)

escape 001	11100001
------------	----------

FCHS = Change Sign of ST(0)

escape 001	11100000
------------	----------

Transcendental

FPTAN = Partial Tangent of ST(0)

escape 001	11110010
------------	----------

FPATAN = Partial Arctangent of ST(0) ÷ ST(1)

escape 001	11110011
------------	----------

F2XM1 = 2^{ST(0)} - 1

escape 001	11110000
------------	----------

FYL2X = ST(1) x Log₂ [ST(0)]

escape 001	11110001
------------	----------

FYL2XP1 = ST(1) x Log₂ [ST(0) + 1]

escape 001	11111001
------------	----------

FINIT = Initialize NPX

escape 011	11100011
------------	----------

FSETPM = Enter Protected Mode

escape 011	11100100
------------	----------

FSTSWAX = Store Control Word

escape 111	11100000
------------	----------

FLDCW = Load Control Word

escape 001	mod 101 r/m
------------	-------------

FSTCW = Store Control Word

escape 001	mod 111 r/m
------------	-------------

FSTSW = Store Status Word

escape 101	mod 101 r/m
------------	-------------

FCLEX = Clear Exceptions

escape 011	11100010
------------	----------

FSTENV = Store Environment

escape 001	mod 110 r/m
------------	-------------

FLDENV = Load Environment

escape 001	mod 100 r/m
------------	-------------

FSAVE = Save State

escape 101	mod 110 r/m
------------	-------------

FRSTOR = Restore State

escape 101	mod 100 r/m
------------	-------------

FINCSTP = Increment Stack Pointer

escape 001	11110111
------------	----------

FDECSTP = Decrement Stack Pointer

escape 001	111100110
------------	-----------

FFREE = Free ST(i)

escape 101	11000ST(i)
------------	------------

FNOP = No Operation

escape 101	11010000
------------	----------

MF is assigned as follows:

MF Memory Format

00	32-bit Real
01	32-bit Integer
10	64-bit Real
11	16-bit Integer

The other abbreviations are as follows:

Term	Definition	Bit = 0	Bit ≠ 0
ST	Stack top	Stack top	(i)= ith register from the top
d	Destination	Dest. is ST(0)	Dest. is ST(i)
P	Pop	No pop	Pop
R	Reverse*	Dest. (op) source	Source (op) dest.
* When d=1, reverse the sense of R.			

Notes:



SECTION 7. CHARACTERS, KEYSTROKES, AND COLORS

Contents

Character Codes 7-3

Quick Reference 7-14

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	á	Alt 134	Note 6	Black	Brown	Normal
87	135	ç	Alt 135	Note 6	Black	Light Grey	Normal
88	136	ê	Alt 136	Note 6	Black	Dark Grey	Non-Display
89	137	ë	Alt 137	Note 6	Black	Light Blue	High Intensity Underline
8A	138	è	Alt 138	Note 6	Black	Light Green	High Intensity
8B	139	ï	Alt 139	Note 6	Black	Light Cyan	High Intensity
8C	140	î	Alt 140	Note 6	Black	Light Red	High Intensity
8D	141	ì	Alt 141	Note 6	Black	Light Magenta	High Intensity
8E	142	Ä	Alt 142	Note 6	Black	Yellow	High Intensity
8F	143	Å	Alt 143	Note 6	Black	White	High Intensity
90	144	É	Alt 144	Note 6	Blue	Black	Normal
91	145	æ	Alt 145	Note 6	Blue	Blue	Underline
92	146	Æ	Alt 146	Note 6	Blue	Green	Normal
93	147	ô	Alt 147	Note 6	Blue	Cyan	Normal
94	148	ö	Alt 148	Note 6	Blue	Red	Normal
95	149	ò	Alt 149	Note 6	Blue	Magenta	Normal
96	150	û	Alt 150	Note 6	Blue	Brown	Normal
97	151	ù	Alt 151	Note 6	Blue	Light Grey	Normal
98	152	ÿ	Alt 152	Note 6	Blue	Dark Grey	High Intensity
99	153	Ö	Alt 153	Note 6	Blue	Light Blue	High Intensity Underline
9A	154	Ü	Alt 154	Note 6	Blue	Light Green	High Intensity

Value		As Characters			As Text Attributes		
					Color/Graphics Monitor Adapter		IBM Monochrome Display Adapter
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground	
9B	155	¢	Alt 155	Note 6	Blue	Light Cyan	High Intensity
9C	156	£	Alt 156	Note 6	Blue	Light Red	High Intensity
9D	157	¥	Alt 157	Note 6	Blue	Light Magenta	High Intensity
9E	158	Pt	Alt 158	Note 6	Blue	Yellow	High Intensity
9F	159	<i>f</i>	Alt 159	Note 6	Blue	White	High Intensity
A0	160	á	Alt 160	Note 6	Green	Black	Normal
A1	161	í	Alt 161	Note 6	Green	Blue	Underline
A2	162	ó	Alt 162	Note 6	Green	Green	Normal
A3	163	ú	Alt 163	Note 6	Green	Cyan	Normal
A4	164	ñ	Alt 164	Note 6	Green	Red	Normal
A5	165	Ñ	Alt 165	Note 6	Green	Magenta	Normal
A6	166	<u>a</u>	Alt 166	Note 6	Green	Brown	Normal
A7	167	<u>o</u>	Alt 167	Note 6	Green	Light Grey	Normal
A8	168	¿	Alt 168	Note 6	Green	Dark Grey	High Intensity
A9	169	┌	Alt 169	Note 6	Green	Light Blue	High Intensity Underline
AA	170	└	Alt 170	Note 6	Green	Light Green	High Intensity
AB	171	½	Alt 171	Note 6	Green	Light Cyan	High Intensity
AC	172	¼	Alt 172	Note 6	Green	Light Red	High Intensity
AD	173	i	Alt 173	Note 6	Green	Light Magenta	High Intensity
AE	174	<<	Alt 174	Note 6	Green	Yellow	High Intensity
AF	175	>>	Alt 175	Note 6	Green	White	High Intensity
B0	176	⋮	Alt 176	Note 6	Cyan	Black	Normal
B1	177	⋮	Alt 177	Note 6	Cyan	Blue	Underline
B2	178	⋮	Alt 178	Note 6	Cyan	Green	Normal
B3	179	▬	Alt 179	Note 6	Cyan	Cyan	Normal
B4	180	▬	Alt 180	Note 6	Cyan	Red	Normal
B5	181	▬	Alt 181	Note 6	Cyan	Magenta	Normal
B6	182	▬	Alt 182	Note 6	Cyan	Brown	Normal

Value		As Characters			As Text Attributes		
					Color/Graphics Monitor Adapter		IBM Monochrome Display Adapter
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground	
B7	183		Alt 183	Note 6	Cyan	Light Grey	Normal
B8	184		Alt 184	Note 6	Cyan	Dark Grey	High Intensity
B9	185		Alt 185	Note 6	Cyan	Light Blue	High Intensity Underline
BA	186		Alt 186	Note 6	Cyan	Light Green	High Intensity
BB	187		Alt 187	Note 6	Cyan	Light Cyan	High Intensity
BC	188		Alt 188	Note 6	Cyan	Light Red	High Intensity
BD	189		Alt 189	Note 6	Cyan	Light Magenta	High Intensity
BE	190		Alt 190	Note 6	Cyan	Yellow	High Intensity
BF	191		Alt 191	Note 6	Cyan	White	High Intensity
C0	192		Alt 192	Note 6	Red	Black	Normal
C1	193		Alt 193	Note 6	Red	Blue	Underline
C2	194		Alt 194	Note 6	Red	Green	Normal
C3	195		Alt 195	Note 6	Red	Cyan	Normal
C4	196		Alt 196	Note 6	Red	Red	Normal
C5	197		Alt 197	Note 6	Red	Magenta	Normal
C6	198		Alt 198	Note 6	Red	Brown	Normal
C7	199		Alt 199	Note 6	Red	Light Grey	Normal
C8	200		Alt 200	Note 6	Red	Dark Grey	High Intensity
C9	201		Alt 201	Note 6	Red	Light Blue	High Intensity Underline
CA	202		Alt 202	Note 6	Red	Light Green	High Intensity
CB	203		Alt 203	Note 6	Red	Light Cyan	High Intensity
CC	204		Alt 204	Note 6	Red	Light Red	High Intensity
CD	205		Alt 205	Note 6	Red	Light Magenta	High Intensity
CE	206		Alt 206	Note 6	Red	Yellow	High Intensity
CF	207		Alt 207	Note 6	Red	White	High Intensity
D0	208		Alt 208	Note 6	Magenta	Black	Normal

Value		As Characters			As Text Attributes		
					Color/Graphics Monitor Adapter		IBM Monochrome Display Adapter
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground	
D1	209		Alt 209	Note 6	Magenta	Blue	Underline
D2	210		Alt 210	Note 6	Magenta	Green	Normal
D3	211		Alt 211	Note 6	Magenta	Cyan	Normal
D4	212		Alt 212	Note 6	Magenta	Red	Normal
D5	213		Alt 213	Note 6	Magenta	Magenta	Normal
D6	214		Alt 214	Note 6	Magenta	Brown	Normal
D7	215		Alt 215	Note 6	Magenta	Light Grey	Normal
D8	216		Alt 216	Note 6	Magenta	Dark Grey	High Intensity
D9	217		Alt 217	Note 6	Magenta	Light Blue	High Intensity Underline
DA	218		Alt 218	Note 6	Magenta	Light Green	High Intensity
DB	219		Alt 219	Note 6	Magenta	Light Cyan	High Intensity
DC	220		Alt 220	Note 6	Magenta	Light Red	High Intensity
DD	221		Alt 221	Note 6	Magenta	Light Magenta	High Intensity
DE	222		Alt 222	Note 6	Magenta	Yellow	High Intensity
DF	223		Alt 223	Note 6	Magenta	White	High Intensity
E0	224	α	Alt 224	Note 6	Brown	Black	Normal
E1	225	β	Alt 225	Note 6	Brown	Blue	Underline
E2	226	Γ	Alt 226	Note 6	Brown	Green	Normal
E3	227	π	Alt 227	Note 6	Brown	Cyan	Normal
E4	228	Σ	Alt 228	Note 6	Brown	Red	Normal
E5	229	σ	Alt 229	Note 6	Brown	Magenta	Normal
E6	230	μ	Alt 230	Note 6	Brown	Brown	Normal
E7	231	τ	Alt 231	Note 6	Brown	Light Grey	Normal
E8	232	Φ	Alt 232	Note 6	Brown	Dark Grey	High Intensity
E9	233	θ	Alt 233	Note 6	Brown	Light Blue	High Intensity Underline
EA	234	Ω	Alt 234	Note 6	Brown	Light Green	High Intensity
EB	235	δ	Alt 235	Note 6	Brown	Light Cyan	High Intensity

Value		As Characters			As Text Attributes		
					Color/Graphics Monitor Adapter		IBM Monochrome Display Adapter
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground	
EC	236	∞	Alt 236	Note 6	Brown	Light Red	High Intensity
ED	237	ϕ	Alt 237	Note 6	Brown	Light Magenta	High Intensity
EE	238	€	Alt 238	Note 6	Brown	Yellow	High Intensity
EF	239	∩	Alt 239	Note 6	Brown	White	High Intensity
F0	240	≡	Alt 240	Note 6	Light Grey	Black	Reverse Video
F1	241	±	Alt 241	Note 6	Light Grey	Blue	Underline
F2	242	≥	Alt 242	Note 6	Light Grey	Green	Normal
F3	243	≤	Alt 243	Note 6	Light Grey	Cyan	Normal
F4	244	∫	Alt 244	Note 6	Light Grey	Red	Normal
F5	245	∫	Alt 245	Note 6	Light Grey	Magenta	Normal
F6	246	÷	Alt 246	Note 6	Light Grey	Brown	Normal
F7	247	≈	Alt 247	Note 6	Light Grey	Light Grey	Normal
F8	248	○	Alt 248	Note 6	Light Grey	Dark Grey	Reverse Video
F9	249	●	Alt 249	Note 6	Light Grey	Light Blue	High Intensity Underline
FA	250	●	Alt 250	Note 6	Light Grey	Light Green	High Intensity
FB	251	√	Alt 251	Note 6	Light Grey	Light Cyan	High Intensity
FC	252	ⁿ	Alt 252	Note 6	Light Grey	Light Red	High Intensity
FD	253	²	Alt 253	Note 6	Light Grey	Light Magenta	High Intensity
FE	254	■	Alt 254	Note 6	Light Grey	Yellow	High Intensity
FF	255	BLANK	Alt 255	Note 6	Light Grey	White	High Intensity

7-12 Characters, Keystrokes, and Colors

Notes

1. Asterisk (*) can be typed using two methods: press the (*) key or, in the shift mode, press the 8 key.
2. Period (.) can be typed using two methods: press the . key or, in the shift or Num Lock mode, press the Del key.
3. Numeric characters 0-9 can be typed using two methods: press the numeric keys on the top row of the keyboard or, in the shift or Num Lock mode, press the numeric keys in the keypad portion of the keyboard.
4. Uppercase alphabetic characters (A-Z) can be typed in two modes: the shift mode or the Caps Lock mode.
5. Lowercase alphabetic characters (a-z) can be typed in two modes: in the normal mode or in Caps Lock and shift mode combined.
6. The three digits after the Alt key must be typed from the numeric keypad. Character codes 1-255 may be entered in this fashion (with Caps Lock activated, character codes 97-122 will display uppercase).

Quick Reference

DECIMAL VALUE	➡	0	16	32	48	64	80	96	112
⬇	HEXA-DECIMAL VALUE	0	1	2	3	4	5	6	7
0	0	BLANK (NULL)	▶	BLANK (SPACE)	0	@	P	'	p
1	1	😊	◀	!	1	A	Q	a	q
2	2	😄	↕		2	B	R	b	r
3	3	♥	!!	#	3	C	S	c	s
4	4	♦	¶	\$	4	D	T	d	t
5	5	♣	§	%	5	E	U	e	u
6	6	♠	▬	&	6	F	V	f	v
7	7	•	↕	'	7	G	W	g	w
8	8	•	↑	(8	H	X	h	x
9	9	○	↓)	9	I	Y	i	y
10	A	○	→	*	:	J	Z	j	z
11	B	♂	←	+	;	K	[k	{
12	C	♀	└	,	<	L	\	l	
13	D	🎵	↔	—	=	M]	m	}
14	E	🎵	▲	.	>	N	^	n	~
15	F	☀	▼	/	?	O	_	o	△



DECIMAL VALUE	➡	128	144	160	176	192	208	224	240
⬇	HEXA-DECIMAL VALUE	8	9	A	B	C	D	E	F
0	0	Ç	É	á	⋮			∞	≡
1	1	ü	æ	í	⋮			β	±
2	2	é	Æ	ó	⋮			Γ	≥
3	3	â	ô	ú				π	≤
4	4	ä	ö	ñ				Σ	∫
5	5	à	ò	Ñ				σ	∫
6	6	â	û	à				μ	÷
7	7	ç	ù	ó				γ	≈
8	8	ê	ÿ	¿				Φ	°
9	9	ë	Ö	┘				Θ	•
10	A	è	Ü	┘				Ω	•
11	B	ï	¢	½				δ	√
12	C	î	£	¼				∞	n
13	D	ì	¥	ì				φ	²
14	E	Ä	Œ	«				€	■
15	F	Å	f	»				∩	BLANK 'FF'

Notes:



SECTION 8. COMMUNICATIONS

Contents

Hardware 8-3

Establishing a Communications Link 8-5

SECTION 8

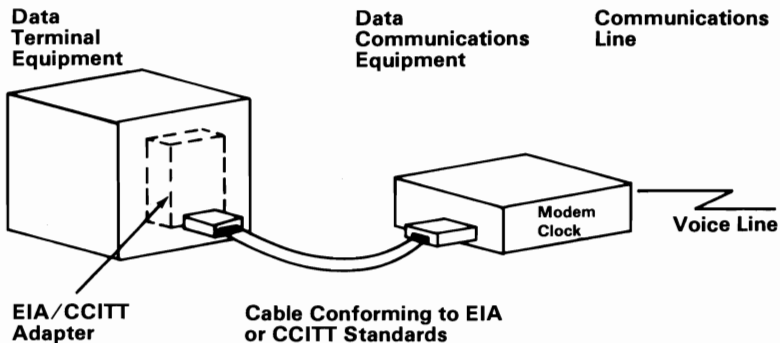
Notes:



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:



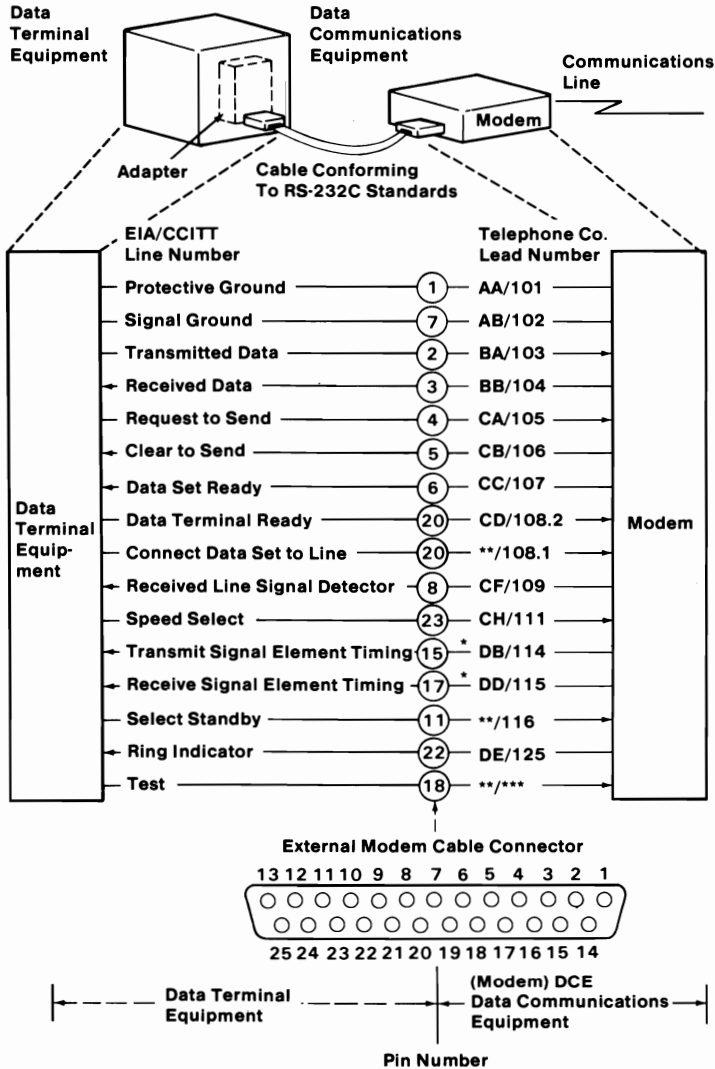
The EIA/CCITT adapter allows the data terminal equipment to be connected to the data communications equipment using EIA or CCITT standardized connections. An external modem is shown in the figure; however, other types of data communications equipment also can be connected to the data terminal equipment 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.

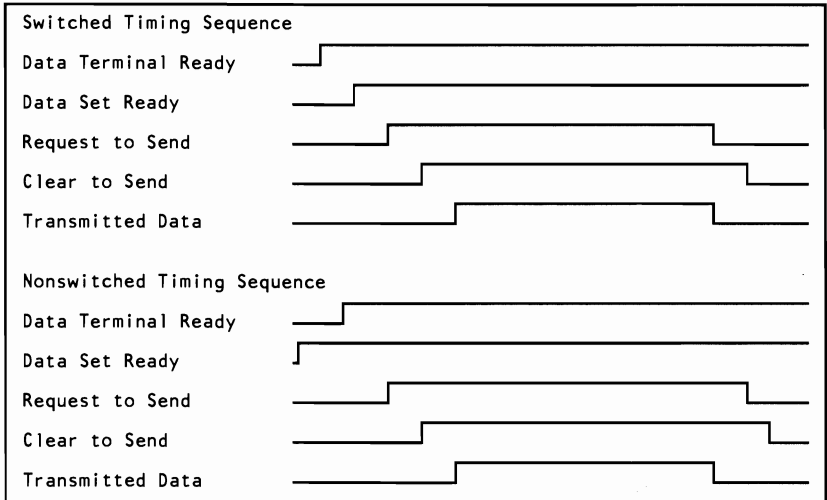
The following is an illustration of data terminal equipment connected to an external modem using connections defined by the RS-232C interface standard:



*Not used when business machine clocking is used.
 **Not standardized by EIA (Electronics Industry Association).
 ***Not standardized by CCITT

Establishing a Communications Link

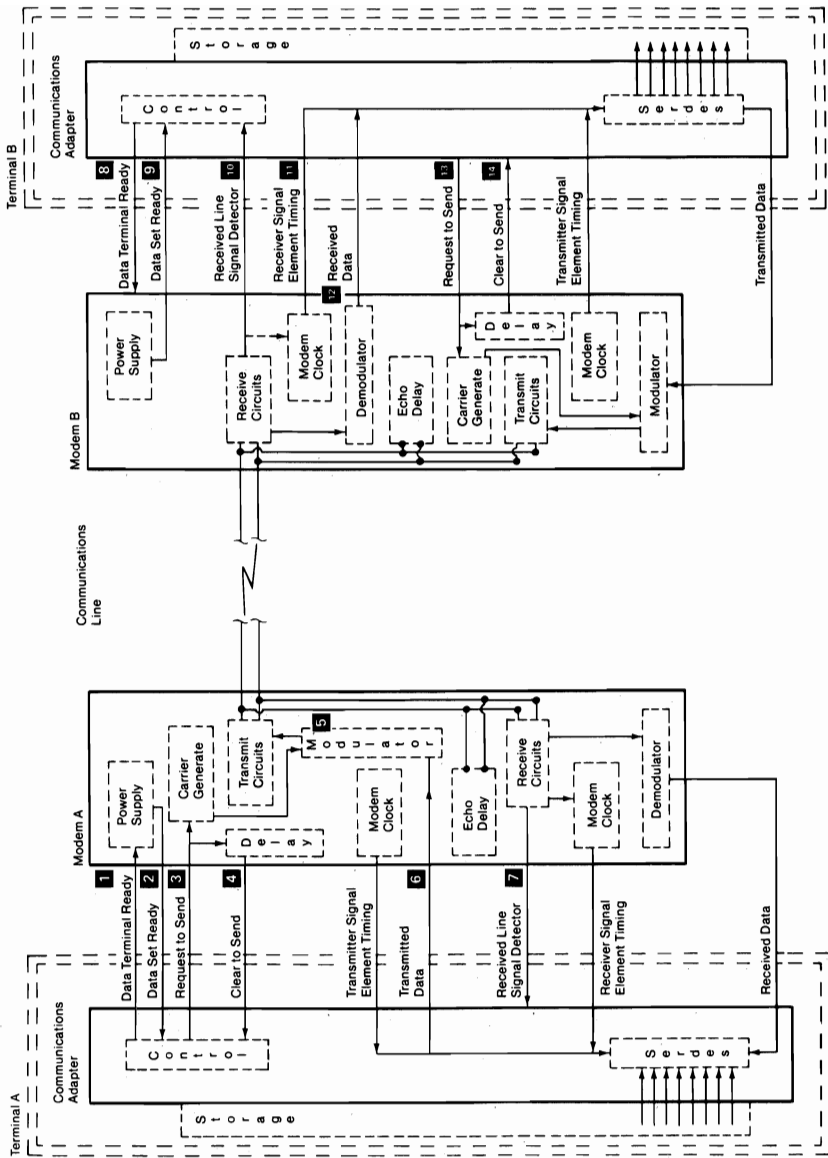
The following bar graphs represent normal timing sequences of operation during the establishment of communication for both switched (dial-up) and nonswitched (direct line) networks.



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

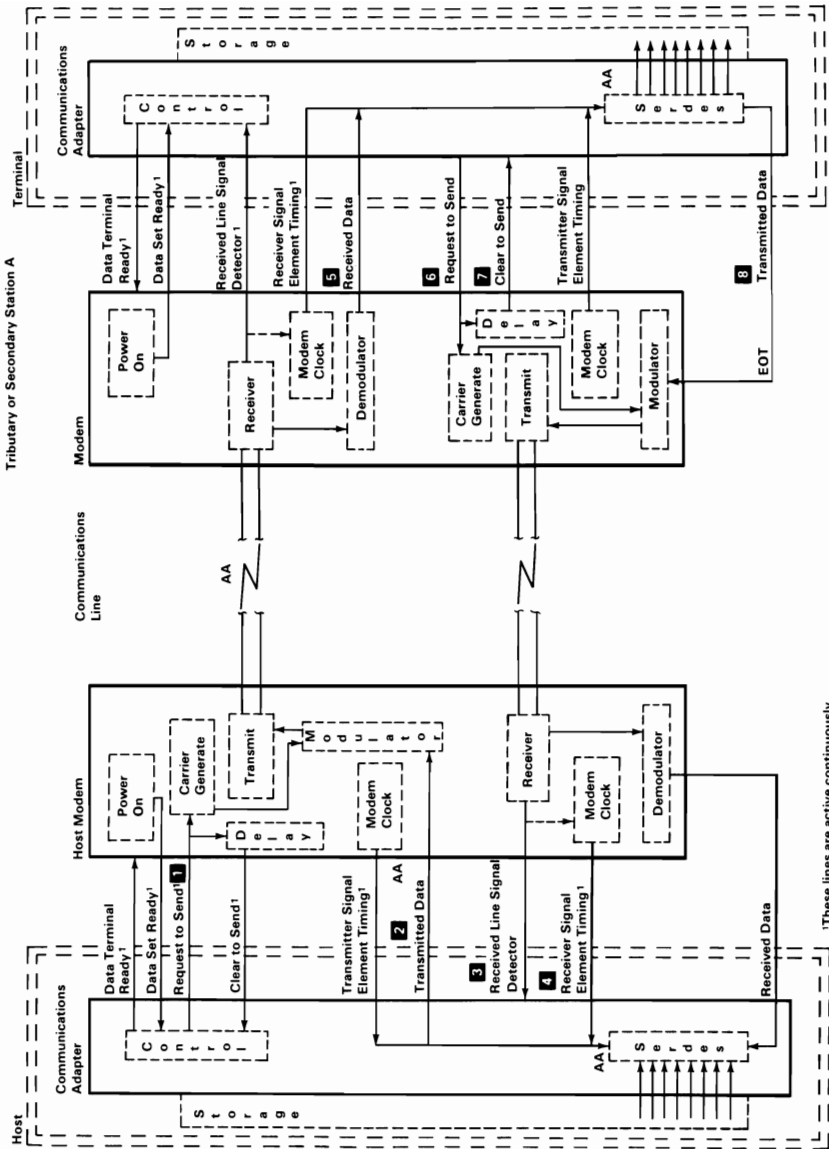
- The terminals at both locations activate the 'data terminal ready' lines **1** and **8**.
- Normally the 'data set ready' lines **2** and **9** from the modems are active whenever the modems are powered on.
- Terminal A activates the 'request to send' line **3**, which causes the modem at terminal 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).



SECTION 8

Establishing a Link on a Nonswitched Multipoint Line

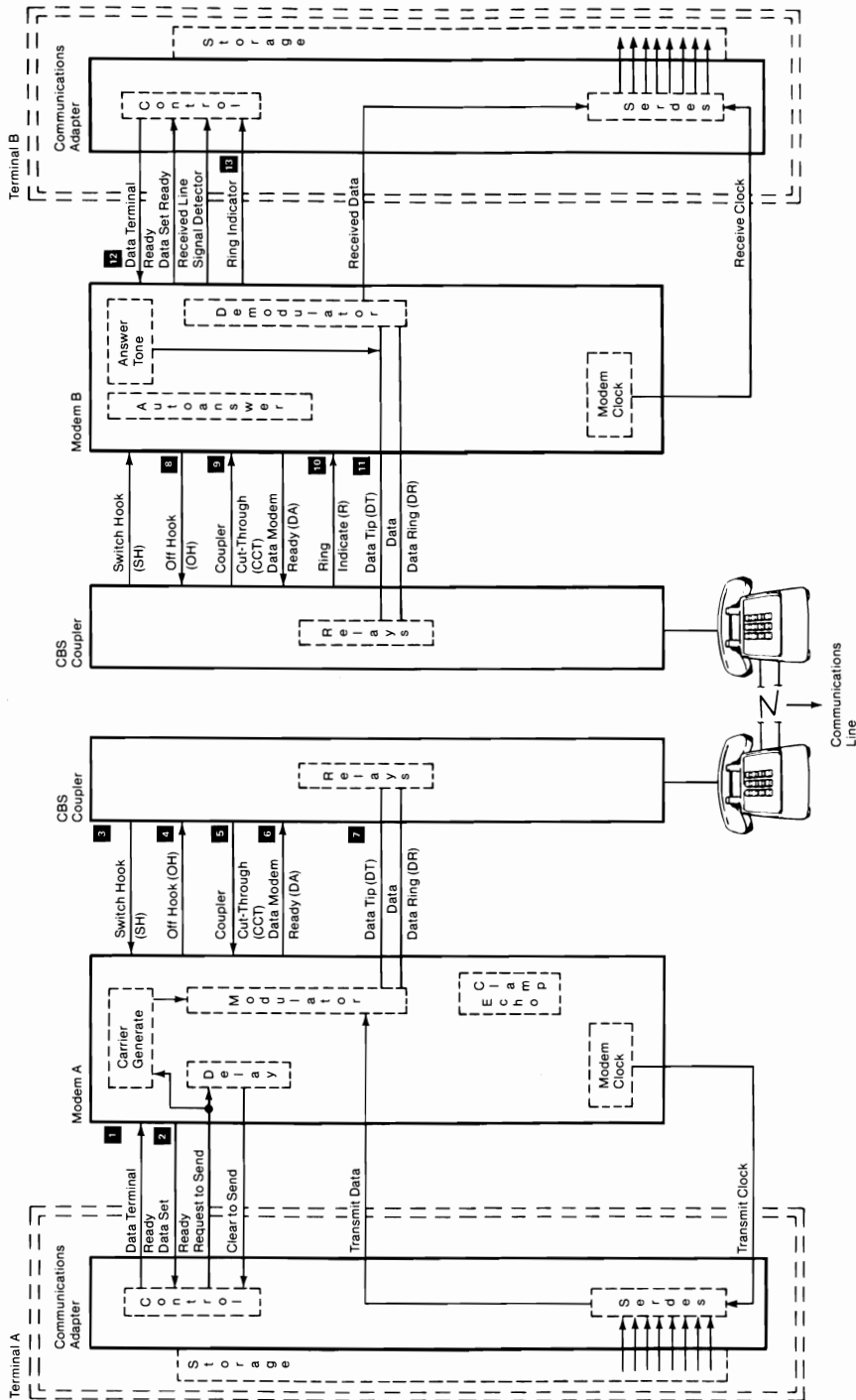
1. The control station serializes the address for the tributary or secondary station (AA) and sends its address to the modem on the 'transmitted data' line **2**.
2. Since the 'request to send' line and, therefore, the modem carrier, is active continuously **1**, the modem immediately modulates the carrier 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.



These lines are active continuously.

Establishing a Link on a Switched Point-to-Point Line

- 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.
- When the terminal A operator lifts the telephone handset, the 'switch hook' line from the coupler is activated **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.
- Modem A activates the 'data modem ready' line **6** to the coupler (the 'data modem ready' line is on continuously in some modems).
- 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.
- 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.
- 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.)
- The autoanswer circuits in modem B activate the 'off hook' line to the coupler **8**.
- 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.
- 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**.
- 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

Hardware Considerations	9-3
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
Application Guidelines	9-7
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
E8 FB FF   CALL $-2       ; CALL within segment
                ; program will continue here
```

- **PUSH SP**

80286 microprocessor pushes the current stack pointer.

8088 microprocessor pushes the new stack pointer.

- Single step interrupt (when $TF=1$) on the interrupt instruction (OP code hex CC,CD):

80286 microprocessor does **not** interrupt on the INT instruction.

8088 microprocessor does interrupt on the INT instruction.

- The divide error exception (interrupt 0):

80286 microprocessor pushes the CS:IP of the instruction, causing the exception.

8088 microprocessor pushes the CS:IP **following** the instruction, causing the exception.

- Shift counts are masked to five bits. Shift counts greater than 31 are treated mod 32. For example, a shift count of 36, shifts the operand four places.

The following describes anomalies which may occur in systems which contain 80286 processors with 1983 and 1984 date codes (S40172, S54036, S40093, S54012).

In protected mode, the contents of the CX register may be unexpectedly altered under the following conditions:

Note: The value in parenthesis indicates the type of error code pushed onto the exception handler's stack.

Exception #NP() = Exception #11 = Not-present Fault

Exception #SS() = Exception #12 = Stack Fault

Exception #GP() = Exception #13 = General Protection Fault

- Exception #GP(0) from attempted access to data segment or extra segment when the corresponding segment register holds a null selector.
- Exception #GP(0) from attempted data read from code segment when code segment has the "execute only" attribute.
- Exception #GP(0) from attempted write to code segment (code segments are not writable in protected mode), or to data segment of extra segment if the data or extra segment has the read only attribute.

- Exception #GP(0) from attempted load of a selector referencing the local descriptor table into CS, DS, ES or SS, when the LDT is not present.
- Exception #GP(0) from attempted input or output instruction when $CPL > IOPL$.
- Exception #GP(selector) from attempted access to a descriptor is GDT, LDT, or IDT, beyond the defined limit of the descriptor table.
- Exception #GP(0) from attempted read or write (except for "PUSH" onto stack) beyond the defined limit of segment.
- Exception #SS(0) from attempted "PUSH" below the defined limit of the stack segment.

Restarting applications which generate the above exceptions may result in errors.

In the protected mode, when any of the null selector values (0000H, 0001H, 0002H, 0003H) are loaded into the DS or ES registers via a MOV or POP instruction or a task switch, the 80286 always loads the null selector 0000H into the corresponding register.

If a coprocessor (80287) operand is read from an "executable and readable" and conforming (ERC) code segment, and the coprocessor operand is sufficiently near the segment's limit that the second or subsequent byte lies outside the limit, no protection exception #9 will be generated.

The following correctly describes the operation of all 80286 parts:

- Instructions longer than 10 bytes (instructions using multiple redundant prefixes) generate exception #13 (General Purpose Exception) in both the real and protected modes.
- If the second operand of an ARPL instruction is a null selector, the instruction generates an exception #13.

Assembler language programs should perform all I/O operations through ROM BIOS or DOS function calls.

- Program interrupts are used for access to these functions. This practice removes the absolute addressing from the program. Only the interrupt number is required.
- The coprocessor detects six different exception conditions that can occur during instruction execution. If the appropriate exception mask within the coprocessor is not set, the coprocessor sets its error signal. This error signal generates a hardware interrupt (interrupt 13) and causes the 'busy' signal to the coprocessor to be held in the busy state. The 'busy' signal may be cleared by an 8-bit I/O Write command to address hex F0 with D0 through D7 equal to 0.

The power-on-self-test code in the system ROM enables hardware IRQ 13 and sets up its vector to point to a routine in ROM. The ROM routine clears the 'busy' signal latch and then transfers control to the address pointed to by the NMI interrupt vector. This allows code written for any IBM Personal Computer to work on an IBM Personal Computer 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 B000 and B800)-- 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

--	(625 ms-read)	--
0FEH	yes (18 second)	printer

The asynchronous support has been omitted. The Serial/Parallel Adapter will generate interrupts, but BIOS does not support it in the interrupt mode. Therefore, the support should be included in the multitasking system code if that device is to be supported.

Time-Outs

To support time-outs properly, the multitasking dispatcher must be aware of time. If a device enters a busy loop, it generally should remain there for a specific amount of time before indicating an error. The dispatcher should return to the BIOS wait loop with the carry bit set if a time-out occurs.

Machine-Sensitive Code

Programs may select machine specific features, but they must test for specific machine type. Location of the specific machine identification codes can be found through interrupt 15 function code AH (See 'Configuration Parameters' in BIOS Listing). The code is two bytes. The first byte shows the machine type and the second byte shows the series type. They are as follows:

First Byte	Second Byte	Machine Identification
FF	00	IBM Personal Computer
FE	00	IBM Personal Computer XT
FE	00	IBM Portable Personal Computer
FD	00	IBM PCjr
FC	00	IBM Personal Computer AT

Machine Identification Code

IBM will define methods for uniquely determining the specific machine type or I/O feature for any new device.

Notes:



Glossary

This glossary includes definitions developed by the American National Standards Institute (ANSI) and the International Organization for Standardization (ISO). This material is reproduced from the *American National Dictionary for Information Processing*, copyright 1977 by the Computer and Business Equipment Manufacturers Association, copies of which may be purchased from the American National Standards Institute, 1430 Broadway, New York, New York 10018.

μ . Prefix micro; 0.000 001.

μ 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.

alphanumeric. 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 alphanumeric.

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. Milliamperere; 0.001 ampere.

machine code. The machine language used for entering text and program instructions onto the recording medium or into storage and which is subsequently used for processing and printout.

machine language. (1) A language that is used directly by a machine. (2) Deprecated term for computer instruction code.

magnetic disk. (1) A flat circular plate with a magnetizable surface layer on which data can be stored by magnetic recording. (2) See also diskette.

main storage. (1) Program-addressable storage from which instructions and other data can be loaded directly into registers for subsequent execution or processing. (2) Contrast with auxiliary storage.

mark. A symbol or symbols that indicate the beginning or the end of a field, of a word, of an item of data, or of a set of data such as a file, a record, or a block.

mask. (1) A pattern of characters that is used to control the retention or elimination of portions of another pattern of characters. (2) To use a pattern of characters to control the retention or elimination of portions of another pattern of characters.

masked. Synonym for disabled.

matrix. (1) A rectangular array of elements, arranged in rows and columns, that may be manipulated according to the rules of

matrix algebra. (2) In computers, a logic network in the form of an array of input leads and output leads with logic elements connected at some of their intersections.

matrix printer. A printer in which each character is represented by a pattern of dots; for example, a stylus printer, a wire printer. Synonymous with dot printer.

Mb. 1,048,576 bytes.

mega (M). Prefix 1,000,000.

megahertz (MHz). 1,000,000 hertz.

memory. Term for main storage.

meter (m). A unit of length (equivalent to 39.37 inches).

MFM. Modified frequency modulation.

MHz. Megahertz; 1,000,000 hertz.

micro (μ). Prefix 0.000,001.

microcode. (1) One or more microinstructions. (2) A code, representing the instructions of an instruction set, implemented in a part of storage that is not program-addressable.

microinstruction. (1) An instruction of microcode. (2) A basic or elementary machine instruction.

microprocessor. An integrated circuit that accepts coded instructions for execution; the instructions may be entered, integrated, or stored internally.

microsecond (μ 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
- STSW 6-27
- STSWAX 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

Numerals

XCHG 6-4
XLAT 6-5
XOR 6-11

OWS 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:





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



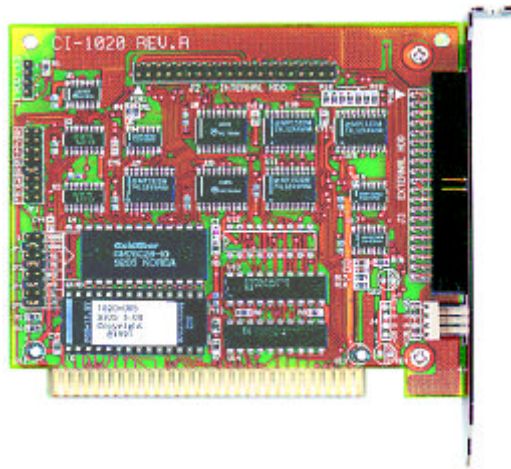
POSTAGE WILL BE PAID BY ADDRESSEE

IBM PERSONAL COMPUTER
READER COMMENT DEPARTMENT
P.O. BOX 1328-C
BOCA RATON, FLORIDA 33429-9960



Fold here

MCT-IDE-8 8-Bit IDE Controller Card



Add Two IDE Drives to your 8-bit system! Support for Internal or External Drives!

With this controller card, you can now use popular IDE disk drives on your older computers. Install a faster, larger capacity, more reliable and less expensive IDE hard drive on your 8088 PC-compatible system or on any system with an available 8-bit ISA slot. Although IDE hard drives were designed for 16-bit systems, with the MCT-IDE-8 controller, you can now use up to two IDE drives on your 8-bit-only system. This card is also compatible with any 16-bit ISA bus, so you can install it in a 286, 386, 486, or 586 based system. This controller card supports drives with capacities up to 528MB. It has external connectors for mini-power and IDE cables, so your drives can be internal or external. The controller can coexist with another hard drive controller, so you can also keep your present hard drive.

The MCT-IDE-8's onboard BIOS detects the presence or absence of another hard drive controller. If another controller exists, the MCT-IDE-8 becomes your secondary controller and the attached IDE drive becomes the second drive (e.g., D:). By jumper selection, you can choose the ROM BIOS and I/O base addresses to avoid conflicts. If you are installing this controller into an 8088 system, the onboard BIOS detects the parameters of attached IDE drives. If you are installing it into a 286 or later system, you will need to tell your CMOS/BIOS Setup Utility the IDE drive types.

Features

- Supports up to two IDE drives
- On-board BIOS provides IDE support
- Coexists with another hard drive controller without special software support
- When installed in an 8088 system, the BIOS determines the parameters of attached drives; when installed in a 286 or later system, the drive types need to be specified in the CMOS Setup (or Setup Disk)

- Can access up to 528MB without partitioning (DOS 5.0 or later required)
- Can function as the primary controller or co-exist with another controller (including controllers built into the motherboard)

What You Get

- MCT-IDE- card
- One IDE cable
- Owner's Reference Guide

What You Need

- One free 8-bit ISA slot in an 8088 system or an 8/16-bit ISA slot in a 80286 compatible system (286, 386, 486, or 586)
- DOS 5.0 or above recommended to access large drives without partitioning

Specifications

- Bus Interface: ISA 8-bit
- I/O Bus address (140h or 170h, the default is 170h) is jumper selected
- ROM BIOS address (C000-DE00h, the default is C800h) is jumper selected
- One 40-pin header for internal IDE attachment
- One 40-pin IDE connector for external IDE attachment
- One 4-pin mini-power connector for powering external drive

Size/Weight

- 4.5"L x 4"H
- Package weight 1 lb

Warranty

This product is backed by JDR's own two year limited warranty. Call for details!

**ORDER TOLL-FREE
TODAY!**

1-800-538-5000

. . . and Ask About Other JDR Products!

Via Internet: <http://www.jdr.com>

Via Fax: 1-800-538-5005

If you have any questions:

Tech Support: 1-800-538-5002