



*Personal Computer
Hardware Reference
Library*

Technical Reference

First Edition (August 1986)

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

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

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

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

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

**FEDERAL COMMUNICATIONS COMMISSION RADIO
FREQUENCY INTERFERENCE STATEMENT**

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

CAUTION

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

Notes:

○

○

○

Preface

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

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

This manual consists of eight sections:

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

A glossary, bibliography, and index are included.

Prerequisite Publications

Guide to Operations for the IBM Personal Computer XT Model 286

Suggested Reading

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

Additional Information

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

Contents

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

Output Voltage Sequencing	3-4
No-Load Operation	3-4
Power-Good Signal	3-4
Connectors	3-6
SECTION 4. KEYBOARD	4-1
Description	4-3
Power-On Routine	4-5
Commands from the System	4-6
Commands to the System	4-13
Keyboard Scan Codes	4-15
Clock and Data Signals	4-27
Keyboard Encoding and Usage	4-30
Keyboard Layouts	4-40
Specifications	4-47
Logic Diagram	4-48
SECTION 5. SYSTEM BIOS	5-1
System BIOS Usage	5-3
Quick Reference	5-14
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. IBM PERSONAL COMPUTER

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

Notes:

INDEX TAB LISTING

Section 1: System Board

SECTION 1

Section 2: Coprocessor

SECTION 2

Section 3: Power Supply

SECTION 3

Section 4: Keyboard

SECTION 4

Section 5: System BIOS

SECTION 5

Section 6: Instruction Set

SECTION 6

Notes:

—

—

—

Section 7: Characters, Keystrokes, and Colors

SECTION 7

Section 8: Compatibility

SECTION 8

Glossary

GLOSSARY

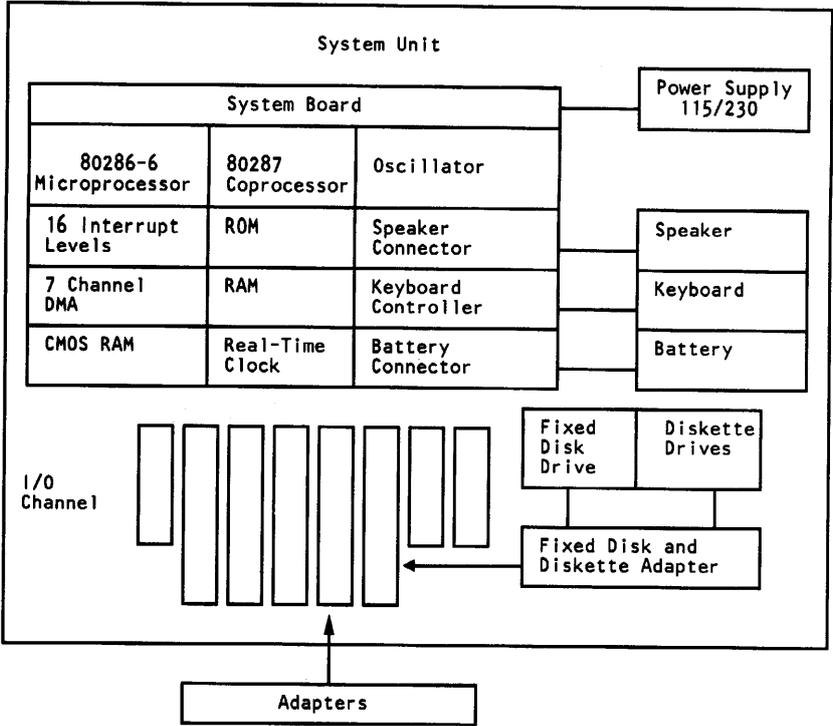
Bibliography

BIBLIOGRAPHY

Index

INDEX

System Block Diagram



SECTION 1. SYSTEM BOARD

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

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

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

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

Memory

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

Microprocessor

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

Real Address Mode

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

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

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

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

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

Protected (Virtual Address) Mode

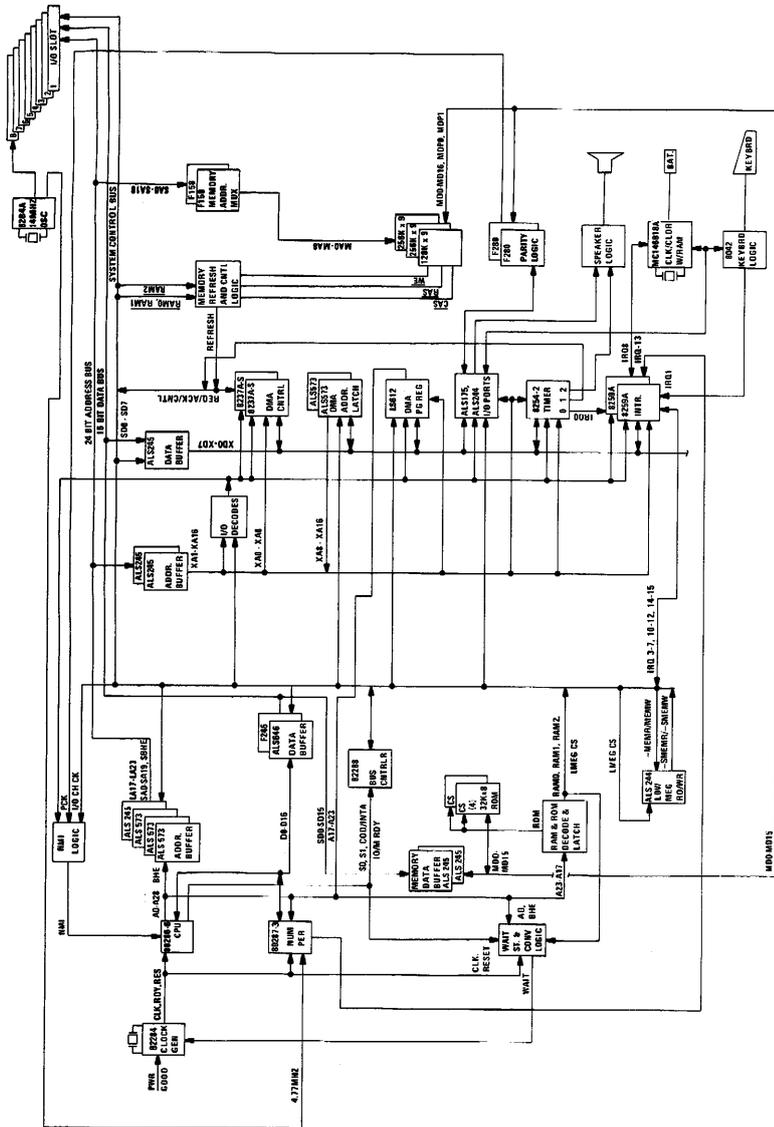
The protected mode offers extended physical and virtual memory address space, memory protection mechanisms, and new operations to support operating systems and virtual memory.

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

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

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

The following is a block diagram of the system board.



1-6 System Board

System Performance

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

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

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

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

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

System Memory Mapping

The following shows the mapping of the system memory.

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

System Memory

Direct Memory Access

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

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

DMA Channels

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

The following figures show address generation for the DMA channels.

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

Address Generation for DMA Channels 0 through 3

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

DMA controller 1 command code addresses follow.

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

DMA Controller 1 (Channels 0-3)

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

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

Address Generation for DMA Channels 5 through 7

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

The following figure shows the addresses for the page register.

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

Page Register Addresses

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

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

DMA controller 2 command code addresses follow.

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

DMA Controller 2 (DMA Channels 4-7)

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

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

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

System Interrupts

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

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

Hardware Interrupt Listing

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

Level	Function
Microprocessor NMI	Parity or I/O Channel Check
Interrupt Controllers CTRL 1 CTRL 2	
IRQ 0 IRQ 1 IRQ 2	Timer Output 0 Keyboard (Output Buffer Full) Interrupt from CTRL 2
<div style="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.) PC Network * PC Network (Alt.) * SDLC
IRQ 4	Serial Port 1 BSC BSC (Alt.) SDLC
IRQ 5 IRQ 6	Parallel Port 2 Diskette Controller Fixed Disk and Diskette Drive
IRQ 7	Parallel Port 1 Data Acquisition and Control ** GPIB *** Voice Communications Adapter ****
* The PC Network is jumper selectable. ** The Data Acquisition Adapter can be set to interrupts 3 through 7. The default interrupt is 7. *** The GPiB Adapter can be set to interrupts 2 through 7. **** The Voice Communications Adapter can be set to interrupts 2, 3, 4, or 7 (Interrupt level 7 is recommended).	

Hardware Interrupt Listing

Interrupt Sharing

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

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

Design Overview

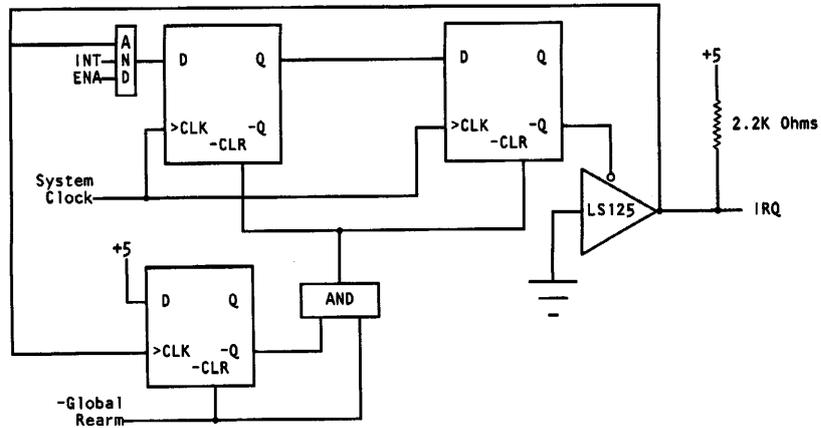
Most interrupt-supporting adapters hold the 'interrupt request' line (IRQ) at a low level and then drive the line high to cause an interrupt. In contrast, the shared-interrupt hardware design allows IRQ to float high through pull-up resistors on each adapter. Each adapter on the line may cause an interrupt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

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

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

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

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



Shared Interrupt Logic Diagram

Program Support

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

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

Linking to a Chain

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

Sharing the Interrupt Level

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

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

Unlinking from the Chain

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

Error Recovery

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

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

ROS Considerations

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

Precautions

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

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

Interrupt Chaining Structure

```
ENTRY: JMP      SHORT PAST      ; Jump around structure
        FPTR     DD      0       ; Forward Pointer
        SIGNATURE DW     424BH    ; Used when unlinking to identify
                                   ; compatible interrupt handlers
        FLAGS    DB
        FIRST   EQU     80H      ; Flags
        JMP     SHORT RESET     ; Flag for being first in chain
        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:     ...                    ; Disable the interrupts
        CLI   AL,EOI            ; Issue non-specific EOI to 8259
        OUT  DX,REARM           ; Rearm the cards
        OUT  DX,AL              ; Restore registers
        POP  ...
        IRET
RESET:    ...                    ; Disable your card
        ...                    ; Return FAR to operating system
ENTRY    ENDP
MYCSEG   ENDS
END      ENTRY

```

Linking Code Example

```
    PUSH    ES
    CLI                    ; Disable interrupts
; Set forward pointer to value of interrupt vector in low memory
    ASSUME  CS:CODESEG,DS:CODESEG
    PUSH    ES
    MOV     AX,350FH       ; DOS get interrupt vector
    INT     21H
    MOV     SI,OFFSET CS:FPTR ; Get offset of your forward pointer
                                ; in an indexable register
    MOV     CS:[SI],BX    ; Store the old interrupt vector
    MOV     CS:[SI+2],ES  ; in your forward pointer for chaining
    CMP     ES:BYTE PTR[BX],CFH ; Test for IRET
    JNZ     SETVECTR
    MOV     CS:FLAGS,FIRST ; Set up first in chain flag
SETVECTR: POP    ES
    PUSH    DS
; Make interrupt vector in low memory point to your handler
    MOV     DX,OFFSET ENTRY ; Make interrupt vector point to your handler
    MOV     AX,SEG ENTRY    ; if DS not = CS, get it
    MOV     DS,AX          ; and put it in DS
    MOV     AX,250FH       ; DOS set interrupt vector
    INT     21H
    POP     DS
; Unmask (enable) interrupts for your level
    IN      AL,IMR         ; Read interrupt mask register
    JMP     $+2            ; 10 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
            ; ES:[BX+6],4B42H ; Is handler using the appropriate
            ; conventions (is SIGNATURE present in
            ; the interrupt chaining structure)?
            ; No, invoke error exception handler
JNE     exception ; Get FPTR's segment and offset
LDS     SI,ES:[BX+2] ; Is this forward pointer pointing to
CMP     SI,OFFSET ENTRY ; your handler's offset?
JNE     UNCHAIN_B ; No, branch
MOV     CX,DS    ; Move to compare
CMP     AX,CX    ; Is this forward pointer pointing to
            ; your handler's segment?
JNE     UNCHAIN_B ; No, branch
; Located your handler in the chain
MOV     AX,WORD PTR CS:FPTR ; Get your FPTR's offset
MOV     ES:[BX+2],AX ; Replace offset of FPTR of handler
            ; that points to you
MOV     AX,WORD PTR CS:FPTR[2] ; Get your FPTR's segment
MOV     ES:[BX+4],AX ; Replace segment of FPTR of handler
            ; that points to you
MOV     AL,CS:FLAGS ; Get your flags
AND     AL,FIRST  ; Isolate FIRST flag
OR     ES:[BX + 6],AL ; Set your first flag into prior routine
JMP     UNCHAIN_X
UNCHAIN_B: MOV     BX,SI ; Move new offset to BX
            PUSH    DS
            PUSH    ES
            JMP     UNCHAIN_A ; Examine next handler in chain
UNCHAIN_X: STI     ; Enable interrupts
            POP     ES
            POP     DS

```

System Timers

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

Channel 0 System Timer

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

Channel 1 Refresh Request Generator

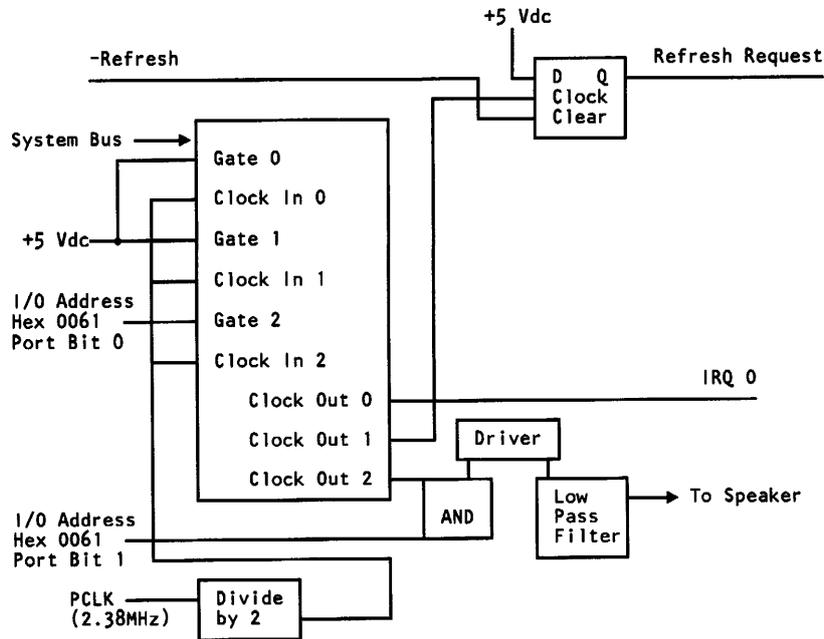
GATE 1 Tied on
CLK IN 1 1.193182 MHz OSC
CLK OUT 1 Request refresh cycle

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

Channel 2 Tone Generation for Speaker

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

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



System-Timer Block Diagram

System Clock

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

ROM Subsystem

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

RAM Subsystem

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

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

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

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

I/O Channel

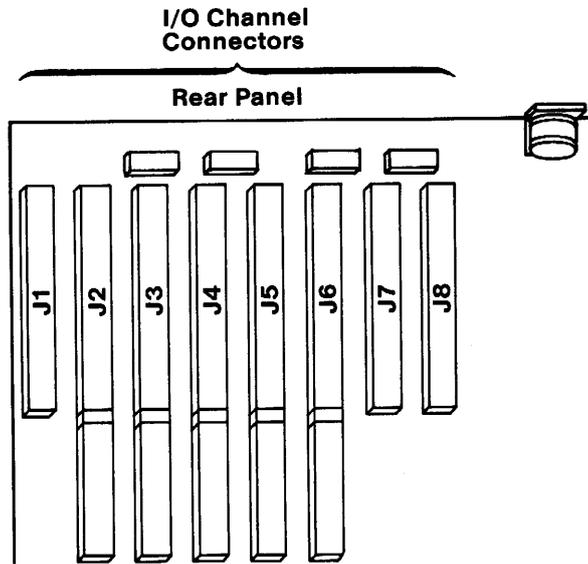
The I/O channel supports:

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

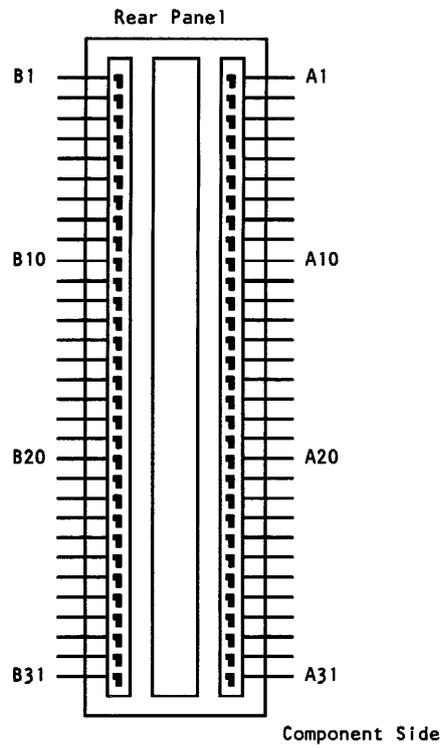
Connectors

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

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

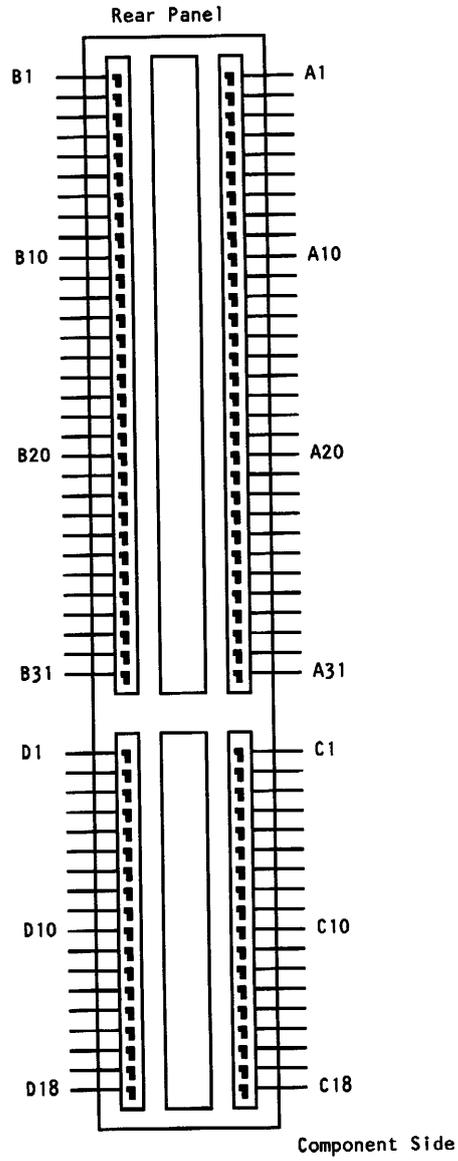


The following figure shows the pin numbering for the 62-pin I/O channel connectors J1, J7 and J8.



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

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



I/O Channel Pin Numbering (J2-J6)

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

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

I/O Channel (A-Side)

I/O Pin	Signal Name	I/O
B1	GND	Ground
B2	RESET DRV	0
B3	+5 Vdc	Power
B4	IRQ 9	I
B5	-5 Vdc	Power
B6	DRQ2	I
B7	-12 Vdc	Power
B8	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	14.318MHz OSC	0
B31	GND	Ground

I/O Channel (B-Side)

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

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

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

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

I/O Channel Signal Description

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

SA0 through SA19 (I/O)

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

LA17 through LA23 (I/O)

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

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

CLK (O)

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

RESET DRV (O)

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

SD0 through SD15 (I/O)

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

BALE (O) (buffered)

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

-I/O CH CK (I)

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

I/O CH RDY (I)

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

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

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

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

-IOR (I/O)

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

-IOW (I/O)

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

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

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

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

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

DRQ0-DRQ3 and DRQ5-DRQ7 (I)

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

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

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

AEN (O)

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

-REFRESH (I/O)

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

T/C (O)

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

-SBHE (I/O)

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

-MASTER (I)

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

-MEM CS16 (I)

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

-I/O CS16 (I)

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

14.318MHz OSC (0)

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

OWS (I)

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

I/O Addresses

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

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

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

I/O Address Map

NMI Controls

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

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

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

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

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

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

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

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

I/O Port (Read/Write)

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

Input (Read)

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

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

Output (Write)

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

Bits 7 to 4 Not used

Bit 3 -Enable I/O Channel Check

0 Enables I/O Channel Check errors.

1 Disables I/O Channel Check errors.

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

Bit 2 -Enable System Board RAM Parity Check

0 Enables system board RAM parity check.

1 Disables system board RAM parity check.

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

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

Bit 0 +Timer 2 Gate Speaker

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

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

Diagnostic-Checkpoint Port

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

Coprocessor Controls

The following is a description of the Math Coprocessor controls.

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

Other Circuits

Speaker

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

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

128K RAM Jumper (J10)

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

Pin	Assignments
1	No Connection
2	Ground
3	RAM Select

RAM Jumper Connector (J10)

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

Note: The normal mode is the enabled mode.

Display Switch

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

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

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

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

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

Keyboard Controller

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

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

Keyboard Controller Initialization

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

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

Receiving Data from the Keyboard

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

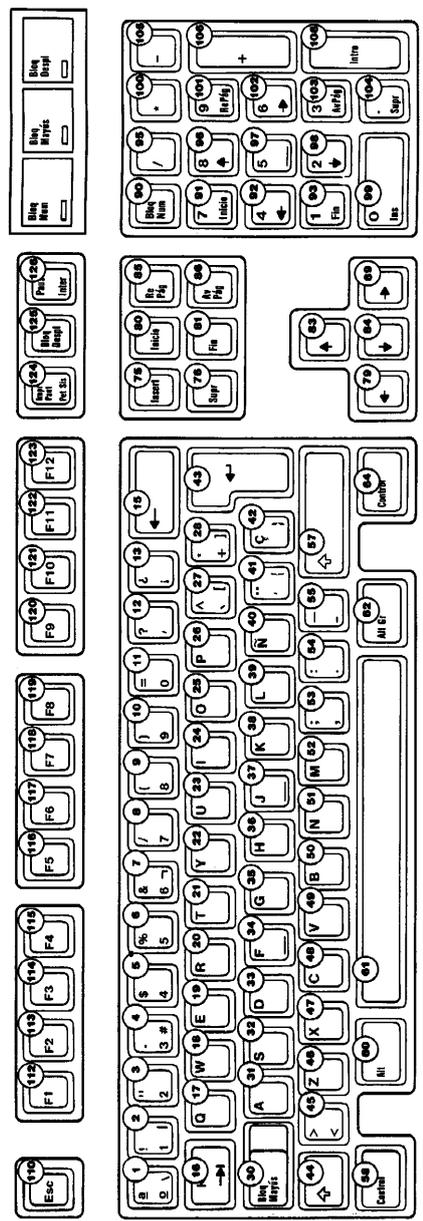
Note: When a receive error occurs in the default mode (bits 5, 6, and 7 of the command byte set to 0), hex 00 is placed in the output buffer instead of hex FF. See

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

Scan Code Translation

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

102-Key Keyboard



The following figure is the scan-code translation table.

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

Scan-Code Translation Table (Part 1 of 3)

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

Scan-Code Translation Table (Part 2 of 3)

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

Scan-Code Translation Table (Part 3 of 3)

The following scan codes are reserved.

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

Reserved Scan-Code Translation Table

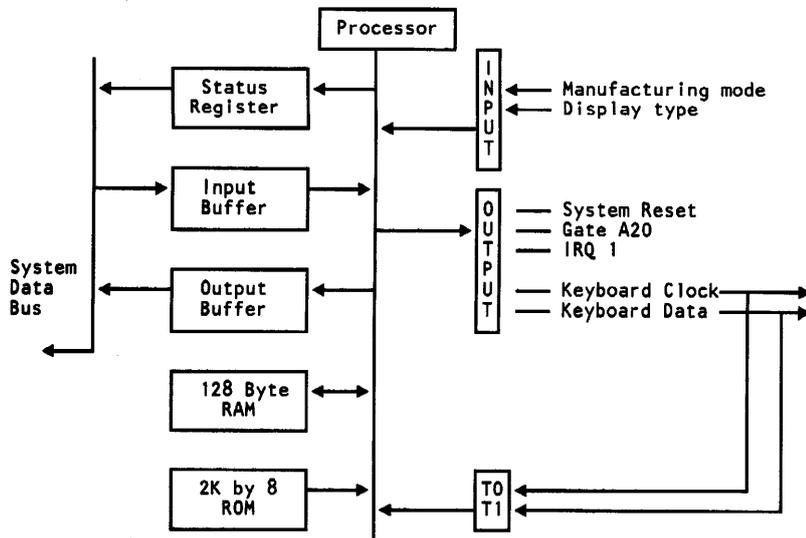
Sending Data to the Keyboard

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

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

Keyboard Controller System Interface

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



Keyboard Controller Interface Block Diagram

Status Register

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

Status-Register Bit Definition

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

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

Bit 4 Always set to 1.

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

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

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

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

Output Buffer

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

Input Buffer

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

Commands (I/O Address Hex 64)

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

Bit 7 Reserved—Should be written as a 0.

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

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

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

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

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

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

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

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

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

I/O Ports

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

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

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

Input-Port Bit Definitions

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

Output-Port Bit Definitions

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

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

Test-Input Bit Definitions

Real-Time Clock CMOS RAM Information

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

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

CMOS RAM Internal Address Map

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

Real-Time Clock Information

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

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

Real-Time Clock Internal Addresses 00 - 0D

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

Status Register A

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

Status Register B

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

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

Status Register C

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

Status Register D

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

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

CMOS RAM Configuration Information

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

Diagnostic Status Byte (Hex 0E)

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

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

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

Bit 1–Bit 0 Reserved

Shutdown Status Byte (Hex 0F)

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

Diskette Drive Type Byte (Hex 10)

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

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

Note: 0100 through 1111 are reserved.

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

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

Note: 0100 through 1111 are reserved.

Hex address 11 contains a reserved byte.

Fixed Disk Type Byte (Hex 12)

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

0000 No fixed disk drive is present.

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

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

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

0000 No fixed disk drive is present.

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

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

The following table shows the BIOS fixed disk parameters.

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

BIOS Fixed Disk Parameters

Hex address 13 contains a reserved byte.

Equipment Byte (Hex 14)

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

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

Bit 5–Bit 4 Primary display

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

Bit 3–Bit 2 Not used.

Bit 1 Math Coprocessor presence bit:

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

Bit 0 Diskette drive presence bit:

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

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

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

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

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

Valid Sizes:

0200H 512K—system board RAM

0280H 640K—system board RAM.

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

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

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

Valid Sizes:

0200H 512K—Expansion Memory

0400H 1024K—Expansion Memory

0600H 1536K—Expansion Memory

through

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

Drive C Extended Byte (Hex 19)

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

00000000 through 00001111 are reserved.

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

Drive D Extended Byte (Hex 1A)

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

00000000 through 00001111 are reserved.

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

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

Note: Types 25 through 255 are reserved.

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

BIOS Fixed Disk Parameters (Extended)

Hex addresses 1B through 2D are reserved.

Checksum (Hex 2E and 2F)

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

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

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

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

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

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

Valid Sizes:

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

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

Date Century Byte (Hex 32)

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

Information Flag (Hex 33)

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

Hex addresses 34 through 3F are reserved.

I/O Operations

Writing to RTC CMOS RAM involves two steps:

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

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

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

Reading CMOS RAM also requires two steps:

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

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

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

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



Specifications

System Unit

Size

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

Weight

- 12.7 kilograms (28 pounds)

Power Cables

- Length: 1.8 meters (6 feet)

Environment

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

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

Heat Output

- 824 British Thermal Units (BTU) per hour

Noise Level

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

Electrical

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

Connectors

The system board has the following additional connectors:

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

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

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

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

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

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

Battery Connector (P2)

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

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

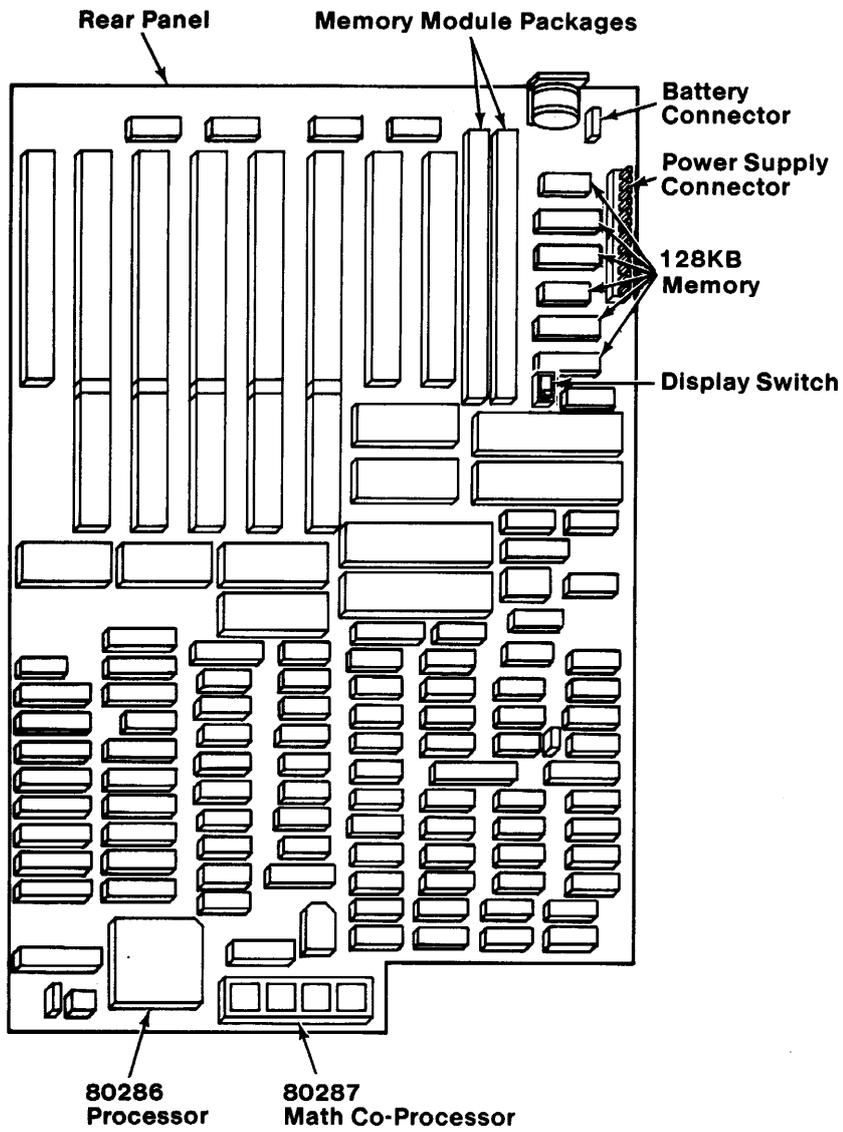
Speaker Connector (P3)

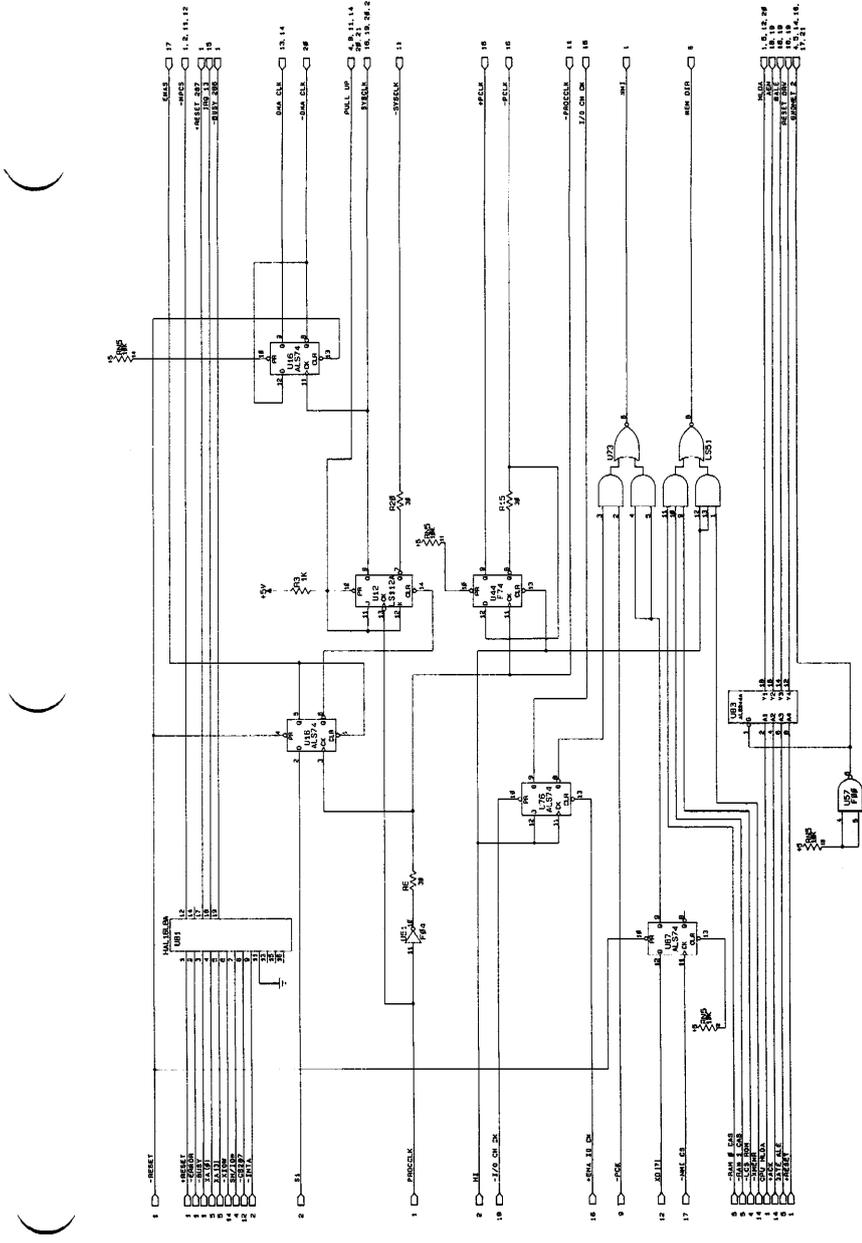
The keyboard connector, J9, is a five-pin, 90-degree Printed Circuit Board (PCB) mounting, DIN connector. For pin numbering, see the "Keyboard" Section. The pin assignments are:

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

Keyboard Connector (J9)

The following figure shows the layout of the system board.

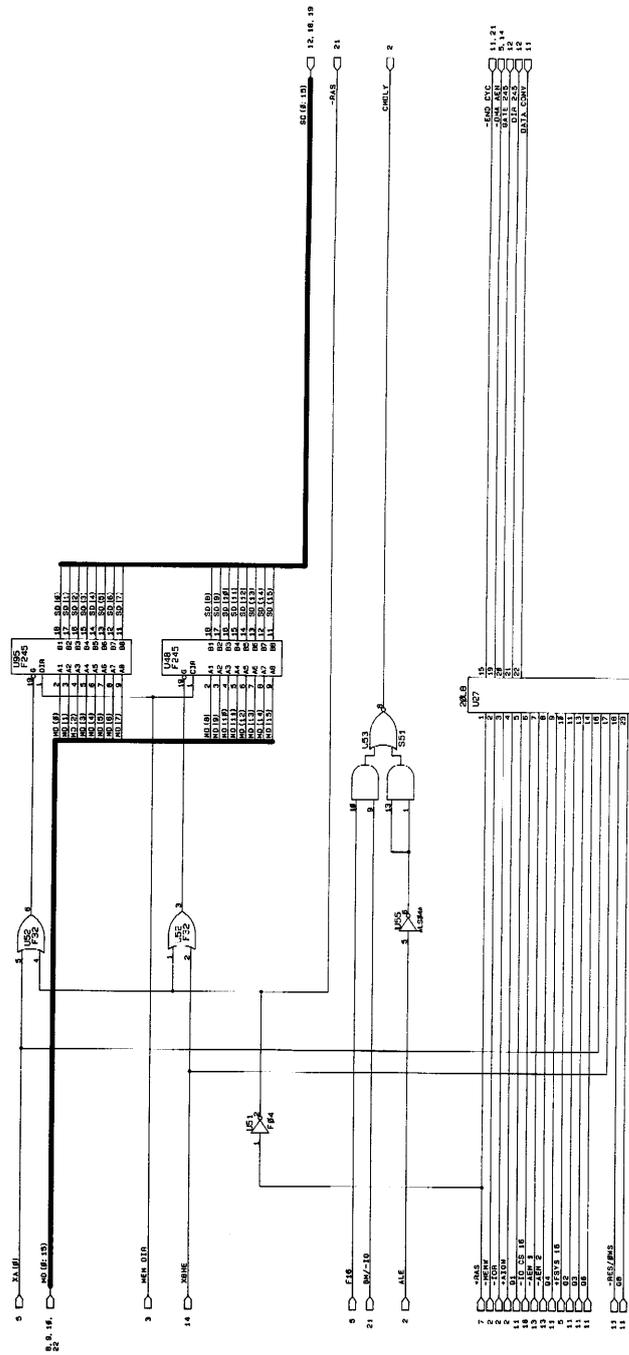




System Board (Sheet 3 of 22)

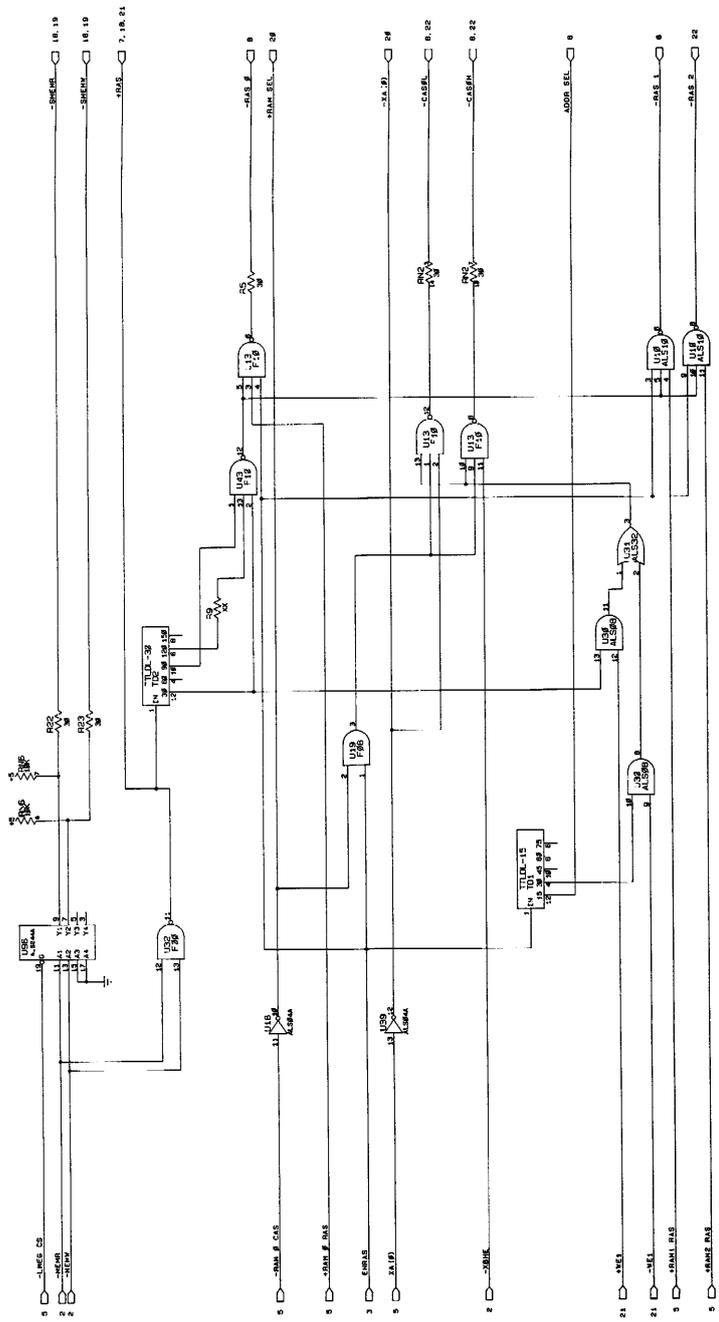
SECTION 1

1-82 System Board

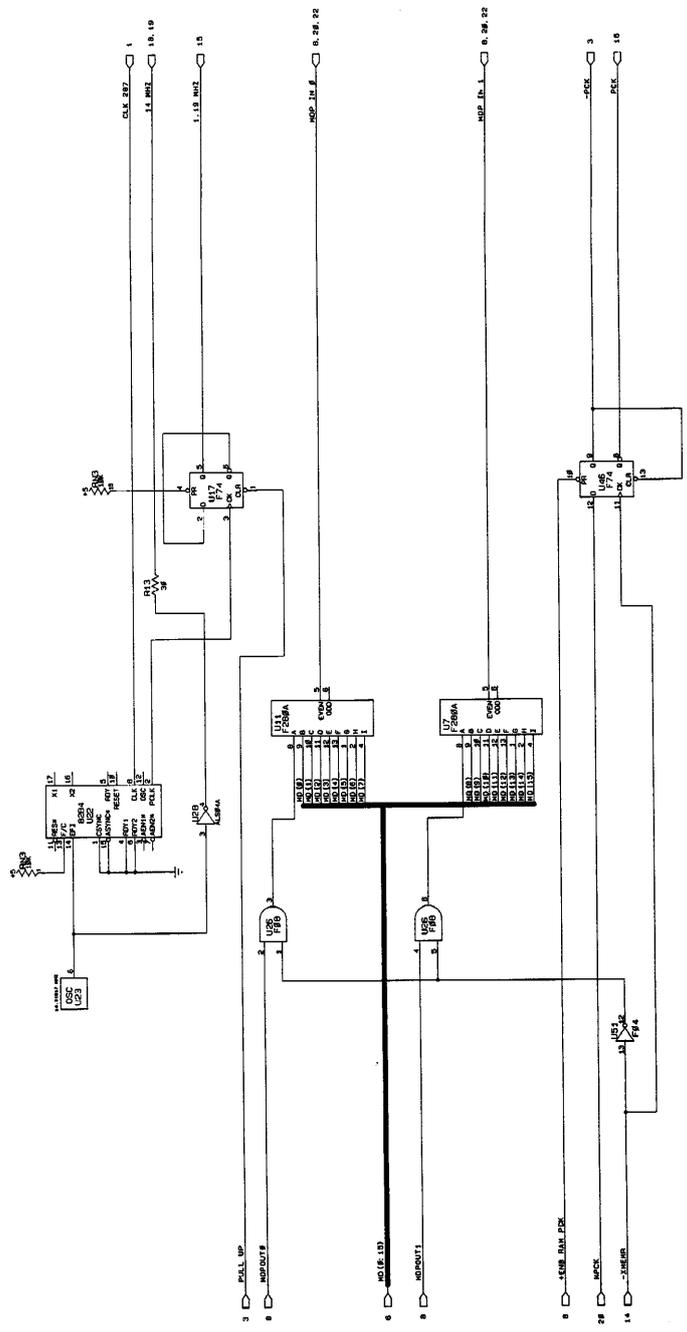


System Board (Sheet 6 of 22)

()



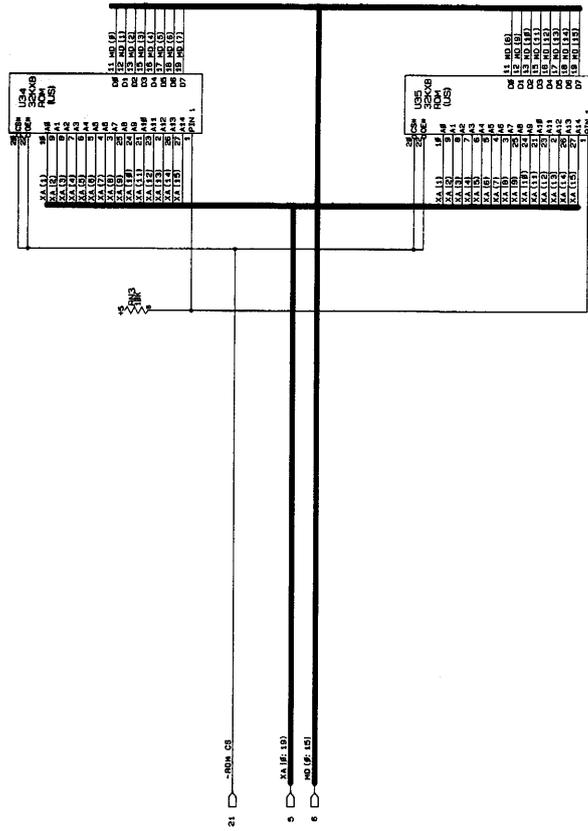
() ()



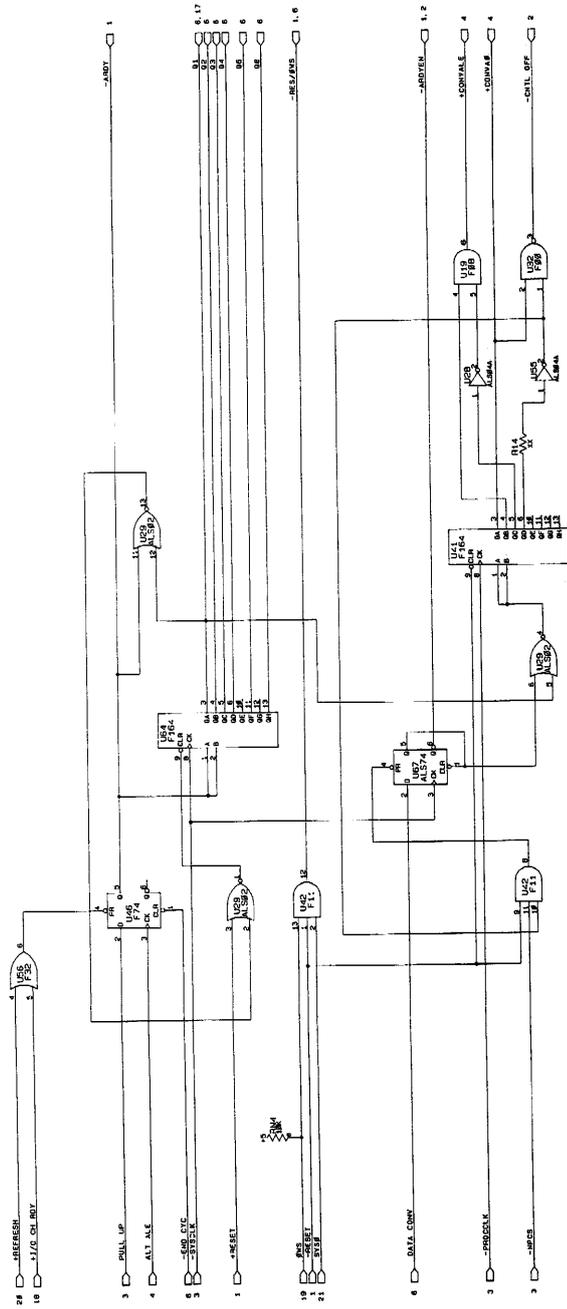
System Board (Sheet 9 of 22)

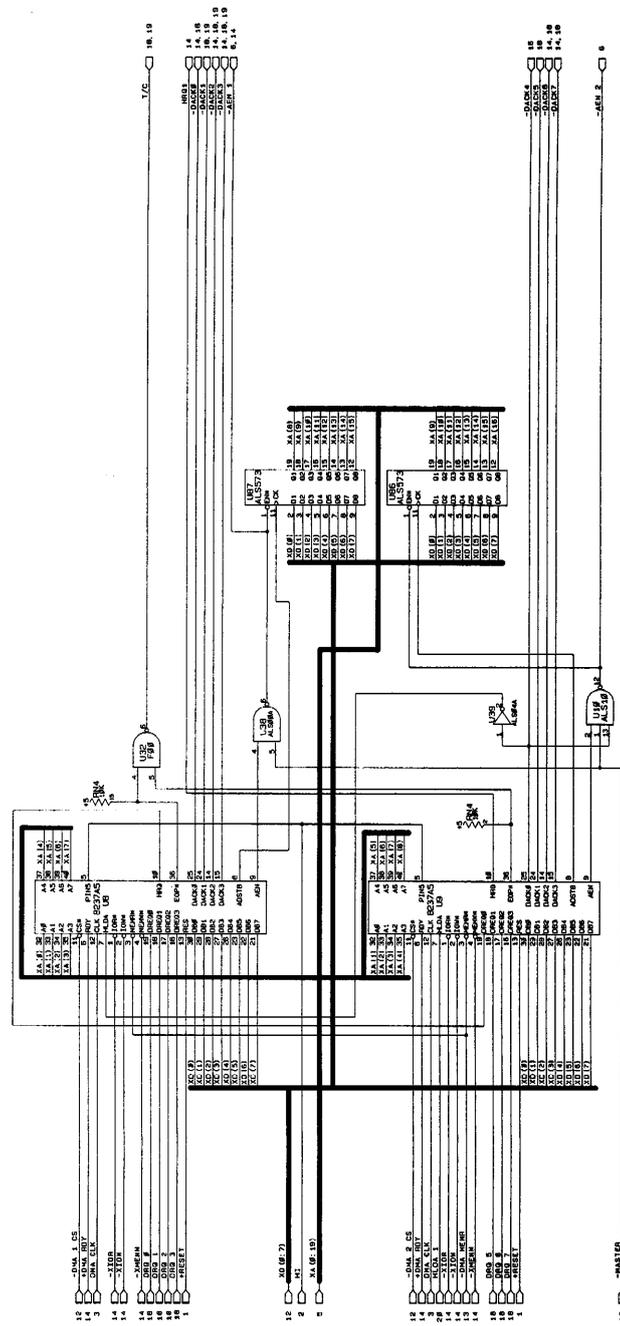
SECTION 1

1-86 System Board



System Board (Sheet 10 of 22)

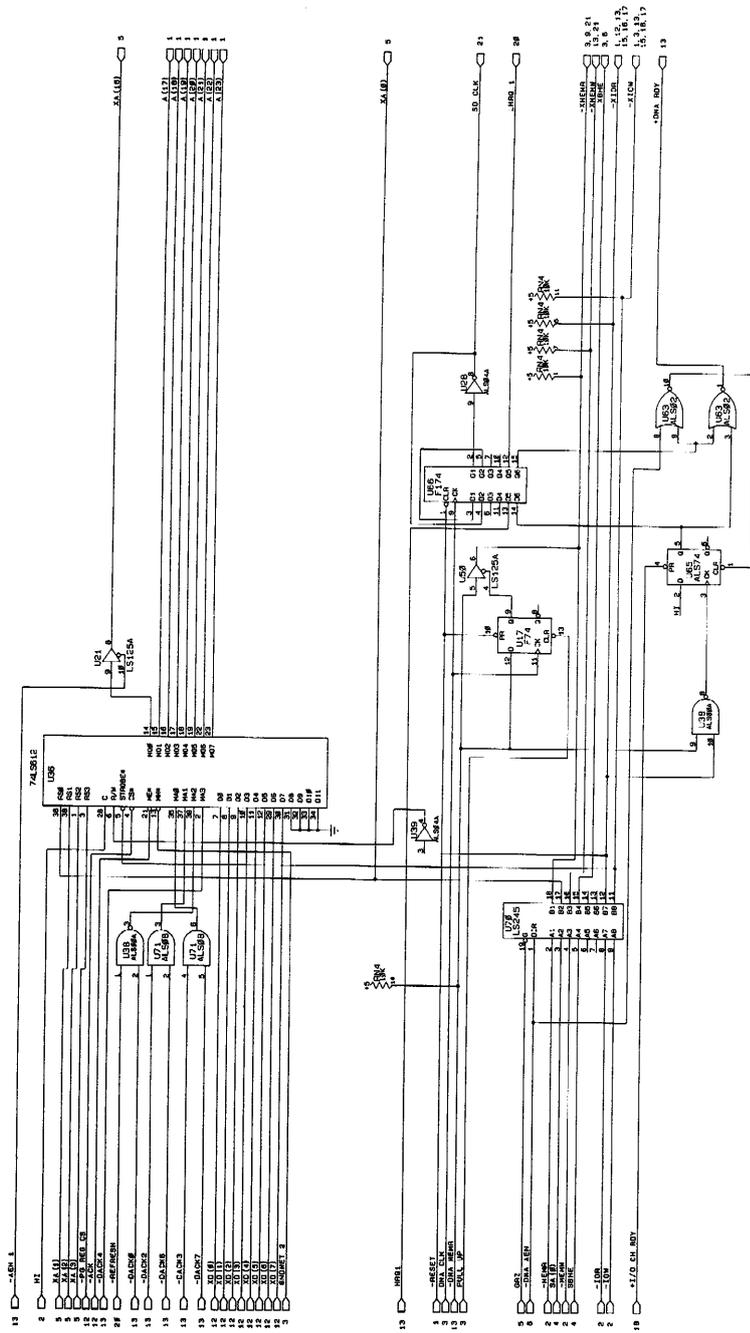




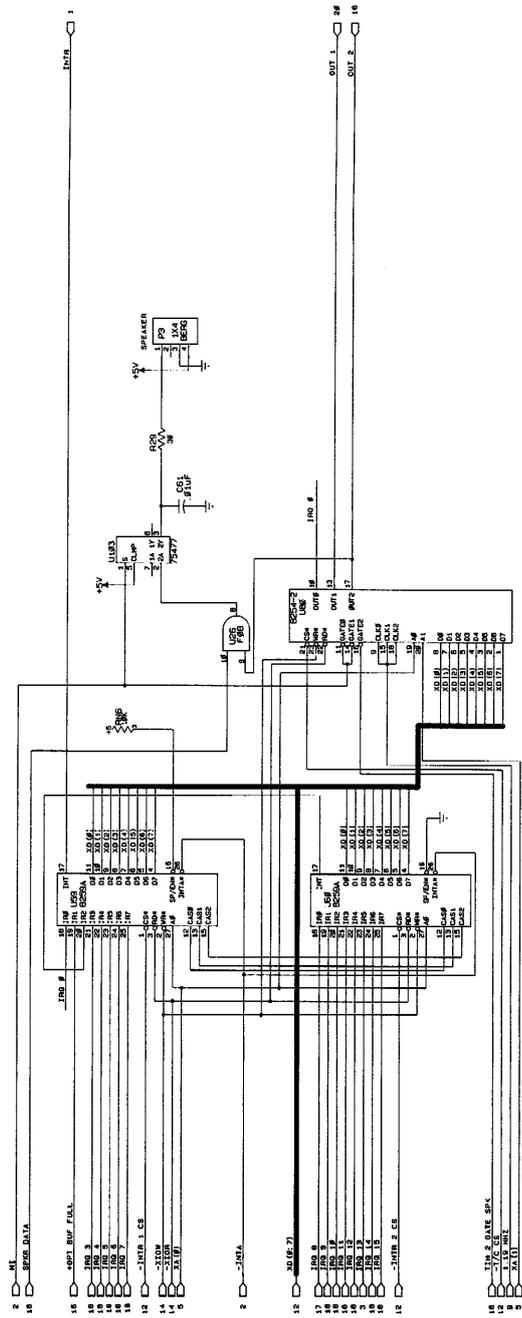
System Board (Sheet 13 of 22)

SECTION 1

1-90 System Board



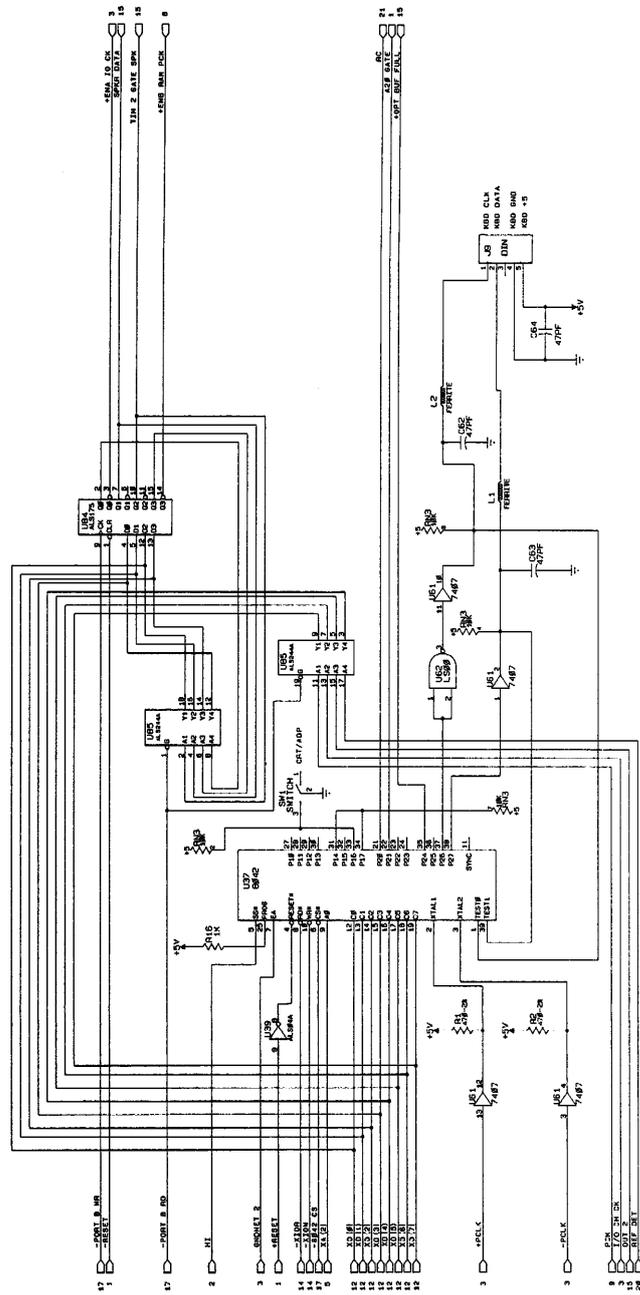
System Board (Sheet 14 of 22)



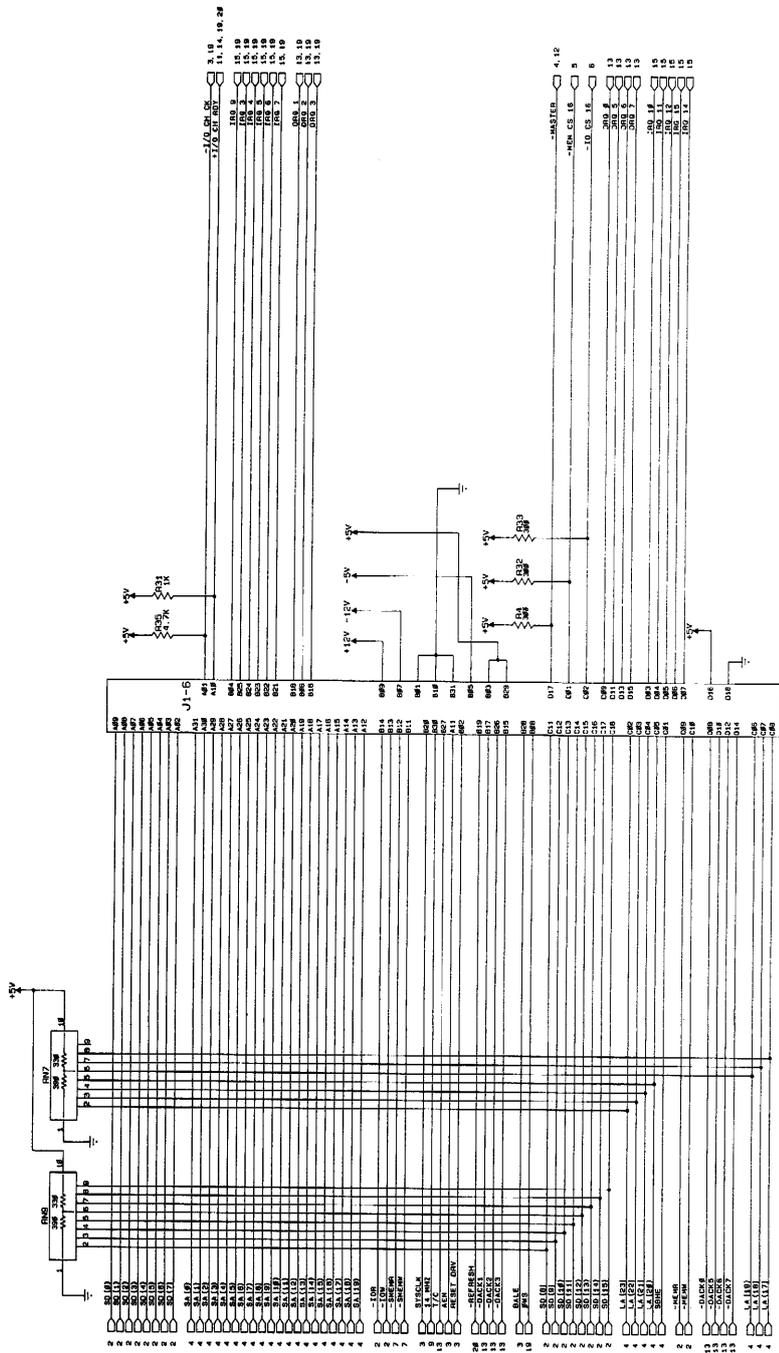
System Board (Sheet 15 of 22)

SECTION 1

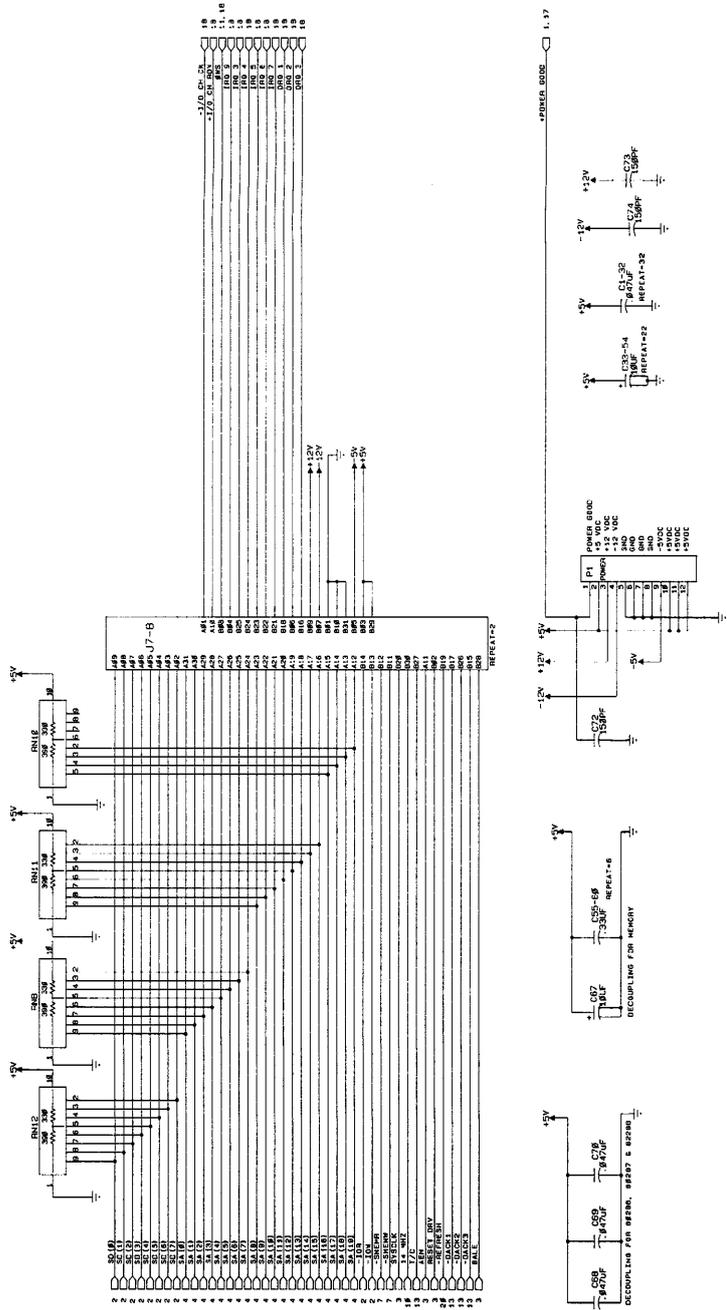
1-92 System Board



1-94 System Board



System Board (Sheet 18 of 22)

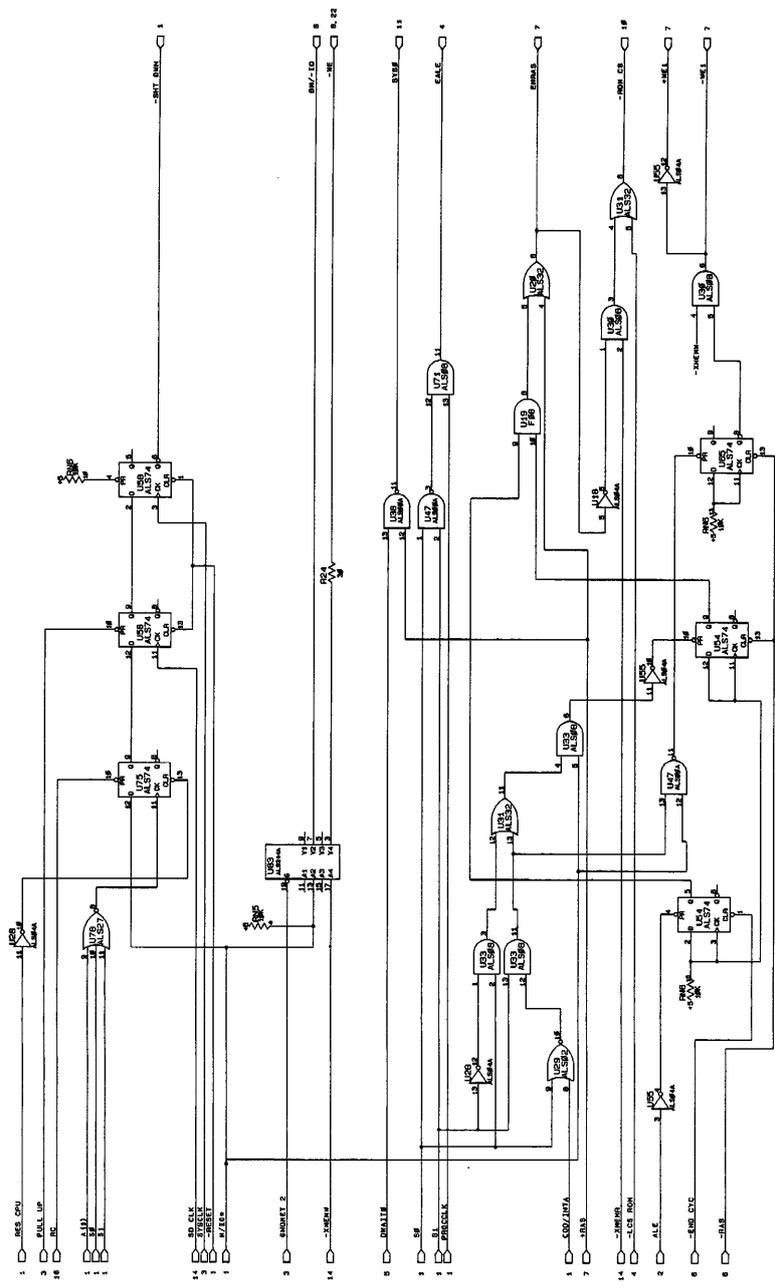


System Board 1-95

System Board (Sheet 19 of 22)

SECTION 1

() ()



SECTION 2. COPROCESSOR

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

Notes:

|

—

—

—

Description

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

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

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

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

Programming Interface

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

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

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

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

Data Types

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

Hardware Interface

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

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

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

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

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

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

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

Notes:

|

—

—

—

SECTION 3. POWER SUPPLY

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

SECTION 3

Notes:

3-2 Power Supply

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

Inputs

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

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

Input Requirements

Outputs

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

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

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

DC Load Requirements

DC Output Protection

An overcurrent condition will not damage the power supply.

Output Voltage Sequencing

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

No-Load Operation

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

Power-Good Signal

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

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

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

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

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

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

Sense Level

Connectors

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

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

Power Supply Output Connectors

SECTION 4. KEYBOARD

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

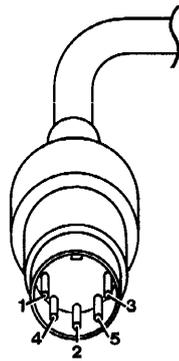
Keyboard Data Output	4-28
Keyboard Data Input	4-29
Keyboard Encoding and Usage	4-30
Character Codes	4-30
Extended Functions	4-34
Shift States	4-36
Special Handling	4-38
Keyboard Layouts	4-40
French Keyboard	4-41
German Keyboard	4-42
Italian Keyboard	4-43
Spanish Keyboard	4-44
U.K. English Keyboard	4-45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

Description

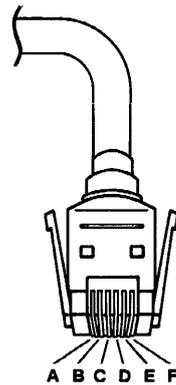
The keyboard has 101 keys (102 in countries outside the U. S.). At system power-on, the keyboard monitors the signals on the 'clock' and 'data' lines and establishes its line protocol. A bidirectional serial interface in the keyboard converts the 'clock' and 'data' signals and sends this information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

Cabling

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



DIN Connector



SDL Connector

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

Sequencing Key-Code Scanning

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

Keyboard Buffer

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

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

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

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

Keys

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

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

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

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

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

Power-On Routine

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

Power-On Reset

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

Basic Assurance Test

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

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

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

Commands from the System

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

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

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

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

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

Default Disable (Hex F5)

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

Echo (Hex EE)

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

Enable (Hex F4)

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

Invalid Command (Hex EF and F1)

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

Read ID (Hex F2)

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

Resend (Hex FE)

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

Reset (Hex FF)

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

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

Select Alternate Scan Codes (Hex F0)

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

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

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

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

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

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

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

Set Default (Hex F6)

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

Set Key Type (Hex FB, FC, FD)

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

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

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

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

Set/Reset Status Indicators (Hex ED)

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

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

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

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

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

Set Typematic Rate/Delay (Hex F3)

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

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

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

where:

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

B = binary value of bits 4 and 3.

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

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

The default values for the system keyboard are as follows:

Typematic rate = 10.9 characters per second \pm 20%.

Delay = 500 milliseconds \pm 20%.

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

Commands to the System

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

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

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

Acknowledge (Hex FA)

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

BAT Completion Code (Hex AA)

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

BAT Failure Code (Hex FC)

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

Echo (Hex EE)

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

Keyboard ID (Hex 83AB)

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

Key Detection Error (Hex 00 or FF)

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

Overrun (Hex 00 or FF)

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

Resend (Hex FE)

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

Keyboard Scan Codes

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

Scan Code Set 1

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

Scan Code Tables (Set 1)

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

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

* 101-key keyboard only.

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

** 102-key keyboard only.

SECTION 4

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

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

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

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

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

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

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

SECTION 4

Scan Code Set 2

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

Scan Code Tables (Set 2)

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

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

* 101-key keyboard only.

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

** 102-key keyboard only.

SECTION 4

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

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

Key No.	Base Case, or Shift+Num Lock Make/Break	Shift Case Make/Break *	Num Lock on Make/Break
75	E0 70 /E0 F0 70	E0 F0 12 E0 70 /E0 F0 70 E0 12	E0 12 E0 70 /E0 F0 70 E0 F0 12
76	E0 71 /E0 F0 71	E0 F0 12 E0 71 /E0 F0 71 E0 12	E0 12 E0 71 /E0 F0 71 E0 F0 12
79	E0 6B /E0 F0 6B	E0 F0 12 E0 6B /E0 F0 6B E0 12	E0 12 E0 6B /E0 F0 6B E0 F0 12
80	E0 6C /E0 F0 6C	E0 F0 12 E0 6C /E0 F0 6C E0 12	E0 12 E0 6C /E0 F0 6C E0 F0 12
81	E0 69 /E0 F0 69	E0 F0 12 E0 69 /E0 F0 69 E0 12	E0 12 E0 69 /E0 F0 69 E0 F0 12
83	E0 75 /E0 F0 75	E0 F0 12 E0 75 /E0 F0 75 E0 12	E0 12 E0 75 /E0 F0 75 E0 F0 12
84	E0 72 /E0 F0 72	E0 F0 12 E0 72 /E0 F0 72 E0 12	E0 12 E0 72 /E0 F0 72 E0 F0 12
85	E0 7D /E0 F0 7D	E0 F0 12 E0 7D /E0 F0 7D E0 12	E0 12 E0 7D /E0 F0 7D E0 F0 12
86	E0 7A /E0 F0 7A	E0 F0 12 E0 7A /E0 F0 7A E0 12	E0 12 E0 7A /E0 F0 7A E0 F0 12
89	E0 74 /E0 F0 74	E0 F0 12 E0 74 /E0 F0 74 E0 12	E0 12 E0 74 /E0 F0 74 E0 F0 12
* If the left Shift key is held down, the F0 12/12 shift make and break is sent with the other scan codes. If the right Shift key is held down, F0 59/59 is sent. If both Shift keys are down, both sets of codes are sent with the other scan code.			

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

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

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

SECTION 4

Scan Code Set 3

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

Scan Code Tables (Set 3)

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

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

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

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

SECTION 4

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

Clock and Data Signals

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

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

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

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

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

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

Data Stream

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

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

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

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

Keyboard Data Output

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

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

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

Keyboard Data Input

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

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

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

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

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

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

Keyboard Encoding and Usage

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

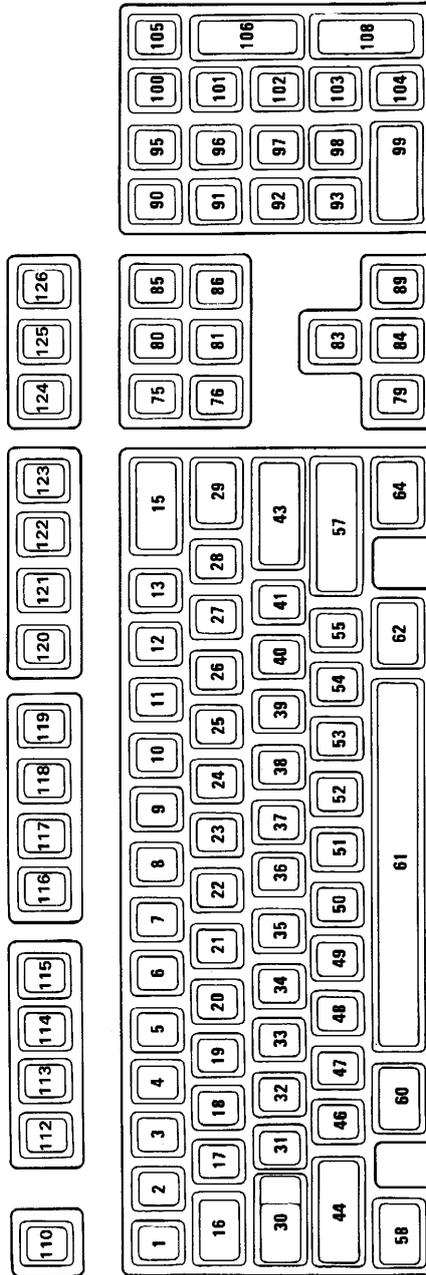
Extended ASCII encompasses 1-byte character codes, with possible values of 0 to 255, an extended code for certain extended keyboard functions, and functions handled within the keyboard routine or through interrupts.

Character Codes

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

The following figure shows the keyboard layout and key positions.

() () ()



SECTION 4

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

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

Character Codes (Part 1 of 2)

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

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

SECTION 4

Character Codes (Part 2 of 2)

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

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

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

Special Character Codes

Extended Functions

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

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

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

Keyboard Extended Functions (Part 1 of 2)

SECTION 4

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

Keyboard Extended Functions (Part 2 of 2)

Shift States

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

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

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

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

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

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

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

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

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

Special Handling

System Reset

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

Break

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

Pause

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

Print Screen

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

System Request

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

Save the previous address.

Overlay interrupt vector hex 15.

Check AH for a value of hex 85:

If yes, process may begin.

If no, go to previous address.

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

Other Characteristics

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

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

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

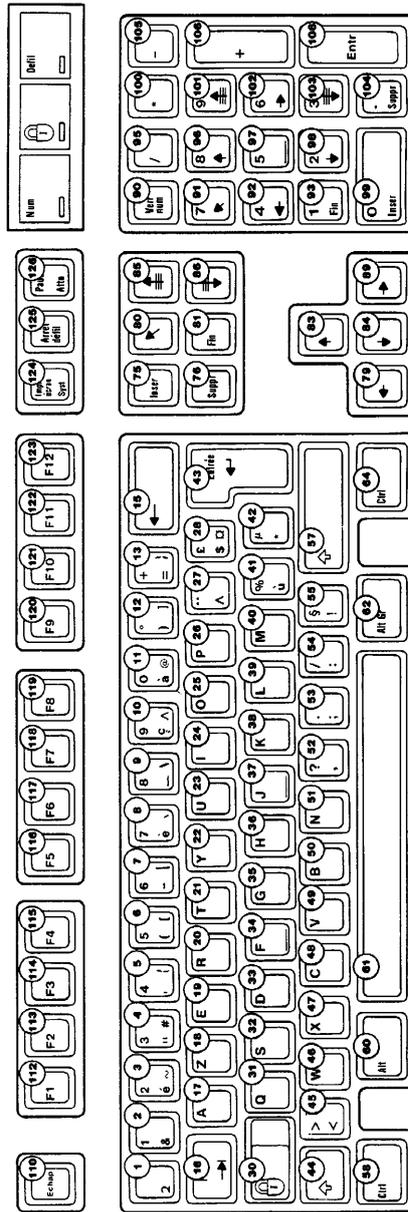
Keyboard Layouts

The keyboard is available in six layouts:

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

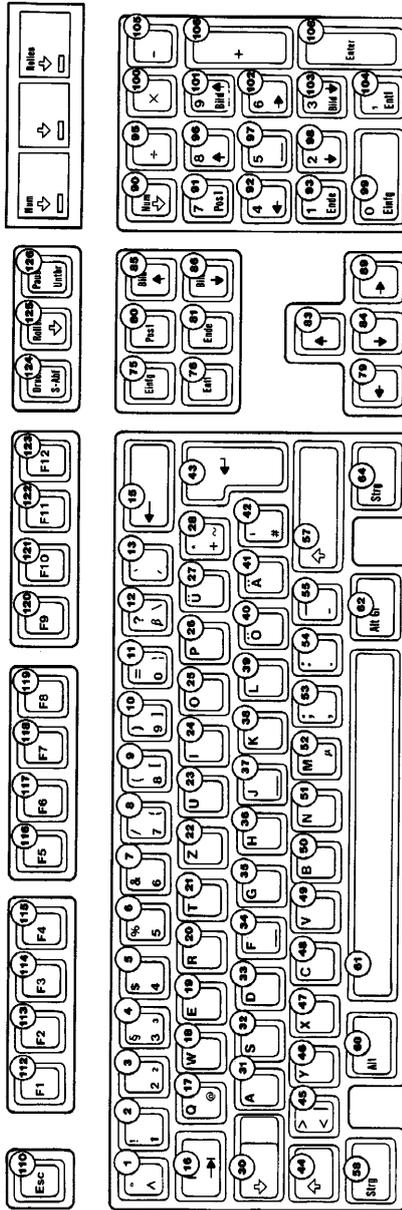
The various layouts are shown in alphabetic order on the following pages. Nomenclature is on both the top and front face of the keybuttons. The number to the upper right designates the keybutton position.

French Keyboard



SECTION 4

German Keyboard



Specifications

The specifications for the keyboard are as follows.

Power Requirements

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

Size

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

Weight

2.25 kilograms (5.0 pounds)

SECTION 5. SYSTEM BIOS

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

Notes:

System BIOS Usage

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

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

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

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

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

Parameter Passing

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

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

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

To read the time of day:

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

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

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

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

80286-2 Program Interrupt Listing (Real Mode Only)

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

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

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

Hardware, Basic, and DOS Interrupts

Vectors with Special Meanings

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

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

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

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

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

This routine may retain control with the following considerations:

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

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

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

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

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

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

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

Other Read/Write Memory Usage

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

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

The following figure shows the reserved memory locations.

Address	Mode	Function
400-4A1	ROM BIOS	See BIOS listing
4A2-4EF		Reserved
4F0-4FF		Reserved as intra-application communication area for any application
500-5FF	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	BASIC	Clock interrupt vector segment:offset store
516-519	BASIC	Break key interrupt vector segment:offset store
51A-51D	BASIC	Disk error interrupt vector segment:offset store

Reserved Memory Locations

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

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

Basic Workspace Variables

Example

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

L	H
Hex 64	Hex 00

The following is a BIOS memory map.

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

BIOS Memory Map

BIOS Programming Hints

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

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

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

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

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

Move Block BIOS

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

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

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

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

Times are approximate

Communication Interrupt Intervals

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

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

Move Block BIOS Timing

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

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

Adapters with System-Accessible ROM Modules

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

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

Byte 0 Hex 55
Byte 1 Hex AA

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

Byte 3 Entry by a CALL FAR

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

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

Quick Reference

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

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

Address	Publics by Name	Address	Publics by Value
F000:1A04	BECP	F000:0000	POST1
F000:1A04	BECP	F000:0008	K6L
F000:1A04	BECP	F000:0010	M4
F000:1A04	BECP	F000:0080	Abas
F000:1A04	BECP	F000:0010	Abas
F000:1A04	BECP	F000:0080	START_1
F000:1A04	BECP	F000:0010	C8042
F000:1A04	BECP	F000:0080	OBF_42
F000:1A04	BECP	F000:0010	C21
F000:1A04	BECP	F000:0080	POST2
F000:1A04	BECP	F000:0010	SHUT3
F000:1A04	BECP	F000:0080	SHUT2
F000:1A04	BECP	F000:0010	SHUT7
F000:1A04	BECP	F000:0080	SHUT6
F000:1A04	BECP	F000:0010	SHUT4
F000:1A04	BECP	F000:0080	POST3
F000:1A04	BECP	F000:0010	CMOS_READ
F000:1A04	BECP	F000:0080	POST4
F000:1A04	BECP	F000:0010	CMOS_WRITE
F000:1A04	BECP	F000:0080	DDS
F000:1A04	BECP	F000:0010	E_MSG
F000:1A04	BECP	F000:0080	P_MSG
F000:1A04	BECP	F000:0010	ERR_BEEP
F000:1A04	BECP	F000:0080	BEEP
F000:1A04	BECP	F000:0010	WAITF
F000:1A04	BECP	F000:0080	CONFIG_BAD
F000:1A04	BECP	F000:0010	XPC_BYTE
F000:1A04	BECP	F000:0080	PRT_HEX
F000:1A04	BECP	F000:0010	PRT_SEG
F000:1A04	BECP	F000:0080	PROT_PRT_HEX
F000:1A04	BECP	F000:0010	ROM_CHECKSUM
F000:1A04	BECP	F000:0080	ROM_CHECK
F000:1A04	BECP	F000:0010	KBD_RESET
F000:1A04	BECP	F000:0080	BLINK_INT
F000:1A04	BECP	F000:0010	SET_T00
F000:1A04	BECP	F000:0080	D11
F000:1A04	BECP	F000:0010	DUMMY_RETURN_1
F000:1A04	BECP	F000:0080	RE_DIRECT
F000:1A04	BECP	F000:0010	INT_287
F000:1A04	BECP	F000:0080	PROC_SHUTDOWN
F000:1A04	BECP	F000:0010	POST5
F000:1A04	BECP	F000:0080	SYSDINITI
F000:1A04	BECP	F000:0010	POST6
F000:1A04	BECP	F000:0080	ST32ST_CNT
F000:1A04	BECP	F000:0010	ROM_ERR
F000:1A04	BECP	F000:0080	XMIT_8042
F000:1A04	BECP	F000:0010	BOOT_STRAP_1
F000:1A04	BECP	F000:0080	DISKETTE_IO_1
F000:1A04	BECP	F000:0010	SEEK
F000:1A04	BECP	F000:0080	DISK_INT_1
F000:1A04	BECP	F000:0010	DISKETTE_SETUP
F000:1A04	BECP	F000:0080	DISK_SETUP
F000:1A04	BECP	F000:0010	DISK_IO
F000:1A04	BECP	F000:0080	HD_INT
F000:1A04	BECP	F000:0010	KEYBOARD_IO_1
F000:1A04	BECP	F000:0080	KB_INT_1
F000:1A04	BECP	F000:0010	K15
F000:1A04	BECP	F000:0080	SND_DATA
F000:1A04	BECP	F000:0010	PRINTER_IO_1
F000:1A04	BECP	F000:0080	RS232_IO_1
F000:1A04	BECP	F000:0010	VIDEO_IO_1
F000:1A04	BECP	F000:0080	SET_MODE
F000:1A04	BECP	F000:0010	SET_CTYPE
F000:1A04	BECP	F000:0080	SET_CPOS
F000:1A04	BECP	F000:0010	READ_CURSOR
F000:1A04	BECP	F000:0080	ACT_DISP_PAGE
F000:1A04	BECP	F000:0010	SET_COLOR
F000:1A04	BECP	F000:0080	VIDEO_STATE
F000:1A04	BECP	F000:0010	SCROLL_UP
F000:1A04	BECP	F000:0080	SCROLL_DOWN
F000:1A04	BECP	F000:0010	READ_AC_CURRENT
F000:1A04	BECP	F000:0080	WRITE_AC_CURRENT
F000:1A04	BECP	F000:0010	WRITE_C_CURRENT
F000:1A04	BECP	F000:0080	READ_DOT
F000:1A04	BECP	F000:0010	WRITE_DOT
F000:1A04	BECP	F000:0080	MEMORY_SIZE_DET_1
F000:1A04	BECP	F000:0010	EQUIPMENT_1
F000:1A04	BECP	F000:0080	NMI_INT_1
F000:1A04	BECP	F000:0010	CASSETTE_IO_1
F000:1A04	BECP	F000:0080	SHUT9
F000:1A04	BECP	F000:0010	GATE_A20
F000:1A04	BECP	F000:0080	TIME_OF_DAY_1
F000:1A04	BECP	F000:0010	RTC_INT
F000:1A04	BECP	F000:0080	PRINT_SCREEN_1
F000:1A04	BECP	F000:0010	TIMER_INT_1
F000:1A04	BECP	F000:0080	FILL
F000:1A04	BECP	F000:0010	BASIC
F000:1A04	BECP	F000:0080	D1
F000:1A04	BECP	F000:0010	D2
F000:1A04	BECP	F000:0080	D2A
F000:1A04	BECP	F000:0010	E101
F000:1A04	BECP	F000:0080	E102
F000:1A04	BECP	F000:0010	E103
F000:1A04	BECP	F000:0080	E104
F000:1A04	BECP	F000:0010	E105
F000:1A04	BECP	F000:0080	E106
F000:1A04	BECP	F000:0010	E107
F000:1A04	BECP	F000:0080	E108
F000:1A04	BECP	F000:0010	E109
F000:1A04	BECP	F000:0080	E161
F000:1A04	BECP	F000:0010	E162
F000:1A04	BECP	F000:0080	E163
F000:1A04	BECP	F000:0010	E164
F000:1A04	BECP	F000:0080	E201
F000:1A04	BECP	F000:0010	E202
F000:1A04	BECP	F000:0080	E203
F000:1A04	BECP	F000:0010	E301
F000:1A04	BECP	F000:0080	E303
F000:1A04	BECP	F000:0010	E304
F000:1A04	BECP	F000:0080	E401
F000:1A04	BECP	F000:0010	E501
F000:1A04	BECP	F000:0080	E601
F000:1A04	BECP	F000:0010	E602
F000:1A04	BECP	F000:0080	F30
F000:1A04	BECP	F000:0010	F3D
F000:1A04	BECP	F000:0080	F3D1
F000:1A04	BECP	F000:0010	FD_TBL
F000:1A04	BECP	F000:0080	FIL
F000:1A04	BECP	F000:0010	FLOPPY
F000:1A04	BECP	F000:0080	GATE_A20
F000:1A04	BECP	F000:0010	KB_INT_1
F000:1A04	BECP	F000:0080	HRB
F000:1A04	BECP	F000:0010	INT_287
F000:1A04	BECP	F000:0080	K10
F000:1A04	BECP	F000:0010	K11
F000:1A04	BECP	F000:0080	K12
F000:1A04	BECP	F000:0010	K14
F000:1A04	BECP	F000:0080	K15
F000:1A04	BECP	F000:0010	K16
F000:1A04	BECP	F000:0080	K6
F000:1A04	BECP	F000:0010	Abas
F000:1A04	BECP	F000:0080	K6L
F000:1A04	BECP	F000:0010	K8
F000:1A04	BECP	F000:0080	KBD_RESET
F000:1A04	BECP	F000:0010	KB_INT_1
F000:1A04	BECP	F000:0080	KEYBOARD_IO_1
F000:1A04	BECP	F000:0010	Abas
F000:1A04	BECP	F000:0080	M4
F000:1A04	BECP	F000:0010	M6
F000:1A04	BECP	F000:0080	M6
F000:1A04	BECP	F000:0010	M7
F000:1A04	BECP	F000:0080	MEMORY_SIZE_DET_1
F000:1A04	BECP	F000:0010	NMI_INT
F000:1A04	BECP	F000:0080	NMI_INT_1
F000:1A04	BECP	F000:0010	OBF_42
F000:1A04	BECP	F000:0080	POST1
F000:1A04	BECP	F000:0010	POST2
F000:1A04	BECP	F000:0080	POST3
F000:1A04	BECP	F000:0010	POST4
F000:1A04	BECP	F000:0080	POST5
F000:1A04	BECP	F000:0010	POST6
F000:1A04	BECP	F000:0080	PRINTER_IO_1
F000:1A04	BECP	F000:0010	PRINT_SCREEN
F000:1A04	BECP	F000:0080	PRINT_SCREEN_1
F000:1A04	BECP	F000:0010	PROC_SHUTDOWN
F000:1A04	BECP	F000:0080	PROT_PRT_HEX
F000:1A04	BECP	F000:0010	PRT_HEX
F000:1A04	BECP	F000:0080	PRT_SEG
F000:1A04	BECP	F000:0010	P_MSG
F000:1A04	BECP	F000:0080	P_O_R
F000:1A04	BECP	F000:0010	READ_AC_CURRENT
F000:1A04	BECP	F000:0080	READ_CURSOR
F000:1A04	BECP	F000:0010	READ_DOT
F000:1A04	BECP	F000:0080	READ_LPEN
F000:1A04	BECP	F000:0010	RE_DIRECT
F000:1A04	BECP	F000:0080	ROM_CHECK
F000:1A04	BECP	F000:0010	ROM_CHECKSUM
F000:1A04	BECP	F000:0080	ROM_ERR
F000:1A04	BECP	F000:0010	RS232_IO_1
F000:1A04	BECP	F000:0080	RTC_INT
F000:1A04	BECP	F000:0010	SCROLL_DOWN
F000:1A04	BECP	F000:0080	SCROLL_UP
F000:1A04	BECP	F000:0010	F000:0000
F000:1A04	BECP	F000:0080	F000:0008
F000:1A04	BECP	F000:0010	F000:0010
F000:1A04	BECP	F000:0080	F000:0080
F000:1A04	BECP	F000:0010	F000:0090
F000:1A04	BECP	F000:0080	F000:00A9
F000:1A04	BECP	F000:0010	F000:00CA3
F000:1A04	BECP	F000:0080	F000:00CA3
F000:1A04	BECP	F000:0010	F000:00E7
F000:1A04	BECP	F000:0080	F000:00F8
F000:1A04	BECP	F000:0010	F000:001620
F000:1A04	BECP	F000:0080	F000:00167F
F000:1A04	BECP	F000:0010	F000:00194F
F000:1A04	BECP	F000:0080	F000:00194F
F000:1A04	BECP	F000:0010	F000:00196B
F000:1A04	BECP	F000:0080	F000:001989
F000:1A04	BECP	F000:0010	F000:001991
F000:1A04	BECP	F000:0080	F000:0019B8
F000:1A04	BECP	F000:0010	F000:0019C6
F000:1A04	BECP	F000:0080	F000:001A04
F000:1A04	BECP	F000:0010	F000:001A4A
F000:1A04	BECP	F000:0080	F000:001A59
F000:1A04	BECP	F000:0010	F000:001A60
F000:1A04	BECP	F000:0080	F000:001A7D
F000:1A04	BECP	F000:0010	F000:001A84
F000:1A04	BECP	F000:0080	F000:001A99
F000:1A04	BECP	F000:0010	F000:001AC5
F000:1A04	BECP	F000:0080	F000:001AD1
F000:1A04	BECP	F000:0010	F000:001B03
F000:1A04	BECP	F000:0080	F000:001B2E
F000:1A04	BECP	F000:0010	F000:001B3C
F000:1A04	BECP	F000:0080	F000:001BE0
F000:1A04	BECP	F000:0010	F000:001C2E
F000:1A04	BECP	F000:0080	F000:001C2F
F000:1A04	BECP	F000:0010	F000:001C47
F000:1A04	BECP	F000:0080	F000:001C4E
F000:1A04	BECP	F000:0010	F000:001D40
F000:1A04	BECP	F000:0080	F000:001ECB
F000:1A04	BECP	F000:0010	F000:001ECB
F000:1A04	BECP	F000:0080	F000:001FCB
F000:1A04	BECP	F000:0010	F000:001FF7
F000:1A04	BECP	F000:0080	F000:002038
F000:1A04	BECP	F000:0010	F000:002169
F000:1A04	BECP	F000:0080	F000:002B1D
F000:1A04	BECP	F000:0010	F000:002C72
F000:1A04	BECP	F000:0080	F000:002C89
F000:1A04	BECP	F000:0010	F000:002CDD
F000:1A04	BECP	F000:0080	F000:002E86
F000:1A04	BECP	F000:0010	F000:0033B7
F000:1A04	BECP	F000:0080	F000:0033DA
F000:1A04	BECP	F000:0010	F000:00354F
F000:1A04	BECP	F000:0080	F000:0035AE
F000:1A04	BECP	F000:0010	F000:003A25
F000:1A04	BECP	F000:0080	F000:003ACD
F000:1A04	BECP	F000:0010	F000:003B4F
F000:1A04	BECP	F000:0080	F000:003C64
F000:1A04	BECP	F000:0010	F000:003C9B
F000:1A04	BECP	F000:0080	F000:003D62
F000:1A04	BECP	F000:0010	F000:003DA7
F000:1A04	BECP	F000:0080	F000:003DCF
F000:1A04	BECP	F000:0010	F000:003DE1
F000:1A04	BECP	F000:0080	F000:003E08
F000:1A04	BECP	F000:0010	F000:003E2E
F000:1A04	BECP	F000:0080	F000:003E4F
F000:1A04	BECP	F000:0010	F000:003EED
F000:1A04	BECP	F000:0080	F000:003F3F
F000:1A04	BECP	F000:0010	F000:003F9C
F000:1A04	BECP	F000:0080	F000:003FCE
F000:1A04	BECP	F000:0010	F000:00407F
F000:1A04	BECP	F000:0080	F000:004090
F000:1A04	BECP	F000:0010	F000:00433E
F000:1A04	BECP	F000:0080	F000:0043C5
F000:1A04	BECP	F000:0010	F000:00446B
F000:1A04	BECP	F000:0080	F000:004475
F000:1A04	BECP	F000:0010	F000:00447F
F000:1A04	BECP	F000:0080	F000:004503
F000:1A04	BECP	F000:0010	F000:004799
F000:1A04	BECP	F000:0080	F000:0048D0
F000:1A04	BECP	F000:0010	F000:004999
F000:1A04	BECP	F000:0080	F000:004B1B
F000:1A04	BECP	F000:0010	F000:004B97
F000:1A04	BECP	F000:0080	F000:004C2D
F000:1A04	BECP	F000:0010	F000:004C76
F000:1A04	BECP	F000:0080	F000:006000

F000:2B1D	SEEK	F000:E379	F1781
F000:FF62	SEEKS_I	F000:E38E	F1782
F000:3E08	SET_COLOR	F000:E3AC	F1790
F000:3DA7	SET_CPOS	F000:E3BF	F1791
F000:3D82	SET_CTYPE	F000:E3D2	F3A
F000:3C98	SET_MODE	F000:E3DF	F3D1
F000:1B3C	SET_TOD	F000:E401	FD_TBL
F000:10C3	SHUT2	F000:E6F6	CONF_TBL
F000:10BF	SHUT3	F000:E729	A1
F000:1620	SHUT4	F000:E87E	K6
F000:10E7	SHUT6	F000:E886	K7
F000:10C6	SHUT7	F000:E88E	K8
F000:4799	SHUT9	F000:E8E6	K10
F000:FF23	SLAVE_VECTOR_TABLE	F000:E92C	K15
F000:3A25	SND_DATA	F000:E98A	K11
F000:0050	START_I	F000:E9C4	K12
F000:1ECB	STGYST_CNT	F000:E9D0	K14
F000:1D40	SYSINIT1	F000:EFC7	DISK_BASE
F000:4C2D	TIMER_INT_I	F000:F0A4	VIDEO_PARMS
F000:4999	TIME_OF_DAY_I	F000:F0E4	M6
F000:FF64	TUTOR	F000:F0EC	M6
F000:FEF3	VECTOR_TABLE	F000:F0F4	M7
F000:3C64	VIDEO_TO_I	F000:FA6E	CRT_CHAR_GEN
F000:F0A4	VIDEO_PARMS	F000:FEF3	VECTOR_TABLE
F000:3E2E	VIDEO_STATE	F000:FF23	SLAVE_VECTOR_TABLE
F000:1A4A	WAIT	F000:FF53	DUMMY_RETURN
F000:3F9C	WRITE_AC_CURRENT	F000:FF54	PRINT_SCREEN
F000:3FC6	WRITE_C_CURRENT	F000:FF5A	HRD
F000:4090	WRITE_DOT	F000:FF5E	FLOPPY
F000:433E	WRITE_TTY	F000:FF62	SEEKS_I
F000:1FF7	XMIT_8042	F000:FF66	TUTOR
F000:1A6D	XPC_BYTE	F000:FFF0	P_U_R

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

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

PAGE 116,121
 TITLE TEST1 ---- 04/21/86 POWER ON SELF TEST (POST)
 .286C

```

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

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

```

-----
MODULE REFERENCE
-----
TEST1.ASM --> POST AND MANUFACTURING TEST ROUTINES
DSEG.INC --> DATA SEGMENTS LOCATIONS
POSTEQU.INC --> COMMON EQUATES FOR POST AND BIOS
SYSDATA.INC --> POWER ON SELF TEST EQUATES FOR PROTECTED MODE
POST TEST.01 THROUGH TEST.16
TEST2.ASM --> POST TEST AND INITIALIZATION ROUTINES
POST TEST.17 THROUGH TEST.22
TEST3.ASM --> POST EXCEPTION INTERRUPT TESTS
TEST4.ASM --> POST AND BIOS UTILITY ROUTINES
CMOS_READ - READ CMOS LOCATION ROUTINE
CMOS_WRITE - WRITE CMOS LOCATION ROUTINE
DDS - LOAD (DS:) WITH DATA SEGMENT
E_MSG - POST ERROR MESSAGE HANDLER
MFG_HALT - MANUFACTURING ERROR TRAP
P_MSG - POST STRING DISPLAY ROUTINE
ERR_BEEP - POST ERROR BEEP PROCEDURE
BEEP - SPEAKER BEEP CONTROL ROUTINE
WAITF - FIXED TIME WAIT ROUTINE
CONFIG_BAD - SET BAD CONFIG IN CMOS_DIAG
XPC_BYTE - DISPLAY HEX BYTE AS 00 - FF
PRT_HEX - DISPLAY CHARACTER
PRT_SEG - DISPLAY SEGMENT FORMAT ADDRESS
PROT_PRT_HEX - POST PROTECTED MODE DISPLAY
ROM_CHECKSUM - CHECK ROM MODULES FOR CHECKSUM
ROM_SCAN - ROM SCAN AND INITIALIZE
KBD_RESET - POST KEYBOARD RESET ROUTINE
BLINK_INT - MANUFACTURING TOGGLE BIT ROUTINE
SET_TOD - SET TIMER FROM CMOS RTC
DI1 - DUMMY INTERRUPT HANDLER ->INT 77H
RE_DIRECT - HARDWARE INT 9 REDIRECT (L 2)
INT_287 - HARDWARE INT 13 REDIRECT (287)
PROC_SHUTDOWN - 80286 RESET ROUTINE
TEST5.ASM --> EXCEPTION INTERRUPT TEST HANDLERS FOR POST TESTS
SYSINIT1 - BUILD PROTECTED MODE POINTERS
GDT_BLD - BUILD THE GDT FOR POST
SIDT_BLD - BUILD THE IDT FOR POST
TEST6.ASM --> POST TESTS AND SYSTEM BOOT STRAP
STGTS_CNT - SEGMENT STORAGE TEST
ROM_ERR - ROM ERROR DISPLAY ROUTINE
XMIT_8042 - KEYBOARD DIAGNOSTIC OUTPUT
BOOT_STRAP - BOOT STRAP LOADER -INT 19H
DSKETTE.ASM --> DISKETTE BIOS
DISKETTE_IO_1 - INT 13H BIOS ENTRY (40H) -INT 13H
DISK_INT_1 - HARDWARE INTERRUPT HANDLER -INT 0EH
DSKETTE_SETUP - POST SETUP DRIVE TYPES
DISK.ASM --> FIXED DISK BIOS
DISK_SETUP - SETUP DISK VECTORS AND TEST
DISK_IO - INT 13H BIOS ENTRY -INT 13H
HD_INT - HARDWARE INTERRUPT HANDLER -INT 76H
KYBD.ASM --> KEYBOARD BIOS
KEYBOARD_IO_1 - INT 16H BIOS ENTRY -INT 16H
KB_INT_1 - HARDWARE INTERRUPT -INT 09H
SIBS_DATA - KEYBOARD TRANSMISSION
PRT.ASM --> PRINTER ADAPTER BIOS
RS232.ASM --> COMMUNICATIONS BIOS FOR RS232 -INT 17H
VIDEO1.ASM --> VIDEO BIOS -INT 14H
BIOS.ASM --> BIOS ROUTINES
MEMORY_SIZE_DET_1 - REAL MODE SIZE -INT 12H
EQUIPMENT_1 - EQUIPMENT DETERMINATION -INT 11H
NMI_INT_1 - NMI HANDLER -INT 02H
BIOS1.ASM --> INTERRUPT 13H BIOS ROUTINES -INT 15H
DEV_OPEN - NULL DEVICE OPEN HANDLER
DEV_CLOSE - NULL DEVICE CLOSE HANDLER
PROC_TERM - NULL PROGRAM TERMINATION
EVENT_WAIT - RTC EVENT WAIT/TIMEOUT ROUTINE
JOY_STICK - JOYSTICK PORT HANDLER
SYS_REQ - NULL SYSTEM REQUEST KEY
WAIT - RTC TIMED WAIT ROUTINE
BLOCKMOVE - EXTENDED MEMORY MOVE INTERFACE
GATE_A20 - ADDRESS BIT 20 CONTROL
EXT_MEMORY - EXTENDED MEMORY SIZE DETERMINE
SET_MODE - SWITCH PROCESSOR TO VIRTUAL MODE
DEVTC_BUSY - NULL DEVICE BUSY HANDLER
INT_COMPLETE - NULL INTERRUPT COMPLETE HANDLER
BIOS2.ASM --> BIOS INTERRUPT ROUTINES
TIME_OF_DAY_1 - TIME OF DAY ROUTINES -INT 1AH
RTC_INT - IRQ LEVEL 8 ALARM HANDLER -INT 70H
PRINT_SCREEN1 - PRINT SCREEN ROUTINE -INT 05H
TIMER1_INT - TIMER1 INTERRUPT HANDLER ->INT 1CH
ORGS.ASM --> COMPATIBILITY MODULE
POST_ERROR_MESSAGES
DISKETTE - DISK - VIDEO DATA TABLES
-----
  
```

.LIST

```

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

SECTION 5

```

175 C PAGE
176 C |-----|
177 C | ROM BIOS DATA AREAS |
178 C |-----|
179 C DATA SEGMENT AT 40H | ADDRESS= 0040:0000
180 0000
181
182 0000 ???? #RS232_BASE DW ? | BASE ADDRESSES OF RS232 ADAPTERS
183 0002 ???? DW ? | SECOND LOGICAL RS232 ADAPTER
184 0004 ???? DW ? | RESERVED
185 0006 ???? DW ? | RESERVED
186 0008 ???? #PRINTER_BASE DW ? | BASE ADDRESSES OF PRINTER ADAPTERS
187 000A ???? DW ? | SECOND LOGICAL PRINTER ADAPTER
188 000C ???? DW ? | THIRD LOGICAL PRINTER ADAPTER
189 000E ???? DW ? | RESERVED
190 0010 ???? #EQUIP_FLAG DW ? | INSTALLED HARDWARE FLAGS
191 0012 ?? #MFG_TST DS ? | INITIALIZATION FLAGS
192 0013 ???? #MEMORY_SIZE DW ? | BASE MEMORY SIZE IN K BYTES (X 1024)
193 0015 ?? #MFG_ERR_FLAG DB ? | SCRATCHPAD FOR MANUFACTURING
194 0016 ?? DB ? | ERROR CODES
195
196 C |-----|
197 C | KEYBOARD DATA AREAS |
198 C |-----|
199
200 0017 ?? #KB_FLAG DB ? | KEYBOARD SHIFT STATE AND STATUS FLAGS
201 0018 ?? #KB_FLAG_1 DB ? | SECOND BYTE OF KEYBOARD STATUS
202 0019 ?? #ALT_INPUT DB ? | STORAGE FOR ALTERNATE KEY PAD ENTRY
203 001A ???? #BUFFER_HEAD DW ? | POINTER TO HEAD OF KEYBOARD BUFFER
204 001C ???? #BUFFER_TAIL DW ? | POINTER TO TAIL OF KEYBOARD BUFFER
205
206 |----- HEAD = TAIL INDICATES THAT THE BUFFER IS EMPTY
207
208 001E 10 [ #KB_BUFFER DW 16 DUP(?) | ROOM FOR 15 SCAN CODE ENTRIES
209 ]
210
211 C |-----|
212 C | DISKETTE DATA AREAS |
213 C |-----|
214
215
216 003E ?? #SEEK_STATUS DB ? | DRIVE RECALIBRATION STATUS
217 | BIT 3-0 = DRIVE 3-0 RECALIBRATION
218 | BEFORE NEXT SEEK IF BIT 15 = 0
219 003F ?? #MOTOR_STATUS DB ? | MOTOR STATUS
220 | BIT 3-0 = DRIVE 3-0 CURRENTLY RUNNING
221 | BIT 7 = CURRENT OPERATION IS A WRITE
222 0040 ?? #MOTOR_COUNT DB ? | TIME OUT COUNTER FOR MOTOR(S) TURN OFF
223 0041 ?? #DSKETTE_STATUS DB ? | RETURN CODE STATUS BYTE
224 | CMD_BLOCK IN STACK FOR DISK OPERATION
225 0042 07 [ #NEC_STATUS DB 7 DUP(?) | STATUS BYTES FROM DISKETTE OPERATION
226 ]
227
228 C |-----|
229 C | VIDEO DISPLAY DATA AREA |
230 C |-----|
231
232
233
234 0049 ?? #CRT_MODE DB ? | CURRENT DISPLAY MODE (TYPE)
235 004A ???? #CRT_COLS DW ? | NUMBER OF COLUMNS ON SCREEN
236 004C ???? #CRT_LEN DW ? | LENGTH OF REGEN BUFFER IN BYTES
237 004E ???? #CRT_START DW ? | STARTING ADDRESS IN REGEN BUFFER
238 0050 08 [ #CURSOR_POSN DW 8 DUP(?) | CURSOR FOR EACH OF UP TO 8 PAGES
239 ]
240
241
242 0060 ???? #CURSOR_MODE DW ? | CURRENT CURSOR MODE SETTING
243 0062 ?? #ACTIVE_PAGE DB ? | CURRENT PAGE BEING DISPLAYED
244 0063 ???? #ADDR_C845 DW ? | BASE ADDRESS FOR ACTIVE DISPLAY CARD
245 0065 ?? #CRT_MODE_SET DB ? | CURRENT SETTING OF THE 3x8 REGISTER
246 0066 ?? #CRT_PALETTE DB ? | CURRENT PALETTE SETTING - COLOR CARD
247
248 C |-----|
249 C | POST AND BIOS WORK DATA AREA |
250 C |-----|
251
252
253 0067 ???? #IO_ROM_INIT DW ? | STACK SAVE, etc.
254 0069 ???? #IO_ROM_SEG DW ? | POINTER TO ROM INITIALIZATION ROUTINE
255 006B ?? #INTR_FLAG DB ? | LENGTH OF REGEN BUFFER IN BYTES
256 | FLAG INDICATING AN INTERRUPT HAPPENED
257
258 C |-----|
259 C | TIMER DATA AREA |
260 C |-----|
261
262 006C ???? #TIMER_LOW DW ? | LOW WORD OF TIMER COUNT
263 006E ???? #TIMER_HIGH DW ? | HIGH WORD OF TIMER COUNT
264 0070 ?? #TIMER_OFL DB ? | TIMER HAS ROLLED OVER SINCE LAST READ
265
266 C |-----|
267 C | SYSTEM DATA AREA |
268 C |-----|
269
270 0071 ?? #BIOS_BREAK DB ? | BIT 7=1 IF BREAK KEY HAS BEEN PRESSED
271 0072 ???? #RESET_FLAG DW ? | WORD=1234H IF KEYBOARD RESET UNDERWAY
272
273 C |-----|
274 C | FIXED DISK DATA AREAS |
275 C |-----|
276
277 0074 ?? #DISK_STATUS1 DB ? | FIXED DISK STATUS
278 0075 ?? #HF_NUM DB ? | COUNT OF FIXED DISK DRIVES
279 0076 ?? #CONTROL_BYTE DB ? | HEAD CONTROL BYTE
280 0077 ?? #PORT_OFF DB ? | RESERVED (PORT OFFSET)

```

```

280 PAGE
281
282 -----
283 | TIME-OUT VARIABLES |
284 -----
285 0078 ?? @PRINT_TIM_OUT DB ? ; TIME OUT COUNTERS FOR PRINTER RESPONSE
286 0079 ?? DB ? ; SECOND LOGICAL PRINTER ADAPTER
287 007A ?? DB ? ; THIRD LOGICAL PRINTER ADAPTER
288 007B ?? DB ? ; RESERVED
289 007C ?? @RS232_TIM_OUT DB ? ; TIME OUT COUNTERS FOR RS232 RESPONSE
290 007D ?? DB ? ; SECOND LOGICAL RS232 ADAPTER
291 007E ?? DB ? ; RESERVED
292 007F ?? DB ? ; RESERVED
293
294 -----
295 | ADDITIONAL KEYBOARD DATA AREA |
296 -----
297
298
299 0080 ???? @BUFFER_START DW ? ; BUFFER LOCATION WITHIN SEGMENT 40H
300 0082 ???? @BUFFER_END DW ? ; OFFSET OF END OF BUFFER
301
302 -----
303 | EGA/PGA DISPLAY WORK AREA |
304 -----
305
306 0084 ?? @ROWS DB ? ; ROWS ON THE ACTIVE SCREEN (LESS 1)
307 0085 ???? @POINTS DW ? ; BITES PER CHARACTER
308 0087 ?? @INFO DB ? ; MODE OPTIONS
309 0088 ?? @INFO_3 DB ? ; FEATURE BIT SWITCHES
310 0089 ?? DB ? ; RESERVED FOR DISPLAY ADAPTERS
311 008A ?? DB ? ; RESERVED FOR DISPLAY ADAPTERS
312
313 -----
314 | ADDITIONAL MEDIA DATA |
315 -----
316
317 008B ?? @LASTRATE DB ? ; LAST DISKETTE DATA RATE SELECTED
318 008C ?? @P_STATUS DB ? ; STATUS REGISTER
319 008D ?? @P_ERROR DB ? ; ERROR REGISTER
320 008E ?? @P_INT_FLAG DB ? ; FIXED DISK INTERRUPT FLAG
321 008F ?? @P_CNTRL DB ? ; COMBO FIXED DISK/DISKETTE CARD BIT 0=1
322 0090 ?? @DSK_STATE DB ? ; DRIVE 0 MEDIA STATE
323 0091 ?? DB ? ; DRIVE 1 MEDIA STATE
324 0092 ?? DB ? ; DRIVE 0 OPERATION START STATE
325 0093 ?? DB ? ; DRIVE 1 OPERATION START STATE
326 0094 ?? @DSK_TRK DB ? ; DRIVE 0 PRESENT CYLINDER
327 0095 ?? DB ? ; DRIVE 1 PRESENT CYLINDER
328
329 -----
330 | ADDITIONAL KEYBOARD FLAGS |
331 -----
332
333 0096 ?? @KB_FLAG_3 DB ? ; KEYBOARD MODE STATE AND TYPE FLAGS
334 0097 ?? @KB_FLAG_2 DB ? ; KEYBOARD LED FLAGS
335
336 -----
337 | REAL TIME CLOCK DATA AREA |
338 -----
339
340 0098 ???? @USER_FLAG DW ? ; OFFSET ADDRESS OF USERS WAIT FLAG
341 009A ???? @USER_FLAG_SEG DW ? ; SEGMENT ADDRESS OF USER WAIT FLAG
342 009C ???? @RTC_LOW DW ? ; LOW WORD OF USER WAIT FLAG
343 009E ???? @RTC_HIGH DW ? ; HIGH WORD OF USER WAIT FLAG
344 00A0 ?? @RTC_WAIT_FLAG DB ? ; WAIT ACTIVE FLAG (01=BUSY, 80=POSTED)
345 ; (00=POST ACKNOWLEDGED)
346
347 -----
348 | AREA FOR NETWORK ADAPTER |
349 -----
350
351 00A1 07 [ ?? ] @NET DB ? DUP(?) ; RESERVED FOR NETWORK ADAPTERS
352
353 -----
354 | EGA/PGA PALETTE POINTER |
355 -----
356
357 00AB ?????????? @SAVE_PTR DD ? ; POINTER TO EGA PARAMETER CONTROL BLOCK
358 ; RESERVED
359
360 -----
361 | DATA AREA - PRINT SCREEN |
362 -----
363
364
365 0100 ORG 100H ; ADDRESS= 0040:0100 (REF 0050:0000)
366
367 0100 ?? @STATUS_BYTE DB ? ; PRINT SCREEN STATUS BYTE
368 ; 00=READY/OK, 01=BUSY, FF=ERROR
369
370 0101 DATA ENDS ; END OF BIOS DATA SEGMENT
371
372 .LIST
    
```

SECTION 5

```

373 PAGE
374 INCLUDE POSTEQU.INC
375
376 -----
377 EQUATES USED BY POST AND BIOS
378 -----
379
380 MODEL_BYTE EQU 0FCH ; SYSTEM MODEL BYTE
381 SUB_MODEL_BYTE EQU 002H ; SYSTEM SUB-MODEL TYPE
382 BIOS_LEVEL EQU 000H ; BIOS REVISION LEVEL
383 RATE_UPPER EQU 0F70H ; UPPER LIMIT + 13%
384 RATE_LOWER EQU 0F9FDH ; LOWER LIMIT - 20%
385
386 -----
387 8042 KEYBOARD INTERFACE AND DIAGNOSTIC CONTROL REGISTERS
388 -----
389 PORT_A EQU 060H ; 8042 KEYBOARD SCAN CODE/CONTROL PORT
390 PORT_B EQU 061H ; PORT B READ/WRITE DIAGNOSTIC REGISTER
391 RAM_PAR_ON EQU 11110011B ; AND MASK FOR PARITY CHECKING ENABLE ON
392 RAM_PAR_OFF EQU 00001100B ; OR MASK FOR PARITY CHECKING ENABLE OFF
393 PARTY_ERR EQU 11000000B ; R/W MEMORY - I/O CHANNEL PARITY ERROR
394 GATE2 EQU 00000010B ; TIMER 2 INPUT GATE CLOCK BIT
395 SPK2 EQU 00000010B ; SPEAKER OUTPUT DATA ENABLE BIT
396 REFRESH_BIT EQU 00100000B ; REFRESH TEST BIT
397 OUT2 EQU 00100000B ; SPEAKER TIMER OUT2 INPUT BIT
398 IO_CHECK EQU 01000000B ; I/O (MEMORY) CHECK OCCURRED BIT MASK
399 PARITY_CHECK EQU 10000000B ; MEMORY PARITY CHECK OCCURRED BIT MASK
400 STATUS_PORT EQU 064H ; 8042 STATUS PORT
401 OUT_BUF_FULL EQU 00000010B ; 0 = +OUTPUT BUFFER FULL
402 INPT_BUF_FULL EQU 00000010B ; 1 = +INPUT BUFFER FULL
403 SYS_FLAG EQU 00000100B ; 2 = -SYSTEM FLAG -POST/-SELF TEST
404 CMD_DATA EQU 00001000B ; 3 = -COMMAND/+DATA
405 KYBD_INH EQU 00100000B ; 4 = +KEYBOARD INHIBITED
406 TRANS_TMOUT EQU 00100000B ; 5 = +TRANSMIT TIMEOUT
407 RCY_TMOUT EQU 01000000B ; 6 = +RECEIVE TIME OUT
408 PARTY_EVEN EQU 10000000B ; 7 = +PARITY IS EVEN
409
410 -----
411 8042 INPUT PORT BIT DEFINITION SAVED IN 0MFG TST
412 -----
413 BASE_MEMB EQU 00001000B ; BASE PLANAR R/W MEMORY EXTENSION 640/X
414 BASE_MEMA EQU 00010000B ; BASE PLANAR R/W MEMORY SIZE 256/512
415 MFG_LOOP EQU 00100000B ; LOOP POST JUMPER BIT FOR MANUFACTURING
416 DSP JMP EQU 01000000B ; DISPLAY TYPE SWITCH JUMPER BIT
417 KEY_BD_INH1B EQU 10000000B ; KEYBOARD INHIBIT SWITCH BIT
418
419 -----
420 8042 COMMANDS
421 -----
422 WRITE_8042_LOC EQU 060H ; WRITE 8042 COMMAND BYTE
423 SELF_TEST EQU 0A4H ; 8042 SELF TEST
424 INTR_FACE_CK EQU 0ABH ; CHECK 8042 INTERFACE COMMAND
425 DIS_RBD EQU 0ADH ; DISABLE KEYBOARD COMMAND
426 ENA_KBD EQU 0AEH ; ENABLE KEYBOARD COMMAND
427 READ_8042_INPT EQU 020H ; READ 8042 INPUT PORT
428 DISABLE_BIT20 EQU 0DDH ; DISABLE ADDRESS LINE BIT 20
429 ENABLE_BIT20 EQU 0DFH ; ENABLE ADDRESS LINE BIT 20
430 KYBD_CLK_DATA EQU 0E0H ; GET KEYBOARD CLOCK AND DATA COMMAND
431 SHUT_CMD EQU 0F2H ; CAUSE A SHUTDOWN COMMAND
432 KYBD_CLK EQU 001H ; KEYBOARD CLOCK BIT 0
433
434 -----
435 KEYBOARD/LED COMMANDS
436 -----
437 KB_RESET EQU 0FFH ; SELF DIAGNOSTIC COMMAND
438 KB_RESEND EQU 0FEH ; RESEND COMMAND
439 KB_MAKE_BREAK EQU 0FAH ; TYPAMATIC COMMAND
440 KB_ENABLE EQU 0F4H ; KEYBOARD ENABLE
441 KB_TYPA_RD EQU 0F3H ; TYPAMATIC RATE/DELAY COMMAND
442 KB_READ_ID EQU 0F2H ; READ KEYBOARD ID COMMAND
443 KB_ECHO EQU 0EEH ; ECHO COMMAND
444 LED_CMD EQU 0EDH ; LED WRITE COMMAND
445
446 -----
447 8042 KEYBOARD RESPONSE
448 -----
449 KB_OVER_RUN EQU 0FFH ; OVER RUN SCAN CODE
450 KB_RESEND EQU 0FEH ; RESEND REQUEST
451 KB_ACK EQU 0FAH ; ACKNOWLEDGE FROM TRANSMISSION
452 KB_BREAK EQU 0FDH ; KEYBOARD BREAK CODE
453 KB_OK EQU 0AAH ; RESPONSE FROM SELF DIAGNOSTIC
454
455 -----
456 KEYBOARD SCAN CODES
457 -----
458 NUM_KEY EQU 69 ; SCAN CODE FOR NUMBER LOCK KEY
459 SCROLL_KEY EQU 70 ; SCAN CODE FOR SCROLL LOCK KEY
460 ALT_KEY EQU 56 ; SCAN CODE FOR ALTERNATE SHIFT KEY
461 CTL_KEY EQU 29 ; SCAN CODE FOR CONTROL KEY
462 CAPS_KEY EQU 58 ; SCAN CODE FOR SHIFT LOCK KEY
463 DEL_KEY EQU 83 ; SCAN CODE FOR DELETE KEY
464 INS_KEY EQU 82 ; SCAN CODE FOR INSERT KEY
465 LEFT_KEY EQU 42 ; SCAN CODE FOR LEFT SHIFT
466 RIGHT_KEY EQU 54 ; SCAN CODE FOR RIGHT SHIFT
467 SYS_KEY EQU 84 ; SCAN CODE FOR SYSTEM KEY
468
469 -----
470 ENHANCED KEYBOARD SCAN CODES
471 -----
472 ID_1 EQU 0ABH ; 1ST ID CHARACTER FOR KBX
473 ID_2 EQU 041H ; 2ND ID CHARACTER FOR KBX
474 ID_2A EQU 054H ; ALTERNATE 2ND ID CHAR FOR KBX
475 F11_M EQU 87 ; F11 KEY MAKE
476 F12_M EQU 88 ; F12 KEY MAKE
477 MC_E0 EQU 224 ; GENERAL MARKER CODE
478 MC_E1 EQU 225 ; PAUSE KEY MARKER CODE

```

```

464 C PAGE
465 C |----- FLAG EQUATES WITHIN #KB_FLAG -----
466 = 0001 C RIGHT_SHIFT EQU 0000001B ; RIGHT SHIFT KEY DEPRESSED
467 = 0002 C LEFT_SHIFT EQU 0000010B ; LEFT SHIFT KEY DEPRESSED
468 = 0004 C CTL_SHIFT EQU 0000100B ; CONTROL SHIFT KEY DEPRESSED
469 = 0008 C ALT_SHIFT EQU 00001000B ; ALTERNATE SHIFT KEY DEPRESSED
470 = 0010 C SCROLL_STATE EQU 00100000B ; SCROLL LOCK STATE IS ACTIVE
471 = 0020 C NUM_STATE EQU 00100000B ; NUM LOCK STATE IS ACTIVE
472 = 0040 C CAPS_STATE EQU 01000000B ; CAPS LOCK STATE IS ACTIVE
473 = 0080 C INS_STATE EQU 10000000B ; INSERT STATE IS ACTIVE
474 C
475 C |----- FLAG EQUATES WITHIN #KB_FLAG |-----
476 = 0001 C L_CTL_SHIFT EQU 0000001B ; LEFT CTL KEY DOWN
477 = 0002 C L_ALT_SHIFT EQU 0000010B ; LEFT ALT KEY DOWN
478 = 0004 C SYS_SHIFT EQU 00000100B ; SYSTEM KEY DEPRESSED AND HELD
479 = 0008 C HOLD_STATE EQU 00001000B ; SUSPEND KEY HAS BEEN TOGGLED
480 = 0010 C SCROLL_SHIFT EQU 00010000B ; SCROLL LOCK KEY IS DEPRESSED
481 = 0020 C NUM_SHIFT EQU 00100000B ; NUM LOCK KEY IS DEPRESSED
482 = 0040 C CAPS_SHIFT EQU 01000000B ; CAPS LOCK KEY IS DEPRESSED
483 = 0080 C INS_SHIFT EQU 10000000B ; INSERT KEY IS DEPRESSED
484 C
485 C |----- FLAGS EQUATES WITHIN #KB_FLAG 2 -----
486 = 0007 C KB_LEDS EQU 00000111B ; KEYBOARD LED STATE BITS
487 C ; 0000001B ; SCROLL LOCK INDICATOR
488 C ; 00000010B ; NUM LOCK INDICATOR
489 C ; 00000100B ; CAPS LOCK INDICATOR
490 C ; 00001000B ; RESERVED (MUST BE ZERO)
491 = 0010 C KB_FA EQU 00010000B ; ACKNOWLEDGMENT RECEIVED
492 = 0020 C KB_FE EQU 00100000B ; RESEND RECEIVED FLAG
493 = 0040 C KB_FR_LED EQU 01000000B ; MODE INDICATOR UPDATE
494 = 0080 C KB_ERR EQU 10000000B ; KEYBOARD TRANSMIT ERROR FLAG
495 C
496 C |----- FLAGS EQUATES WITHIN #KB_FLAG 3 -----
497 = 0001 C LC_E1 EQU 0000001B ; LAST CODE WAS THE E1 HIDDEN CODE
498 = 0002 C LC_E0 EQU 0000010B ; LAST CODE WAS THE E0 HIDDEN CODE
499 = 0004 C R_CTL_SHIFT EQU 00000100B ; RIGHT CTL KEY DOWN
500 = 0008 C R_ALT_SHIFT EQU 00001000B ; RIGHT ALT KEY DOWN
501 = 0008 C GRAPH_ON EQU 00001000B ; ALT GRAPHICS KEY DOWN (WT ONLY)
502 = 0010 C KBX EQU 00010000B ; ENHANCED KEYBOARD INSTALLED
503 = 0020 C SET_NUM_LK EQU 00100000B ; FORCE NUM LOCK IF READ ID AND KBX
504 = 0040 C LC_AB EQU 01000000B ; LAST CHARACTER WAS FIRST ID CHARACTER
505 = 0080 C RD_ID EQU 10000000B ; DOING A READ ID (MUST BE BIT0)
    
```

SECTION 5

```

506
507
508
509
510 = 0070
511 = 0071
512 = 0080
513
514
515
516 = 0000
517 = 0001
518 = 0002
519 = 0003
520 = 0004
521 = 0005
522 = 0006
523 = 0007
524 = 0008
525 = 0009
526 = 000A
527 = 000B
528 = 000C
529 = 000D
530 = 000E
531 = 000F
532 = 0010
533
534 = 0012
535
536 = 0014
537 = 0015
538 = 0016
539 = 0017
540 = 0018
541 = 0019
542 = 001A
543
544 = 002E
545 = 002F
546 = 0030
547 = 0031
548 = 0032
549 = 0033
550
551
552
553 = 0004
554 = 0005
555 = 0010
556 = 0020
557 = 0040
558 = 0080
559
560
561 = 0080
562
563
564
565
566 = 0001
567 = 0080
568 = 0080
569 = 0010
570 = 0000
571 = 0004
572 = 0030
573 = 000A
574 = 0002
575 = 000F
576 = 0014
577 = 0025
578
579
580 = 0080
581 = 0040
582 = 0020
583 = 0010
584 = 000C
585 = 0009
586 = 0008
587 = 0006
588 = 0004
589 = 0003
590 = 0002
591 = 0001
592
593
594 = 0001
595 = 0002
596
597
598 = 0001
599 = 0002
600 = 0004
601 = 0010
602 = 0020
603 = 00C0
604 = 0000
605 = 0040
606 = 0080
607 = 000C
608 = 00C0
609
610
611 = 0000
612 = 0001
613 = 0002
614 = 0007

C PAGE
C -----
C |
C |          CMOS EQUATES FOR THIS SYSTEM
C |-----
C CMOS_PORT EQU 070H          ; I/O ADDRESS OF CMOS ADDRESS PORT
C CMOS_DATA EQU 071H          ; I/O ADDRESS OF CMOS DATA PORT
C NMI EQU 1000000B           ; DISABLE NMI INTERRUPTS MASK
C                               ; HIGH BIT OF CMOS LOCATION ADDRESS
C
C |----- CMOS TABLE LOCATION ADDRESS'S ## -----
C CMOS_SECONDS EQU 000H      ; SECONDS
C CMOS_SEC_ALARM EQU 001H    ; SECONDS ALARM ## NOTE: ALL LOCATIONS
C CMOS_MINUTES EQU 002H      ; MINUTES          IN THE CMOS AREA
C CMOS_MIN_ALARM EQU 003H    ; MINUTES ALARM   AND SUBJECT TO
C CMOS_HOURS EQU 004H        ; HOURS           ARE IBM USE ONLY
C CMOS_HR_ALARM EQU 005H     ; HOURS ALARM     CHANGE, ONLY THE
C CMOS_DAY_WEEK EQU 006H     ; DAY OF THE WEEK POST & BIOS CODE
C CMOS_DAY_MONTH EQU 007H    ; DAY OF THE MONTH SHOULD DIRECTLY
C CMOS_YEAR EQU 008H         ; MONTH           ACCESS LOCATIONS
C CMOS_REC_A EQU 009H        ; YEAR (TWO DIGITS) IN CMOS STORAGE.
C CMOS_REC_B EQU 00AH        ; STATUS REGISTER A
C CMOS_REC_C EQU 00BH        ; STATUS REGISTER B ALARM
C CMOS_REC_D EQU 00CH        ; STATUS REGISTER C FLAGS
C CMOS_REG_D EQU 00DH        ; STATUS REGISTER D BATTERY
C CMOS_DIAG EQU 00EH        ; POST DIAGNOSTIC STATUS RESULTS BYTE
C CMOS_SHUT_DOWN EQU 00FH    ; SHUTDOWN STATUS COMMAND BYTE
C CMOS_DISKETTE EQU 010H     ; DISKETTE DRIVE TYPE BYTE
C CMOS_DISK EQU 011H        ; - RESERVED
C CMOS_EQUIP EQU 012H        ; FIXED DISK TYPE BYTE
C CMOS_B_M_S_LO EQU 013H     ; EQUIPMENT WORD LOW BYTE
C CMOS_B_M_S_HI EQU 014H     ; BASE MEMORY SIZE - LOW BYTE (X1024)
C CMOS_E_M_S_LO EQU 015H     ; BASE MEMORY SIZE - HIGH BYTE
C CMOS_E_M_S_HI EQU 016H     ; EXPANSION MEMORY SIZE - LOW BYTE
C CMOS_DISK_1 EQU 017H       ; EXPANSION MEMORY SIZE - HIGH BYTE
C CMOS_DISK_2 EQU 018H       ; FIXED DISK TYPE - DRIVE C EXTENSION
C CMOS_CHKSUM_HI EQU 019H    ; FIXED DISK TYPE - DRIVE D EXTENSION
C CMOS_CHKSUM_LO EQU 01AH    ; - 1BH THROUGH 20H - RESERVED
C CMOS_U_M_S_HI EQU 01BH     ; CMOS CHECKSUM - HIGH BYTE
C CMOS_U_M_S_LO EQU 01CH    ; CMOS CHECKSUM - LOW BYTE
C CMOS_USABLE_MEM_HI EQU 01DH ; USABLE MEMORY ABOVE 1 MEG - LOW BYTE
C CMOS_USABLE_MEM_LO EQU 01EH ; USABLE MEMORY ABOVE 1 MEG - HIGH BYTE
C CMOS_DATE_CENTURY EQU 01FH ; DATE CENTURY BYTE (BCD)
C CMOS_INFO128 EQU 020H     ; 128KB INFORMATION STATUS FLAG BYTE
C                               ; - 34H THROUGH 3FH - RESERVED
C
C |----- CMOS DIAGNOSTIC STATUS ERROR FLAGS WITHIN CMOS DIAG -----
C CMOS_CLK_FAIL EQU 00000100B ; CMOS CLOCK NOT UPDATING OR NOT VALID
C HF_FAIL EQU 00001000B      ; FIXED DISK FAILURE ON INITIALIZATION
C W_MEM_SIZE EQU 00010000B   ; MEMORY SIZE NOT EQUAL TO CONFIGURATION
C BAD_CONFIG EQU 00100000B   ; MINIMUM CONFIG USED INSTEAD OF CMOS
C BAD_CHKSUM EQU 01000000B   ; CHECKSUM ERROR
C BAD_BAT EQU 10000000B      ; DEAD BATTERY - CMOS LOST POWER
C
C |----- CMOS INFORMATION FLAGS -----
C M640K EQU 10000000B        ; 512K -> 640K OPTION INSTALLED (128K)
C                               ; FLAG USED BY CMOS SETUP UTILITY
C
C |----- DISKETTE EQUATES -----
C DUAL EQU 00000001B         ; MASK FOR COMBO/DSP ADAPTER
C INT_FLAG EQU 10000000B    ; INTERRUPT OCCURRENCE FLAG
C DSK_CHG EQU 00000000B     ; DISKETTE CHANGE FLAG MASK BIT
C DETERMINED EQU 00010000B  ; SET STATE DETERMINED IN STATE BITS
C HOME EQU 00010000B        ; TRACK 0 MASK
C SENSE_DRV_ST EQU 00001000B ; SENSE DRIVE STATUS COMMAND
C TRK_SLAP EQU 030H         ; CRASH STOP (48 TPI DRIVES)
C QUIET_SEEK EQU 00AH       ; SEEK TO TRACK 10
C MAX_DRV EQU 2             ; MAX NUMBER OF DRIVES
C HD320_SETTLE EQU 15       ; 1.2 M HEAD SETTLE TIME
C HD320_WAIT EQU 20         ; 320 K HEAD SETTLE TIME
C MOTOR_WAIT EQU 37         ; 2 SECONDS OF COUNTS FOR MOTOR TURN OFF
C
C |----- DISKETTE ERRORS -----
C TIME_OUT EQU 080H         ; ATTACHMENT FAILED TO RESPOND
C BAD_SEEK EQU 040H         ; SEEK OPERATION FAILED
C BAD_CRC EQU 020H         ; DISKETTE CONTROLLER HAS FAILED
C BAD_CMD EQU 010H         ; BAD CRC ON DISKETTE READ
C MED_NOT_FND EQU 00CH      ; MEDIA TYPE NOT FOUND
C DMA_BOUNDARY EQU 009H     ; ATTEMPT TO DMA ACROSS 64K BOUNDARY
C MEDIA_CHANGE EQU 008H     ; DMA OVERRUN ON OPERATION
C RECORD_NOT_FND EQU 006H   ; MEDIA REMOVED ON DUAL ATTACH CARD
C WRITE_PROTECT EQU 004H    ; REQUESTED SECTOR NOT FOUND
C BAD_ADDR_MARK EQU 003H    ; WRITE ATTEMPTED ON WRITE PROTECT DISK
C BAD_CMD EQU 002H         ; ADDRESS MARK NOT FOUND
C BAD_CMD EQU 001H         ; BAD COMMAND PASSED TO DISKETTE I/O
C
C |----- DISK CHANGE LINE EQUATES -----
C NOCHGLN EQU 001H         ; NO DISK CHANGE LINE AVAILABLE
C CHGLN EQU 002H          ; DISK CHANGE LINE AVAILABLE
C
C |----- MEDIA/DRIVE STATE INDICATORS -----
C TRK_CAPA EQU 00000010B    ; 80 TRACK CAPABILITY
C FMT_CAPA EQU 00000010B    ; MULTIPLE FORMAT CAPABILITY (1.2M)
C DRV_DET EQU 00000100B    ; DRIVE DETERMINED
C MED_DET EQU 00010000B    ; MEDIA DETERMINED BIT
C DBL_STEP EQU 00100000B   ; DOUBLE STEP BIT
C RATE_MSK EQU 11000000B   ; MASK FOR CLEARING ALL BUT RATE
C RATE_500 EQU 00000000B   ; 500 KBS DATA RATE
C RATE_300 EQU 01000000B   ; 300 KBS DATA RATE
C RATE_250 EQU 10000000B   ; 250 KBS DATA RATE
C STRT_MSK EQU 00001100B   ; OPERATION START RATE MASK
C SEND_MSK EQU 11000000B   ; MASK FOR SEND RATE BITS
C
C |----- MEDIA/DRIVE STATE INDICATORS COMPATIBILITY -----
C M320U EQU 00000000B      ; 360 MEDIA/DRIVE NOT ESTABLISHED
C M321U EQU 00000001B      ; 360 MEDIA/DRIVE NOT ESTABLISHED
C M121U EQU 00000010B      ; 1.2 MEDIA/DRIVE NOT ESTABLISHED
C MED_UNK EQU 00000111B    ; NONE OF THE ABOVE
    
```

```

615 C PAGE
616 C |----- INTERRUPT EQUATES -----|
617 = 0020 C EQ1 EQU 020H ; END OF INTERRUPT COMMAND TO 8259
618 = 0020 C INTA00 EQU 020H ; 8259 PORT
619 = 0021 C INTA01 EQU 021H ; 8259 PORT
620 = 00A0 C INTB00 EQU 0A0H ; 2ND 8259
621 = 00A1 C INTB01 EQU 0A1H ;
622 = 0070 C INT_TYPE EQU 070H ; START OF 8259 INTERRUPT TABLE LOCATION
623 = 0010 C INT_VIDEO EQU 010H ; VIDEO VECTOR
624 C |-----|
625 = 0008 C DMA08 EQU 008H ; DMA STATUS REGISTER PORT ADDRESS
626 = 0000 C DMA EQU 000H ; DMA CH.0 ADDRESS REGISTER PORT ADDRESS
627 = 0000 C DMA18 EQU 000H ; 2ND DMA STATUS PORT ADDRESS
628 = 00C0 C DMA1 EQU 0C0H ; 2ND DMA CH.0 ADDRESS REGISTER ADDRESS
629 C |-----|
630 = 0040 C TIMER EQU 040H ; 8254 TIMER - BASE ADDRESS
631 C |-----|
632 C |----- MANUFACTURING PORT -----|
633 = 0080 C MFG_PORT EQU 80H ; MANUFACTURING AND POST CHECKPOINT PORT
634 ; DMA CHANNEL 0 PAGE REGISTER ADDRESS
635 C |-----|
636 C |----- MANUFACTURING BIT DEFINITION FOR #MFG_ERR_FLAG+1 -----|
637 = 0001 C MEM_FAIL EQU 00000001B ; STORAGE TEST FAILED (ERROR 20X)
638 = 0002 C PRD_FAIL EQU 00000010B ; VIRTUAL MODE TEST FAILED (ERROR 104)
639 = 0004 C LMCS_FAIL EQU 00000100B ; LOW MEG CHIP SELECT FAILED (ERROR 109)
640 = 0008 C KYCLK_FAIL EQU 00001000B ; KEYBOARD CLOCK TEST FAILED (ERROR 304)
641 = 0010 C KY_SYS_FAIL EQU 00010000B ; KEYBOARD OR SYSTEM TEST FAILED (ERROR 303)
642 = 0020 C KYBD_FAIL EQU 00100000B ; KEYBOARD FAILED (ERROR 301)
643 = 0040 C DSK_FAIL EQU 01000000B ; DISKETTE TEST FAILED (ERROR 401)
644 = 0080 C KEY_FAIL EQU 10000000B ; KEYBOARD LOCKED (ERROR 302)
645 C |-----|
646 C |-----|
647 = 0081 C DMA_PAGE EQU 081H ; START OF DMA PAGE REGISTERS
648 = 008F C LAST_DMA_PAGE EQU 08FH ; LAST DMA PAGE REGISTER
649 C |-----|
650 C |-----|
651 = 00F0 C X287 EQU 0F0H ; MATH COPROCESSOR CONTROL PORT
652 C |-----|
653 C |-----|
654 = 8000 C POST_SS EQU 8000H ; POST STACK SEGMENT
655 = 8000 C POST_SP EQU 8000H ; POST STACK POINTER
656 C |-----|
657 C |-----|
658 = 000D C CR EQU 000DH ; CARRIAGE RETURN CHARACTER
659 = 000A C LF EQU 000AH ; LINE FEED CHARACTER
660 = 0008 C RVRT EQU 00001000B ; VIDEO VERTICAL RETRACE BIT
661 = 0001 C RHRZ EQU 00000001B ; VIDEO HORIZONTAL RETRACE BIT
662 = 0100 C H EQU 256 ; HIGH BYTE FACTOR (X 100H)
663 = 0101 C X EQU H+1 ; HIGH AND LOW BYTE FACTOR (X 101H)
664
665 .LIST
    
```

SECTION 5

```

666 PAGE
667 INCLUDE SYSDATA.INC
668
669 -----
670 |----- PROTECTED MODE EQUATES FOR POST TESTS AND BIOS ROUTINES
671 |-----
672 |----- LENGTH EQUATES FOR PROTECTED MODE TESTS
673
674 = 0300 SDA_LEN EQU 00300H ; SYSTEM DATA AREA LENGTH
675 = 0800 SYS_IDT_LEN EQU 256*8 ; 256 SYSTEM IDT ENTRIES, 8 BYTES EACH
676 = 0008 GDT_LEN EQU TYPE GDT_DEF ; GDT STRUCTURE LENGTH
677 = 0008 DESC_LEN EQU TYPE DATA_DESC ; LENGTH OF A DESCRIPTOR
678 = 1000 MCRT_SIZE EQU 4*1024 ; MONOCHROME CRT SIZE
679 = 4000 CCRT_SIZE EQU 16*1024 ; COMPATIBLE COLOR CRT SIZE
680 = FFFF ECRT_SIZE EQU 0FFFFH ; SIZE OF EACH PORTION OF THE ENHANCED
681 = FFFF MAX_SEG_LEN EQU 0FFFFH ; MAXIMUM SEGMENT LENGTH = 64K
682 = 0000 NULL_SEG_LEN EQU 00000H ; NULL SEGMENT LENGTH = 0
683
684 |----- LOCATION EQUATES FOR PROTECTED MODE TESTS
685
686 = 00A0 SYS_IDT_LOC EQU 00A00H ; THE SYSTEM IDT IS AT THE BOTTOM
687 = 0400 SDA_LOC EQU 00400H ; SAME AS REAL
688 = 08A0 GDT_LOC EQU (SYS_IDT_LOC + SYS_IDT_LEN)
689 = 0000 MCRT_LO EQU 0000H ; MONOCHROME CRT ADDRESS
690 = 000B MCRT_HI EQU 000BH ; (000000H)
691 = 8000 CCRT_LO EQU 8000H ; COMPATIBLE COLOR CRT ADDRESS
692 = 000B CCRT_HI EQU 000BH ; (008000H)
693 = 0000 ECRTS_LO_LO EQU 0000H
694 = 000A ECRTS_LO_HI EQU 000AH ; (0A0000H)
695 = 0000 ECRTS_HI_LO EQU 0000H
696 = 000B ECRTS_HI_HI EQU 000BH ; (0B0000H)
697 = 0000 CSEG_LO EQU 0000H ; CODE SEGMENT POST/BIOS
698 = 000F CSEG_HI EQU 000FH ; (0F0000H) FOR TESTS
699 = 0000 NSEG_LO EQU 0000H ; ABS0
700 = 0000 NSEG_HI EQU 000H
701
702 |----- DEFINITIONS FOR ACCESS RIGHTS BYTES
703
704 = 00F3 CPL3_DATA_ACCESS EQU 11110011B ; PRESENT
705 ; DPL = 3
706 ; CODE/DATA SEGMENT
707 ; NOT EXECUTABLE
708 ; GROW-UP (OFFSET <= LIMIT)
709 ; WRITABLE
710 ; ACCESSED
711 = 0093 CPL0_DATA_ACCESS EQU 10010011B ; DPL = 0
712 = 009B CPL0_CODE_ACCESS EQU 10011011B ; CPL 0 - NON-CONFORMING
713 = 00E2 LDT_DESC EQU 11100010B
714 = 0081 FRIB_TSS EQU 10000001B
715 = 0086 INT_GATE EQU 10000110B
716 = 0087 TRAP_GATE EQU 10000111B
717
718 = 0001 VIRTUAL_ENABLE EQU 0000000000000001B ; PROTECTED MODE ENABLE
719
720 |----- THE GLOBAL DESCRIPTOR TABLE DEFINITION FOR POWER ON SELF TESTS
721
722 GDT_DEF STRUC
723 0000 ?????????????????? GDT_PTR DQ ? ; UNUSED ENTRY
724 0008 ?????????????????? SYS_IDT_PTR DQ ? ; THIS ENTRY POINTS TO THIS TABLE
725 0018 ?????????????????? RSDA_PTR DQ ? ; POST INTERRUPT DESCRIPTOR TABLE
726 0018 ?????????????????? C_BWCRT_PTR DQ ? ; THE REAL SYSTEM DATA AREA FOR POST
727 0020 ?????????????????? C_CCRT_PTR DQ ? ; COMPATIBLE BW CRT FOR POST
728 0028 ?????????????????? E_CCRT_PTR DQ ? ; COMPATIBLE COLOR CRT FOR POST
729 0030 ?????????????????? E_CCRT_PTR2 DQ ? ; ENHANCED COLOR GRAPHICS CRT (16 BYTES)
730 0038 ?????????????????? SYS_ROM_CS DQ ? ; CS - POST IDT, ROM RESIDENT
731 0040 ?????????????????? ES_TEMP DQ ? ; DYNAMIC POINTER FOR ES
732 0048 ?????????????????? CS_TEMP DQ ? ; DYNAMIC POINTER FOR CS
733 0050 ?????????????????? SS_TEMP DQ ? ; DYNAMIC POINTER FOR SS
734 0058 ?????????????????? DS_TEMP DQ ? ; DYNAMIC POINTER FOR DS
735 0060 ?????????????????? POST_TR DQ ? ; TR VALUE FOR THIS MACHINE'S TSS
736 0068 ?????????????????? POST_TSS_PTR DQ ?
737 0070 ?????????????????? POST_LDR DQ ? ; LDR VALUE FOR THIS MACHINE'S LDT
738 0078 ?????????????????? POST_IDT_PTR DQ ?
739 0080 ?????????????????? GDT_DEF ENDS
740 0088
741
742 |----- SEGMENT DESCRIPTOR TABLE ENTRY STRUCTURE
743
744 DATA_DESC STRUC
745 0000 7777 SEG_LIMIT DW ? ; SEGMENT LIMIT (1 - 65536 BYTES)
746 0002 7777 BASE_LO_WORD DW ? ; 24 BIT SEGMENT PHYSICAL
747 0004 ?? BASE_HI_BYTE DB ? ; ADDRESS (0 - (16M-1))
748 0005 ?? DATA_ACC_RIGHTS DB ? ; ACCESS RIGHTS BYTE
749 0006 7777 DATA_RESERVED DW ? ; RESERVED - MUST BE 0000 FOR THE 80286
750 0008 DATA_DESC ENDS
751
752 |----- GATE DESCRIPTOR TABLE ENTRY STRUCTURE
753
754 GATE_DESC STRUC
755 0000 7777 ENTRY_POINT DW ? ; DESTINATION ROUTINE ENTRY POINT
756 0002 7777 CS_SELECTOR DW ? ; SELECTOR FOR DESTINATION SEGMENT
757 0004 ?? WORD_COUNT DB ? ; NUMBER OF WORDS TO COPY FROM STACK
758 0005 ?? GATE_ACC_RIGHTS DB ? ; ACCESS RIGHTS BYTE
759 0006 7777 GATE_RESERVED DW ? ; RESERVED - MUST BE 0000 FOR THE 80286
760 0008 GATE_DESC ENDS
761
762 .LIST
763

```

```

764          PAGE
765 0000      CODE
766          SEGMENT WORD PUBLIC
767          PUBLIC C8042
768          PUBLIC OBF_42
769          PUBLIC POST1
770          PUBLIC START_1
771
772          EXTRN CMOS_READ:NEAR
773          EXTRN CMOS_WRITE:NEAR
774          EXTRN CONFTG_BAD:NEAR
775          EXTRN D11:NEAR
776          EXTRN DDS:NEAR
777          EXTRN DUMMY_RETURN:NEAR
778          EXTRN ERR_BEEP:NEAR
779          EXTRN GATE_A20:NEAR
780          EXTRN KBD_RESET:NEAR
781          EXTRN NMI_INT:NEAR
782          EXTRN POST2:NEAR
783          EXTRN PRINT_SCREEN:NEAR
784          EXTRN PROC_SHUTDOWN:NEAR
785          EXTRN ROM_CHECK:NEAR
786          EXTRN SHUT2:NEAR
787          EXTRN SHUT3:NEAR
788          EXTRN SHUT4:NEAR
789          EXTRN SHUT6:NEAR
790          EXTRN SHUT7:NEAR
791          EXTRN SHUT9:NEAR
792          EXTRN SLAVE_VECTOR_TABLE:NEAR
793          EXTRN STGTST_CNT:NEAR
794          EXTRN SYSINIT:NEAR
795          EXTRN VECTOR_TABLE:NEAR
796          EXTRN VIDEO_PARMS:BYTE
797
798          ASSUME CS:CODE,DS:NOTHING,ES:NOTHING,SS:NOTHING
800 0000      POST1 PROC NEAR
801
802          BEGIN EQU $
803          DB '78X7462COPR. IBM CORP. 1981,1986 ' ;COPYRIGHT NOTICE
804          DB '
805          DB '
806          DB '
807          DB '
808          DB '
809          DB '
810          DB '
811          DB '
812          DB '
813          DB '
814          DB '
815          DB '
816          DB '
817          DB '
818          DB '
819          DB '
820          DB '
821          DB '
822          DB '
823          DB '
824          DB '
825          DB '
826          DB '
827          DB '
828          DB '
829          DB '
830          DB '
831          DB '
832          DB '
833          DB '
834          DB '
835          DB '
836          DB '
837          DB '
838          DB '
839          DB '
840          DB '
841          DB '
842          DB '
843          DB '
844          DB '
845          DB '
846          DB '
847          DB '
848          DB '
849          DB '
850          DB '
851          DB '
852          DB '
853          DB '
854          DB '
855          DB '
856          DB '
857          DB '
858          DB '
859          DB '
860          DB '
861          DB '
862          DB '
863          DB '
864          DB '
865          DB '
866          DB '
867          DB '
868          DB '
869          DB '
870          DB '
871          DB '
872          DB '
873          DB '
874          DB '
875          DB '
876          DB '
877          DB '
878          DB '
879          DB '
880          DB '
881          DB '
882          DB '
883          DB '
884          DB '
885          DB '
886          DB '
887          DB '
888          DB '
889          DB '
890          DB '
891          DB '
892          DB '
893          DB '
894          DB '
895          DB '
896          DB '
897          DB '
898          DB '
899          DB '
900          DB '
901          DB '
902          DB '
903          DB '
904          DB '
905          DB '
906          DB '
907          DB '
908          DB '
909          DB '
910          DB '
911          DB '
912          DB '
913          DB '
914          DB '
915          DB '
916          DB '
917          DB '
918          DB '
919          DB '
920          DB '
921          DB '
922          DB '
923          DB '
924          DB '
925          DB '
926          DB '
927          DB '
928          DB '
929          DB '
930          DB '
931          DB '
932          DB '
933          DB '
934          DB '
935          DB '
936          DB '
937          DB '
938          DB '
939          DB '
940          DB '
941          DB '
942          DB '
943          DB '
944          DB '
945          DB '
946          DB '
947          DB '
948          DB '
949          DB '
950          DB '
951          DB '
952          DB '
953          DB '
954          DB '
955          DB '
956          DB '
957          DB '
958          DB '
959          DB '
960          DB '
961          DB '
962          DB '
963          DB '
964          DB '
965          DB '
966          DB '
967          DB '
968          DB '
969          DB '
970          DB '
971          DB '
972          DB '
973          DB '
974          DB '
975          DB '
976          DB '
977          DB '
978          DB '
979          DB '
980          DB '
981          DB '
982          DB '
983          DB '
984          DB '
985          DB '
986          DB '
987          DB '
988          DB '
989          DB '
990          DB '
991          DB '
992          DB '
993          DB '
994          DB '
995          DB '
996          DB '
997          DB '
998          DB '
999          DB '
1000         DB '
    
```

SECTION 5

```

878                                     PAGE
879                                     |----- CHECK FOR SHUTDOWN 09
880 CTB:                                     |
881 MOV AL,CMOS_SHUT_DOWN+NM1           ; CMOS ADDRESS FOR SHUTDOWN BYTE
882 OUT CMOS_PORT,AL
883 JMP $+2                               ; I/O DELAY
884 IN AL,CMOS_DATA                     ; GET REQUEST NUMBER
885 CMP AL,09H                           ; WAS IT SHUTDOWN REQUEST ?
886 XCHG AL,AH                           ; SAVE THE SHUTDOWN REQUEST
887 JE CTC                                ; BYPASS INITIALIZING INTERRUPT CHIPS
888
889                                     |----- CHECK FOR SHUTDOWN 0A
890
891 CMP AH,0AH                            ; WAS IT SHUTDOWN REQUEST A?
892 JE CTC                                ; BYPASS INITIALIZING INTERRUPT CHIPS
893
894 SUB AL,AL                              ; INSURE MATH PROCESSOR RESET
895 OUT X287+1,AL
896
897 -----
898 |----- RE-INITIALIZE THE 8259 INTERRUPT #1 CONTROLLER CHIP :
899 |
900 MOV AL,11H                            ; ICW1 - EDGE, MASTER, ICW4
901 OUT INTA00,AL
902 JMP $+2                               ; WAIT STATE FOR I/O
903 MOV AL,08H                            ; SETUP ICW2 - INTERRUPT TYPE 8H (8-F)
904 OUT INTA01,AL
905 JMP $+2                               ; WAIT STATE FOR I/O
906 MOV AL,04H                            ; SETUP ICW3 - MASTER LEVEL 2
907 OUT INTA01,AL
908 JMP $+2                               ; I/O WAIT STATE
909 MOV AL,01H                            ; SETUP ICW4 - MASTER,8086 MODE
910 OUT INTA01,AL
911 JMP $+2                               ; WAIT STATE FOR I/O
912 MOV AL,0FFH                           ; MASK ALL INTERRUPTS OFF
913 OUT INTA01,AL                       ; (VIDEO ROUTINE ENABLES INTERRUPTS)
914
915 |----- RE-INITIALIZE THE 8259 INTERRUPT #2 CONTROLLER CHIP :
916 |
917 MOV AL,11H                            ; ICW1 - EDGE, SLAVE ICW4
918 OUT INTB00,AL
919 JMP $+2                               ; WAIT STATE FOR I/O
920 MOV AL,INT_TYPE                       ; SETUP ICW2 - INTERRUPT TYPE 70 (70-7F)
921 OUT INTB01,AL
922 MOV AL,02H                            ; SETUP ICW3 - SLAVE LEVEL 2
923 JMP $+2                               ; WAIT STATE FOR I/O
924 MOV AL,01H                            ; SETUP ICW4 - 8086 MODE, SLAVE
925 OUT INTB01,AL
926 JMP $+2                               ; WAIT STATE FOR I/O
927 MOV AL,0FFH                           ; MASK ALL INTERRUPTS OFF
928 OUT INTB01,AL
929
930 -----
931 | SHUTDOWN - RESTART
932 | RETURN CONTROL AFTER A SHUTDOWN COMMAND IS ISSUED
933 | DESCRIPTION
934 | A TEST IS MADE FOR THE SYSTEM FLAG BEING SET. IF THE SYSTEM FLAG IS
935 | SET, THE SHUTDOWN BYTE IN CMOS IS USED TO DETERMINE WHERE CONTROL IS
936 | RETURNED.
937 |
938 | CMOS = 0 SOFT RESET OR UNEXPECTED SHUTDOWN
939 | CMOS = 1 SHUT DOWN AFTER MEMORY SIZE
940 | CMOS = 2 SHUT DOWN AFTER MEMORY TEST
941 | CMOS = 3 SHUT DOWN WITH MEMORY ERROR
942 | CMOS = 4 SHUT DOWN WITH BOOT LOADER REQUEST
943 | CMOS = 5 JMP DWORD REQUEST - (INTERRUPT CHIPS & 287 ARE INITIALIZED)
944 | CMOS = 6 PROTECTED MODE TEST3 PASSED
945 | CMOS = 7 PROTECTED MODE TEST3 FAILED
946 | CMOS = 8 PROTECTED MODE TEST1 FAILED
947 | CMOS = 9 BLOCK MOVE SHUTDOWN REQUEST
948 | CMOS = A JMP DWORD REQUEST - (W/O INTERRUPT CHIPS INITIALIZED)
949 |
950 | NOTES: RETURNS ARE MADE WITH INTERRUPTS AND NMI DISABLED.
951 | USER MUST RESTORE SS:SP (POST DEFAULT SET = 0000:0400),
952 | ENABLE NON-MASKABLE INTERRUPTS (NMI) WITH AN OUT TO
953 | PORT 70H WITH HIGH ORDER BIT OFF, AND THEN ISSUE A
954 | STI TO ENABLE INTERRUPTS. FOR SHUTDOWN (8) THE USER
955 | MUST ALSO RESTORE THE INTERRUPT MASK REGISTERS.
956 |
957 -----
958 |----- CHECK FROM WHERE
959 CTB:
960 MOV AL,CMOS_SHUT_DOWN+NM1           ; CLEAR CMOS BYTE
961 OUT CMOS_PORT,AL
962 NOP                                    ; I/O DELAY
963 SUB AL,AL                              ; SET BYTE TO 0
964 OUT CMOS_DATA,AL
965 XCHG AH,AL
966 CMP AL,0AH                           ; COMPARE WITH MAXIMUM TABLE ENTRIES
967 JA SHUT0                               ; SKIP TO POST IF GREATER THAN MAXIMUM
968 MOV SI,OFFSET BRANCH                 ; POINT TO THE START OF THE BRANCH TABLE
969 ADD SI,AX
970 ADD SI,AX
971 MOV BX,CS:[SI]                       ; POINT TO BRANCH ADDRESS
972 MOV BX,CS:[SI]                       ; MOVE BRANCH TO ADDRESS TO BX REGISTER
973
974 |----- SET TEMPORARY STACK FOR POST
975
976 MOV AX,ABS0                           ; SET STACK SEGMENT TO ABS0 SEGMENT
977 MOV SS,AX
978 MOV SP,OFFSET *TOS                   ; SET STACK POINTER TO END OF VECTORS
979 JMP BX                                ; JUMP BACK TO RETURN ROUTINE
980
981 BRANCH: SHUT0                         ; NORMAL POWER UP/UNEXPECTED SHUTDOWN
982 DW SHUT1                             ; SHUT DOWN AFTER MEMORY SIZE
983 DW SHUT2                             ; SHUT DOWN AFTER MEMORY TEST
984 DW SHUT3                             ; SHUT DOWN WITH MEMORY ERROR
985 DW SHUT4                             ; SHUT DOWN WITH BOOT LOADER REQUEST
986 DW SHUT5                             ; JMP DWORD REQUEST WITH INTERRUPT INIT
987 DW SHUT6                             ; PROTECTED MODE TEST7 PASSED
988 DW SHUT7                             ; PROTECTED MODE TEST7 FAILED
989 DW SHUT8                             ; PROTECTED MODE TEST1 FAILED
990 DW SHUT9                             ; BLOCK MOVE SHUTDOWN REQUEST
991 DW SHUTA                             ; JMP DWORD REQUEST (W/O INTERRUPT INIT)

```



```

1676
1677
1678          |----- FILL REGEN AREA WITH BLANK
1679 048F 33 FF          Z_3: XOR    D1,D1          ; SET UP POINTER FOR REGEN
1680 0481 88 B000        MOV    AX,0B000H       ; SET UP ES TO VIDEO REGEN
1681 0464 8E C0          MOV    ES,AX
1682
1683 0466 89 0800        MOV    CX,2048        ; NUMBER OF WORDS IN MONOCHROME CARD
1684 0469 88 0720        MOV    AX,' *+?*'H    ; FILL CHARACTER FOR ALPHA * ATTRIBUTE
1685 046C F3/ AB        REP    STOSW          ; FILL THE REGEN BUFFER WITH BLANKS
1686
1687 046E 33 FF          XOR    D1,D1          ; CLEAR COLOR VIDEO BUFFER MEMORY
1688 0470 88 B800        MOV    BX,0B800H     ; SET UP ES TO COLOR VIDEO MEMORY
1689 0473 8E C3          MOV    ES,BX
1690 0475 89 2000        MOV    CX,8192
1691 0478 F3/ AB        REP    STOSW          ; FILL WITH BLANKS
1692
1693          |----- ENABLE VIDEO AND CORRECT PORT SETTING
1694
1695 047A BA 03B8        MOV    DX,3BBH
1696 047D B0 29          MOV    AL,29H
1697 047F EE            OUT    DX,AL          ; SET VIDEO ENABLE PORT
1698
1699          |----- SET UP OVERSCAN REGISTER
1700
1701 0480 42            INC    DX              ; SET OVERSCAN PORT TO A DEFAULT
1702 0481 B0 30          MOV    AL,30H         ; VALUE 30H FOR ALL MODES EXCEPT 640X200
1703 0483 EE            OUT    DX,AL         ; OUTPUT THE CORRECT VALUE TO 3D9 PORT
1704
1705          |----- ENABLE COLOR VIDEO AND CORRECT PORT SETTING
1706
1707 0484 BA 03D8        MOV    DX,3D8H
1708 0487 B0 28          MOV    AL,28H
1709 0489 EE            OUT    DX,AL         ; SET VIDEO ENABLE PORT
1710
1711          |----- SET UP OVERSCAN REGISTER
1712
1713 048A 42            INC    DX              ; SET OVERSCAN PORT TO A DEFAULT
1714 048B B0 30          MOV    AL,30H         ; VALUE 30H FOR ALL MODES EXCEPT 640X200
1715 048D EE            OUT    DX,AL         ; OUTPUT THE CORRECT VALUE TO 3D9 PORT
1716
1717          |----- DISPLAY FAILING CHECKPOINT AND
1718
1719 048E 8C C8          MOV    AX,CS          ; SET STACK SEGMENT TO CODE SEGMENT
1720 0490 BE D0          MOV    SI,AX
1721
1722 0492 BB B000        MOV    BX,0B000H     ; SET DS TO B/W DISPLAY BUFFER
1723 0495 BE DB          MOV    DS,BX
1724
1725 0497 B0 30          MOV    AL,'0'         ; DISPLAY BANK 000000
1726 0499 B9 0006        MOV    CX,6
1727 049C 2B FF          SUB    D1,D1          ; START AT 0
1728 049E 86 05          MOV    [D1],AL        ; WRITE TO DISPLAY REGEN BUFFER
1729 04A0 47            INC    D1             ; POINT TO NEXT POSITION
1730 04A1 47            INC    D1
1731 04A2 E2 FA        LOOP  Z
1732
1733 04A4 80 FF B8        CMP    BH,0B8H        ; CHECK THAT COLOR BUFFER WRITTEN
1734 04A7 74 0C          JZ     Z_1            ; POINT TO START OF BUFFER
1735 04A9 2B FF          SUB    DT,D1
1736
1737 04AB B7 B0          MOV    BH,0B0H
1738 04AD BE C3          MOV    ES,BX          ; ES = MONOCHROME
1739 04AF B7 B8          MOV    BH,0B8H       ; SET SEGMENT TO COLOR
1740 04B1 BE DB          MOV    DS,BX          ; DS = COLOR
1741 04B3 EB E2          JMP    Z_0
1742
1743          |----- PRINT FAILING BIT PATTERN
1744
1745 04B5 B0 20          Z_1: MOV    AL,' '         ; DISPLAY A BLANK
1746 04B7 86 06          MOV    [D1],AL        ; WRITE TO COLOR BUFFER
1747 04B9 26: 88 05        MOV    ES:[D1],AL     ; WRITE TO MONOCHROME REGEN BUFFER
1748 04BC 47            INC    D1             ; POINT TO NEXT POSITION
1749 04BD 47            INC    D1
1750 04BE E4 81          IN    AL,MFG_PORT+1  ; GET THE HIGH BYTE OF FAILING PATTERN
1751 04C0 81 04          MOV    CL,4          ; SHIFT COUNT
1752 04C2 D2 E8          SHR    AL,CL          ; NIBBLE SWAP
1753 04C4 BC 05F R      MOV    SP,OFFSET Z1_0
1754 04C7 EB 18          JMP    SHORT PR
1755
1756 04C9 E4 81          Z1: IN    AL,MFG_PORT+1  ; ISOLATE TO LOW NIBBLE
1757 04CB 24 0F          AND    AL,0FH
1758 04CD BC 05B1 R      MOV    SP,OFFSET Z2_0
1759 04D0 EB 12          JMP    SHORT PR
1760 04D2 E4 82          Z2: IN    AL,MFG_PORT+2  ; GET THE HIGH BYTE OF FAILING PATTERN
1761 04D4 B1 04          MOV    CL,4          ; SHIFT COUNT
1762 04D6 D2 E8          SHR    AL,CL          ; NIBBLE SWAP
1763 04D8 BC 05B3 R      MOV    SP,OFFSET Z3_0
1764 04DB EB 07          JMP    SHORT PR
1765 04DD E4 82          Z3: IN    AL,MFG_PORT+2  ; ISOLATE TO LOW NIBBLE
1766 04DF 24 0F          AND    AL,0FH
1767 04E1 BC 05B5 R      MOV    SP,OFFSET Z4_0 ; RETURN TO Z4:
1768
1769          |----- CONVERT AND PRINT
1770
1771 04E4 04 90          PR: ADD    AL,090H       ; CONVERT 00-0F TO ASCII CHARACTER
1772 04E6 27            DAA                    ; ADD FIRST CONVERSION FACTOR
1773 04E7 14 40          ADC    AL,040H        ; ADJUST FOR NUMERIC AND ALPHA RANGE
1774 04E9 27            DAA                    ; ADD CONVERSION AND ADJUST LOW NIBBLE
1775 04EB 27            DAA                    ; ADJUST HIGH NIBBLE TO ASCII RANGE
1776 04ED 8B 05          MOV    [D1],AL        ; WRITE TO COLOR BUFFER
1777 04EF 26: 88 05        MOV    ES:[D1],AL     ; WRITE TO MONOCHROME BUFFER
1778 04F1 47            INC    D1             ; POINT TO NEXT POSITION
1779 04F3 47            INC    D1
1780 04F5 C3            RET
1781
1782          |----- DISPLAY 201 ERROR
1783
1784 04F2 B0 20          Z4: MOV    AL,' '         ; DISPLAY A BLANK
1785 04F4 86 05          MOV    [D1],AL        ; WRITE TO DISPLAY REGEN BUFFER
1786 04F6 26: 88 05        MOV    ES:[D1],AL     ; WRITE TO MONOCHROME BUFFER
1787 04F9 47            INC    D1             ; POINT TO NEXT POSITION
1788 04FA 47            INC    D1
1789 04FB B0 32          MOV    AL,'2'         ; DISPLAY 201 ERROR
    
```

SECTION 5


```

1904                                MOV     AL,11H,AL                ;
1905 05A4 B0 11                        OUT     MFG_PORT,AL          ;
1906 05A6 E6 80                        ;
1907                                ;
1908                                ;
1909                                ;
1910 05A8 32 DB                        ;
1911 05AA 33 C9                        ;
1912                                ;
1913 05AC                                C34:
1914 05AC E4 61                        IN      AL,PORT_B          ;
1915 05AE A8 10                        TEST   AL,REFRESH_BIT     ;
1916 05B0 E1 FA                        LOOPZ  C34                 ;
1917 05B2                                ;
1918 05B2 E4 61                        IN      AL,PORT_B          ;
1919 05B4 A8 10                        TEST   AL,REFRESH_BIT     ;
1920 05B6 E0 FA                        LOOPNZ C35                 ;
1921                                ;
1922 05B8 FE CB                        DEC     BL                  ;
1923 05BA 75 F0                        JNZ    C34                 ;
1924                                ;
1925 05BC 81 F9 F780                   CMP     CX,RATE_UPPER      ;
1926 05C0 73 07                        JAE    C36E1              ;
1927 05C2                                ;
1928 05C2 BA 0101                       MOV     DX,0101H          ;
1929 05C5 E8 0000 E                     CALL   ERR_BEEP           ;
1930 05C8 F4                            HLT                                     ;
1931 05C9                                C36:
1932 05C9 81 F9 F9FD                   CMP     CX,RATE_LOWER      ;
1933 05CD 77 F3                        JAE    C36E1              ;
1934                                ;
1935                                ;
1936                                ;
1937 05CF E4 82                        IN      AL,DMA_PAGE+1     ;
1938 05D1 24 F8                        AND     AL,KEY_BD_INHIB+DSP_AMP+MFG_LOOP+BASE_MEM+BASE_MEMB ;
1939 05D3 A2 0012 R                     MOV     MFG_TST,AL        ;
1940 05D6 2A C0                        SUB     AL,AL              ;
1941 05D8 E6 82                        OUT     DMA_PAGE+1,AL     ;
1942                                ;
1943                                ;
1944                                ;
1945                                ;
1946                                ;
1947                                ;
1948                                ;
1949                                ;
1950                                ;
1951                                ;
1952                                ;
1953                                ;
1954                                ;
1955 05DA 0F 01 E0                       + DB     00FH,001H,0E0H   ;
1956 05DD A9 000F                       TEST   AX,0FH             ;
1957 05E0 75 34                        JNZ    ERR_PROT           ;
1958                                ;
1959                                ;
1960                                ;
1961 05E2 B0 12                        MOV     AL,12H            ;
1962 05E4 E6 80                        OUT     MFG_PORT,AL      ;
1963                                ;
1964 05E6 1E                            PUSH   DS                  ;
1965 05E7 07                            POP    DS                  ;
1966 05E8 BF D0A0                       MOV     DI,SYS_IDT_LOC    ;
1967 05EB B9 0003                       MOV     CX,3              ;
1968 05EE B8 AAAA                       MOV     AX,0AAAAH        ;
1969 05F1 E8 0619 R                     CALL   WRT_PAT            ;
1970 05F4 B8 5555                       MOV     AX,05555H        ;
1971 05F7 E8 0619 R                     CALL   WRT_PAT            ;
1972 05FA 2B C0                        SUB     AX,AX              ;
1973 05FC E8 0619 R                     CALL   WRT_PAT            ;
1974                                ;
1975                                ;
1976                                ;
1977 05FF FD                            STD    PUSHF              ;
1978 0600 9C                            PUSHF                      ;
1979 0601 58                            POP     AX                 ;
1980 0602 A9 0200                       TEST   AX,0200H          ;
1981 0605 75 0F                        JNZ    ERR_PROT           ;
1982 0607 A9 0400                       TEST   AX,0400H          ;
1983 060A 74 0A                        JZ     ERR_PROT           ;
1984 060C FC                            CLD                        ;
1985 060D 9C                            PUSHF                      ;
1986 060E 58                            POP     AX                 ;
1987 060F A9 0400                       TEST   AX,0400H          ;
1988 0612 75 02                        JNZ    ERR_PROT           ;
1989                                ;
1990 0614 EB 3D                        JMP     SHORT C37A        ;
1991 0616                                ERR_PROT:
1992 0616 F4                            HLT                        ;
1993 0617 EB FD                        JMP     SHORT ERR_PROT    ;
1994                                ;
1995                                ;
1996                                ;
1997 0619 B9 0003                       WRT_PAT:MOV  CX,3          ;
1998 061C F3 AB                         REP     STOSW              ;
1999 061E BD D0A0                       MOV     BP,SYS_IDT_LOC    ;
2000                                ;
2001 0621 26                            DB     026H                ;
2002                                ;
2003 0622 0F                            LIDT  [BP]                 ;
2004 0623                                ;
2005 0623 8B 5E 00                       + ?70001 LABEL  BYTE      ;
2006 0626                                ;
2007 0623                                ;
2008 0623 01                            + ?70002 LABEL  BYTE      ;
2009 0626                                ;
2010 0626 BD D0A0                       + ?70004 LABEL  BYTE      ;
2011                                ;
2012 0629 26                            + ?70005 LABEL  BYTE      ;
2013                                ;
2014 062A 0F                            ;
2015 062B                                ;
2016 062B 8B 5E 00                       ;
2017 062E                                ;

```

SECTION 5


```

2244
2247 075F BC 0000      MOV     SP,POST_SS      ; SET STACK FOR SYSINIT1
2248 0762 BE D4        MOV     SS,SP
2249 0764 BC 8000      MOV     SP,POST_SP
2250 0767 E8 0000 E    CALL    SYSINIT1       ; CALL THE DESCRIPTOR TABLE BUILDER
2251                                     ; AND REAL-TO-PROTECTED MODE SWITCHER
2252
2253 076A B0 IA        MOV     AL,IAH          ;
2254 076C E6 80        OUT     MFG_PORT,AL    ;
2255                                     ;
2256                                     ;
2257 ----- SET TEMPORARY STACK
2258 076E 6A 08        PUSH   BYTE PTR GD_T_PTR ; SET (DS:) SELECTOR TO GD_T SEGMENT
2259 0770 IF          POP     DS
2260 0771 C7 06 008A 0000 MOV     DS:SS_TEMP.BASE_LO_WORD,0
2261 0777 C6 06 008C 00 MOV     BYTE PTR DS:(SS_TEMP.BASE_HI_BYTE),0
2262 077C BE 0058      MOV     SI,SS_TEMP
2263 077F BE D6        MOV     SS,SI
2264 0781 BC FFFD      MOV     SP,MAX_SEG_LEN-2
2265
2266 -----
2267 ; TEST.13
2268 ; PROTECTED MODE TEST AND MEMORY SIZE DETERMINE ( 0 --> 640K )
2269 ;
2270 ; DESCRIPTION:
2271 ; THIS ROUTINE RUNS IN PROTECTED MODE IN ORDER TO ADDRESS ALL OF STORAGE.
2272 ; IT CHECKS THE MACHINE STATUS WORD (MSW) FOR PROTECTED MODE AND THE BASE
2273 ; MEMORY SIZE IS DETERMINED AND SAVED. BIT 4 OF THE CMOS DIAGNOSTIC
2274 ; STATUS BYTE IS SET IF 512K --> 640K MEMORY IS INSTALLED.
2275 ; DURING A POWER UP SEQUENCE THE MEMORY SIZE DETERMINE IS DONE WITH
2276 ; PLANAR AND I/O PARITY CHECKS DISABLED. DURING A SOFT RESET THE MEMORY
2277 ; SIZE DETERMINE WILL CHECK FOR PARITY ERRORS.
2278 -----
2279 ;----- INSURE PROTECTED MODE
2280
2281 SMSW AX             ; GET THE MACHINE STATUS WORD
2282 DB 00FH,001H,0E0H
2283 0784 0F 01 E0      * TEST AX,VIRTUAL_ENABLE ; ARE WE IN PROTECTED MODE
2284 0787 A9 0001      TEST AX,VIRTUAL_ENABLE
2285 078A 75 0C        JNZ    VIR_OK
2286
2287 078C BB 080F      SHUT_8: MOV AX,8*H+(CMOS_SHUT_DOWN+NM1) ; SET THE RETURN ADDRESS
2288 078F EB 0000 E    CALL CMOS_WRITE ; AND SET SHUTDOWN 8
2289 0792 E9 0000 E    JMP    PROC_SHUTDOWN ; CAUSE A SHUTDOWN
2290
2291 ;----- VIRTUAL MODE ERROR HALT
2292
2293 0795 F4          SHUT_8: HLT ; ERROR HALT
2294 0796 EB FD
2295
2296 ;----- 64K SEGMENT LIMIT
2297
2298 0798 C7 06 0048 FFFF VIR_OK: MOV DS:ES_TEMP.SEG_LIMIT,MAX_SEG_LEN
2299
2300 ;----- CPL0, DATA ACCESS RIGHTS
2301
2302 079E C6 06 004D 93 MOV     BYTE PTR DS:(ES_TEMP.DATA_ACC_RIGHTS),CPL0_DATA_ACCESS
2303
2304 ;----- START WITH SEGMENT ADDRESS 01-0000 (SECOND 64K)
2305
2306 07A3 C6 06 004C 01 MOV     BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE),01H
2307 07A8 C7 06 004A 0000 MOV     DS:ES_TEMP.BASE_LO_WORD,0H
2308
2309 07AE B0 1B        MOV     AL,1BH          ;
2310 07B0 E6 80        OUT     MFG_PORT,AL    ;
2311                                     ;
2312 07B2 BB 0040      MOV     BX,16*4         ; SET THE FIRST 64K DONE
2313
2314 ;----- START STORAGE SIZE/CLEAR
2315
2316 07B5 NOT_DONE:
2317 07B5 6A 48        PUSH   BYTE PTR ES_TEMP ; POINT ES TO DATA
2318 07B7 07          POP     ES              ; POINT TO SEGMENT TO TEST
2319 07B8 EB 07D4 R    CALL    HOW_BIG        ; DO THE FIRST 64K
2320 07BB 74 03        JZ     NOT_FIN         ; GO THE FIRST 64K
2321 07BD E9 0872 R    JMP    DONE            ; CHECK IF TOP OF MEMORY
2322
2323 07C0 NOT_FIN:
2324 07C0 83 C3 40      ADD     BX,16*4         ; BUMP MEMORY COUNT BY 64K
2325
2326 ;----- DO NEXT 64K (0X0000) BLOCK
2327
2328 07C3 FE 06 004C INC     BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE)
2329
2330 ;----- CHECK FOR END OF FIRST 640K (END OF BASE MEMORY)
2331
2332 07C7 80 3E 004C 0A CMP     BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE),0AH
2333 07CC 75 E7        JNZ    NOT_DONE        ; GO IF NOT
2334 07CE EB 084F R    CALL    HOW_BIG_END    ; GO SET MEMORY SIZE
2335 07D1 E9 0872 R    JMP    DONE
2336
2337 ;----- FILL/CHECK LOOP
2338
2339 07D4 HOW_BIG:
2340 07D4 2B FF        SUB     DI,DI           ; TEST PATTERN
2341 07D6 BB AA55      MOV     AX,0AA55H      ; SAVE PATTERN
2342 07D9 BB C5        MOV     CX,AX          ; SAVE PATTERN
2343 07DB E6 89 05    MOV     ES:[DI],AX     ; WRITE PATTERN TO MEMORY
2344 07DE B0 0F        MOV     AL,0FH         ; PUT SOMETHING IN AL
2345 07E0 26 8B 05    MOV     AX,ES:[DI]     ; GET PATTERN
2346 07E3 26 89 05    MOV     ES:[DI],AX     ; INSURE NO PARITY I/O CHECK
2347 07E6 33 C1        XOR     AX,CX          ; COMPARE PATTERNS
2348 07E8 75 65        JNZ    HOW_BIG_END     ; GO END IF NO COMPARE
2349
2350 07EA 1E          PUSH   DS              ; POINT TO SYSTEM DATA AREA
2351 07EB 6A 18        PUSH   BYTE PTR RSDA_PTR ; GET (DS:)
2352 07ED IF          POP     DS
2353
2354 ;----- IS THIS A SOFT RESET
2355
2356 07EE 81 3E 0072 R 1234 CMP     0RESET_FLAG,1234H ; SOFT RESET
2357 07F4 IF          POP     DS              ; RESTORE DS
2358 07F5 75 36        JNZ    HOW_BIG_2       ; GO IF NOT SOFT RESET
2359

```



```

2474 0891 83 C3 40      DONEA: ADD    BX,16*4      ; BUMP MEMORY COUNT BY 64K
2475
2476                  ;----- DO NEXT 64K (XX0000) BLOCK
2477
2478 0894 FE 06 004C      INC    BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE)
2479
2480                  ;----- CHECK FOR TOP OF MEMORY (FE0000)
2481
2482 0898 80 3E 004C FE  CMP    BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE),0FEH ; LAST OF MEMORY?
2483 089D 75 E7          JNZ    NOT_DONE1      ; GO IF NOT
2484 089F E8 0919 R      CALL   HOW_BIG_END1   ; GO SET MEMORY SIZE
2485 08A2 E9 092C R      JMP    DONE1
2486
2487                  ;----- FILL/CHECK LOOP
2488
2489 08A5 29 FF          HOW_BIG1: SUB    DI,DI
2490 08A8 B5 A555        MOV    AX,0AA55H      ; TEST PATTERN
2491 08AB 8B C8          MOV    CX,AX          ; SAVE PATTERN
2492 08AC 26 89 05      MOV    ES:[DI],AX     ; SEND PATTERN TO MEMORY
2493 08AF 80 0F          MOV    AL,0FH         ; PUT SOMETHING IN AL
2494 08B1 26 8B 05      MOV    AX,ES:[DI]     ; GET PATTERN
2495 08B4 26 89 05      MOV    ES:[DI],AX     ; INSURE NO PARITY I/O CHECK
2496 08B7 33 C1          XOR    AX,CX          ; COMPARE PATTERNS
2497 08B9 76 5E          JNZ    HOW_BIG_END1   ; GO END IF NO COMPARE
2498
2499                  ;----- IS THIS A SOFT RESET
2500
2501
2502 08BB 1E             PUSH   DS              ; POINT TO SYSTEM DATA AREA
2503 08BC 6A 18          PUSH  BYTE PTR RSQA_PTR
2504 08BE 1F             POP    DS
2505 08BF 81 3E 0072 R 1234  CMP    0RESET_FLAG,1234H ; SOFT RESET
2506 08C5 1F             POP    DS              ; RESTORE DS
2507 08C6 76 2F          JNZ    HOW_BIG_2A     ; GO IF NOT SOFT RESET
2508
2509                  ;----- CHECK PARITY WITH PARITY BITS OFF
2510
2511 08C8 26: CT 05 0101 MOV    WORD PTR ES:[DI],0101H ; TURN OFF BOTH PARITY BITS
2512 08CD 6A FF          PUSH  BYTE PTR OFFH    ; PLACE 0FFFFH IN STACK (BUS BITS ON)
2513 08CF 58             POP    AX              ; DELAY - CAUSING BUS BITS ON
2514 08D0 26: 8B 05      MOV    AX,ES:[DI]     ; CHECK PARITY
2515
2516 08D3 E4 61          IN    AL,PORT_B        ; CHECK FOR PLANAR OR I/O PARITY CHECK
2517 08D5 24 C0          AND   AL,PARITY_ERR    ;
2518 08D7 26: 89 05      MOV    ES:[DI],AX     ; CLEAR POSSIBLE PARITY ERROR
2519 08DA 75 3D          JNZ    HOW_BIG_END1   ; GO IF PLANAR OR I/O PARITY CHECK
2520
2521                  ;----- CHECK ALL BITS
2522
2523 08DC 26: CT 05 FFFF MOV    WORD PTR ES:[DI],0FFFFH ; TURN ON ALL BITS
2524 08E1 6A 00          PUSH  BYTE PTR 0       ; PLACE 0000H IN STACK (BUS BITS OFF)
2525 08E3 58             POP    AX              ; DELAY - CAUSING BUS BITS OFF
2526 08E4 26: 8B 05      MOV    AX,ES:[DI]     ; CHECK FOR FFFFH
2527 08E7 50             PUSH  AX               ; SAVE RESULTS
2528 08E8 E4 61          IN    AL,PORT_B        ; CHECK FOR PLANAR OR I/O PARITY CHECK
2529 08EA 24 C0          AND   AL,PARITY_ERR    ;
2530 08EC 26: 89 05      MOV    ES:[DI],AX     ; CLEAR POSSIBLE PARITY ERROR
2531 08EF 58             POP    AX              ; GET RESULTS
2532 08F0 75 27          JNZ    HOW_BIG_END1   ; GO IF PLANAR OR I/O PARITY CHECK
2533 08F2 3D FFFF      CMP    AX,0FFFFH
2534 08F5 75 22          JNZ    HOW_BIG_END1
2535
2536                  ;----- CLEAR 64K BLOCK OF MEMORY
2537
2538 08F7 2B C0          HOW_BIG_2A: SUB    AX,AX           ; WRITE ZEROS
2539 08F7 2B C0          MOV    CX,2000H*4     ; SET COUNT FOR 32K WORDS
2540 08F9 B9 8000      REP    STOSW           ; FILL 32K WORDS
2541 08FC F3/ AB
2542
2543                  ;----- CHECK 64K BLOCK FOR PARITY CHECK (VALID TEST DURING SOFT RESET ONLY)
2544
2545 08FE 1E             PUSH   DS
2546 08FF 06             PUSH  ES
2547 0900 06             PUSH  ES              ; GET ES TO DS
2548 0901 1F             POP    DS
2549 0902 B9 8000      MOV    CX,2000H*4     ; SET COUNT FOR 32K WORDS
2550 0905 2B F6          SUB    SI,S1
2551 0907 F3/ AD      REP    LODSW
2552 0909 2B FF          SUB    DI,DI           ; SET TO BEGINNING OF BLOCK
2553 090B E4 61          IN    AL,PORT_B        ; CHECK FOR PLANAR OR I/O PARITY CHECK
2554 090D 24 C0          AND   AL,PARITY_ERR    ;
2555 090F 26: CT 05 0000 MOV    WORD PTR ES:[DI],0   ; CLEAR POSSIBLE PARITY ERROR
2556 0914 07             POP    ES              ; RESTORE SEGMENT
2557 0916 1F             POP    DS
2558 0916 76 01          JNZ    HOW_BIG_END1   ; GO IF PLANAR OR I/O PARITY CHECK
2559
2560 0918 C3             RET
2561
2562 0919
2563 0919 80 1E          HOW_BIG_END1: MOV    AL,1EH          ; 0000000000000000
2564 091B E6 50          OUT   MFG_PORT,AL     ; 00 CHECKPOINT 1E 00
2565
2566                  ;----- SET EXPANSION MEMORY SIZE DETERMINED IN CMOS
2567
2568 091D 80 B0          MOV    AL,CMOS_U_M_S_LO+NM1 ; ADDRESS LOW BYTE
2569 091F 8A E3          MOV    AH,BL          ; GET LOW MEMORY SIZE
2570 0921 E8 0000 E      CALL   CMOS_WRITE     ; SET LOW BYTE
2571 0924 80 B1          MOV    AL,CMOS_U_M_S_HI+NM1 ; ADDRESS HI BYTE
2572 0926 8A E7          MOV    AH,BH          ; GET THE HIGH MEMORY SIZE
2573 0928 E8 0000 E      CALL   CMOS_WRITE     ; PLACE IN CMOS
2574 092B C3             RET
2575
2576                  ;----- TEST ADDRESS LINES 19 - 23
2577
2578 092C 80 1F          DONE1: MOV    AL,1FH          ; 0000000000000000
2579 092E E6 80          OUT   MFG_PORT,AL     ; 00 CHECKPOINT 1F 00
2580 0930 C6 06 004C 00 MOV    BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE),00H
2581 0932 2B FF          SUB    DI,DI           ; SET LOCATION POINTER TO ZERO
2582 0934 E8 FFFF R      MOV    DX,0FFFFH     ; WRITE FFFF AT ADDRESS 0
2583 093A E8 0969 R      CALL   S00            ;
2584 093D 2B D2          SUB    DX,DX           ; WRITE 0
2585
2586 093F C6 06 004C 08 MOV    BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE),08H
2587 0944 E8 0969 R      CALL   S00
    
```



```

2702
2703 0A04 81 C1 0080      ADD    CX,080H          ; POINT TO NEXT 2K BLOCK
2704 0A08 81 F9 C800      CMP    CX,0C800H       ; TOP OF VIDEO ROM AREA YET?
2705 0A0C 7C E3           JL     CHK_VIDEO1      ; TRY AGAIN
2706 0A0E 23 C9           AND    CX,CX           ; SET NON ZERO FLAG
2707 0A10           CHK_VIDEO2:           RET                     ; RETURN TO CALLER
2708 0A10 C3
2709
2710 ;----- CMOS VIDEO BITS NON ZERO (CHECK FOR PRIMARY DISPLAY AND NO VIDEO ROM)
2711
2712 0A11           MOS_OK_1:           CALL  CHK_VIDEO        ; IS THE VIDEO ROM INSTALLED?
2713 0A11 E8 09EE R        JZ     BAD_MOS         ; WRONG CONFIGURATION IN CONFIG BYTE
2714 0A14 74 26           MOV    AL,AH           ; RESTORE CONFIGURATION
2715
2716 0A16 8A C4           TEST  #MFG_TST,DSP_JMP ; CHECK FOR DISPLAY JUMPER
2717 0A18 F6 06 0012 R 40  JZ     MOS_OK_2        ; GO IF COLOR CARD IS PRIMARY DISPLAY
2718 0A1D 74 0A
2719
2720 ;----- MONOCHROME CARD IS PRIMARY DISPLAY (NO JUMPER INSTALLED)
2721
2722 0A1F 24 30           AND    AL,30H         ; INSURE MONOCHROME IS PRIMARY
2723 0A21 3C 30           CMP    AL,30H         ; CONFIGURATION OK?
2724 0A23 75 17           JNZ   BAD_MOS         ; GO IF NOT
2725 0A25 8A C4           MOV    AL,AH           ; RESTORE CONFIGURATION
2726 0A27 EB 08           JMP    SHORT MOS_OK    ; USE THE CONFIGURATION BYTE FOR DISPLAY
2727
2728 ;----- COLOR CARD
2729
2730 0A29           MOS_OK_2:           AND    AL,30H         ; STRIP UNWANTED BITS
2731 0A29 24 30           CMP    AL,30H         ; MUST NOT BE MONO WITH JUMPER INSTALLED
2732 0A2B 3C 30           MOV    AL,AH           ; RESTORE CONFIGURATION
2733 0A2D 8A C4           JZ     BAD_MOS         ; GO IF YES
2734 0A2F 74 0B
2735
2736 ;----- CONFIGURATION MUST HAVE AT LEAST ONE DISKETTE
2737
2738 0A31 A8 01           MOS_OK:  TEST   AL,01H         ; MUST HAVE AT LEAST ONE DISKETTE
2739 0A33 75 26           JNZ   NORMAL_CONFIG   ; GO SET CONFIGURATION IF OK
2740 0A35 F6 06 0012 R 20  TEST  #MFG_TST,MFG_LOOP ; EXCEPT IF MFG JUMPER IS INSTALLED
2741 0A3A 74 1F           JZ     NORMAL_CONFIG   ; GO IF INSTALLED
2742
2743 ;----- MINIMUM CONFIGURATION WITH BAD CMOS OR NON VALID VIDEO
2744
2745 0A3C           BAD_MOS:           MOV    AX,CMOS_DIAG+NM1 ; GET THE DIAGNOSTIC STATUS
2746 0A3C BB 008E          CALL   CMOS_READ       ; CMOS READ
2747 0A3F E8 0900 E        TEST  AL,BAD_BAT+BAD_CKSUM ; WAS BATTERY DEFECTIVE OR BAD CHECKSUM
2748 0A42 AB C0           JNZ   BAD_MOS1        ; GO IF YES
2749 0A44 75 03
2750
2751 0A46 E8 0000 E        CALL   CONFIG_BAD      ; SET THE MINIMUM CONFIGURATION FLAG
2752 0A49
2753 0A49 E8 09EE R        CALL   CHK_VIDEO       ; CHECK FOR VIDEO ROM
2754 0A4C 80 01           MOV    AL,01H         ; DISKETTE ONLY
2755 0A4E 74 0B           JZ     NORMAL_CONFIG   ; GO IF VIDEO ROM PRESENT
2756
2757 0A50 F6 06 0012 R 40  TEST  #MFG_TST,DSP_JMP ; CHECK FOR DISPLAY JUMPER
2758 0A55 80 11           MOV    AL,11H         ; DEFAULT TO 40X25 COLOR
2759 0A57 74 02           JZ     NORMAL_CONFIG   ; GO IF JUMPER IS INSTALLED
2760
2761 0A59 80 31           MOV    AL,31H         ; DISKETTE / B/W DISPLAY 80X25
2762
2763 ;-----
2764 ;----- CONFIGURATION AND MFG MODE -----
2765 ;-----
2766
2767 0A5B           NORMAL_CONFIG:     TEST  #MFG_TST,MFG_LOOP ; IS THE MANUFACTURING JUMPER INSTALLED
2768 0A5B F6 06 0012 R 20  JNZ   NORM1           ; GO IF NOT
2769 0A60 75 02           AND    AL,03EH        ; STRIP DISKETTE FOR MFG TEST
2770 0A62 24 3E
2771
2772 0A64 2A E4           NORM1:  SUB    AH,AH           ; SAVE SWITCH INFORMATION
2773 0A66 A3 0010 R        MOV    #EQUIP_FLAG,AX  ; BYPASS IF SOFT RESET
2774 0A69 81 3E 0072 R 1234  CMP    #RESET_FLAG,1234H
2775 0A6F 74 2C           JZ     E6
2776
2777 ;----- GET THE FIRST SELF TEST RESULTS FROM KEYBOARD
2778
2779 0A71 80 60           MOV    AL,WRITE_8042_LOC ; ENABLE KEYBOARD
2780 0A73 E8 039D R        CALL   C8042           ; ISSUE WRITE BYTE COMMAND
2781 0A76 80 4D           MOV    AL,4DH         ; ENABLE OUTPUT BUFFER FULL INTERRUPT,
2782
2783 0A78 E6 60           OUT   PORT_A,AL       ; SET SYSTEM FLAG, PC 1 COMPATIBILITY,
2784
2785 0A7A 2B C9           SUB    CX,CX           ; INHIBIT OVERRIDE, ENABLE KEYBOARD
2786 0A7C E8 03A2 R        CALL   C42_1          ; WAIT FOR COMMAND ACCEPTED
2787
2788 0A7F 89 7FFF          MOV    CX,07FFFH      ; SET LOOP COUNT FOR APPROXIMATELY 100MS
2789
2790 0A82 E4 64           TST6:  IN     AL,STATUS_PORT ; TO RESPOND
2791 0A84 A8 01           TEST  AL,OUT_BUF_FULL ; WAIT FOR OUTPUT BUFFER FULL
2792 0A86 E1 FA           LOOPZ TST6            ; TRY AGAIN IF NOT
2793
2794 0A88 9C           PUSHF                  ; SAVE FLAGS
2795 0A89 80 AD           MOV    AL,DIS_KBD     ; DISABLE KEYBOARD
2796 0A8B E8 039D R        CALL   C8042           ; ISSUE THE COMMAND
2797 0A8E 9D           POPF                   ; RESTORE FLAGS
2798 0A8F 74 0C           JZ     E6              ; CONTINUE WITHOUT RESULTS
2799
2800 0A91 E4 60           IN     AL,PORT_A       ; GET INPUT FROM KEYBOARD
2801 0A93 A2 0072 R        MOV    BYTE PTR #RESET_FLAG,AL ; TEMPORARY SAVE FOR AA RECEIVED
2802
2803 ;----- CHECK FOR MFG REQUEST
2804
2805 0A96 3C 65           CMP    AL,065H        ; LOAD MANUFACTURING TEST REQUEST?
2806 0A98 75 03           JNE   E6              ; CONTINUE IF NOT
2807 0A9A E9 0C34 R        JMP    MFG_BOOT       ; ELSE GO TO MANUFACTURING BOOTSTRAP

```

```

2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821 0A9D A1 0010 R
2822 0AA0 50
2823 0AA1 B0 30
2824 0AA3 A3 0010 R
2825 0AA6 2B C0
2826 0AA8 CD 10
2827 0AAA B0 20
2828 0AAC A3 0010 R
2829 0AAF B8 0003
2830 0AB2 CD 10
2831 0AB4 B8 0001
2832 0AB7 CD 10
2833 0AB9 58
2834 0ABA A3 0010 R
2835 0ABD 24 30
2836 0ABF 75 11
2837 0AC1 1E
2838 0AC2 50
2839 0AC3 2B C0
2840 0AC5 8E D8
2841 0ACT BF 0040 R
2842 0ACA CT 05 0000 E
2843 0ACE 58
2844 0ACF 1F
2845 0AD0 EB 7F
2846 0AD2
2847 0AD2 3C 30
2848 0AD4 74 08
2849 0AD6 FE C4
2850 0AD8 3C 20
2851 0ADA 75 02
2852 0ADC B4 03
2853 0ADE
2854 0ADE 86 E0
2855 0AEO 50
2856 0AE1 2A E4
2857 0AE3 CD 10
2858 0AEE 58
2859 0AE6 50
2860 0AET B8 B000
2861 0AEA B4 0388
2862 0AED B9 0800
2863 0AF0 80 FC 30
2864 0AF3 74 07
2865 0AF5 B7 B8
2866 0AF7 BA 03D8
2867 0AFA B5 20
2868 0AFC
2869 0AFC A0 0065 R
2870 0AFF 24 37
2871 0B01 EE
2872 0B02 8E C3
2873 0B04 8E D8
2874 0B06 D1 C9
2875 0B08 E8 1000 E
2876 0B0B 78 78
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887 0B0D B0 22
2888 0B0F E6 80
2889 0B11 58
2890 0B12 50
2891 0B13 B4 00
2892 0B15 CD 10
2893 0B17 B8 7020
2894 0B1A 2B FF
2895 0B1C B9 0028
2896 0B1F F3/ AB
2897
2898
2899
2900
2901
2902
2903
2904
2905
2906
2907 0B21 58
2908 0B22 50
2909 0B23 80 FC 30
2910 0B26 BA 03BA
2911 0B29 74 03
2912 0B2B BA 03DA
2913 0B2E
2914 0B2E B4 08
2915 0B30
2916 0B30 2B C9
2917 0B32 EC
2918 0B33 22 C4
2919 0B35 75 04
2920 0B37 E2 F9
2921 0B39 EB 4D

```

PAGE

```

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

```

```

E6: MOV AX,@EQUIP_FLAG ; GET SENSE INFORMATION
PUSH AX ; SAVE IT
MOV AL,30H ; FORCE MONOCHROME TYPE
MOV @EQUIP_FLAG,AX ; INTO EQUIPMENT FLAG
SUB AX,AX ; MODE SET COMMAND FOR DEFAULT MODE
INT INT_VIDEO ; SEND INITIALIZATION TO B/W CARD
MOV AL,30H ; FORCE COLOR AT 80 BY 25
MOV @EQUIP_FLAG,AX ; INTO EQUIPMENT FLAG TO CLEAR BUFFERS
MOV AX,0003H ; AND INITIALIZATION COLOR CARD 80X25
INT INT_VIDEO ; MODE SET 80 X 25
MOV AX,0001H ; SET COLOR 40 X 25 MODE
INT INT_VIDEO ; SET DEFAULT COLOR MODE
POP AX ; RECOVER REAL SWITCH INFORMATION
MOV @EQUIP_FLAG,AX ; RESTORE IT
AND AL,30H ; ISOLATE VIDEO SWITCHES
JNZ ET ; VIDEO SWITCHES SET TO 0?
PUSH DS ; SAVE THE DATA SEGMENT
SUB AX,AX ; SET DATA SEGMENT TO 0
MOV DI,OFFSET @VIDEO_INT ; SET INTERRUPT 10H TO DUMMY
MOV WORD PTR [DI],OFFSET DUMMY ; RETURN ; RETURN IF NO VIDEO CARD
POP AX ; RESTORE REGISTERS
POP DS
JMP SHORT E18_1 ; BYPASS VIDEO TEST

```

```

ET: CMP AL,30H ; B/W CARD ATTACHED?
JNC E8 ; YES - SET MODE FOR B/W CARD
INC AH ; SET COLOR MODE FOR COLOR CARD
CMP AL,20H ; 80X25 MODE SELECTED?
JNE E8 ; NO - SET MODE FOR 40X25
MOV AH,3 ; SET MODE FOR 80X25

```

```

E8: XCHG AH,AL ; SAVE VIDEO MODE ON STACK
PUSH AX ; INITIALIZE TO ALPHANUMERIC MD
SUB AH,AH ; CALL VIDEO 10
INT INT_VIDEO ; RESTORE VIDEO SENSE SWITCHES IN AH
POP AX ; RESTORE VIDEO SENSE SWITCHES IN AH
PUSH AX ; SAVE VALUE
MOV BX,0B000H ; STARTING VIDEO MEMORY ADDRESS B/W CARD
MOV DX,3B8H ; MODE REGISTER FOR B/W
MOV CX,2048 ; MEMORY WORD COUNT FOR B/W CARD
CMP AH,30H ; B/W VIDEO CARD ATTACHED?
JE E9 ; YES - GO TEST VIDEO STORAGE
MOV BH,0B8H ; STARTING MEMORY ADDRESS FOR COLOR CARD
MOV DX,3D8H ; MODE REGISTER FOR COLOR CARD
MOV CH,20H ; MEMORY WORD COUNT FOR COLOR CARD

```

```

E9: MOV AL,@CRT_MODE_SET ; GET CURRENT MODE SET VALUE
AND AL,037H ; SET VIDEO BIT OFF
OUT DX,AL ; DISABLE VIDEO FOR COLOR CARD
MOV ES,BX ; POINT ES TO VIDEO MEMORY
MOV DS,BX ; POINT DS TO VIDEO MEMORY
ROR CX,1 ; DIVIDE BY 2 FOR WORD COUNT
CALL STGTEST_CNT ; GO TEST VIDEO READ/WRITE STORAGE
JNE E17 ; R/W MEMORY FAILURE - BEEP SPEAKER

```

```

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

```

```

MOV AL,22H ;
OUT MFG_PORT,AL ;
POP AX ; GET VIDEO SENSE SWITCHES (AH)
PUSH AX ; SAVE IT
MOV AH,0 ; ENABLE VIDEO AND SET MODE
INT INT_VIDEO ; VIDEO
MOV AX,7020H ; WRITE BLANKS IN REVERSE VIDEO
SUB DI,DI ; SETUP STARTING LOCATION
MOV CX,40 ; NUMBER OF BLANKS TO DISPLAY
REP STOSW ; WRITE VIDEO STORAGE

```

```

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

```

```

E11: POP AX ; GET VIDEO SENSE SWITCH INFORMATION
PUSH AX ; SAVE IT
CMP AH,30H ; B/W CARD ATTACHED?
MOV DX,03BAH ; SETUP ADDRESS OF B/W STATUS PORT
JE E11 ; YES - GO TEST LINES
MOV DX,03DAH ; COLOR CARD IS ATTACHED

```

```

E12: MOV AH,8
SUB CX,CX
IN AL,DX ; READ CRT STATUS PORT
AND E1,AH ; CHECK VIDEO/HORIZONTAL LINE
JNZ E13 ; ITS ON - CHECK IF IT GOES OFF
LOOP E13 ; LOOP UNTIL ON OR TIMEOUT
JMP SHORT E17 ; GO PRINT ERROR MESSAGE

```

SECTION 5


```
3036 0C00 81 26 0010 R FFCF      AND    @EQUIP_FLAG,OFFCFH      ; TURN OFF VIDEO BITS
3037 0C06 81 0E 0010 R 0010      OR     @EQUIP_FLAG,10H        ; SET COLOR 40X24
3038 0C0C 80 01                MOV    AL,01H
3039 0C0E 2A E4                SUB    AH,AH
3040 0C10 CD 10                INT    INT_VIDEO
3041 0C12                E17_1:
3042 0C12 58                POP    AX                    ; SET NEW VIDEO TYPE ON STACK
3043 0C13 A1 0010 R          MOV    AX,@EQUIP_FLAG
3044 0C16 24 30                AND    AL,30H
3045 0C18 3C 30                CMP    AL,30H                ; IS IT THE B/W?
3046 0C1A 2A C0                SUB    AL,AL
3047 0C1C 74 02                JZ     E17_2                ; GO IF YES
3048 0C1E FE C0                JC     AL                    ; INITIALIZE FOR 40X25
3049 0C20                E17_2:
3050 0C20 50                PUSH   AX
3051 0C21                E17_4:
3052 0C21 E9 0B4C R          JMP    E18
3053                ;----- BOTH VIDEO CARDS FAILED SET DUMMY RETURN IF RETRACE FAILURE
3054
3055                E17_3:
3056 0C24                PUSH   DS
3057 0C24 1E                AX,AX                        ; SET DS SEGMENT TO 0
3058 0C25 2B C0                SUB    DS,AX
3059 0C27 8E D8                MOV    DS,AX
3060 0C29 BF 0040 R          MOV    DI,OFFSET @VIDEO_INT ; SET INTERRUPT 10H TO DUMMY
3061 0C2C C7 05 0000 E      WORD PTR [DI],OFFSET DUMMY_RETURN ; RETURN IF NO VIDEO CARD
3062 0C30 1F                POP    DS
3063 0C31 E9 0B51 R          JMP    E18_1                ; BYPASS REST OF VIDEO TEST
```

SECTION 5

```

3064                                     PAGE
3065 -----
3066 | MANUFACTURING BOOT TEST CODE ROUTINE
3067 | LOAD A BLOCK OF TEST CODE THROUGH THE KEYBOARD PORT FOR MANUFACTURING
3068 | TESTS.
3069 | THIS ROUTINE WILL LOAD A TEST (MAX LENGTH=FAPFH) THROUGH THE KEYBOARD
3070 | PORT. CODE WILL BE LOADED AT LOCATION 0000:0500. AFTER LOADING,
3071 | CONTROL WILL BE TRANSFERRED TO LOCATION 0000:0500. THE STACK WILL
3072 | BE LOCATED AT 0000:0400. THIS ROUTINE ASSUMES THAT THE FIRST 2 BYTES
3073 | TRANSFERRED CONTAIN THE COUNT OF BYTES TO BE LOADED
3074 | (BYTE 1=COUNT LOW, BYTE 2=COUNT HI.)
3075 -----
3076
3077 |----- DEGATE ADDRESS LINE 20
3078
3079 OC34 MFG_BOOT:
3080 OC34 B4 DD MOV AH,DISABLE_BIT20 ; DEGATE COMMAND FOR ADDRESS LINE 20
3081 OC36 E8 0000 E CALL GATE_A20 ; ISSUE TO KEYBOARD ADAPTER AND CLI
3082
3083 |----- SETUP HARDWARE INTERRUPT VECTOR TABLE LEVEL 0-7 AND SOFTWARE INTERRUPTS
3084
3085 OC39 68 ---- R PUSH ABS0 ; SET ES SEGMENT REGISTER TO ABS0
3086 OC3C 07 POP ES ;
3087 OC3D B9 0018 MOV CX,24 ; GET VECTOR COUNT
3088 OC40 8C C8 MOV AX,C5 ; GET THE CURRENT CODE SEGMENT VALUE
3089 OC42 8E D8 MOV DS,AX ; SETUP DS SEGMENT REGISTER TO
3090 OC44 BE 0000 E MOV SI,OFFSET VECTOR_TABLE ; POINT TO THE ROUTINE ADDRESS TABLE
3091 OC47 BF 0020 R MOV DI,OFFSET *INT_PTR ; SET DESTINATION TO FIRST USED VECTOR
3092
3093 OC4A A5 MFG_B1: MOVSW ; MOVE ONE ROUTINE OFFSET ADDRESS
3094 OC4B AB STOSW ; INSERT CODE SEGMENT VALUE
3095 OC4C E2 FC LOOP MFG_B1 ; MOVE THE NUMBER OF ENTRIES REQUIRED
3096
3097 |----- SETUP HARDWARE INTERRUPT VECTORS LEVEL 8-15 (VECTORS START AT INT 10 H)
3098
3099 OC4E B9 0008 MOV CX,08 ; GET VECTOR COUNT
3100 OC51 BE 0000 E MOV SI,OFFSET SLAVE_VECTOR_TABLE ;
3101 OC54 BF 01C0 R MOV DI,OFFSET *SLAVE_INT_PTR ;
3102
3103 OC57 A5 MFG_B2: MOVSW ; MOVE ONE ROUTINE OFFSET ADDRESS
3104 OC58 AB STOSW ; INSERT CODE SEGMENT VALUE
3105 OC59 E2 FC LOOP MFG_B2 ;
3106
3107 |----- SET UP OTHER INTERRUPTS AS NECESSARY
3108
3109 ASSUME DS:ABS0,ES:ABS0
3110 OC5B 06 PUSH ES ; ES= ABS0
3111 OC5C 1F POP DS ; SET DS TO ABS0
3112 OC60 C7 06 0008 R 0000 E MOV WORD PTR *NMI_PTR,OFFSET NMI_INT ; NMI INTERRUPT
3113 OC63 C7 06 0014 R 0000 E MOV WORD PTR *INT5_PTR,OFFSET PRINT_SCREEN ; PRINT SCREEN
3114 OC69 C7 06 0062 R F600 MOV WORD PTR *BASIC_PTR+2,OF600H ; CASSETTE BASIC SEGMENT
3115
3116 |----- ENABLE KEYBOARD PORT
3117
3118 OC6F B0 60 MOV AL,60H ; WRITE 8042 MEMORY LOCATION 0
3119 OC71 E8 039D R CALL C8042 ; ISSUE THE COMMAND
3120 OC74 B0 09 MOV AL,00001001B ; SET INHIBIT OVERRIDE/ENABLE OBF
3121 OC76 E6 60 OUT PORT_A,AL ; INTERRUPT AND NOT PC COMPATIBLE
3122
3123 OC78 E8 0C9A R CALL MFG_B4 ; GET COUNT LOW
3124 OC7B 8A F8 MOV BH,AL ; SAVE IT
3125 OC7D E8 0C9A R CALL MFG_B4 ; GET COUNT HI
3126 OC80 8A E8 MOV CH,AL ;
3127 OC82 8A CF MOV CL,BH ; CX NOW HAS COUNT
3128 OC84 FC CLD ; SET DIRECTION FLAG TO INCREMENT
3129 OC85 BF 0500 R MOV DI,OFFSET *MFG_TEST_RTN ; SET TARGET OFFSET (DS=0000)
3130 OC88
3131 OC88 E4 64 MFG_B3: IN AL,STATUS_PORT ; GET 8042 STATUS PORT
3132 OC8A A8 01 TEST AL,OUT_BUF_FULL ; KEYBOARD REQUEST PENDING?
3133 OC8C 74 FA JZ MFG_B3 ; LOOP TILL DATA PRESENT
3134 OC8E E4 60 IN AL,PORT_A ; GET DATA
3135 OC90 AA STOSB ; STORE IT
3136 OC91 E6 80 OUT MFG_PORT,AL ; DISPLAY CHARACTER AT MFG PORT
3137 OC93 E2 F3 LOOP MFG_B3 ; LOOP TILL ALL BYTES READ
3138
3139 OC95 EA 0500 ---- R JMP *MFG_TEST_RTN ; FAR JUMP TO CODE THAT WAS JUST LOADED
3140
3141 OC9A MFG_B4:
3142 OC9A E4 64 IN AL,STATUS_PORT ; CHECK FOR OUTPUT BUFFER FULL
3143 OC9C A8 01 TEST AL,OUT_BUF_FULL ; HANG HERE IF NO DATA AVAILABLE
3144 OC9E E1 FA LOOPZ MFG_B4 ;
3145
3146 OCAD E4 60 IN RET AL,PORT_A ; GET THE COUNT
3147 OCA2 C3 RET
3148
3149 OCA3 POST1 ENDP
3150 OCA3 CODE ENDS
3151

```



```

343 0171 261 C7 06 006A 0000      MOV     ES,SS TEMP.BASE_LO_WORD,0
344 0178 261 C6 06 005C 00        MOV     BYTE PTR ES!(SS_TEMP.BASE_HI_BYTE),0
345 017E BE 0058                    MOV     SI,SS_TEMP
346 0181 8E D6                       MOV     SS,SI
347 0183 BC FFFD                    MOV     SP,MAX_SEQ_LEN-2
348
349
350
351 0186 6A 18                        PUSH   BYTE PTR RSDA_PTR      ; POINT TO DATA AREA
352 0188 1F                          POP     DS
353
354 0189 80 80                        MOV     AL,PARITY_CHECK       ; SET CHECK PARITY
355 018B E6 87                        OUT     DMA_PAGE+6,AL        ; SAVE WHICH CHECK TO USE
356
357
358
359 018D BB 0040                      MOV     AX,64                 ; STARTING AMOUNT OF MEMORY OK
360 0190 EB 09A0 R                    CALL   PRT_OK                 ; POST 64K OK MESSAGE
361
362
363
364 0193 80 B0B1                      ;----- GET THE MEMORY SIZE DETERMINED (PREPARE BX AND DX FOR BAD CMOS)
365 0196 EB 0000 E                    MOV     AX,(CMOS_U_M_S_LO+NMI)*H+CMOS_U_M_S_HI+NMI
366 0199 86 E0                        CALL   CMOS_READ              ; HIGH BYTE
367 019B EA 0000 E                    XCHG  AH,AL                  ; SAVE HIGH BYTE
368 019E 8B 1E 0013 R                CALL   CMOS_READ              ; LOW BYTE
369 01A2 8B D3                        MOV     BX,#MEMORY_SIZE       ; LOAD THE BASE MEMORY SIZE
370 01A4 03 D8                        MOV     DX,BX                 ; SAVE BASE MEMORY SIZE
371
372
373
374 01A6 B0 8E                        ;----- IS CMOS GOOD?
375 01A8 EB 0000 E                    MOV     AL,CMOS_DIAG+NMI     ; DETERMINE THE CONDITION OF CMOS
376
377 01AB A8 C0                        CALL   CMOS_READ              ; GET THE CMOS STATUS
378 01AD 74 02                        TEST   AL,BAD_BAT+BAD_CKSUM ; CMOS OK?
379 01AF EB 8B                        JZ     E20B0                  ; GO IF YES
380
381
382
383 01B1
384 01B1 8B 9596                      ;----- GET THE BASE 0->640K MEMORY SIZE FROM CONFIGURATION IN CMOS
385 01B4 EB 0000 E                    MOV     AX,(CMOS_B_M_S_LO+NMI)*H+CMOS_B_M_S_HI+NMI
386 01B7 24 3F                        CALL   CMOS_READ              ; HIGH BYTE
387 01B9 86 E0                        AND    AL,03FH                ; MASK OFF THE MANUFACTURING TEST BITS
388 01BB EB 0000 E                    XCHG  AH,AL                  ; SAVE HIGH BYTE
389 01BE 8B D0                        CALL   CMOS_READ              ; LOW BYTE
390 01C0 74 13                        CMP    DX,AX                  ; IS MEMORY SIZE GREATER THAN CONFIG?
391
392
393
394 01C2 50                          ;----- SET MEMORY SIZE DETERMINE NOT EQUAL TO CONFIGURATION
395 01C3 BB 8E8E                      MOV     AX,X*(CMOS_DIAG+NMI) ; SAVE AX
396 01C6 EB 0000 E                    CALL   CMOS_READ              ; ADDRESS THE STATUS BYTE
397 01C9 0C 10                        OR     AL,#MEM_SIZE           ; GET THE STATUS
398 01CB 86 C4                        XCHG  AL,AH                  ; SET CMOS FLAG
399 01CD EB 0000 E                    CALL   CMOS_WRITE             ; SAVE AL AND GET ADDRESS
400 01D0 58                          POP    AX                     ; WRITE UPDATED STATUS
401 01D1 8B D0                        CMP    DX,AX                  ; RESTORE AX
402 01D3 77 37                        JA     E20B1                  ; IS MEMORY SIZE GREATER THAN CONFIG ?
403 01D5
404 01D5 8B D8                      E20B1: MOV     BX,AX                 ; SET BASE MEMORY SIZE IN TOTAL REGISTER
405 01D7 8B D0                      MOV     DX,AX                 ; SAVE IN BASE SIZE REGISTER
406
407
408
409 01D9 8B 9798                      ;----- CHECK MEMORY SIZE ABOVE 640K FROM CONFIGURATION
410 01DC EB 0000 E                    MOV     AX,(CMOS_E_M_S_LO+NMI)*H+(CMOS_E_M_S_HI+NMI)
411 01DF 86 E0                        CALL   CMOS_READ              ; HIGH BYTE
412 01E1 EB 0000 E                    XCHG  AH,AL                  ; SAVE HIGH BYTE
413 01E4 8B C8                        CALL   CMOS_READ              ; LOW BYTE
414
415
416
417 01E6 8B B0B1                      ;----- ABOVE 640K SIZE FROM MEMORY SIZE DETERMINE
418 01E9 EB 0000 E                    MOV     AX,(CMOS_U_M_S_LO+NMI)*H+(CMOS_U_M_S_HI+NMI)
419 01EE EB 0000 E                    CALL   CMOS_READ              ; HIGH BYTE
420
421
422
423 01F1 8B C8                      ;----- WHICH IS GREATER - AX = MEMORY SIZE DETERMINE
424 01F3 74 0F                        ;----- CX = CONFIGURATION (ABOVE 640) BX = SIZE (BELOW 640)
425
426
427
428 01F5 50                          ;----- SET MEMORY SIZE DETERMINE NOT EQUAL TO CONFIGURATION
429 01F6 BB 8E8E                      MOV     AX,X*(CMOS_DIAG+NMI) ; SAVE AX
430 01F9 EB 0000 E                    CALL   CMOS_READ              ; ADDRESS THE STATUS BYTE
431 01FC 0C 10                        OR     AL,#MEM_SIZE           ; GET THE STATUS
432 01FE 86 C4                        XCHG  AL,AH                  ; SET CMOS FLAG
433 0200 EB 0000 E                    CALL   CMOS_WRITE             ; SAVE AL
434 0203 58                          POP    AX                     ; UPDATE STATUS BYTE
435
436
437 0204
438 0204 8B C8                      SET_MEM1: CMP    CX,AX                ; IS CONFIG GREATER THAN DETERMINED?
439 0206 77 02                        JA     SET_MEM                ; GO IF YES
440 0208 8B C8                      MOV     CX,AX                 ; USE MEMORY SIZE DETERMINE IF NOT
441 020A
442 020A 03 D9                      ADD    BX,CX                  ; SET TOTAL MEMORY SIZE
443 020C
444 020C 81 FA 0201                  E20C: CMP    DX,513             ; CHECK IF BASE MEMORY LESS 612K
445 0210 72 0D                        JB     NO_640                 ; GO IF YES
446
447
448 0212 8B 83B3                      MOV     AX,X*(CMOS_INFO128+NMI) ; SET 640K BASE MEMORY BIT
449 0215 EB 0000 E                    CALL   CMOS_READ              ; GET THE CURRENT STATUS
450 0218 0C 80                        CALL   AL,M640K               ; TURN ON 640K BIT IF NOT ALREADY ON
451 021A 86 C4                        XCHG  AL,AH                  ; SAVE THE CURRENT DIAGNOSTIC STATUS
452 021C EB 0000 E                    CALL   CMOS_WRITE             ; RESTORE THE STATUS
453
454
455
456 021F 89 1E 0017 R                NO_640: MOV     WORD PTR #KB_FLAG,BX ; SAVE TOTAL SIZE FOR LATER TESTING
457 0223 C1 EB 06                      SHR   BX,6                   ; DIVIDE BY 64
458 0226 49                          DEC   BX                      ; IS 64K ALREADY DONE
459 0227 C1 EA 06                      SHR   DX,6                   ; DIVIDE BY 64 FOR BASE
460

```



```

753 PAGE
754 -----
755 MEMORY ERROR REPORTING (R/W/ MEMORY OR PARITY ERRORS)
756
757 DESCRIPTION FOR ERRORS 201 (CMP ERROR OR PARITY)
758 OR 202 (ADDRESS LINE 0-15 ERROR)
759
760 "AABBCC DDEE 201" (OR 202)
761 AA=HIGH BYTE OF 24 BIT ADDRESS
762 BB=MIDDLE BYTE OF 24 BIT ADDRESS
763 CC=LOW BYTE OF 24 BIT ADDRESS
764 DD=HIGH BYTE OF XOR FAILING BIT PATTERN
765 EE=LOW BYTE OF XOR FAILING BIT PATTERN
766
767 DESCRIPTION FOR ERROR 202 (ADDRESS LINE 00-15)
768 A WORD OF FFFF IS WRITTEN AT THE FIRST WORD AND LAST WORD
769 OF EACH 64K BLOCK WITH ZEROS AT ALL OTHER LOCATIONS OF THE
770 BLOCK, A SCAN OF THE BLOCK IS MADE TO INSURE ADDRESS LINE
771 0-15 ARE FUNCTIONING.
772
773 DESCRIPTION FOR ERROR 203 (ADDRESS LINE 16-23)
774 AT THE LAST PASS OF THE STORAGE TEST, FOR EACH BLOCK OF
775 64K, THE CURRENT STORAGE SIZE (ID) IS WRITTEN AT THE FIRST
776 WORD OF EACH BLOCK. IT IS USED TO FIND ADDRESSING FAILURES.
777
778 "AABBCC DDEE 203"
779 SAME AS ABOVE EXCEPT FOR DDEE
780
781 GENERAL DESCRIPTION FOR BLOCK ID (DDEE WILL NOW CONTAINED THE ID)
782 DD=HIGH BYTE OF BLOCK ID
783 EE=LOW BYTE OF BLOCK ID
784
785 BLOCK ID ADDRESS RANGE
786 0000 000000 --> 00FFFF
787 0040 010000 --> 01FFFF
788 //
789 0200 090000 --> 09FFFF (512->576K) IF 640K BASE
790 100000 --> 10FFFF (1024->1088K) IF 512K BASE
791
792 EXAMPLE (640K BASE MEMORY + 512K I/O MEMORY = 1152K TOTAL)
793 NOTE: THE CORRECT BLOCK ID FOR THIS FAILURE IS 0280 HEX.
794 DUE TO AN ADDRESS FAILURE THE BLOCK ID+128K OVERLAYED
795 THE CORRECT BLOCK ID.
796
797 00640K OK <-- LAST OK MEMORY
798 10000 0300 202 <-- ERROR DUE TO ADDRESS FAILURE
799
800 IF A PARITY LATCH WAS SET THE CORRESPONDING MESSAGE WILL DISPLAY.
801
802 *PARITY CHECK 1* (OR 2)
803
804 DMA PAGE REGISTERS ARE USED AS TEMPORARY SAVE AREAS FOR SEGMENT
805 DESCRIPTOR VALUES.
806 -----
807
808 03BC SHUT3: CALL DDS ; ENTRY FROM PROCESSOR SHUTDOWN 3
809 03BC E8 0000 E ; SET REAL MODE DATA SEGMENT
810
811
812 03BF C6 06 0016 R 01 MOV #MFG_ERR_FLAG+1, MEM_FAIL ; CLEAR AND SET MANUFACTURING ERROR FLAG
813 03C4 B0 0D MOV AL, CR ; CARRIAGE RETURN
814 03C6 E8 0000 E CALL PRT_HEX ;
815 03C9 B0 0A MOV AL, LF ; LINE FEED
816 03CB E8 0000 E CALL PRT_HEX ;
817 03CE E4 84 IN AL, DMA_PAGE+3 ; GET THE HIGH BYTE OF 24 BIT ADDRESS
818 03D0 E8 0000 E CALL XPC_BYTE ; CONVERT AND PRINT CODE
819 03D2 E4 86 IN AL, DMA_PAGE+4 ; GET THE MIDDLE BYTE OF 24 BIT ADDRESS
820 03D5 E8 0000 E CALL XPC_BYTE ;
821 03D8 E4 88 IN AL, DMA_PAGE+5 ; GET THE LOW BYTE OF 24 BIT ADDRESS
822 03DA E8 0000 E CALL XPC_BYTE ;
823 03DD B0 20 MOV AL, ' ' ; SPACE TO MESSAGE
824 03DF E8 0000 E CALL PRT_HEX ;
825 03E2 E4 89 IN AL, DMA_PAGE+2 ; GET HIGH BYTE FAILING BIT PATTERN
826 03E4 E8 0000 E CALL XPC_BYTE ; CONVERT AND PRINT CODE
827 03E7 E4 82 IN AL, DMA_PAGE+1 ; GET LOW BYTE FAILING BIT PATTERN
828 03E9 E8 0000 E CALL XPC_BYTE ; CONVERT AND PRINT CODE
829
830
831
832 03EC E4 80 IN AL, MFG_PORT ; GET THE CHECKPOINT
833 03EE 3C 33 CMP AL, 33H ; IS IT AN ADDRESS FAILURE?
834 03F0 9E 0000 E MOV SI, OFFSET E203 ; LOAD ADDRESS ERROR 16->23
835 03F3 74 0A JZ ERR2 ; GO IF YES
836
837 03F5 BE 0000 E MOV SI, OFFSET E202 ; LOAD ADDRESS ERROR 00->15
838 03F6 3C 32 CMP AL, 32H ; GO IF YES
839 03FA 74 03 JZ ERR2
840
841 03FC BE 0000 E MOV SI, OFFSET E201 ; SETUP ADDRESS OF ERROR MESSAGE
842 03FF 03FF E8 0000 E CALL E_MSG ; PRINT ERROR MESSAGE
843 0402 E4 88 IN AL, DMA_PAGE+7 ; GET THE PORT_B VALUE
844
845
846
847
848 0404 A8 80 TEST AL, PARITY_CHECK ; CHECK FOR PLANAR ERROR
849 0406 74 08 JZ ; SKIP IF NOT
850
851 0408 80 PUSH AX ; SAVE STATUS
852 0409 EB 0990 R CALL PADING ; INSERT BLANKS
853 040C BE 0000 E MOV SI, OFFSET D1 ; PLANAR ERROR ADDRESS *PARITY CHECK 1*
854 040F E8 0000 E CALL P_MSG ; DISPLAY *PARITY CHECK 1* MESSAGE
855 0412 58 POP AX ; AND RECOVER STATUS
856 0413
857 0413 A8 40 TEST AL, IO_CHECK ; I/O PARITY CHECK ?
858 0415 74 09 JZ ; SKIP IF CORRECT ERROR DISPLAYED
859
860 0417 EB 0990 R CALL PADING ; INSERT BLANKS
861 041A BE 0000 E MOV SI, OFFSET D2 ; ADDRESS OF *PARITY CHECK 2* MESSAGE
862 041D E8 0000 E CALL P_MSG ; DISPLAY *PARITY CHECK 2* ERROR
863 0420
864 ; CONTINUE TESTING SYSTEM ....

```



```

1203          PAGE
1204          I----- SETUP KEYBOARD PARAMETERS
1205
1206          MOV     @INTR_FLAG,00H          ; SET STRAY INTERRUPT FLAG = 00
1207          BE     001E R                   ; SETUP KEYBOARD PARAMETERS
1208          MOV     @BUFFER_HEAD,S1
1209          MOV     @BUFFER_TAIL,S1
1210          MOV     @BUFFER_START,S1
1211          ADD     S1,32                   ; DEFAULT BUFFER OF 32 BYTES
1212          MOV     @BUFFER_END,S1
1213
1214          I----- SET PRINTER TIMEOUT DEFAULT
1215
1216          MOV     DI,OFFSET @PRINT_TIM_OUT; SET DEFAULT PRINTER TIMEOUT
1217          PUSH   DS
1218          POP     ES
1219          MOV     AX,1414H                 ; DEFAULT=20
1220          STOSW
1221          STOSW
1222
1223          I----- SET RS232 DEFAULT
1224
1225          MOV     AX,0101H                 ; RS232 DEFAULT=01
1226          STOSW
1227          STOSW
1228
1229          I----- ENABLE TIMER INTERRUPTS
1230
1231          IN     AL,INTA01
1232          AND     AL,0FEH                 ; ENABLE TIMER INTERRUPTS
1233          JMP     $+2                       ; I/O DELAY
1234          OUT    INTA01,AL
1235
1236          I----- CHECK CMOS BATTERY AND CHECKSUM
1237
1238          TEST    @MFG_TST,MFG_LOOP       ; MFG JUMPER?
1239          JNZ    B1_OK                     ; GO IF NOT
1240          JMP     F15C                     ; BYPASS IF YES
1241          B1_OK:
1242          MOV     AL,CMOS_DIAG+NMI        ; ADDRESS DIAGNOSTIC STATUS BYTE
1243          CALL    CMOS_READ               ; READ IT FROM CMOS
1244
1245          MOV     SI,OFFSET E161          ; LOAD BAD BATTERY MESSAGE 161
1246          TEST    AL,BAD_BAT              ; BATTERY BAD?
1247          JNZ    B1_ER                    ; DISPLAY ERROR IF BAD
1248
1249          MOV     SI,OFFSET E162          ; LOAD CHECKSUM BAD MESSAGE 162
1250          TEST    AL,BAD_CHKSUM+BAD_CONFIG ; CHECK FOR CHECKSUM OR NO DISKETTE
1251          JZ     B1_ER                     ; SKIP AND CONTINUE TESTING CMOS CLOCK
1252          B1_ER:
1253          CALL    E_MSG                    ; ELSE DISPLAY ERROR MESSAGE
1254          OR     BP,08000H                 ; FLAG "SET SYSTEM OPTIONS" DISPLAYED
1255          JMP     SHORT H_OK1A             ; SKIP CLOCK TESTING IF ERROR
1256
1257          I----- TEST CLOCK UPDATING
1258
1259          C_OK:  MOV     BL,04H             ; OUTER LOOP COUNT
1260          D_OK:  SUB     CX,CX              ; INNER LOOP COUNT
1261          E_OK:  MOV     AL,CMOS_REG_A+NMI  ; GET THE CLOCK UPDATE BYTE
1262          CALL    CMOS_READ
1263          TEST    AL,80H                  ; CHECK FOR UPDATE IN PROGRESS
1264          JNZ    G_OK                     ; GO IF YES
1265          LOOP   E_OK                      ; TRY AGAIN
1266          DEC    BL                        ; DEC OUTER LOOP
1267          JNZ    D_OK                      ; TRY AGAIN
1268          MOV     SI,OFFSET E163          ; PRINT MESSAGE
1269          CALL    E_MSG
1270
1271          I----- SET CMOS DIAGNOSTIC STATUS TO 04 (CLOCK ERROR)
1272
1273          MOV     AX,X*CMOS_DIAG+NMI      ; SET CLOCK ERROR
1274          CALL    CMOS_READ               ; GET THE CURRENT STATUS
1275          OR     AL,CMOS_CLK_FAIL         ; SET NEW STATUS
1276          XCHG   AL,AH                    ; GET STATUS ADDRESS AND SAVE NEW STATUS
1277          CALL    CMOS_WRITE              ; MOVE NEW DIAGNOSTIC STATUS TO CMOS
1278          JMP     SHORT H_OK              ; CONTINUE
1279
1280          I----- CHECK CLOCK UPDATE
1281
1282          G_OK:  MOV     CX,800             ; LOOP COUNT
1283          I_OK:  MOV     AL,CMOS_REG_A+NMI  ; CHECK FOR OPPOSITE STATE
1284          CALL    CMOS_READ
1285          TEST    AL,80H
1286          LOOPNZ I_OK                     ; TRY AGAIN
1287          JCXZ   F_OK                     ; PRINT ERROR IF TIMEOUT
1288
1289          I----- CHECK MEMORY SIZE DETERMINED = CONFIGURATION
1290
1291          H_OK:  MOV     AL,CMOS_DIAG+NMI   ; GET THE STATUS BYTE
1292          CALL    CMOS_READ
1293          TEST    AL,MEM_SIZE             ; WAS THE CONFIG= MEM_SIZE_DETERMINED?
1294          JZ     F_OK                      ; GO IF YES
1295
1296          I----- MEMORY SIZE ERROR
1297
1298          MOV     SI,OFFSET E164          ; PRINT SIZE ERROR
1299          CALL    E_MSG                    ; DISPLAY ERROR
1300
1301          I----- CHECK FOR CRT ADAPTER ERROR
1302
1303          H_OK1A: CMP     @MFG_ERR_FLAG,0CH ; CHECK FOR MONOCHROME CRT ERROR
1304          MOV     SI,OFFSET E401          ; LOAD MONOCHROME CRT ERROR
1305          JZ     H_OK1B                    ; GO IF YES
1306
1307          CMP     @MFG_ERR_FLAG,0DH       ; CHECK FOR COLOR CRT ADAPTER ERROR
1308          JNZ    J_OK                      ; CONTINUE IF NOT
1309          MOV     SI,OFFSET E501          ; CRT ADAPTER ERROR MESSAGE
1310          H_OK1B: MOV     ST,OFFSET E501
1311          CALL    E_MSG
1312          MOV     ST,OFFSET E501
    
```



```

1769 ;----- EXIT TO SYSTEM CODE
1770
1771
1772 0979 FF IE 0067 R CALL DWORD PTR @IO_ROM_INIT ; GO TO SYSTEM CODE
1773 ; VIA CALL THROUGH DATA AREA LOCATION
1774
1775 ;----- ENABLE NMI INTERRUPTS + ENTRY FROM SHUTDOWN WITH BOOT REQUEST
1776
1777 097D B0 0F SHUT4: MOV AL,CMOS_SHUT_DOWN ; ENABLE NMI AND READ DEFAULT ADDRESS
1778 097F E8 0000 E CALL CMOS_READ ; OPEN STANDBY LATCH
1779 0982 E4 61 IN AL,PORT_B ; ENABLE PARITY
1780 0984 24 F3 AND AL,RAM_PAR_ON ; ENABLE MEMORY PARITY CHECK / I/O CHECK
1781 0986 E6 61 OUT PORT_B,AL
1782
1783 0988 B0 43 MOV AL,43H ;
1784 098A E6 80 OUT MFG_PORT,AL ; <><> CHECKPOINT 43 <><>
1785 098C FB STI ; ENABLE INTERRUPTS IF DISABLED
1786
1787 098D CD 19 INT 19H ; GO TO BOOT LOADER
1788
1789 098F F4 HLT
1790
1791
1792 0990 PADING PROC NEAR ; INSERT PADDING
1793 0990 MOV CX,15 ; GET BLANK CHARACTER COUNT
1794 0992 PADI:
1795 0993 B0 20 MOV AL,' ' ; GET FILL SPACE
1796 0995 E8 0000 E CALL PRT_HEX ; WRITE A SPACE
1797 0998 E2 F9 LOOP PADT ; LOOP TILL INSERT DONE
1798 099A B0 2D MOV AL,'-' ; GET DASH CHARACTER
1799 099C E8 0000 E CALL PRT_HEX ; WRITE TO DISPLAY
1800 099F C3 RET
1801 09A0 PADING ENDP
1802
1803
1804 09A0 PRT_OK PROC NEAR ; PRINT "00000 KB OK"
1805 09A0 PUSH AX ; SAVE WORK REGISTER
1806 09A1 BB 000A MOV BX,10 ; SET DECIMAL CONVERT
1807
1808 ;----- CONVERT AND SAVE
1809
1810 09A4 B9 0005 MOV CX,5 ; OF 5 NIBBLES XX,XXX KB
1811 09A7 2B FF SUB DI,DI ; DISPLAY REGEN BUFFER POSITION
1812 09A9 PRT_DIV:
1813 09A9 33 D2 XOR DX,DX
1814 09AB F7 F3 DIV BX ; DIVIDE BY 10
1815 09AD 80 CA 30 OR DL,30H ; MAKE INTO ASCII
1816 09B0 52 PUSH DX ; SAVE
1817 09B1 E2 F6 LOOP PRT_DIV
1818
1819 ;----- DISPLAY LAST OK MEMORY
1820
1821 09B3 B9 0005 MOV CX,5
1822 09B6 PRT_DEC:
1823 09B6 58 POP AX ; RECOVER A NUMBER
1824 09B7 E8 0000 E CALL PROT_PRT_HEX ; POINT TO DISPLAY REGEN BUFFER
1825 09BA 47 INC DI
1826 09BB E2 F9 LOOP PRT_DEC ; POINT TO DISPLAY REGEN BUFFER
1827 09BD B9 0007 MOV CX,OFFSET F3B_PAD-OFFSET F3B ; LOAD MESSAGE LENGTH
1828 09C0 BE 09CF R MOV SI,OFFSET F3B ; POINT TO PRINT ' KB OK', ' MESSAGE
1829 09C3 PRT_LOOP:
1830 09C3 2E 8A 04 MOV AL,CS:[SI]
1831 09C6 46 INC SI
1832 09C7 E8 0000 E CALL PROT_PRT_HEX ; INCREMENT BUFF PTR
1833 09CA 47 INC DI
1834 09CB E2 F6 LOOP PRT_LOOP ; RECOVER WORK REGISTERS
1835 09CD 58 POP AX
1836 09CE C3 RET
1837
1838 09CF 20 4B 42 20 4F 4B F3B DB ' KB OK' ; OK MESSAGE
1839 09D5 20 F3B_OK DB ' ' ; PAD A SPACE
1840 = 09D6 F3B_PAD EQU $
1841
1842 09D6 PRT_OK ENDP
1843
1844 ;-----
1845 ; PRINTER TABLE ;
1846 ;-----
1847
1848 09D6 03BC F4 DW 03BCH ; ADDRESS OF MONOCHROME PARALLEL ADAPTER
1849 09D8 0378 DW 0378H ; BASE ADDRESS STANDARD PARALLEL ADAPTER
1850 09DA 0278 DW 0278H ; ADDRESS OF ALTERNATE PARALLEL ADAPTER
1851 09DC F4E LABEL WORD
1852
1853 09DC POST2 ENDP
1854 09DC CODE ENDS
1855 END
    
```

SECTION 5


```

1 PAGE 118,121
2 TITLE TEST4 ---- 06/10/85 POST AND BIOS UTILITY ROUTINES
3 .286C
4 .LIST
5 0000 CODE SEGMENT BYTE PUBLIC
6
7 PUBLIC BEEP
8 PUBLIC BLINK_INT
9 PUBLIC CMOS_READ
10 PUBLIC CMOS_WRITE
11 PUBLIC CONFIG_BAD
12 PUBLIC D11
13 PUBLIC DDS
14 PUBLIC DUMMY_RETURN_I
15 PUBLIC ERR_BEEP
16 PUBLIC E_MSG
17 PUBLIC INT_287
18 PUBLIC KBD_RESET
19 PUBLIC POST4
20 PUBLIC PROT_PRT_HEX
21 PUBLIC PROC_SHUTDOWN
22 PUBLIC PRT_HEX
23 PUBLIC PRT_MSG
24 PUBLIC P_MSG
25 PUBLIC RE_DIRECT
26 PUBLIC ROM_CHECK
27 PUBLIC ROM_CHECKSUM
28 PUBLIC SET_TOD
29 PUBLIC WAITF
30 PUBLIC XPC_BYTE
31
32 EXTRN E163:NEAR
33 EXTRN OBF_42:NEAR
34 EXTRN ROM_ERR:NEAR
35 EXTRN XMIT_8042:NEAR
36
37 ASSUME CS:CODE,DS:DATA
38
39 0000 POST4:
40 ;----- CMOS_READ -----
41 ; READ BYTE FROM CMOS SYSTEM CLOCK CONFIGURATION TABLE ;
42 ; INPUT: (AL) = CMOS TABLE ADDRESS TO BE READ ;
43 ; BIT 7 = 0 FOR NMI ENABLED AND 1 FOR NMI DISABLED ON EXIT ;
44 ; BITS 6-0 = ADDRESS OF TABLE LOCATION TO READ ;
45 ; OUTPUT: (AL) VALUE AT LOCATION (AL) MOVED INTO (AL). IF BIT 7 OF (AL) WAS ;
46 ; ON THEN NMI LEFT DISABLED. DURING THE CMOS READ BOTH NMI AND ;
47 ; NORMAL INTERRUPTS ARE DISABLED TO PROTECT CMOS DATA INTEGRITY. ;
48 ; THE CMOS ADDRESS REGISTER IS POINTED TO A DEFAULT VALUE AND ;
49 ; THE INTERRUPT FLAG RESTORED TO THE ENTRY STATE ON RETURN. ;
50 ; ONLY THE (AL) REGISTER AND THE NMI STATE IS CHANGED. ;
51 ;-----
52
53 0000 CMOS_READ PROC NEAR ; READ LOCATION (AL) INTO (AL)
54 0000 9C PUSHF ; SAVE INTERRUPT ENABLE STATUS AND FLAGS
55 0001 D0 C0 ROL AL,1 ; MOVE NMI BIT TO LOW POSITION
56 0003 F9 STC ; FORCE NMI BIT ON IN CARRY FLAG
57 0004 D0 D8 RCR AL,1 ; HIGH BIT ON TO DISABLE NMI - OLD IN CY
58 0006 FA CLI ; DISABLE INTERRUPTS
59 0007 E6 70 OUT CMOS_PORT,AL ; ADDRESS LOCATION AND DISABLE NMI
60 0009 90 NOP ; I/O DELAY
61 000A E4 71 IN AL,CMOS_DATA ; READ THE REQUESTED CMOS LOCATION
62 000C 50 PUSH AX ; SAVE (AH) REGISTER VALUE AND CMOS BYTE
63 000D B0 IE MOV AL,CMOS_SHUT_DOWN*2 ; GET ADDRESS OF DEFAULT LOCATION
64 000F D0 D8 RCR AL,1 ; PUT ORIGINAL NMI MASK BIT INTO ADDRESS
65 0011 E6 70 OUT CMOS_PORT,AL ; SET DEFAULT TO READ ONLY REGISTER
66 0013 90 NOP ; I/O DELAY
67 0014 E4 71 IN AL,CMOS_DATA ; OPEN STANDBY LATCH
68 0016 58 POP AX ; RESTORE (AH) AND (AL) = CMOS BYTE
69 0017 0E PUSH CS ; *PLACE CODE SEGMENT IN STACK AND
70 0018 E3 0039 R CALL CMOS_POPF ; *HANDLE POPF FOR B- LEVEL 80286
71 001B C3 RET ; RETURN WITH FLAGS RESTORED
72
73 001C CMOS_READ ENDP
74
75 ;----- CMOS_WRITE -----
76 ; WRITE BYTE TO CMOS SYSTEM CLOCK CONFIGURATION TABLE ;
77 ; INPUT: (AL) = CMOS TABLE ADDRESS TO BE WRITTEN TO ;
78 ; BIT 7 = 0 FOR NMI ENABLED AND 1 FOR NMI DISABLED ON EXIT ;
79 ; (AH) = NEW VALUE TO BE PLACED IN THE ADDRESSED TABLE LOCATION ;
80 ; OUTPUT: VALUE IN (AH) PLACED IN LOCATION (AL) WITH NMI LEFT DISABLED ;
81 ; IF BIT 7 OF (AL) IS ON. DURING THE CMOS UPDATE BOTH NMI AND ;
82 ; NORMAL INTERRUPTS ARE DISABLED TO PROTECT CMOS DATA INTEGRITY. ;
83 ; THE CMOS ADDRESS REGISTER IS POINTED TO A DEFAULT VALUE AND ;
84 ; THE INTERRUPT FLAG RESTORED TO THE ENTRY STATE ON RETURN. ;
85 ; ONLY THE CMOS LOCATION AND THE NMI STATE IS CHANGED. ;
86 ;-----
87
88 001C CMOS_WRITE PROC NEAR ; WRITE (AH) TO LOCATION (AL)
89 001C 9C PUSHF ; SAVE INTERRUPT ENABLE STATUS AND FLAGS
90 001D 50 PUSH AX ; SAVE WORK REGISTER VALUES
91 001E D0 C0 ROL AL,1 ; MOVE NMI BIT TO LOW POSITION
92 0020 F9 STC ; FORCE NMI BIT ON IN CARRY FLAG
93 0021 D0 D8 RCR AL,1 ; HIGH BIT ON TO DISABLE NMI - OLD IN CY
94 0023 FA CLI ; DISABLE INTERRUPTS
95 0024 E6 70 OUT CMOS_PORT,AL ; ADDRESS LOCATION AND DISABLE NMI
96 0026 8A C4 MOV AL,AH ; GET THE DATA BYTE TO WRITE
97 0028 E6 71 OUT CMOS_DATA,AL ; PLACE IN REQUESTED CMOS LOCATION
98 002A B0 IE MOV AL,CMOS_SHUT_DOWN*2 ; GET ADDRESS OF DEFAULT LOCATION
99 002C D0 D8 RCR AL,1 ; PUT ORIGINAL NMI MASK BIT INTO ADDRESS
100 002E E6 70 OUT CMOS_PORT,AL ; SET DEFAULT TO READ ONLY REGISTER
101 0030 90 NOP ; I/O DELAY
102 0031 E4 71 IN AL,CMOS_DATA ; OPEN STANDBY LATCH
103 0033 58 POP AX ; RESTORE WORK REGISTERS
104 0034 0E PUSH CS ; *PLACE CODE SEGMENT IN STACK AND
105 0036 E3 0039 R CALL CMOS_POPF ; *HANDLE POPF FOR B- LEVEL 80286
106 0038 C3 RET
107
108 0039 CMOS_WRITE ENDP
109
110 0039
111
112 0039

```

```

113 PAGE
114 0039 CMOS_POPF PROC NEAR ; POPF FOR LEVEL B- PARTS
115 0039 CF IRET ; RETURN FAR AND RESTORE FLAGS
116
117 003A CMOS_POPF ENDP
118
119
120 003A DDS PROC NEAR ; LOAD (DS) TO DATA AREA
121 003A 2E 1E 1E 0040 R MOV DS,CS;DSDATA ; PUT SEGMENT VALUE OF DATA AREA INTO DS
122 003F C3 RET ; RETURN TO USER WITH (DS)= DATA
123
124 0040 ---- R DSDATA DW DATA ; SEGMENT SELECTOR VALUE FOR DATA AREA
125
126 0042 DDS ENDP
127
128 ;----- E_MSG --- P_MSG --- THIS SUBROUTINE WILL PRINT A MESSAGE ON THE DISPLAY
129 ;
130 ; ENTRY REQUIREMENTS:
131 ; SI = OFFSET (ADDRESS) OF MESSAGE BUFFER
132 ; CX = MESSAGE BYTE COUNT
133 ; MAXIMUM MESSAGE LENGTH IS 36 CHARACTERS
134 ; BP = BIT 0=E161/E162, BIT 1=CONFIG_BAD, 2-15= FIRST MSG OFFSET
135 ;-----
136
137
138 0042 E_MSG PROC NEAR
139 0042 F7 C5 3FFF TEST BP,03FFFF ; CHECK FOR NOT FIRST ERROR MESSAGE
140 0046 75 08 JNZ E_MSG1 ; SKIP IF NOT FIRST ERROR MESSAGE
141
142 0048 56 PUSH SI ; SAVE MESSAGE POINTER
143 0049 81 E6 3FFF AND SI,03FFFF ; USE LOW 14 BITS OF MESSAGE OFFSET
144 004D 0B EE OR BP,SI ; AS FIRST ERROR MESSAGE FLAG
145 004F 5E POP SI ; (BIT 0 = E161/E162, BIT 1 = CONFIG_BAD)
146 0050
147 0050 EB 0069 R CALL P_MSG ; PRINT MESSAGE
148 0053 1E PUSH DS ; SAVE CALLERS (DS)
149 0054 E8 003A R CALL DDS ; POINT TO POST/BIOS DATA SEGMENT
150 0057 F6 06 0010 R 01 TEST BYTE PTR [EQUIP_FLAG,01H]; LOOP/HALT ON ERROR SWITCH ON ?
151 005C 74 02 JZ MFG_HALT ; YES - THEN GO TO MANUFACTURING HALT
152
153 005E 1F POP DS ; RESTORE CALLERS (DS)
154 005F C3 RET
155
156 0060 MFG_HALT: ; MANUFACTURING LOOP MODE ERROR TRAP
157 0060 FA CLI ; DISABLE INTERRUPTS
158 0061 A0 0015 R MOV AL,[MFG_ERR_FLAG] ; RECOVER ERROR INDICATOR
159 0064 E4 80 OUT MFG_PORT,AL ; SET INTO MANUFACTURING PORT
160 0066 F4 HLT ; HALT SYSTEM
161 0067 EB F7 JMP MFG_HALT ; HOT NMI TRAP
162
163 0069 E_MSG ENDP
164
165
166 0069 P_MSG PROC NEAR ; DISPLAY STRING FROM (CS:)
167 0069 2E 1E 8A 04 MOV AL,CS:[SI] ; PUT CHARACTER IN (AL)
168 006C 46 INC SI ; POINT TO NEXT CHARACTER
169 006D 50 PUSH AX ; SAVE PRINT CHARACTER
170 006E E8 012E R CALL PRT_HEX ; CALL VIDEO_IO
171 0071 58 POP AX ; RECOVER PRINT CHARACTER
172 0072 3C 0A CMP AL,LF ; WAS IT LINE FEED?
173 0074 75 F3 JNE P_MSG ; NO, KEEP PRINTING STRING
174 0076 C3 RET
175
176 0077 P_MSG ENDP
177
178 ;----- ERR_BEEP
179 ; THIS PROCEDURE WILL ISSUE LONG TONES (1-3/4 SECONDS) AND ONE OR
180 ; MORE SHORT TONES (9/32 SECOND) TO INDICATE A FAILURE ON THE
181 ; PLANAR BOARD, A BAD MEMORY MODULE, OR A PROBLEM WITH THE CRT.
182 ; ENTRY PARAMETERS:
183 ; DH = NUMBER OF LONG TONES TO BEEP.
184 ; DL = NUMBER OF SHORT TONES TO BEEP.
185 ;-----
186
187 0077 ERR_BEEP PROC NEAR
188 0077 9C PUSHF ; SAVE FLAGS
189 0078 FA CLI ; DISABLE SYSTEM INTERRUPTS
190 0079 0A F6 OR DH,DH ; ANY LONG ONES TO BEEP
191 007B 74 1E JZ G1 ; NO, DO THE SHORT ONES
192 007D ; LONG BEEPS
193 007D B3 70 MOV BL,112 ; COUNTER FOR LONG BEEPS (1-3/4 SECONDS)
194 007F B9 0500 MOV CX,1280 ; DIVISOR FOR 932 HZ
195 0082 E8 0085 R CALL BEEP ; DO THE BEEP
196 0085 B9 C233 MOV CX,49715 ; 2/3 SECOND DELAY AFTER LONG BEEP
197 0088 E8 00FB R CALL WAITF ; DELAY BETWEEN BEEPS
198 008B FE CE DEC DH ; ANY MORE LONG BEEPS TO DO
199 008D 76 EE JNZ G1 ; LOOP TILL DONE
200
201 008F 1E PUSH DS ; SAVE DS REGISTER CONTENTS
202 0090 E8 003A R CALL DDS ; MANUFACTURING TEST MODE?
203 0093 80 3E 0012 R 01 CMP [MFG_TST,01H]
204 0096 1F POP DS ; RESTORE ORIGINAL CONTENTS OF (DS)
205 0099 74 C5 JE MFG_HALT ; YES - STOP BLINKING LED
206
207 009B ; SHORT BEEPS
208 009B B3 12 MOV BL,18 ; COUNTER FOR A SHORT BEEP (9/32)
209 009D B9 048B MOV CX,1208 ; DIVISOR FOR 987 HZ
210 00A0 E8 0085 R CALL BEEP ; DO THE SOUND
211 00A3 B9 8178 MOV CX,33144 ; 1/2 SECOND DELAY AFTER SHORT BEEP
212 00A6 E8 00FB R CALL WAITF ; DELAY BETWEEN BEEPS
213 00A9 FE CA DEC DL ; DONE WITH SHORT BEEPS COUNT
214 00AB 76 EE JNZ G3 ; LOOP TILL DONE
215
216 00AD B9 8178 MOV CX,33144 ; 1/2 SECOND DELAY AFTER LAST BEEP
217 00B0 E8 00FB R CALL WAITF ; MAKE IT ONE SECOND DELAY BEFORE RETURN
218 00B3 9D POPF ; RESTORE FLAGS TO ORIGINAL SETTINGS
219 00B4 C3 RET ; RETURN TO CALLER
220
221 00B5 ERR_BEEP ENDP
    
```

SECTION 5

```

222 PAGE
223 ----- BEEP -----
224 | ROUTINE TO SOUND THE BEEPER USING TIMER 2 FOR TONE |
225 | ENTRY: |
226 | (BL) = DURATION COUNTER ( 1 FOR 1/64 SECOND ) |
227 | (CX) = FREQUENCY DIVISOR (1193180/FREQUENCY) (1331 FOR 886 HZ) |
228 | EXIT: |
229 | (AX),(BL),(CX) MODIFIED. |
230 -----
231
232 BEEP PROC NEAR ; SETUP TIMER 2
233 PUSHF ; SAVE INTERRUPT STATUS
234 CLI ; BLOCK INTERRUPTS DURING UPDATE
235 MOV AL,10110110B ; SELECT TIMER 2,LSB,MSB,BINARY
236 OUT TIMER+3,AL ; WRITE THE TIMER MODE REGISTER
237 JMP $+2 ; I/O DELAY
238 MOV AL,CL ; DIVISOR FOR HZ (LOW)
239 OUT TIMER+2,AL ; WRITE TIMER 2 COUNT - LSB
240 JMP $+2 ; I/O DELAY
241 MOV AL,CH ; DIVISOR FOR HZ (HIGH)
242 OUT TIMER+2,AL ; WRITE TIMER 2 COUNT - MSB
243 IN AL,PORT_B ; GET CURRENT SETTING OF PORT
244 OR AL,AL ; SAVE THAT SETTING
245 OR AL,GATE2+SPK2 ; GATE TIMER 2 AND TURN SPEAKER ON
246 OUT PORT_B,AL ; AND RESTORE INTERRUPT STATUS
247 POPF
248
249 GT: MOV CX,1035 ; 1/64 SECOND PER COUNT (BL)
250 CALL WAITF ; DELAY COUNT FOR 1/64 OF A SECOND
251 DEC BL ; GO TO BEEP DELAY 1/64 COUNT
252 JNZ GT ; (BL) LENGTH COUNT EXPIRED?
253 ; NO - CONTINUE BEEPING SPEAKER
254
255 BEEP PROC NEAR ; SETUP TIMER 2
256 PUSHF ; SAVE INTERRUPT STATUS
257 CLI ; BLOCK INTERRUPTS DURING UPDATE
258 MOV AL,PORT_B ; GET CURRENT PORT VALUE
259 OR AL,NOT (GATE2+SPK2) ; ISOLATE CURRENT SPEAKER BITS IN CASE
260 AND AH,AL ; SOMEONE TURNED THEM OFF DURING BEEP
261 MOV AL,AH ; RECOVER VALUE OF PORT
262 AND AL,NOT (GATE2+SPK2) ; FORCE SPEAKER DATA OFF
263 OUT PORT_B,AL ; AND STOP SPEAKER TIMER
264 POPF
265 MOV CX,1035 ; RESTORE INTERRUPT FLAG STATE
266 CALL WAITF ; FORCE 1/64 SECOND DELAY (SHORT)
267 PUSHF ; MINIMUM DELAY BETWEEN ALL BEEPS
268 CLI ; SAVE INTERRUPT STATUS
269 IN AL,PORT_B ; BLOCK INTERRUPTS DURING UPDATE
270 OR AL,GATE2+SPK2 ; GET CURRENT PORT VALUE IN CASE
271 AND AH,AL ; SOMEONE TURNED THEM ON
272 OUT PORT_B,AL ; RECOVER VALUE OF PORT_B
273 POPF ; RESTORE SPEAKER STATUS
274 RET ; RESTORE INTERRUPT FLAG STATE
275
276 BEEP ENDP
277
278 ----- WAITF -----
279 | FIXED TIME WAIT ROUTINE (HARDWARE CONTROLLED - NOT PROCESSOR) |
280 | ENTRY: |
281 | (CX) = COUNT OF 15.085737 MICROSECOND INTERVALS TO WAIT |
282 | MEMORY REFRESH TIMER 1 OUTPUT USED AS REFERENCE |
283 | EXIT: |
284 | AFTER (CX) TIME COUNT (PLUS OR MINUS 16 MICROSECONDS) |
285 | (CX) = 0 |
286 -----
287
288 WAITF PROC NEAR ; DELAY FOR (CX)*15.085737 US
289 PUSH AX ; SAVE WORK REGISTER (AH)
290
291 WAITF1: IN AL,PORT_B ; USE TIMER 1 OUTPUT BITS
292 AND AL,REFRESH_BIT ; READ CURRENT COUNTER OUTPUT STATUS
293 CMP AL,AH ; MASK FOR REFRESH DETERMINE BIT
294 JE WAITF1 ; DID IT JUST CHANGE
295 ; WAIT FOR A CHANGE IN OUTPUT LINE
296
297 MOV AH,AL ; SAVE NEW FLAG STATE
298 LOOP WAITF1 ; DECREMENT HALF CYCLES TILL COUNT END
299
300 POP AX ; RESTORE (AH)
301 RET ; RETURN (CX) = 0
302
303 WAITF ENDP
304
305 ----- CONFIG_BAD -----
306 | SET CMOS_DIAG WITH CONFIG ERROR BIT (WITH NMI DISABLED) |
307 | (BP) BIT 14 SET ON TO INDICATE CONFIGURATION ERROR |
308 -----
309
310 CONFIG_BAD PROC NEAR
311 PUSH AX
312 MOV AX,X*(CMOS_DIAG+NMI) ; ADDRESS CMOS DIAGNOSTIC STATUS BYTE
313 CALL CMOS_READ ; GET CURRENT VALUE
314 OR AL,BAD_CONFIG ; SET BAD CONFIGURATION BIT
315 XCHG AH,AL ; SETUP FOR WRITE
316 CALL CMOS_WRITE ; UPDATE CMOS WITH BAD CONFIGURATION
317 POP AX
318 OR BP,04000H ; SET CONFIGURATION BAD FLAG IN (BP)
319 RET
320
321 CONFIG_BAD ENDP

```

```

321
322
323
324
325
326
327
328
329
330 011E
331 011E 50
332 011F C0 E8 04
333 0122 E8 0128 R
334 0125 58
335 0126 24 0F
336
337
338 0128
339 0128 04 90
340 012A 27
341 012B 14 40
342 012D 27
343
344 012E
345 012E B4 0E
346 0130 B7 00
347 0132 CD 10
348 0134 C3
349
350 0135
351 0135
352 0135
353
354
355
356
357
358
359 0135
360 0135 BA C6
361 0137 E8 011E R
362 013A BA C2
363 013C E8 011E R
364 013F B0 30
365 0141 E8 012E R
366 0144 B0 20 012E R
367 0146 E8 012E R
368 0149 C3
369
370 014A
371
372
373
374
375
376
377
378
379
380 014A
381 014A 06
382 014B 87
383 014C D1 E7
384
385
386
387 014E 6A 20
388 0150 07
389 0151 AA
390 0152 4F
391
392
393
394 0153 6A 30
395 0155 07
396 0156 AA
397 0157 4F
398 0158 6A 38
399 015A 07
400 015B AA
401 015C 4F
402
403
404
405 015D 6A 28
406 015F 07
407 0160 53
408 0161 82
409 0162 51
410 0163 32 C9
411 0165 BA 03DA
412 0168 93
413 0169
414 0169 EC
415 016A A8 09
416 016C E1 FB
417 016E 93
418 016F AA
419
420 0170 59
421 0171 5A
422 0172 5B
423 0173 5F
424 0174 07
425 0175 C3
426
427 0176
    
```

```

PAGE
-----
; XPC_BYTE -- XLATE_PR -- PRT_HEX -----
;
; CONVERT AND PRINT ASCII CODE CHARACTERS
;
; AL CONTAINS NUMBER TO BE CONVERTED.
; AX AND BX DESTROYED.
;
-----
XPC_BYTE PROC NEAR
; DISPLAY TWO HEX DIGITS
PUSH AX
; SAVE FOR LOW NIBBLE DISPLAY
SHR AL,4
; NIBBLE SWAP
CALL XLAT_PR
; DO THE HIGH NIBBLE DISPLAY
POP AX
; RECOVER THE NIBBLE
AND AL,0FH
; ISOLATE TO LOW NIBBLE
; FALL INTO LOW NIBBLE CONVERSION
XLAT_PR PROC NEAR
; CONVERT 00-0F TO ASCII CHARACTER
ADD AL,090H
; ADD FIRST CONVERSION FACTOR
ADC AL,040H
; ADD CONVERSION AND ADJUST LOW NIBBLE
DAA
; ADJUST HIGH NIBBLE TO ASCII RANGE
PRT_HEX PROC NEAR
; DISPLAY CHARACTER IN (AL) COMMAND
MOV AH,0EH
MOV BH,0
INT 10H
; CALL VIDEO_IO
RET
PRT_HEX ENDP
XLAT_PR ENDP
XPC_BYTE ENDP
ENDP
;----- PRT_SEG -----
; PRINT A SEGMENT VALUE TO LOOK LIKE A 21 BIT ADDRESS
; DX MUST CONTAIN SEGMENT VALUE TO BE PRINTED
;
-----
PRT_SEG PROC NEAR
; GET MSB
MOV AL,DH
; DISPLAY SEGMENT HIGH BYTE
CALL XPC_BYTE
; LSB
MOV AL,DL
; DISPLAY SEGMENT LOW BYTE
CALL XPC_BYTE
; PRINT A '0 '
MOV AL,'0'
; TO MAKE LOOK LIKE ADDRESS
CALL PRT_HEX
; ADD ENDING SPACE
MOV AL,' '
CALL PRT_HEX
RET
PRT_SEG ENDP
;----- PROT_PRT_HEX -----
; PUT A CHARACTER TO THE DISPLAY BUFFERS WHEN IN PROTECTED MODE
;
; (AL)= ASCII CHARACTER
; (DI)= DISPLAY REGEN BUFFER POSITION
;
-----
PROT_PRT_HEX PROC NEAR
; SAVE CURRENT SEGMENT REGISTERS
PUSH ES
PUSH DI
SAL DI,1
; MULTIPLY OFFSET BY TWO
;----- MONOCHROME VIDEO CARD -----
; GET MONOCHROME BUFFER SEGMENT SELECTOR
PUSH BYTE PTR C_BWCRT_PTR
POP ES
; LOAD SEGMENT SELECTOR
STOSB
; PLACE CHARACTER IN BUFFER
DEC DI
; ADJUST POINTER BACK
;----- ENHANCED GRAPHICS ADAPTER -----
; ENHANCED COLOR DISPLAY POINTER LOW 64K
PUSH BYTE PTR E_CCRT_PTR
POP ES
; LOAD SEGMENT SELECTOR
STOSB
; PLACE CHARACTER IN BUFFER
DEC DI
; ADJUST POINTER BACK
; ENHANCED COLOR DISPLAY POINTER HI 64K
PUSH BYTE PTR E_CCRT_PTR2
POP ES
; LOAD SEGMENT SELECTOR
STOSB
; PLACE CHARACTER IN BUFFER
DEC DI
; ADJUST POINTER BACK
;----- COMPATIBLE COLOR -----
; SET (DS) TO COMPATIBLE COLOR MEMORY
PUSH BYTE PTR C_CCRT_PTR
POP DS
; LOAD SEGMENT SELECTOR
POP ES
; SAVE WORK REGISTERS
PUSH BX
PUSH DX
PUSH CX
; TIMEOUT LOOP FOR "BAD" HARDWARE
XOR CX,CX
; STATUS ADDRESS OF COLOR CARD
MOV DX,03DAH
; SAVE IN (BX) REGISTER
XCHG AX,BX
PROT_S1 IN AL,DX
; GET COLOR CARD STATUS
TEST AL,RVRT+RHRZ
; CHECK FOR VERTICAL RETRACE (OR HORZ)
LOOPZ PROT_S
; TIMEOUT LOOP TILL FOUND
XCHG AX,BX
; RECOVER CHARACTERS
STOSB
; PLACE CHARACTER IN BUFFER
POP CX
POP DX
POP BX
POP DI
POP ES
RET
; RESTORE REGISTERS
PROT_PRT_HEX ENDP
    
```

SECTION 5

```

428 PAGE
429 -----
430 ROM CHECKSUM SUBROUTINE
431 -----
432
433 0176 ROM_CHECKSUM PROC NEAR
434 0176 2B C9 SUB CX,CX ; NUMBER OF BYTES TO ADD IS 64K
435
436 0178 ROM_CHECKSUM_CNT: ; ENTRY FOR OPTIONAL ROM TEST
437 0179 32 C0 XOR AL,AL
438 017A ADD AL,[BX] ; GET (DS:BX)
439 017A 02 07 INC BX ; POINT TO NEXT BYTE
440 017C 42 LOOP ROM_L ; ADD ALL BYTES IN ROM MODULE
441 017D E2 FB
442
443 017F 0A C0 OR AL,AL ; SUM = 0?
444 0181 C3 RET
445
446 0182 ROM_CHECKSUM ENDP
447
448 -----
449 THIS ROUTINE CHECKS OPTIONAL ROM MODULES AND
450 IF CHECKSUM IS OK, CALLS INITIALIZATION/TEST CODE IN MODULE
451 -----
452
453 0182 ROM_CHECK PROC NEAR
454 0182 B8 ---- R MOV AX,DATA ; POINT ES TO DATA AREA
455 0185 9E C0 MOV ES,AX
456 0187 2A E4 SUB AH,AH ; ZERO OUT AH
457 0189 5A 47 02 MOV AL,[BX+2] ; GET LENGTH INDICATOR
458 018C C1 E0 09 SHL AX,9 ; MULTIPLY BY 512
459 018F 9B C8 MOV CX,AX ; SET COUNT
460 0191 C1 E8 04 SHR AX,4 ; SET POINTER TO NEXT MODULE
461 0194 03 D0 ADD DX,AX ; DO CHECKSUM
462 0196 E8 0178 R CALL ROM_CHECKSUM_CNT
463 0199 74 05 JZ ROM_CHECK_1
464
465 019B E8 0000 E CALL ROM_ERR ; POST CHECKSUM ERROR
466 019E EB 13 JMP SHORT ROM_CHECK_END ; AND EXIT
467
468 01A0 ROM_CHECK_1:
469 01A0 52 PUSH DX ; SAVE POINTER
470 01A1 26: C7 06 0067 R 0003 MOV ES:010 ROM_INIT,0003H ; LOAD OFFSET
471 01A8 26: BC 1E 0069 R MOV ES:010 ROM_SEG,DS ; LOAD SEGMENT
472 01AD 26: FF 1E 0067 R CALL DWORD PTR ES:010 ROM_INIT; CALL INITIALIZE/TEST ROUTINE
473 01B2 5A POP DX
474
475 01B3 ROM_CHECK_END:
476 01B3 C3 RET ; RETURN TO CALLER
477
478 01B4 ROM_CHECK ENDP
479
480 -----
481 KBD RESET
482 -----
483 THIS PROCEDURE WILL SEND A SOFTWARE RESET TO THE KEYBOARD.
484 SCAN CODE 0AAH SHOULD BE RETURNED TO THE PROCESSOR.
485 SCAN CODE 065H IS DEFINED FOR MANUFACTURING TEST
486 -----
487
488 01B4 KBD_RESET PROC NEAR
489 01B4 B0 FF MOV AL,OFFH ; SET KEYBOARD RESET COMMAND
490 01B6 E8 0000 E CALL XMIT_8042 ; GO ISSUE THE COMMAND
491 01B9 E3 29 JCXZ G13 ; EXIT IF ERROR
492
493 01BB 3C FA CMP AL,KB_ACK
494 01BD 75 1F JNZ G13
495
496 01BF 80 FD MOV AL,0FDH ; ENABLE KEYBOARD INTERRUPTS
497 01C1 E6 21 OUT INTA01,AL ; WRITE 8259 INTERRUPT MASK REGISTER
498 01C3 C6 06 006B R 00 MOV INTR_FLAG,0 ; RESET INTERRUPT INDICATOR
499 01C8 FB STI ; ENABLE INTERRUPTS
500 01C9 B3 0A MOV BL,10 ; TRY FOR 400 MILLISECONDS
501 01CB 2B C9 SUB CX,CX ; SETUP INTERRUPT TIMEOUT COUNT
502 01CD F6 06 006B R 02 G11: TEST 0INTR_FLAG,02H ; DID A KEYBOARD INTERRUPT OCCUR ?
503 01D2 75 06 JNZ G12 ; YES - READ SCAN CODE RETURNED
504 01D4 E2 F7 LOOP G11 ; NO - LOOP TILL TIMEOUT
505
506 01D6 FE CB DEC BL ; TRY AGAIN
507 01D8 75 F3 JNZ G11
508 01DA E4 60 G12: IN AL,PORT_A ; READ KEYBOARD SCAN CODE
509 01DC 8A D8 MOV BL,AL ; SAVE SCAN CODE JUST READ
510 01DE G13: RET ; RETURN TO CALLER
511 01DE C3
512
513 01DF KBD_RESET ENDP
514
515 -----
516 BLINK LED PROCEDURE FOR MFG RUN-IN TESTS
517 IF LED IS ON, TURN IT OFF. IF OFF, TURN ON.
518 -----
519
520 01DF BLINK_INT PROC NEAR
521 01DF FB STI ; SAVE AX REGISTER CONTENTS
522 01E0 80 PUSH AX ; READ CURRENT VALUE OF MFG_PORT
523 01E1 E4 80 IN AL,MFG_PORT ; FLIP CONTROL BIT
524 01E3 34 40 XOR AL,01000000H
525 01E5 E4 80 OUT MFG_PORT,AL
526 01E7 B0 20 MOV AL,B0 ; RESTORE AX REGISTER
527 01E9 E6 20 OUT INTA00,AL
528 01EB 58 POP AX
529 01EC CF IRET
530
531 01ED BLINK_INT ENDP

```

```

532 PAGE
533 -----
534 THIS ROUTINE INITIALIZES THE TIMER DATA AREA IN THE ROM BIOS
535 DATA AREA. IT IS CALLED BY THE POWER ON ROUTINES. IT CONVERTS
536 HR:MIN:SEC FROM CMOS TO TIMER TICS. IF CMOS IS INVALID, TIMER
537 IS SET TO ZERO.
538
539 INPUT NONE PASSED TO ROUTINE BY CALLER
540 CMOS LOCATIONS USED FOR TIME
541
542 OUTPUT @TIMER_LOW
543 @TIMER_HIGH
544 @TIMER_OFL
545 ALL REGISTERS UNCHANGED
546 -----
547 COUNTS_SEC EQU 18 ; TIMER DATA CONVERSION EQUATES
548 COUNTS_MIN EQU 1092
549 COUNTS_HOUR EQU 7 ; 65543 - 65536
550 UPDATE_TIMER EQU 10000000B ; RTC UPDATE IN PROCESS BIT MASK
551
552 SET_TOD PROC NEAR
553 PUSH DS
554 CALL DDSE ; ESTABLISH SEGMENT
555 MOV AX,AX
556 MOV @TIMER_OFL,AL ; RESET TIMER ROLL OVER INDICATOR
557 MOV @TIMER_LOW,AX ; AND TIMER COUNT
558 MOV @TIMER_HIGH,AX
559 MOV AL,CMOS_DIAG+NM1 ; CHECK CMOS VALIDITY
560 CALL CMOS_READ ; READ DIAGNOSTIC LOCATION IN CMOS
561 AND BAD_BAT+BAD_CKSUM+CMOS_CLK_FAIL ; BAD BATTERY, CKSUM ERROR, CLOCK ERROR
562 ; CMOS NOT VALID -- TIMER SET TO ZERO
563
564 JNZ POD_DONE
565 SUB CX,CX
566
567 UIP: MOV AL,CMOS_REG_A+NM1 ; ACCESS REGISTER A
568 CALL CMOS_READ ; READ CMOS CLOCK REGISTER A
569 TEST AL,UPDATE_TIMER ; WAIT TILL UPDATE BIT IS ON
570 LOOPZ UIP
571
572 JCXZ POD_DONE ; CMOS CLOCK STUCK IF TIMEOUT
573 SUB CX,CX
574
575 UIPOFF: MOV AL,CMOS_REG_A+NM1 ; ACCESS REGISTER A
576 CALL CMOS_READ ; READ CMOS CLOCK REGISTER A
577 TEST AL,UPDATE_TIMER ; NEXT WAIT TILL END OF UPDATE
578 LOOPNZ UIPOFF
579
580 JCXZ POD_DONE ; CMOS CLOCK STUCK IF TIMEOUT
581
582 MOV AL,CMOS_SECONDS+NM1 ; TIME JUST UPDATED
583 CALL CMOS_READ ; ACCESS SECONDS VALUE IN CMOS
584 CMP AL,59H ; ARE THE SECONDS WITHIN LIMITS?
585 JAE TOD_ERROR ; GO IF NOT
586
587 CALL CVT_BINARY ; CONVERT IT TO BINARY
588 MOV CX,AX ; MOVE COUNT TO ACCUMULATION REGISTER
589 SHR CX,2 ; ADJUST FOR SYSTEMATIC SECONDS ERROR
590 MOV BL,COUNTS_SEC ; COUNT FOR SECONDS
591 MUL BL
592 ADD CX,AX
593 MOV AL,CMOS_MINUTES+NM1 ; ACCESS MINUTES VALUE IN CMOS
594 CALL CMOS_READ ; ARE THE MINUTES WITHIN LIMITS?
595 CMP AL,59H ; GO IF NOT
596 JAE TOD_ERROR ; CONVERT IT TO BINARY
597 CALL CVT_BINARY ; SAVE MINUTES COUNT
598 MOV AX,AX ; ADJUST FOR SYSTEMATIC MINUTES ERROR
599 SHR AX,1 ; ADD ADJUSTMENT TO COUNT
600 ADD CX,AX ; RECOVER BCD MINUTES VALUE
601 POP AX
602 MOV BX,COUNTS_MIN ; COUNT FOR MINUTES
603 MUL BX ; ADD TO ACCUMULATED VALUE
604 ADD CX,AX
605 MOV AL,CMOS_HOURS+NM1 ; ACCESS HOURS VALUE IN CMOS
606 CALL CMOS_READ ; ARE THE HOURS WITHIN LIMITS?
607 CMP AL,23H ; GO IF NOT
608 JAE TOD_ERROR ; CONVERT IT TO BINARY
609 CALL CVT_BINARY
610 MOV DX,AX
611 MOV BL,COUNTS_HOUR ; COUNT FOR HOURS
612 MUL BL
613 ADD AX,CX
614 ADC DX,0000H
615 MOV @TIMER_HIGH,DX
616 MOV @TIMER_LOW,AX
617
618 POD_DONE: POP DS
619 RET
620
621 TOD_ERROR: POP DS ; RESTORE SEGMENT
622 POPA ; RESTORE REGISTERS
623 MOV SI,OFFSET E163 ; DISPLAY CLOCK ERROR
624 CALL E_MSG
625 MOV AX,X* (CMOS_DIAG+NM1) ; SET CLOCK ERROR IN STATUS
626 CALL CMOS_READ ; READ DIAGNOSTIC CMOS LOCATION
627 OR AL,CMOS_CLK_FAIL ; SET NEW STATUS WITH CMOS CLOCK ERROR
628 XCHG AL,AH ; MOVE NEW STATUS TO WORK REGISTER
629 CALL CMOS_WRITE ; UPDATE STATUS LOCATION
630 RET
631
632 SET_TOD ENDP
633
634 CVT_BINARY PROC NEAR
635 MOV AH,AL ; UNPACK 2 BCD DIGITS IN AL
636 SHR AH,4 ; RESULT IS IN AX
637 AND AL,0FH ; CONVERT UNPACKED BCD TO BINARY
638 AAD
639 RET
640
641 CVT_BINARY ENDP
642
643
644

```

SECTION 5

```

645 PAGE
646 ----- D11 -- INT ?? H -- ( IRQ LEVEL ?? ) -----
647 | TEMPORARY INTERRUPT SERVICE ROUTINE FOR POST |
648 | |
649 | THIS ROUTINE IS ALSO LEFT IN PLACE AFTER THE POWER ON DIAGNOSTICS |
650 | TO SERVICE UNUSED INTERRUPT VECTORS. LOCATION "INTR_FLAG" WILL |
651 | CONTAIN EITHER: |
652 | 1) LEVEL OF HARDWARE INTERRUPT THAT CAUSED CODE TO BE EXECUTED, OR |
653 | 2) "FF" FOR A NON-HARDWARE INTERRUPT THAT WAS EXECUTED ACCIDENTALLY. |
654 |-----|
655
656 0291 D11 PROC NEAR
657 0291 50 PUSH AX ; SAVE REGISTER AX CONTENTS
658 0292 53 PUSH BX
659 0293 80 0B MOV AL,0BH ; READ IN-SERVICE REGISTER
660 0294 E6 20 OUT INTA00,AL ; (FIND OUT WHAT LEVEL BEING
661 0295 EB 00 JMP $+2 ; SERVICED)
662 0296 E4 20 IN AL,INTA00 ; GET LEVEL
663 0297 8A E0 MOV AH,AL ; SAVE IT
664 0298 0A C4 OR AL,AH ; 00? (NO HARDWARE ISR ACTIVE)
665 0299 75 04 JNZ HW_INT
666
667 02A1 B4 FF MOV AH,0FFH
668 02A2 EB 2F JMP SHORT SET_INTR_FLAG ; SET FLAG TO "FF" IF NON-HARDWARE
669 02A5
670 02A5 B0 0B HW_INT: MOV AL,0BH ; READ IN-SERVICE REGISTER FROM
671 02A7 E6 A0 OUT INTB00,AL ; INTERRUPT CHIP #2
672 02A8 EB 00 JMP $+2 ; I/O DELAY
673 02AB E4 A0 IN AL,INTB00 ; CHECK THE SECOND INTERRUPT CHIP
674 02AD 8A F8 MOV BH,AL ; SAVE IT
675 02AF 0A FF OR BH,BH
676 02B1 74 10 JZ NOT_SEC ; CONTINUE IF NOT
677
678 02B3 E4 A1 IN AL,INTB01 ; GET SECOND INTERRUPT MASK
679 02B5 0A CT OR AL,BH ; MASK OFF LEVEL BEING SERVICED
680 02B7 EB 00 JMP $+2 ; I/O DELAY
681 02B9 E6 A1 OUT INTB01,AL
682 02BB E0 20 MOV AL,E01 ; SEND E01 TO SECOND CHIP
683 02BD EB 00 JMP $+2 ; I/O DELAY
684 02BF E6 A0 OUT INTB00,AL
685 02C1 EB 0D JMP SHORT IS_SEC
686 02C3
687 02C3 E4 21 NOT_SEC: IN AL,INTA01 ; GET CURRENT MASK VALUE
688 02C5 EB 00 JMP $+2 ; I/O DELAY
689 02C7 80 E4 FB AND AH,0FBH ; DO NOT DISABLE SECOND CONTROLLER
690 02CA 0A C4 OR AL,AH ; MASK OFF LEVEL BEING SERVICED
691 02CC E6 21 OUT INTA01,AL ; SET NEW INTERRUPT MASK
692 02CE EB 00 JMP $+2 ; I/O DELAY
693 02D0
694 02D0 B0 20 IS_SEC: MOV AL,E01
695 02D2 E6 20 OUT INTA00,AL
696 02D4
697 02D4 5B SET_INTR_FLAG: POP BX ; RESTORE (BX) FROM STACK
698 02D5 1E PUSH DS ; SAVE ACTIVE (DS)
699 02D6 E8 003A R CALL DDS ; SET DATA SEGMENT
700 02D9 88 26 006B R MOV DI,INTR_FLAG,AH ; SET FLAG
701 02DB 1F POP DS
702 02DE 58 POP AX ; RESTORE REGISTER AX CONTENTS
703 02DF DUMMY_RETURN: IRET ; NEED IRET FOR VECTOR TABLE
704 02DF CF
705
706 02E0 D11 ENDP
707
708 -----
709 | --- HARDWARE INT ?I H -- ( IRQ LEVEL ? ) -- TO INT 0A H -----
710 | REDIRECT SLAVE INTERRUPT 9 TO INTERRUPT LEVEL 2 |
711 | THIS ROUTINE FIELDS LEVEL 9 INTERRUPTS AND |
712 | CONTROL IS PASSED TO MASTER INTERRUPT LEVEL 2 |
713 |-----|
714 02E0 RE_DIRECT PROC NEAR
715 02E0 50 PUSH AX ; SAVE (AX)
716 02E1 B0 20 MOV AL,E01
717 02E3 E6 A0 OUT INTB00,AL ; E01 TO SLAVE INTERRUPT CONTROLLER
718 02E5 58 POP AX ; RESTORE (AX)
719 02E6 CD 0A INT 0AH ; GIVE CONTROL TO HARDWARE LEVEL 2
720
721 02E8 CF IRET ; RETURN
722
723 02E9 RE_DIRECT ENDP
724
725 -----
726 | --- HARDWARE INT ?I H -- ( IRQ LEVEL ? ) -----
727 | SERVICE X287 INTERRUPTS |
728 | THIS ROUTINE FIELDS X287 INTERRUPTS AND CONTROL |
729 | IS PASSED TO THE NMI INTERRUPT HANDLER FOR |
730 | COMPATIBILITY. |
731 |-----|
732 02E9 INT_287 PROC NEAR
733 02E9 50 PUSH AX ; SAVE (AX)
734 02EA 32 C0 XOR AL,AL
735 02EC E6 F0 OUT X287,AL ; REMOVE THE INTERRUPT REQUEST
736
737 02EE B0 20 MOV AL,E01 ; ENABLE THE INTERRUPT
738 02F0 E6 A0 OUT INTB00,AL ; THE SLAVE
739 02F2 E6 20 OUT INTA00,AL ; THE MASTER
740 02F4 58 POP AX ; RESTORE (AX)
741 02F5 CD 02 INT 02H ; GIVE CONTROL TO NMI
742
743 02F7 CF IRET ; RETURN
744
745 02F8 INT_287 ENDP
746
747 02F8 PROC_SHUTDOWN PROC ; COMMON 80286 SHUTDOWN WAIT
748
749 02F8 B0 FE MOV AL,SHUT_CMD ; SHUTDOWN COMMAND
750 02FA E6 64 OUT STATUS_PORT,AL ; SEND TO KEYBOARD CONTROL PORT
751 02FC
752 02FC F4 PROC_S: HLT ; WAIT FOR 80286 RESET
753 02FD EB FD JMP PROC_S ; INSURE HALT
754
755 02FF PROC_SHUTDOWN ENDP
756 02FF ENDS
757 CODE ENDS

```

```

1 PAGE 118,121
2 TITLE TEST5 ---- 06/10/85 EXCEPTION INTERRUPT TEST HANDLERS
3 .286C
4 .LIST
5 0000 SEGMENT BYTE PUBLIC
6 PUBLIC POSTS
7 PUBLIC SYSINIT1
8
9
10 -----
11 | EXCEPTION INTERRUPT ROUTINE |
12 |-----|
13
14 ASSUME CS:CODE,DS:ABS0
15
16 POSTS:
17 EXC_00:
18 0000 MOV AL,90H ; GO TEST IF EXCEPTION WAS EXPECTED
19 0002 E9 00B2 R JMP TEST_EXC
20 0005 B0 91 ; GO TEST IF EXCEPTION WAS EXPECTED
21 0007 E9 00B2 R JMP TEST_EXC
22 000A ; GO TEST IF EXCEPTION WAS EXPECTED
23 000A B0 92 JMP TEST_EXC
24 000C E9 00B2 R ; GO TEST IF EXCEPTION WAS EXPECTED
25 000F ; GO TEST IF EXCEPTION WAS EXPECTED
26 000F B0 93 JMP TEST_EXC
27 0011 E9 00B2 R ; GO TEST IF EXCEPTION WAS EXPECTED
28 0014 ; GO TEST IF EXCEPTION WAS EXPECTED
29 0014 B0 94 JMP TEST_EXC
30 0016 E9 00B2 R ; GO TEST IF EXCEPTION WAS EXPECTED
31 0019 ; GO TEST IF EXCEPTION WAS EXPECTED
32 0019 06 PUSH ES ; LOAD ES REGISTER WITH SELECTOR
33 001A 6A 48 PUSH BYTE PTR ES_TEMP
34 001C 07 POP ES
35
36 |-----|
37 |----| FIX BOUND PARAMETERS
38 SUB DI,DI ; POINT BEGINNING OF THE BLOCK
39 001F 261 CT 05 0000 MOV WORD PTR ES:[DI],0 ; SET FIRST WORD TO ZERO
40 0024 261 CT 45 02 7FFF MOV WORD PTR ES:[DI+2],07FFF ; SET SECOND TO 07FFF
41 002A 07 POP ES
42 002B B0 95 ; GO TEST IF EXCEPTION WAS EXPECTED
43 002D E9 00B2 R JMP TEST_EXC
44
45 EXC_06:
46 0030 MOV AL,96H ; GO TEST IF EXCEPTION WAS EXPECTED
47 0032 EB 7E JMP SHORT TEST_EXC
48 0034 ; GO TEST IF EXCEPTION WAS EXPECTED
49 0034 B0 97 JMP SHORT TEST_EXC
50 0036 EB 7A ; GO TEST IF EXCEPTION WAS EXPECTED
51 0038 ; GO TEST IF EXCEPTION WAS EXPECTED
52 0038 B0 98 JMP SHORT TEST_EXC
53 003A EB 76 ; GO TEST IF EXCEPTION WAS EXPECTED
54 003C ; GO TEST IF EXCEPTION WAS EXPECTED
55 003C B0 99 JMP SHORT TEST_EXC
56 003E EB 72 ; GO TEST IF EXCEPTION WAS EXPECTED
57 0040 ; GO TEST IF EXCEPTION WAS EXPECTED
58 0040 B0 9A JMP SHORT TEST_EXC
59 0042 EB 6E ; GO TEST IF EXCEPTION WAS EXPECTED
60 0044 ; GO TEST IF EXCEPTION WAS EXPECTED
61 0044 B0 9B JMP SHORT TEST_EXC
62 0046 EB 6A ; GO TEST IF EXCEPTION WAS EXPECTED
63 0048 ; GO TEST IF EXCEPTION WAS EXPECTED
64 0048 B0 9C JMP SHORT TEST_EXC
65 004A EB 66 ; GO TEST IF EXCEPTION WAS EXPECTED
66 004C ; GO TEST IF EXCEPTION WAS EXPECTED
67 004C B0 9D JMP SHORT TEST_EXC
68 004E EB 62 ; GO TEST IF EXCEPTION WAS EXPECTED
69 0050 ; GO TEST IF EXCEPTION WAS EXPECTED
70 0050 B0 9E JMP SHORT TEST_EXC
71 0052 EB 5E ; GO TEST IF EXCEPTION WAS EXPECTED
72 0054 ; GO TEST IF EXCEPTION WAS EXPECTED
73 0054 B0 9F JMP SHORT TEST_EXC
74 0056 EB 5A ; GO TEST IF EXCEPTION WAS EXPECTED
75 0058 ; GO TEST IF EXCEPTION WAS EXPECTED
76 0058 B0 A0 JMP SHORT TEST_EXC
77 005A EB 56 ; GO TEST IF EXCEPTION WAS EXPECTED
78 005C ; GO TEST IF EXCEPTION WAS EXPECTED
79 005C B0 A1 JMP SHORT TEST_EXC
80 005E EB 52 ; GO TEST IF EXCEPTION WAS EXPECTED
81 0060 ; GO TEST IF EXCEPTION WAS EXPECTED
82 0060 B0 A2 JMP SHORT TEST_EXC
83 0062 EB 4E ; GO TEST IF EXCEPTION WAS EXPECTED
84 0064 ; GO TEST IF EXCEPTION WAS EXPECTED
85 0064 B0 A3 JMP SHORT TEST_EXC
86 0066 EB 4A ; GO TEST IF EXCEPTION WAS EXPECTED
87 0068 ; GO TEST IF EXCEPTION WAS EXPECTED
88 0068 B0 A4 JMP SHORT TEST_EXC
89 006A EB 46 ; GO TEST IF EXCEPTION WAS EXPECTED
90 006C ; GO TEST IF EXCEPTION WAS EXPECTED
91 006C B0 A5 JMP SHORT TEST_EXC
92 006E EB 42 ; GO TEST IF EXCEPTION WAS EXPECTED
93 0070 ; GO TEST IF EXCEPTION WAS EXPECTED
94 0070 B0 A6 JMP SHORT TEST_EXC
95 0072 EB 3E ; GO TEST IF EXCEPTION WAS EXPECTED
96 0074 ; GO TEST IF EXCEPTION WAS EXPECTED
97 0074 B0 A7 JMP SHORT TEST_EXC
98 0076 EB 3A ; GO TEST IF EXCEPTION WAS EXPECTED
99 0078 ; GO TEST IF EXCEPTION WAS EXPECTED
100 0078 B0 A8 JMP SHORT TEST_EXC
101 007A EB 36 ; GO TEST IF EXCEPTION WAS EXPECTED
102 007C ; GO TEST IF EXCEPTION WAS EXPECTED
103 007C B0 A9 JMP SHORT TEST_EXC
104 007E EB 32 ; GO TEST IF EXCEPTION WAS EXPECTED
105 0080 ; GO TEST IF EXCEPTION WAS EXPECTED
106 0080 B0 AA JMP SHORT TEST_EXC
107 0082 EB 2E ; GO TEST IF EXCEPTION WAS EXPECTED
108 0084 ; GO TEST IF EXCEPTION WAS EXPECTED
109 0084 B0 AB JMP SHORT TEST_EXC
110 0086 EB 2A ; GO TEST IF EXCEPTION WAS EXPECTED
111 0088 ; GO TEST IF EXCEPTION WAS EXPECTED
112 0088 B0 AC JMP SHORT TEST_EXC
113 008A EB 26 ; GO TEST IF EXCEPTION WAS EXPECTED
114

```

SECTION 5


```

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

```

```

435 PAGE
436 |----- (POST_TR)
437 TR_LOC:
438 DW 00800H ; SEGMENT LIMIT
439 DW 0C000H ; SEGMENT BASE ADDRESS - LOW WORD
440 DB 0 ; SEGMENT BASE ADDRESS - HIGH BYTE
441 DB FREE_TSS ; ACCESS RIGHTS BYTE
442 DW 0 ; RESERVED - MUST BE ZERO
443
444 |----- (POST_TSS_PTR)
445
446 DW 00800H ; SEGMENT LIMIT
447 DW TR_LOC ; SEGMENT BASE ADDRESS - LOW WORD
448 DB 0 ; SEGMENT BASE ADDRESS - HIGH BYTE
449 DB CPLD_DATA_ACCESS ; ACCESS RIGHTS BYTE
450 DW 0 ; RESERVED - MUST BE ZERO
451
452 |----- (POST_LDTR)
453 LDT_LOC:
454 DW GDT_LEN ; SEGMENT LIMIT
455 DW 0D00H ; SEGMENT BASE ADDRESS - LOW WORD
456 DB 0 ; SEGMENT BASE ADDRESS - HIGH BYTE
457 DB LDT_DESC ; ACCESS RIGHTS BYTE
458 DW 0 ; RESERVED - MUST BE ZERO
459
460 |----- (POST_LDT_PTR)
461
462 DW GDT_LEN ; SEGMENT LIMIT
463 DW LDT_LOC ; SEGMENT BASE ADDRESS - LOW WORD
464 DB 0 ; SEGMENT BASE ADDRESS - HIGH BYTE
465 DB CPLD_DATA_ACCESS ; ACCESS RIGHTS BYTE
466 DW 0 ; RESERVED - MUST BE ZERO
467
468 EQU $
469 GDT_DATA_END EQU $
470 |----- END OF PRE-ALLOCATED GDT
471
472 |----- ENTRY POINTS FOR THE FIRST 32 SYSTEM INTERRUPTS
473
474 |-----
475 SYS_IDT_OFFSETS LABEL WORD
476
477 DW OFFSET EXC_00 ; INTERRUPTS AS DEFINED
478 DW OFFSET EXC_01 ; EXCPT 00 - DIVIDE ERROR
479 DW OFFSET EXC_02 ; EXCPT 01 - SINGLE STEP
480 DW OFFSET EXC_03 ; EXCPT 02 - NMI, SYSTEM REQUEST FOR D
481 DW OFFSET EXC_04 ; EXCPT 03 - BREAKPOINT
482 DW OFFSET EXC_05 ; EXCPT 04 - INTO DETECT
483 DW OFFSET EXC_06 ; EXCPT 05 - BOUND
484 DW OFFSET EXC_07 ; EXCPT 06 - INVALID OPCODE
485 DW OFFSET EXC_08 ; EXCPT 07 - PROCESSOR EXT NOT AVAIL
486 DW OFFSET EXC_09 ; EXCPT 08 - DOUBLE EXCEPTION
487 DW OFFSET EXC_10 ; EXCPT 09 - PROCESSOR EXT SEGMENT ERR
488 DW OFFSET EXC_11 ; EXCPT 10 - TSS BAD IN GATE TRANSFER
489 DW OFFSET EXC_12 ; EXCPT 11 - SEGMENT NOT PRESENT
490 DW OFFSET EXC_13 ; EXCPT 12 - STACK SEGMENT NOT PRESENT
491 DW OFFSET EXC_14 ; EXCPT 13 - GENERAL PROTECTION
492 DW OFFSET EXC_15 ;
493 DW OFFSET EXC_16 ; EXCPT 16 - PROCESSOR EXTENSION ERROR
494 DW OFFSET EXC_17 ;
495 DW OFFSET EXC_18 ;
496 DW OFFSET EXC_19 ;
497 DW OFFSET EXC_20 ;
498 DW OFFSET EXC_21 ;
499 DW OFFSET EXC_22 ;
500 DW OFFSET EXC_23 ;
501 DW OFFSET EXC_24 ;
502 DW OFFSET EXC_25 ;
503 DW OFFSET EXC_26 ;
504 DW OFFSET EXC_27 ;
505 DW OFFSET EXC_28 ;
506 DW OFFSET EXC_29 ;
507 DW OFFSET EXC_30 ;
508 DW OFFSET EXC_31 ;
509
510 |----- FORMAT INTERRUPT DESCRIPTORS (GATES) 32 - 255
511
512 FREE_INTS DW OFFSET IRET_ADDR ; DESTINATION OFFSET
513 DW SYS_ROM_CS ; DESTINATION SEGMENT
514 DB 0,INT_GATE ; UNUSED AND ACCESS RIGHTS BYTE
515
516 SIDT_BLD ENDP
517 CODE ENDS
518 END
    
```

SECTION 5

```

1      PAGE 118,121
2      TITLE TEST6 ---- 06/10/85 POST TESTS AND SYSTEM BOOT STRAP
3      .286C
4      .LIST
5      0000      CODE      SEGMENT BYTE PUBLIC
6
7      PUBLIC   BOOT_STRAP_1
8      PUBLIC   POSTS
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    EG02:NEAR
17     EXTRN    ERR_BEEP:NEAR
18     EXTRN    E_MSG:NEAR
19     EXTRN    FSA:NEAR
20     EXTRN    PRT_SEG:NEAR
21
22     ASSUME   CS:CODE,DS:DATA
23
24     0000      PROC      NEAR
25
26     ;-----
27     ; THIS SUBROUTINE PERFORMS A READ/WRITE STORAGE TEST ON A BLOCK ;
28     ; OF STORAGE. ;
29     ; ENTRY REQUIREMENTS: ;
30     ; ES = ADDRESS OF STORAGE SEGMENT BEING TESTED ;
31     ; DS = ADDRESS OF STORAGE SEGMENT BEING TESTED ;
32     ; CX = WORD COUNT OF STORAGE BLOCK TO BE TESTED ;
33     ; EXIT PARAMETERS: ;
34     ; ZERO FLAG = 0 IF STORAGE ERROR (DATA COMPARE OR PARITY ;
35     ; CHECK). AL=0 DENOTES A PARITY CHECK, ELSE AL=XOR'ED ;
36     ; BIT PATTERN OF THE EXPECTED DATA PATTERN VS THE ACTUAL ;
37     ; DATA READ. ;
38     ; AX,BX,CX,DX,DI, AND SI ARE ALL DESTROYED. ;
39     ;-----
40     0000      MOV      BX,CX          ; SAVE WORD COUNT OF BLOCK TO TEST
41     0002 84 D9      IN      AL,PORT_B
42     0004 0C 0C      OR      AL,RAM_PAR_OFF      ; TOGGLE PARITY CHECK LATCHES
43     0006 E6 61      OUT     PORT_B,AL      ; TO RESET ANY PENDING ERROR
44     0008 24 F3      AND     AL,RAM_PAR_ON
45     000A E6 61      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 89 09 10   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     0018 89 15      MOV     [DI],DX      ; STORE DATA PATTERN
57     001A 8B 05      MOV     AX,[DI]      ; GET THE DATA WRITTEN
58     001C 33 C2      XOR     AX,DX        ; INSURE DATA AS EXPECTED (CLEAR CARRY)
59     001E E1 F6      LOOPZ  C1           ; LOOP TILL DONE OR ERROR
60
61     0020 75 66      JNZ    C13          ; EXIT IF ERROR
62
63     ;----- 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     AL,[DI],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 88 35      MOV     [DI],DH      ; WRITE A BYTE OF FFH AT EVEN LOCATION
74     0034 86 F2      XCHG   DH,DL        ; SET DX= 000FFH AND BUS SETTLE
75     0036 8B 05      MOV     AX,[DI]      ; GET THE DATA
76     0038 33 C2      XOR     AX,DX        ; CHECK THE FIRST WRITTEN
77     003A 75 4C      JNZ    C13          ; EXIT IF NOT
78
79     ;----- 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 84 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 8B 00 00   MOV     AX,0         ; RESTORE AX TO 0000
89     0047 75 3F      JNZ    C13          ; EXIT IF PARITY ERROR
90
91     0049 BA AA55    MOV     DX,0AA55H    ; WRITE THE INITIAL DATA PATTERN
92     004C
93     C3:           SUB     DI,DI         ; START AT BEGINNING OF BLOCK
94     004E 2B F6      SUB     SI,SI         ; INITIALIZE DESTINATION POINTER
95     0050 8B C8      MOV     CX,BX        ; SETUP BYTE COUNT FOR LOOP
96     0052 8B C2      MOV     AX,DX        ; GET THE PATTERN
97     0054 F3 AB      REP   STOSW          ; STORE 4AK BYTES (32K WORDS)
98     0056 8B C8      MOV     CX,BX        ; SET COUNT
99     0058 2B F6      SUB     SI,SI         ; START AT BEGINNING
100    005A
101    C6:           LODSW          ; GET THE FIRST WRITTEN
102    005B 33 C2      XOR     AX,DX        ; INSURE DATA AS EXPECTED
103    005D E1 FB      LOOPZ  C6           ; LOOP TILL DONE OR ERROR
104
105    005F 75 27      JNZ    C13          ; EXIT IF NOT EXPECTED (ERROR BITS ON)
106
107     ;----- 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

```



```

343          ;----- CLEAR #BOOT_LOCN
344          XOR    AX,AX
345 017E 32 C0    MOV    CX,256          ; CLEAR 256 WORDS
346 0180 B9 0100 MOV    DI,OFFSET #BOOT_LOCN
347 0183 BF 7C00 R REP    STOSW
348 0186 F3/ AB
349          ;----- LOAD SYSTEM FROM DISKETTE -- CX HAS RETRY COUNT
350
351          STI
352 0188 FB      ; SET RETRY COUNT
353 0189 B9 0004 MOV    CX,4          ; IPL SYSTEM
354 018C 51      ; RESET THE DISKETTE SYSTEM
355 018D B4 00   H1:  MOV    AH,0
356 018F CD 13   INT    13H          ; DISKETTE_IO
357 0191 T2 0F   JC     H2            ; IF ERROR, TRY AGAIN
358
359 0193 B8 0201 MOV    AX,201H       ; READ IN THE SINGLE SECTOR
360 0196 2B D2   SUB    DX,DX         ; TO THE BOOT LOCATION
361 0198 BE C2   MOV    ES,DX
362 019A B8 7C00 R MOV    BX,OFFSET #BOOT_LOCN ; DRIVE 0, HEAD 0
363 019D B9 0001 MOV    CX,1          ; SECTOR 1, TRACK 0
364 01A0 CD 13   INT    13H          ; DISKETTE_IO
365 01A2 59      ; RECOVER RETRY COUNT
366 01A3 73 09   JNC    H4           ; CARRY FLAG SET BY UNSUCCESSFUL READ
367 01A5 80 FC 80 CMP    AH,80H       ; IF TIME OUT, NO RETRY
368 01A8 74 22   JZ     H5           ; TRY FIXED DISK
369 01AA E2 E0   LOOP  H1           ; DO IT FOR RETRY TIMES
370 01AC EB 1E   JMP    SHORT H5     ; TRY FIXED DISK
371
372          ;----- BOOT RECORD READ SUCCESSFUL
373          ; INSURE FIRST BYTE OF LOADED BOOT RECORD IS VALID (NOT ZERO)
374
375 01AE 80 3E 7C00 R 06 H4:  CMP    BYTE PTR #BOOT_LOCN,06H ; CHECK FOR FIRST INSTRUCTION INVALID
376 01B3 72 71   JB     H1           ; IF BOOT NOT VALID PRINT MESSAGE HALT
377
378          ;----- INSURE DATA PATTERN FIRST 8 WORDS NOT ALL EQUAL
379
380 01B5 BF 7C00 R   MOV    DI,OFFSET #BOOT_LOCN ; CHECK DATA PATTERN
381 01B8 B9 0008   MOV    CX,8          ; CHECK THE NEXT 8 WORDS
382 01BB A1 7C00 R   MOV    AX,WORD PTR #BOOT_LOCN
383
384 01BE 83 C7 02 H4A:  ADD    DI,2          ; POINT TO NEXT LOCATION
385 01C1 3B 02   CMP    AX,[DI]       ; CHECK DATA PATTERN FOR A FILL PATTERN
386 01C3 E1 F9   LOOPZ H4A           ;
387 01C6 74 5F   JZ     H1           ; BOOT NOT VALID PRINT MESSAGE HALT
388
389 01C7 EA 7C00 ---- R H4_A:  JMP    #BOOT_LOCN
390
391          ;----- ATTEMPT BOOTSTRAP FROM FIXED DISK
392
393 01CC B0 44     H5:  MOV    AL,044H       ;
394 01CE E6 80     OUT    MFG_PORT,AL ;
395          ASSUME DS:DATA ;
396 01D0 E8 0000 E CALL  D05           ;
397 01D3 F6 06 00B8 R 01 TEST  #ALASTRATE,DUAL ; FLOPPY/FIXED DISK CARD INSTALLED
398          ASSUME DS:ABS0 ;
399 01D8 B8 ---- R   MOV    AX,ABS0       ; ESTABLISH ADDRESSING
400 01DB BE 08     MOV    DS,AX         ;
401 01DD 74 3D     JZ     H9           ; GO IF NOT
402
403          ;----- CHECK FOR FIXED DISK INITIALIZATION ERROR
404
405 01DF B0 0E     MOV    AL,CMOS_DIAG ; GET POST POWER ON STATUS (NMI ENABLED)
406 01E1 E8 0000 E CALL  CMOS_READ    ; FROM DIAGNOSTIC STATUS BYTE
407 01E4 A8 00     TEST   AL,#FF_FAIL ; DID WE HAVE A FIXED DISK FAILURE?
408 01E6 75 34     JNZ    H9           ; GO IF YES
409
410 01E8 2B CD     SUB    AX,AX         ; RESET DISKETTE
411 01EA 2B D2     SUB    DX,DX
412 01EC CD 13     INT    13H          ;
413 01EE B9 0003   MOV    CX,3          ; RETRY COUNT
414 01F1
415 01F1 51      H6:  PUSH   CX           ; SAVE RETRY COUNT
416 01F2 BA 0080   MOV    DX,0080H     ; FIXED DISK ZERO
417 01F5 B8 0201   MOV    AX,0201H     ; READ IN A SINGLE SECTOR
418 01F8 2B D8     SUB    BX,BX
419 01FA 8E C3     MOV    ES,BX
420 01FC B8 7C00 R   MOV    BX,OFFSET #BOOT_LOCN ; TO THE BOOT LOCATION
421 01FF B9 0001   MOV    CX,1          ; SECTOR 1, TRACK 0
422 0202 CD 13     INT    13H          ; FILE I/O CALL
423 0204 59      POP    CX           ; RECOVER RETRY COUNT
424 0208 72 08     JC     H8           ;
425 020T 81 3E TDFE R AA55 CMP    WORD PTR #BOOT_LOCN+510D,0AA55H ; TEST FOR GENERIC BOOT BLOCK
426 020D 74 B6     JZ     H4_A
427
428 020F 51      H8:  PUSH   CX           ; FIXED DISK ZERO
429 0210 BA 0080   MOV    DX,0080H     ; RESET THE FIXED DISK
430 0213 2B C0     SUB    AX,AX         ;
431 0215 CD 13     INT    13H          ; FILE I/O CALL
432 0217 59      POP    CX           ; RESTORE LOOP COUNT
433 0218 72 08     JC     H6           ; IF ERROR, TRY AGAIN
434 021A E2 D5     LOOP  H6           ; DO IT FOR RETRY TIMES
435
436          ;----- UNABLE TO IPL FROM THE DISKETTE OR FIXED DISK
437
438 021C B0 46     H9:  MOV    AL,045H       ;
439 021E E6 80     OUT    MFG_PORT,AL ;
440
441 0220 CD 18     INT    18H          ; GO TO RESIDENT BASIC
442
443          ;----- HARD FILE RESET FAILURE
444
445 0222 E2 EB     H10A: LOOP  H8           ; TRY RESET AGAIN
446 0224 EB F6     JMP    H9           ; GO TO RESIDENT BASIC
447
448          ;----- IF DISKETTE READ OK BUT BOOT RECORD IS NOT STOP SYSTEM ALLOW SOFT RESET
449
450 0226 BE 0000 E H10:  MOV    SI,OFFSET E602 ; PRINT DISKETTE BOOT
451 0229 E8 0000 E CALL  E_MSG         ; PRINT MESSAGE
452 022C EB FE     JMP    HT1
453 022E          H11:  JMP    ENDP
454 022E          BOOT_STRAP
455 022E          POSTS
456 022E          CODE      ENDP
                     ENDS
                     END

```

SECTION 5

```

1 PAGE 118,121
2 TITLE DSKETTE -- 04/21/86 DSKETTE BIOS
3 .286C
4 .LIST
5 SUBTTL (DSK1.ASM)
6 .LIST
7
8 -----INT 13 H-----
9 DISKETTE I/O
10 THIS INTERFACE PROVIDES DISK ACCESS TO THE 5.25 INCH 360 KB,
11 1.2 MB, 720 KB, AND 1.44 MB DISKETTE DRIVES.
12
13 INPUT
14 (AH)=00H RESET DISKETTE SYSTEM
15 HARD RESET TO NEC, PREPARE COMMAND, RECALIBRATE REQUIRED
16 ON ALL DRIVES
17
18 -----(AH)=01H READ THE STATUS OF THE SYSTEM INTO (AH)
19 @DSKETTE_STATUS FROM LAST OPERATION IS USED
20 -----
21 REGISTERS FOR READ/WRITE/VERIFY/FORMAT
22 (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)
23 (DH) - HEAD NUMBER (0-1 ALLOWED, NOT VALUE CHECKED)
24 (CH) - TRACK NUMBER (NOT VALUE CHECKED)
25
26 MEDIA DRIVE TRACK NUMBER
27 320/360 320/360 0-39
28 320/360 1.2M 0-39
29 1.2M 1.2M 0-79
30 720K 720K 0-79
31 1.44M 1.44M 0-79
32
33 (CL) - SECTOR NUMBER (NOT VALUE CHECKED, NOT USED FOR FORMAT)
34 MEDIA DRIVE SECTOR NUMBER
35 320/360 320/360 1-8/9
36 320/360 1.2M 1-8/9
37 1.2M 1.2M 1-15
38 720K 720K 1-9
39 1.44M 1.44M 1-18
40
41 (AL) - NUMBER OF SECTORS (NOT VALUE CHECKED, NOT USED FOR FORMAT)
42 MEDIA DRIVE MAX NUMBER OF SECTORS
43 320/360 320/360 8/9
44 320/360 1.2M 8/9
45 1.2M 1.2M 15
46 720K 720K 9
47 1.44M 1.44M 18
48
49 (ES:BX) - ADDRESS OF BUFFER (NOT REQUIRED FOR VERIFY)
50
51 -----(AH)=02H READ THE DESIRED SECTORS INTO MEMORY
52 -----
53 (AH)=03H WRITE THE DESIRED SECTORS FROM MEMORY
54
55 -----(AH)=04H VERIFY THE DESIRED SECTORS
56 -----
57 (AH)=05H FORMAT THE DESIRED TRACK
58 (ES:BX) MUST POINT TO THE COLLECTION OF DESIRED ADDRESS FIELDS
59 FOR THE TRACK. EACH FIELD IS COMPOSED OF 4 BYTES. (C,H,R,N),
60 WHERE C = TRACK NUMBER, H=HEAD NUMBER, R = SECTOR NUMBER,
61 N= NUMBER OF BYTES PER SECTOR (00=128,01=256,02=512,03=1024).
62 THERE MUST BE ONE ENTRY FOR EVERY SECTOR ON THE TRACK.
63 THIS INFORMATION IS USED TO FIND THE REQUESTED SECTOR DURING
64 READ/WRITE ACCESS.
65
66 PRIOR TO FORMATTING A DISKETTE, IF THERE EXISTS MORE THAN
67 ONE SUPPORTED MEDIA FORMAT TYPE WITHIN THE DRIVE IN QUESTION,
68 THEN "SET DASH TYPE" (INT 13H, AH = 17H) OR "SET MEDIA TYPE"
69 (INT 13H, AH = 18H) MUST BE CALLED TO SET THE DISKETTE TYPE
70 THAT IS TO BE FORMATTED. IF "SET DASH TYPE" OR "SET MEDIA TYPE"
71 IS NOT CALLED, THE FORMAT ROUTINE WILL ASSUME THE MEDIA FORMAT
72 TO BE THE MAXIMUM CAPACITY OF THE DRIVE.
73
74 THESE PARAMETERS OF DISK BASE MUST BE CHANGED IN ORDER TO
75 FORMAT THE FOLLOWING MEDIAS:
76
77 MEDIA DRIVE PARM 1 PARM 2
78 320K 320K/360K/1.2M 50H 8
79 360K 320K/360K/1.2M 50H 9
80 1.2M 1.2M 54H 15
81 720K 720K/1.44M 50H 9
82 1.44M 1.44M 6CH 18
83
84 NOTES: - PARM 1 = GAP LENGTH FOR FORMAT
85 - PARM 2 = EOT (LAST SECTOR ON TRACK)
86 - DISK BASE IS POINTED TO BY DISK POINTER LOCATED
87 AT ABSOLUTE ADDRESS 0178.
88 - WHEN FORMAT OPERATIONS ARE COMPLETE, THE PARAMETERS
89 SHOULD BE RESTORED TO THEIR RESPECTIVE INITIAL VALUES
90
91 -----(AH)=08H READ DRIVE PARAMETERS
92 -----
93 REGISTERS
94 INPUT
95 (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)
96
97 OUTPUT
98 (ES:DI) POINTS TO DRIVE PARAMETERS TABLE
99 (CH) - LOW ORDER 8 OF 10 BITS MAXIMUM NUMBER OF TRACKS
100 (CL) - BITS 7 & 6 - HIGH ORDER TWO BITS OF MAXIMUM TRACKS
101 - BITS 5 THRU 0 - MAXIMUM SECTORS PER TRACK
102 (DH) - MAXIMUM HEAD NUMBER
103 (DL) - NUMBER OF DISKETTE DRIVES INSTALLED
104 (BH) - 0
105 (BL) - BITS 7 THRU 4 = 0
106 - BITS 3 THRU 0 = VALID DRIVE TYPE VALUE IN CMOS
107 (AX) - 0
108
109 UNDER THE FOLLOWING CIRCUMSTANCES:
110 (1) THE DRIVE NUMBER IS INVALID,
111 (2) THE DRIVE TYPE IS UNKNOWN AND CMOS IS NOT PRESENT,
112 (3) THE DRIVE TYPE IS UNKNOWN AND CMOS IS BAD,
113 (4) OR THE DRIVE TYPE IS UNKNOWN AND THE CMOS DRIVE TYPE IS INVALID
114 THEN ES:AX,BX,CX,DX,D1=0 ; DL=NUMBER OF DRIVES.
115 IF NO DRIVES ARE PRESENT THEN: ES,AX,BX,CX,DX,D1=0.
116 @DSKETTE_STATUS = 0 AND CY IS RESET.
117
118 -----(AH)=15H READ DASH TYPE
119 OUTPUT REGISTERS

```

```

115 | (AH) - ON RETURN IF CARRY FLAG NOT SET, OTHERWISE ERROR
116 |     00 - DRIVE NOT PRESENT
117 |     01 - DISKETTE, NO CHANGE LINE AVAILABLE
118 |     02 - DISKETTE, CHANGE LINE AVAILABLE
119 |     03 - RESERVED
120 | (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)
121 | -----
122 | (AH)=16H DISK CHANGE LINE STATUS
123 | OUTPUT REGISTERS
124 | (AH) - 00 - DISK CHANGE LINE NOT ACTIVE
125 |     06 - DISK CHANGE LINE ACTIVE & CARRY BIT ON
126 | (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)
127 | -----
128 | (AH)=17H SET DASH TYPE FOR FORMAT
129 | INPUT REGISTERS
130 | (AL) - 00 - NOT USED
131 |     01 - DISKETTE 320/360K IN 360K DRIVE
132 |     02 - DISKETTE 360K IN 1.2M DRIVE
133 |     03 - DISKETTE 1.2M IN 1.2M DRIVE
134 |     04 - DISKETTE 720K IN 720K DRIVE
135 | (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)
136 |     DO NOT USE WHEN DISKETTE ATTACH CARD USED)
137 | -----
138 | (AH)=18H SET MEDIA TYPE FOR FORMAT
139 | INPUT REGISTERS
140 | (CH) - LOW ORDER 8 OF 10 BITS MAXIMUM NUMBER OF TRACKS
141 | (CL) - BITS 7 & 6 - HIGH ORDER TWO BITS OF MAXIMUM TRACKS
142 |     - BITS 5 THRU 0 - MAXIMUM SECTORS PER TRACK
143 | (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)
144 | OUTPUT REGISTERS
145 | (ESI:DI) - POINTER TO DRIVE PARAMETERS TABLE FOR THIS MEDIA TYPE.
146 |     UNCHANGED IF (AH) IS NON-ZERO
147 | (AH) - 00H, CY = 0, TRACK AND SECTORS/TRACK COMBINATION IS SUPPORTED
148 |     01H, CY = 1, FUNCTION IS NOT AVAILABLE
149 |     02H, CY = 1, TRACK AND SECTORS/TRACK COMBINATION IS NOT SUPPORTED
150 |     OR DRIVE TYPE UNKNOWN
151 |     80H, CY = 1, TIME OUT (DISKETTE NOT PRESENT)
152 | -----
153 | DISK CHANGE STATUS IS ONLY CHECKED WHEN A MEDIA SPECIFIED IS OTHER
154 | THAN 360 KB DRIVE. IF THE DISK CHANGE LINE IS FOUND TO BE
155 | ACTIVE THE FOLLOWING ACTIONS TAKE PLACE:
156 |     ATTEMPT TO RESET DISK CHANGE LINE TO INACTIVE STATE.
157 |     IF ATTEMPT SUCCEEDS SET DASH TYPE FOR FORMAT AND RETURN DISK
158 |     CHANGE ERROR CODE
159 |     IF ATTEMPT FAILS RETURN TIMEOUT ERROR CODE AND SET DASH TYPE
160 |     TO A PREDETERMINED STATE INDICATING MEDIA TYPE UNKNOWN.
161 |     IF THE DISK CHANGE LINE IN INACTIVE PERFORM SET DASH TYPE FOR FORMAT.
162 | -----
163 | DATA VARIABLE -- #DISK POINTER
164 | DOUBLE WORD POINTER TO THE CURRENT SET OF DISKETTE PARAMETERS
165 | -----
166 | OUTPUT FOR ALL FUNCTIONS
167 | AH = STATUS OF OPERATION
168 | STATUS BITS ARE DEFINED IN THE EQUATES FOR #DISKETTE_STATUS
169 | VARIABLE IN THE DATA SEGMENT OF THIS MODULE
170 | CY = 0 SUCCESSFUL OPERATION (AH=0 ON RETURN, EXCEPT FOR READ DASH
171 |     TYPE AHS(16)).
172 | CY = 1 FAILED OPERATION (AH HAS ERROR REASON)
173 | FOR READ/WRITE/VERIFY
174 |     AI - COUNT OF SECTORS TRANSFERRED
175 |     DS,BX,DX,CX PRESERVED
176 | NOTE: IF AN ERROR IS REPORTED BY THE DISKETTE CODE, THE APPROPRIATE
177 | ACTION IS TO RESET THE DISKETTE, THEN RETRY THE OPERATION.
178 | ON READ ACCESSES, NO MOTOR START DELAY IS TAKEN, SO THAT
179 | THREE RETRIES ARE REQUIRED ON READS TO ENSURE THAT THE
180 | PROBLEM IS NOT DUE TO MOTOR START-UP.
181 | -----
182 | .LIST
183 | DISKETTE STATE MACHINE - ABSOLUTE ADDRESS 40:90 (DRIVE A) & 91 (DRIVE B)
184 | .LIST
185 |
186 |
187 |
188 |
189 |
190 |
191 |
192 |
193 |
194 |
195 |
196 |
197 |
198 |
199 |
200 |
201 |
202 |
203 |
204 |
205 |
206 |
207 |
208 |
209 |
210 |
211 |
212 |
213 |
214 |
215 |
216 |
217 |
218 |
219 |
220 |
221 |
222 |
223 |
224 |
225 |

```

SECTION 5

```

224 PAGE
225
226 MD_STRUC STRUC
227 MD_SPEC1 DB ? ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
228 MD_SPEC2 DB ? ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
229 MD_OFF_TIM DB ? ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
230 MD_BYT_SEC DB ? ; 512 BYTES/SECTOR
231 MD_SEC_TRK DB ? ; EOT ( LAST SECTOR ON TRACK)
232 MD_GAP DB ? ; GAP LENGTH
233 MD_DTL DB ? ; DTL
234 MD_GAPS DB ? ; GAP LENGTH FOR FORMAT
235 MD_FIL_BYT DB ? ; FILL BYTE FOR FORMAT
236 MD_HD_TIM DB ? ; HEAD SETTLE TIME (MILLISECONDS)
237 MD_STR_TIM DB ? ; MOTOR START TIME (1/8 SECONDS)
238 MD_MAX_TRK DB ? ; MAX. TRACK NUMBER
239 MD_RATE DB ? ; DATA TRANSFER RATE
240 MD_STRUC ENDS
241
242 = 007F BITTOFF EQU 7FH
243 = 0080 BITTON EQU 80H
244
245
246 PUBLIC DISK_INT_1
247 PUBLIC SEEN
248 PUBLIC DISKETTE_SETUP
249 PUBLIC DISKETTE_IO_1
250
251 EXTRN CMOS_READ:NEAR
252 EXTRN DDS:NEAR
253 EXTRN DISK_BASE:NEAR
254 EXTRN WAITF:NEAR
255
256 = 0000 CODE SEGMENT BYTE PUBLIC
257
258 ASSUME CS:CODE,DS:DATA,ES:DATA
259
260
261
262 -----
263 ; DRIVE TYPE TABLE
264 ; -----
265 DR_TYPE DB 01 ; DRIVE TYPE, MEDIA TABLE
266 DW OFFSET MD_TBL1
267 DB 02+BITTON
268 DW OFFSET MD_TBL2
269 DR_DEFAULT DB 02
270 DW OFFSET MD_TBL3
271 DB 03
272 DW OFFSET MD_TBL4
273 DB 04+BITTON
274 DW OFFSET MD_TBL5
275 DB 04
276 DW OFFSET MD_TBL6
277 DR_TYPE_E EQU (DR_TYPE_E-DR_TYPE)/3
278 DR_CNT EQU (DR_TYPE_E-DR_TYPE)/3
279
280 -----
281 ; MEDIA/DRIVE PARAMETER TABLES
282 ; -----
283
284 ;
285 ; 360 KB MEDIA IN 360 KB DRIVE
286 ; -----
287 MD_TBL1 LABEL BYTE
288 DB 1101111B ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
289 DB 2 ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
290 DB MOTOR_WAIT ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
291 DB 2 ; 512 BYTES/SECTOR
292 DB 09 ; EOT ( LAST SECTOR ON TRACK)
293 DB 02AH ; GAP LENGTH
294 DB 0FFH ; DTL
295 DB 050H ; GAP LENGTH FOR FORMAT
296 DB 0F6H ; FILL BYTE FOR FORMAT
297 DB 15 ; HEAD SETTLE TIME (MILLISECONDS)
298 DB 8 ; MOTOR START TIME (1/8 SECONDS)
299 DB 39 ; MAX. TRACK NUMBER
300 DB RATE_260 ; DATA TRANSFER RATE
301
302 -----
303 ; 360 KB MEDIA IN 1.2 MB DRIVE
304 ; -----
305 MD_TBL2 LABEL BYTE
306 DB 1101111B ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
307 DB 2 ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
308 DB MOTOR_WAIT ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
309 DB 2 ; 512 BYTES/SECTOR
310 DB 09 ; EOT ( LAST SECTOR ON TRACK)
311 DB 02AH ; GAP LENGTH
312 DB 0FFH ; DTL
313 DB 050H ; GAP LENGTH FOR FORMAT
314 DB 0F6H ; FILL BYTE FOR FORMAT
315 DB 15 ; HEAD SETTLE TIME (MILLISECONDS)
316 DB 8 ; MOTOR START TIME (1/8 SECONDS)
317 DB 39 ; MAX. TRACK NUMBER
318 DB RATE_300 ; DATA TRANSFER RATE
319
320 -----
321 ; 1.2 MB MEDIA IN 1.2 MB DRIVE
322 ; -----
323 MD_TBL3 LABEL BYTE
324 DB 1101111B ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
325 DB 2 ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
326 DB MOTOR_WAIT ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
327 DB 2 ; 512 BYTES/SECTOR
328 DB 15 ; EOT ( LAST SECTOR ON TRACK)
329 DB 018H ; GAP LENGTH
330 DB 0FFH ; DTL
331 DB 054H ; GAP LENGTH FOR FORMAT
332 DB 0F6H ; FILL BYTE FOR FORMAT
333 DB 15 ; HEAD SETTLE TIME (MILLISECONDS)
334 DB 8 ; MOTOR START TIME (1/8 SECONDS)
335 DB 79 ; MAX. TRACK NUMBER
336 DB RATE_500 ; DATA TRANSFER RATE
337 DB 038H 00
    
```

```

338 -----
339 |
340 | 720 KB MEDIA IN 720 KB DRIVE |
341 |-----|
342 |
343 MD_TBL4 LABEL BYTE
344 DB 11011111B ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
345 DB 2 ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
346 DB MOTOR_WAIT ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
347 DB 2 ; 512 BYTES/SECTOR
348 DB 09 ; EOT ( LAST SECTOR ON TRACK)
349 DB 02AH ; GAP LENGTH
350 DB 0FFH ; DTL
351 DB 050H ; GAP LENGTH FOR FORMAT
352 DB 0F6H ; FILL BYTE FOR FORMAT
353 DB 15 ; HEAD SETTLE TIME (MILLISECONDS)
354 DB 8 ; MOTOR START TIME (1/8 SECONDS)
355 DB 79 ; MAX. TRACK NUMBER
356 DB RATE_250 ; DATA TRANSFER RATE
357 |-----|
358 |
359 | 720 KB MEDIA IN 1.44 MB DRIVE |
360 |-----|
361 |
362 MD_TBL5 LABEL BYTE
363 DB 11011111B ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
364 DB 2 ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
365 DB MOTOR_WAIT ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
366 DB 2 ; 512 BYTES/SECTOR
367 DB 09 ; EOT ( LAST SECTOR ON TRACK)
368 DB 02AH ; GAP LENGTH
369 DB 0FFH ; DTL
370 DB 050H ; GAP LENGTH FOR FORMAT
371 DB 0F6H ; FILL BYTE FOR FORMAT
372 DB 15 ; HEAD SETTLE TIME (MILLISECONDS)
373 DB 8 ; MOTOR START TIME (1/8 SECONDS)
374 DB 79 ; MAX. TRACK NUMBER
375 DB RATE_250 ; DATA TRANSFER RATE
376 |-----|
377 |
378 | 1.44 MB MEDIA IN 1.44 MB DRIVE |
379 |-----|
380 |
381 MD_TBL6 LABEL BYTE
382 DB 10101111B ; SRT=A, HD UNLOAD=0F - 1ST SPECIFY BYTE
383 DB 2 ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
384 DB MOTOR_WAIT ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
385 DB 2 ; 512 BYTES/SECTOR
386 DB 18 ; EOT ( LAST SECTOR ON TRACK)
387 DB 01BH ; GAP LENGTH
388 DB 0FFH ; DTL
389 DB 06CH ; GAP LENGTH FOR FORMAT
390 DB 0F6H ; FILL BYTE FOR FORMAT
391 DB 15 ; HEAD SETTLE TIME (MILLISECONDS)
392 DB 8 ; MOTOR START TIME (1/8 SECONDS)
393 DB 79 ; MAX. TRACK NUMBER
394 DB RATE_500 ; DATA TRANSFER RATE
395 |-----|
396 |
397 DISKETTE_IQ_1 PROC FAR ;>>> ENTRY POINT FOR ORG 0EC69H
398 |
399 STI ; INTERRUPTS BACK ON
400 PUSH BP ; USER REGISTER
401 PUSH DI ; USER REGISTER
402 PUSH DX ; HEAD #, DRIVE # OR USER REGISTER
403 PUSH BX ; BUFFER OFFSET PARAMETER OR REGISTER
404 PUSH CX ; TRACK #-SECTOR # OR USER REGISTER
405 MOV BP,SP ; BP => PARAMETER LIST DEP. ON AH
406 | [BP] = SECTOR #
407 | [BP+1] = TRACK #
408 | [BP+2] = BUFFER OFFSET
409 | FOR RETURN OF DRIVE PARAMETERS:
410 | CL/[BP] = BITS 7&6 HI BITS OF MAX CYL
411 | CH/[BP+1] = BITS 0-5 MAX SECTORS/TRACK
412 | BL/[BP+2] = LOW 8 BITS OF MAX CYL.
413 | | [BP+3] = BITS 7-4 = 0
414 | | [BP+4] = BITS 3-0 = VALID CMOS TYPE
415 | | [BP+5] = 0
416 | | [BP+6] = # DRIVES INSTALLED
417 | | [BP+7] = MAX HEAD #
418 | | [BP+8] = OFFSET TO DISK BASE
419 | | [BP+9] = BUFFER SEGMENT PARM OR USER REGISTER
420 | | [BP+10] = USER REGISTERS
421 | | [BP+11] = SEGMENT OF BIOS DATA AREA TO DS
422 | | [BP+12] = CHECK FOR > LARGEST FUNCTION
423 | | [BP+13] = FUNCTION OK
424 | | [BP+14] = REPLACE WITH KNOWN INVALID FUNCTION
425 |-----|
426 OK_FUNC:
427 CMP AH,1 ; RESET OR STATUS ?
428 JBE OK_DRV ; IF RESET OR STATUS DRIVE ALWAYS OK
429 CMP AH,8 ; READ DRIVE PARMS ?
430 JZ OK_DRV ; IF SO DRIVE CHECKED LATER
431 CMP DL,1 ; DRIVES 0 AND 1 OK
432 JBE OK_DRV ; IF 0 OR 1 THEN JUMP
433 MOV AH,14H ; REPLACE WITH KNOWN INVALID FUNCTION
434 |-----|
435 OK_DRV:
436 MOV CL,AH ; CL = FUNCTION
437 XOR CH,CH ; CX = FUNCTION
438 SHL CL,1 ; FUNCTION TIMES 2
439 ADD BX,CX ; LOAD START OF FUNCTION TABLE
440 MOV AX,DX ; ADD OFFSET INTO TABLE => ROUTINE
441 MOV SI,AX ; ADD HEAD #, # OF SECTORS OR DASD TYPE
442 XOR DH,DH ; DX = DRIVE #
443 MOV SI,AX ; SI = HEAD #, # OF SECTORS OR DASD TYPE
444 MOV DI,DX ; DI = DRIVE #
445 MOV AH,@DSKETTE_STATUS ; LOAD STATUS TO AH FOR STATUS FUNCTION
446 MOV @DSKETTE_STATUS,0 ; INITIALIZE FOR ALL OTHERS
447 |-----|
448 |
449 | THROUGHOUT THE DISKETTE BIOS, THE FOLLOWING INFORMATION IS CONTAINED IN
450 | THE FOLLOWING MEMORY LOCATIONS AND REGISTERS. NOT ALL DISKETTE BIOS
451 | FUNCTIONS REQUIRE ALL OF THESE PARAMETERS.

```

SECTION 5

```

452          ;          D1      : DRIVE #
453          ;          SI-HI   : HEAD #
454          ;          SI-LOW  : # OF SECTORS OR DASD TYPE FOR FORMAT
455          ;          ES      : BUFFER SEGMENT
456          ;          [BP]    : SECTOR #
457          ;          [BP+1] : TRACK #
458          ;          [BP+2] : BUFFER OFFSET
459          ;
460          ;
461          ; ACROSS CALLS TO SUBROUTINES THE CARRY FLAG (CY=1), WHERE INDICATED IN
462          ; SUBROUTINE PROLOGUES, REPRESENTS AN EXCEPTION RETURN (NORMALLY AN ERROR
463          ; CONDITION). IN MOST CASES, WHEN CY = 1, #DSKETTE_STATUS CONTAINS THE
464          ; SPECIFIC ERROR CODE.
465          ;
466          ;          (AH) = #DSKETTE_STATUS
467          ;          CALL WORD PTR CS:[BX] ; CALL THE REQUESTED FUNCTION
468          ;
469          ;          POP SI
470          ;          POP DS
471          ;          POP CX
472          ;          POP BX
473          ;          POP DX
474          ;          POP DI
475          ;          MOV BP,SP
476          ;          PUSH AX
477          ;          MOV [BP+6],AX
478          ;          POP AX
479          ;          POP BP
480          ;          IRET
481          ;
482          ;-----
483          ; FNC_TAB DW DISK_RESET ; AH = 00H; RESET
484          ; DW DISK_STATUS ; AH = 01H; STATUS
485          ; DW DISK_READ ; AH = 02H; READ
486          ; DW DISK_WRITE ; AH = 03H; WRITE
487          ; DW DISK_VERIFY ; AH = 04H; VERIFY
488          ; DW DISK_FORMAT ; AH = 05H; FORMAT
489          ; DW FNC_ERR ; AH = 06H; INVALID
490          ; DW FNC_ERR ; AH = 07H; INVALID
491          ; DW DISK_PARAMS ; AH = 08H; READ DRIVE PARAMETERS
492          ; DW FNC_ERR ; AH = 09H; INVALID
493          ; DW FNC_ERR ; AH = 0AH; INVALID
494          ; DW FNC_ERR ; AH = 0BH; INVALID
495          ; DW FNC_ERR ; AH = 0CH; INVALID
496          ; DW FNC_ERR ; AH = 0DH; INVALID
497          ; DW FNC_ERR ; AH = 0EH; INVALID
498          ; DW FNC_ERR ; AH = 0FH; INVALID
499          ; DW FNC_ERR ; AH = 10H; INVALID
500          ; DW FNC_ERR ; AH = 11H; INVALID
501          ; DW FNC_ERR ; AH = 12H; INVALID
502          ; DW FNC_ERR ; AH = 13H; INVALID
503          ; DW FNC_ERR ; AH = 14H; INVALID
504          ; DW DISK_TYPE ; AH = 15H; READ DASD TYPE
505          ; DW DISK_CHANGE ; AH = 16H; CHANGE STATUS
506          ; DW FORMAT_SET ; AH = 17H; SET DASD TYPE
507          ; DW SET_MEDIA ; AH = 18H; SET MEDIA TYPE
508          ; $ ; END
509          ;
510          ; DISKETTE_IO_1 ENDP
511          ;
512          ; DISK_RESET (AH = 00H)
513          ; RESET THE DISKETTE SYSTEM.
514          ;
515          ; ON EXIT: #DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION ;
516          ;-----
517          ; DISK_RESET PROC NEAR
518          ; MOV DX,03F2H ; ADAPTER CONTROL PORT
519          ; CL ; NO INTERRUPTS
520          ; MOV AL,#MOTOR_STATUS ; GET DIGITAL OUTPUT REGISTER REFLECTION
521          ; AND AL,00111111B ; KEEP SELECTED AND MOTOR ON BITS
522          ; ROL AL,4 ; MOTOR VALUE TO HIGH NIBBLE
523          ; OR AL,00001000B ; DRIVE SELECT TO LOW NIBBLE
524          ; OUT DX,AL ; TURN ON INTERRUPT ENABLE
525          ; MOV #SEEK_STATUS,0 ; SET RECALIBRATE REQUIRED ON ALL DRIVES
526          ; JMP $+2 ; WAIT FOR I/O
527          ; JMP $+2 ; WAIT FOR I/O (TO INSURE MINIMUM
528          ; ; PULSE WIDTH)
529          ; OR AL,00000100B ; TURN OFF RESET BIT
530          ; OUT DX,AL ; RESET THE ADAPTER
531          ; STI ; ENABLE THE INTERRUPTS
532          ; CALL WAIT_INT ; WAIT FOR THE INTERRUPT
533          ; JC DR_ERR ; IF ERROR, RETURN IT
534          ; MOV CX,11000000B ; CL = EXPECTED #NEC_STATUS
535          ;
536          ; NEXT_DRY:
537          ; PUSH CX ; SAVE FOR CALL
538          ; MOV AX,OFFSET DR_POP_ERR ; LOAD NEC_OUTPUT ERROR ADDRESS
539          ; PUSH AX ;
540          ; MOV AX,08H ; SENSE INTERRUPT STATUS COMMAND
541          ; CALL NEC_OUTPUT ;
542          ; POP AX ; THROW AWAY ERROR RETURN
543          ; CALL RESULTS ; READ IN THE RESULTS
544          ; POP CX ; RESTORE AFTER CALL
545          ; JC DR_ERR ; ERROR RETURN
546          ; CMP CL,#NEC_STATUS ; TEST FOR DRIVE READY TRANSITION
547          ; JNZ DR_ERR ; EVERYTHING OK
548          ; INC CL ; NEXT EXPECTED #NEC_STATUS
549          ; CMP CL,11000011B ; ALL POSSIBLE DRIVES CLEARED
550          ; JBE NEXT_DRY ; FALL THRU IF 11000100B OR >
551          ;
552          ; CALL SEND_SPEC ; SEND SPECIFY COMMAND TO NEC
553          ;
554          ; RESBAC:
555          ; CALL SETUP_END ; VARIOUS CLEANUPS
556          ; MOV DX,SI ; GET SAVED AL TO BL
557          ; MOV AL,BL ; PUT BACK FOR RETURN
558          ; RET ;
559          ;
560          ; DR_POP_ERR:
561          ; POP CX ; CLEAR STACK
562          ;
563          ; DR_ERR:
564          ; OR #DSKETTE_STATUS,BAD_NEC ; SET ERROR CODE
565          ; JMP SHORT RESBAC ; RETURN FROM RESET
566          ; DISK_RESET ENDP
    
```

```

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
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
;-----
; DISK_STATUS (AH = 01H)
; DISKETTE STATUS
; ON ENTRY: AH = STATUS OF PREVIOUS OPERATION
; ON EXIT:  #DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
;-----
DISK_STATUS PROC NEAR
MOV #DSKETTE_STATUS,AH ; PUT BACK FOR SETUP_END
CALL SETUP_END ; VARIOUS CLEANUPS
MOV BX,SI ; GET SAVED AL TO BL
MOV AL,BL ; PUT BACK FOR RETURN
RET
DISK_STATUS ENDP
;-----
; DISK_READ (AH = 02H)
; DISKETTE READ.
; ON ENTRY: DI = DRIVE #
; SI-HI = HEAD #
; SI-LOW = # OF SECTORS
; ES = BUFFER SEGMENT
; [BP] = SECTOR #
; [BP+1] = TRACK #
; [BP+2] = BUFFER OFFSET
; ON EXIT:  #DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
;-----
DISK_READ PROC NEAR
AND #MOTOR_STATUS,01111111B ; INDICATE A READ OPERATION
MOV AX,DE44H ; AX = NEC COMMAND, DMA COMMAND
CALL RD_WR_VF ; COMMON READ/WRITE/VERIFY
RET
DISK_READ ENDP
;-----
; DISK_WRITE (AH = 03H)
; DISKETTE WRITE.
; ON ENTRY: DI = DRIVE #
; SI-HI = HEAD #
; SI-LOW = # OF SECTORS
; ES = BUFFER SEGMENT
; [BP] = SECTOR #
; [BP+1] = TRACK #
; [BP+2] = BUFFER OFFSET
; ON EXIT:  #DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
;-----
DISK_WRITE PROC NEAR
MOV AX,C054H ; AX = NEC COMMAND, DMA COMMAND
AND #MOTOR_STATUS,10000000B ; INDICATE WRITE OPERATION
CALL RD_WR_VF ; COMMON READ/WRITE/VERIFY
RET
DISK_WRITE ENDP
;-----
; DISK_VERF (AH = 04H)
; DISKETTE VERIFY.
; ON ENTRY: DI = DRIVE #
; SI-HI = HEAD #
; SI-LOW = # OF SECTORS
; ES = BUFFER SEGMENT
; [BP] = SECTOR #
; [BP+1] = TRACK #
; [BP+2] = BUFFER OFFSET
; ON EXIT:  #DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
;-----
DISK_VERF PROC NEAR
AND #MOTOR_STATUS,01111111B ; INDICATE A READ OPERATION
MOV AX,DE42H ; AX = NEC COMMAND, DMA COMMAND
CALL RD_WR_VF ; COMMON READ/WRITE/VERIFY
RET
DISK_VERF ENDP
;-----
; DISK_FORMAT (AH = 05H)
; DISKETTE FORMAT.
; ON ENTRY: DI = DRIVE #
; SI-HI = HEAD #
; SI-LOW = # OF SECTORS
; ES = BUFFER SEGMENT
; [BP] = SECTOR #
; [BP+1] = TRACK #
; [BP+2] = BUFFER OFFSET
; #DISK_POINTER POINTS TO THE PARAMETER TABLE OF
; THIS DRIVE
; ON EXIT:  #DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
;-----
DISK_FORMAT PROC NEAR
CALL XLAT_NEW ; TRANSLATE STATE TO PRESENT ARCH.
CALL FMT_INIT ; ESTABLISH STATE IF UNESTABLISHED
OR #MOTOR_STATUS,10000000B ; INDICATE WRITE OPERATION
CALL MED_CHANGE ; CHECK MEDIA CHANGE AND RESET IF SO
CALL FM_DON ; MEDIA CHANGED, SKIP
CALL SEND_SPEC ; SEND SPECIFY COMMAND TO NEC
CALL CHK_LAstrate ; ZF=1 ATTEMPT RATE IS SAME AS LAST RATE
CALL FM_WR ; YES, SKIP SPECIFY COMMAND
CALL SEND_RATE ; SEND DATA RATE TO CONTROLLER
FM_WR: CALL FMTDMA_SET ; SET UP THE DMA FOR FORMAT
CALL FM_DON ; RETURN WITH ERROR
MOV AH,04DH ; ESTABLISH THE FORMAT COMMAND
CALL NEC_INIT ; INITIALIZE THE NEC
CALL NEC_OUTPUT ; ERROR - EXIT
MOV AX,OFFSET FM_DON ; LOAD ERROR ADDRESS
PUSH AX ; PUSH NEC OUT ERROR RETURN
MOV DL,3 ; BYTES/SECTOR VALUE TO NEC
CALL GET_PARM
CALL NEC_OUTPUT
MOV DL,7 ; SECTORS/TRACK VALUE TO NEC
CALL GET_PARM
CALL NEC_OUTPUT
MOV DL,7 ; GAP LENGTH VALUE TO NEC
CALL GET_PARM
CALL NEC_OUTPUT

```

SECTION 5

```

(DSK2.ASM)
680 01AF B2 08      MOV     DL,8           ; FILLER BYTE TO NEC
681 01B1 E8 0905 R   CALL    GET_PARM
682 01B4 E8 09F8 R   CALL    NEC_OUTPUT
683 01B7 58         POP     AX             ; THROW AWAY ERROR
684 01BB E8 075B R   CALL    NEC_TERM      ; TERMINATE, RECEIVE STATUS, ETC.
685 01B8             FM_DONE:
686 01BB E8 0435 R   CALL    XLAT_OLD      ; TRANSLATE STATE TO COMPATIBLE MODE
687 01BE E8 0854 R   CALL    SETUP_END     ; VARIOUS CLEANUPS
688 01C1 8B DE      MOV     BX,SI          ; GET SAVED AL TO BL
689 01C3 8A C3      MOV     AL,BL         ; PUT BACK FOR RETURN
690 01C5 C3         RET
691 01C6             DISK_FORMAT ENDP
692
693 -----
694 FNC_ERR:          INVALID FUNCTION REQUESTED OR INVALID DRIVE;
695                 SET BAD COMMAND IN STATUS.
696
697
698 ON EXIT:         @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
699
700 FNC_ERR PROC     NEAR
701                 MOV     AX,SI           ; INVALID FUNCTION REQUEST
702                 MOV     AH,BAD_CMD     ; RESTORE AL
703                 MOV     @DSKETTE_STATUS,AH ; SET BAD COMMAND ERROR
704                 STC                     ; STORE IN DATA AREA
705                 RET                     ; SET CARRY INDICATING ERROR
706
707 FNC_ERR ENDP
708 -----
709 DISK_PARAMS (AH = 08H)
710             READ DRIVE PARAMETERS.
711 ON ENTRY:    DI = DRIVE #
712 ON EXIT:
713             CL/[BP] = BITS 7 & 6 HIGH 2 BITS OF MAX CYLINDER
714             CH/[BP+1] = LOW 6 BITS OF MAX CYLINDER
715             BL/[BP+2] = BITS 7-4 = 0
716             BH/[BP+3] = BITS 3-0 = VALID CMOS DRIVE TYPE
717             DL/[BP+4] = # DRIVES INSTALLED
718             DH/[BP+5] = MAX HEAD #
719             DI/[BP+6] = OFFSET OF MEDIA/DRIVE PARAMETER TABLE
720             ES = SEGMENT OF MEDIA/DRIVE PARAMETER TABLE
721             AX = 0
722
723 NOTE: THE ABOVE INFORMATION IS STORED IN THE USERS STACK AT
724 THE LOCATIONS WHERE THE MAIN ROUTINE WILL POP THEM
725 INTO THE APPROPRIATE REGISTERS BEFORE RETURNING TO THE
726 CALLER.
727 -----
728 DISK_PARAMS PROC NEAR
729             CALL    XLAT_NEW           ; TRANSLATE STATE TO PRESENT ARCH.
730             MOV     WORD PTR [BP+2],0 ; DRIVE TYPE = 0
731             MOV     AX,@EQUIP_FLAG    ; LOAD EQUIPMENT FLAG FOR # DISKETTES
732             AND     AL,11000001B     ; KEEP DISKETTE DRIVE BITS
733             MOV     DL,2              ; DISKETTE DRIVES = 2
734             CMP     AL,01000001B     ; 2 DRIVES INSTALLED ?
735             JZ      STO_DL           ; IF YES JUMP
736             DEC     DL                ; DISKETTE DRIVES = 1
737             CMP     AL,00000001B     ; 1 DRIVE INSTALLED ?
738             JNZ     NON_DRY          ; IF NO JUMP
739
740 STO_DL:      MOV     [BP+4],DL        ; STORE NUMBER OF DRIVES
741             CMP     DI,1             ; CHECK FOR VALID DRIVE
742             JA      NON_DRY         ; DRIVE INVALID
743             MOV     BYTE PTR [BP+5],1 ; MAXIMUM HEAD NUMBER = 1
744             CALL    CMOS_TYPE       ; RETURN DRIVE TYPE IN AL
745             JC      CHK_EST         ; ON CMOS BAD CHECK ESTABLISHED
746             OR     AL,AL            ; TEST FOR NO DRIVE TYPE
747             JZ      JUMP_IF_SO     ; JUMP IF SO
748             CALL    DR_TYPE_CHECK   ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
749             JC      CHK_EST         ; TYPE NOT IN TABLE (POSSIBLE BAD CMOS)
750             MOV     [BP+2],AL        ; STORE VALID CMOS DRIVE TYPE
751             MOV     CL,CS:[BX].MD_SEC_TRK ; GET SECTOR/TRACK
752             MOV     CH,CS:[BX].MD_MAX_TRK ; GET MAX. TRACK NUMBER
753             JMP     SHORT STO_CX    ; CMOS GOOD, USE CMOS
754
755 CHK_EST:     MOV     AH,@DSK_STATE[DI] ; LOAD STATE FOR THIS DRIVE
756             TEST    AH,MED_DET      ; CHECK FOR ESTABLISHED STATE
757             JZ      NON_DRY         ; CMOS BAD/INVALID AND UNESTABLISHED
758
759 USE_EST:     AND     AH,RATE_MSK     ; ISOLATE STATE
760             CMP     AH,RATE_250     ; RATE 250 ?
761             JNE     USE_EST2        ; NO, GO CHECK OTHER RATE
762
763 I--- DATA RATE IS 250 KBS, TRY 360 KB TABLE FIRST
764
765 PARM144:     MOV     AL,04           ; DRIVE TYPE 4 (1.44MB)
766             CALL    DR_TYPE_CHECK   ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
767             MOV     CL,CS:[BX].MD_SEC_TRK ; GET SECTOR/TRACK
768             MOV     CH,CS:[BX].MD_MAX_TRK ; GET MAX. TRACK NUMBER
769
770 STO_CX:     MOV     [BP],CX          ; SAVE IN STACK FOR RETURN
771             MOV     [BP+4],BX        ; ADDRESS OF MEDIA/DRIVE PARAM TABLE
772             MOV     AX,CS           ; SEGMENT MEDIA/DRIVE PARAMETER TABLE
773             ES,AX                   ; ES IS SEGMENT OF TABLE
774
775 OP_OUT:     CALL    XLAT_OLD        ; TRANSLATE STATE TO COMPATIBLE MODE
776             XOR     AX,AX           ; CLEAR
777             CLC
778             RET
    
```

```

794
795
796
797 0253
798 0253 C6 46 04 00
799
800 0257
801 0257 81 FF 0080
802 025B 72 09
803
804
805
806 025D E8 0435 R
807 0260 8B C6
808 0262 B4 01
809 0264 F9
810 0265 C3
811
812 0266
813 0266 33 C0
814 0268 89 46 00
815 026B 88 66 05
816 026E 89 46 06
817 0271 8E C0
818 0273 EB D7
819
820
821
822 0275
823 0275 80 02
824 0277 E8 038B R
825 027A 2E1 8A 4F 04
826 027E 2E1 8A 4F 0B
827 0282 80 FC 40
828 0285 74 B8
829 0287 EB AC
830
831 0289
832
833
834
835
836
837
838
839
840 0289
841 0289 E8 040F R
842 028C 8A 85 0090 R
843 0290 8A C0
844 0292 74 13
845 0294 B4 01
846 0296 A8 01
847 0298 74 02
848 029A B4 02
849
850 029C
851 029C 50
852 029D E8 0435 R
853 02A0 88 C0
854 02A1 F8
855 02A2 8B DE
856 02A4 8A C3
857 02A5 C3
858 02A7
859 02A7 32 E4
860 02A9 EB F1
861 02AB
862
863
864
865
866
867
868
869
870
871
872 02AB
873 02AB E8 040F R
874 02AE 8A 85 0090 R
875 02B2 0A C0
876 02B4 74 19
877 02B6 A8 01
878 02B8 74 05
879
880 02BA E8 0B28 R
881 02BD 74 05
882
883 02BF C6 06 0041 R 06
884
885 02C4 E8 0435 R
886 02C7 E8 0884 R
887 02CA 8B DE
888 02CC 8A C3
889 02CE C3
890
891 02CF
892 02CF 80 0E 0041 R 80
893 02D4 EB EE
894 02D5
895
896
897
898
899
900
901
902
903
904
905
906
907

;----- NO DRIVE PRESENT HANDLER
NON_DRIVE:
MOV     BYTE PTR [BP+4],0      ; CLEAR NUMBER OF DRIVES
NON_DRIVE:
CMP     DI,80H                ; CHECK FOR FIXED MEDIA TYPE REQUEST
JB      NON_DRIVE2            ; CONTINUE IF NOT REQUEST FALL THROUGH
;----- FIXED DISK REQUEST FALL THROUGH ERROR
CALL    XLAT_OLD              ; ELSE TRANSLATE TO COMPATIBLE MODE
MOV     AX,ST                  ; RESTORE AL
MOV     AH,BAD_CMD            ; SET BAD COMMAND ERROR
STC     ; SET ERROR RETURN CODE
RET
NON_DRIVE2:
XOR     AX,AX                  ; CLEAR PARMS IF NO DRIVES OR CMOS BAD
MOV     [BP],AX               ; TRACKS, SECTORS/TRACK = 0
MOV     [BP+5],AH              ; HEAD = 0
MOV     [BP+6],AX              ; OFFSET TO DISK BASE = 0
MOV     ES,AX                  ; ES IS SEGMENT OF TABLE
JMP     SHORT DP_OUT
;--- DATA RATE IS EITHER 300 KBS OR 500 KBS, TRY 1.2 MB TABLE FIRST
USE_EST2:
MOV     AL,02                  ; DRIVE TYPE 2 (1.2MB)
CALL    DR_TYPE_CHECK         ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
MOV     CL,CS:[BX].MD_SEC_TRK ; GET SECTOR/TRACK
MOV     CH,CS:[BX].MD_MAX_TRK ; GET MAX. TRACK NUMBER
CMP     AH,RATE_300           ; RATE 300 ?
JE      STD_CX                 ; MUST BE 1.2MB DRIVE
JMP     SHORT PARM144         ; ELSE, IT IS 1.44MB DRIVE
DISK_PARMS     ENDP
;-----
; DISK_TYPE (AH = 18H)
; THIS ROUTINE RETURNS THE TYPE OF MEDIA INSTALLED.
; ON ENTRY: DI = DRIVE #
; ON EXIT: AH = DRIVE TYPE, CY=0
;-----
DISK_TYPE     PROC NEAR
CALL    XLAT_NEW              ; TRANSLATE STATE TO PRESENT ARCH.
MOV     AL,#DISK_STATE[DI]    ; GET PRESENT STATE INFORMATION
OR      AL,AL                  ; CHECK FOR NO DRIVE
NO_DRIVE:
NO_DRIVE:
JZ      AH,NOCHGLN            ; NO CHANGE LINE FOR 40 TRACK DRIVE
TEST   AL,TRK_CAPA           ; IS THIS DRIVE AN 80 TRACK DRIVE?
JZ      DT_BACK               ; IF NO JUMP
MOV     AH,CHGLN              ; CHANGE LINE FOR 80 TRACK DRIVE
DT_BACK:
PUSH    AX                     ; SAVE RETURN VALUE
CALL    XLAT_OLD              ; TRANSLATE STATE TO COMPATIBLE MODE
POP     AX                     ; RESTORE RETURN VALUE
CLC     ; NO ERROR
MOV     BX,SI                  ; GET SAVED AL TO BL
MOV     AL,BL                  ; PUT BACK FOR RETURN
RET
NON_DRIVE:
XOR     AH,AH                  ; NO DRIVE PRESENT OR UNKNOWN
JMP     SHORT DT_BACK
DISK_TYPE     ENDP
;-----
; DISK_CHANGE (AH = 18H)
; THIS ROUTINE RETURNS THE STATE OF THE DISK CHANGE LINE.
; ON ENTRY: DI = DRIVE #
; ON EXIT: AH = #DISKETTE STATUS
;         00 - DISK CHANGE LINE INACTIVE, CY = 0
;         05 - DISK CHANGE LINE ACTIVE, CY = 1
;-----
DISK_CHANGE   PROC NEAR
CALL    XLAT_NEW              ; TRANSLATE STATE TO PRESENT ARCH.
MOV     AL,#DISK_STATE[DI]    ; GET MEDIA STATE INFORMATION
OR      AL,AL                  ; DRIVE PRESENT ?
JZ      DC_NON                ; JUMP IF NO DRIVE
TEST   AL,TRK_CAPA           ; 80 TRACK DRIVE ?
JZ      SETIT                  ; IF 50, CHECK CHANGE LINE
DC01:
CALL    READ_DSKCHNG          ; GO CHECK STATE OF DISK CHANGE LINE
JZ      FINIS                  ; CHANGE LINE NOT ACTIVE
SETIT:
MOV     #DISKETTE_STATUS,MEDIA_CHANGE ; INDICATE MEDIA REMOVED
FINIS:
CALL    XLAT_OLD              ; TRANSLATE STATE TO COMPATIBLE MODE
JMP     SETUP_END             ; VARIOUS CLEANUPS
MOV     BX,SI                  ; GET SAVED AL TO BL
MOV     AL,BL                  ; PUT BACK FOR RETURN
RET
DC_NON:
OR      #DISKETTE_STATUS,TIME_OUT ; SET TIMEOUT, NO DRIVE
JMP     SHORT FINIS
DISK_CHANGE   ENDP
;-----
; FORMAT SET (AH = 17H)
; THIS ROUTINE IS USED TO ESTABLISH THE TYPE OF
; MEDIA TO BE USED FOR THE FOLLOWING FORMAT OPERATION.
; ON ENTRY: SI LOW = DASD TYPE FOR FORMAT
;         DI = DRIVE #
; ON EXIT: #DISKETTE STATUS REFLECTS STATUS
;         AH = #DISKETTE_STATUS
;         CY = 1 IF ERROR
;-----

```

SECTION 5

```

908 02D6          FORMAT_SET      PROC NEAR
909 02D6 E8 040F R      CALL XLAT_NEW          ; TRANSLATE STATE TO PRESENT ARCH.
910 02D9 56           PUSH SI              ; SAVE DASD TYPE
911 02DA 88 C4        MOV AX,SI            ; AH = 7 , AL = DASD TYPE
912 02DC 32 E4        XOR AH,AH           ; AH = 0 , AL = DASD TYPE
913 02DE 88 F0        MOV SI,AX           ; SI = DASD TYPE
914 02E0 80 A5 0090 R DF AND #0,SI           ; CLEAR STATE
915 02E5 4E           DEC SI              ; CHECK FOR 320/360K MEDIA & DRIVE
916 02E6 75 07        JNZ NOT_320        ; BYPASS IF NOT
917 02E8 80 8D 0090 R 90 OR #0,SI           ; SET TO 320/360
918 02ED EB 37        JMP SHORT S0
919
920 02EF          NOT_320:
921 02EF E8 05E9 R      CALL MED_CHANGE      ; CHECK FOR TIME_OUT
922 02F2 80 3E 0041 R 80 CMP #0,TIME_OUT    ; BYPASS IF NOT
923 02F7 74 2D        JZ S0              ; IF TIME OUT TELL CALLER
924
925 02F9 4E           DEC SI              ; CHECK FOR 320/360K IN 1.2M DRIVE
926 02FA 75 07        JNZ NOT_320_12     ; BYPASS IF NOT
927 02FC 80 8D 0090 R 70 OR #0,SI           ; SET STATE
928 0301 EB 23        JMP SHORT S0
929
930 0303          NOT_320_12:
931 0303 4E           DEC SI              ; CHECK FOR 1.2M MEDIA IN 1.2M DRIVE
932 0304 75 07        JNZ NOT_12         ; BYPASS IF NOT
933 0306 80 8D 0090 R 10 OR #0,SI           ; SET STATE VARIABLE
934 030B EB 19        JMP SHORT S0       ; RETURN TO CALLER
935
936 030D          NOT_12:
937 030D 4E           DEC SI              ; CHECK FOR SET DASD TYPE 04
938 030E 75 20        JNZ FS_ERR        ; BAD COMMAND EXIT IF NOT VALID TYPE
939
940 0310 F6 85 0090 R 04 TEST #0,SI         ; DRIVE DETERMINED ?
941 0315 74 09        JZ ASSUME         ; IF STILL NOT DETERMINED ASSUME
942 0317 B0 50        MOV AL,MED_DET+RATE_300
943 0319 F6 85 0090 R 02 TEST #0,SI         ; MULTIPLE FORMAT CAPABILITY ?
944 031E 75 02        JNZ OR_IT_IN     ; IF 1.2 M THEN DATA RATE 300
945
946 0320          ASSUME:
947 0320 B0 90        MOV AL,MED_DET+RATE_250 ; SET UP
948
949 0322          OR_IT_IN:
950 0322 08 85 0090 R OR #0,SI         ; OR IN THE CORRECT STATE
951
952 0326 EB 0435 R     S0: CALL XLAT_OLD      ; TRANSLATE STATE TO COMPATIBLE MODE
953 0329 EB 0854 R     CALL SETUP_END    ; VARIOUS CLEANUPS
954 032C 5B           POP BX            ; GET SAVED AL TO BL
955 032D 8A C3        MOV AL,BL        ; PUT BACK FOR RETURN
956 032F C3           RET
957
958 0330          FS_ERR:
959 0330 C6 06 0041 R 01 MOV #0,BAD_CMD    ; UNKNOWN STATE,BAD COMMAND
960 0335 EB EF        JMP SHORT S0
961
962 0337          FORMAT_SET      ENDP
963
964 -----
965 ; SET_MEDIA (AH = 18H)
966 ; THIS ROUTINE SETS THE TYPE OF MEDIA AND DATA RATE
967 ; TO BE USED FOR THE FOLLOWING FORMAT OPERATION.
968 ;
969 ; ON ENTRY:
970 ; [BP] = SECTOR PER TRACK
971 ; [BP+1] = TRACK #
972 ; DI = DRIVE #
973 ;
974 ; ON EXIT:
975 ; #DSKETTE_STATUS REFLECTS STATUS
976 ; IF NO ERROR:
977 ; AH = 0
978 ; CY = 0
979 ; ES = SEGMENT OF MEDIA/DRIVE PARAMETER TABLE
980 ; DI/[BP+6] = OFFSET OF MEDIA/DRIVE PARAMETER TABLE
981 ; IF ERROR:
982 ; AH = #DSKETTE_STATUS
983 ; CY = 1
984 -----
985 0337          SET_MEDIA  PROC NEAR
986 0337 E8 040F R      CALL XLAT_NEW      ; TRANSLATE STATE TO PRESENT ARCH.
987 033A F6 85 0090 R 01 TEST #0,SI         ; CHECK FOR CHANGE LINE AVAILABLE
988 033D 74 0F        JZ JUMP_40_TRACK  ; JUMP IF 40 TRACK DRIVE
989 033E 80 3E 0041 R 80 CMP #0,TIME_OUT    ; IF TIME OUT TELL CALLER
990 0343 C6 06 0041 R 00 JE SM_RTN         ; CLEAR STATUS
991 0345          SM_CMOS:
992 0345 E8 08EC R      CALL CMOS_TYPE    ; RETURN DRIVE TYPE IN (AL)
993 0348 72 38        JC MD_NOT_FND    ; ERROR IN CMOS
994 034A 0A C0        OR AL,AL         ; TEST FOR NO DRIVE
995 034C 74 58        JZ SM_RTN       ; RETURN IF SO
996 034E 8B 58        MOV DI,DR_TYPE_CHECK ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
997 0350 74 2F        JC MD_NOT_FND    ; TYPE NOT IN TABLE (BAD CMOS)
998 0352 67           PUSH DI          ; SAVE REG.
999 0353 23 DB        XOR BX,BX        ; BX = INDEX TO DR_TYPE TABLE
1000 0355 B9 0006      MOV CX,DR_CNT    ; CX = LOOP COUNT
1001 0358          DR_SEARCH:
1002 0358 2E 8A A7 0000 R MOV AH,CS:DR_TYPE[BX] ; GET DRIVE TYPE
1003 035B 80 E4 7F    AND AH,BITOFF    ; MASK OUT MSB
1004 035D 3A C4        CMP AL,AH        ; DRIVE TYPE MATCH ?
1005 035F 75 17        JNE NXT_MD       ; NO, CHECK NEXT DRIVE TYPE
1006
1007 0370 2E 8B BF 0001 R MOV DI,CS:WORD_PTR DR_TYPE[BX+1] ; DI = MEDIA/DRIVE PARAMETER TABLE
1008
1009 0375 2E 8A 65 04  MOV AH,CS:[DI].MD_SEC_TRK ; GET SECTOR/TRACK
1010 0379 38 66 00    CMP [BP],AH      ; MATCH ?
1011 037C 75 09        JNE NXT_MD       ; NO, CHECK NEXT MEDIA
1012 037E 2E 8A 65 0B MOV AH,CS:[DI].MD_MAX_TRK ; GET MAX. TRACK #
1013 0382 38 66 01    CMP [BP+1],AH    ; MATCH ?
1014 0385 74 0D        JE MD_FND        ; YES, GO GET RATE
1015 0387          NXT_MD:
1016 0387 83 C3 03    ADD BX,3         ; CHECK NEXT DRIVE TYPE
1017 038A E2 D8        DR_SEARCH
1018 038C 5F           POP DI          ; RESTORE REG.
1019 038D          MD_NOT_FND:
1020 038D C6 06 0041 R 0C MOV #0,BAD_CMD    ; ERROR, MEDIA TYPE NOT FOUND
1021 0392 EB 1D        JMP SHORT SM_RTN ; RETURN

```

```

1022
1023 0394 MD_FND: MOV AL,CS:[DI],MD_RATE ; GET RATE
1024 0394 2E: 8A 45 0C CMP AL,RATE_300 ; DOUBLE STEP REQUIRED FOR RATE 300
1025 0398 3C 40 JNE MD_SET
1026 039A 76 02 OR AL,DBL_STEP
1027 039C 0C 20
1028 039E MD_SET: MOV [BP+6],DI ; SAVE TABLE POINTER IN STACK
1029 039E 89 7E 04 OR AL,MED_DET ; SET MEDIA ESTABLISHED
1030 03A1 0C 10 POP DI ; RESTORE REG.
1031 03A3 8F AND #DSK_STATE[DI],NOT MED_DET+DBL_STEP+RATE_MSK ; CLEAR STATE
1032 03A4 80 A5 0090 R OF OR #DSK_STATE[DI],AL ; SET STATE
1033 03A9 08 85 0090 R MOV AX,CS ; SEGMENT MEDIA/DRIVE PARAMETER TABLE
1034 03AD 8C C5 MOV ES,AX ; ES IS SEGMENT OF TABLE
1035 03AF 8E C0
1036 03B1 SM_RTN: CALL XLAT_OLD ; TRANSLATE STATE TO COMPATIBLE MODE
1037 03B1 E8 0435 R CALL SETUP_END ; VARIOUS CLEANUPS
1038 03B4 E8 0854 R RET
1039 03B7 C5
1040 03B8 SET_MEDIA ENDP
1041
1042
1043 ; DR_TYPE_CHECK
1044 ; CHECK IF THE GIVEN DRIVE TYPE IN REGISTER (AL)
1045 ; IS SUPPORTED IN BIOS DRIVE TYPE TABLE
1046 ; ON ENTRY:
1047 ; AL = DRIVE TYPE
1048 ; ON EXIT:
1049 ; CS = SEGMENT OF MEDIA/DRIVE PARAMETER TABLE (CODE)
1050 ; CY = 0 DRIVE TYPE SUPPORTED
1051 ; BX = OFFSET TO MEDIA/DRIVE PARAMETER TABLE
1052 ; CY = DRIVE TYPE NOT SUPPORTED
1053 ; REGISTERS ALTERED: BX
1054
1055 03B8 DR_TYPE_CHECK PROC NEAR
1056 03B8 50 PUSH AX
1057 03B9 51 PUSH CX
1058 03BA 33 DB XOR BX,BX ; BX = INDEX TO DR_TYPE TABLE
1059 03BC 89 0006 MOV CX,DR_CNT ; CX = LOOP COUNT
1060 03BF TYPE_CHK: MOV AH,CS:DR_TYPE[BX] ; GET DRIVE TYPE
1061 03BF 2E: 8A AT 0000 R CMP AL,AH ; DRIVE TYPE MATCH ?
1062 03C4 3A C4 JE DR_TYPE_VALID ; YES, RETURN WITH CARRY RESET
1063 03C6 74 08 JNC ADD_BX ; CHECK NEXT DRIVE TYPE
1064 03C8 83 C3 03 ADD BX,3
1065 03CB E2 F2 LLOOP TYPE_CHK
1066 03CD F9 JMP SHORT TYPE_RTN ; DRIVE TYPE NOT FOUND IN TABLE
1067 03CE EB 05
1068 03D0 DR_TYPE_VALID: MOV BX,CS:WORD PTR DR_TYPE[BX+1] ; BX = MEDIA TABLE
1069 03D0 2E: 8B 9F 0001 R TYPE_RTN: POP CX
1070 03D5 POP AX
1071 03D5 89 RET
1072 03D6 58
1073 03D7 C3
1074 03D8 DR_TYPE_CHECK ENDP
1075
1076
1077 ; SEND_SPEC
1078 ; SEND THE SPECIFY COMMAND TO CONTROLLER USING DATA FROM
1079 ; THE DRIVE PARAMETER TABLE POINTED BY #DISK_POINTER
1080 ; ON ENTRY: #DISK_POINTER = DRIVE PARAMETER TABLE
1081 ; ON EXIT: NONE
1082 ; REGISTERS ALTERED: CX, DX
1083
1084 03D8 SEND_SPEC PROC NEAR
1085 03D8 50 PUSH AX ; SAVE AX
1086 03D9 2E: 8B 03F3 R MOV AX,OFFSET SPECBAC ; LOAD ERROR ADDRESS
1087 03DC 50 PUSH AX ; PUSH NEC_OUT ERROR RETURN
1088 03DD B4 03 MOV AH,03H ; SPECIFY COMMAND
1089 03DF E8 09F8 R CALL NEC_OUTPUT ; OUTPUT THE COMMAND
1090 03E2 2A D2 SUB DL,D ; FIRST SPECIFY BYTE
1091 03E4 E8 0905 R CALL GET_PARM ; GET PARAMETER TO AH
1092 03E7 E8 09F8 R CALL NEC_OUTPUT ; OUTPUT THE COMMAND
1093 03EA B6 01 MOV DL,1 ; SECOND SPECIFY BYTE
1094 03EC E8 0905 R CALL GET_PARM ; GET PARAMETER TO AH
1095 03EF E8 09F8 R CALL NEC_OUTPUT ; OUTPUT THE COMMAND
1096 03F2 58 POP AX ; POP ERROR RETURN
1097 03F3 SPECBAC: POP AX ; RESTORE ORIGINAL AX VALUE
1098 03F3 58 RET
1099 03F4 C3
1100 03F5 SEND_SPEC ENDP
1101
1102
1103 ; SEND_SPEC_MD
1104 ; SEND THE SPECIFY COMMAND TO CONTROLLER USING DATA FROM
1105 ; THE MEDIA/DRIVE PARAMETER TABLE POINTED BY (CS:BX)
1106 ; ON ENTRY: CS:BX = MEDIA/DRIVE PARAMETER TABLE
1107 ; ON EXIT: NONE
1108 ; REGISTERS ALTERED: AX
1109
1110 03F5 SEND_SPEC_MD PROC NEAR
1111 03F5 50 PUSH AX ; SAVE RATE DATA
1112 03F6 2E: 8B 040D R MOV AX,OFFSET SPEC_ESBAC ; LOAD ERROR ADDRESS
1113 03F9 50 PUSH AX ; PUSH NEC_OUT ERROR RETURN
1114 03FA B4 03 MOV AH,03H ; SPECIFY COMMAND
1115 03FC E8 09F8 R CALL NEC_OUTPUT ; OUTPUT THE COMMAND
1116 03FF 2E: 8A 27 MOV AH,CS:[BX].MD_SPEC1 ; GET 1ST SPECIFY BYTE
1117 0402 E8 09F8 R CALL NEC_OUTPUT ; OUTPUT THE COMMAND
1118 0405 2E: 8A 67 01 MOV AH,CS:[BX].MD_SPEC2 ; GET SECOND SPECIFY BYTE
1119 0409 E8 09F8 R CALL NEC_OUTPUT ; OUTPUT THE COMMAND
1120 040C 58 POP AX ; POP ERROR RETURN
1121 040D SPEC_ESBAC: POP AX ; RESTORE RATE
1122 040D 58 RET
1123 040E C3
1124 040F SEND_SPEC_MD ENDP

```

SECTION 5

```

1125 PAGE
1126 -----
1127 ; XLAT_NEW
1128 ; TRANSLATES DISKETTE STATE LOCATIONS FROM COMPATIBLE
1129 ; MODE TO NEW ARCHITECTURE.
1130 ;
1131 ; ON ENTRY: DI ; DRIVE
1132 -----
1133 XLAT_NEW PROC NEAR
1134 040F 83 FF 01 CMP DI,1 ; VALID DRIVE ?
1135 0412 77 IC JA XN_OUT ; IF INVALID BACK
1136 0414 80 B0 0090 R 00 CMP #DSK_STATE[DI],0 ; NO DRIVE ?
1137 0419 74 16 JZ DO_DET ; IF NO DRIVE ATTEMPT DETERMINE
1138 041B 8B CF MOV CX,DI ; CX = DRIVE NUMBER
1139 041D C0 E1 02 SHL CL,2 ; CL = SHIFT COUNT, A=0, B=4
1140 0420 AD 00BF R MOV AL,#HF_CNTRL ; DRIVE INFORMATION
1141 0423 02 C8 ROR AL,CL ; TO LOW NIBBLE
1142 0425 24 07 AND AL,DRV_DET+FMT_CAPA+TRK_CAPA ; KEEP DRIVE BITS
1143 0427 80 A5 0090 R F8 AND #DSK_STATE[DI],NOT DRV_DET+FMT_CAPA+TRK_CAPA
1144 042C 08 85 0090 R OR #DSK_STATE[DI],AL ; UPDATE DRIVE STATE
1145 0430
1146 0430 C3 XN_OUT: RET
1147
1148 0431
1149 0431 E8 0B32 R DO_DET: CALL DRIVE_DET ; TRY TO DETERMINE
1150 0434 C3 RET
1151
1152 0435 XLAT_NEW ENDP
1153 -----
1154 ; XLAT_OLD
1155 ; TRANSLATES DISKETTE STATE LOCATIONS FROM NEW
1156 ; ARCHITECTURE TO COMPATIBLE MODE.
1157 ;
1158 ; ON ENTRY: DI ; DRIVE
1159 -----
1160 XLAT_OLD PROC NEAR
1161 0435 83 FF 01 CMP DI,1 ; VALID DRIVE ?
1162 0438 77 73 JA XO_OUT ; IF INVALID BACK
1163 043A 80 B0 0090 R 00 CMP #DSK_STATE[DI],0 ; NO DRIVE ?
1164 043F 74 6C JZ XO_OUT ; IF NO DRIVE TRANSLATE DONE
1165
1166 ;----- TEST FOR SAVED DRIVE INFORMATION ALREADY SET
1167
1168 0441 8B CF MOV CX,DI ; CX = DRIVE NUMBER
1169 0443 C0 E1 02 SHL CL,2 ; CL = SHIFT COUNT, A=0, B=4
1170 0446 B4 02 MOV AH,FMT_CAPA ; LOAD MULTI DATA RATE BIT MASK
1171 0448 D2 CC ROR AH,CL ; ROTATE BY MASK
1172 044A 84 26 00BF R TEST #HF_CNTRL,AH ; MULTI-DATA RATE DETERMINED ?
1173 044E 75 16 JNZ SAVE_SET ; IF SO, NO NEED TO RE-SAVE
1174
1175 ;----- ERASE DRIVE BITS IN #HF_CNTRL FOR THIS DRIVE
1176
1177 0450 B4 07 MOV AH,DRV_DET+FMT_CAPA+TRK_CAPA ; MASK TO KEEP
1178 0452 D2 CC ROR AH,CL ; FIX MASK TO KEEP
1179 0454 F6 D4 NOT AH ; TRANSLATE MASK
1180 0456 20 26 00BF R AND #HF_CNTRL,AH ; KEEP BITS FROM OTHER DRIVE INTACT
1181
1182 ;----- ACCESS CURRENT DRIVE BITS AND STORE IN #HF_CNTRL
1183
1184 045A 8A 85 0090 R MOV AL,#DSK_STATE[DI] ; ACCESS STATE
1185 045E 24 07 AND AL,DRV_DET+FMT_CAPA+TRK_CAPA ; KEEP DRIVE BITS
1186 0460 D2 CC ROR AL,CL ; FIX FOR THIS DRIVE
1187 0462 08 06 00BF R OR #HF_CNTRL,AL ; UPDATE SAVED DRIVE STATE
1188
1189 ;----- TRANSLATE TO COMPATIBILITY MODE
1190
1191 0466 SAVE_SET:
1192 0466 8A A5 0090 R MOV AH,#DSK_STATE[DI] ; ACCESS STATE
1193 046A 8A FC MOV BH,AH ; TO BH FOR LATER
1194 046C 80 E4 C0 AND AH,RATE_MSK ; KEEP ONLY RATE
1195 046F 80 FC 00 CMP AH,RATE_500 ; RATE 500 ?
1196 0472 74 10 JZ CHK_144 ; YES, 1.2/1.2 OR 1.44/1.44
1197 0474 B0 01 MOV AL,#D3IU ; AL = 360 IN 1.2 UNESTABLISHED
1198 0476 80 FC 40 CMP AH,RATE_900 ; RATE 300 ?
1199 0479 75 16 JNZ CHK_250 ; NO, 360/360, 720/720 OR 720/1.44
1200 047B F6 C7 20 TEST BH,DSL_STEP ; YES, DOUBLE STEP ?
1201 047E 75 1D JNZ TST_DET ; YES, MUST BE 360 IN 1.2
1202
1203 0480 UNKNO:
1204 0480 B0 07 MOV AL,MED_UNK ; 'NONE OF THE ABOVE'
1205 0482 EB 20 JMP SHORT AL_SET ; PROCESS COMPLETE
1206
1207 0484 CHK_144:
1208 0484 E8 0BEC R CALL CMOS_TYPE ; RETURN DRIVE TYPE IN (AL)
1209 0487 72 F7 JC UNKNO ; ERROR, SET 'NONE OF THE ABOVE'
1210 0489 3C 02 CMP AL,02 ; 1.2MB DRIVE ?
1211 048B 75 F3 JNE UNKNO ; NO, GO SET 'NONE OF THE ABOVE'
1212 048D B0 02 MOV AL,MIDIU ; AL = 1.2 IN 1.2 UNESTABLISHED
1213 048F EB 0C JMP SHORT TST_DET
1214
1215 0491 CHK_250:
1216 0491 B0 00 MOV AL,#D3IU ; AL = 360 IN 360 UNESTABLISHED
1217 0493 80 FC 80 CMP AH,RATE_250 ; RATE 250 ?
1218 0496 75 E8 JNZ UNKNO ; IF SO FALL THRU
1219 0498 F6 C7 01 TEST BH,TRK_CAPA ; 80 TRACK CAPABILITY ?
1220 049B 75 E8 JNZ UNKNO ; IF SO JUMP, FALL THRU TEST DET
1221
1222 049D TST_DET:
1223 049D F6 C7 10 TEST BH,MED_DET ; DETERMINED ?
1224 04A0 74 02 JZ AL_SET ; IF NOT THEN SET
1225 04A2 04 03 ADD AL,3 ; MAKE DETERMINED/ESTABLISHED
1226
1227 04A4 AL_SET:
1228 04A4 80 A5 0090 R F8 AND #DSK_STATE[DI],NOT DRV_DET+FMT_CAPA+TRK_CAPA ; CLEAR DRIVE
1229 04A9 08 85 0090 R OR #DSK_STATE[DI],AL ; REPLACE WITH COMPATIBLE MODE
1230 04AD C3 XN_OUT: RET
1231 04AD C3
1232 04AE XLAT_OLD ENDP
    
```

```

1233 PAGE
1234 -----
1235 ; RD_WR_VF
1236 ; COMMON READ, WRITE AND VERIFY;
1237 ; MAIN LOOP FOR STATE RETRIES.
1238 ;
1239 ; ON ENTRY: AH: READ/WRITE/VERIFY NEC PARAMETER
1240 ; AL: READ/WRITE/VERIFY DMA PARAMETER
1241 ;
1242 ; ON EXIT: 0DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION ;
1243 -----
1244 RD_WR_VF PROC NEAR
1245 04AE 50 ; SAVE DMA, NEC PARAMETERS
1246 04AF EB 040F R ; TRANSLATE STATE TO PRESENT ARCH.
1247 04B2 EB 056A R ; INITIALIZE START AND END RATE
1248 04B5 55 ; RESTORE READ/WRITE/VERIFY
1249
1250 04B6 50 ; SAVE READ/WRITE/VERIFY PARAMETER
1251 04B7 EB 05E9 R ; MEDIA CHANGE AND RESET IF CHANGED
1252 04BA 58 ; RESTORE READ/WRITE/VERIFY
1253 ; MEDIA CHANGE ERROR OR TIME-OUT
1254 ;
1255 04BB 73 03 ;
1256 04BD E9 055B R ;
1257 04C0 ;
1258 04C0 50 ; SAVE READ/WRITE/VERIFY PARAMETER
1259 ;
1260 04C1 8A B5 0090 R ; GET RATE STATE OF THIS DRIVE
1261 04C5 80 E6 C0 ; KEEP ONLY RATE
1262 04C8 EB 08EC R ; CHOS TYPE
1263 04CB 72 46 ; RETURN DRIVE TYPE IN (AL)
1264 04CD 3C 01 ; ERROR IN CMOS
1265 04CF 75 0B ; 40 TRACK DRIVE?
1266 04D1 F6 85 0090 R 01 ; NO, BYPASS CMOS VALIDITY CHECK
1267 04D6 74 0F ; TEST
1268 04D8 B0 02 ; *DSR STATE[D1],TRK_CAPA ; CHECK FOR 40 TRACK DRIVE
1269 04DA EB 0B ; RWY_2 ; YES, CMOS IS CORRECT
1270 04DC ; RWY_2 ; CHANGE TO 1.2 M
1271 04DC 72 09 ; JZ ; CONTINUE
1272 04DE F6 85 0090 R 01 ; RWY_2 ; NO DRIVE SPECIFIED, CONTINUE
1273 04E3 75 02 ; TEST ; IS IT REALLY 40 TRACK?
1274 04E5 B0 01 ; JNZ ; NO, 80 TRACK
1275 04E7 ; MOV ; IT'S 40 TRACK, FIX CMOS VALUE
1276 ; RWY_2:
1277 04E7 ; OR ; TEST FOR NO DRIVE
1278 04E9 74 28 ; JZ ; ASSUME TYPE, USE MAX TRACK
1279 04EB EB 03B8 R ; CALL ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
1280 04EE 72 23 ; JC ; TYPE NOT IN TABLE (BAD CMOS)
1281
1282 ;--- SEARCH FOR MEDIA/DRIVE PARAMETER TABLE
1283 ;
1284 04F0 57 ; PUSH ; SAVE DRIVE #
1285 04F1 33 DB ; XOR ; BX = INDEX TO DR_TYPE TABLE
1286 04F3 89 0006 ; MOV ; CX,DR_CNT ; CX = LOOP COUNT
1287 04F6 ; RWY_DR_SEARCH:
1288 04F6 2E1 8A AT 0000 R ; AH,CS:DR_TYPE[BX] ; GET DRIVE TYPE
1289 04FB 80 E4 7F ; AND ; MASK OUT MSB
1290 04FE 3A C4 ; CMP ; DRIVE TYPE MATCH ?
1291 0500 75 0B ; JNE ; NO, CHECK NEXT DRIVE TYPE
1292 0502 ; RWY_DR_FND:
1293 0502 2E1 8B BF 0001 R ; MOV ; D1,WORD PTR CS:DR_TYPE[BX+1] ; D1 = MEDIA/DRIVE PARAMETER TABLE
1294 0507 ; RWY_MD_SEARCH:
1295 0507 2E1 3A 75 0C ; CMP ; DH,CS:[D1].MD_RATE ; MATCH ?
1296 050B 74 16 ; JE ; YES, GO GET 1ST SPECIFY BYTE
1297 050D ; RWY_NXT_MD:
1298 050D 83 C3 03 ; ADD ; BX,3 ; CHECK NEXT DRIVE TYPE
1299 0510 E2 E4 ; LOOP ; RWY_DR_SEARCH
1300 0512 5F ; POP ; RESTORE DRIVE #
1301
1302 ;--- ASSUME PRIMARY DRIVE IS INSTALLED AS SHIPPED
1303 ;
1304 0513 ; RWY_ASSUME:
1305 0513 BB 0012 R ; MOV ; BX,OFFSET MD_TBL1 ; POINT TO 40 TK 250 KBS
1306 0516 F6 85 0090 R 01 ; TEST ; *DSK_STATE[D1],TRK_CAPA ; TEST FOR 80 TRACK
1307 051B 74 09 ; JZ ; RWY_MD_FND1 ; MUST BE 40 TRACK
1308 051D BB 002C R ; MOV ; BX,OFFSET MD_TBL3 ; POINT TO 80 TK 500 KBS
1309 0520 EB 04 90 ; JMP ; RWY_MD_FND1 ; GO SET SPECIFY PARAMETERS
1310
1311 ;--- CS:BX POINTS TO MEDIA/DRIVE PARAMETER TABLE
1312 ;
1313 0523 ; RWY_MD_FND:
1314 ;
1315 0523 8B DF ; MOV ; BX,D1 ; BX = MEDIA/DRIVE PARAMETER TABLE
1316 0525 5F ; POP ; RESTORE DRIVE #
1317 0526 ; RWY_MD_FND1:
1318 ;
1319 ;--- SEND THE SPECIFY COMMAND TO THE CONTROLLER
1320 ;
1321 0526 EB 03F5 R ; CALL ; SEND_SPEC_MD
1322 0529 EB 063D R ; CALL ; CHK_LASRATE ; ZF=1 ATTEMPT RATE IS SAME AS LAST RATE
1323 052C 74 03 ; JZ ; RWY_DBL ; YES, SKIP SEND RATE COMMAND
1324 052E EB 0624 R ; CALL ; SEND_RATE ; SEND DATA RATE TO NEC
1325
1326 ;
1327 0531 ; RWY_DBL:
1328 0531 53 ; PUSH ; BX ; SAVE MEDIA/DRIVE PARAM ADDRESS
1329 0532 EB 086E R ; CALL ; SETUP_DBL ; CHECK FOR DOUBLE STEP
1330 0535 5B ; POP ; BX ; RESTORE ADDRESS
1331 0536 72 1A ; JC ; CHK_RET ; RESTORE ADDRESS
1332 0538 5B ; POP ; AX ; ERROR FROM READ ID, POSSIBLE RETRY
1333 0539 80 ; POP ; AX ; RESTORE NEC,DMA COMMAND
1334 053A 53 ; PUSH ; BX ; SAVE NEC COMMAND
1335 053B EB 064D R ; CALL ; DMA_SETUP ; SAVE MEDIA/DRIVE PARAM ADDRESS
1336 053E 5B ; POP ; BX ; SET UP THE DMA
1337 053F 5B ; POP ; AX ; RESTORE ADDRESS
1338 0540 72 1F ; JC ; RWY_BAC ; CHECK FOR DMA BOUNDARY ERROR
1339 0542 50 ; PUSH ; AX ; SAVE NEC COMMAND
1340 0543 53 ; PUSH ; BX ; SAVE MEDIA/DRIVE PARAM ADDRESS
1341 0544 EB 0700 R ; CALL ; NEC_INIT ; INITIALIZE NEC
1342 0547 5B ; POP ; BX ; RESTORE ADDRESS
1343 0548 72 08 ; JC ; CHK_RET ; ERROR - EXIT
1344 054A EB 0725 R ; CALL ; RWY_COM ; OP CODE COMMON TO READ/WRITE/VERIFY
1345 054D 72 03 ; JC ; CHK_RET ; ERROR - EXIT
1346 054F EB 075B R ; CALL ; NEC_TERM ; TERMINATE, GET STATUS, ETC.

```

SECTION 5

```

1347
1348 0552
1349 0552 E8 07E5 R      CHK_RET: CALL RETRY          ; CHECK FOR SETUP RETRY
1350 0555 58             POP AX              ; RESTORE READ/WRITE/VERIFY PARAMETER
1351 0556 73 03          JNC RWV_END        ; CY = 0 NO RETRY
1352 0558 E9 04B6 R      JMP DD_Again       ; CY = 1 MEANS RETRY
1353
1354 055B
1355 055B E8 07AD R      RWV_END: CALL DSTATE        ; ESTABLISH STATE IF SUCCESSFUL
1356 056E E8 0827 R      CALL NUM_TRANS     ; AL = NUMBER TRANSFERRED
1357
1358 0561
1359 0561 50             RWV_BAC: PUSH AX           ; BAD DMA ERROR ENTRY
1360 0562 E8 0435 R      CALL XLAT_OLD      ; SAVE NUMBER TRANSFERRED
1361 0565 58             POP AX              ; TRANSLATE STATE TO COMPATIBLE MODE
1362 0566 E8 0854 R      POP AX              ; RESTORE NUMBER TRANSFERRED
1363 0569 C3             RET                 ; VARIOUS CLEANUPS
1364 056A
1365
1366
1367
-----
; SETUP_STATE: INITIALIZES START AND END RATES.
;
1368 056A
1369 056A F6 85 0090 R 10  SETUP_STATE PROC NEAR
1370 056F 75 29          TEST #DSK_STATE[D1],MED_DET ; MEDIA DETERMINED ?
1371 0571 B6 0040          JNC JIC             ; NO STATES IF DETERMINED
1372 0574 F6 85 0090 R 04  MOV AX,RATE_500*H+RATE_300 ; AH = START RATE, AL = END RATE
1373 0579 74 0A          JZ AX_SET           ; DRIVE ?
1374 057B F6 85 0090 R 02  TEST #DSK_STATE[D1],FMT_CAPA ; DO NOT KNOW DRIVE
1375 0580 75 03          JNZ AX_SET          ; MULTI-RATE ?
1376 0582 B6 8080          MOV AX,RATE_250*X  ; JUMP IF YES
1377
1378
1379 0585
1380 0585 80 A5 0090 R 1F  AX_SET: AND #DSK_STATE[D1],NOT RATE_MSK+DBL_STEP ; TURN OFF THE RATE
1381 058A 08 A5 0090 R 08  OR #DSK_STATE[D1],AH ; RATE FIRST TO TRY
1382 058E 80 26 008B R F3  AND #LASTRATE,NOT STRT_MSK ; ERASE LAST TO TRY RATE BITS
1383 0593 C0 C8 04          ROR AL,4            ; TO OPERATION LAST RATE LOCATION
1384 059A                OR #LASTRATE,AL    ; LAST RATE
1385 059A C3             RET
1386 059B
1387
1388
1389
-----
; FMT_INIT: ESTABLISH STATE IF UNESTABLISHED AT FORMAT TIME.
;
1390 059B
1391 059B F6 85 0090 R 10  FMT_INIT PROC NEAR
1392 05A0 75 42          TEST #DSK_STATE[D1],MED_DET ; IS MEDIA ESTABLISHED
1393 05A2 E8 08EC R      CALL FI_ODT        ; IF SO RETURN
1394 05A5 72 3E          JC CL_DRV          ; RETURN DRIVE TYPE IN AL
1395 05A7 FE C8          DEC AL             ; ERROR IN CMOS ASSUME NO DRIVE
1396 05A9 78 3A          JS CL_DRV          ; MAKE ZERO ORIGIN
1397 05AB 8A A5 0090 R 08  MOV AH,#DSK_STATE[D1] ; NO DRIVE IF AL 0
1398 05AF 80 E4 0F          AND AH,NOT MED_DET+DBL_STEP+RATE_MSK ; AH = CURRENT STATE
1399 05B2 0A C0          OR AL,AL          ; CLEAR
1400 05B4 75 05          JNZ N_360         ; CHECK FOR 360
1401 05B6 80 CC 90          OR #N_360,AL      ; IF 360 WILL BE 0
1402 05B9 EB 25          JMP SHORT SKP_STATE ; ESTABLISH MEDIA
1403
1404 05BB
1405 05BB FE C8          N_360: DEC AL             ; CHECK FOR TYPE 3
1406 05BD 75 05          JNZ N_12          ; JUMP IF NOT
1407 05BF 80 CC 10          OR #N_12,AL      ; RETURN DRIVE TYPE IN AL
1408 05C2 EB 1C          JMP SHORT SKP_STATE ; SET FORMAT RATE
1409
1410 05C4
1411 05C4 FE C8          N_12: DEC AL             ; CHECK FOR TYPE 3
1412 05C6 75 0F          JNZ N_720        ; JUMP IF NOT
1413 05C8 F6 C4 04          TEST AR,DRV_DET   ; IS DRIVE DETERMINED
1414 05CB 74 10          JZ ISNT_12        ; TREAT AS NON 1.2 DRIVE
1415 05CD F6 C4 02          TEST AH,FMT_CAPA  ; IS 1.2M
1416 05D0 74 0B          JZ ISNT_12        ; IS 1.2M
1417 05D2 80 CC 50          OR #N_300,AL     ; JUMP IF NOT
1418 05D5 EB 09          JMP SHORT SKP_STATE ; RATE 300
1419
1420 05D7
1421 05D7 FE C8          N_720: DEC AL             ; CHECK FOR TYPE 4
1422 05D9 78 0A          JNZ CL_DRV        ; NO DRIVE, CMOS BAD
1423 05DB EB E2          JMP SHORT FI_RATE
1424
1425 05DD
1426 05DD 80 CC 90          ISNT_12: OR #AH,MED_DET+RATE_250 ; MUST BE RATE 280
1427
1428 05E0
1429 05E0 88 A5 0090 R 08  SKP_STATE: MOV #DSK_STATE[D1],AH ; STORE AWAY
1430
1431 05E4
1432 05E4 C3             FI_OUT: RET
1433
1434 05E5
1435 05E5 32 E4          CL_DRV: XOR AH,AH          ; CLEAR STATE
1436 05E7 EB F7          JMP SHORT SKP_STATE ; SAVE IT
1437 05E9
1438
1439
-----
; MED_CHANGE
; CHECKS FOR MEDIA CHANGE, RESETS MEDIA CHANGE,
; CHECKS MEDIA CHANGE AGAIN.
;
1440
1441
1442
1443
1444
1445
1446 05E9
1447 05E9 E8 0B28 R      MED_CHANGE PROC NEAR
1448 05EC 74 34          CALL READ_DSKCHNG ; READ DISK CHANGE LINE STATE
1449 05EE 80 A5 0090 R 08  JZ MC_OUT         ; BYPASS HANDLING DISK CHANGE LINE
1450
1451
1452
1453
1454
1455 05F3 8B CF          AND #DSK_STATE[D1],NOT MED_DET ; CLEAR STATE FOR THIS DRIVE
1456 05F5 80 01          AND #0,AL         ; CLEAR STATE FOR THIS DRIVE
1457 05F7 D2 E0          MOV AL,1          ; CL = DRIVE #
1458 05F9 F6 D0          SHL AL,CL         ; MOTOR ON BIT MASK
1459 05FB FA             NOT AL            ; TO APPROPRIATE POSITION
1460 05FC 20 06 003F R 08  AND #MOTOR_STATUS,AL ; KEEP ALL BUT MOTOR ON
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500

```

```

1461 0600 FB          STI          ; INTERRUPTS ENABLED
1462 0601 E8 091A R   CALL     MOTOR_ON    ; TURN MOTOR ON
1463
1464 ;----- THIS SEQUENCE OF SEEKS IS USED TO RESET DISKETTE CHANGE SIGNAL
1465
1466 0604 E8 00E7 R   CALL     DISK_RESET  ; RESET NEC
1467 0607 B5 01      MOV      CH,01H      ; MOVE TO CYLINDER 1
1468 0609 E8 0A24 R   CALL     SEEK        ; ISSUE SEEK
1469 060C 32 ED      XOR      CH,CH        ; MOVE TO CYLINDER 0
1470 060E E8 0A24 R   CALL     SEEK        ; ISSUE SEEK
1471 0611 C6 05 0041 R 06 MOV      #DSKETTE_STATUS,MEDIA_CHANGE ; STORE IN STATUS
1472
1473 0616 E8 0B28 R   OK1: CALL    READ_DSKCHNG ; CHECK MEDIA CHANGED AGAIN
1474 0619 74 05      JZ       OK2         ; IF ACTIVE, NO DISKETTE, TIMEOUT
1475
1476 061B C6 06 0041 R 80 OK4: MOV      #DSKETTE_STATUS,TIME_OUT; TIMEOUT IF DRIVE EMPTY
1477
1478 0620 F9          OK2: STC          ; MEDIA CHANGED, SET CY
1479 0621 C3          RET
1480 0622            MC_OUT: CLC          ; NO MEDIA CHANGED, CLEAR CY
1481 0622 F8          RET
1482 0623 C3          RET
1483 0624
1484 ;-----
1485 ; SEND_RATE
1486 ; SENDS DATA RATE COMMAND TO NEC
1487 ; ON ENTRY:  DI = DRIVE #
1488 ; ON EXIT:   NONE
1489 ; REGISTERS ALTERED: DX
1490 ;-----
1491 0624            SEND_RATE PROC    NEAR
1492
1493 0624 50          PUSH   AX          ; SAVE REG.
1494 0625 80 26 008B R 3F AND     #LAstrate,NOT SEND_MSK ; ELSE CLEAR LAST RATE ATTEMPTED
1495 062A 8A B5 0090 R MOV     AL,#DSK_STATE[D1] ; GET RATE STATE OF THIS DRIVE
1496 062E 24 C0      AND     AL,SEND_MSK ; KEEP ONLY RATE BITS
1497 0630 08 06 008B R OR      #LAstrate,AL ; SAVE NEW RATE FOR NEXT CHECK
1498 0634 C0 C0 02  ROL     AL,2 ; MOVE TO BIT OUTPUT POSITIONS
1499 0637 BA 03F7   MOV     DX,03F7H ; OUTPUT NEW DATA RATE
1500 063A EE          OUT     DX,AL
1501
1502 063B 58          POP    AX          ; RESTORE REG.
1503 063C C3          RET
1504 063D
1505 ;-----
1506 ; CHK_LAstrate
1507 ; CHECK PREVIOUS DATA RATE SENT TO THE CONTROLLER.
1508 ; ON ENTRY:  DI = DRIVE #
1509 ; ON EXIT:   ZF = 1 DATA RATE IS THE SAME AS LAST RATE SENT TO NEC ;
1510 ;           ZF = 0 DATA RATE IS DIFFERENT FROM LAST RATE ;
1511 ; REGISTERS ALTERED: NONE
1512 ;-----
1513 063D            CHK_LAstrate PROC    NEAR
1514
1515 063D 50          PUSH   AX          ; SAVE REG
1516 063E 8A 26 008B R 3F MOV     AH,#LAstrate ; GET LAST DATA RATE SELECTED
1517 0642 8A B5 0090 R MOV     AL,#DSK_STATE[D1] ; GET RATE STATE OF THIS DRIVE
1518 0646 25 C0C0   AND     AX,SEND_MSK*X ; KEEP ONLY RATE BITS OF BOTH
1519 0649 3A C4      CMP     AL,AH ; COMPARE TO PREVIOUSLY TRIED
1520
1521 064B 58          POP    AX          ; RESTORE REG.
1522 064C C3          RET
1523 064D
1524 064E            CHK_LAstrate ENDP
1525
1526
1527 SUBTTL (DSK3.ASM)
    
```

SECTION 5

```

1528          PAGE
1529          |-----|
1530          | DMA_SETUP
1531          | THIS ROUTINE SETS UP THE DMA FOR READ/WRITE/VERIFY
1532          | OPERATIONS.
1533          |
1534          | ON ENTRY:  AL = DMA COMMAND
1535          |
1536          | ON EXIT:  @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
1537          |-----|
1538 064D          DMA_SETUP PROC NEAR
1539 064D FA          CLI
1540 064E E6 0C          OUT DMA+12,AL
1541 0650 EB 00          JMP $+2
1542 0652 E6 0B          OUT DMA+11,AL
1543 0654 3C 42          CMP AL,42H
1544 0656 75 04          JNE NOT_VERF
1545 0658 33 C0          XOR AX,AX
1546 065A EB 10          JMP SHORT J33
1547 065C          NOT_VERF:
1548 065C 8C C0          MOV AX,ES
1549 065E C1 C0 04          ROL AX,4
1550 0661 8A EA          MOV CH,AL
1551 0663 24 F0          AND AL,11110000B
1552 0665 03 46 02          ADD AX,[BP+2]
1553 0668 73 02          JNC J33
1554 066A FE C5          INC CH
1555 066C          J33:
1556 066C 50          PUSH AX
1557 066D E6 04          OUT DMA+4,AL
1558 066F EB 00          JMP $+2
1559 0671 8A C4          MOV AL,AH
1560 0673 E6 04          OUT DMA+4,AL
1561 0675 8A C5          MOV AL,CH
1562 0677 EB 00          JMP $+2
1563 0679 24 0F          AND AL,00001111B
1564 067B E6 01          OUT 081H,AL
1565          |-----|
1566          |-----| DETERMINE COUNT
1567          |
1568 067D 8B C6          MOV AX,S1
1569 067F 86 C4          XCHG AL,AH
1570 0681 2A C0          SUB AL,AL
1571 0683 D1 E8          SHR AX,1
1572 0685 50          PUSH AX
1573 0686 B2 03          MOV DL,3
1574 0688 EB 09 05 R          CALL GET_PARM
1575 068B 8A CC          MOV CL,AH
1576 068D 58          POP AX
1577 068E D3 F0          SHL AX,CL
1578 0690 48          DEC AX
1579 0691 50          PUSH AX
1580 0692 E6 05          OUT DMA+5,AL
1581 0694 EB 00          JMP $+2
1582 0696 8A C4          MOV AL,AH
1583 0698 E6 05          OUT DMA+5,AL
1584 069A FB          STI
1585 069B 59          POP CX
1586 069C 58          POP AX
1587 069D 03 C1          ADD AX,CX
1588 069F B0 02          MOV AL,2
1589 06A1 EB 00          JMP $+2
1590 06A3 E6 0A          OUT DMA+10,AL
1591          |
1592 06A5 73 05          JNC NO_BAD
1593 06A7 C6 06 00 41 R 09          MOV @DSKETTE_STATUS,DMA_BOUNDARY
1594          |
1595          NO_BAD:
1596 06AC          RET
1597 06AD          ENDP
DMA_SETUP
    
```

```

PAGE
1598 ; FMTDMA_SET
1599 ; THIS ROUTINE SETS UP THE DMA CONTROLLER FOR A FORMAT
1600 ; OPERATION.
1601 ;
1602 ;
1603 ; ON ENTRY:  NOTHING REQUIRED
1604 ;
1605 ;
1606 ; ON EXIT:   #DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
1607 ;-----
1608 FMTDMA_SET PROC NEAR
1609     MOV     AL,04AH                ; WILL WRITE TO THE DISKETTE
1610     CLI     DMA+12,AL              ; DISABLE INTERRUPTS DURING DMA SET-UP
1611     OUT     $+2                    ; SET THE FIRST/LAST F/F
1612     JMP     DMA+11,AL              ; WAIT FOR I/O
1613     OUT     DMA+11,AL              ; OUTPUT THE MODE BYTE
1614 ;
1615     MOV     AX,ES                  ; GET THE ES VALUE
1616     ROL     CH,AL                  ; ROTATE LEFT
1617     MOV     CH,AL                  ; GET HIGHEST NIBBLE OF ES TO CH
1618     AND     AL,11110000B          ; ZERO THE LOW NIBBLE FROM SEGMENT
1619     ADD     AX,[BP+2]              ; TEST FOR CARRY FROM ADDITION
1620     JCSA   J33A1                  ; CARRY MEANS HIGH 4 BITS MUST BE INC
1621     JNC     CH                      ;
1622     INC     CH                      ;
1623     PUSH   AX                      ; SAVE START ADDRESS
1624     OUT     DMA+4,AL                ; OUTPUT LOW ADDRESS
1625     JMP     $+2                    ; WAIT FOR I/O
1626     MOV     AL,AH                  ; OUTPUT HIGH ADDRESS
1627     OUT     DMA+4,AL                ; GET HIGH 4 BITS
1628     MOV     AL,CH                  ; I/O WAIT STATE
1629     JMP     $+2                    ;
1630     AND     AL,00001111B          ;
1631     OUT     081H,AL                ; OUTPUT HIGH 4 BITS TO PAGE REGISTER
1632 ;-----
1633 ;----- DETERMINE COUNT
1634 ;
1635     MOV     DL,4                    ; SECTORS/TRACK VALUE IN PARM TABLE
1636     CALL   GET_PARM                ;
1637     XCHG   AL,AH                    ; AL = SECTORS/TRACK VALUE
1638     SUB     AH,AH                    ; AX = SECTORS/TRACK VALUE
1639     SHL     AX,2                    ; AX = SEC/TRK * 4 (OFFSET FOR C,H,R,N)
1640     DEC     AX                      ; -1 FOR DMA VALUE
1641     PUSH   AX                      ; SAVE # OF BYTES TO BE TRANSFERRED
1642     OUT     DMA+5,AL                ; LOW BYTE OF COUNT
1643     JMP     $+2                    ; WAIT FOR I/O
1644     MOV     AL,AH                    ; HIGH BYTE OF COUNT
1645     OUT     DMA+5,AL                ; RE-ENABLE INTERRUPTS
1646     STI     EB                      ; RECOVER COUNT VALUE
1647     POP     CX                      ; RECOVER ADDRESS VALUE
1648     POP     AX                      ; ADD. TEST FOR 64K OVERFLOW
1649     ADD     AX,CX                    ; MODE FOR 8237
1650     MOV     AL,2                    ; WAIT FOR I/O
1651     JMP     $+2                    ;
1652     OUT     DMA+10,AL               ; INITIALIZE THE DISKETTE CHANNEL
1653 ;
1654     JNC     FMTDMA_OK                ; CHECK FOR ERROR
1655     MOV     #DSKETTE_STATUS,DMA_BOUNDARY ; SET ERROR
1656 ;-----
1657 FMTDMA_OK:  RET                     ; CY SET BY ABOVE IF ERROR
1658 ;-----
1659 FMTDMA_SET ENDP
1660 ;-----
1661 ; NEC_INIT
1662 ; THIS ROUTINE SEEKS TO THE REQUESTED TRACK AND
1663 ; INITIALIZES THE NEC FOR THE READ/WRITE/VERIFY/FORMAT
1664 ; OPERATION.
1665 ;
1666 ; ON ENTRY:  AH ; NEC COMMAND TO BE PERFORMED
1667 ;
1668 ; ON EXIT:   #DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
1669 ;-----
1670 NEC_INIT PROC NEAR
1671     PUSH   AX                      ; SAVE NEC COMMAND
1672     CALL   MOTOR_ON                 ; TURN MOTOR ON FOR SPECIFIC DRIVE
1673 ;----- DO THE SEEK OPERATION
1674 ;
1675     MOV     CH,[BP+1]                ; CH = TRACK #
1676     CALL   SEEK                      ; MOVE TO CORRECT TRACK
1677     POP     AX                      ; RECOVER COMMAND
1678     JC     ER_1                      ; ERROR ON SEEK
1679     MOV     BX,OFFSET ER_1           ; LOAD ERROR ADDRESS
1680     PUSH   BX                      ; PUSH NEC_OUT ERROR RETURN
1681 ;----- SEND OUT THE PARAMETERS TO THE CONTROLLER
1682 ;
1683     CALL   NEC_OUTPUT                ; OUTPUT THE OPERATION COMMAND
1684     MOV     AX,S1                    ; AH = HEAD #
1685     MOV     BX,D1                    ; BL = DRIVE #
1686     SAL     AH,2                    ; MOVE IT TO BIT 2
1687     AND     AH,00000100B            ; ISOLATE THAT BIT
1688     OR     DR,AH                    ; DR IN THE DRIVE NUMBER
1689     CALL   NEC_OUTPUT                ; FALL THRU CY SET IF ERROR
1690     POP     BX                      ; THROW AWAY ERROR RETURN
1691 ER_1:  RET
1692 ;-----
1693 NEC_INIT ENDP
1694 ;-----
1695 ; RWV_COM
1696 ; THIS ROUTINE SENDS PARAMETERS TO THE NEC SPECIFIC
1697 ; TO THE READ/WRITE/VERIFY OPERATIONS.
1698 ;
1699 ; ON ENTRY:  CS:BX = ADDRESS OF MEDIA/DRIVE PARAMETER TABLE
1700 ;
1701 ; ON EXIT:   #DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
1702 ;-----
1703 RWV_COM PROC NEAR
1704     MOV     AX,OFFSET ER_2           ; LOAD ERROR ADDRESS
1705     PUSH   AX                      ; PUSH NEC_OUT ERROR RETURN
1706     MOV     AH,[BP+1]                ; OUTPUT TRACK #
1707     CALL   NEC_OUTPUT                ;
1708     MOV     AX,S1                    ; OUTPUT HEAD #
1709     CALL   NEC_OUTPUT                ;
1710     MOV     AX,[BP]                  ; OUTPUT SECTOR #
1711 ;-----
    
```

SECTION 5

```

1712 0737 E8 09F8 R      CALL    NEC_OUTPUT          ; BYTES/SECTOR PARAMETER FROM BLOCK
1713 073A B2 03         MOV     DL,3                ; TO THE NEC
1714 073C E8 0905 R      CALL    GET_PARM           ; OUTPUT TO CONTROLLER
1715 073F E8 09F8 R      CALL    NEC_OUTPUT          ; EOT PARAMETER FROM BLOCK
1716 0742 B2 04         MOV     DL,4                ; TO THE NEC
1717 0744 E8 0905 R      CALL    GET_PARM           ; OUTPUT TO CONTROLLER
1718 0747 E8 09F8 R      CALL    NEC_OUTPUT          ; OUTPUT TO CONTROLLER
1719
1720 074A 2E 1 8A 67 05   MOV     AH,C3:[BX].MD_GAP   ; GET GAP LENGTH
1721 074E E8 09F8 R      R15:  CALL    NEC_OUTPUT          ;
1722 0751 B2 06         MOV     DL,6                ; DTL PARAMETER FROM BLOCK
1723 0753 E8 0905 R      CALL    GET_PARM           ; TO THE NEC
1724 0756 E8 09F8 R      CALL    NEC_OUTPUT          ; OUTPUT TO CONTROLLER
1725 0759 B8             POP     AX                  ; THROW AWAY ERROR EXIT
1726 075A
1727 075A C3             RET
1728 075B
1729
1730 ; NEC_TERM
1731 ; THIS ROUTINE WAITS FOR THE OPERATION THEN ACCEPTS
1732 ; THE STATUS FROM THE NEC FOR THE READ/WRITE/VERIFY/
1733 ; FORMAT OPERATION.
1734 ;
1735 ; ON EXIT:  #DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION ;
1736 ;-----
1737 075B PROC    NEAR
1738
1739 ;----- LET THE OPERATION HAPPEN
1740
1741 075B 56             PUSH    SI                  ; SAVE HEAD #, # OF SECTORS
1742 075C E8 0AC1 R      CALL    WAIT_INT           ; WAIT FOR THE INTERRUPT
1743 075F 9C             PUSHF
1744 0760 E8 0AE9 R      CALL    RESULTS           ; GET THE NEC STATUS
1745 0763 72 45         JC     SET_END_POP        ;
1746 0765 9D             POPF
1747 0766 72 3A         JC     SET_END             ; LOOK FOR ERROR
1748
1749 ;----- CHECK THE RESULTS RETURNED BY THE CONTROLLER
1750
1751 0768 FC             CLD                        ; SET THE CORRECT DIRECTION
1752 0769 BE 0042 R      MOV     SI,OFFSET #NEC_STATUS ; POINT TO STATUS FIELD
1753 076C AC             LODS                       ; GET ST0
1754 076D 24 C0         AND     AL,11000000B      ; TEST FOR NORMAL TERMINATION
1755 076F 74 31         JZ     SET_END            ;
1756 0771 3C 40         CMP     AL,01000000B      ; TEST FOR ABNORMAL TERMINATION
1757 0773 75 27         JNZ    J18                ; NOT ABNORMAL, BAD NEC
1758
1759 ;----- ABNORMAL TERMINATION, FIND OUT WHY
1760
1761 0778 AC             LODS                       ; GET ST1
1762 077A D0 E0         SAL    AL,1               ; TEST FOR EOT FOUND
1763 077B B4 04         MOV     AH,RECORD_NOT_FND
1764 077A 72 22         JC     J19                ;
1765 077C C0 E0 02     SAL    AL,2               ;
1766 077F B4 10         MOV     AH,BAD_CRC
1767 0781 72 1B         JC     J19                ;
1768 0783 D0 00         SAL    AL,1               ; TEST FOR DMA OVERRUN
1769 0785 B4 08         MOV     AH,BAD_DMA
1770 0787 72 15         JC     J19                ;
1771 0789 C0 E0 02     SAL    AL,2               ; TEST FOR RECORD NOT FOUND
1772 078C B4 04         MOV     AH,RECORD_NOT_FND
1773 078E 72 0E         JC     J19                ;
1774 0790 D0 E0         SAL    AL,1               ;
1775 0792 B4 03         MOV     AH,WRITE_PROTECT  ; TEST FOR WRITE_PROTECT
1776 0794 72 08         JC     J19                ;
1777 0796 D0 E0         SAL    AL,1               ; TEST MISSING ADDRESS MARK
1778 0798 B4 02         MOV     AH,BAD_ADDR_MARK
1779 079A 72 02         JC     J19                ;
1780
1781 ;----- NEC MUST HAVE FAILED
1782 079C
1783 079C B4 20         J18:  MOV     AH,BAD_NEC
1784 079E
1785 079E 08 26 0041 R    OR     #DSKETTE_STATUS,AH
1786 07A2
1787 07A2 80 3E 0041 R 01 SET_END:  CMP     #DSKETTE_STATUS,1 ; SET ERROR CONDITION
1788 07A7 F5             CMC
1789 07A8 5E             POP     SI                  ; RESTORE HEAD #, # OF SECTORS
1790 07A9 C3             RET
1791
1792 07AA
1793 07AA 9D             SET_END_POP:  POPF
1794 07AB EB F5             JMP     SHORT SET_END
1795 07AD
1796
1797 ;----- ESTABLISH STATE UPON SUCCESSFUL OPERATION.
1798 ;-----
1799 07AD PROC    NEAR
1800 07AD 80 3E 0041 R 00 CMP     #DSKETTE_STATUS,0 ; CHECK FOR ERROR
1801 07B2 75 30         JNZ    SETBAC             ; IF ERROR JUMP
1802 07B4 80 0D 0090 R 10 OR     #DSK_STATE[D1],MED_DET ; NO ERROR, MARK MEDIA AS DETERMINED
1803 07B9 F6 85 0090 R 04 TEST   #DSK_STATE[D1],DRV_DET ; DRIVE DETERMINED ?
1804 07BE 75 24         JNZ    SETBAC             ; IF DETERMINED NO TRY TO DETERMINE
1805 07C0 8A 85 0090 R MOV     AL,#DSK_STATE[D1] ; LOAD STATE
1806 07C4 24 C0         AND     AL,RATE_MASK      ; KEEP ONLY RATE
1807 07C6 3C 80         CMP     AL,RATE_250      ; RATE 250 ?
1808 07C8 75 15         JNE    M_12               ; NO, MUST BE 1.2M OR 1.44M DRV
1809
1810 ;--- CHECK IF IT IS 1.44M
1811
1812 07CA E8 08EC R      CALL    CMOS_TYPE         ; RETURN DRIVE TYPE IN (AL)
1813 07CD 72 10         JC     M_12               ; CMOS BAD
1814 07CF 3C 04         CMP     AL,04             ; 1.44MB DRIVE ?
1815 07D1 74 0C         JE     M_12               ; YES
1816 07D3
1817 07D3 80 A5 0090 R FD M_720:  AND     #DSK_STATE[D1],NOT_FMT_CAPA ; TURN OFF FORMAT CAPA
1818 07D8 80 8D 0090 R 04 OR     #DSK_STATE[D1],DRV_DET  ; MARK DRIVE DETERMINED
1819 07DD EB 05             JMP     BACK              ; BACK
1820
1821 07DF 80 8D 0090 R 06 M_12:  OR     #DSK_STATE[D1],DRV_DET+_FMT_CAPA ; TURN ON DETERMINED & FMT CAPA
1822
1823 07E4
1824 07E4 C3             SETBAC:  RET
1825 07E5
1826 07E5
1827 07E5
1828 07E5
1829 07E5
1830 07E5
1831 07E5
1832 07E5
1833 07E5
1834 07E5
1835 07E5
1836 07E5
1837 07E5
1838 07E5
1839 07E5
1840 07E5
1841 07E5
1842 07E5
1843 07E5
1844 07E5
1845 07E5
1846 07E5
1847 07E5
1848 07E5
1849 07E5
1850 07E5
1851 07E5
1852 07E5
1853 07E5
1854 07E5
1855 07E5
1856 07E5
1857 07E5
1858 07E5
1859 07E5
1860 07E5
1861 07E5
1862 07E5
1863 07E5
1864 07E5
1865 07E5
1866 07E5
1867 07E5
1868 07E5
1869 07E5
1870 07E5
1871 07E5
1872 07E5
1873 07E5
1874 07E5
1875 07E5
1876 07E5
1877 07E5
1878 07E5
1879 07E5
1880 07E5
1881 07E5
1882 07E5
1883 07E5
1884 07E5
1885 07E5
1886 07E5
1887 07E5
1888 07E5
1889 07E5
1890 07E5
1891 07E5
1892 07E5
1893 07E5
1894 07E5
1895 07E5
1896 07E5
1897 07E5
1898 07E5
1899 07E5
1900 07E5
1901 07E5
1902 07E5
1903 07E5
1904 07E5
1905 07E5
1906 07E5
1907 07E5
1908 07E5
1909 07E5
1910 07E5
1911 07E5
1912 07E5
1913 07E5
1914 07E5
1915 07E5
1916 07E5
1917 07E5
1918 07E5
1919 07E5
1920 07E5
1921 07E5
1922 07E5
1923 07E5
1924 07E5
1925 07E5
1926 07E5
1927 07E5
1928 07E5
1929 07E5
1930 07E5
1931 07E5
1932 07E5
1933 07E5
1934 07E5
1935 07E5
1936 07E5
1937 07E5
1938 07E5
1939 07E5
1940 07E5
1941 07E5
1942 07E5
1943 07E5
1944 07E5
1945 07E5
1946 07E5
1947 07E5
1948 07E5
1949 07E5
1950 07E5
1951 07E5
1952 07E5
1953 07E5
1954 07E5
1955 07E5
1956 07E5
1957 07E5
1958 07E5
1959 07E5
1960 07E5
1961 07E5
1962 07E5
1963 07E5
1964 07E5
1965 07E5
1966 07E5
1967 07E5
1968 07E5
1969 07E5
1970 07E5
1971 07E5
1972 07E5
1973 07E5
1974 07E5
1975 07E5
1976 07E5
1977 07E5
1978 07E5
1979 07E5
1980 07E5
1981 07E5
1982 07E5
1983 07E5
1984 07E5
1985 07E5
1986 07E5
1987 07E5
1988 07E5
1989 07E5
1990 07E5
1991 07E5
1992 07E5
1993 07E5
1994 07E5
1995 07E5
1996 07E5
1997 07E5
1998 07E5
1999 07E5
2000 07E5
    
```

```

1826 ; RETRY DETERMINES WHETHER A RETRY IS NECESSARY. IF RETRY IS
1827 ; REQUIRED THEN STATE INFORMATION IS UPDATED FOR RETRY.
1829 ; ON EXIT: CY = 1 FOR RETRY, CY = 0 FOR NO RETRY
1831
1832
1833 RETRY PROC NEAR
1834 07E5 80 3E 0041 R 00 CMP #DSKETTE_STATUS,0 ; GET STATUS OF OPERATION
1835 07E4 74 39 JZ NO_RETRY ; SUCCESSFUL OPERATION
1836 07EC 80 3E 0041 R 80 CMP #DSKETTE_STATUS,TIME_OUT ; IF TIME OUT NO RETRY
1837 07F1 74 32 JZ NO_RETRY
1838 07F3 8A A5 0090 R MOV AH,#DSK_STATE[D1] ; GET MEDIA STATE OF DRIVE
1839 07F7 F4 C4 10 TEST AH,#MED_DET ; ESTABLISHED/DETERMINED ?
1840 07FA 75 29 JNZ NO_RETRY ; IF ESTABLISHED STATE THEN TRUE ERROR
1841 07FC 80 E4 C0 AND AH,RATE_MSK ; ISOLATE RATE
1842 07FF 8A 2E 008B R MOV CH,#PLAstrate ; GET START OPERATION STATE
1843 0803 C5 04 ROL CH,4 ; TO CORRESPONDING BITS
1844 0806 80 E5 C0 AND CH,RATE_MSK ; ISOLATE RATE BITS
1845 0809 3A EC CMP CH,AH ; ALL RATES TRIED
1846 080B 74 18 JE NO_RETRY ; IF YES, THEN TRUE ERROR
1847
1848 ; SETUP STATE INDICATOR FOR RETRY ATTEMPT TO NEXT RATE
1849 00000000B (500) -> 10000000B (250)
1850 10000000B (250) -> 01000000B (500)
1851 01000000B (500) -> 00000000B (500)
1852
1853 0800 80 FC 01 CMP AH,RATE_500+1 ; SET CY FOR RATE 500
1854 0810 00 DC RCR AH,1 ; TO NEXT STATE
1855 0812 80 E4 C0 AND AH,RATE_MSK ; KEEP ONLY RATE BITS
1856 0815 80 A5 0090 R 1F AND #DSK_STATE[D1],NOT RATE_MSK+DBL_STEP ; RATE, DBL STEP OFF
1857 081A 08 A5 0090 R OR #DSK_STATE[D1],AH ; TURN ON NEW RATE
1858 081E 06 06 0041 R 00 MOV #DSKETTE_STATUS,0 ; RESET STATUS FOR RETRY
1859 0823 F9 STC ; SET CARRY FOR RETRY
1860 0824 C3 RET ; RETRY RETURN
1861
1862 0825 NO_RETRY: CLC ; CLEAR CARRY NO RETRY
1863 0825 F8 RET ; NO RETRY RETURN
1864 0826 C3 RET
1865 0827 ENDP
1866
1867 ; NUM_TRANS THIS ROUTINE CALCULATES THE NUMBER OF SECTORS THAT
1868 ; WERE ACTUALLY TRANSFERRED TO/FROM THE DISKETTE.
1869
1870 ; ON ENTRY: [BP+1] = TRACK
1871 ; SI=H1 = HEAD
1872 ; [BP] = START SECTOR
1873
1874 ; ON EXIT: AL = NUMBER ACTUALLY TRANSFERRED
1875
1876
1877 NUM_TRANS PROC NEAR
1878 0827 32 C0 XOR AL,AL ; CLEAR FOR ERROR
1879 0829 80 3E 0041 R 00 CMP #DSKETTE_STATUS,0 ; CHECK FOR ERROR
1880 082E 75 23 JNZ NT_OUT ; IF ERROR 0 TRANSFERRED
1881 0830 B2 04 MOV DL,4 ; SECTORS/TRACK OFFSET TO DL
1882 0832 EB 0905 R CALL GET_PARM ; AH = SECTORS/TRACK
1883 0835 8A 1E 0047 R MOV BL,#NEC_STATUS+5 ; GET ENDING SECTOR
1884 0839 9B EC MOV CX,S1 ; CH = HEAD # STARTED
1885 083B 3A 2E 0046 R CMP CH,#NEC_STATUS+4 ; GET HEAD ENDED UP ON
1886 083F 75 0B JNZ DIF_HD ; IF ON SAME HEAD, THEN NO ADJUST
1887
1888 0841 8A 2E 0045 R MOV CH,#NEC_STATUS+3 ; GET TRACK ENDED UP ON
1889 0845 3A 6E 01 CMP CH,[BP+1] ; IS IT ASKED FOR TRACK
1890 0848 74 04 JZ SAME_TRK ; IF SAME TRACK NO INCREASE
1891
1892 084A 02 DC ADD BL,AH ; ADD SECTORS/TRACK
1893 084C DIF_HD: ADD BL,AH ; ADD SECTORS/TRACK
1894 084C 02 DC SAME_TRK: ADD BL,AH
1895 084E SAME_TRK: SUB BL,[BP] ; SUBTRACT START FROM END
1896 084E 2A 5E 00 MOV AL,BL ; TO AL
1897 0851 8A C3
1898
1899 0853 NT_OUT: RET
1900 0853 C3
1901 0854 NUM_TRANS ENDP
1902
1903 ; SETUP_END RESTORES #MOTOR COUNT TO PARAMETER PROVIDED IN TABLE
1904 ; AND LOADS #DSKETTE_STATUS TO AH, AND SETS CY.
1905 ; ON EXIT: AH, #DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
1906
1907
1908
1909 0854 SETUP_END PROC NEAR
1910 0854 B2 02 MOV DL,2 ; GET THE MOTOR WAIT PARAMETER
1911 0856 50 PUSH AX ; SAVE NUMBER TRANSFERRED
1912 0857 EB 0905 R CALL GET_PARM ; GET #MOTOR_COUNT,AH
1913 085A 8B 26 0040 R MOV #MOTOR_COUNT,AH ; STORE UPON RETURN
1914 085E 58 POP AX ; RESTORE NUMBER TRANSFERRED
1915 085F 8A 26 0041 R MOV AH,#DSKETTE_STATUS ; GET STATUS OF OPERATION
1916 0863 0A E4 OR AH,AH ; CHECK FOR ERROR
1917 0865 74 02 JZ NUM_ERR ; NO ERROR
1918 0867 32 C0 XOR AL,AL ; CLEAR NUMBER RETURNED
1919
1920 0869 NUM_ERR: CMP AH,1 ; SET THE CARRY FLAG TO INDICATE
1921 0869 80 FC 01 CMC ; SUCCESS OR FAILURE
1922 086C F5 RET
1923 086D C3
1924 086E SETUP_END ENDP
    
```

SECTION 5

```

1925 PAGE
1926
1927
1928
1929
1930
1931
1932
1933 086E
1934 086E 8A A5 0090 R
1935 0872 F6 C4 10
1936 0876 78 5E
1937
1938
1939
1940 0877 C6 06 003E R 00
1941 087C E8 091A R
1942 087F B5 00
1943 0881 E8 0A24 R
1944 0884 E8 08D7 R
1945 0887 72 37
1946
1947
1948
1949 0889 B9 0450
1950 08C F6 08 0090 R 01
1951 0891 74 02
1952 0893 B1 A0
1953
1954
1955
1956
1957
1958 0895 C6 06 0040 R FF
1959 089A 51
1960 089B C6 06 0041 R 00
1961 08A0 33 C0 10
1962 08A2 D0 ED
1963 08A4 C0 D0 03
1964 08A7 50
1965 08AB E8 0A24 R
1966 08AB 58
1967 08AC 08 F8
1968 08AE E8 08D7 R
1969 08B1 9C
1970 08B2 81 E7 00FB
1971 08B6 90
1972 08B7 59
1973 08B8 73 08
1974 08BA FE C5
1975 08BC 3A E9
1976 08BE 75 05
1977
1978
1979
1980 08C0 F9
1981 08C1 C3
1982
1983 08C2 8A 0E 0045 R
1984 08C6 88 8D 0094 R
1985 08CA D0 ED
1986 08CC 3A E9
1987 08CE 74 08
1988 08D0 80 8D 0090 R 20
1989
1990 08D5 F8
1991 08D6 C3
1992 08D7
1993
1994
1995
1996
1997
1998
1999
2000
2001 08D7
2002 08D7 B8 08EB R
2003 08DA 50
2004 08DB B4 4A
2005 08DD E8 09FB R
2006 08E0 8B C7
2007 08E2 8A E0
2008 08E4 E8 09FB R
2009 08E7 E8 075B R
2010 08EA 58
2011 08EB
2012 08EB C3
2013 08EC
2014
2015
2016
2017
2018
2019
2020
2021
2022 08EC
2023 08EC B0 0E
2024 08EE EA 0000 E
2025 08F1 AB C0
2026 08F3 F9
2027 08F4 75 0E
2028
2029 08F6 B0 10
2030 08F8 E8 0000 E
2031 08FB 0B FF
2032 08FD 75 03
2033 08FF C0 C8 04
2034 0902
2035 0902 24 0F
2036 0904
2037 0904 C3
2038 0905
    
```

```

PAGE
-----
: SETUP_DBL
: CHECK DOUBLE STEP.
:
: ON ENTRY:
: DI = DRIVE
: CY = 1 MEANS ERROR
:
-----
SETUP_DBL PROC NEAR
MOV AH,DSK_STATE[DI] ; ACCESS STATE
TEST AH,0 ; ESTABLISHED STATE ?
JNZ NO_DBL ; IF ESTABLISHED THEN DOUBLE DONE
:
:----- CHECK FOR TRACK 0 TO SPEED UP ACKNOWLEDGE OF UNFORMATTED DISKETTE
MOV #SEEK_STATUS,0 ; SET RECALIBRATE REQUIRED ON ALL DRIVES
CALL MOTOR_ON ; ENSURE MOTOR STAY ON
MOV CH,0 ; LOAD TRACK 0
CALL SEEK ; SEEK TO TRACK 0
CALL READ_ID ; READ ID FUNCTION
JC SD_ERR ; IF ERROR NO TRACK 0
:
:----- INITIALIZE START AND MAX TRACKS (TIMES 2 FOR BOTH HEADS)
MOV CX,0480H ; START, MAX TRACKS
TEST #DSK_STATE[DI],TRK_CAPA ; TEST FOR 80 TRACK CAPABILITY
JZ CNT_OK ; IF NOT COUNT IS SETUP
MOV CL,0A0H ; MAXIMUM TRACK 1.2 MB
:
: ATTEMPT READ ID OF ALL TRACKS, ALL HEADS UNTIL SUCCESS; UPON SUCCESS,
: MUST SEE IF ASKED FOR TRACK IN SINGLE STEP MODE = TRACK ID READ; IF NOT
: THEN SET DOUBLE STEP ON.
CNT_OK: MOV #MOTOR_COUNT,OFFH ; ENSURE MOTOR STAYS ON FOR OPERATION
CX ; SAVE TRACK, COUNT
MOV #DSKETTE_STATUS,0 ; CLEAR STATUS, EXPECT ERRORS
XOR AX,AX ; CLEAR AX
SHR CH,1 ; HALVE TRACK, CY = HEAD
RCL AL,3 ; AX = HEAD IN CORRECT BIT
PUSH AX ; SAVE HEAD
CALL SEEK ; SEEK TO TRACK
POP AX ; RESTORE HEAD
OR DI,AX ; DI = HEAD OR'ED DRIVE
CALL READ_ID ; READ ID HEAD 0
PUSHF ; SAVE RETURN FROM READ_ID
AND DI,11111011B ; TURN OFF HEAD 1 BIT
POPF ; RESTORE ERROR RETURN
POP CX ; RESTORE COUNT
JNC DO_CHK ; IF OK; ASKED = RETURNED TRACK ?
INC CH ; INC FOR NEXT TRACK
CMP CH,CL ; REACHED MAXIMUM YET
JNZ CNT_OK ; CONTINUE TILL ALL TRIED
:
:----- FALL THRU, READ ID FAILED FOR ALL TRACKS
SD_ERR: STC ; SET CARRY FOR ERROR
RET ; SETUP_DBL ERROR EXIT
:
DO_CHK: MOV CL,#NEC_STATUS+3 ; LOAD RETURNED TRACK
MOV #DSK_TRK[DI],CL ; STORE TRACK NUMBER
SHR CH,1 ; HALVE TRACK
CMP CH,CL ; IS IT THE SAME AS ASKED FOR TRACK
JZ NO_DBL ; IF SAME THEN NO DOUBLE STEP
OR #DSK_STATE[DI],DBL_STEP ; TURN ON DOUBLE STEP REQUIRED
:
NO_DBL: CLC ; CLEAR ERROR FLAG
RET
SETUP_DBL ENDP
:
:----- READ ID FUNCTION.
:
: ON ENTRY: DI = BIT 2 = HEAD; BITS 1,0 = DRIVE
:
: ON EXIT: DI = BIT 2 IS RESET, BITS 1,0 = DRIVE
: #DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
:
-----
READ_ID PROC NEAR
MOV AX,OFFSET ER_3 ; MOVE NEC OUTPUT ERROR ADDRESS
PUSH AX
MOV AH,4AH ; READ ID COMMAND
CALL NEC_OUTPUT ; TO CONTROLLER
MOV AX,DI ; DRIVE # TO AH, HEAD 0
MOV AH,AL ; TO CONTROLLER
CALL NEC_OUTPUT ; WAIT FOR OPERATION, GET STATUS
CALL NEC_TERM ; THROW AWAY ERROR ADDRESS
POP AX
ER_3: RET
READ_ID ENDP
:
:----- CMOS_TYPE
: RETURNS DISKETTE TYPE FROM CMOS
:
: ON ENTRY: DI = DRIVE #
:
: ON EXIT: AL = TYPE; CY REFLECTS STATUS
:
-----
CMOS_TYPE PROC NEAR
MOV AL,CMOS_DIAG ; CMOS DIAGNOSTIC STATUS BYTE ADDRESS
CALL CMOS_READ ; GET CMOS STATUS
TEST AL,BAD_BAT+BAD_CKSUM ; BATTERY GOOD AND CHECKSUM VALID ?
STC ; SET CY = 1 INDICATING ERROR FOR RETURN
JNZ BAD_CM ; ERROR IF EITHER BIT ON
:
MOV AL,CMOS_DISKETTE ; ADDRESS OF DISKETTE BYTE IN CMOS
CALL CMOS_READ ; GET DISKETTE BYTE
OR DI,DI ; SEE WHICH DRIVE IN QUESTION
JNZ TB ; IF DRIVE 1, DATA IN LOW NIBBLE
ROR AL,4 ; EXCHANGE NIBBLES IF SECOND DRIVE
TB: AND AL,00FH ; KEEP ONLY DRIVE DATA, RESET CY = 0
BAD_CM: RET
CMOS_TYPE ENDP
    
```

```

2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052 0905
2053 0905 IE
2054 0906 56
2055 0907 28 C0
2056 0909 8E D8
2057 090B 87 D3
2058 090D 2A FF
2059
2060 090F C5 36 0078 R
2061 0913 8A 20
2062 0915 87 D3
2063 0917 8E
2064 0918 1F
2065 0919 C3
2066
2067 091A
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088 091A
2089 091A 53
2090 091B E8 0965 R
2091 091E 72 43
2092 0920 E8 0435 R
2093 0923 B8 90FD
2094 0926 CD 15
2095 0928 9C
2096 0929 E8 040F R
2097 092C 9D
2098 092D 73 05
2099 092F E8 0965 R
2100 0932 72 2F
2101
2102 0934
2103 0934 B2 0A
2104 0936 E8 0905 R
2105 0939 8A C4
2106 093B 32 E4
2107 093D 3C 08
2108 093F 73 02
2109 0941 B0 08
2110
2111
2112
2113 0943 50
2114 0944 BA F424
2115 0947 F7 E2
2116 0949 8B CA
2117 094B 8B 00
2118 094D F8
2119 094E D1 D2
2120 0950 D1 D1
2121 0952 B4 86
2122 0954 CD 15
2123 0956 58
2124 0957 73 0A
2125
2126
2127
2128 0959
2129 0959 B9 205E
2130 095C E8 0000 E
2131 095F FE C8
2132 0961 75 F6
2133
2134 0963
2135 0963 5B
2136 0964 C3
2137 0965
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148 0965
2149 0965 8B DF
2150 0967 8A CB
2151 0969 C0 C3 04
2152 096C FA
    
```

```

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

SECTION 5

```

2153 096D C6 06 0040 R FF      MOV     #MOTOR_COUNT,0FFH      ; ENSURE MOTOR STAYS ON FOR OPERATION
2154 0972 A0 003F R            MOV     AL,#MOTOR_STATUS      ; GET DIGITAL OUTPUT REGISTER REFLECTION
2155 0975 24 30                AND     AL,00110000B         ; KEEP ONLY DRIVE SELECT BITS
2156 0977 B4 01                MOV     AH,1                  ; MASK FOR DETERMINING MOTOR BIT
2157 0979 D2 E4                SHL     AH,CL                 ; AH = MOTOR ON, A=00000001, B=00000010
2158
2159                ; AL = DRIVE SELECT FROM #MOTOR_STATUS
2160                ; BL = DRIVE SELECT DESIRED
2161                ; AH = MOTOR ON MASK DESIRED
2162
2163 097B 3A C3                CMP     AL,BL                 ; REQUESTED DRIVE ALREADY SELECTED ?
2164 097D 75 06                JNZ     TURN_IT_ON           ; IF NOT SELECTED JUMP
2165 097F B4 26 003F R         TEST    AH,#MOTOR_STATUS     ; TEST MOTOR ON BIT
2166 0983 75 2C                JNZ     NO_MOT_WAIT         ; JUMP IF MOTOR ON AND SELECTED
2167
2168 0985                TURN_IT_ON:
2169 0985 0A E3                OR     AH,BL                 ; AH = DRIVE SELECT AND MOTOR ON
2170 0987 8A 3E 003F R         MOV     BH,#MOTOR_STATUS     ; SAVE COPY OF #MOTOR_STATUS BEFORE
2171 098B 80 E7 0F                AND     BH,00001111B         ; KEEP ONLY MOTOR BITS
2172 098E 80 26 003F R CF       AND     #MOTOR_STATUS,11001111B ; CLEAR OUT DRIVE SELECT
2173 0993 08 26 003F R         OR     #MOTOR_STATUS,AH      ; OR IN DRIVE SELECTED AND MOTOR ON
2174 0997 A0 003F R            MOV     AL,#MOTOR_STATUS     ; GET DIGITAL OUTPUT REGISTER REFLECTION
2175 099A 8A D8                MOV     BL,AL                 ; BL=#MOTOR_STATUS AFTER, BH=BEFORE
2176 099C 80 E3 0F                AND     BL,00001111B         ; KEEP ONLY MOTOR BITS
2177 099F FB                    STI
2178 09A0 24 3F                AND     AL,00111111B         ; ENABLE INTERRUPTS AGAIN
2179 09A2 C0 C0 04                ROL     AL,4                  ; STRIP AWAY UNWANTED BITS
2180 09A5 0C 04                OR     AL,00001100B         ; PUT BITS IN DESIRED POSITIONS
2181 09A7 BA 03F2              MOV     DX,03F2H             ; NO RESET, ENABLE DMA/INTERRUPT
2182 09AA EE                    OUT     DX,AL                 ; SELECT DRIVE AND TURN ON MOTOR
2183 09AB 3A DF                CMP     BL,BH                 ; NEW MOTOR TURNED ON ?
2184 09AD 74 02                JZ     NO_MOT_WAIT           ; NO WAIT REQUIRED IF JUST SELECT
2185 09AF F8                    CLC
2186 09B0 C3                RET
2187
2188 09B1                NO_MOT_WAIT:
2189 09B1 F9                    STC
2190 09B2 FB                    STI
2191 09B3 C3                RET
2192 09B4                INTERRUPTS BACK ON
2193
2194                -----
2195                ; HD_WAIT
2196                ; WAIT FOR HEAD SETTLE TIME.
2197                ;
2198                ; ON ENTRY:  DI ; DRIVE #
2199                ;
2200                ; ON EXIT:  AX,BX,CX,DX DESTROYED
2201                ;
2202                -----
2201 09B4                HD_WAIT
2202 09B4 B2 09                MOV     DL,9                  ; GET HEAD SETTLE PARAMETER
2203 09B6 E8 0905 R            CALL    GET_PARM              ; GET PARM
2204 09B9 F6 06 003F R 80       TEST    #MOTOR_STATUS,10000000B ; SEE IF A WRITE OPERATION
2205 09BE 74 14                JZ     ISNT_WRITE            ; IF NOT, DO NOT ENFORCE ANY VALUES
2206 09C0 0A E4                OR     AH,AH                  ; CHECK FOR ANY WAIT?
2207 09C2 75 14                JNZ     DO_WAIT              ; IF THERE DO NOT ENFORCE
2208 09C4 B4 0F                MOV     AH,#HD12_SETTLE      ; LOAD 1.2M HEAD SETTLE MINIMUM
2209 09C5 8A 85 0090 R         MOV     AL,#DSK_STATE[D1]    ; LOAD STATE
2210 09CA 24 C0                AND     AL,RATE_MSK          ; KEEP ONLY RATE
2211 09CC 3C 80                CMP     AL,RATE_260          ; 1.2 M DRIVE ?
2212 09CE 75 08                JNZ     DO_WAIT              ; DEFAULT HEAD SETTLE LOADED
2213
2214 09D0 B4 14                GP3:  MOV     AH,#HD320_SETTLE ; USE 320/360 HEAD SETTLE
2215 09D2 EB 04                JMP     SHORT_DO_WAIT        ;
2216
2217 09D4                ISNT_WRITE:
2218 09D4 0A E4                OR     AH,AH                  ; CHECK FOR NO WAIT
2219 09D6 74 1F                JZ     HW_DONE                ; IF NOT WRITE AND 0 ITS OK
2220
2221                ;----- AH CONTAINS NUMBER OF MILLISECONDS TO WAIT
2222
2223 09D8 8A C4                DO_WAIT: MOV    AL,AH              ; AL = # MILLISECONDS
2224 09DA 32 E4                XOR     AH,AH                 ; AX = # MILLISECONDS
2225 09DC 50                    PUSH    AX                     ; SAVE HEAD SETTLE PARAMETER
2226 09DD BA 03E8              MOV     DX,1000               ; SET UP FOR MULTIPLY TO MICROSECONDS
2227 09E0 F7 E2                MUL     DX                     ; DX,AX = # MICROSECONDS
2228 09E2 8B CA                MOV     CX,DX                  ; CX,AX = # MICROSECONDS
2229 09E4 8B D0                MOV     DX,AX                  ; CX,DX = # MICROSECONDS
2230 09E6 B4 86                MOV     AH,86H                ; LOAD WAIT CODE
2231 09E8 0C 15                INT     15H                   ; PERFORM WAIT
2232 09EA 58                    POP     AX                     ; RESTORE HEAD SETTLE PARAMETER
2233 09EB 73 0A                JNC     HW_DONE                ; CHECK FOR EVENT WAIT ACTIVE
2234
2235 09ED                JZ9:
2236 09ED B9 0042              MOV     CX,66                  ; 1 MILLISECOND LOOP
2237 09F0 E8 0000 E            CALL    WAITF                  ; COUNT AT 15.085737 US PER COUNT
2238 09F3 FE C8                DEC     AL                     ; DELAY FOR 1 MILLISECOND
2239 09F5 75 F6                JNZ     JZ9                    ; DECREMENT THE COUNT
2240 09F7                ; DO AL MILLISECOND # OF TIMES
2241 09F7 C3                HW_DONE: RET
2242 09F8                HD_WAIT ENDP
2243
2244                -----
2245                ; NEC_OUTPUT
2246                ; THIS ROUTINE SENDS A BYTE TO THE NEC CONTROLLER AFTER :
2247                ; TESTING FOR CORRECT DIRECTION AND CONTROLLER READY THIS :
2248                ; ROUTINE WILL TIME OUT IF THE BYTE IS NOT ACCEPTED WITHIN :
2249                ; A REASONABLE AMOUNT OF TIME, SETTING THE DISKETTE STATUS :
2250                ; ON COMPLETION.
2251                ;
2252                ; ON ENTRY:  AH = BYTE TO BE OUTPUT
2253                ;
2254                ; ON EXIT:
2255                ; CY = 0 SUCCESS
2256                ; CY = 1 FAILURE -- DISKETTE STATUS UPDATED
2257                ; IF A FAILURE HAS OCCURRED, THE RETURN IS MADE :
2258                ; ONE LEVEL HIGHER THAN THE CALLER OF NEC_OUTPUT. :
2259                ; THIS REMOVES THE REQUIREMENT OF TESTING AFTER :
2260                ; EVERY CALL OF NEC_OUTPUT.
2261                ;
2262                ; AX,CX,DX DESTROYED
2263                -----
2262 09F8                NEC_OUTPUT
2263 09F8 53                    PROC    NEAR
2264 09F9 BA 03F4              PUSH    BX                     ; SAVE REG.
2265 09FC B3 02                MOV     BL,2                   ; STATUS PORT
2266 09FE 33 C9                XOR     CX,CX                   ; HIGH ORDER COUNTER
2267                                ; COUNT FOR TIME OUT

```

```

2267
2268 0A00 EC          J23:  IN    AL,DX          ; GET STATUS
2269 0A01 24 C0      AND    AL,11000000B    ; KEEP STATUS AND DIRECTION
2270 0A03 3C 80      CMP    AL,10000000B    ; STATUS 1 AND DIRECTION 0 ?
2271 0A05 74 0F      JZ     J27             ; STATUS AND DIRECTION OK
2272 0A07 E2 F7      LOOP  J23             ; CONTINUE TILL CX EXHAUSTED
2273
2274 0A09 FE CB      DEC    BL             ; DECREMENT COUNTER
2275 0A0B 75 F3      JNZ   J23             ; REPEAT TILL DELAY FINISHED, CX = 0
2276
2277
2278
2279 0A0D 80 0E 0041 R 80  ;----- FALL THRU TO ERROR RETURN
2280 0A12 5B          OR     #DSKETTE_STATUS,TIME_OUT
2281 0A13 5E          POP    BX             ; RESTORE REG.
2282 0A14 F9          POP    AX             ; DISCARD THE RETURN ADDRESS
2283 0A15 C3          STC                    ; INDICATE ERROR TO CALLER
2284
2285
2286
2287 0A16          J27:  MOV    AL,AH          ; GET BYTE TO OUTPUT
2288 0A16 8A C4      INC    DX             ; DATA PORT = STATUS PORT + 1
2289 0A18 42          OUT    DX,AL         ; OUTPUT THE BYTE
2290 0A19 EE
2291
2292 0A1A 9C          PUSHF                 ; SAVE FLAGS
2293 0A1B B9 0003     MOV    CX,3           ; 30 TO 45 MICROSECOND WAIT FOR
2294 0A1E E8 0000 E   CALL  WAITF           ; NEC FLAGS UPDATE CYCLE
2295 0A21 9D          POPF                  ; RESTORE FLAGS FOR EXIT
2296 0A22 5B          POP    BX             ; RESTORE REG.
2297 0A23 C3          RET                   ; CY = 0 FROM TEST INSTRUCTION
2298 0A24
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312 0A24          ;----- DIRECTION AND STATUS OK; OUTPUT BYTE
2313 0A24 24 DF      PROC    NEAR
2314 0A26 B0 01      MOV    BX,DI         ; BX = DRIVE #
2315 0A28 56 CB      MOV    AL,1          ; ESTABLISH MASK FOR RECALIBRATE TEST
2316 0A2A D6 C0      XCHG  CL,BL         ; GET DRIVE VALUE INTO CL
2317 0A2C 86 C9      ROL   AL,CL         ; SHIFT MASK BY THE DRIVE VALUE
2318 0A2E 84 06 003E R XCHG  CL,BL         ; RECOVER TRACK VALUE
2319 0A32 75 1C      TEST  AL,#SEEK_STATUS ; TEST FOR RECALIBRATE REQUIRED
2320
2321 0A34 08 06 003E R J28A   J28A                ; JUMP IF RECALIBRATE NOT REQUIRED
2322 0A38 E8 0A83 R   OR     #SEEK_STATUS,AL ; TURN ON THE NO RECALIBRATE BIT IN FLAG
2323 0A3B 73 0A      CALL  RECAL          ; RECALIBRATE DRIVE
2324
2325
2326
2327 0A3D C6 06 0041 R 80 ;----- ISSUE RECALIBRATE FOR 80 TRACK DISKETTES
2328 0A42 E8 0A83 R   MOV    #DSKETTE_STATUS,0 ; CLEAR OUT INVALID STATUS
2329 0A45 72 3B      CALL  RECAL          ; RECALIBRATE DRIVE
2330
2331 0A47          ; IF RECALIBRATE FAILS TWICE THEN ERROR
2332 0A47 C6 85 0094 R 00 ;----- AFT_RECAL:
2333 0A4C 0A ED      MOV    #DSK_TRK[D1],0 ; SAVE NEW CYLINDER AS PRESENT POSITION
2334 0A4E 74 2D      OR     CH,CH         ; CHECK FOR SEEK TO TRACK 0
2335
2336
2337
2338 0A50 F6 85 0090 R 20 ;----- DRIVE IS IN SYNCHRONIZATION WITH CONTROLLER, SEEK TO TRACK
2339 0A55 74 02      J28A: TEST #DSK_STATE[D1],DBL_STEP ; CHECK FOR DOUBLE STEP REQUIRED
2340 0A57 D0 E5      RT     CH,1          ; SINGLE STEP REQUIRED BYPASS DOUBLE
2341
2342 0A59 3A AD 0094 R ;----- DRIVE IS IN SYNCHRONIZATION WITH CONTROLLER, SEEK TO TRACK
2343 0A5D 74 23      RT:   CMP #DSK_TRK[D1] ; SEE IF ALREADY AT THE DESIRED TRACK
2344
2345 0A5F BA 0A82 R   JE     RB           ; IF YES, DO NOT NEED TO SEEK
2346 0A62 52          MOV    DX,OFFSET NEC_ERR ; LOAD RETURN ADDRESS
2347 0A63 B8 AD 0094 R ;----- AFT_RECAL:
2348 0A67 94 0F      MOV    #DSK_TRK[D1],CH ; ON STACK FOR NEC_OUTPUT ERROR
2349 0A69 E8 09F8 R   MOV    AH,0FH       ; SAVE NEW CYLINDER AS PRESENT POSITION
2350 0A6C 8B DF      CALL  NEC_OUTPUT    ; SEEK COMMAND TO NEC
2351 0A6E 8A E3      MOV    BX,DI         ; BX = DRIVE #
2352 0A70 E8 09F8 R   MOV    AH,BL        ; OUTPUT DRIVE NUMBER
2353 0A73 8A A5 0094 R ;----- DRIVE IS IN SYNCHRONIZATION WITH CONTROLLER, SEEK TO TRACK
2354 0A77 E8 09F8 R   CALL  NEC_OUTPUT    ; GET CYLINDER NUMBER
2355 0A7A E8 0A9A R   CALL  NEC_OUTPUT    ; ENDING INTERRUPT AND SENSE STATUS
2356
2357
2358
2359 0A7D          ;----- WAIT FOR HEAD SETTLE
2360 0A7D 9C          DO_WAIT: PUSHF                ; SAVE STATUS
2361 0A7E E8 09B4 R   CALL  HD_WAIT       ; WAIT FOR HEAD SETTLE TIME
2362 0A81 9D          OR     POPF           ; RESTORE STATUS
2363 0A82
2364 0A82          RB:  NEC_ERR:
2365 0A82 C3          RET                   ; RETURN TO CALLER
2366 0A83
2367
2368
2369
2370
2371
2372
2373
2374
2375 0A83          ;----- RECALIBRATE DRIVE
2376 0A83 51          RECAL: PROC    NEAR
2377 0A84 B8 0A9B R   MOV    CX,AX         ; LOAD NEC_OUTPUT ERROR
2378 0A87 50          MOV    AX,OFFSET RC_BACK ; RECALIBRATE COMMAND
2379 0A88 B4 07      MOV    AH,07H
2380 0A8A E8 09F8 R   CALL  NEC_OUTPUT

```

SECTION 5

```

2381 0ABD 08 DF          MOV     BX,DI          ; BX = DRIVE #
2382 0ABF 8A E3          MOV     AH,BL
2383 0A91 E8 09F8 R     CALL    NEC_OUTPUT    ; OUTPUT THE DRIVE NUMBER
2384 0A94 E8 0A9A R     CALL    CHK_STAT_2    ; GET THE INTERRUPT AND SENSE INT STATUS
2385 0A97 58             POP     AX             ; THROW AWAY ERROR
2386 0A98
2387 0A98 59           RC_BACK: POP     CX
2388 0A99 C3           RET
2389 0A9A
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399 0A9A
2400 0A9A 88 0AB8 R     MOV     AX,OFFSET CS_BACK ; LOAD NEC_OUTPUT ERROR ADDRESS
2401 0A9D 58             PUSH    AX
2402 0A9E E8 0AC1 R     CALL    WAIT_INT      ; WAIT FOR THE INTERRUPT
2403 0AA1 72 14         JC      J34            ; IF ERROR, RETURN IT
2404 0AA3 B4 08         MOV     AH,08H        ; SENSE INTERRUPT STATUS COMMAND
2405 0AA5 E8 09F8 R     CALL    NEC_OUTPUT    ; READ IN THE RESULTS
2406 0AA8 E8 0A9A R     CALL    RESULTS
2407 0AAB 72 0A         JC      J34
2408 0AAD AD 0042 R     MOV     AL,NEC_STATUS  ; GET THE FIRST STATUS BYTE
2409 0AB0 24 60         AND     AL,01100000B   ; ISOLATE THE BITS
2410 0AB2 3C 60         CMP     AL,01100000B   ; TEST FOR CORRECT VALUE
2411 0AB4 74 03         JZ      J36            ; IF ERROR, GO MARK IT
2412 0AB6 F8           CLC
2413 0AB7
2414 0AB7 58           J34:  POP     AX        ; THROW AWAY ERROR RETURN
2415 0AB8
2416 0AB8 C3           CS_BACK: RET
2417
2418 0AB9
2419 0AB9 80 0E 0041 R 40 J35:  OR      #DSKETTE_STATUS,BAD_SEEK ; ERROR RETURN CODE
2420 0ABE F9           STC
2421 0ABF EB F6         JMP     SHORT J34
2422 0AC1
2423
2424
2425
2426
2427
2428
2429
2430
2431 0AC1
2432 0AC1 FB           WAIT_INT: PROC NEAR
2433 0AC2 F8           STI
2434 0AC3 88 9001      CLC
2435 0AC6 C0 15         MOV     AX,09001H     ; TURN ON INTERRUPTS, JUST IN CASE
2436 0AC8 72 11         INT     J36A          ; CLEAR TIMEOUT INDICATOR
2437 0ACA B8 0A         JC      J36A          ; LOAD WAIT CODE AND TYPE
2438 0ACD 93 C9         MOV     BL,10         ; PERFORM OTHER FUNCTION
2439 0ACE             XOR     CX,CX          ; BYPASS TIMING LOOP IF TIMEOUT DONE
2440 0ACE F6 06 003E R 80 J36:  TEST   #SEEK_STATUS,INT_FLAG ; TEST FOR INTERRUPT OCCURRING
2441 0AD3 75 0C         JNB    J37            ; COUNT DOWN WHILE WAITING
2442 0AD5 E2 F7         LOOP   J36           ; SECOND LEVEL COUNTER
2443 0AD7 FE 0B         DEC     BL
2444 0AD9 75 F3         JNZ    J36
2445
2446 0ADB 80 0E 0041 R 80 J36A: OR      #DSKETTE_STATUS,TIME_OUT ; NOTHING HAPPENED
2447 0AE0 F9           STC
2448 0AE1
2449 0AE1 9C           J37:  AND     #SEEK_STATUS,NOT_INT_FLAG ; SAVE CURRENT CARRY
2450 0AE2 80 26 003E R 7F AND     #SEEK_STATUS,NOT_INT_FLAG ; TURN OFF INTERRUPT FLAG
2451 0AE7 9D           POPF
2452 0AE8 C3           RET
2453 0AE9
2454
2455
2456
2457
2458
2459
2460
2461
2462 0AE9
2463 0AE9 87           RESULTS: PROC NEAR
2464 0AEA BF 0042 R     PUSH   DI             ; POINTER TO DATA AREA
2465 0AED B3 07         MOV     BL,7          ; MAX STATUS BYTES
2466 0AEF BA 03F4      MOV     DX,03F4H     ; STATUS PORT
2467
2468
2469
2470 0AF2 87 02         R10:  MOV     BH,2         ; HIGH ORDER COUNTER
2471 0AF4 33 C9         XOR     CX,CX         ; COUNTER
2472 0AF6
2473 0AF6 EC           J39:  IN      AL,DX        ; GET STATUS
2474 0AF7 24 C0         AND     AL,11000000B ; KEEP ONLY STATUS AND DIRECTION
2475 0AF9 3C C0         CMP     AL,11000000B ; STATUS 1 AND DIRECTION 1 ?
2476 0AFB 74 0E         JZ      J42           ; STATUS AND DIRECTION OK
2477 0AFD E2 F7         LOOP   J39           ; LOOP TILL TIMEOUT
2478
2479 0AFF FE CF         DEC     BH            ; DECREMENT HIGH ORDER COUNTER
2480 0B01 75 F3         JNZ    J39           ; REPEAT TILL DELAY DONE
2481
2482 0B03 80 0E 0041 R 80 OR      #DSKETTE_STATUS,TIME_OUT
2483 0B08 F9           STC
2484 0B09 EB 1B         JMP     SHORT POPRES  ; SET ERROR RETURN
2485
2486
2487
2488 0B0B
2489
2490 0B0B 42           J42:  JMP     $+2          ; I/O DELAY
2491 0B0C EC           INC     DX            ; POINT AT DATA PORT
2492 0B0D 88 05         IN      AL,DX        ; GET THE DATA
2493 0B0F 47           MOV     [DI],AL      ; STORE THE BYTE
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2999
3000

```

```

2495 0B10 B9 0003      MOV     CX,3           ; MINIMUM 24 MICROSECONDS FOR NEC
2496 0B13 E8 0000 E    CALL   WAITF         ; WAIT 30 TO 45 MICROSECONDS
2497 0B16 4A          DEC     DX             ; POINT AT STATUS PORT
2498 0B17 EC          IN      AL,DX         ; GET STATUS
2499 0B18 A9 10       TEST    AL,00010000B ; TEST FOR NEC STILL BUSY
2500 0B1A 74 0A       JZ      POPRES        ; RESULTS DONE ?
2501
2502 0B1C FE CB       DEC     BL             ; DECREMENT THE STATUS COUNTER
2503 0B1E 75 D2       JNZ    R10            ; GO BACK FOR MORE
2504 0B20 80 0E 0041 R 20 OR      #DSKETTE_STATUS,BAD_NEC ; TOO MANY STATUS BYTES
2505 0B25 F9          STC                    ; SET ERROR FLAG
2506
2507                      ;----- RESULT OPERATION IS DONE
2508
2509 0B26              POPRES: POP     DI             ; RETURN WITH CARRY SET
2510 0B26 5F          RET
2511 0B27 C3          RET
2512 0B28
2513
2514                      ;----- READ_DSCKCHNG
2515                      ; READS THE STATE OF THE DISK CHANGE LINE.
2516                      ; ON ENTRY: DI = DRIVE #
2517                      ; ON EXIT: DI = DRIVE #
2518                      ; ZF = 0 ; DISK CHANGE LINE INACTIVE
2519                      ; ZF = 1 ; DISK CHANGE LINE ACTIVE
2520                      ; AX,CX,DX DESTROYED
2521
2522                      ;----- READ_DSCKCHNG
2523                      ; PROC NEAR
2524 0B28          CALL   PROC_NEAR        ; TURN ON THE MOTOR IF OFF
2525 0B28 E8 091A R    MOV     DX,03F7H     ; ADDRESS DIGITAL INPUT REGISTER
2526 0B2B BA 03F7     IN      AL,DX         ; INPUT DIGITAL INPUT REGISTER
2527 0B2E EC          MOV     CH,TRK_SLAP ; CHECK FOR DISK CHANGE LINE ACTIVE
2528 0B2F A8 80       TEST    AL,DSK_CHG   ; RETURN TO CALLER WITH ZERO FLAG SET
2529 0B31 C3          RET
2530 0B32
2531                      ;----- DRIVE_DET
2532                      ; DETERMINES WHETHER DRIVE IS 80 OR 40 TRACKS AND
2533                      ; UPDATES STATE INFORMATION ACCORDINGLY.
2534                      ; ON ENTRY: DI = DRIVE #
2535
2536                      ;----- DRIVE_DET
2537                      ; PROC NEAR
2538 0B32          CALL   PROC_NEAR        ; TURN ON MOTOR IF NOT ALREADY ON
2539 0B32 E8 0A83 R    CALL   RECAL         ; RECALIBRATE DRIVE
2540 0B35 E8 0A83 R    JC      DO_BAC        ; ASSUME NO DRIVE PRESENT
2541 0B38 72 3C       JC      CH,TRK_SLAP   ; SEEK TO TRACK 48
2542 0B3A B5 30       MOV     CH,TRK_SLAP
2543 0B3C E8 0A24 R    CALL   SEEK          ; ERROR NO DRIVE
2544 0B3F 72 36       JC      DO_BAC        ; SEEK TO TRACK 10
2545 0B41 B5 08       MOV     CH,QUIET_SEEK+1
2546 0B43
2547 0B43 FE CD       DEC     CH             ; DECREMENT TO NEXT TRACK
2548 0B45 51          PUSH    CX             ; SAVE TRACK
2549 0B46 E8 0A24 R    CALL   SEEK          ; POP AND RETURN
2550 0B49 72 2C       JC      POP_BAC       ; LOAD NEC OUTPUT ERROR ADDRESS
2551 0B4B 88 0B77 R    MOV     AX,OFFSET POP_BAC
2552 0B4E 50          PUSH    AX             ; SENSE DRIVE STATUS COMMAND BYTE
2553 0B4F B4 04       MOV     AH,SENSE_DRV_ST ; OUTPUT TO NEC
2554 0B51 E8 09F8 R    CALL   NEC_OUTPUT    ; AL = DRIVE
2555 0B54 8B C7       MOV     AL,DI          ; AH = DRIVE
2556 0B56 B4 E0       MOV     AH,AL          ; OUTPUT TO NEC
2557 0B58 E8 09F8 R    CALL   NEC_OUTPUT    ; GO GET STATUS
2558 0B5B E8 0AE9 R    CALL   RESULTS       ; THROW AWAY ERROR ADDRESS
2559 0B5E 58          POP     AX             ; RESTORE TRACK
2560 0B5F 59          POP     CX             ; TRACK 0 ?
2561 0B60 F4 06 0042 R 10 TEST    #NEC_STATUS,HOME ; GO TILL TRACK 0
2562 0B65 74 DC       JZ      SK_CTN        ; IS HOME AT TRACK 0 ?
2563 0B67 0A ED       OR      CH,CH         ; MUST BE 80 TRACK DRIVE
2564 0B69 74 06       JZ      IS_80
2565
2566                      ; DRIVE IS A 360; SET DRIVE TO DETERMINED;
2567                      ; SET MEDIA TO DETERMINED AT RATE 250.
2568
2569 0B6B 80 80 0090 R 94 OR      #DSK_STATE[DI],DRV_DET+MED_DET+RATE_250
2570 0B70 C3          RET                    ; ALL INFORMATION SET
2571
2572 0B71                      IS_80: OR      #DSK_STATE[DI],TRK_CAPA ; SETUP 80 TRACK CAPABILITY
2573 0B71 80 80 0090 R 01
2574 0B76          DO_BAC: RET
2575 0B76 C3          RET
2576
2577 0B77          POP_BAC: POP     CX             ; THROW AWAY
2578 0B77 59          RET
2579 0B78 C3          RET
2580
2581 0B79          DRIVE_DET  ENDP
2582
2583                      ;----- DISK_INT
2584                      ; THIS ROUTINE HANDLES THE DISKETTE INTERRUPT.
2585                      ; ON EXIT: THE INTERRUPT FLAG IS SET IN #SEEK_STATUS.
2586
2587                      ;----- DISK_INT
2588 0B79          PROC   FAR
2589 0B79 60          PUSH    AX             ; ENTRY POINT FOR ORG 0EF57H
2590 0B7A 1E          PUSH    DS             ; SAVE WORK REGISTER
2591 0B7B E8 0000 E    CALL   DDS            ; SAVE REGISTERS
2592 0B7E 80 0E 002E R 80 OR      #SEEK_STATUS,INT_FLAG ; SETUP DATA ADDRESSING
2593 0B83 1F          POP     DS             ; TURN ON INTERRUPT OCCURRED
2594 0B84 80 20       MOV     AL,#01        ; RESTORE USER (DS)
2595 0B86 E6 20       OUT    INTA00,AL      ; END OF INTERRUPT MARKER
2596 0B88 FB       STI                    ; INTERRUPT CONTROL PORT
2597 0B89 B8 9101     MOV     AX,09101H     ; RE-ENABLE INTERRUPTS
2598 0B8C CD 15       INT    15H            ; INTERRUPT POST CODE AND TYPE
2599 0B8E 58          POP     AX             ; GO PERFORM OTHER TASK
2600 0B8F CF       IRET                   ; RECOVER REGISTER
2601 0B90          DISK_INT_1  ENDP
    
```

SECTION 5

```

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

```

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

```

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

```

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

```

: REGISTERS USED FOR FIXED DISK OPERATIONS
:
: (DL) - DRIVE NUMBER (80H-81H FOR DISK, VALUE CHECKED)
: (DH) - HEAD NUMBER (0-15 ALLOWED, NOT VALUE CHECKED)
: (CH) - CYLINDER NUMBER (0-1023, NOT VALUE CHECKED) (SEE CL)
: (CL) - SECTOR NUMBER (1-17, NOT VALUE CHECKED)
:
: NOTE: HIGH 2 BITS OF CYLINDER NUMBER ARE PLACED
: IN THE HIGH 2 BITS OF THE CL REGISTER
: (10 BITS TOTAL)
:
: (AL) - NUMBER OF SECTORS (MAXIMUM POSSIBLE RANGE 1-80H,
: FOR READ/WRITE LONG 1-79H)
:
: (ES:BX) - ADDRESS OF BUFFER FOR READS AND WRITES,
: (NOT REQUIRED FOR VERIFY)
:
: FORMAT (AH=8) ES:BX POINTS TO A 512 BYTE BUFFER. THE FIRST
: 2*(SECTORS/TRACK) BYTES CONTAIN F,N FOR EACH SECTOR.
: F = 00H FOR A GOOD SECTOR
: 80H FOR A BAD SECTOR
: N = SECTOR NUMBER
: FOR AN INTERLEAVE OF 2 AND 17 SECTORS/TRACK
: THE TABLE SHOULD BE:
:
: DB 00H,01H,00H,0AH,00H,02H,00H,0BH,00H,03H,00H,0CH
: DB 00H,04H,00H,0DH,00H,05H,00H,0EH,00H,06H,00H,0FH
: DB 00H,07H,00H,10H,00H,08H,00H,11H,00H,09H
:
:-----
    
```

SECTION 5

99
 100
 101
 102
 103
 104
 105
 106
 107
 108
 109
 110
 111
 112
 113
 114
 115
 116
 117
 118
 119
 120
 121
 122
 123
 124
 125
 126
 127
 128
 129
 130
 131
 132
 133
 134
 135
 136
 137
 138
 139
 140
 141
 142 = 00FF
 143 = 00E0
 144 = 00CC
 145 = 00BB
 146 = 00AA
 147 = 0000
 148 = 0040
 149 = 0020
 150 = 0011
 151 = 0010
 152 = 000B
 153 = 000A
 154 = 0009
 155 = 0007
 156 = 0005
 157 = 0004
 158 = 0002
 159 = 0001
 160
 161
 162
 163
 164
 165
 166
 167
 168
 169
 170
 171
 172
 173
 174
 175
 176
 177
 178
 179
 180
 181
 182
 183
 184
 185
 186
 187

```

PAGE
-----
OUTPUT
AH = STATUS OF CURRENT OPERATION
STATUS BITS ARE DEFINED IN THE EQUATES BELOW
CY = 0 SUCCESSFUL OPERATION (AH=0 ON RETURN)
CY = 1 FAILED OPERATION (AH HAS ERROR REASON)

NOTE: ERROR 11H INDICATES THAT THE DATA READ HAD A RECOVERABLE
      ERROR WHICH WAS CORRECTED BY THE ECC ALGORITHM. THE DATA
      IS PROBABLY GOOD, HOWEVER THE BIOS ROUTINE INDICATES AN
      ERROR TO ALLOW THE CONTROLLING PROGRAM A CHANCE TO DECIDE
      FOR ITSELF. THE ERROR MAY NOT RECUR IF THE DATA IS
      REWRITTEN.

IF DRIVE PARAMETERS WERE REQUESTED (DL >= 80H),
INPUT:
(DL) = DRIVE NUMBER
OUTPUT:
(DL) = NUMBER OF CONSECUTIVE ACKNOWLEDGING DRIVES ATTACHED (1-2)
      (CONTROLLER CARD ZERO TALLY ONLY)
(DH) = MAXIMUM USEABLE VALUE FOR HEAD NUMBER
(CH) = MAXIMUM USEABLE VALUE FOR CYLINDER NUMBER
(CL) = MAXIMUM USEABLE VALUE FOR SECTOR NUMBER
      AND CYLINDER NUMBER HIGH BITS

IF READ DASH TYPE WAS REQUESTED,
AH = 0 - NOT PRESENT
      1 - DISKETTE - NO CHANGE LINE AVAILABLE
      2 - DISKETTE - CHANGE LINE AVAILABLE
      3 - FIXED DISK
CX,DX = NUMBER OF 512 BYTE BLOCKS WHEN AH = 3

REGISTERS WILL BE PRESERVED EXCEPT WHEN THEY ARE USED TO RETURN
INFORMATION.

NOTE: IF AN ERROR IS REPORTED BY THE DISK CODE, THE APPROPRIATE
      ACTION IS TO RESET THE DISK, THEN RETRY THE OPERATION.
    
```

```

SENSE_FAIL EQU 0FFH ; NOT IMPLEMENTED
NO_ERR EQU 0E0H ; STATUS ERROR/ERROR REGISTER=0
WRITE_FAULT EQU 0CCH ; WRITE FAULT ON SELECTED DRIVE
UNDEF_ERR EQU 0BBH ; UNDEFINED ERROR OCCURRED
NOT_RDY EQU 0AAH ; DRIVE NOT READY
TIME_OUT EQU 80H ; ATTACHMENT FAILED TO RESPOND
BAD_SEEK EQU 40H ; SEEK OPERATION FAILED
BAD_CNTLR EQU 20H ; CONTROLLER HAS FAILED
DATA_CORRECTED EQU 11H ; ECC CORRECTED DATA ERROR
BAD_ECC EQU 10H ; BAD ECC ON DISK READ
BAD_TRACK EQU 0BH ; NOT IMPLEMENTED
BAD_SECTOR EQU 0AH ; BAD SECTOR FLAG DETECTED
DMA_BOUNDARY EQU 09H ; DATA EXTENDS TOO FAR
INIT_FAIL EQU 07H ; DRIVE PARAMETER ACTIVITY FAILED
BAD_RESET EQU 05H ; RESET FAILED
RECORD_NOT_FND EQU 04H ; REQUESTED SECTOR NOT FOUND
BAD_ADDR_MARK EQU 02H ; ADDRESS MARK NOT FOUND
BAD_CMD EQU 01H ; BAD COMMAND PASSED TO DISK I/O
    
```

```

-----
FIXED DISK PARAMETER TABLE
- THE TABLE IS COMPOSED OF A BLOCK DEFINED AS:
+0 (1 WORD) - MAXIMUM NUMBER OF CYLINDERS
+2 (1 BYTE) - MAXIMUM NUMBER OF HEADS
+3 (1 WORD) - NOT USED/SEE PC-XT
+5 (1 WORD) - STARTING WRITE PRECOMPENSATION CYL
+7 (1 BYTE) - MAXIMUM ECC DATA BURST LENGTH
+8 (1 BYTE) - CONTROL BYTE
      BIT 7 DISABLE RETRIES -OR-
      BIT 6 DISABLE RETRIES
+9 (3 BYTES) - NOT USED/SEE PC-XT
      BIT 3 MORE THAN 8 HEADS
+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 | > WHEN READ FROM:
196 | HF_PORT+0 - READ DATA (FROM CONTROLLER TO CPU)
197 | HF_PORT+1 - GET ERROR REGISTER
198 | HF_PORT+2 - GET SECTOR COUNT
199 | HF_PORT+3 - GET SECTOR NUMBER
200 | HF_PORT+4 - GET CYLINDER LOW
201 | HF_PORT+5 - GET CYLINDER HIGH (2 BITS)
202 | HF_PORT+6 - GET SIZE/DRIVE/HEAD
203 | HF_PORT+7 - GET STATUS REGISTER
204 |
205 | > WHEN WRITTEN TO:
206 | HF_PORT+0 - WRITE DATA (FROM CPU TO CONTROLLER)
207 | HF_PORT+1 - SET PRECOMPENSATION CYLINDER
208 | HF_PORT+2 - SET SECTOR COUNT
209 | HF_PORT+3 - SET SECTOR NUMBER
210 | HF_PORT+4 - SET CYLINDER LOW
211 | HF_PORT+5 - SET CYLINDER HIGH (2 BITS)
212 | HF_PORT+6 - SET SIZE/DRIVE/HEAD
213 | HF_PORT+7 - SET COMMAND REGISTER
214 |
215 |-----|
216 |
217 = 01F0 HF_PORT EQU 01F0H ; DISK PORT
218 = 03F6 HF_REG_PORT EQU 03F6H
219 |
220 |-----|
221 | STATUS REGISTER
222 | ST_ERROR EQU 0000001B ;
223 | ST_INDEX EQU 0000010B ;
224 | ST_CORRCTD EQU 0000100B ; ECC CORRECTION SUCCESSFUL
225 | ST_DRQ EQU 0001000B ;
226 | ST_SEEK_COMPL EQU 0010000B ; SEEK COMPLETE
227 | ST_WRT_FLT EQU 0010000B ; WRITE FAULT
228 | ST_READY EQU 0100000B ;
229 | ST_BUSY EQU 1000000B ;
230 |
231 |-----|
232 | ERROR REGISTER
233 | ERR_DAM EQU 0000001B ; DATA ADDRESS MARK NOT FOUND
234 | ERR_TRK_0 EQU 0000010B ; TRACK 0 NOT FOUND ON RECAL
235 | ERR_ABORT EQU 0000100B ; ABORTED COMMAND
236 | ERR_ID EQU 0001000B ; NOT USED
237 | HF_PORT+0 EQU 0001000B ; ID NOT FOUND
238 | ERR_DATA_ECC EQU 0010000B ; NOT USED
239 | ERR_BAD_BLOCK EQU 0100000B ;
240 |
241 |
242 |
243 = 0010 RECAL_CMD EQU 0001000B ; DRIVE RECAL (10H)
244 = 0020 READ_CMD EQU 0010000B ; READ (20H)
245 = 0030 WRITE_CMD EQU 0011000B ; WRITE (30H)
246 = 0040 VERIFY_CMD EQU 0100000B ; VERIFY (40H)
247 = 0050 FMTTRK_CMD EQU 0101000B ; FORMAT TRACK (50H)
248 = 0060 INIT_CMD EQU 0110000B ; INITIALIZE (60H)
249 = 0070 SEEK_CMD EQU 0111000B ; SEEK (70H)
250 = 0090 DIAG_CMD EQU 1001000B ; DIAGNOSTIC (90H)
251 = 0091 SET_PARM_CMD EQU 1001001B ; DRIVE PARAMS (91H)
252 = 0001 NO_RETRIES EQU 0000001B ; CMD MODIFIER (01H)
253 = 0002 ECC_MODE EQU 0000010B ; CMD MODIFIER (02H)
254 = 0008 BUFFER_MODE EQU 0000100B ; CMD MODIFIER (08H)
255 |
256 = 0002 MAX_FILE EQU 2
257 = 0002 S_MAX_FILE EQU 2
258 |
259 = 0025 DELAY_1 EQU 25H ; DELAY FOR OPERATION COMPLETE
260 = 0600 DELAY_2 EQU 0600H ; DELAY FOR READY
261 = 0100 DELAY_3 EQU 0100H ; DELAY FOR DATA REQUEST
262 |
263 = 0008 HF_FAIL EQU 08H ; CMOS FLAG IN BYTE 0EH
264 |
265 |-----|
266 | COMMAND BLOCK REFERENCE
267 = *CMD_BLOCK EQU BYTE PTR [BP]-8 ; *CMD_BLOCK REFERENCES BLOCK HEAD IN SS
268 | ; [BP] POINTS TO COMMAND BLOCK TAIL
269 | ; AS DEFINED BY THE "ENTER" PARAMS
    
```

SECTION 5

```

270 PAGE
271 ;-----
272 ; FIXED DISK I/O SETUP
273 ;
274 ; - ESTABLISH TRANSFER VECTORS FOR THE FIXED DISK
275 ; - PERFORM POWER ON DIAGNOSTICS
276 ; SHOULD AN ERROR OCCUR A "1701" MESSAGE IS DISPLAYED
277 ;
278 ;-----
279 ASSUME CS:CODE,DS:ABS0 ; WORK OFF DS REGISTER
280
281 DISK_SETUP PROC NEAR
282 0000 FA CL;
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 0100 R MOV WORD PTR #DISK_VECTOR,AX ; INTO INT 40H
287 000C A1 004E R MOV AX,WORD PTR #ORG VECTOR+2
288 000F A3 0102 R MOV WORD PTR #DISK_VECTOR+2,AX
289 0012 C7 06 004C R 01A9 R MOV WORD PTR #ORG_VECTOR,OFFSET DISK_ID ; FIXED DISK HANDLER
290 0018 8C 0E 004E R MOV WORD PTR #ORG_VECTOR+2,CS
291 001C C7 06 01D8 R 06DA 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 8C 0E 0106 R MOV WORD PTR #HF_TBL_VEC+2,CS
295 0030 C7 06 0118 R 0000 E MOV WORD PTR #HF1_TBL_VEC,OFFSET FD_TBL ; PARM TABLE DRIVE 81
296 0036 8C 0E 011A R MOV WORD PTR #HF1_TBL_VEC+2,CS
297 003A E4 A1 IN AL,INTB01 ; TURN ON SECOND INTERRUPT CHIP
298 003C 24 BF AND AL,0BFH
299 003E EB 00 JMP $+2
300 0040 E6 A1 OUT INTB01,AL
301 0042 E4 21 IN AL,INTA01 ; LET INTERRUPTS PASS THRU TO
302 0044 24 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 IE PUSH DS
309 004C 07 POP ES ; MOVE ABS0 POINTER TO
310 004D E8 0000 E CALL DDS ; EXTRA SEGMENT POINTER
311 0050 C6 06 0074 R 00 MOV #DISK_STATUS1,0 ; ESTABLISH DATA SEGMENT
312 0055 C6 06 0075 R 00 MOV #HF_NUM,0 ; RESET THE STATUS INDICATOR
313 005A C6 06 0076 R 00 MOV #CONTROL_BYTE,0 ; ZERO NUMBER OF FIXED DISKS
314 005F B0 8E MOV AL,CMOS_DIAG+NM1
315 0061 E8 0000 E CALL CMOS_READ ; CHECK CMOS VALIDITY
316 0064 8A E0 MOV AH,AL ; SAVE CMOS FLAG
317 0066 24 C0 AND AL,BAD_BAT+BAD_CKSUM ; CHECK FOR VALID CMOS
318 0068 74 03 JZ L1
319 006A E9 00F8 R JMP POD_DONE ; CMOS NOT VALID -- NO FIXED DISKS
320 006D
321 006D 80 E4 F7 L1: AND AH,NOT HF_FAIL ; ALLOW FIXED DISK IPL
322 0070 B0 8E MOV AL,CMOS_DIAG+NM1 ; WRITE IT BACK
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 ; ZERO CARD OFFSET
327 007F 8A D8 MOV BX,AL ; SAVE FIXED DISK BYTE
328 0081 25 00F0 AND AX,000F0H ; GET FIRST DRIVE TYPE AS OFFSET
329 0084 74 72 JZ POD_DONE ; NO FIXED DISKS
330
331 0086 3C F0 CMP AL,0F0H ; CHECK FOR EXTENDED DRIVE TYPE BYTE USE
332 0088 75 10 JNE L2 ; USE DRIVE TYPE 1 --> 14 IF NOT IN USE
333
334 008A B0 99 MOV AL,CMOS_DISK_1+NM1 ; GET EXTENDED TYPE FOR DRIVE C:
335 008C E8 0000 E CALL CMOS_READ ; FROM CMOS
336 008F 3C 00 CMP AL,0 ; IS TYPE SET TO ZERO
337 0091 74 65 JE POD_DONE ; EXIT IF NOT VALID AND NO FIXED DISKS
338 0093 3C 2F CMP AL,47 ; IS TYPE WITHIN VALID RANGE
339 0095 77 61 JA POD_DONE ; EXIT WITH NO FIXED DISKS IF NOT VALID
340 0097 C1 E0 04 SHL AX,4 ; ADJUST TYPE TO HIGH NIBBLE
341 009A
342 009A 05 FFF0 E L2: ADD AX,OFFSET FD_TBL-16D ; COMPUTE OFFSET OF FIRST DRIVE TABLE
343 009D 26: A3 0104 R MOV WORD PTR #HF_TBL_VEC,AX ; SAVE IN VECTOR POINTER
344 00A1 C6 06 0075 R 01 MOV #HF_NUM,1 ; AT LEAST ONE DRIVE
345 00A6 8A C3 MOV AL,BL
346 00A8 C0 E0 04 SHL AL,4 ; GET SECOND DRIVE TYPE
347 00AB 74 2A JZ SHORT L4 ; ONLY ONE DRIVE
348 00AD B4 00 MOV AH,0
349
350 00AF 3C F0 CMP AL,0F0H ; CHECK FOR EXTENDED DRIVE TYPE BYTE USE
351 00B1 75 10 JNE L3 ; USE DRIVE TYPE 1 --> 14 IF NOT IN USE
352
353 00B3 B0 9A MOV AL,CMOS_DISK_2+NM1 ; GET EXTENDED TYPE FOR DRIVE D:
354 00B5 E8 0000 E CALL CMOS_READ ; FROM CMOS
355 00B8 3C 00 CMP AL,0 ; IS TYPE SET TO ZERO
356 00BA 74 1D JE L4 ; SKIP IF SECOND FIXED DISK NOT VALID
357 00BC 3C 2F CMP AL,47 ; IS TYPE WITHIN VALID RANGE
358 00BE 77 17 JA L4 ; SKIP IF NOT VALID
359 00C0 C1 E0 04 SHL AX,4 ; ADJUST TYPE TO HIGH NIBBLE
360 00C3
361 00C3 05 FFF0 E L3: ADD AX,OFFSET FD_TBL-16D ; COMPUTE OFFSET FOR SECOND FIXED DISK
362 00C6 8B D8 MOV BX,AX
363 00C8 2E: 83 3F 00 CMP WORD PTR CS:[BX],0 ; CHECK FOR ZERO CYLINDERS IN TABLE
364 00CC 74 09 JE L4 ; SKIP DRIVE IF NOT A VALID TABLE ENTRY
365 00CE 26: A3 0118 R MOV WORD PTR #HF1_TBL_VEC,AX ; TWO DRIVES
366 00D2 C6 06 0075 R 02 MOV #HF_NUM,2
367 00D7
368 00D7 B2 80 L4: MOV DL,80H ; CHECK THE CONTROLLER
369 00D9 B4 14 MOV AH,14H ; USE CONTROLLER DIAGNOSTIC COMMAND
370 00DB CD 13 INT 13H ; CALL BIOS WITH DIAGNOSTIC COMMAND
371 00DD 12 1A JC CTL_ERRX ; DISPLAY ERROR MESSAGE IF BAD RETURN
372 00DF A1 006C R MOV AX,#TIMER_LOW ; GET START TIMER COUNTS
373 00E2 8B D8 MOV BX,AX
374 00E4 05 0444 ADD AX,6*182 ; 60 SECONDS* 18.2
375 00E7 8B C5 MOV CX,AX
376 00E9 E8 0104 R CALL HD_RESET_1 ; SET UP DRIVE 0
377 00EC 80 3E 0075 R 01 CMP #HF_NUM,T ; WERE THERE TWO DRIVES?
378 00F1 76 05 JBE POD_DONE ; NO-ALL DONE
379 00F3 B2 81 MOV DL,81H ; SET UP DRIVE 1
380 00F5 E8 0104 R CALL HD_RESET_1
381 00F8
382 00F8 C3 POD_DONE: RET
383

```

```

384 ;----- POD ERROR
385
386 00F9          CTL_ERRX:
387 00F9 BE 0000 E   MOV     SI,OFFSET F1782      ; CONTROLLER ERROR
388 00FC E8 017C R   CALL    SET_FAIL           ; DO NOT IPL FROM DISK
389 00FF E8 0000 E   CALL    E_MSG              ; DISPLAY ERROR AND SET (BP) ERROR FLAG
390 0102 EB F4      JMP     POD_DONE
391
392
393 0104          HD_RESET:
394 0104 53        PROC    NEAR
395 0105 51        PUSH   BX
396 0106 B4 09    RES_1: MOV   AH,09H           ; SET DRIVE PARAMETERS
397 0108 CD 13    INT    13H
398 010A 72 06    JC     RES_2
399 010C B4 11    MOV   AH,11H           ; 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 FF    JNC   RES_1
404 0117 BE 0000 E RES_FL: MOV   SI,OFFSET F1781 ; INDICATE DISK 1 FAILURE
405 011A F6 C2 01 JNZ   RES_E1
406 011D 75 57    JZ    RES_E1
407 011F BE 0000 E MOV   SI,OFFSET F1780      ; INDICATE DISK 0 FAILURE
408 0122 E8 017C R CALL    SET_FAIL           ; DO NOT TRY TO IPL DISK 0
409 0125 EB 4F    JMP    SHORT_RES_E1
410 0127 B4 00    JE    RES_RS           ; RESET THE DRIVE
411 0129 CD 13    INT    13H
412 012B B4 08    RES_OK: MOV  AH,08H        ; GET MAX CYLINDER,HEAD,SECTOR
413 012D 8A 0A    MOV  DL,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 0NEC_STATUS,CX ; SAVE MAX CYLINDER, SECTOR
417 0137 8A 03    MOV  DL,BL             ; RESTORE DRIVE CODE
418 0139 BB 0401   RES_3: MOV  AX,0401H      ; VERIFY THE LAST SECTOR
419 013C CD 13    INT    13H
420 013E 73 39    JNC   RES_OK           ; VERIFY OK
421 0140 80 FC 0A CMP   AH,BAD_SECTOR    ; OK ALSO IF JUST 10 READ
422 0143 74 34    JE    RES_OK
423 0145 80 FC 11 CMP   AH,DATA_CORRECTED
424 0148 74 2F    JE    RES_OK
425 014A 80 FC 10 CMP   AH,BAD_ECC
426 014D 74 2A    JE    RES_OK
427 014F E8 018A R CALL    POD_TCHK        ; CHECK FOR TIME OUT
428 0152 72 17    JC    RES_ER           ; FAILED
429 0154 8B 0E 0042 R MOV  CX,WORD PTR 0NEC_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
433 015E 74 C7    JZ    TRY_PREVIOUS_ONE ; TRY PREVIOUS ONE
434 0160 80 E1 C0 AND  CL,0C0H          ; WE'VE TRIED ALL SECTORS ON TRACK
435 0163 0A CB    OR   CL,AL             ; KEEP CYLINDER BITS
436 0165 89 0E 0042 R MOV  WORD PTR 0NEC_STATUS,CX ; MERGE SECTOR WITH CYLINDER BITS
437 0169 EB CE    JMP   RES_3           ; SAVE CYLINDER, NEW SECTOR NUMBER
438 016B BE 0000 E RES_ER: MOV   SI,OFFSET F1791 ; TRY AGAIN
439 016E F4 C2 01 DL,1
440 0171 75 03    JNZ   RES_E1          ; INDICATE DISK 1 ERROR
441 0173 BE 0000 E MOV   SI,OFFSET F1790      ; INDICATE DISK 0 ERROR
442 0176          RES_E1:
443 0178 E8 0000 E   CALL    E_MSG              ; DISPLAY ERROR AND SET (BP) ERROR FLAG
444 0179          RES_OK:
445 0179 59        POP    CX
446 017A 5B        POP    BX
447 017B C3        RET
448 017C          HD_RESET_1:
449          ENDP
450 017C          SET_FAIL:
451 017C B8 BE8E    PROC    NEAR
452 017F E8 0000 E   MOV   AX,X*(CMDS_DIAG+NM1) ; GET CMOS ERROR BYTE
453 0182 0C 08    CALL  CMOS_READ         ; SET DO NOT IPL FROM DISK FLAG
454 0184 86 E0    OR   AL,HF_FAIL         ; SAVE IT
455 0186 E8 0000 E   CALL  CMDS_WRITE        ; PUT IT OUT
456 0189 C3        RET
457 018A          SET_FAIL:
458          ENDP
459 018A          POD_TCHK:
460 018A 58        PROC    NEAR
461 018B 5A        AX
462 018C 5B        POP    CX
463 018D 53        POP    BX
464 018E 51        PUSH  BX
465 018F 50        PUSH  CX
466 0190 A1 006C R MOV   AX,*TIMER_LOW
467          AX
468          ; CHECK FOR 30 SECOND TIME OUT
469          ; SAVE RETURN
470          ; GET TIME OUT LIMITS
471          ; AND SAVE THEM AGAIN
472          ; RESTORE RETURN
473          ; AX = CURRENT TIME
474          ; BX = START TIME
475          ; CX = END TIME
476          ; START < END
477          ; END < START < CURRENT
478          ; END, CURRENT < START
479          ; CURRENT < START < END
480          ; START < CURRENT < END
481          ; OR CURRENT < END < START
482          ; CARRY SET INDICATES TIME OUT
483          ; INDICATE STILL TIME
484          ;
485 01A9          TCHK1:
486          TCHK2:
487          TCHKNG:
488          TCHKNG:
489          TCHKNG:
490          TCHKNG:
491          TCHKNG:
492          TCHKNG:
493          TCHKNG:
494          TCHKNG:
495          TCHKNG:
496          TCHKNG:
497          TCHKNG:
498          TCHKNG:
499          TCHKNG:
500          TCHKNG:
501          TCHKNG:
502          TCHKNG:
503          TCHKNG:
504          TCHKNG:
505          TCHKNG:
506          TCHKNG:
507          TCHKNG:
508          TCHKNG:
509          TCHKNG:
510          TCHKNG:
511          TCHKNG:
512          TCHKNG:
513          TCHKNG:
514          TCHKNG:
515          TCHKNG:
516          TCHKNG:
517          TCHKNG:
518          TCHKNG:
519          TCHKNG:
520          TCHKNG:
521          TCHKNG:
522          TCHKNG:
523          TCHKNG:
524          TCHKNG:
525          TCHKNG:
526          TCHKNG:
527          TCHKNG:
528          TCHKNG:
529          TCHKNG:
530          TCHKNG:
531          TCHKNG:
532          TCHKNG:
533          TCHKNG:
534          TCHKNG:
535          TCHKNG:
536          TCHKNG:
537          TCHKNG:
538          TCHKNG:
539          TCHKNG:
540          TCHKNG:
541          TCHKNG:
542          TCHKNG:
543          TCHKNG:
544          TCHKNG:
545          TCHKNG:
546          TCHKNG:
547          TCHKNG:
548          TCHKNG:
549          TCHKNG:
550          TCHKNG:
551          TCHKNG:
552          TCHKNG:
553          TCHKNG:
554          TCHKNG:
555          TCHKNG:
556          TCHKNG:
557          TCHKNG:
558          TCHKNG:
559          TCHKNG:
560          TCHKNG:
561          TCHKNG:
562          TCHKNG:
563          TCHKNG:
564          TCHKNG:
565          TCHKNG:
566          TCHKNG:
567          TCHKNG:
568          TCHKNG:
569          TCHKNG:
570          TCHKNG:
571          TCHKNG:
572          TCHKNG:
573          TCHKNG:
574          TCHKNG:
575          TCHKNG:
576          TCHKNG:
577          TCHKNG:
578          TCHKNG:
579          TCHKNG:
580          TCHKNG:
581          TCHKNG:
582          TCHKNG:
583          TCHKNG:
584          TCHKNG:
585          TCHKNG:
586          TCHKNG:
587          TCHKNG:
588          TCHKNG:
589          TCHKNG:
590          TCHKNG:
591          TCHKNG:
592          TCHKNG:
593          TCHKNG:
594          TCHKNG:
595          TCHKNG:
596          TCHKNG:
597          TCHKNG:
598          TCHKNG:
599          TCHKNG:
600          TCHKNG:
601          TCHKNG:
602          TCHKNG:
603          TCHKNG:
604          TCHKNG:
605          TCHKNG:
606          TCHKNG:
607          TCHKNG:
608          TCHKNG:
609          TCHKNG:
610          TCHKNG:
611          TCHKNG:
612          TCHKNG:
613          TCHKNG:
614          TCHKNG:
615          TCHKNG:
616          TCHKNG:
617          TCHKNG:
618          TCHKNG:
619          TCHKNG:
620          TCHKNG:
621          TCHKNG:
622          TCHKNG:
623          TCHKNG:
624          TCHKNG:
625          TCHKNG:
626          TCHKNG:
627          TCHKNG:
628          TCHKNG:
629          TCHKNG:
630          TCHKNG:
631          TCHKNG:
632          TCHKNG:
633          TCHKNG:
634          TCHKNG:
635          TCHKNG:
636          TCHKNG:
637          TCHKNG:
638          TCHKNG:
639          TCHKNG:
640          TCHKNG:
641          TCHKNG:
642          TCHKNG:
643          TCHKNG:
644          TCHKNG:
645          TCHKNG:
646          TCHKNG:
647          TCHKNG:
648          TCHKNG:
649          TCHKNG:
650          TCHKNG:
651          TCHKNG:
652          TCHKNG:
653          TCHKNG:
654          TCHKNG:
655          TCHKNG:
656          TCHKNG:
657          TCHKNG:
658          TCHKNG:
659          TCHKNG:
660          TCHKNG:
661          TCHKNG:
662          TCHKNG:
663          TCHKNG:
664          TCHKNG:
665          TCHKNG:
666          TCHKNG:
667          TCHKNG:
668          TCHKNG:
669          TCHKNG:
670          TCHKNG:
671          TCHKNG:
672          TCHKNG:
673          TCHKNG:
674          TCHKNG:
675          TCHKNG:
676          TCHKNG:
677          TCHKNG:
678          TCHKNG:
679          TCHKNG:
680          TCHKNG:
681          TCHKNG:
682          TCHKNG:
683          TCHKNG:
684          TCHKNG:
685          TCHKNG:
686          TCHKNG:
687          TCHKNG:
688          TCHKNG:
689          TCHKNG:
690          TCHKNG:
691          TCHKNG:
692          TCHKNG:
693          TCHKNG:
694          TCHKNG:
695          TCHKNG:
696          TCHKNG:
697          TCHKNG:
698          TCHKNG:
699          TCHKNG:
700          TCHKNG:
701          TCHKNG:
702          TCHKNG:
703          TCHKNG:
704          TCHKNG:
705          TCHKNG:
706          TCHKNG:
707          TCHKNG:
708          TCHKNG:
709          TCHKNG:
710          TCHKNG:
711          TCHKNG:
712          TCHKNG:
713          TCHKNG:
714          TCHKNG:
715          TCHKNG:
716          TCHKNG:
717          TCHKNG:
718          TCHKNG:
719          TCHKNG:
720          TCHKNG:
721          TCHKNG:
722          TCHKNG:
723          TCHKNG:
724          TCHKNG:
725          TCHKNG:
726          TCHKNG:
727          TCHKNG:
728          TCHKNG:
729          TCHKNG:
730          TCHKNG:
731          TCHKNG:
732          TCHKNG:
733          TCHKNG:
734          TCHKNG:
735          TCHKNG:
736          TCHKNG:
737          TCHKNG:
738          TCHKNG:
739          TCHKNG:
740          TCHKNG:
741          TCHKNG:
742          TCHKNG:
743          TCHKNG:
744          TCHKNG:
745          TCHKNG:
746          TCHKNG:
747          TCHKNG:
748          TCHKNG:
749          TCHKNG:
750          TCHKNG:
751          TCHKNG:
752          TCHKNG:
753          TCHKNG:
754          TCHKNG:
755          TCHKNG:
756          TCHKNG:
757          TCHKNG:
758          TCHKNG:
759          TCHKNG:
760          TCHKNG:
761          TCHKNG:
762          TCHKNG:
763          TCHKNG:
764          TCHKNG:
765          TCHKNG:
766          TCHKNG:
767          TCHKNG:
768          TCHKNG:
769          TCHKNG:
770          TCHKNG:
771          TCHKNG:
772          TCHKNG:
773          TCHKNG:
774          TCHKNG:
775          TCHKNG:
776          TCHKNG:
777          TCHKNG:
778          TCHKNG:
779          TCHKNG:
780          TCHKNG:
781          TCHKNG:
782          TCHKNG:
783          TCHKNG:
784          TCHKNG:
785          TCHKNG:
786          TCHKNG:
787          TCHKNG:
788          TCHKNG:
789          TCHKNG:
790          TCHKNG:
791          TCHKNG:
792          TCHKNG:
793          TCHKNG:
794          TCHKNG:
795          TCHKNG:
796          TCHKNG:
797          TCHKNG:
798          TCHKNG:
799          TCHKNG:
800          TCHKNG:
801          TCHKNG:
802          TCHKNG:
803          TCHKNG:
804          TCHKNG:
805          TCHKNG:
806          TCHKNG:
807          TCHKNG:
808          TCHKNG:
809          TCHKNG:
810          TCHKNG:
811          TCHKNG:
812          TCHKNG:
813          TCHKNG:
814          TCHKNG:
815          TCHKNG:
816          TCHKNG:
817          TCHKNG:
818          TCHKNG:
819          TCHKNG:
820          TCHKNG:
821          TCHKNG:
822          TCHKNG:
823          TCHKNG:
824          TCHKNG:
825          TCHKNG:
826          TCHKNG:
827          TCHKNG:
828          TCHKNG:
829          TCHKNG:
830          TCHKNG:
831          TCHKNG:
832          TCHKNG:
833          TCHKNG:
834          TCHKNG:
835          TCHKNG:
836          TCHKNG:
837          TCHKNG:
838          TCHKNG:
839          TCHKNG:
840          TCHKNG:
841          TCHKNG:
842          TCHKNG:
843          TCHKNG:
844          TCHKNG:
845          TCHKNG:
846          TCHKNG:
847          TCHKNG:
848          TCHKNG:
849          TCHKNG:
850          TCHKNG:
851          TCHKNG:
852          TCHKNG:
853          TCHKNG:
854          TCHKNG:
855          TCHKNG:
856          TCHKNG:
857          TCHKNG:
858          TCHKNG:
859          TCHKNG:
860          TCHKNG:
861          TCHKNG:
862          TCHKNG:
863          TCHKNG:
864          TCHKNG:
865          TCHKNG:
866          TCHKNG:
867          TCHKNG:
868          TCHKNG:
869          TCHKNG:
870          TCHKNG:
871          TCHKNG:
872          TCHKNG:
873          TCHKNG:
874          TCHKNG:
875          TCHKNG:
876          TCHKNG:
877          TCHKNG:
878          TCHKNG:
879          TCHKNG:
880          TCHKNG:
881          TCHKNG:
882          TCHKNG:
883          TCHKNG:
884          TCHKNG:
885          TCHKNG:
886          TCHKNG:
887          TCHKNG:
888          TCHKNG:
889          TCHKNG:
890          TCHKNG:
891          TCHKNG:
892          TCHKNG:
893          TCHKNG:
894          TCHKNG:
895          TCHKNG:
896          TCHKNG:
897          TCHKNG:
898          TCHKNG:
899          TCHKNG:
900          TCHKNG:
901          TCHKNG:
902          TCHKNG:
903          TCHKNG:
904          TCHKNG:
905          TCHKNG:
906          TCHKNG:
907          TCHKNG:
908          TCHKNG:
909          TCHKNG:
910          TCHKNG:
911          TCHKNG:
912          TCHKNG:
913          TCHKNG:
914          TCHKNG:
915          TCHKNG:
916          TCHKNG:
917          TCHKNG:
918          TCHKNG:
919          TCHKNG:
920          TCHKNG:
921          TCHKNG:
922          TCHKNG:
923          TCHKNG:
924          TCHKNG:
925          TCHKNG:
926          TCHKNG:
927          TCHKNG:
928          TCHKNG:
929          TCHKNG:
930          TCHKNG:
931          TCHKNG:
932          TCHKNG:
933          TCHKNG:
934          TCHKNG:
935          TCHKNG:
936          TCHKNG:
937          TCHKNG:
938          TCHKNG:
939          TCHKNG:
940          TCHKNG:
941          TCHKNG:
942          TCHKNG:
943          TCHKNG:
944          TCHKNG:
945          TCHKNG:
946          TCHKNG:
947          TCHKNG:
948          TCHKNG:
949          TCHKNG:
950          TCHKNG:
951          TCHKNG:
952          TCHKNG:
953          TCHKNG:
954          TCHKNG:
955          TCHKNG:
956          TCHKNG:
957          TCHKNG:
958          TCHKNG:
959          TCHKNG:
960          TCHKNG:
961          TCHKNG:
962          TCHKNG:
963          TCHKNG:
964          TCHKNG:
965          TCHKNG:
966          TCHKNG:
967          TCHKNG:
968          TCHKNG:
969          TCHKNG:
970          TCHKNG:
971          TCHKNG:
972          TCHKNG:
973          TCHKNG:
974          TCHKNG:
975          TCHKNG:
976          TCHKNG:
977          TCHKNG:
978          TCHKNG:
979          TCHKNG:
980          TCHKNG:
981          TCHKNG:
982          TCHKNG:
983          TCHKNG:
984          TCHKNG:
985          TCHKNG:
986          TCHKNG:
987          TCHKNG:
988          TCHKNG:
989          TCHKNG:
990          TCHKNG:
991          TCHKNG:
992          TCHKNG:
993          TCHKNG:
994          TCHKNG:
995          TCHKNG:
996          TCHKNG:
997          TCHKNG:
998          TCHKNG:
999          TCHKNG:
1000         TCHKNG:

```

SECTION 5

```

486                                     PAGE
487 |-----|
488 |         FIXED DISK BIOS ENTRY POINT         |
489 |-----|
490
491 01A9          DISK_IO PROC          FAR
492          ASSUME DS:DATA,ES:NOTHING
493          JMP     DL,80H              | TEST FOR FIXED DISK DRIVE
494          JAE     A1                  | YES, HANDLE HERE
495          INT     40H                | DISKETTE HANDLER
496          RET     2                  | BACK TO CALLER
497          RET     2
498
499 01B3          A1:
500          STI     OR                 | ENABLE INTERRUPTS
501          JNZ     AH,AH              |
502          INT     40H                | RESET NEC WHEN AH=0
503          SUB     AH,AH              |
504          CMP     DL,(80H + S_MAX_FILE - 1)
505          JAE     RET_2              |
506          RET     2
507
508 01C1          A2:
509          CMP     AH,00H             | GET PARAMETERS IS A SPECIAL CASE
510          JNZ     GET_PARM_N
511          CMP     AH,TSH
512          JNZ     A4
513          JMP     READ_DASD_TYPE
514
515 01D1          A4:
516          ENTER  8,0                | SAVE REGISTERS DURING OPERATION
517          PUSH  BX                   | SAVE (BP) AND MAKE ROOM FOR @CMD_BLOCK
518          PUSH  CX                   | IN THE STACK. THE COMMAND_BLOCK IS:
519          PUSH  DX                   | @CMD_BLOCK == BYTE PTR [BP]-8
520          PUSH  DS
521          PUSH  ES
522          PUSH  SI
523          PUSH  DI
524          OR     AH,AH               | CHECK FOR RESET
525          JNZ     A5
526          MOV   DL,80H              | FORCE DRIVE 80 FOR RESET
527          CALL  DISK_IO_CONT        | PERFORM THE OPERATION
528          CALL  DOS                 | ESTABLISH SEGMENT
529          MOV   AH,@DISK_STATUS1    | GET STATUS FROM OPERATION
530          CMP   AH,1                | SET THE CARRY FLAG TO INDICATE
531          JNC   SUCCESS             | SUCCESS OR FAILURE
532          POP   DI                  | RESTORE REGISTERS
533          POP   SI
534          POP   ES
535          POP   DS
536          POP   DX
537          POP   CX
538          POP   BX
539          LEAVE
540          RET     2                  | ADJUST (SP) AND RESTORE (BP)
541          RET     2                  | THROW AWAY SAVED FLAGS
542
543 01FB          MI LABEL WORD
544          DW     DISK_RESET          | FUNCTION TRANSFER TABLE
545          DW     RETURN_STATUS       | 000H
546          DW     DISK_READ           | 001H
547          DW     DISK_WRITE         | 002H
548          DW     DISK_VERIFY        | 003H
549          DW     FMT_TRK            | 004H
550          DW     BAD_COMMAND        | 005H
551          DW     BAD_COMMAND        | 006H
552          DW     BAD_COMMAND        | 007H
553          DW     INIT_DRV           | 008H
554          DW     RD_LONG            | 009H
555          DW     WR_LONG            | 00AH
556          DW     DISK_SEEK          | 00BH
557          DW     DISK_RESET         | 00CH
558          DW     BAD_COMMAND        | 00DH
559          DW     TEST_RDY           | 00EH
560          DW     HDISK_RECAL        | 00FH
561          DW     BAD_COMMAND        | 010H
562          DW     BAD_COMMAND        | 011H
563          DW     CTLR_DIAGNOSTIC    | 012H
564          DW     CTLR_DIAGNOSTIC    | 013H
565          EQU   $-MI                | 014H
566
567 0225          MIL EQU
568
569 0225          DISK_IO_CONT PROC NEAR
570          CALL  DOS NEAR            | ESTABLISH SEGMENT
571          CMP   AH,01H              | RETURN STATUS
572          JNZ   SU0
573          RET   RETURN_STATUS
574
575 0230          SU0:
576          MOV   @DISK_STATUS1,0     | RESET THE STATUS INDICATOR
577          PUSH  BX                   | SAVE DATA ADDRESS
578          MOV   BL,@HF_NLM          | GET NUMBER OF DRIVES
579          AND   DL,7FH              | GET DRIVE AS 0 OR 1
580          CMP   BL,DL               | INVALID DRIVE
581          JBE   BAD_COMMAND_POP     |
582          PUSH  ES
583          CALL  GET_VEC              | GET DISK PARAMETERS
584          MOV   AX,BORD PTR ES:[BX][6] | GET WRITE PRE-COMPENSATION CYLINDER
585          SHR   AX,2
586          MOV   @CMD_BLOCK,AL       |
587          MOV   AL,BYTE PTR ES:[BX][8] | GET CONTROL BYTE MODIFIER
588          PUSH  DX
589          MOV   DX,HF_REG_PORT      |
590          OUT  DX,AL                | SET EXTRA HEAD OPTION
591          POP   DX
592          POP   ES
593          MOV   AH,@CONTROL_BYTE     | SET EXTRA HEAD OPTION IN
594          AND   AH,0COH             | CONTROL BYTE
595          OR   AH,AL
596          MOV   @CONTROL_BYTE,AH
597          POP   AX
598          MOV   @CMD_BLOCK+1,AL      | SECTOR COUNT
599          PUSH  AX
600          MOV   AL,CL                | GET SECTOR NUMBER
601          AND   AL,3FH
    
```

```

600 0271 88 46 FA      MOV     @CMD_BLOCK+2,AL
601 0274 88 6E FB      MOV     @CMD_BLOCK+3,CH      ; GET CYLINDER NUMBER
602 0277 8A C1         MOV     AL,CL
603 0279 C0 E8 06      SHR     AL,6
604 027C 88 46 FC      MOV     @CMD_BLOCK+4,AL      ; CYLINDER HIGH ORDER 2 BITS
605 027F 8A C2         MOV     AL,CL                ; 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 028F 50             PUSH   AX
613 0290 8A C4         MOV     AL,AH                ; GET INTO LOW BYTE
614 0292 32 E4         XOR     AH,AH                ; ZERO HIGH BYTE
615 0294 D1 E0         SAL     AX,1                 ; *2 FOR TABLE LOOKUP
616 0296 8B F0         MOV     SI,AX                ; PUT INTO SI FOR BRANCH
617 0298 3D 002A       CMP     AX,NIL               ; TEST WITHIN RANGE
618 029B 73 1A         JNB     BAD_COMMAND_POP
619 029D 58             POP     AX                    ; RESTORE AX
620 029E 5B             POP     BX                    ; AND DATA ADDRESS
621 029F 51             PUSH   CX
622 02A0 50             PUSH   AX                    ; ADJUST ES:BX
623 02A1 8B CB         MOV     CX,BX                ; GET 3 HIGH ORDER NIBBLES OF BX
624 02A3 C1 E9 04      SHR     CX,3
625 02A6 8C C0         MOV     AX,ES
626 02A8 03 C1         ADD     AX,CX
627 02AA 8E C0         MOV     ES,AX
628 02AC B1 E3 000F    AND     BX,000FH            ; ES:BX CHANGED TO ES:000X
629 02B0 58             POP     AX
630 02B1 59             POP     CX
631 02B2 2E: FF A4 01FB R JMP     WORD PTR CS:[SI + OFFSET MI]
632 02B7             BAD_COMMAND_POP:
633 02B7 58             POP     AX
634 02B8 5B             POP     BX
635 02B9             BAD_COMMAND:
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             DISK_IO_CONT ENDP
640
641 ;-----
642 ; RESET THE DISK SYSTEM (AH=00H) ;
643 ;-----
644
645 02C1             DISK_RESET PROC NEAR
646 02C1 FA             CLI
647 02C2 E4 A1         IN      AL,INTB01           ; GET THE MASK REGISTER
648 02C4 EB 00         JMP     $+2
649 02C6 24 BF         AND     AL,0BFH            ; ENABLE FIXED DISK INTERRUPT
650 02C8 E6 A1         OUT     INTB01,AL          ; START INTERRUPTS
651 02CA FB             STI
652 02CB 80 04         MOV     AL,04H
653 02CD 8A 03F6       MOV     DX,HF_REG_PORT
654 02D0 EE             OUT     DX,AL
655 02D1 B9 000A       MOV     CX,10
656 02D4 49             DEC     CX
657 02D5 75 FD         JNZ     DRD                ; WAIT 4.8 MICRO-SEC
658 02D7 A0 0076 R     MOV     AL,@CONTROL_BYTE
659 02DA 24 0F         AND     AL,0FH
660 02DC EE             OUT     DX,AL
661 02DD EB 08F3 R     CALL   NOT_BUSY
662 02E0 75 2D         JNZ     DRERR              ; TIME OUT ON RESET
663 02E2 BA 01F1       MOV     DX,HF_PORT+1
664 02E5 EC             IN      AL,DX
665 02E6 3C 01         CMP     AL,1
666 02E8 75 25         JNZ     DRERR              ; BAD RESET STATUS
667 02EA 80 66 FD EF   AND     @CMD_BLOCK+5,0EFH  ; SET TO DRIVE 0
668 02EE 2A D2         SUB     DL,DL
669 02F0 E8 03F1 R     CALL   INIT_DRY            ; SET MAX HEADS
670 02F3 EB 0466 R     CALL   HDISK_RECAL         ; RECAL TO RESET SEEK SPEED
671 02F6 80 3E 0075 R 01 CMP     @HF_NUM,1
672 02F8 76 0C         JBE     DRE
673 02FD 80 4E FD 10   OR      @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 08 DRERR:  MOV     @DISK_STATUS1,BAD_RESET ; CARD FAILED
680 0314 C3             RET
681 0315             DISK_RESET ENDP
682
683 ;-----
684 ; DISK STATUS ROUTINE (AH = 01H) ;
685 ;-----
686
687 0315             RETURN_STATUS PROC NEAR
688 0315 A0 0074 R     MOV     AL,@DISK_STATUS1    ; OBTAIN PREVIOUS STATUS
689 0318 C6 06 0074 R 00 MOV     @DISK_STATUS1,0    ; RESET STATUS
690 031D C3             RET
691 031E             RETURN_STATUS ENDP
    
```

SECTION 5

```

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_VERF PROC NEAR
716 032C C6 46 FE 40 MOV #CMD_BLOCK+6,VERIFY_CMD
717 0330 E8 08C R CALL COMMAND
718 0333 75 08 JNZ VERF_EXIT ; CONTROLLER STILL BUSY
719 0336 E8 08C2 R CALL WAIT
720 0338 75 03 JNZ VERF_EXIT ; TIME OUT
721 033A E8 0630 R CALL CHECK_STATUS
722 033D VERF_EXIT:
723 033D C9 RET
724 033E DISK_VERF ENDP
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 06C4 R CALL GET_VEC
735 0347 26 8A 47 0E MOV AL,ES:[BX][14] ; GET DISK PARAMETERS ADDRESS
736 0348 88 46 F9 MOV #CMD_BLOCK+1,AL ; GET SECTORS/TRACK
737 034E 5B POP BX ; SET SECTOR COUNT IN COMMAND
738 034F 07 POP ES
739 0350 E9 050A R JMP CMD_OF ; GO EXECUTE THE COMMAND
740 0353 FMT_TRK ENDP
741
742 ;-----
743 ; READ DASD TYPE (AH = 15H) ;
744 ;-----
745
746 READ_DASD_TYPE LABEL NEAR
747 0353 READ_D_T PROC FAR ; GET DRIVE PARAMETERS
748 0353 IE PUSH DS ; SAVE REGISTERS
749 0354 06 PUSH ES
750 0355 53 PUSH BX
751 0356 53 ASSUME DS:DATA
752 0356 E8 0000 E CALL DDS ; ESTABLISH ADDRESSING
753 0359 C6 06 0074 R 00 MOV #DISK_STATUS1,0
754 035E 8A 1E 0075 R MOV BL,#HF_NUM ; GET NUMBER OF DRIVES
755 0362 80 E2 7F AND DL,7FH ; GET DRIVE NUMBER
756 0365 3A DA CMP BL,DL
757 0367 76 22 JBE RDT_NOT_PRESENT ; RETURN DRIVE NOT PRESENT
758 0369 E8 06C4 R CALL GET_VEC ; GET DISK PARAMETER ADDRESS
759 036C 26 8A 47 0E MOV AL,ES:[BX][2] ; HEADS
760 0370 26 8A 4F 0E MOV CL,ES:[BX][14]
761 0374 F6 E9 IMUL CL ; * NUMBER OF SECTORS
762 0376 26 8B 0F MOV CX,ES:[BX] ; MAX NUMBER OF CYLINDERS
763 0379 49 DEC CX ; LEAVE ONE FOR DIAGNOSTICS
764 037A FT E9 IMUL CX ; NUMBER OF SECTORS
765 037C 8B CA MOV CX,DX ; HIGH ORDER HALF
766 037E 8B D0 MOV DX,AX ; LOW ORDER HALF
767 0380 2B C0 SUB AX,AX
768 0382 B4 03 MOV AH,03H ; INDICATE FIXED DISK
769 0384 5B POP BX ; RESTORE REGISTERS
770 0385 07 POP ES
771 0386 1F POP DS
772 0387 F8 CLC ; CLEAR CARRY
773 0388 CA 0002 RET 2
774 038B RDT_NOT_PRESENT:
775 038B 2B C0 SUB AX,AX ; DRIVE NOT PRESENT RETURN
776 038D 8B C8 MOV CX,AX ; ZERO BLOCK COUNT
777 038F 8B D0 MOV DX,AX
778 0391 EB F1 JMP RDT2
779 0393 READ_D_T ENDP
780 0393
    
```

```

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 ASSUME DS:ABS0
792 0396 BB ---- R MOV AX,ABS0 ; ESTABLISH ADDRESSING
793 0399 8E DB MOV DS,AX
794 039B F8 C8 01 TEST DL,1 ; CHECK FOR DRIVE 1
795 039E 74 06 JZ GO,1
796 03A0 C4 1E 0118 R LES BX,#HP1_TBL_VEC
797 03A4 EB 04 JMP SHORT GT
798 03A6 C4 1E 0104 R LES BX,#HP1_TBL_VEC
799 ASSUME DS:DATA
800 03AA CALL DDS ; ESTABLISH SEGMENT
801 03AA EB 0000 E SUB DL,80H
802 03AD 80 EA 80 CMP DL,MAX_FILE ; TEST WITHIN RANGE
803 03B0 80 FA 02 JAE G4
804 03B3 73 2C MOV #DISK_STATUS1,0
805 03B5 C6 06 0074 R 00 MOV AX,EST[BX] ; MAX NUMBER OF CYLINDERS
806 03B8 26 0B 07 AND AX,2 ; ADJUST FOR 0-N
807 03BD 2D 0002 SUB CH,AL
808 03C0 8A E8 MOV AX,0300H ; HIGH TWO BITS OF CYLINDER
809 03C2 E5 0300 AND SHR AX,1
810 03C5 01 E8 SHR AX,1
811 03C7 D1 E8 OR AL,ES:[BX][14] ; SECTORS
812 03C9 26 0A 47 0E CL--AL MOV DL,ES:[BX][2] ; HEADS
813 03CD 8A C8 MOV DH,ES:[BX][2] ; HEADS
814 03CF 26 18 77 02 DEC DH ; 0-N RANGE
815 03D3 FE CE MOV DL,#HP_NUM ; DRIVE COUNT
816 03D5 9A 16 0075 R SUB AX,AX
817 03D9 2B C0
818 03DB
819 03DB 5B POP BX ; RESTORE REGISTERS
820 03DC 07 C8 POP ES
821 03DD 1F POP DS
822 03DE CA 0002 RET 2
823 03E1
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 GS
831 03F1 GET_PARM ENDP
832
833 |-----|
834 | INITIALIZE DRIVE (AH = 09H) |
835 |-----|
836
837 03F1 INIT_DRV PROC NEAR
838 03F1 C6 46 FE 91 MOV #CMD_BLOCK+6,SET_PARM_CMD
839 03F8 E8 05C4 R CALL GET_VEC ; ES:BX -> PARAMETER BLOCK
840 03FC FE C8 MOV AL,ES:[BX][2] ; GET NUMBER OF HEADS
841 03FE 8A 66 FD DEC AL ; CONVERT TO 0-INDEX
842 0401 80 E4 F0 MOV AH,#CMD_BLOCK+5 ; GET SOH REGISTER
843 0404 0A E8 AND AH,0FH ; CHANGE HEAD NUMBER
844 0406 88 66 FD OR AH,AL ; TO MAX HEAD
845 0409 26 8A 47 0E MOV #CMD_BLOCK+5,AH
846 040D 88 46 F9 MOV AL,ES:[BX][14] ; MAX SECTOR NUMBER
847 0410 2B C0 SUB #CMD_BLOCK+1,AL
848 0412 88 46 FB MOV AX,AX ; ZERO FLAGS
849 0415 EB 055C R CALL COMMAND ; TELL CONTROLLER
850 0418 75 08 JNZ INIT_EXIT ; CONTROLLER BUSY ERROR
851 041A E8 05F3 R CALL NOT_BUSY ; WAIT FOR IT TO BE DONE
852 041D 75 03 JNZ INIT_EXIT ; TIME OUT
853 041F EB 0630 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+6,READ_CMD OR ECC_MODE
864 0427 E9 04C6 R JMP COMMAND
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 COMMAND
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 055C R CALL COMMAND
883 0438 75 14 JNZ DS_EXIT ; CONTROLLER BUSY ERROR
884 043A E8 05C2 R CALL WAIT ; TIME OUT ON SEEK
885 043D 75 0F JNZ DS_EXIT
886 043F E8 0630 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 044F DISK_SEEK ENDP
893

```

SECTION 5

```

894 PAGE
895 |-----|
896 | TEST DISK READY (AH = 10H) |
897 |-----|
898
899 044F PROC NEAR
900 044F EB 08F3 R CALL NOT_BUSY ; WAIT FOR CONTROLLER
901 0452 75 11 JNZ TR_EX
902 0454 5A 46 FD MOV AL,*CMD_BLOCK+5 ; SELECT DRIVE
903 0457 BA 01F6 MOV DX,HF_PORT+6
904 045A EE 01F6 OUT DX,AL
905 045B EB 0642 R CALL CHECK_ST ; CHECK STATUS ONLY
906 045E 75 05 JNZ TR_EX
907 0460 C6 06 0074 R 00 MOV *DISK_STATUS,0 ; WIPE OUT DATA CORRECTED ERROR
908 0465 C3 RET
909 0466 TST_RDY ENDP
910
911 |-----|
912 | RECALIBRATE (AH = 11H) |
913 |-----|
914
915 0466 PROC NEAR
916 0466 C6 46 FE 10 MOV *CMD_BLOCK+6,RECAL_CMD
917 046A EB 085C R CALL COMMAND ; START THE OPERATION
918 046D 75 19 JNZ RECAL_EXIT ; ERROR
919 046F EB 08C2 R CALL WAIT ; WAIT FOR COMPLETION
920 0472 74 05 JZ RECAL_X ; TIME OUT ONE OK ?
921 0474 EB 08C2 R CALL WAIT ; WAIT FOR COMPLETION LONGER
922 0477 75 0F JNZ RECAL_EXIT ; TIME OUT TWO TIMES IS ERROR
923 0479
924 0479 EB 0630 R CALL CHECK_STATUS
925 047C 80 3E 0074 R 40 CMP *DISK_STATUS,BAD_SEEK ; SEEK NOT COMPLETE
926 0481 75 05 JNE RECAL_EXIT ; IS OK
927 0483 C6 06 0074 R 00 MOV *DISK_STATUS,0
928 0488 RECAL_EXIT:
929 0488 80 3E 0074 R 00 CMP *DISK_STATUS,0
930 048D C3 RET
931 048E HDISK_RECAL ENDP
932
933 |-----|
934 | CONTROLLER DIAGNOSTIC (AH = 14H) |
935 |-----|
936
937 048E 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 E4 A1 OUT INTB01,AL
943 0497 E4 21 IN AL,INTA01 ; LET INTERRUPTS PASS THRU TO
944 0499 24 FB AND AL,0FBH ; SECOND CHIP
945 049B EB 00 JMP $+2
946 049D E4 21 OUT INTA01,AL
947 049F FB STI
948 04A0 EB 08F3 R CALL NOT_BUSY ; WAIT FOR CARD
949 04A3 75 1A JZ CD_ERR ; BAD CARD
950 04A5 BA 01F7 MOV DX,HF_PORT+7
951 04A8 B0 90 MOV AL,DIAG_CMD ; START DIAGNOSE
952 04AA EE 01F6 OUT DX,AL
953 04AB EB 08F3 R CALL NOT_BUSY ; WAIT FOR IT TO COMPLETE
954 04AE B4 80 MOV AH,TIME_OUT
955 04B0 75 0F JNZ CD_EXIT ; TIME OUT ON DIAGNOSTIC
956 04B2 BA 01F1 MOV DX,HF_PORT+1 ; GET ERROR REGISTER
957 04B5 EC IN AL,DX
958 04B6 A2 008D R MOV *HF_ERROR,AL ; SAVE IT
959 04B9 B4 00 MOV AH,0 ; CHECK FOR ALL OK
960 04BB 3C 01 CMP AL,1
961 04BD 74 02 JE SHORT CD_EXIT
962 04BF B4 20 MOV AH,BAD_CRTL
963 04C1 CD_ERR:
964 04C1 88 26 0074 R MOV *DISK_STATUS,AH
965 04C5 C3 RET
966 04C6 CTLR_DIAGNOSTIC ENDP
967
968 |-----|
969 | COMMAND |
970 | REPEATEDLY INPUTS DATA TILL |
971 | NSECTOR RETURNS ZERO |
972 |-----|
973 04C6 PROC NEAR
974 04C6 EB 06A1 R CALL CHECK_DMA ; CHECK 64K BOUNDARY ERROR
975 04C9 72 3E JC CMD_ABORT
976 04CB 8B FB MOV DI,BX
977 04CD EB 055C R CALL COMMAND ; OUTPUT COMMAND
978 04D0 75 32 JNZ CMD_ABORT
979 04D2
980 04D2 EB 08C2 R CALL WAIT ; WAIT FOR DATA REQUEST INTERRUPT
981 04D5 75 20 JNZ TM_OUT ; TIME OUT
982 04D7 89 0100 MOV CX,ESD ; SECTOR SIZE IN WORDS
983 04DA BA 01F0 MOV DX,HF_PORT
984 04DD FA CLI
985 04DE FC CLD
986 04DF F3 6D REP
987 04E1 FB STI
988 04E2 F6 46 FE 02 TEST *CMD_BLOCK+6,ECC_MODE ; CHECK FOR NORMAL INPUT
989 04E5 74 12 JZ CMD_TS
990 04E8 EB 061A R CALL WAIT_DRQ ; WAIT FOR DATA REQUEST
991 04EB 72 17 JC TM_OUT
992 04ED BA 01F0 MOV DX,HF_PORT
993 04F0 89 0004 MOV CX,4
994 04F3 EC CALL IN AL,DX ; GET ECC BYTES
995 04F4 26 88 05 MOV ES:BYTE PTR [DI],AL ; GO SLOW FOR BOARD
996 04F7 47 INC DI
997 04F8 E2 F9 LOOP CMD_12
998 04FA EB 0630 R CALL CHECK_STATUS
999 04FD 75 05 JNZ CMD_ABORT ; ERROR RETURNED
1000 04FF FE 4E F9 DEC *CMD_BLOCK+1 ; CHECK FOR MORE
1001 0502 75 0E JNZ SHORT CMD_11
1002 0504 CMD_12:
1003 0504 TM_OUT:
1004 0504 CS RET

```

```

1005 PAGE
1006 -----
1007 ; COMMAND0
1008 ; REPEATEDLY OUTPUTS DATA TILL
1009 ; NSECTOR RETURNS ZERO
1010 -----
1011 0505 COMMAND0:
1012 0505 EB 06A1 R CALL CHECK_DMA ; CHECK 64K BOUNDARY ERROR
1013 0508 72 FA JC CMD_ABORT
1014 050A BB F3 MOV SI,BX
1015 050C EB 05BC R CALL COMMAND ; OUTPUT COMMAND
1016 050F 75 F3 JNZ CMD_ABORT
1017 0511 EB 061A R CALL WAIT_DRQ ; WAIT FOR DATA REQUEST
1018 0514 72 EE JC TM_OUT ; TOO LONG
1019 0516 1E CMD_01: PUSH 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 CLD
1025 0520 FC CLD
1026 0521 F3/ 6F REP OUTSW
1027 0523 FB STI
1028 0524 1F POP DS ; RESTORE DS
1029 0525 F4 46 FE 02 TEST %CMD_BLOCK+6,ECC_MODE ; CHECK FOR NORMAL OUTPUT
1030 0529 74 12 JZ CMD_D3
1031 052B EB 061A R CALL WAIT_DRQ ; WAIT FOR DATA REQUEST
1032 052E 72 D4 JC DX_HF_PORT
1033 0530 BA 01F0 MOV DX,HF_PORT
1034 0533 B9 0004 MOV CX,4 ; OUTPUT THE ECC BYTES
1035 0536 26 8A 04 CMD_02: MOV AL,ES:BYTE PTR [SI]
1036 0539 EE OUT DX,AL
1037 053A 44 INC SI
1038 053B E2 F9 LOOP CMD_02
1039 053D CMD_03:
1040 053D EB 05C2 R CALL WAIT ; WAIT FOR SECTOR COMPLETE INTERRUPT
1041 0540 75 C2 JNZ TM_OUT ; ERROR RETURNED
1042 0542 EB 0630 R CALL CHECK_STATUS
1043 0545 75 BD JNZ CMD_ABORT
1044 0547 F4 06 008C R 08 TEST %HF_STATUS,ST_DRQ ; CHECK FOR MORE
1045 054C 75 C8 JNZ SHORT_CMD_01
1046 054E BA 01F2 MOV DX,HF_PORT+2 ; CHECK RESIDUAL SECTOR COUNT
1047 0551 EC IN AL,DX
1048 0552 AB FF TEST AL,0FFH
1049 0554 74 05 JZ CMD_04 ; COUNT = 0 OK
1050 0556 C6 06 0074 R BB MOV %DISK_STATUS1,UNDEF_ERR ; OPERATION ABORTED - PARTIAL TRANSFER
1051 055B CMD_04:
1052 055B C3 RET
1053
1054 ;-----
1055 ; COMMAND
1056 ; THIS ROUTINE OUTPUTS THE COMMAND BLOCK
1057 ; OUTPUT
1058 ; BL = STATUS
1059 ; BH = ERROR REGISTER
1060 ;-----
1061
1062 055C COMMAND PROC NEAR
1063 055C 53 PUSH BX ; WAIT FOR SEEK COMPLETE AND READY
1064 055D B9 0600 MOV CX,DELAY_2 ; SET INITIAL DELAY BEFORE TEST
1065 0560 CMD_01:
1066 0560 51 PUSH CX ; SAVE LOOP COUNT
1067 0561 EB 044F R CALL TST_RDY ; CHECK DRIVE READY
1068 0564 59 POP CX
1069 0565 74 0B JZ COMMAND2 ; DRIVE IS READY
1070 0567 80 3E 0074 R 80 CMP %DISK_STATUS1,TIME_OUT ; TST_RDY TIMED OUT--GIVE UP
1071 056C 74 48 JZ CMD_TMOUT
1072 056E E2 F0 LOOP CMD_01 ; KEEP TRYING FOR A WHILE
1073 0570 EB 49 JMP SHORT_CMD_04 ; ITS NOT GOING TO GET READY
1074 0572 CMD_02:
1075 0572 5B POP DI
1076 0573 57 PUSH DI
1077 0574 C6 06 008E R 00 MOV %HF_INT_FLAG,0 ; RESET INTERRUPT FLAG
1078 0579 FA CLI ; INHIBIT INTERRUPTS WHILE CHANGING MASK
1079 057A E4 A1 IN AL,INTB01 ; TURN ON SECOND INTERRUPT CHIP
1080 057C 24 BF AND AL,0BFH
1081 057E EB 00 JMP $+2
1082 0580 E6 A1 OUT INTB01,AL ; LET INTERRUPTS PASS THRU TO
1083 0582 E4 21 IN AL,INTA01 ; SECOND CHIP
1084 0584 24 BF AND AL,0BFH
1085 0586 EB 00 JMP $+2
1086 0588 E6 21 OUT INTA01,AL
1087 058A FB STI
1088 058B 33 XOR DI,D1 ; INDEX THE COMMAND TABLE
1089 058D BA 01F1 MOV DX,HF_PORT+1 ; DISK ADDRESS
1090 0590 F6 06 0076 R C0 TEST %CONTROL_BYTE,0C0H ; CHECK FOR RETRY SUPPRESSION
1091 0595 74 11 JZ COMMAND3
1092 0597 8A 46 FE MOV AL,%CMD_BLOCK+6 ; YES=GET OPERATION CODE
1093 059A 24 F0 AND AL,0F0H ; GET RID OF MODIFIERS
1094 059C 3C 20 CMP AL,20H ; 20H-40H IS READ, WRITE, VERIFY
1095 059E 72 08 JB COMMAND3
1096 05A0 3C 40 CMP AL,40H
1097 05A2 77 04 JA COMMAND3
1098 05A4 80 4E FE 01 OR %CMD_BLOCK+6,NO_RETRIES ; VALID OPERATION FOR RETRY SUPPRESS
1099 05A8 CMD_03:
1100 05A8 8A 43 F8 MOV AL,[%CMD_BLOCK+D1] ; GET THE COMMAND STRING BYTE
1101 05AB EE OUT DX,AL ; GIVE IT TO CONTROLLER
1102 05AC 47 INC DI ; NEXT BYTE IN COMMAND BLOCK
1103 05AD 42 INC DX ; NEXT DISK ADAPTER REGISTER
1104 05AE 81 FA 01F8 CMP DX,HF_PORT+8 ; ALL DONE?
1105 05B2 75 F4 JNZ COMMAND3 ; NO--GO DO NEXT ONE
1106 05B4 BF POP DI
1107 05B5 C3 RET ; ZERO FLAG IS SET
1108 05B6 CMD_TMOUT:
1109 05B6 C6 06 0074 R 20 MOV %DISK_STATUS1,BAD_CNTRL
1110 05B9 CMD_04:
1111 05B9 5B POP BX
1112 05BC 80 3E 0074 R 00 CMP %DISK_STATUS1,0 ; SET CONDITION CODE FOR CALLER
1113 05C1 C3 RET
1114 05C2 COMMAND ENDP
    
```

SECTION 5

```

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

```

```

1228 PAGE
1229 |-----|
1230 | CHECK FIXED DISK ERROR REGISTER |
1231 |-----|
1232 0676 CHECK_ER PROC NEAR
1233 0676 BA 01F1 MOV DX,HF_PORT+1 ; GET THE ERROR REGISTER
1234 0679 EC IN AL,DX
1235 067A A2 008D R MOV #HF_ERROR,AL
1236 067D 53 PUSH BX
1237 067E B9 0008 MOV CX,8 ; TEST ALL 8 BITS
1238 0681 D0 E0 SHL AL,1 ; MOVE NEXT ERROR BIT TO CARRY
1239 0683 72 02 JC CK2 ; FOUND THE ERROR
1240 0685 E2 FA LOOP CK1 ; KEEP TRYING
1241 0687 BB 0698 R CKZ1: MOV BX,OFFSET_ERR_TBL ; COMPUTE ADDRESS OF
1242 068A 03 D9 ADD BX,CX ; ERROR CODE
1243 068C 2E1 8A 27 MOV AH,BYTE PTR CS:[BX] ; GET ERROR CODE
1244 068F 88 26 0074 R CKEX: MOV #DISK_STATUS1,AH ; SAVE ERROR CODE
1245 0693 8B POP BX
1246 0694 80 FC 00 CMP AH,0
1247 0697 C3 RET
1248 0698 E0 ERR_TBL DB NO_ERR
1249 0699 02 40 01 BB DB BAD_ADDR_MARK,BAD_SEEK,BAD_CMD,UNDEF_ERR
1250 069D 04 BB 10 0A DB RECORD_NOT_FND,UNDEF_ERR,BAD_ECC,BAD_SECTOR
1251 06A1 CHECK_ER ENDP
1252 |-----|
1253 | CHECK DMA |
1254 | -CHECK ES:BX AND # SECTORS TO MAKE SURE THAT IT WILL |
1255 | FIT WITHOUT SEGMENT OVERFLOW. |
1256 | -ES:BX HAS BEEN REVISED TO THE FORMAT SSS:00X |
1257 | -OK IF # SECTORS < 80H (7FH IF LONG READ OR WRITE) |
1258 | -OK IF # SECTORS = 80H (7FH) AND BX <= 00H (04H) |
1259 | -ERROR OTHERWISE |
1260 |-----|
1261 06A1 CHECK_DMA PROC NEAR
1262 06A1 50 PUSH AX ; SAVE REGISTERS
1263 06A2 B8 8000 MOV AX,8000H ; AH = MAX # SECTORS AL = MAX OFFSET
1264 06A5 F6 46 FE 02 TEST #CMD_BLOCK+6,ECC_MODE
1265 06A9 74 03 JZ CKD1
1266 06AB B8 7F04 MOV AX,7F04H ; ECC IS 4 MORE BYTES
1267 06AE 3A 66 F9 CMP AH,#CMD_BLOCK+1 ; NUMBER OF SECTORS
1268 06B1 77 06 JA CKDOK ; IT WILL FIT
1269 06B3 72 07 JB CKDERR ; TOO MANY
1270 06B5 3A C3 CMP AL,BL ; CHECK OFFSET ON MAX SECTORS
1271 06B7 72 03 JB CKDERR ; ERROR
1272 06B9 F8 CLC ; CLEAR CARRY
1273 06BA 58 POP AX
1274 06BB C3 RET ; NORMAL RETURN
1275 06BD C6 06 0074 R 09 CKDERR: STC ; INDICATE ERROR
1276 06C0 F9 MOV #DISK_STATUS1,DMA_BOUNDARY
1277 06C2 58 POP AX
1278 06C3 C3 RET
1279 06C4 CHECK_DMA ENDP
1280 |-----|
1281 | SET UP ES:BX-> DISK PARAMS |
1282 |-----|
1283 06C4 GET_VEC PROC NEAR
1284 06C4 2B C0 SUB AX,AX ; GET DISK PARAMETER ADDRESS
1285 06C6 8E C0 MOV ES,AX
1286 06C8 F6 C2 01 ASSUME ES:ABS0
1287 06CB 74 07 TEST DL,1
1288 06CD 26: C4 1E 0118 R JZ GV_0
1289 06D0 EB 05 LES BX,#HF1_TBL_VEC ; ES:BX -> DRIVE PARAMETERS
1290 06D2 58 SHORT GV_EXIT
1291 06D4 GV_0: LES BX,#HF1_TBL_VEC ; ES:BX -> DRIVE PARAMETERS
1292 06D9 GV_EXIT: RET
1293 06DA GET_VEC ENDP
1294 |-----|
1295 |--- HARDWARE INT 76H -- ( IRQ LEVEL 14 ) ---|
1296 | | |
1297 | | |
1298 | | |
1299 | | |
1300 | | |
1301 | | |
1302 | | |
1303 | | |
1304 |-----|
1305 06DA HD_INT PROC NEAR
1306 06DA 50 PUSH AX
1307 06DB 1E PUSH DS
1308 06DD E8 0000 E CALL DOS
1309 06DF C6 06 008E R FF MOV #HF_INT_FLAG,OFFH ; ALL DONE
1310 06E4 B0 20 MOV AL,E01 ; NON-SPECIFIC END OF INTERRUPT
1311 06E6 E6 A0 OUT INTB00,AL ; FOR CONTROLLER #2
1312 06E8 EB 00 JMP $+2 ; WAIT
1313 06EA E6 20 OUT INTA00,AL ; FOR CONTROLLER #1
1314 06EC 1F POP DS
1315 06ED FB STI
1316 06EE B6 9100 MOV AX,9100H ; RE-ENABLE INTERRUPTS
1317 06F1 CD 15 INT 15H ; DEVICE POST
1318 06F3 58 POP AX ; INTERRUPT
1319 06F4 CF IRET ; RETURN FROM INTERRUPT
1320 06F5 HD_INT ENDP
1321 |-----|
1322 06F5 31 31 2F 31 35 2F DB '11/15/85' ; RELEASE MARKER
1323 06FD 38 35
1324 06FD CODE ENDS
1325 06FD END
    
```

SECTION 5

```

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

```

```

115 0013 FE CC          DEC AH          ; CHECK FOR (AH)= 03H
116 0015 74 6C        JZ K300          ; SET TYPAMATIC RATE/DELAY
117 0017 80 EC 02     SUB AH,2        ; CHECK FOR (AH)= 05H
118 001A 75 03        JNZ K101        ;
119 001C E9 00A4 R    JMP K500        ; KEYBOARD WRITE
120 001F 80 EC 0B     SUB AH,11       ; AH = 10
121 0022 74 0C        JZ K1E          ; EXTENDED ASCII READ
122 0024 FE CC        DEC AH          ; CHECK FOR (AH)= 11H
123 0026 74 1A        JZ K2E          ; EXTENDED ASCII STATUS
124 0028 FE CC        DEC AH          ; CHECK FOR (AH)= 2H
125 002A 74 39        JZ K3E          ; EXTENDED_SHIFT_STATUS
126 002C              K10_EXIT:
127 002C 59           POP CX          ; RECOVER REGISTER
128 002D 5B           POP BX          ; RECOVER REGISTER
129 002E 1F           POP DS          ; RECOVER SEGMENT
130 002F CF           IRET          ; INVALID COMMAND
131
132
133
134 0030 E8 00C7 R    K1E: CALL K1S          ; GET A CHARACTER FROM THE BUFFER (EXTENDED)
135 0032 E8 0125 R    CALL K10_E_XLAT ; ROUTINE TO XLATE FOR EXTENDED CALLS
136 0034 EB F4        JMP K10_EXIT    ; GIVE IT TO THE CALLER
137
138 0038 E8 00C7 R    K1:  CALL K1S          ; GET A CHARACTER FROM THE BUFFER
139 003B E8 0130 R    CALL K10_S_XLAT ; ROUTINE TO XLATE FOR STANDARD CALLS
140 003E 72 F8        JC K1           ; CARRY SET MEANS THROW CODE AWAY
141 0040 EB EA        K1A: JMP K10_EXIT    ; RETURN TO CALLER
142
143
144
145 0042 E8 0103 R    K2E: CALL K2S          ; TEST FOR CHARACTER IN BUFFER (EXTENDED)
146 0045 74 18        JZ K2B          ; RETURN IF BUFFER EMPTY
147 0047 9C           PUSHF          ; SAVE ZF FROM TEST
148 0048 E8 0125 R    CALL K10_E_XLAT ; ROUTINE TO XLATE FOR EXTENDED CALLS
149 004B EB 11        JMP SHORT K2A   ; GIVE IT TO THE CALLER
150
151 004D E8 0103 R    K2:  CALL K2S          ; TEST FOR CHARACTER IN BUFFER
152 0050 74 0D        JZ K2B          ; RETURN IF BUFFER EMPTY
153 0052 9C           PUSHF          ; SAVE ZF FROM TEST
154 0053 E8 0130 R    CALL K10_S_XLAT ; ROUTINE TO XLATE FOR STANDARD CALLS
155 0056 73 06        JNC K2A        ; CARRY CLEAR MEANS PASS VALID CODE
156 0058 9D           POPF          ; INVALID CODE FOR THIS TYPE OF CALL
157 0059 E8 00C7 R    CALL K1S        ; THROW THE CHARACTER AWAY
158 005C EB EF        JMP K2          ; GO LOOK FOR NEXT CHAR, IF ANY
159
160 005E 9D           POPF          ; RESTORE ZF FROM TEST
161 005F 59           POP CX          ; RECOVER REGISTER
162 0060 5B           POP BX          ; RECOVER REGISTER
163 0061 1F           POP DS          ; RECOVER SEGMENT
164 0062 CA 0002     RET 2          ; THROW AWAY FLAGS
165
166
167
168 0065              K3E:
169 0065 8A 26 0018 R  MOV AH,#KB_FLAG_1 ; GET THE EXTENDED SHIFT STATUS FLAGS
170 0069 80 E4 04     AND AH,SYS_SHIFT ; MASK ALL BUT SYS KEY BIT
171 006C B1 06        MOV CL,S        ; SHIFT THE SYSTEM KEY BIT OVER TO
172 006E D2 E4        SHL AH,CL       ; BIT 7 POSITION
173 0070 A0 0018 R    MOV AL,#KB_FLAG_1 ; GET SHIFT STATES BACK
174 0073 24 73       AND AL,01110011B ; ELIMINATE SYS_SHIFT, HOLD_STATE, AND INS_SHIFT
175 0075 0A E0       OR AH,AL        ; MERGE THE REMAINING BITS INTO AH
176 0077 A0 0096 R    MOV AL,#KB_FLAG_3 ; GET RIGHT CTL AND ALT
177 007A 24 0C       AND AL,00001100B ; ELIMINATE LC_EO AND LC_EI
178 007C 0A E0       OR AH,AL        ; OR THE SHIFT FLAGS TOGETHER
179 007E A0 0017 R    MOV AL,#KB_FLAG ; GET THE SHIFT STATUS FLAGS
180 0081 EB A9        JMP K3          ; RETURN TO CALLER
181
182
183
184 0083 3C 05        K300: CMP AL,5        ; CORRECT FUNCTION CALL?
185 0085 75 A5        JNE K10_EXIT    ; NO, RETURN
186 0087 F6 C3 E0     TEST BL,0E0h   ; TEST FOR OUT-OF-RANGE RATE
187 008A 75 A0        JNZ K10_EXIT    ; RETURN IF SO
188 008C F6 C7 FC     TEST BH,0Fch   ; TEST FOR OUT-OF-RANGE DELAY
189 008F 75 9B        JNZ K10_EXIT    ; RETURN IF SO
190 0091 B0 F3        MOV AL,KB_TYPA_RD ; COMMAND FOR TYPAMATIC RATE/DELAY
191 0093 E8 064B R    CALL SND_DATA   ; SEND TO KEYBOARD
192 0096 B9 0005     MOV CX,5       ; SHIFT COUNT
193 0099 D2 E7        SHL BH,CL      ; SHIFT DELAY OVER
194 009B 8A C3        MOV AL,BL      ; PUT IN RATE
195 009D 0A C7        OR AL,BH       ; AND DELAY
196 009F E8 064B R    CALL SND_DATA   ; SEND TO KEYBOARD
197 00A2 EB 88        JMP K10_EXIT    ; RETURN TO CALLER
198
199
200
201 00A4 86              K500: PUSH SI          ; SAVE SI
202 00A5 FA          CLJ            ;
203 00A6 8B 1E 001C R  MOV BX,#BUFFER_TAIL ; GET THE "IN TO" POINTER TO THE BUFFER
204 00AA 8B F3        MOV SI,BX      ; SAVE A COPY IN CASE BUFFER NOT FULL
205 00AC E8 0168 R    CALL K4        ; BUMP THE POINTER TO SEE IF BUFFER IS FULL
206 00AF 3B 1E 001A R  CMP BX,#BUFFER_HEAD ; WILL THE BUFFER OVERRUN IF WE STORE THIS?
207 00B3 74 0B        JE K502        ; YES - INFORM CALLER OF ERROR
208 00B5 89 0C        MOV [SI],CX    ; NO - PUT THE ASCII/SCAN CODE INTO BUFFER
209 00B7 89 1E 001C R  MOV #BUFFER_TAIL,BX ; ADJUST "IN TO" POINTER TO REFLECT CHANGE
210 00BB 2A C0        SUB AL,AL      ; TELL CALLER THAT OPERATION WAS SUCCESSFUL
211 00BD EB 03 90     JMP K504        ; SUB INSTRUCTION ALSO RESETS CARRY FLAG
212 00C0
213 00C0 80 01        K502: MOV AL,01H     ; BUFFER FULL INDICATION
214 00C2
215 00C2 FB          K504: STI          ; RECOVER SI
216 00C3 5E          POP SI         ;
217 00C4 E9 002C R    JMP K10_EXIT    ; RETURN TO CALLER WITH STATUS IN AL
218
219 00C7          KEYBOARD_IO_1 ENDP

```

SECTION 5

```

220                                     PAGE
221 |----- READ THE KEY TO FIGURE OUT WHAT TO DO -----
222
223 00C7                                     K15 PROC NEAR
224 00C7 BB 1E 001A R                       MOV BX,#BUFFER_HEAD ; GET POINTER TO HEAD OF BUFFER
225 00CB 3B 1E 001C R                       CMP BX,#BUFFER_TAIL ; TEST END OF BUFFER
226 00CF 75 07                               JNE K1U ; IF ANYTHING IN BUFFER DONT DO INTERRUPT
227
228 00D1 B8 9002                             MOV AX,09002H ; MOVE IN WAIT CODE & TYPE
229 00D4 CD 15                               INT 15H ; PERFORM OTHER FUNCTION
230 00D6                                     K17: ; ASCII READ
231 00D6 FB ; INTERRUPTS BACK ON DURING LOOP
232 00D7 90 ; ALLOW AN INTERRUPT TO OCCUR
233 00D8 FA ; INTERRUPTS BACK OFF
234 00D9 8B 1E 001A R                       MOV BX,#BUFFER_HEAD ; GET POINTER TO HEAD OF BUFFER
235 00DD 3B 1E 001C R                       CMP BX,#BUFFER_TAIL ; TEST END OF BUFFER
236 00E1 53 ; SAVE ADDRESS
237 00E2 9C ; SAVE FLAG
238 00E3 EB 06D8 R                         CALL MAKE_LED ; GO GET MODE INDICATOR DATA BYTE
239 00E6 8A 1E 0097 R                       MOV BL,#MODE_FLAG_2 ; GET PREVIOUS BITS
240 00EA 32 D8 ; SEE IF ANY DIFFERENT
241 00EC 80 E3 07 ; ISOLATE INDICATOR BITS
242 00EF 74 04 ; IF NO CHANGE BYPASS UPDATE
243
244 00F1 EB 069A R                         CALL SND_LED1 ; GO TURN ON MODE INDICATORS
245 00F4 FA ; DISABLE INTERRUPTS
246 00F5 9D ; GET HEAD POINTER
247 00F6 5B ; RESTORE FLAGS
248 00F7 74 DD ; RESTORE ADDRESS
249 ; LOOP UNTIL SOMETHING IN BUFFER
250 00F9 8B 07 ; MOV AX,[BX] ; GET SCAN CODE AND ASCII CODE
251 00FB EB 0168 R                         CALL K4 ; MOVE POINTER TO NEXT POSITION
252 00FE 89 1E 001A R                       MOV #BUFFER_HEAD,BX ; STORE VALUE IN VARIABLE
253 0102 C3 ; RET ; RETURN
254 0103                                     K15 ENDP
255
256 |----- READ THE KEY TO SEE IF ONE IS PRESENT -----
257
258
259 0103                                     K25 PROC NEAR
260 0103 FA ; INTERRUPTS OFF
261 0104 8B 1E 001A R                       MOV BX,#BUFFER_HEAD ; GET HEAD POINTER
262 0108 3B 1E 001C R                       CMP BX,#BUFFER_TAIL ; IF EQUAL (Z=1) THEN NOTHING THERE
263 010C 8B 07 ; MOV AX,[BX] ; SAVE FLAGS
264 010E 9C ; SAVE FLAG
265
266 010F 80 ; PUSH AX ; SAVE CODE
267 0110 EB 06D8 R                         CALL MAKE_LED ; GO GET MODE INDICATOR DATA BYTE
268 0113 8A 1E 0097 R                       MOV BL,#MODE_FLAG_2 ; GET PREVIOUS BITS
269 0117 32 D8 ; XOR BL,AL ; SEE IF ANY DIFFERENT
270 0119 80 E3 07 ; AND BL,07H ; ISOLATE INDICATOR BITS
271 011C 74 03 ; JZ K2T ; IF NO CHANGE BYPASS UPDATE
272
273 011E EB 0687 R                         CALL SND_LED ; GO TURN ON MODE INDICATORS
274 0121 5B ; POP AX ; RESTORE CODE
275 0122 9D ; POPF ; RESTORE FLAGS
276 0123 FB ; STI ; INTERRUPTS BACK ON
277 0124 C3 ; RET ; RETURN
278 0125                                     K25 ENDP
279
280 |----- ROUTINE TO TRANSLATE SCAN CODE PAIRS FOR EXTENDED CALLS -----
281
282
283 0125                                     K10_E_XLAT:
284 0125 3C F0 ; CMP AL,0F0h ; IS IT ONE OF THE FILL-INs?
285 0127 75 06 ; JNE K10_E_RET ; NO, PASS IT ON
286 0129 0A E4 ; OR AH,AH ; AH = 0 IS SPECIAL CASE
287 012B 74 02 ; JZ K10_E_RET ; PASS THIS ON UNCHANGED
288 012D 3C C0 ; XOR AL,AL ; OTHERWISE SET AL = 0
289 012F ; K10_E_RET: ; GO BACK
290 012F C3
291
292 |----- ROUTINE TO TRANSLATE SCAN CODE PAIRS FOR STANDARD CALLS -----
293
294
295 0130                                     K10_S_XLAT:
296 0130 80 FC E0 ; CMP AH,0E0h ; IS IT KEYPAD ENTER OR / ?
297 0133 75 12 ; JNE K10_S2 ; NO, CONTINUE
298 0135 3C 0D ; CMP AL,0Dh ; KEYPAD ENTER CODE?
299 0137 74 09 ; JE K10_S1 ; YES, MESSAGE A BIT
300 0139 3C 0A ; CMP AL,0Ah ; CTRL KEYPAD ENTER CODE?
301 013B 74 08 ; JE K10_S1 ; YES, MESSAGE THE SAME
302 013D 34 B5 ; MOV AH,35h ; NO, MUST BE KEYPAD /
303 013F EB 23 90 ; JMP K10_USE ; GIVE TO CALLER
304 0142 84 1C ; MOV AH,TCh ; CONVERT TO COMPATIBLE OUTPUT
305 0144 EB 1E 90 ; JMP K10_USE ; GIVE TO CALLER
306
307 0147 80 FC 84 ; CMP AH,84h ; IS IT ONE OF THE EXTENDED ONES?
308 014A 77 1A ; JA K10_D15 ; YES, THROW AWAY AND GET ANOTHER CHAR
309
310 014C 3C F0 ; CMP AL,0F0h ; IS IT ONE OF THE FILL-INs?
311 014E 75 07 ; JNE K10_S3 ; NO, TRY LAST TEST
312 0150 0A E4 ; OR AH,AH ; AH = 0 IS SPECIAL CASE
313 0152 74 10 ; JZ K10_USE ; PASS THIS ON UNCHANGED
314 0154 EB 10 90 ; JMP K10_D15 ; THROW AWAY THE REST
315
316 0157 3C E0 ; CMP AL,0E0h ; IS IT AN EXTENSION OF A PREVIOUS ONE?
317 0159 75 09 ; JNE K10_USE ; NO, MUST BE A STANDARD CODE
318 015B 0A E4 ; OR AH,AH ; AH = 0 IS SPECIAL CASE
319 015D 74 05 ; JZ K10_USE ; JUMP IF AH = 0
320 015F 3C C0 ; XOR AL,AL ; CONVERT TO COMPATIBLE OUTPUT
321 0161 EB 01 90 ; JMP K10_USE ; PASS IT ON TO CALLER
322
323 0164                                     K10_USE:
324 0164 F8 ; CLC ; CLEAR CARRY TO INDICATE GOOD CODE
325 0165 C3 ; RET ; RETURN
326 0166
327 0166 F9 ; STC ; SET CARRY TO INDICATE DISCARD CODE
328 0167 C3 ; RET ; RETURN

```

```

329                                     PAGE
330                                     ;-----
331                                     ; INCREMENT BUFFER POINTER ROUTINE
332                                     ;-----
333
334 0168 K4 PROC NEAR
335 0168 43 INC BX ; MOVE TO NEXT WORD IN LIST
336 0169 43 INC BX
337
338 016A 3B 1E 0082 R CMP BX,#BUFFER_END ; AT END OF BUFFER?
339 016E 75 04 JNE KB ; NO, CONTINUE
340 0170 8B 1E 0080 R MOV BX,#BUFFER_START ; YES, RESET TO BUFFER BEGINNING
341 0174 C3 RET
342 0175 K4 ENDP
343
344                                     ;--- HARDWARE INT 09 H --- ( IRQ LEVEL 1 )
345                                     ;
346                                     ; KEYBOARD INTERRUPT ROUTINE
347                                     ;
348                                     ;-----
349
350 0175 KB_INT_1 PROC FAR
351 0176 FB STI ; ENABLE INTERRUPTS
352 017A 55 PUSH BP
353 0177 50 PUSH AX
354 0178 53 PUSH BX
355 0179 51 PUSH CX
356 017A 52 PUSH DX
357 017B 56 PUSH SI
358 017C 57 PUSH DI
359 017D 1E PUSH DS
360 017E 06 PUSH ES
361 017F FC CLD ; FORWARD DIRECTION
362 0180 E8 0000 E CALL DDS ; SET UP ADDRESSING
363
364 ;----- WAIT FOR KEYBOARD DISABLE COMMAND TO BE ACCEPTED
365
366 0183 80 AD MOV AL,DIS_KBD ; DISABLE THE KEYBOARD COMMAND
367 0185 E8 069C R CALL SHIP_IT ; EXECUTE DISABLE
368 0188 FA CLI ; DISABLE INTERRUPTS
369 0189 2B C9 SUB CX,CX ; SET MAXIMUM TIMEOUT
370 018B KB_INT_01:
371 018B E4 64 IN AL,STATUS_PORT ; READ ADAPTER STATUS
372 018D A8 02 TEST AL,INPT_BUF_FULL ; CHECK INPUT BUFFER FULL STATUS BIT
373 018F E0 FA LOOPNZ KB_INT_01 ; WAIT FOR COMMAND TO BE ACCEPTED
374
375 ;----- READ CHARACTER FROM KEYBOARD INTERFACE
376
377 0191 E4 60 IN AL,PORT_A ; READ IN THE CHARACTER
378
379 ;----- SYSTEM HOOK INT 15H - FUNCTION 4FH (ON HARDWARE INTERRUPT LEVEL 9H)
380
381 0193 B4 4F MOV AH,04FH ; SYSTEM INTERCEPT - KEY CODE FUNCTION
382 0195 F9 STC ; SET CY= 1 (IN CASE OF IRET)
383 0196 CD 15 INT 15H ; CASSETTE CALL, (AL)= KEY SCAN CODE
384 ; RETURNS CY= 1 FOR INVALID FUNCTION
385 0198 72 03 JC KB_INT_02 ; CONTINUE IF CARRY FLAG SET ((AL)=CODE)
386 019A E9 03A0 R JMP K26 ; EXIT IF SYSTEM HANDLED SCAN CODE
387 ; EXIT HANDLES HARDWARE EOI AND ENABLE
388
389 ;----- CHECK FOR A RESEND COMMAND TO KEYBOARD
390
391 019D KB_INT_02:
392 019D FB STI ; (AL)= SCAN CODE
393 019E 3C FE CMP AL,KB_RESEND ; ENABLE INTERRUPTS AGAIN
394 01A0 74 0D JE KB_INT_4 ; IS THE INPUT A RESEND
395 ; GO IF RESEND
396
397 ;----- CHECK FOR RESPONSE TO A COMMAND TO KEYBOARD
398
399 01A2 3C FA CMP AL,KB_ACK ; IS THE INPUT AN ACKNOWLEDGE
400 01A4 75 12 JNZ KB_INT_2 ; GO IF NOT
401
402 ;----- A COMMAND TO THE KEYBOARD WAS ISSUED
403
404 01A6 FA CLI ; DISABLE INTERRUPTS
405 01A7 80 0E 0097 R 10 OR #KB_FLAG_2,KB_FA ; INDICATE ACK RECEIVED
406 01AC E9 03A0 R JMP K26 ; RETURN IF NOT (ACK RETURNED FOR DATA)
407
408 ;----- RESEND THE LAST BYTE
409
410 01AF KB_INT_4:
411 01AF FA CLI ; DISABLE INTERRUPTS
412 01B0 80 0E 0097 R 20 OR #KB_FLAG_2,KB_FE ; INDICATE RESEND RECEIVED
413 01B5 E9 03A0 R JMP K26 ; RETURN IF NOT (ACK RETURNED FOR DATA)
414
415 ;----- UPDATE MODE INDICATORS IF CHANGE IN STATE
416
417 01B8 KB_INT_2:
418 01B8 50 PUSH AX ; SAVE DATA IN
419 01B9 E8 06D8 R CALL MAKE_LED ; GO GET MODE INDICATOR DATA BYTE
420 01BC 8A 1E 0097 R MOV BL,#KB_FLAG_2 ; GET PREVIOUS BITS
421 01C0 32 D5 XOR BL,AL ; SEE IF ANY DIFFERENT
422 01C2 80 E3 07 AND BL,KB_LEDS ; ISOLATE INDICATOR BITS
423 01C5 74 03 JZ UP0 ; IF NO CHANGE BYPASS UPDATE
424 01C7 E8 06B7 R CALL SHD_LED ; GO TURN ON MODE INDICATORS
425 01CA 58 POP AX ; RESTORE DATA IN

```

SECTION 5

```

426                                     PAGE
427                                     |-----|
428                                     | START OF KEY PROCESSING |-----|
429                                     |-----|
430
431 01CB 8A E0                          MOY    AH,AL          ; SAVE SCAN CODE IN AH ALSO
432
433 |----- TEST FOR OVERRUN SCAN CODE FROM KEYBOARD
434
435 01CD 3C FF                          CMP     AL,KB_OVER_RUN ; IS THIS AN OVERRUN CHAR?
436 01CF 75 03                          JNZ    K16           ; NO, TEST FOR SHIFT KEY
437 01D1 E9 062D R                      JMP     K62          ; BUFFER_FULL_BEEP
438
439 01D4 0E                              K16:   PUSH   CS
440 01D5 07                              POP    ES           ; ESTABLISH ADDRESS OF TABLES
441 01D6 8A 3E 0096 R                  MOY    BH,*KB_FLAG_3 ; LOAD FLAGS FOR TESTING
442
443 |----- TEST TO SEE IF A READ_ID IS IN PROGRESS
444
445 01DA F6 C7 C0                      TEST   BH,RD_ID+LC_AB ; ARE WE DOING A READ ID?
446 01DD 74 34                          JZ     NOT_ID       ; CONTINUE IF NOT
447 01DF 79 10                          JNS   TST_ID_2     ; IS THE RD_ID FLAG ON?
448 01E1 3C AB                          CMP    AL,ID_1     ; IS THIS THE 1ST ID CHARACTER?
449 01E3 75 0687 R                    JNE   RST_RD_ID   ;
450 01E5 80 0E 0096 R 40              OR     *KB_FLAG_3,LC_AB ; INDICATE 1ST ID WAS OK
451 01EA                                RST_RD_ID:
452 01EA 80 26 0096 R 7F              AND    *KB_FLAG_3,NOT_RD_ID ; RESET THE READ ID FLAG
453 01EF EB 1F                          JMP    SHORT_ID_EX ; AND EXIT
454
455 01F1                                TST_ID_2:
456 01F1 80 26 0096 R BF              AND    *KB_FLAG_3,NOT_LC_AB ; RESET FLAG
457 01F6 3C 54                          CMP    AL,ID_2A    ; IS THIS THE 2ND ID CHARACTER?
458 01F8 74 11                          JE     KX_BIT      ; JUMP IF SO
459 01FA 3C 41                          CMP    AL,ID_2     ; IS THIS THE 2ND ID CHARACTER?
460 01FC 75 12                          JNE   ID_EX       ; LEAVE IF NOT
461
462 |----- A READ ID SAID THAT IT WAS ENHANCED KEYBOARD
463
464 01FE F6 C7 20                      TEST   BH,SET_NUM_LK ; SHOULD WE SET NUM LOCK?
465 0201 74 08                          JZ     KX_BIT      ; EXIT IF NOT
466 0203 80 0E 0017 R 20              OR     *KB_FLAG_NUM_STATE ; FORCE NUM LOCK ON
467 0205 80 0E 0017 R 20              CALL  SMO_LED     ; GO SET THE NUM LOCK INDICATOR
468 020B 80 0E 0096 R 10              KX_BIT: OR *KB_FLAG_3,KBX ; INDICATE ENHANCED KEYBOARD WAS FOUND
469 0210 E9 03A0 R                    JMP    K26        ; EXIT
470
471 0213                                NOT_ID:
472 0213 3C E0                          CMP    AL,MC_E0    ; IS THIS THE GENERAL MARKER CODE?
473 0215 75 07                          JNE   TEST_E1     ;
474 0217 80 0E 0096 R 12              OR     *KB_FLAG_3,LC_E0+KBX ; SET FLAG BIT, SET KBX, AND
475 021C EB 09                          JMP    SHORT_EXIT  ; THROW AWAY THIS CODE
476
477 021E                                TEST_E1:
478 021E 3C E1                          CMP    AL,MC_E1    ; IS THIS THE PAUSE KEY?
479 0220 75 08                          JNE   NOT_HC      ;
480 0222 80 0E 0096 R 11              OR     *KB_FLAG_3,LC_E1+KBX ; SET FLAG, PAUSE KEY MARKER CODE
481 0227 E9 03A0 R                    JMP    K26A       ; THROW AWAY THIS CODE
482
483 022A                                NOT_HC:
484 022A 24 7F                          AND    AL,07FH    ; TURN OFF THE BREAK BIT
485 022C 76 C7 02                      TEST   BH,LC_E0   ; LAST CODE: THE E0 MARKER CODE?
486 022F 74 0C                          JZ     NOT_LC_E0  ; JUMP IF NOT
487
488 0231 89 0002                        MOV    CX,2       ; LENGTH OF SEARCH
489 0234 BF 0006 E                    MOV    DI,OFFSET K6+6 ; IS THIS A SHIFT KEY?
490 0237 F2/ AE                       REPNE SCASB      ; CHECK IT
491 0239 75 65                          JNE   K16A       ; NO, CONTINUE KEY PROCESSING
492 023B EB 49                          JMP    SHORT_K16B ; YES, THROW AWAY & RESET FLAG
493
494 023D                                NOT_LC_E0:
495 023D 76 C7 01                      TEST   BH,LC_E1   ; LAST CODE: THE E1 MARKER CODE?
496 0240 74 1D                          JZ     T_SYS_KEY  ; JUMP IF NOT
497
498 0242 89 0004                        MOV    CX,4       ; LENGTH OF SEARCH
499 0245 BF 0004 E                    MOV    DI,OFFSET K6+4 ; IS THIS AN ALT, CTL, OR SHIFT?
500 0248 F2/ AE                       REPNE SCASB      ; CHECK IT
501 024A 74 DB                          JE     EXIT       ; THROW AWAY IF SO
502
503 024C 3C 45                          CMP    AL,NUM_KEY ; IS IT THE PAUSE KEY?
504 024E 75 36                          JNE   K16B       ; NO, THROW AWAY & RESET FLAG
505 0250 F6 C4 80                      TEST   AH,80H    ; YES, IS IT THE BREAK OF THE KEY?
506 0253 75 31                          JNZ   K16B       ; YES, THROW THIS AWAY, TOO
507 0255 F6 06 0018 R 08              TEST  *KB_FLAG_1,HOLD_STATE ; NO, ARE WE PAUSED ALREADY?
508 025A 75 2A                          JNZ   K16B       ; YES, THROW AWAY
509 025C E9 04DB R                    JMP    K39P      ; NO, THIS IS THE REAL PAUSE STATE
510
511 |----- TEST FOR SYSTEM KEY
512
513 025F                                T_SYS_KEY:
514 025F 3C 54                          CMP    AL,SYS_KEY ; IS IT THE SYSTEM KEY?
515 0261 75 3D                          JNE   K16A       ; CONTINUE IF NOT
516
517 0263 F6 C4 80                      TEST   AH,080H   ; CHECK IF THIS A BREAK CODE
518 0266 75 21                          JNZ   K16C       ; DON'T TOUCH SYSTEM INDICATOR IF TRUE
519
520 0268 F6 06 0018 R 04              TEST  *KB_FLAG_1,SYS_SHIFT ; SEE IF IN SYSTEM KEY HELD DOWN
521 026D 75 17                          JZ     K16B      ; IF YES, DON'T PROCESS SYSTEM INDICATOR
522
523 026F 80 0E 0018 R 04              OR     *KB_FLAG_1,SYS_SHIFT ; INDICATE SYSTEM KEY DEPRESSED
524 0274 80 20                          MOV    AL,E01    ; END OF INTERRUPT COMMAND
525 0276 E6 20                          OUT    020H,AL   ; SEND COMMAND TO INTERRUPT CONTROL PORT
526
527 0278 80 AE                          MOV    AL,ENA_KBD ; INSURE KEYBOARD IS ENABLED
528 027A EB 083C R                    CALL  SHIP_IT    ; EXECUTE ENABLE
529 027D 88 8500                      MOV    AX,08500H ; FUNCTION VALUE FOR MAKE OF SYSTEM KEY
530 0280 FB                          STI                                ; MAKE SURE INTERRUPTS ENABLED
531 0281 CD 15                          INT    15H      ; USER INTERRUPT
532 0283 E9 03AF R                    JMP    K27A     ; END PROCESSING
533
534 0286 E9 03A0 R                    K16B:   JMP    K26      ; IGNORE SYSTEM KEY
535
536 0289 80 26 0018 R FB              K16C:   AND    *KB_FLAG_1,NOT_SYS_SHIFT ; TURN OFF SHIFT KEY HELD DOWN
537 028E 80 20                          AND    AL,E01    ; END OF INTERRUPT COMMAND
538 0290 E6 20                          OUT    020H,AL   ; SEND COMMAND TO INTERRUPT CONTROL PORT

```

```

540                                MOV     AL,ENA_KBD           | INTERRUPT-RETURN-NO-EOI
541                                CALL    SHIP_IT           | INSURE KEYBOARD IS ENABLED
542 0294 B0 AE                      MOV     AX,08501H        | EXECUTE ENABLE
543 0297 B8 8501                    STI     AX,08501H        | FUNCTION VALUE FOR BREAK OF SYSTEM KEY
544 029A FB                          INT     $H              | MAKE SURE INTERRUPTS ENABLED
545 029B CD 15                       JMP     K27A            | USER INTERRUPT
546 029D E9 03AF R                   JMP     K27A            | IGNORE SYSTEM KEY
547
548                                ;----- TEST FOR SHIFT KEYS
549
550 02A0 8A 1E 0017 R                K16A: MOV     BL,*KB_FLAG           | PUT STATE FLAGS IN BL
551 02A4 BF 0000 E                   MOV     DI,OFFSET K6     | SHIFT KEY TABLE
552 02A7 B9 0000 E                   MOV     CX,OFFSET K&L   | LENGTH
553 02AA F2/ AE                      REPNE   SCASB           | LOOK THROUGH THE TABLE FOR A MATCH
554 02AC 8A C4                       MOV     AL,AH           | RECOVER SCAN CODE
555 02AE 74 03                       JNE     K17             | JUMP IF MATCH FOUND
556 02B0 E9 03BC R                   JMP     K25             | IF NO MATCH, THEN SHIFT NOT FOUND
557
558                                ;----- SHIFT KEY FOUND
559
560 02B3 81 EF 0001 E                K17: SUB     DI,OFFSET K6+1       | ADJUST PTR TO SCAN CODE MTCH
561 02B7 2E1 BA AS 0000 E           MOV     AH,CS:[K7[DI]]  | GET MASK INTO AH
562 02BC B1 02                       MOV     CL,2            | SET UP COUNT FOR FLAG SHIFTS
563 02BE A8 80                       TEST    AL,B0H         | TEST FOR BREAK KEY
564 02C0 74 03                       JZ     K17C            | K17C
565 02C2 EB 78 90                   JMP     K23            | JUMP IF BREAK
566
567                                ;----- SHIFT MAKE FOUND, DETERMINE SET OR TOGGLE
568
569 02C5 80 FC 10                   K17C: CMP     AH,SCROLL_SHIFT | IF SCROLL SHIFT OR ABOVE, TOGGLE KEY
570 02C8 73 21                       JAE     K18            |
571
572                                ;----- PLAIN SHIFT KEY, SET SHIFT ON
573
574 02CA 08 26 0017 R                OR     *KB_FLAG,AH     | TURN ON SHIFT BIT
575 02CE F6 C4 0C                   TEST    AH,CTL_SHIFT+ALT_SHIFT | IS IT ALT OR CTRL?
576 02D1 75 03                       JNZ    K17D           | YES, MORE FLAGS TO SET
577 02D3 E9 03A0 R                   JMP     K26            | NO, INTERRUPT_RETURN
578 02D6 F6 C7 02                   K17D: TEST    BH,LC_E0         | IS THIS ONE OF THE NEW KEYS?
579 02D9 74 07                       JZ     K17E           | NO, JUMP
580 02DB 08 26 0096 R               OR     *KB_FLAG_3,AH   | SET BITS FOR RIGHT CTRL, ALT
581 02DF E9 03A0 R                   JMP     K26            | INTERRUPT_RETURN
582 02E2 D2 EC                       SHR     AH,CL          | MOVE FLAG BITS TWO POSITIONS
583 02E4 08 26 0018 R               OR     *KB_FLAG_1,AH   | SET BITS FOR LEFT CTRL, ALT
584 02E8 E9 03A0 R                   JMP     K26            | INTERRUPT_RETURN
585
586                                ;----- TOGGLED SHIFT KEY, TEST FOR 1ST MAKE OR NOT
587
588 02EB F6 C3 04                   K18: TEST    BL,CTL_SHIFT     | SHIFT-TOGGLE
589 02EE 74 03                       JZ     K18A           | CHECK CTL SHIFT STATE
590 02F0 E9 03BC R                   JMP     K25            | JUMP IF NOT CTL STATE
591 02F3 3C 52                       K18A: CMP     AL,INS_KEY    | CHECK FOR INSERT KEY
592 02F5 75 21                       JNE    K22            | JUMP IF NOT INSERT KEY
593 02F7 F6 C3 08                   TEST    BL,ALT_SHIFT   | CHECK FOR ALTERNATE SHIFT
594 02FA 74 03                       JZ     K18B           | JUMP IF NOT ALTERNATE SHIFT
595 02FC E9 03BC R                   JMP     K25            | JUMP IF ALTERNATE SHIFT
596 02FF F6 C7 02                   K18B: TEST    BH,LC_E0         | IS THIS THE NEW INSERT KEY?
597 0302 75 14                       JNZ    K22            | YES, THIS ONE'S NEVER A "0"
598 0304 F6 C3 20                   K19: TEST    BL,NUM_STATE  | CHECK FOR BASE STATE
599 0307 75 0A                       JNZ    K21            | JUMP IF NUM LOCK IS ON
600 0309 F6 C3 03                   K19: TEST    BL,LEFT_SHIFT+RIGHT_SHIFT | TEST FOR SHIFT STATE
601 030C 74 0A                       JZ     K22            | JUMP IF BASE STATE
602 030E 8A EA 90                   K20: MOV     AH,AL      | PUT SCAN CODE BACK IN AH
603 0310 EB 7A 90                   JMP     K25            | NUMERAL "0", STNRD. PROCESSING
604
605                                K21: TEST    BL,LEFT_SHIFT+RIGHT_SHIFT | MIGHT BE NUMERIC
606 0313 F6 C3 03                   ; IS NUMERIC, STD. PROC.
607 0316 74 F6                       JZ     K22            |
608
609 0318                                K22: TEST    AH,*KB_FLAG_1 | SHIFT TOGGLE KEY HIT? PROCESS IT
610 031B 84 26 0018 R               ; IS KEY ALREADY DEPRESSED?
611 031E 74 03                       JZ     K26            |
612 0320 E9 03A0 R                   JMP     K26            | JUMP IF KEY ALREADY DEPRESSED
613 0323 08 26 0018 R               K22A: OR     *KB_FLAG_1,AH | INDICATE THAT THE KEY IS DEPRESSED
614 0325 30 26 0017 R               XOR     *KB_FLAG,AH    | TOGGLE THE SHIFT STATE
615
616                                ;----- TOGGLE LED IF CAPS, NUM, OR SCROLL KEY DEPRESSED
617
618 0329 F6 C4 70                   TEST    AH,CAPS_SHIFT+NUM_SHIFT+SCROLL_SHIFT | SHIFT TOGGLE?
619 032C 74 05                       JZ     K22B           | GO IF NOT
620 032E 50                          PUSH   AX              | SAVE SCAN CODE AND SHIFT MASK
621 032F E8 0687 R                   CALL    SNO_LED        | GO TURN MODE INDICATORS ON
622 0332 58                          POP     AX             | RESTORE SCAN CODE
623
624 0333 3C 52                   K22B: CMP     AL,INS_KEY    | TEST FOR 1ST MAKE OF INSERT KEY
625 0335 75 69                   JNE    K26            | JUMP IF NOT INSERT KEY
626 0337 8A E0                   MOV     AH,AL          | SCAN CODE IN BOTH HALVES OF AX
627 0339 EB 7F 90                   JMP     K26            | FLAGS UPDATED, PROC. FOR BUFFER
628
629                                ;----- BREAK SHIFT FOUND
630
631 033C                                K23:                                | BREAK-SHIFT-FOUND
632 033C 80 FC 10                   CMP     AH,SCROLL_SHIFT | IS THIS A TOGGLE KEY?
633 033F F6 04                   NOT     AH              | INVERT MASK
634 0341 73 43                   JAE     K24            | YES, HANDLE BREAK TOGGLE
635 0343 20 26 0017 R               AND     *KB_FLAG,AH    | TURN OFF SHIFT BIT
636 0347 80 FC FB                   CMP     AH,NOT_CTL_SHIFT | IS THIS ALT OR CTL?
637 034A 77 26                       JA     K23B           | NO, ALL DONE
638
639 034C F6 C7 02                   TEST    BH,LC_E0       | 2ND ALT OR CTL?
640 034F 74 06                       JZ     K23A           | NO, HANDLE NORMALLY
641 0351 20 26 0096 R               AND     *KB_FLAG_3,AH  | RESET BIT FOR RIGHT ALT OR CTL
642 0355 EB 06                       JMP     SHORT K23B     | CONTINUE
643 0357 D2 FC                   SAR     AH,CL          | MOVE THE MASK BIT TWO POSITIONS
644 0359 20 26 0018 R               AND     *KB_FLAG_1,AH  | RESET BIT FOR LEFT ALT OR CTL
645 035D 8A E0                   MOV     AH,AL          | SAVE SCAN CODE
646 035F A0 0096 R                   MOV     AL,*KB_FLAG_3  | GET RIGHT ALT & CTRL FLAGS
647 0362 D2 EB                   SHR     AL,CL          | MOVE TO BITS 1 & 0
648 0364 0A 06 0018 R               OR     AL,*KB_FLAG_1   | PUT IN LEFT ALT & CTL FLAGS
649 0366 D2 E0                   SHL     AL,CL          | MOVE BACK TO BITS 3 & 2
650 036A 24 0C                   AND     AL,ALT_SHIFT+CTL_SHIFT | FILTER OUT OTHER GARBAGE
651 036C 08 06 0017 R               OR     *KB_FLAG,AL    | PUT RESULT IN THE REAL FLAGS
652 0370 8A C4                       MOV     AL,AH          | RECOVER SAVED SCAN CODE
653

```

SECTION 5

```

654 0372 3C B8      K23D:  CMP     AL,ALT_KEY+80H      ; IS THIS ALTERNATE SHIFT RELEASE
655 0374 75 2A      JNE     K26      ; INTERRUPT_RETURN
656
657 ;----- ALTERNATE SHIFT KEY RELEASED, GET THE VALUE INTO BUFFER
658
659 0376 A0 0019 R   MOV     AL,@ALT_INPUT
660 0379 B4 00      MOV     AH,0      ; SCAN CODE OF 0
661 037B 88 26 0019 R MOV     @ALT_INPUT,AH ; ZERO OUT THE FIELD
662 037F 3C 00      CMP     AL,0      ; WAS THE INPUT = 0?
663 0381 74 1D      JE     K26      ; INTERRUPT_RETURN
664 0383 E9 0601 R   JMP     K61      ; IT WASN'T, SO PUT IN BUFFER
665
666 0386             ; BREAK-TOGGLE
667 0386 20 26 0018 R K24:   AND     @KB_FLAG_1,AH ; INDICATE NO LONGER DEPRESSED
668 038A EB 14      JMP     SHORT K26 ; INTERRUPT_RETURN
669
670 ;----- TEST FOR HOLD STATE
671
672 038C             ; AL, AH = SCAN CODE
673 038C 3C 80      K25:   CMP     AL,80H    ; NO-SHIFT-FOUND
674 038E 73 10      JAE     K26      ; TEST FOR BREAK KEY
675 0390 F6 06 0018 R 08 TEST   @KB_FLAG_1,HOLD_STATE ; NOTHING FOR BREAK CHARS FROM HERE ON
676 0398 74 23      JZ     K28      ; ARE WE IN HOLD STATE
677 0397 3C 45      CMP     AL,NUM_KEY ; BRANCH AROUND TEST IF NOT
678 0399 74 05      JZ     K26      ; CAN'T END HOLD ON NUM LOCK
679 039B 80 26 0018 R F7 AND     @KB_FLAG_1,NOT HOLD_STATE ; TURN OFF THE HOLD STATE BIT
680
681 03A0             ; RESET LAST CHAR H.C. FLAG
682 03A0 80 26 0096 R FC K26:   AND     @KB_FLAG_3,NOT LC_E0+LC_E1
683
684 03A5             ; INTERRUPT-RETURN
685 03A5 FA          K26A:  CLI     ; TURN OFF INTERRUPTS
686 03A6 B0 20      MOV     AL,E01    ; END OF INTERRUPT COMMAND
687 03A8 E6 20      OUT    020H,AL   ; SEND COMMAND TO INTERRUPT CONTROL PORT
688
689 03AA             ; INTERRUPT-RETURN-NO-E01
690 03AA B0 AE      K27:   MOV     AL,ENA_KBD ; INSURE KEYBOARD IS ENABLED
691 03AC E8 063C R   CALL   SHIP_IT   ; EXECUTE ENABLE
692
693 03AF FA          K27A:  CLI     ; DISABLE INTERRUPTS
694 03B0 07          ; RESTORE REGISTERS
695 03B1 1F          POP     ES
696 03B2 5F          POP     DS
697 03B3 5E          POP     DI
698 03B4 5A          POP     SI
699 03B5 59          POP     DX
700 03B6 58          POP     CX
701 03B7 5B          POP     BX
702 03B8 5D          POP     AX
703 03B9 CF          POP     BP
704 03B9 CF          IRET      ; RETURN

```

```

704                                     PAGE
705 |----- NOT IN HOLD STATE
706
707 03BA                                     K28:  CMP AL,88          ; AL, AH = SCAN CODE (ALL MAKES)
708 03BA 3C 58                             ; NO-HOLD-STATE
709 03BC 77 E2                             ; TEST FOR OUT-OF-RANGE SCAN CODES
710                                     ; IGNORE IF OUT-OF-RANGE
711 03BE F6 C3 08                         ; ARE WE IN ALTERNATE SHIFT?
712 03C1 74 0C                             ; JUMP IF NOT ALTERNATE
713
714 03C3 F6 C7 10                         ; IS THIS THE ENHANCED KEYBOARD?
715 03C5 74 0A                             ; NO, ALT STATE IS REAL
716
717 03C8 F6 06 0018 R 04                  ; YES, IS SYSREQ KEY DOWN?
718 03CD 74 03                             ; NO, ALT STATE IS REAL
719 03CF E9 04A3 R                        ; YES, THIS IS PHONY ALT STATE
720                                     ; DUE TO PRESSING SYSREQ
721
722 |----- TEST FOR RESET KEY SEQUENCE (CTL ALT DEL)
723
724 03D2                                     K29:  TEST BL,CTL_SHIFT ; TEST-RESET
725 03D2 F6 C3 04                         ; ARE WE IN CONTROL SHIFT ALSO?
726 03D6 74 81                             ; NO-RESET
727 03D7 3C 53                             ; TEST AL,DEL_KEY
728 03D9 75 2D                             ; SHIFT STATE IS THERE, TEST KEY
729                                     ; NO-RESET, IGNORE
730
731 |----- CTL-ALT-DEL HAS BEEN FOUND, DO I/O CLEANUP
732 03DB C7 06 0072 R 1234                ; MOV 0RESET_FLAG,1234H ; SET FLAG FOR RESET FUNCTION
733 03E1 E9 0000 E                          ; JMP START_T           ; JUMP TO POWER ON DIAGNOSTICS
734
735 |----- TABLES FOR ALT CASE -----
736 |----- ALT-INPUT-TABLE
737 03E4                                     K30:  LABEL BYTE
738 03E4 62 4F 50 51 48                   ; DB 82,79,80,81,75
739 03E9 4C 4D 47 48 49                   ; DB 76,77,71,72,73 ; 10 NUMBERS ON KEYPAD
740
741 |----- SUPER-SHIFT-TABLE
742 03EE 10 11 12 13 14 15                ; DB 16,17,18,19,20,21 ; A-Z TYPEWRITER CHARS
743 03FA 16 17 18 19 1E 1F                ; DB 22,23,24,25,30,31
744 0400 26 2C 2D 2E 2F 30                ; DB 32,33,34,35,36,37
745 0406 31 32                             ; DB 38,44,45,46,47,48
746                                     ; DB 49,50
747
748 |----- IN ALTERNATE SHIFT, RESET NOT FOUND
749
750 0408                                     K31:  CMP AL,57          ; NO-RESET
751 040A 3C 39                             ; TEST FOR SPACE KEY
752 040C 80 20                             ; JNE K311             ; NOT THERE
753 040E E9 05F5 R                          ; MOV AL,' '          ; SET SPACE CHAR
754 0411                                     ; JMP K57              ; BUFFER_FILL
755
756 0411 3C 0F                             ; TEST AL,15          ; TEST FOR TAB KEY
757 0413 75 06                             ; JNE K312            ; NOT THERE
758 0415 B8 A500                            ; MOV AX,0A500h       ; SET SPECIAL CODE FOR ALT-TAB
759 041B                                     ; JMP K57              ; BUFFER_FILL
760
761 041B 3C 4A                             ; TEST AL,74          ; TEST FOR KEYPAD -
762 041D 74 79                             ; JE K37B             ; GO PROCESS
763 041F 3C 4E                             ; CMP AL,78           ; TEST FOR KEYPAD +
764 0421 74 75                             ; JE K37B             ; GO PROCESS
765
766 |----- LOOK FOR KEY PAD ENTRY
767
768 0423                                     K32:  MOV D1,OFFSET K30 ; ALT-KEY-PAD
769 0426 B9 000A                            ; MOV CX,10           ; ALT-INPUT-TABLE
770 0429 F2 AE                             ; REPNE SCASB         ; LOOK FOR ENTRY USING KEYPAD
771 042B 75 1A                             ; JNE K33             ; LOOK FOR MATCH
772 042D F6 C7 02                         ; TEST BH,LC_E0       ; NO ALT KEYPAD
773 0430 75 6B                             ; JNZ K37C            ; IS THIS ONE OF THE NEW KEYS?
774 0432 81 EF 03E5 R                      ; SUB DI,OFFSET K30+1 ; YES, JUMP, NOT NUMPAD KEY
775 0436 A0 0019 R                          ; MOV AL,0AL_INPUT    ; DI NOW HAS ENTRY VALUE
776 0439 B4 0A                             ; MOV AH,10           ; GET THE CURRENT BYTE
777 043B F6 E4                             ; MULL AH             ; MULTIPLY BY 10
778 043D 03 C7                             ; ADD AX,D1           ; ADD IN THE LATEST ENTRY
779 043F A2 0019 R                          ; MOV 0AL_INPUT,AL   ; STORE IT AWAY
780 0442 E9 03A0 R                          ; JMP K26             ; THROW AWAY THAT KEYSTROKE
781
782 |----- LOOK FOR SUPERSHIFT ENTRY
783
784 0445                                     K33:  MOV 0AL_INPUT,0 ; NO-ALT-KEYPAD
785 0448 C6 06 0019 R 00                    ; MOV CX,25           ; ZERO ANY PREVIOUS ENTRY INTO INPUT
786 044A B9 001A                            ; REPNE SCASB         ; DI,ES ALREADY POINTING
787 044D F2 AE                             ; LOOK FOR MATCH IN ALPHABET
788 044F 74 42                             ; JE K37A             ; MATCH FOUND, GO FILL THE BUFFER
789
790 |----- LOOK FOR TOP ROW OF ALTERNATE SHIFT
791
792 0451                                     K34:  CMP AL,2          ; ALT-TOP-ROW
793 0451 3C 02                             ; JB K37B             ; KEY WITH '!' ON IT
794 0453 72 43                             ; CMP AL,13           ; MUST BE ESCAPE
795 0456 3C 0D                             ; JA K35             ; IS IT IN THE REGION
796 0457 77 05                             ; JNC K35            ; NO, ALT-SOMETHING ELSE
797 0459 80 C4 76                          ; ADD AH,118         ; CONVERT PSEUDO SCAN CODE TO RANGE
798 045C EB 35                             ; JMP SHORT K37A     ; GO FILL THE BUFFER
799
800 |----- TRANSLATE ALTERNATE SHIFT PSEUDO SCAN CODES
801
802 045E                                     K35:  CMP AL,F11_M       ; ALT-FUNCTION
803 045E 3C 57                             ; JB K35A            ; IS IT F11?
804 0460 72 09                             ; CMP AL,F12_M       ; NO, BRANCH
805 0462 3C 58                             ; JA K35A            ; IS IT F12?
806 0464 77 05                             ; JNC K35A           ; NO, BRANCH
807 0466 80 C4 34                          ; ADD AH,52          ; CONVERT TO PSEUDO SCAN CODE
808 0469 EB 28                             ; JMP SHORT K37A     ; GO FILL THE BUFFER
809
810 046B F6 C7 02                         ; TEST BH,LC_E0       ; DO WE HAVE ONE OF THE NEW KEYS?
811 046E 74 18                             ; JB K37              ; NO, JUMP
812 0470 3C 1C                             ; CMP AL,28           ; TEST FOR KEYPAD ENTER
813 0472 75 06                             ; JNE K35B           ; NOT THERE
814 0474 B8 A500                            ; MOV AX,0A500h       ; SPECIAL CODE
815 0477 E9 05F5 R                          ; JMP K57              ; BUFFER_FILL
816 047A 3C 53                             ; CMP AL,83           ; TEST FOR DELETE KEY
817 047C 74 1F                             ; JE K37C            ; HANDLE WITH OTHER EDIT KEYS

```

SECTION 5

```

818 047E 3C 25          CMP     AL,53          ; TEST FOR KEYPAD /
819 0480 75 C0          JNE     K32A          ; NOT THERE, NO OTHER E0 SPECIALS
820 0482 B8 A400        MOV     AX,0A400h     ; SPECIAL CODE
821 0485 E9 05F5 R      JMP     K57           ; BUFFER FILL
822
823 0488 3C 3B          CMP     AL,59          ; TEST FOR FUNCTION KEYS (F1)
824 048A 72 0C          JB      K37B          ; NO FN, HANDLE W/OTHER EXTENDED
825 048C 3C 44          CMP     AL,68          ; IN KEYPAD REGION?
826                                     ; OR NUMLOCK, SCROLLLOCK?
827 048E 77 B2          JA      K32A          ; IF SO, IGNORE
828 0490 80 C4 2D        ADD     AH,45          ; CONVERT TO PSEUDO SCAN CODE
829
830 0493 B0 00          MOV     AL,0           ; ASCII CODE OF ZERO
831 0495 E9 05F5 R      JMP     K57           ; PUT IT IN THE BUFFER
832
833 0498 B0 F0          MOV     AL,0F0h       ; USE SPECIAL ASCII CODE
834 049A E9 05F5 R      JMP     K57           ; PUT IT IN THE BUFFER
835
836 049D 04 50          ADD     AL,80          ; CONVERT SCAN CODE (EDIT KEYS)
837 049F 8A E0          MOV     AH,AL          ; YES, START PROCESSING
838 04A1 EB F0          JMP     K37A          ; PUT IT IN THE BUFFER
839
840                                     ;----- NOT IN ALTERNATE SHIFT
841
842 04A3                                     K38:          ; NOT-ALT-SHIFT
843                                     ; BL STILL HAS SHIFT FLAGS
844 04A3 F6 C3 04        TEST    BL,CTL_SHIFT  ; ARE WE IN CONTROL SHIFT?
845 04A6 75 03          JNZ    K38A          ; YES, START PROCESSING
846 04A8 E9 0535 R      JMP     K44           ; NOT-CTL-SHIFT
847
848                                     ;----- CONTROL SHIFT, TEST SPECIAL CHARACTERS
849
850                                     ;----- TEST FOR BREAK
851
852 04AB 3C 46          CMP     AL,SCROLL_KEY ; TEST FOR BREAK
853 04AD 75 23          JNE     K39           ; JMP, NO-BREAK
854 04AF F6 C7 10        TEST    BH,KBX        ; IS THIS THE ENHANCED KEYBOARD?
855 04B2 74 05          JB      K38B          ; NO, BREAK IS VALID
856 04B4 F6 C7 02        TEST    BH,LC_E0      ; YES, WAS LAST CODE AN E0?
857 04B7 74 19          JB      K39           ; NO-BREAK, TEST FOR PAUSE
858
859 04B9 8B 1E 001A R    MOV     BX,#BUFFER_HEAD ; RESET BUFFER TO EMPTY
860 04BD 89 1E 001C R    MOV     #BUFFER_TAIL,BX ;
861 04C1 C6 06 0071 R 80 MOV     #BIOS_BREAK,80H ; TURN ON BIOS_BREAK BIT
862
863                                     ;----- ENABLE KEYBOARD
864
865 04C6 B0 AE          MOV     AL,ENA_KBD    ; ENABLE KEYBOARD
866 04C8 EB 063C R      CALL    SHIP_IT       ; EXECUTE ENABLE
867 04CB CD 1B          INT     1BH           ; BREAK INTERRUPT VECTOR
868 04CD 2B C0          SUB     AX,AX          ; PUT OUT DUMMY CHARACTER
869 04CF E9 05F5 R      JMP     K57           ; BUFFER_FILL
870
871                                     ;----- TEST FOR PAUSE
872
873 04D2                                     K39:          ; NO-BREAK
874 04D2 F6 C7 10        TEST    BH,KBX        ; IS THIS THE ENHANCED KEYBOARD?
875 04D5 75 2A          JNE     K41           ; YES, THEN THIS CAN'T BE PAUSE
876 04D7 3C 46          CMP     AL,NUM_KEY    ; LOOK FOR PAUSE KEY
877 04D9 75 26          JNE     K41           ; NO-PAUSE
878 04DB 80 0E 0018 R 80 OR     #KB_FLAG_1,HOLD_STATE ; TURN ON THE HOLD FLAG
879
880                                     ;----- ENABLE KEYBOARD
881
882 04E0 B0 AE          MOV     AL,ENA_KBD    ; ENABLE KEYBOARD
883 04E2 EB 063C R      CALL    SHIP_IT       ; EXECUTE ENABLE
884 04E5 B0 20          MOV     AL,E01        ; END OF INTERRUPT TO CONTROL PORT
885 04E7 E6 20          OUT    020h,AL        ; ALLOW FURTHER KEYSTROKE INTS
886
887                                     ;----- DURING PAUSE INTERVAL, TURN CRT BACK ON
888
889 04E9 80 3E 0049 R 07 CMP     #CRT_MODE,7   ; IS THIS BLACK AND WHITE CARD
890 04EE 74 07          JE      K40           ; YES, NOTHING TO DO
891 04F0 BA 03DB        MOV     DX,03DBh      ; PORT FOR COLOR CARD
892 04F3 AD 0065 R      MOV     AL,#CRT_MODE_SET ; GET THE VALUE OF THE CURRENT MODE
893 04F6 E6             OUT    DX,AL          ; SET THE CRT MODE, SO THAT CRT IS ON
894 04F7                                     ; PAUSE-LOOP
895 04F7 F6 06 0018 R 08 TEST    #KB_FLAG_1,HOLD_STATE ;
896 04FC 75 F9          JNE     K40           ; LOOP UNTIL FLAG TURNED OFF
897 04FE E9 03AA R      JMP     K27           ; INTERRUPT_RETURN_NO_E01
898
899                                     ;----- TEST SPECIAL CASE KEY 55
900
901 0501                                     K41:          ; NO-PAUSE
902 0501 3C 37          CMP     AL,55          ; TEST FOR #/PRTSK KEY
903 0503 75 10          JNE     K42           ; NOT-KEY-55
904 0505 F6 C7 10        TEST    BH,KBX        ; IS THIS THE ENHANCED KEYBOARD?
905 0508 74 05          JB      K41A          ; NO, CTL-PRTSK IS VALID
906 050A F6 C7 02        TEST    BH,LC_E0      ; YES, WAS LAST CODE AN E0?
907 050D 74 20          JB      K42B          ; NO, TRANSLATE TO A FUNCTION
908 050F B8 7200        MOV     AX,114*256    ; START/STOP PRINTING SWITCH
909 0512 E9 05F5 R      JMP     K57           ; BUFFER_FILL
910
911                                     ;----- SET UP TO TRANSLATE CONTROL SHIFT
912
913 0515                                     K42:          ; NOT-KEY-55
914 0515 3C 0F          CMP     AL,15          ; IS IT THE TAB KEY?
915 0517 74 16          JE      K42B          ; YES, XLATE TO FUNCTION CODE
916 0519 3C 35          CMP     AL,53          ; IS IT THE / KEY?
917 051B 75 0B          JNE     K42A          ; NO, NO MORE SPECIAL CASES
918 051D F6 C7 02        TEST    BH,LC_E0      ; YES, IS IT FROM THE KEYPAD?
919 0520 74 06          JB      K42A          ; NO, JUST TRANSLATE
920 0522 B8 9500        MOV     AX,9500h      ; YES, SPECIAL CODE FOR THIS ONE
921 0525 E9 05F5 R      JMP     K57           ; BUFFER FILL
922
923 0528 B8 0000 E      MOV     BX,OFFSET_K8  ; SET UP TO TRANSLATE CTL
924 052B 3C 3B          CMP     AL,59          ; IS IT IN CHARACTER TABLE?
925 052D 75 3E          JNE     K45F          ; YES, GO TRANSLATE CHAR
926 052F B8 0000 E      MOV     BX,OFFSET_K8  ; SET UP TO TRANSLATE CTL
927 0532 E9 05E4 R      JMP     K64           ; NO, GO TRANSLATE_SCAN
928
929                                     ;----- NOT IN CONTROL SHIFT
930
931 0535 3C 37          CMP     AL,55          ; PRINT SCREEN KEY?

```

```

932 0537 76 26             JNE    K46             ; NOT-PRINT-SCREEN
933 0539 F6 C7 10         TEST   BH,KBX          ; IS THIS ENHANCED KEYBOARD?
934 053C 74 07             JZ     K44A           ; NO, TEST FOR SHIFT STATE
935 053E F6 C7 02         TEST   BH,LC_E0        ; YES, LAST CODE A MARKER?
936 0541 75 07             JNZ   K44B           ; YES, IS PRINT SCREEN
937 0543 EB 3B             JMP    SHORT K45C      ; NO, XLATE TO "*" CHARACTER
938 0545 F6 C3 03         K44A: TEST   BL,LEFT_SHIFT+RIGHT_SHIFT ; NOT 101 KBD, SHIFT KEY DOWN?
939 0548 74 36             JZ     K45C           ; NO, XLATE TO "*" CHARACTER
940
941
942 054A 80 AE             ;----- ISSUE INTERRUPT TO PERFORM PRINT SCREEN FUNCTION
943 054C EB 06 3C R       K44B: MOV    AL,ENA_KBD    ; INSURE KEYBOARD IS ENABLED
944 054F 80 20             CALL   SHIP 1T        ; EXECUTE ENABLE
945 0551 E6 20             MOV    AL,E01         ; END OF CURRENT INTERRUPT
946 0553 85 20             OUT   020H,AL        ; SO FURTHER THINGS CAN HAPPEN
947 0554 CD 05             PUSH  BP              ; SAVE POINTER
948 0556 8D 05             INT   $H              ; ISSUE PRINT SCREEN INTERRUPT
949 0557 80 26 00 96 R FC POP    BP              ; RESTORE POINTER
950 055C E9 03 AA R       AND   *KB_FLAG_3,NOT LC_E0+LC_E1 ; ZERO OUT THESE FLAGS
951
952
953
954 056F                     ;----- HANDLE THE IN-CORE KEYS
955 055F 3C 3A             K45:  CMP    AL,68     ; NOT-PRINT-SCREEN
956 0561 77 2C             JA     K46             ; TEST FOR IN-CORE AREA
957
958 0563 3C 35             CMP    AL,53           ; IS THIS THE "/" KEY?
959 0565 75 05             JNE   K45A            ; NO, JUMP
960 0567 F6 C7 02         TEST   BH,LC_E0        ; WAS LAST CODE THE MARKER?
961 056A 75 14             JNZ   K45C            ; YES, TRANSLATE TO CHARACTER
962
963 056C 89 00 1A         K45A: MOV    CX,26      ; LENGTH OF SEARCH
964 056F BF 03 EE R       MOV    DI,OFFSET K30+10 ; POINT TO TABLE OF A-Z CHARS
965 0572 F2 7A AE         REPNE SCASB           ; IS THIS A LETTER KEY?
966 0574 75 05             JNE   K45B            ; NO, SYMBOL KEY
967
968 0576 F6 C3 40         TEST   BL,CAPS_STATE   ; ARE WE IN CAPS_LOCK?
969 0579 75 0A             JNZ   K45D            ; TEST FOR SURE
970 057B F6 C3 03         K45B: TEST   BL,LEFT_SHIFT+RIGHT_SHIFT ; ARE WE IN SHIFT STATE?
971 057E 75 0A             JNZ   K45E            ; YES, UPPERCASE
972
973 0580 8B 00 00 E       K45C: MOV    BX,OFFSET K10 ; NO, LOWERCASE
974 0583 EB 50             JMP    SHORT K56       ; TRANSLATE TO LOWERCASE LETTERS
975 0585
976 0585 F6 C3 03         K45D: TEST   BL,LEFT_SHIFT+RIGHT_SHIFT ; ALMOST-CAPS-STATE
977 0588 75 F6             JNZ   K45C            ; CL ON, IS SHIFT ON, TOO?
978 058A 8B 00 00 E       K45E: MOV    BX,OFFSET K11 ; SHIFTED TEMP OUT OF CAPS STATE
979 058D EB 46             JMP    SHORT K56       ; TRANSLATE TO UPPERCASE LETTERS
980
981
982
983 058F                     ;----- TEST FOR KEYS F1 - F10
984 058F 3C 44             K46:  CMP    AL,68     ; NOT IN-CORE AREA
985 0591 77 02             JA     K47             ; TEST FOR F1 - F10
986 0593 EB 36             JMP    SHORT K53       ; JUMP IF NOT
987
988
989
990 0595                     ;----- HANDLE THE NUMERIC PAD KEYS
991 0595 3C 53             K47:  CMP    AL,53     ; NOT F1 - F10
992 0597 77 2C             JA     K52             ; TEST FOR NUMPAD KEYS
993
994
995 0599 3C 4A             ;----- KEYPAD KEYS, MUST TEST NUM LOCK FOR DETERMINATION
996 059B 74 ED             K48:  CMP    AL,74     ; SPECIAL CASE FOR MINUS
997 059D 3C 4E             JE     K45E            ; GO TRANSLATE
998 059F 74 E9             CMP    AL,78           ; SPECIAL CASE FOR PLUS
999 05A1 F6 C7 02         JE     K45E            ; GO TRANSLATE
1000 05A4 75 0A             TEST   BH,LC_E0        ; IS THIS ONE OF THE NEW KEYS?
1001 05A7 75 0A             JNZ   K49             ; YES, TRANSLATE TO BASE STATE
1002
1003 05A6 F6 C3 20         TEST   BL,NUM_STATE    ; ARE WE IN NUM LOCK?
1004 05A9 75 13             JNZ   K50             ; TEST FOR SURE
1005 05AB F6 C3 03         TEST   BL,LEFT_SHIFT+RIGHT_SHIFT ; ARE WE IN SHIFT STATE?
1006 05AE 75 13             JNZ   K51             ; IF SHIFTED, REALLY NUM STATE
1007
1008 05B0 3C 4C             ;----- BASE CASE FOR KEYPAD
1009 05B2 75 05             K49:  CMP    AL,76     ; SPECIAL CASE FOR BASE STATE 5
1010 05B4 80 F0             JNE   K49A            ; CONTINUE IF NOT KEYPAD 5
1011 05B6 EB 3D 90         MOV    AL,0F0h        ; SPECIAL ASCII CODE
1012 05B9 8B 00 00 E       JMP    K57             ; BUFFER FILL
1013 05BC EB 25             K49A: MOV    BX,OFFSET K10 ; BASE CASE TABLE
1014 05BE EB 25             JMP    SHORT K64       ; CONVERT TO PSEUDO SCAN
1015
1016 05BE F6 C3 03         ;----- MIGHT BE NUM LOCK, TEST SHIFT STATUS
1017 05C1 75 ED             K50:  TEST   BL,LEFT_SHIFT+RIGHT_SHIFT ; ALMOST-NUM-STATE
1018 05C3 EB C5             JNZ   K51             ; SHIFTED TEMP OUT OF NUM STATE
1019
1020
1021
1022
1023 05C5                     ;----- TEST FOR THE NEW KEY ON WT KEYBOARDS
1024 05C5 3C 56             K52:  CMP    AL,86     ; NOT A NUMPAD KEY
1025 05C7 75 02             JNE   K53             ; IS IT THE NEW WT KEY?
1026 05C9 EB B0             JMP    SHORT K45B      ; JUMP IF NOT
1027
1028
1029
1030
1031 05CB F6 C3 03         ;----- MUST BE F11 OR F12
1032 05CE 74 E0             K53:  TEST   BL,LEFT_SHIFT+RIGHT_SHIFT ; F1 - F10 COME HERE, TOO
1033 05D0 75 02             JZ     K49             ; TEST SHIFT STATE
1034 05D0 8B 00 00 E       MOV    BX,OFFSET K11   ; JUMP, LOWERCASE PSEUDO SC'S
1035 05D3 EB 0F             JMP    SHORT K64       ; UPPER CASE PSEUDO SCAN CODES
1036
1037
1038
1039 05D5                     ;----- TRANSLATE THE CHARACTER
1040 05D5 FE C8             K56:  DEC    AL           ; TRANSLATE-CHAR
1041 05D7 2E 1D             XLAT  CS:K11          ; CONVERT ORIGIN
1042 05D9 F6 06 00 96 R 02 TEST   *KB_FLAG_3,LC_E0 ; CONVERT THE SCAN CODE TO ASCII
1043 05DE 74 15             JZ     K57             ; IS THIS A NEW KEY?
1044 05E0 B4 E0             MOV    AH,MC_E0       ; NO, GO FILL BUFFER
1045 05E2 EB 11             JMP    SHORT K57       ; YES, PUT SPECIAL MARKER IN AH
1046

```

SECTION 5

```

1046
1047
1048
1049 05E4
1050 05E4 FE C8
1051 05E6 2E D7
1052 05E8 8A E0
1053 05EA B0 00
1054 05EC F6 06 0096 R 02
1055 05F1 74 02
1056 05F3 B0 E0
1057
1058
1059
1060 05F5
1061 05F5 3C FF
1062 05F7 74 05
1063 05F9 80 FC FF
1064 05FC 75 03
1065
1066 05FE
1067 05FE E9 03A0 R
1068
1069 0601
1070 0601 BB IE 001C R
1071 0605 BB F3
1072 0607 EB 0168 R
1073 060A 9B IE 001A R
1074 060E 74 1D
1075 0610 89 04
1076 0612 89 IE 001C R
1077 0616 FA
1078 0617 B0 20
1079 0619 E6 20
1080 061B B0 AE
1081 061D E8 063C R
1082 0620 B8 9102
1083 0623 CD 15
1084 0625 80 26 0096 R FC
1085 062A E9 03AF R
1086
1087
1088
1089 062D
1090 062D B0 20
1091 062F E6 20
1092 0631 B9 02A6
1093 0634 B3 04
1094 0636 E8 0000 E
1095 0639 E9 03AA R
1096
1097 063C

```

```

;----- TRANSLATE SCAN FOR PSEUDO SCAN CODES
K64:
DEC AL ; TRANSLATE-SCAN-ORGD
XLAT CS:KB ; CONVERT ORIGIN
MOV AH,AL ; CTL TABLE SCAN
MOV AL,0 ; PUT VALUE INTO AH
TEST KB_FLAG_3,LC_E0 ; ZERO ASCII CODE
JZ K57 ; IS THIS A NEW KEY?
MOV AL,MC_E0 ; NO, GO FILL BUFFER
; YES, PUT SPECIAL MARKER IN AL
;----- PUT CHARACTER INTO BUFFER
K57:
CMP AL,-1 ; BUFFER-FILL
JE K59 ; IS THIS AN IGNORE CHAR
CMP AH,-1 ; YES, DO NOTHING WITH IT
JNE K61 ; LOOK FOR -1 PSEUDO SCAN
; NEAR_INTERRUPT_RETURN
K59:
JMP K26 ; NEAR-INTERRUPT-RETURN
; INTERRUPT_RETURN
K61:
MOV BX,BUFFER_TAIL ; GET THE END POINTER TO THE BUFFER
MOV SI,BX ; SAVE THE VALUE
CALL K4 ; ADVANCE THE TAIL
CMP BX,BUFFER_HEAD ; HAS THE BUFFER WRAPPED AROUND
JE K62 ; BUFFER FULL_BEEP
MOV [SI],AX ; STORE THE VALUE
MOV #BUFFER_TAIL,BX ; MOVE THE POINTER UP
CLI ; TURN OFF INTERRUPTS
MOV AL,E01 ; END OF INTERRUPT COMMAND
OUT INTA00,AL ; SEND COMMAND TO INTERRUPT CONTROL PORT
MOV AL,ENA_KBD ; INSURE KEYBOARD IS ENABLED
CALL SHIP_IT ; EXECUTE ENABLE
MOV AX,09102H ; MOVE IN POST CODE & TYPE
INT 15H ; PERFORM OTHER FUNCTION
AND KB_FLAG_3,NOT LC_E0+LC_E1 ; RESET LAST CHAR H.C. FLAG
JMP K27A ; INTERRUPT_RETURN
;----- BUFFER IS FULL SOUND THE BEEPER
K62:
MOV AL,E01 ; ENABLE INTERRUPT CONTROLLER CHIP
OUT INTA00,AL ;
MOV CX,678 ; DIVISOR FOR 1760 HZ
MOV BL,4 ; SHORT BEEP COUNT (1/16 + 1/64 DELAY)
CALL BEEP ; GO TO COMMON BEEP HANDLER
JMP K27 ; EXIT
KB_INT_1 ENDP

```

```

1098 PAGE
1099 -----
1100 |
1101 | SHIP_IT |
1102 | THIS ROUTINE HANDLES TRANSMISSION OF COMMAND AND DATA BYTES |
1103 | TO THE KEYBOARD CONTROLLER. |
1104 |
1105 -----
1106 SHIP_IT PROC NEAR
1107 063C PUSH AX ; SAVE DATA TO SEND
1108 063C 50
1109
1110 |----- WAIT FOR COMMAND TO BE ACCEPTED
1111 063D FA ; DISABLE INTERRUPTS TILL DATA SENT
1112 063E 2B C9 ; CLEAR TIMEOUT COUNTER
1113 0640
1114 0640 E4 64 S10: IN AL,STATUS_PORT ; READ KEYBOARD CONTROLLER STATUS
1115 0642 A8 02 TEST AL,INPT_BUF_FULL ; CHECK FOR ITS INPUT BUFFER BUSY
1116 0644 E0 FA LOOPNZ S10 ; WAIT FOR COMMAND TO BE ACCEPTED
1117
1118 0646 58 POP AX ; GET DATA TO SEND
1119 0647 E6 64 OUT STATUS_PORT,AL ; SEND TO KEYBOARD CONTROLLER
1120 0649 FB STI ; ENABLE INTERRUPTS AGAIN
1121 064A C3 RET ; RETURN TO CALLER
1122 064B SHIP_IT ENDP
1123
1124 |-----
1125 |
1126 | SND_DATA |
1127 |
1128 | THIS ROUTINE HANDLES TRANSMISSION OF COMMAND AND DATA BYTES |
1129 | TO THE KEYBOARD AND RECEIPT OF ACKNOWLEDGEMENTS. IT ALSO |
1130 | HANDLES ANY RETRIES IF REQUIRED |
1131 |-----
1132
1133 SND_DATA PROC NEAR
1134 064B PUSH AX ; SAVE REGISTERS
1135 064B 50 PUSH BX ; *
1136 064C 53 PUSH CX ; *
1137 064D 51 MOV BH,AL ; SAVE TRANSMITTED BYTE FOR RETRIES
1138 064E 8A F8 MOV BL,3 ; LOAD RETRY COUNT
1139 0650 B3 03 SD0: CLI ; DISABLE INTERRUPTS
1140 0652 FA AND *KB_FLAG_2,NOT (KB_FE+KB_FA) ; CLEAR ACK AND RESEND FLAGS
1141 0653 80 26 0097 R CF
1142
1143 |----- WAIT FOR COMMAND TO BE ACCEPTED
1144
1145 0658 2B C9 SUB CX,CX
1146 065A E4 64 SD5: IN AL,STATUS_PORT
1147 065C A8 02 TEST AL,INPT_BUF_FULL
1148 065E E0 FA LOOPNZ SD5 ; WAIT FOR COMMAND TO BE ACCEPTED
1149
1150 0660 8A CT MOV AL,BH ; REESTABLISH BYTE TO TRANSMIT
1151 0662 E6 60 OUT PORT_A,AL ; SEND BYTE
1152 0664 FB STI ; ENABLE INTERRUPTS
1153 0665 B9 1A00 MOV CX,01A00H ; LOAD COUNT FOR 10ms+
1154 0668 F6 06 0097 R 30 SD1: TEST *KB_FLAG_2,KB_FE+KB_FA ; SEE IF EITHER BIT SET
1155 066D 75 0D JNZ SD5 ; IF SET, SOMETHING RECEIVED GO PROCESS
1156 066F E2 F7 LOOP SD1 ; OTHERWISE WAIT
1157
1158 0671 FE CB SD2: DEC BL ; DECREMENT RETRY COUNT
1159 0673 75 0D JNZ SD0 ; RETRY TRANSMISSION
1160 0675 80 0E 0097 R 80 OR *KB_FLAG_2,KB_ERR ; TURN ON TRANSMIT ERROR FLAG
1161 067A EB 07 JMP SHORT SD4 ; RETRIES EXHAUSTED FORGET TRANSMISSION
1162
1163 067C F6 06 0097 R 10 SD3: TEST *KB_FLAG_2,KB_FA ; SEE IF THIS IS AN ACKNOWLEDGE
1164 0681 74 EE JZ SD2 ; IF NOT, GO RESEND
1165
1166 0683 59 POP CX ; RESTORE REGISTERS
1167 0684 5B POP BX ; *
1168 0685 58 POP AX ; *
1169 0686 C3 RET ; RETURN, GOOD TRANSMISSION
1170 0687 SND_DATA ENDP

```

SECTION 5

```

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

```

```

1 PAGE 118,121
2 TITLE PRT ----- 06/10/85 PRINTER ADAPTER BIOS
3 .286C
4 .LIST
5 0000 CODE SEGMENT BYTE PUBLIC
6 PUBLIC PRINTER_IO_1
7 EXTRN DDS:INEAR
8
9
10 ;----- INT 17 H -----
11 ; PRINTER_IO
12 ; THIS ROUTINE PROVIDES COMMUNICATION WITH THE PRINTER
13 ; INPUT
14 ; (AH)= 00H PRINT THE CHARACTER IN (AL)
15 ; ON RETURN, (AH)= 1 IF CHARACTER NOT BE PRINTED (TIME OUT)
16 ; OTHER BITS SET AS ON NORMAL STATUS CALL
17 ; (AH)= 01H INITIALIZE THE PRINTER PORT
18 ; RETURNS WITH (AH) SET WITH PRINTER STATUS
19 ; (AH)= 02H READ THE PRINTER STATUS INTO (AH)
20 ;
21 ;
22 ;
23 ;
24 ;
25 ;
26 ;
27 ;
28 ;
29 ;
30 ;
31 ;
32 ;
33 ;
34 ;
35 ;
36 ;
37 ; (DX) = PRINTER TO BE USED (0,1,2) CORRESPONDING TO ACTUAL VALUES
38 ; IN *PRINTER_BASE AREA
39 ; DATA AREA *PRINTER_BASE CONTAINS THE BASE ADDRESS OF THE PRINTER CARD(S)
40 ; AVAILABLE (LOCATED AT BEGINNING OF DATA SEGMENT, 408H ABSOLUTE, 3 WORDS)
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
49 PRINTER_IO_1 PROC FAR ; ENTRY POINT FOR ORG 0EFD2H
50 STT ; INTERRUPTS BACK ON
51 PUSH DS ; SAVE SEGMENT
52 PUSH SI
53 PUSH DX
54 PUSH CX
55 PUSH BX
56 CALL DDS ; ADDRESS DATA SEGMENT
57 MOV SI,DX ; GET PRINTER PARAMETER
58 SHR DX,2 ; TEST PARAMETER
59 JNZ B10 ; RETURN IF NOT IN RANGE
60 MOV BL,*PRINT_TIM_OUT[SI] ; LOAD TIMEOUT VALUE
61 MOV DI,*PRINTER_BASE[SI] ; WORD OFFSET INTO TABLE INTO (SI)
62 OR DX,DX ; GET BASE ADDRESS FOR PRINTER CARD
63 JZ B10 ; TEST DX = ZERO, INDICATING NO PRINTER
64 OR AH,AH ; EXIT: NO PRINTER ADAPTER AT OFFSET
65 JZ B20 ; TEST FOR (AH)= 00H
66 DEC AH ; PRINT CHARACTER IN (AL)
67 JZ B80 ; TEST FOR (AH)= 01H
68 DEC AH ; INITIALIZE PRINTER
69 JZ B50 ; TEST FOR (AH)= 02H
70 ; GET PRINTER STATUS
71 POP BX ; RETURN
72 POP CX
73 POP DX
74 POP SI
75 POP DS ; RECOVER REGISTERS
76 IRET ; RETURN TO CALLING PROGRAM
77
78 ;----- PRINT THE CHARACTER IN (AL) -----
79
80 B20: PUSH AX ; SAVE VALUE TO PRINT
81 OUT DX,AL ; OUTPUT CHARACTER TO DATA PORT
82 INC DX ; POINT TO STATUS PORT
83
84 ;----- CHECK FOR PRINTER BUSY -----
85
86 PUSH BX ; SAVE TIMEOUT BASE COUNT
87 IN AL,DX ; GET STATUS PORT VALUE
88 TEST AL,80H ; IS THE PRINTER CURRENTLY BUSY
89 JNZ B25 ; SKIP SYSTEM DEVICE BUSY CALL IF NOT
90
91 ;----- INT 15 H -- DEVICE BUSY -----
92
93 MOV AX,90FEH ; FUNCTION 90 PRINTER ID
94 INT 15H ; SYSTEM CALL
95
96 ;----- WAIT BUSY -----
97
98 B25: SUB BH,BH ; ADJUST OUTER LOOP COUNT
99 RCL BX,2 ; CLEAR (BH)
100 ; MULTIPLY BY 4
101 B30: SUB CX,CX ; INNER LOOP (64K)
102 B35:
103 IN AL,DX ; GET STATUS
104 MOV AH,AL ; STATUS TO (AH) ALSO
105 TEST AL,80H ; IS THE PRINTER CURRENTLY BUSY
106 JNZ B40 ; GO TO OUTPUT STROBE
107 LOOP B35 ; LOOP IF NOT
108 DEC BX ; DECREMENT OUTER LOOP COUNT
109 JNZ B30 ; MAKE ANOTHER PASS IF NOT ZERO
110
111 POP BX ; CLEAR (BX) FROM STACK
112 OR AH,1 ; SET ERROR FLAG
113 AND AH,0F9H ; TURN OFF THE UNUSED BITS
114 JMP SHORT B70 ; RETURN WITH ERROR FLAG SET

```

SECTION 5

```

115 005A          B40:          ; SEND STROBE PULSE
116 005A 8B      POP     BX      ; RESTORE (BX) WITH TIMEOUT COUNT
117 005B 80 0D   MOV     AL,0DH   ; SET THE STROBE LOW (BIT ON)
118 005D 42      INC     DX      ; OUTPUT STROBE TO CONTROL PORT
119 005E FA      CLI          ; PREVENT INTERRUPT PULSE STRETCHING
120 005F EE      OUT     DX,AL   ; OUTPUT STROBE BIT > 1us < 5us
121 0060 EB 00   JMP     $+2      ; I/O DELAY TO ALLOW FOR LINE LOADING
122 0062 EB 00   JMP     $+2      ; AND FOR CORRECT PULSE WIDTH
123 0064 80 0C   MOV     AL,0CH   ; SET THE -STROBE HIGH
124 0066 EE      OUT     DX,AL
125 0067 FB      STI          ; INTERRUPTS BACK ON
126 0068 58      POP     AX      ; RECOVER THE OUTPUT CHAR
127
128             ;----- PRINTER STATUS
129
130 0069          B50:          ; SAVE (AL) REGISTER
131 0069 50      PUSH    AX
132 006A          B60:          ; GET PRINTER ATTACHMENT BASE ADDRESS
133 006A 8B 94 0008 R MOV     DX,PRINTER_BASE[S1] ; POINT TO CONTROL PORT
134 006E 42      INC     DX      ; PRE-CHARGE +BUSY LINE IF FLOATING
135 006F EC      IN     AL,DX   ; GET PRINTER STATUS HARDWARE BITS
136 0070 EC      IN     AL,DX   ; SAVE
137 0071 8A E0   MOV     AH,AL   ; TURN OFF UNUSED BITS
138 0073 80 E4 F8 AND     AH,0F6H
139 0076          B70:          ; RECOVER (AL) REGISTER
140 0076 5A      POP     DX
141 0077 8A C2   MOV     AL,DL   ; MOVE CHARACTER INTO (AL)
142 0079 80 F4 48 XOR     AH,48H  ; FLIP A COUPLE OF BITS
143 007C EB AC   JMP     B10     ; RETURN FROM ROUTINE WITH STATUS IN AH
144
145             ;----- INITIALIZE THE PRINTER PORT
146
147 007E          B80:          ; SAVE (AL)
148 007E 50      PUSH    AX      ; POINT TO OUTPUT PORT
149 007F 42      INC     DX
150 0080 42      INC     DX
151 0081 80 08   MOV     AL,8    ; SET INIT LINE LOW
152 0083 EE      OUT     DX,AL
153 0084 B5 0FA0 MOV     AX,1000*4 ; ADJUST FOR INITIALIZATION DELAY LOOP
154 0087          B90:          ; INIT LOOP
155 0087 48      DEC     AX      ; LOOP FOR RESET TO TAKE
156 0088 75 FD   JNZ    B90     ; INIT LOOP
157 008A 80 0C   MOV     AL,0CH  ; NO INTERRUPTS, NON AUTO LF, INIT HIGH
158 008C EE      OUT     DX,AL
159 008D EB D8   JMP     B60     ; EXIT THROUGH STATUS ROUTINE
160
161 008F          PRINTER_IO_1  ENDP
162
163 008F          CODE      ENDS
164             END

```

```

1 PAGE 118,121
2 TITLE RS232 ---- 06/10/85 COMMUNICATIONS BIOS (RS232)
3 .286C
4 .LIST
5 0000 CODE SEGMENT BYTE PUBLIC
6 PUBLIC RS232_10_1
7 EXTRN A1:NEAR
8 EXTRN DS:NEAR
9
10 ;----- INT 14 H -----
11 RS232_10
12 ; THIS ROUTINE PROVIDES BYTE STREAM I/O TO THE COMMUNICATIONS
13 ; PORT ACCORDING TO THE PARAMETERS:
14 ;
15 ; (AH)= 00H INITIALIZE THE COMMUNICATIONS PORT
16 ; (AL) HAS PARAMETERS FOR INITIALIZATION
17 ;
18 ; 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 - 180 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 - 3 ALLOWED)
67 ;
68 ; DATA AREA #RS232_BASE CONTAINS THE BASE ADDRESS OF THE 8250 ON THE CARD
69 ; LOCATION #00H 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 ASSUME CS:CODE,DS:DATA
78 0000 RS232_10_1 PROC FAR
79 ;----- VECTOR TO APPROPRIATE ROUTINE
80 ;
81 0000 FB STI ; INTERRUPTS BACK ON
82 0001 IE PUSH DS ; SAVE SEGMENT
83 0002 52 PUSH DX
84 0003 56 PUSH SI
85 0004 57 PUSH DI
86 0005 51 PUSH CX
87 0006 53 PUSH BX
88 0007 8B F2 MOV SI,DX ; RS232 VALUE TO (SI)
89 0009 8B FA MOV DI,DX ; AND TO (DI) (FOR TIMEOUTS)
90 000B C1 EA 02 SHR DX,2 ; TEST PARAMETER
91 000E 75 20 JNZ A3 ; RETURN IF NOT IN RANGE
92 0010 D1 E6 SHL SI,1 ; WORD OFFSET
93 0012 EB 0000 E CALL DS
94 0015 8B 94 0000 R MOV DX,#RS232_BASE[SI] ; GET BASE ADDRESS
95 0019 0B D2 OR DX,DX ; TEST FOR 0 BASE ADDRESS
96 001B 74 13 JZ A3 ; RETURN
97 001D 0A E4 OR AH,AH ; TEST FOR (AH)= 00H
98 001F 74 16 JZ A4 ; COMMO INITIALIZATION
99 0021 FE CC DEC AH ; TEST FOR (AH)= 01H
100 0023 74 AB JZ A5 ; SEND (AL)
101 0025 FE CC DEC AH ; TEST FOR (AH)= 02H
102 0027 74 70 JZ A12 ; RECEIVE INTO (AL)
103 0029 A2: DEC AH ; TEST FOR (AH)= 03H
104 0029 FE CC JZ A3
105 002B 75 03 JNZ A3 ; COMMUNICATION STATUS
106 002D E9 00BB R JMP A18 ; RETURN FROM RS232
107 0030 A3:
108 0030 5B POP BX
109 0031 59 POP CX
110 0032 5F POP DI
111 0033 5E POP SI
112 0034 5A POP DX
113 0035 1F POP DS
114 0036 CF IRET ; RETURN TO CALLER, NO ACTION

```

SECTION 5

```

115 PAGE
116 ;----- INITIALIZE THE COMMUNICATIONS PORT
117
118 0037 A4:
119 0037 8A E0 MOV AH,AL ; SAVE INITIALIZATION PARAMETERS IN (AH)
120 0039 83 C2 03 ADD DX,3 ; POINT TO 8250 CONTROL REGISTER
121 003C 80 80 MOV AL,80H
122 003E EE OUT DX,AL ; SET DLAB=1
123
124 ;----- DETERMINE BAUD RATE DIVISOR
125
126 003F 8A D4 MOV DL,AH ; GET PARAMETERS TO (DL)
127 0041 B1 04 MOV CL,4
128 0043 D2 C2 ROL DL,CL
129 0045 81 E2 000E AND DX,0E0H ; ISOLATE THEM
130 0049 BF 0000 E MOV DI,OFFSET A1 ; BASE OF TABLE
131 004C 03 FA ADD DI,DX ; PUT INTO INDEX REGISTER
132 004E 8B 94 0000 R MOV DX,RS232_BASE[SI] ; POINT TO HIGH ORDER OF DIVISOR
133 0052 42 INC DX
134 0053 2E 8A 45 01 MOV AL,CS:[DI]+1 ; GET HIGH ORDER OF DIVISOR
135 0057 EE OUT DX,AL ; SET ms OF DIVISOR TO 0
136 0058 4A DEC DX
137 0059 EB 00 JMP $+2 ; I/O DELAY
138 005B 2E 8A 05 MOV AL,CS:[DI] ; GET LOW ORDER OF DIVISOR
139 005E EE OUT DX,AL ; SET LOW OF DIVISOR
140 005F 83 C2 03 ADD DX,3
141 0062 8A C4 MOV AL,AH ; GET PARAMETERS BACK
142 0064 24 1F AND AL,01FH ; STRIP OFF THE BAUD BITS
143 0066 EE OUT DX,AL ; LINE CONTROL TO 8 BITS
144 0067 4A DEC DX
145 0068 4A DEC DX
146 0069 EB 00 JMP $+2 ; I/O DELAY
147 006B 8D 00 MOV AL,0 ; INTERRUPT ENABLES ALL OFF
148 006D EE OUT DX,AL ; COM_STATUS
149 006E EB 4B JMP SHORT A18
150
151 ;----- SEND CHARACTER IN (AL) OVER COMMO LINE
152
153 0070 A5:
154 0070 50 PUSH AX ; SAVE CHAR TO SEND
155 0071 83 C2 04 ADD DX,4 ; MODEM CONTROL REGISTER
156 0074 80 03 OUT DX,AL ; DTR AND RTS
157 0076 EE OUT DX,AL ; DATA TERMINAL READY, REQUEST TO SEND
158 0077 42 INC DX ; MODEM STATUS REGISTER
159 0078 42 INC DX
160 0079 B7 30 MOV BH,30H ; DATA SET READY & CLEAR TO SEND
161 007B E8 00CA R CALL WAIT_FOR_STATUS ; ARE BOTH TRUE
162 007E 74 08 JE A9 ; YES, READY TO TRANSMIT CHAR
163 0080
164 0080 89 POP CX
165 0081 8A C1 MOV AL,CL ; RELOAD DATA BYTE
166 0083
167 0083 80 CC 80 A8: OR AH,80H ; INDICATE TIME OUT
168 0086 EB A8 JMP A3 ; RETURN
169
170 0088 A9: DEC DX ; CLEAR TO SEND
171 0088 4A INC DX ; LINE STATUS REGISTER
172 0089 ; WAIT SEND
173 0089 B7 20 MOV BH,20H ; IS TRANSMITTER READY
174 008B E8 00CA R CALL WAIT_FOR_STATUS ; TEST FOR TRANSMITTER READY
175 008E 75 F0 JNZ A7 ; RETURN WITH TIME OUT SET
176 0090 ; OUT_CHAR
177 0090 83 EA 05 SUB DX,5 ; DATA PORT
178 0093 59 POP CX ; RECOVER IN CX TEMPORARILY
179 0094 8A C1 MOV AL,CL ; MOVE CHAR TO AL FOR OUT, STATUS IN AH
180 0096 EE OUT DX,AL ; OUTPUT CHARACTER
181 0097 EB 97 JMP A3 ; RETURN
182
183 ;----- RECEIVE CHARACTER FROM COMMO LINE
184
185 0099 A12:
186 0099 83 C2 04 ADD DX,4 ; MODEM CONTROL REGISTER
187 009C 80 01 MOV AL,1 ; DATA TERMINAL READY
188 009E EE OUT DX,AL ; TEST FOR DSR
189 009F 42 INC DX ; MODEM STATUS REGISTER
190 00A0 42 INC DX
191 00A1 B7 20 MOV BH,20H ; WAIT_DSR
192 00A1 B7 20 CALL WAIT_FOR_STATUS ; TEST FOR DSR
193 00A3 E8 00CA R JNZ A8 ; RETURN WITH ERROR
194 00A6 75 DB ; WAIT_DSR END
195 00A8 ; LINE STATUS REGISTER
196 00A8 4A DEC DX ; WAIT RECV
197 00A9 ; RECEIVE BUFFER FULL
198 00A9 B7 01 MOV BH,1 ; TEST FOR RECEIVE BUFFER FULL
199 00AB E8 00CA R CALL WAIT_FOR_STATUS ; SET TIME OUT ERROR
200 00AE 75 D3 JNZ A8 ; GET_CHAR
201 00B0 ; TEST FOR ERROR CONDITIONS ON RECEIVE
202 00B0 80 E4 1E AND AH,00011110B
203
204 00B3 8B 94 0000 R MOV DX,RS232_BASE[SI] ; DATA PORT
205 00B7 EC IN AL,DX ; GET CHARACTER FROM LINE
206 00B8 E9 0030 R JMP A3 ; RETURN
207
208 ;----- COMMO PORT STATUS ROUTINE
209
210 00BB A18:
211 00BB 8B 94 0000 R MOV DX,RS232_BASE[SI]
212 00BF 83 C2 05 ADD DX,5 ; CONTROL PORT
213 00C2 EC IN AL,DX ; GET LINE CONTROL STATUS
214 00C3 8A E0 MOV AH,AL ; PUT IN (AH) FOR RETURN
215 00C5 42 INC DX ; POINT TO MODEM STATUS REGISTER
216 00C6 EC IN AL,DX ; GET MODEM CONTROL STATUS
217 00C7 E9 0030 R JMP A3 ; RETURN

```

```

218 PAGE
219 ;-----
220 ; WAIT FOR STATUS ROUTINE
221 ; ENTRY: (BH) = STATUS BIT(S) TO LOOK FOR
222 ; (DX) = ADDRESS OF STATUS REG
223 ; EXIT: ZERO FLAG ON = STATUS FOUND
224 ; ZERO FLAG OFF = TIMEOUT
225 ; (AH) = LAST STATUS READ
226 ;-----
227
228 00CA WAIT_FOR_STATUS PROC NEAR
229 00CA 8A 9D 007C R MOV BL,#RS232_TIM_OUT[D1] ; LOAD OUTER LOOP COUNT
230
231 ;----- ADJUST OUTER LOOP COUNT
232
233 00CE 55 PUSH BP ; SAVE (BP)
234 00CF 53 PUSH BX ; SAVE (BX)
235 00D0 8D POP BP ; USE BP FOR OUTER LOOP COUNT
236 00D1 81 E5 00FF AND BP,00FFH ; STRIP HIGH BITS
237 00D5 D1 D5 RCL BP,1 ; MULTIPLY OUTER COUNT BY 4
238 00D7 D1 D5 RCL BP,1
239 00D9 WFS0:
240 00D9 2B C9 SUB CX,CX
241 00DB WFS1:
242 00DB EC IN AL,DX ; GET STATUS
243 00DC 8A E0 MOV AH,AL ; MOVE TO (AH)
244 00DE 22 C7 AND AL,BH ; ISOLATE BITS TO TEST
245 00E0 3A C7 CMP AL,BH ; EXACTLY = TO MASK
246 00E2 74 07 JE WFS_END ; RETURN WITH ZERO FLAG ON
247
248 00E4 E2 F5 LOOP WFS1 ; TRY AGAIN
249
250 00E6 4D DEC BP
251 00E7 75 F0 JNZ WFS0
252
253 00E9 0A FF OR BH,BH ; SET ZERO FLAG OFF
254 00EB WFS_END:
255 00EB 5D POP BP ; RESTORE (BP)
256 00EC C3 RET
257
258 00ED WAIT_FOR_STATUS ENDP
259 RS232_10_1 ENDP
260
261 00ED CODE ENDS
262 END
263

```

SECTION 5

```

1 PAGE 118,121
2 TITLE VIDEO1 --- 03/24/86 VIDEO DISPLAY BIOS
3 .LIST
4 0000 CODE SEGMENT BYTE PUBLIC
5
6 PUBLIC ACT_DISP_PAGE
7 PUBLIC READ_AC_CURRENT
8 PUBLIC READ_CURSOR
9 PUBLIC READ_DOT
10 PUBLIC READ_LPEN
11 PUBLIC SCROLL_DOWN
12 PUBLIC SCROLL_UP
13 PUBLIC SET_COLOR
14 PUBLIC SET_CPOS
15 PUBLIC SET_CTYPE
16 PUBLIC SET_MODE
17 PUBLIC WRITE_AC_CURRENT
18 PUBLIC WRITE_C_CURRENT
19 PUBLIC WRITE_DOT
20 PUBLIC WRITE_TTY
21 PUBLIC VIDEO_IO_1
22 PUBLIC VIDEO_STATE
23
24 EXTRN BEEP_NEAR ; SPEAKER BEEP ROUTINE
25 EXTRN CRT_CHAR_GEN_NEAR ; CHARACTER GENERATOR GRAPHICS TABLE
26 EXTRN DDSINEAR ; LOAD (DS) WITH DATA SEGMENT SELECTOR
27 EXTRN M5:WORD ; REGEN BUFFER LENGTH TABLE
28 EXTRN M6:BYTE ; COLUMNS PER MODE TABLE
29 EXTRN M7:BYTE ; MODE SET VALUE PER MODE TABLE
30
31
32 --- INT 10 H ---
33
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 (AL) = 04H 320X200 COLOR
44 (AL) = 05H 320X200 BW MODE
45 (AL) = 06H 640X200 BW MODE
46 (AL) = 07H 80X25 MONOCHROME (USED INTERNAL TO VIDEO ONLY)
47 *** NOTES -BW MODES OPERATE SAME AS COLOR MODES, BUT COLOR
48 -BURST IS NOT ENABLED
49 -CURSOR IS NOT DISPLAYED IN GRAPHICS MODE
50
51 (AH) = 01H SET CURSOR TYPE
52 (CH) = BITS 4-0 = START LINE FOR CURSOR
53 ** HARDWARE WILL ALWAYS CAUSE BLINK
54 ** SETTING BIT 5 OR 6 WILL CAUSE ERRATIC BLINKING
55
56 (CL) = BITS 4-0 = END LINE FOR CURSOR
57
58 (AH) = 02H SET CURSOR POSITION
59 (DH,DL) = ROW,COLUMN (00H,00H) IS UPPER LEFT
60 (BH) = PAGE NUMBER (MUST BE 00H FOR GRAPHICS MODES)
61
62 (AH) = 03H READ CURSOR POSITION
63 (BH) = PAGE NUMBER (MUST BE 00H FOR GRAPHICS MODES)
64 ON EXIT (CH,CL) = CURSOR MODE CURRENTLY SET
65
66 (AH) = 04H READ LIGHT PEN POSITION
67 ON EXIT:
68 (AH) = 00H -- LIGHT PEN SWITCH NOT DOWN/NOT TRIGGERED
69 (AH) = 01H -- VALID LIGHT PEN VALUE IN REGISTERS
70 (DH,DL) = ROW,COLUMN OF CHARACTER LP POSITION
71 (CH) = RASTER LINE (0-199)
72 (BX) = PIXEL COLUMN (0-319,639)
73
74 (AH) = 05H SELECT ACTIVE DISPLAY PAGE (VALID ONLY FOR ALPHA MODES)
75 (AL) = NEW PAGE VALUE (0-7 FOR MODES 0&1, 0-3 FOR MODES 2&3)
76
77 (AH) = 06H SCROLL ACTIVE PAGE UP
78 (AL) = NUMBER OF LINES, ( LINES BLANKED AT BOTTOM 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 (AH) = 07H SCROLL ACTIVE PAGE DOWN
85 (AL) = NUMBER OF LINES, INPUT LINES BLANKED AT TOP OF WINDOW
86 (AL) = 00H MEANS BLANK ENTIRE WINDOW
87 (CH,CL) = ROW,COLUMN OF UPPER LEFT CORNER OF SCROLL
88 (DH,DL) = ROW,COLUMN OF LOWER RIGHT CORNER OF SCROLL
89 (BH) = ATTRIBUTE TO BE USED ON BLANK LINE
90
91 CHARACTER HANDLING ROUTINES
92
93 (AH) = 08H READ ATTRIBUTE/CHARACTER AT CURRENT CURSOR POSITION
94 (BH) = DISPLAY PAGE (VALID FOR ALPHA MODES ONLY)
95 ON EXIT:
96 (AL) = CHAR READ
97 (AH) = ATTRIBUTE OF CHARACTER READ (ALPHA MODES ONLY)
98
99 (AH) = 09H WRITE ATTRIBUTE/CHARACTER AT CURRENT CURSOR POSITION
100 (BH) = DISPLAY PAGE (VALID FOR ALPHA MODES ONLY)
101 (CX) = COUNT OF CHARACTERS TO WRITE
102 (AL) = CHAR TO WRITE
103 (BL) = ATTRIBUTE OF CHARACTER (ALPHA)/COLOR OF CHAR (GRAPHICS)
104
105 (AH) = 0AH WRITE CHARACTER ONLY AT CURRENT CURSOR POSITION
106 (BH) = DISPLAY PAGE (VALID FOR ALPHA MODES ONLY)
107 (CX) = COUNT OF CHARACTERS TO WRITE
108 (AL) = CHAR TO WRITE
109
110 NOTE: USE FUNCTION (AH) = 09H IN GRAPHICS MODES
111 FOR READ/WRITE CHARACTER INTERFACE WHILE IN GRAPHICS MODE, THE
112 CHARACTERS ARE FORMED FROM A CHARACTER GENERATOR IMAGE
113 MAINTAINED IN THE SYSTEM ROM. ONLY THE 1ST 128 CHARS
114 ARE CONTAINED THERE. TO READ/WRITE THE SECOND 128 CHARS,
115 THE USER MUST INITIALIZE THE POINTER AT INTERRUPT 1FH
116 (LOCATION 0007CH) TO POINT TO THE 1K BYTE TABLE CONTAINING
117 THE CODE POINTS FOR THE SECOND 128 CHARS (128-255).
118
119 FOR WRITE CHARACTER INTERFACE IN GRAPHICS MODE, THE REPLICATION FACTOR
120 CONTAINED IN (CX) ON ENTRY WILL PRODUCE VALID RESULTS ONLY
121 FOR CHARACTERS CONTAINED ON THE SAME ROW. CONTINUATION TO
122 SUCCEEDING LINES WILL NOT PRODUCE CORRECTLY.

```

```

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

SECTION 5

```

229 0048 76 03          JNE     M2                ; SKIP IF NOT MONOCHROME CARD
230 004A BF B000        MOV     DI,0B000H         ; ELSE GET SEGMENT FOR MONOCHROME CARD
231 004D                M2:
232 004D 8E C7          MOV     ES,DI            ; SET UP TO POINT AT VIDEO MEMORY AREAS
233 004F 8A C4          MOV     AL,AH           ; PLACE COMMAND IN LOW BYTE OF (AX)
234 0051 98            CBW                     ; AND FORM BYTE OFFSET WITH COMMAND
235 0052 01 E0          SAL     AX,1            ; TIMES 2 FOR WORD TABLE LOOKUP
236 0054 96            XCHG   SI,AX           ; MOVE OFFSET INTO LOOK UP REGISTER (SI)
237                    ; AND RESTORE COMMAND/DATA INTO (AX)
238 0055 8A 26 0049 R   MOV     AH,*CRT_MODE    ; MOVE CURRENT MODE INTO (AH) REGISTER
239                    ;
240 0059 2E1 FF A4 0000 R JMP     WORD PTR CS:[SI+OFFSET M1] ; GO TO SELECTED FUNCTION
241                    ;
242 005E                M4:
243 005E CF            IRET                    ; COMMAND NOT VALID
244 005F                ; DO NOTHING IF NOT IN VALID RANGE
245                    ;
246                    ;-----
247                    ; SET_MODE
248                    ; THIS ROUTINE INITIALIZES THE ATTACHMENT TO
249                    ; THE SELECTED MODE. THE SCREEN IS BLANKED.
250                    ;
251                    ; INPUT
252                    ; *EQUIP_FLAG BITS 5-4 = MODE/WIDTH
253                    ; 11 = MONOCHROME (FORCES MODE 7)
254                    ; 01 = COLOR ADAPTER 40x25 (MODE 0 DEFAULT)
255                    ; 10 = COLOR ADAPTER 80x25 (MODE 2 DEFAULT)
256                    ; (AL) = COLOR MODE REQUESTED ( RANGE 0 - 6 )
257                    ;
258                    ; OUTPUT
259                    ; NONE
260                    ;-----
261 005F                SET_MODE PROC NEAR
262 005F BA 03D4        MOV     DX,03D4H         ; ADDRESS OF COLOR CARD
263 0061 88 3E 0010 R   MOV     DI,*EQUIP_FLAG ; GET EQUIPMENT FLAGS SETTING
264 0063 81 ET 0030    AND     DI,30H          ; ISOLATE CRT SWITCHES
265 0065 83 FF 30     CMP     DI,30H          ; IS BW CARD INSTALLED AS PRIMARY
266 0067 75 06        JNE     M6C             ; SKIP AND CHECK IF COLOR
267 0069 80 07        MOV     AL,7            ; ELSE INDICATE INTERNAL BW CARD MODE
268 006B 82 84        MOV     DL,0B4H         ; SET ADDRESS OF BW (MONOCHROME) CARD
269 006D EB 0D        JMP     SHORT M8        ; CONTINUE WITH FORCED MODE 7
270                    ;
271                    ; M6C:
272                    ; AL,7
273                    ; CHECK FOR VALID COLOR MODES 0-6
274                    ; CONTINUE IF BELOW MODE 7
275                    ; AL,0
276                    ; FORCE DEFAULT 40x25 BW MODE
277                    ; DI,10H
278                    ; CHECK FOR *EQUIP_FLAG AT 40x25 COLOR
279                    ; JE
280                    ; M8
281                    ; CONTINUE WITH MODE 0 IF NOT
282                    ; MOV
283                    ; AL,2
284                    ; ELSE FORCE MODE 2
285                    ;
286                    ; M8:
287                    ; MOV
288                    ; *CRT_MODE,AL
289                    ; SAVE MODE IN GLOBAL VARIABLE
290                    ; *ADDR_6845,DX
291                    ; SAVE ADDRESS OF BASE
292                    ; *ROWS,25-1
293                    ; INITIALIZE DEFAULT ROW COUNT OF 25
294                    ; DS
295                    ; SAVE POINTER TO DATA SEGMENT
296                    ; AX
297                    ; SAVE MODE NUMBER (AL)
298                    ; CBW
299                    ; CLEAR HIGH BYTE OF MODE
300                    ; MOV
301                    ; SI,AX
302                    ; SET TABLE POINTER INDEXED BY MODE
303                    ; AL,CS:[SI + OFFSET M7]
304                    ; GET THE MODE SET VALUE FROM TABLE
305                    ; MOV
306                    ; *CRT_MODE_SET,AL
307                    ; SAVE THE MODE SET VALUE
308                    ; AND
309                    ; AL,037H
310                    ; VIDEO OFF, SAVE HIGH RESOLUTION BIT
311                    ; PUSH
312                    ; DX
313                    ; SAVE OUTPUT PORT VALUE
314                    ; ADD
315                    ; DX,4
316                    ; POINT TO CONTROL REGISTER
317                    ; OUT
318                    ; DX,AL
319                    ; RESET VIDEO TO OFF TO SUPPRESS ROLLING
320                    ; POP
321                    ; DX
322                    ; BACK TO BASE REGISTER
323                    ; ASSUME
324                    ; DS:ABS0
325                    ; SET UP FOR ABS0 SEGMENT
326                    ; SUB
327                    ; BX,BX
328                    ; ESTABLISH VECTOR TABLE ADDRESSING
329                    ; MOV
330                    ; DS,BX
331                    ; GET POINTER TO VIDEO PARMS
332                    ; LDS
333                    ; BX,*PARAM_PTR
334                    ; DS:CODE
335                    ; ASSUME
336                    ; DS:DATA
337                    ; POP
338                    ; AX
339                    ; RECOVER MODE NUMBER IN (AL)
340                    ; MOV
341                    ; CX,16
342                    ; LENGTH OF EACH ROW OF TABLE
343                    ; CMP
344                    ; AL,2
345                    ; DETERMINE WHICH ONE TO USE
346                    ; JC
347                    ; M9
348                    ; MODE IS 0 OR 1
349                    ; ADD
350                    ; BX,CX
351                    ; NEXT ROW OF INITIALIZATION TABLE
352                    ; CMP
353                    ; AL,4
354                    ; M9
355                    ; MODE IS 2 OR 3
356                    ; ADD
357                    ; BX,CX
358                    ; MOVE TO GRAPHICS ROW OF INIT_TABLE
359                    ; CMP
360                    ; AL,7
361                    ; M9
362                    ; MODE IS 4,5, OR 6
363                    ; JC
364                    ; ADD
365                    ; BX,CX
366                    ; MOVE TO BW CARD ROW OF INIT_TABLE
367                    ;
368                    ; M9:
369                    ; OUT
370                    ; INIT
371                    ; AX
372                    ; SAVE MODE IN (AL)
373                    ; MOV
374                    ; AX,[BX+10]
375                    ; GET THE CURSOR MODE FROM THE TABLE
376                    ; XCHG
377                    ; AH,AX
378                    ; PUT CURSOR MODE IN CORRECT POSITION
379                    ; PUSH
380                    ; DS
381                    ; SAVE TABLE SEGMENT POINTER
382                    ; ASSUME
383                    ; DS:DATA
384                    ; POINT DS TO DATA SEGMENT
385                    ; CALL
386                    ; DDS
387                    ; PLACE INTO BIOS DATA SAVE AREA
388                    ; MOV
389                    ; *CURSOR_MODE,AX
390                    ; ASSUME
391                    ; DS:CODE
392                    ; POP
393                    ; DS
394                    ; RESTORE THE TABLE SEGMENT POINTER
395                    ; XOR
396                    ; AH,AH
397                    ; AH IS REGISTER NUMBER DURING LOOP
398                    ;
399                    ; M10:
400                    ; LOOP THROUGH TABLE, OUTPUTTING REGISTER ADDRESS, THEN VALUE FROM TABLE
401                    ; MOV
402                    ; AL,AH
403                    ; INITIALIZATION LOOP
404                    ; OUT
405                    ; DX,AL
406                    ; GET 6845 REGISTER NUMBER
407                    ; INC
408                    ; DX
409                    ; POINT TO DATA PORT
410                    ; INC
411                    ; AH
412                    ; NEXT REGISTER VALUE
413                    ; MOV
414                    ; AL,[BX]
415                    ; GET TABLE VALUE
416                    ; OUT
417                    ; DX,AL
418                    ; OUT TO CHIP
419                    ; INC
420                    ; DX
421                    ; NEXT IN TABLE
422                    ; DEC
423                    ; DX
424                    ; BACK TO POINTER REGISTER
425                    ; LOOP
426                    ; M10
427                    ; DO THE WHOLE TABLE
428                    ; POP
429                    ; AX
430                    ; GET MODE BACK INTO (AL)
431                    ; POP
432                    ; DS
433                    ; RECOVER SEGMENT VALUE
434                    ; ASSUME
435                    ; DS:DATA
436                    ;
437                    ; M11:
438                    ; FILL REGEN AREA WITH BLANK
439                    ; MOV
440                    ; DI,DI
441                    ; SET UP POINTER FOR REGEN
442                    ; MOV
443                    ; *CRT_START,DI
444                    ; START ADDRESS SAVED IN GLOBAL
445                    ; MOV
446                    ; *ACTIVE_PAGE,0
447                    ; SET PAGE VALUE
448                    ; MOV
449                    ; CX,8192
450                    ; NUMBER OF WORDS IN COLOR CARD
451                    ; CMP
452                    ; AL,4
453                    ; TEST FOR GRAPHICS

```

```

343 00F0 72 0A          JC      M12           ; NO GRAPHICS_INIT
344 00F2 3C 07          CMP     AL,7         ; TEST FOR BW CARD
345 00F4 74 04          JE      M11         ; BW CARD INIT
346 00F6 33 C0          XOR     AX,AX        ; FILL FOR GRAPHICS MODE
347 00F8 EB 05          JMP     SHORT M13    ; CLEAR BUFFER
348 00FA                M11:   MOV     CH,08H      ; BW CARD INIT
349 00FA BB 08          MOV     CH,08H      ; BUFFER SIZE ON BW CARD (2048)
350 00FC                M12:   MOV     AX,' '*7'H  ; NO GRAPHICS_INIT
351 00FC BB 0720        MOV     AX,' '*7'H  ; FILL CHAR FOR ALPHA + ATTRIBUTE
352 00FF                M13:   REP     STOSW      ; CLEAR BUFFER
353 00FF F3/ AB          REP     STOSW      ; FILL THE REGEN BUFFER WITH BLANKS
354
355                ;----- ENABLE VIDEO AND CORRECT PORT SETTING
356
357 0101 8B 16 0063 R    MOV     DX,@ADDR_6845 ; PREPARE TO OUTPUT TO VIDEO ENABLE PORT
358 0103 83 C2 04        ADD     DX,4         ; POINT TO THE MODE CONTROL REGISTER
359 0105 AD 0065 R      MOV     AL,@CRT_MODE_SET ; GET THE MODE SET VALUE
360 010B EE 0065 R      OUT     DX,AL        ; SET VIDEO ENABLE PORT
361
362                ;----- DETERMINE NUMBER OF COLUMNS, BOTH FOR ENTIRE DISPLAY
363                ;----- AND THE NUMBER TO BE USED FOR TTY INTERFACE
364
365 010C 2E 18 84 0000 E  MOV     AL,CS:[SI + OFFSET M5] ; GET NUMBER OF COLUMNS ON THIS SCREEN
366 0111 78                CMBW   ; CLEAR HIGH BYTE
367 0112 A3 004A R      MOV     @CRT_COLS,AX ; INITIALIZE NUMBER OF COLUMNS COUNT
368
369                ;----- SET CURSOR POSITIONS
370
371 0115 81 E6 000E      AND     SI,000EH    ; WORD OFFSET INTO CLEAR LENGTH TABLE
372 0119 2E 18 84 0000 E  MOV     AX,CS:[SI + OFFSET M5] ; LENGTH TO CLEAR
373 011E A3 004C R      MOV     @CRT_LEN,AX ; SAVE LENGTH OF CRT --- NOT USED FOR BW
374 0121 B9 0008        MOV     CH,8        ; CLEAR ALL CURSOR POSITIONS
375 0124 BF 0050 R      MOV     DI,OFFSET @CURSOR_POSN
376 0127 1E            PUSH   DS           ; ESTABLISH SEGMENT
377 0128 07            POP    ES           ; ADDRESSING
378 0129 33 C0          XOR     AX,AX        ; FILL WITH ZEROS
379 012B F3/ AB          REP     STOSW
380
381                ;----- SET UP OVERSCAN REGISTER
382
383 012D 42            INC     DX           ; SET OVERSCAN PORT TO A DEFAULT
384 012E 80 30          MOV     AL,30H      ; 30H VALUE FOR ALL MODES EXCEPT 640X200
385 0130 83 2E 0049 R 06  CMP     @CRT_MODE,6 ; SEE IF THE MODE IS 640X200 BW
386 0136 75 02          JNZ    M14          ; IF NOT 640X200, THEN GO TO REGULAR
387 0137 B0 3F          MOV     AL,3FH      ; IF IT IS 640X200, THEN PUT IN 3FH
388 0139                M14:   OUT     DX,AL        ; OUTPUT THE CORRECT VALUE TO 3D9 PORT
389 0139 EE            MOV     @CRT_PALETTE,AL ; SAVE THE VALUE FOR FUTURE USE
390 013A A2 0066 R
391
392                ;----- NORMAL RETURN FROM ALL VIDEO RETURNS
393
394 013D                VIDEO_RETURN:
395 013D 5D            POP     BP
396 013E 5F            POP     DI
397 013F 5E            POP     SI
398 0140 5B            POP     BX
399 0141                M15:   POP     CX           ; VIDEO_RETURN_C
400 0141 59            POP     DX
401 0142 5A            POP     DS
402 0143 1F            POP     ES           ; RECOVER SEGMENTS
403 0144 07            POP     ES           ; ALL DONE
404 0146 CF            IRET
405 0146                SET_MODE     ENDP
406
407                ;----- SET_CTYPE
408                ; THIS ROUTINE SETS THE CURSOR VALUE
409                ; INPUT (CX) HAS CURSOR VALUE CH-START LINE, CL-STOP LINE
410                ; OUTPUT (CX) HAS CURSOR VALUE CH-START LINE, CL-STOP LINE
411                ; NONE
412                ;-----
413                ;-----
414 0146                SET_CTYPE   PROC    NEAR
415 0146 B4 0A          MOV     AH,10       ; 6845 REGISTER FOR CURSOR SET
416 0148 89 0E 0060 R  MOV     @CURSOR_MODE,CX ; SAVE IN DATA AREA
417 014C E8 0151 R      CALL    M16         ; OUTPUT CX REGISTER
418 014F EB EC          JMP     VIDEO_RETURN
419
420                ;----- THIS ROUTINE OUTPUTS THE CX REGISTER TO THE 6845 REGISTERS NAMED IN (AH)
421
422                M16:   MOV     DX,@ADDR_6845 ; ADDRESS REGISTER
423 0151                MOV     AL,AH       ; GET VALUE
424 0151 8B 16 0063 R    MOV     DX,AL       ; REGISTER SET
425 0155 8A C4          OUT     DX,AL       ; DATA REGISTER
426 0157 EE            INC     DX           ; I/O DELAY
427 0158 42            JMP     $+2         ; DATA
428 0159 EB 00          MOV     AL,CH       ; I/O DELAY
429 015B 8A C5          OUT     DX,AL       ; DATA
430 015D EE            DEC     DX           ; I/O DELAY
431 015E 4A            JMP     $+2         ; SET FOR SECOND REGISTER
432 015F 8A C4          MOV     AL,AH       ; POINT TO OTHER DATA REGISTER
433 0161 FE C0          OUT     DX,AL       ; SET FOR SECOND REGISTER
434 0163 EE            INC     DX           ; I/O DELAY
435 0164 42            JMP     $+2         ; SECOND DATA VALUE
436 0165 EB 00          MOV     AL,CL       ; I/O DELAY
437 0167 8A C1          OUT     DX,AL       ; SECOND DATA VALUE
438 0169 EE            DEC     DX           ; ALL DONE
439 016A C3            JMP     $+2
440 016B                SET_CTYPE   ENDP
441
442                ;-----
443                ;-----
444                ; SET_CPOS
445                ; THIS ROUTINE SETS THE CURRENT CURSOR POSITION TO THE
446                ; NEW X-Y VALUES PASSED
447                ; INPUT DX - ROW,COLUMN OF NEW CURSOR
448                ; BH - DISPLAY PAGE OF CURSOR
449                ; OUTPUT CURSOR IS SET AT 6845 IF DISPLAY PAGE IS CURRENT DISPLAY
450                ;-----
451                ;-----
452 016B                SET_CPOS   PROC    NEAR
453 016B 8A C7          MOV     AL,BH       ; MOVE PAGE NUMBER TO WORK REGISTER
454 016D 93            CBW                ; CONVERT PAGE TO WORD VALUE
455 016E D1 E0          SAL     AX,1        ; WORD OFFSET
456 0170 96            XCHG   AX,SI       ; USE INDEX REGISTER

```

SECTION 5

```

457 0171 89 94 0050 R      MOV     [SI+OFFSET @CURSOR_POSN],DX      ; SAVE THE POINTER
458 0175 38 3E 0062 R      CMP     @ACTIVE_PAGE,BH
459 0179 76 05              JNZ     M17                               ; SET_CPOS_RETURN
460 017B BB C2              MOV     AX,DX                             ; GET_ROW/COLUMN TO AX
461 017D EB 0182 R          CALL    M18                               ; CURSOR_SET
462 0180                    M17:                                       ; SET_CPOS_RETURN
463 0180 EB BB              SET_CPOS JMP     VIDEO_RETURN
464 0182                    SET_CPOS ENDP
465
466
467
468 0182                    ;----- SET CURSOR POSITION, AX HAS ROW/COLUMN FOR CURSOR
469 0182 EB 0204 R          M18 PROC NEAR
470 0185 BB C8              CALL    POSITION                            ; DETERMINE LOCATION IN REGEN BUFFER
471 0187 03 0E 004E R        ADD     CX,@CRT_START                      ; ADD IN THE START ADDRESS FOR THIS PAGE
472 018B D1 F9              SAR     CX,1                               ; DIVIDE BY 2 FOR CHAR ONLY COUNT
473 018D B4 0E              MOV     AH,14                             ; REGISTER NUMBER FOR CURSOR
474 018F E8 0151 R          CALL    M16                               ; OUTPUT THE VALUE TO THE 6845
475 0192 C3                RET
476 0193                    M18 ENDP
477
478
479
480
481
482
483
484
485
486
487 0193                    READ_CURSOR
488 0193 8A DF              PROC NEAR
489 0195 32 FF              MOV     BL,BH
490 0197 D1 E3              XOR     BH,BH
491 0199 BB 97 0050 R        SAL     BX,1                               ; WORD OFFSET
492 019B BB 0E 0060 R        MOV     DX,[BX+OFFSET @CURSOR_POSN]
493 01A1 5D                POP     BP
494 01A2 5F                POP     DI
495 01A3 5E                POP     SI
496 01A4 5B                POP     BX
497 01A5 58                POP     AX
498 01A6 58                POP     AX
499 01A7 1F                POP     DS
500 01A8 07                POP     ES
501 01A9 CF                IRET
502 01AA                    READ_CURSOR ENDP
503
504
505
506
507
508
509
510
511
512 01AA                    ACT_DISP_PAGE
513 01AA A2 0062 R          PROC NEAR
514 01AD 98 C8              MOV     @ACTIVE_PAGE,AL
515 01AE 50                PUSH   AX
516 01AF F7 26 004C R        MUL     WORD PTR @CRT_LEN
517 01B3 A3 004E R          MOV     @CRT_START,AX
518 01B6 BB C8              MOV     CX,AX
519 01B8 D1 F9              SAR     CX,1
520 01BA B4 0E              MOV     AH,12
521 01BC E8 0151 R          CALL    M16
522 01BF 5B                POP     BX
523 01C0 D1 E3              SAL     BX,1
524 01C2 BB 87 0050 R        MOV     AX,[BX + OFFSET @CURSOR_POSN]
525 01C6 E8 0182 R          CALL    M18
526 01C9 E9 013D R          JMP     VIDEO_RETURN
527 01CC                    ACT_DISP_PAGE ENDP
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544 01CC                    SET_COLOR
545 01CC BB 16 0063 R        MOV     DX,@ADDR_6845
546 01D0 83 C2 06          ADD     DX,6
547 01D3 A0 0066 R          MOV     AL,@CRT_PALETTE
548 01D6 0A FF              OR      BH,BH
549 01D8 75 0E              JNZ     M20
550
551
552
553
554
555
556
557
558
559
560
561
562
563 01E8                    ;----- HANDLE COLOR 0 BY SETTING THE BACKGROUND COLOR
564 01E8 24 E0              AND     AL,0E0H
565 01EA D0 E3 1F          AND     BL,01FH
566 01EC 0A C3              OR      AL,BL
567 01EE 0C 20              OUT     DX,AL
568 01F0 EB EF              MOV     @CRT_PALETTE,AL
569 01F2                    JMP     VIDEO_RETURN
570
571
572
573 01E8                    ;----- HANDLE COLOR 1 BY SELECTING THE PALETTE TO BE USED
574 01E8 24 DF              AND     BL,1
575 01EA D0 E3 1F          SHR     BL,1
576 01EC 73 F3              JNC     M19
577 01EE 0C 20              OR      AL,20H
578 01F0 EB EF              JMP     M19
579 01F2                    SET_COLOR ENDP
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

```

```

571 ;-----
572 ; VIDEO STATE
573 ; RETURNS THE CURRENT VIDEO STATE IN AX
574 ; AH = NUMBER OF COLUMNS ON THE SCREEN
575 ; AL = CURRENT VIDEO MODE
576 ; BH = CURRENT ACTIVE PAGE
577 ;-----
578 VIDEO_STATE PROC NEAR
579 MOV AH, BYTE PTR #CRT_COLS ; GET NUMBER OF COLUMNS
580 MOV AL, #CRT_MODE ; CURRENT MODE
581 MOV BH, #ACTIVE_PAGE ; GET CURRENT ACTIVE PAGE
582 POP BP ; RECOVER REGISTERS
583 POP DI
584 POP SI
585 POP CX ; DISCARD SAVED BX
586 JMP N16 ; RETURN TO CALLER
587 ENDP
588 ;-----
589 ; POSITION
590 ; THIS SERVICE ROUTINE CALCULATES THE REGEN BUFFER ADDRESS
591 ; OF A CHARACTER IN THE ALPHA MODE
592 ; INPUT
593 ; AX = ROW, COLUMN POSITION
594 ; OUTPUT
595 ; AX = OFFSET OF CHAR POSITION IN REGEN BUFFER
596 ;-----
597 POSITION PROC NEAR
598 PUSH BX ; SAVE REGISTER
599 XCHG BX, AX ; SAVE ROW/COLUMN POSITION IN (BX)
600 MOV AL, BYTE PTR #CRT_COLS ; GET COLUMNS PER ROW COUNT
601 MUL BH ; DETERMINE BYTES TO ROW
602 XOR BH, BH
603 ADD AX, BX ; ADD IN COLUMN VALUE
604 MOV SI, 1 ; * 2 FOR ATTRIBUTE BYTES
605 POP BX
606 RET
607 ENDP
608 ;-----
609 ; SCROLL UP
610 ; THIS ROUTINE MOVES A BLOCK OF CHARACTERS UP
611 ; ON THE SCREEN
612 ; INPUT
613 ; (AH) = CURRENT CRT MODE
614 ; (AL) = NUMBER OF ROWS TO SCROLL
615 ; (CX) = ROW/COLUMN OF UPPER LEFT CORNER
616 ; (DX) = ROW/COLUMN OF LOWER RIGHT CORNER
617 ; (BH) = ATTRIBUTE TO BE USED ON BLANKED LINE
618 ; (DS) = DATA SEGMENT
619 ; (ES) = REGEN BUFFER SEGMENT
620 ; OUTPUT
621 ; NONE -- THE REGEN BUFFER IS MODIFIED
622 ;-----
623 ASSUME DS:DATA, ES:DATA
624 SCROLL_UP PROC NEAR
625 CALL TEST_LINE_COUNT
626 CMP AH, 4 ; TEST FOR GRAPHICS MODE
627 JC N1 ; HANDLE SEPARATELY
628 MOV SI, 7 ; TEST FOR BW CARD
629 CMP SI, 7
630 JE N1
631 JMP GRAPHICS_UP
632 N1:
633 PUSH BX ; UP CONTINUE
634 MOV AX, CX ; SAVE FILL ATTRIBUTE IN BH
635 CALL SCROLL_POSITION ; UPPER LEFT POSITION
636 MOV SI, 0 ; DO SETUP FOR SCROLL
637 ADD SI, AX ; BLANK FIELD
638 MOV SI, AX ; FROM ADDRESS
639 SUB SI, AX ; # ROWS IN BLOCK
640 MOV SI, BP ; # ROWS TO BE MOVED
641 CALL N2 ; ROW LOOP
642 ADD SI, BP ; MOVE ONE ROW
643 ADD DI, BP ; POINT TO NEXT LINE IN BLOCK
644 DEC AH ; COUNT OF LINES TO MOVE
645 JNZ N2 ; ROW LOOP
646 N2:
647 POP AX ; CLEAR ENTRY
648 MOV AL, ' ' ; RECOVER ATTRIBUTE IN AH
649 JMP N4 ; CLEAR LOOP
650 N4:
651 CALL N1 ; CLEAR THE ROW
652 ADD DI, BP ; POINT TO NEXT LINE
653 DEC N4 ; COUNTER OF LINES TO SCROLL
654 JNZ N4 ; CLEAR LOOP
655 N5:
656 CALL DDS ; SCROLL_END
657 CMP #CRT_MODE, 7 ; IS THIS THE BLACK AND WHITE CARD
658 JE N6 ; IF SO, SKIP THE MODE RESET
659 MOV AL, #CRT_MODE_SET ; GET THE VALUE OF THE MODE SET
660 OUT DX, AL ; ALWAYS SET COLOR CARD PORT
661 N6:
662 JMP VIDEO_RETURN ; VIDEO_RET_HERE
663 N7:
664 MOV BL, DH ; BLANK FIELD
665 JMP N3 ; GET ROW COUNT
666 N3:
667 SCROLL_UP ENDP
668 ;-----
669 ; HANDLE COMMON SCROLL SET UP HERE
670 ;-----
671 SCROLL_POSITION PROC NEAR
672 CALL POSITION ; CONVERT TO REGEN POINTER
673 ADD AX, #CRT_START ; OFFSET OF ACTIVE PAGE
674 MOV DI, AX ; TO ADDRESS FOR SCROLL
675 MOV SI, AX ; TO ADDRESS FOR SCROLL
676 SUB DX, CX ; DX = #ROWS, #COLS IN BLOCK
677 INC DH ; INCREMENT FOR 0 ORIGIN
678 INC DL ; SET HIGH BYTE OF COUNT TO ZERO
679 MOV BP, #CRT_COLS ; GET NUMBER OF COLUMNS IN DISPLAY
680 ADD BP, BP ; TIMES 2 FOR ATTRIBUTE BYTE
681 MOV AL, BYTE PTR #CRT_COLS ; GET CHARACTERS PER LINE COUNT
682 MUL BL ; DETERMINE OFFSET TO FROM ADDRESS
683 ADD AX, AX ; * 2 FOR ATTRIBUTE BYTE
684 PUSH AX ; SAVE LINE COUNT
    
```

SECTION 5

```

685 0281 A0 0049 R      MOV     AL,@CRT_MODE      ; GET CURRENT MODE
686 0284 06            PUSH    ES                ; ESTABLISH ADDRESSING TO REGEN BUFFER
687 0285 1F            POP     DS                ; FOR BOTH POINTERS
688 0286 3C 02         CMP     AL,2              ; TEST FOR COLOR CARD SPECIAL CASES HERE
689 0288 72 13         JB     AL,3               ; HAVE TO HANDLE 80X25 SEPARATELY
690 028A 3C 03         CMP     AL,3
691 028C 77 0F         JA     N9
692
693 028E 82            I----- PUSH    DX                ; 80X25 COLOR CARD SCROLL
694 028F 8A 03DA       MOV     DX,3DAH          ; GUARANTEED TO BE COLOR CARD HERE
695 0292
696 0292 EC           N8:    IN     AL,DX           ; WAIT_DISP_ENABLE
697 0293 A8 08         TEST   AL,RVRT          ; GET PORT
698 0295 74 FB         JZ     N8                ; WAIT FOR VERTICAL RETRACE
699 0297 B0 25         MOV     AL,25H           ; WAIT_DISP_ENABLE
700 0299 B2 D8         MOV     DL,0D8H          ; ADDRESS CONTROL PORT
701 029B EE            POP     DX,AL            ; TURN OFF VIDEO DURING VERTICAL RETRACE
702 029C 5A            POP     DX
703 029D
704 029D 58           N9:    POP     AX                ; RESTORE LINE COUNT
705 029E 0A DB         OR     BL,BL            ; 0 SCROLL MEANS BLANK FIELD
706 02A0 C3            RET                     ; RETURN WITH FLAGS SET
707 02A1
708
709
710 02A1
711 02A1 8A CA       I----- MOVE_ROW
712 02A3 8A CA       N10:   PROC    NEAR
713 02A4 67           MOV     CL,DL            ; GET # OF COLS TO MOVE
714 02A5 F3 / A5     PUSH    SI               ; SAVE START ADDRESS
715 02A7 8F           REP     MOVSW            ; MOVE THAT LINE ON SCREEN
716 02A8 8E           POP     DI               ; RECOVER ADDRESSES
717 02A9 C3            RET
718 02AA
719
720
721 02AA
722 02AA 8A CA       N11:   PROC    NEAR
723 02AC 67           MOV     CL,DL            ; GET # COLUMNS TO CLEAR
724 02AD F3 / AB     PUSH    DI               ; STORE THE FILL CHARACTER
725 02AF 8F           REP     STOSW
726 02B0 C3            RET
727 02B1
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
    
```

```

799
800
801 PAGE
802 ; READ_AC_CURRENT
803 ; THIS ROUTINE READS THE ATTRIBUTE AND CHARACTER AT THE CURRENT
804 ; CURSOR POSITION AND RETURNS THEM TO THE CALLER
805 ; INPUT
806 ; (AH) = CURRENT CRT MODE
807 ; (BH) = DISPLAY PAGE ( ALPHA MODES ONLY )
808 ; (DS) = DATA SEGMENT
809 ; (ES) = REGEN SEGMENT
810 ; OUTPUT
811 ; (AL) = CHARACTER READ
812 ; (AH) = ATTRIBUTE READ
813 -----
814 ASSUME DS:DATA,ES:DATA
815
816 READ_AC_CURRENT PROC NEAR
817   CMP AH,4 ; IS THIS GRAPHICS
818   JC P10
819   CMP AH,7 ; IS THIS BW CARD
820   JE P10
821   JMP GRAPHICS_READ
822 P10: CALL FIND_POSITION ; READ AC CONTINUE
823   MOV SI,DI ; GET REGEN LOCATION AND PORT ADDRESS
824   PUSH ES ; ESTABLISH ADDRESSING IN SI
825   POP DS ; GET REGEN SEGMENT FOR QUICK ACCESS
826
827 ;----- WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
828
829   OR BL,BL ; CHECK MODE FLAG FOR COLOR CARD IN 80
830   JNZ P11 ; ELSE SKIP RETRACE WAIT - DO FAST READ
831   STI ; WAIT FOR HORZ RETRACE LOW OR VERTICAL
832   NOP ; ENABLE INTERRUPTS FIRST
833   CLI ; ALLOW FOR SMALL INTERRUPT WINDOW
834   IN AL,DX ; BLOCK INTERRUPTS FOR SINGLE LOOP
835   TEST AL,RHRZ ; GET STATUS FROM THE ADAPTER
836   JNZ P11 ; IS HORIZONTAL RETRACE LOW
837   IN AL,DX ; WAIT UNTIL IT IS
838   TEST AL,RVRT+RHRZ ; NOW WAIT FOR EITHER RETRACE HIGH
839   JZ P12 ; GET STATUS
840   IN AL,DX ; IS HORIZONTAL OR VERTICAL RETRACE HIGH
841   TEST AL,RVRT+RHRZ ; WAIT UNTIL EITHER IS ACTIVE
842   JZ P13
843   LODSW VIDEO_RETURN ; GET THE CHARACTER AND ATTRIBUTE
844   JMP P10 ; EXIT WITH (AX)
845
846 READ_AC_CURRENT ENDP
847
848
849
850
851
852 FIND_POSITION PROC NEAR
853   XCHG AH,BL ; SETUP FOR BUFFER READ OR WRITE
854   MOV BP,AX ; SWAP MODE TYPE WITH ATTRIBUTE
855   SUB BL,2 ; SAVE CHARACTER/ATTR IN (BP) REGISTER
856   SHR BL,1 ; CONVERT DISPLAY MODE TYPE TO A
857   MOV AL,BH ; ZERO VALUE FOR COLOR IN 80 COLUMN
858   CBW ; MOVE DISPLAY PAGE TO LOW BYTE
859   MOV DI,AX ; CLEAR HIGH BYTE FOR BYTE OFFSET
860   MOV DX,[DI+OFFSET #CURSOR_POSN] ; MOVE DISPLAY PAGE (COUNT) TO WORK REG
861   JZ P20 ; TIMES 2 FOR WORD OFFSET
862   XOR DI,DI ; GET ROW/COLUMN OF THAT PAGE
863   ; SKIP BUFFER ADJUSTMENT IF PAGE ZERO
864   ; ELSE SET BUFFER START ADDRESS TO ZERO
865   ADD DI,#CRT_LEN ; ADD LENGTH OF BUFFER FOR ONE PAGE
866   DEC AX ; DECREMENT PAGE COUNT
867   JNZ P20 ; LOOP TILL PAGE COUNT EXHAUSTED
868
869   MOV AL, BYTE PTR #CRT_COLS ; DETERMINE LOCATION IN REGEN IN PAGE
870   MUL DH ; GET COLUMNS PER ROW COUNT
871   XOR DH,DH ; DETERMINE BYTES TO ROW
872   ADD AX,DX ; ADD IN COLUMN VALUE
873   SAL AX,1 ; * 2 FOR ATTRIBUTE BYTES
874   ADD DI,AX ; ADD LOCATION TO START OF REGEN PAGE
875   MOV DX,#ADDR_6845 ; GET BASE ADDRESS OF ACTIVE DISPLAY
876   ADD DI,DX ; DX= STATUS PORT ADDRESS OF ADAPTER
877   RET ; BP= ATTRIBUTE/CHARACTER (FROM BL/AL)
878   ; DI= POSITION (OFFSET IN REGEN BUFFER)
879   ; BL= MODE FLAG (ZERO FOR 80X25 COLOR)
880
881 FIND_POSITION ENDP

```

SECTION 5

```

882                                     PAGE
883                                     -----
884                                     ; WRITE_AC_CURRENT
885                                     ; THIS ROUTINE WRITES THE ATTRIBUTE AND CHARACTER
886                                     ; AT THE CURRENT CURSOR POSITION
887                                     ; INPUT
888                                     ; (AH) = CURRENT CRT MODE
889                                     ; (BH) = DISPLAY PAGE
890                                     ; (CX) = COUNT OF CHARACTERS TO WRITE
891                                     ; (AL) = CHAR TO WRITE
892                                     ; (BL) = ATTRIBUTE OF CHAR TO WRITE
893                                     ; (DS) = DATA SEGMENT
894                                     ; (ES) = REGEN SEGMENT
895                                     ; OUTPUT
896                                     ; DISPLAY REGEN BUFFER UPDATED
897                                     -----
898
899 0360 WRITE_AC_CURRENT PROC NEAR
900 0360 80 FC 04 CMP AH,4 ; IS THIS GRAPHICS
901 0363 72 08 JC P30
902 0366 80 FC 07 CMP AH,7 ; IS THIS BW CARD
903 0368 74 03 JE P30
904 036A E9 058E R JMP GRAPHICS_WRITE
905 036D CALL FIND_POSITION ; WRITE AC CONTINUE
906 036E E8 032C R ; GET REGEN LOCATION AND PORT ADDRESS
907 ; ADDRESS IN (DI) REGISTER
908 0370 0A DB OR BL,BL ; CHECK MODE FLAG FOR COLOR CARD AT 80
909 0372 74 06 JZ P32 ; SKIP TO RETRACE WAIT IF COLOR AT 80
910
911 0374 95 XCHG AX,BP ; GET THE ATTR/CHAR SAVED FOR FAST WRITE
912 0375 F3 AB REP STOSW ; STRING WRITE THE ATTRIBUTE & CHARACTER
913 0377 EB 16 JMP SHORT P35 ; EXIT FAST WRITE ROUTINE
914
915 ;----- WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
916
917 0379 P31: XCHG BP,AX ; LOOP FOR EACH ATTR/CHAR WRITE
918 0379 95 ; PLACE ATTR/CHAR BACK IN SAVE REGISTER
919 037A STI ; WAIT FOR HORZ RETRACE LOW OR VERTICAL
920 037A FB ; ENABLE INTERRUPTS FIRST
921 037B 90 NOP ; ALLOW FOR INTERRUPT WINDOW
922 037C FA CLI ; BLOCK INTERRUPTS FOR SINGLE LOOP
923 037D EC IN AL,DX ; GET STATUS FROM THE ADAPTER
924 037E A8 08 TEST AL,RVRT ; CHECK FOR VERTICAL RETRACE FIRST
925 0380 75 09 JNZ P34 ; DO FAST WRITE NOW IF VERTICAL RETRACE
926 0382 A8 01 TEST AL,RHRZ ; IS HORIZONTAL RETRACE LOW THEN
927 0384 75 F4 JNZ P32 ; WAIT UNTIL IT IS
928 0386 P33: IN AL,DX ; WAIT FOR EITHER RETRACE HIGH
929 0386 EC ; GET STATUS AGAIN
930 0387 A8 09 TEST AL,RVRT+RHRZ ; IS HORIZONTAL OR VERTICAL RETRACE HIGH
931 0389 74 FB JZ P34 ; WAIT UNTIL EITHER IS ACTIVE
932 038B P34: XCHG AX,BP ; GET THE ATTR/CHAR SAVED IN (BP)
933 038B 95 ; WRITE THE ATTRIBUTE AND CHARACTER
934 038C AB STOSW ; AS MANY TIMES AS REQUESTED - TILL CX=0
935 038D E2 EA LOOP P31
936 038F P35: JMP VIDEO_RETURN ; EXIT
937 038F E9 013D R
938
939 0392 WRITE_AC_CURRENT ENDP
940
941
942                                     ; WRITE_C_CURRENT
943                                     ; THIS ROUTINE WRITES THE CHARACTER AT
944                                     ; THE CURRENT CURSOR POSITION, ATTRIBUTE UNCHANGED
945                                     ; INPUT
946                                     ; (AH) = CURRENT CRT MODE
947                                     ; (BH) = DISPLAY PAGE
948                                     ; (CX) = COUNT OF CHARACTERS TO WRITE
949                                     ; (AL) = CHAR TO WRITE
950                                     ; (DS) = DATA SEGMENT
951                                     ; (ES) = REGEN SEGMENT
952                                     ; OUTPUT
953                                     ; DISPLAY REGEN BUFFER UPDATED
954                                     -----
955
956 0392 WRITE_C_CURRENT PROC NEAR
957 0392 80 FC 04 CMP AH,4 ; IS THIS GRAPHICS
958 0395 72 08 JC P40
959 0397 80 FC 07 CMP AH,7 ; IS THIS BW CARD
960 039A 74 03 JE P40
961 039C E9 058E R JMP GRAPHICS_WRITE
962 039F CALL FIND_POSITION ; GET REGEN LOCATION AND PORT ADDRESS
963 039F E8 032C R ; ADDRESS OF LOCATION IN (DI)
964
965 ;----- WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
966
967 03A2 P41: STI ; WAIT FOR HORZ RETRACE LOW OR VERTICAL
968 03A2 FB ; ENABLE INTERRUPTS FIRST
969 03A3 0A DB OR BL,BL ; CHECK MODE FLAG FOR COLOR CARD IN 80
970 03A3 78 0F JNZ P43 ; ELSE SKIP RETRACE WAIT - DO FAST WRITE
971 03A5 78 0F JNZ P43 ; ELSE SKIP RETRACE WAIT - DO FAST WRITE
972 03A7 FA CLI ; BLOCK INTERRUPTS FOR SINGLE LOOP
973 03A8 EC IN AL,DX ; GET STATUS FROM THE ADAPTER
974 03A9 A8 08 TEST AL,RVRT ; CHECK FOR VERTICAL RETRACE FIRST
975 03AB 75 09 JNZ P43 ; DO FAST WRITE NOW IF VERTICAL RETRACE
976 03AD A8 01 TEST AL,RHRZ ; IS HORIZONTAL RETRACE LOW THEN
977 03AF 75 F1 JNZ P41 ; WAIT UNTIL IT IS
978 03B1 P42: IN AL,DX ; WAIT FOR EITHER RETRACE HIGH
979 03B1 EC ; GET STATUS AGAIN
980 03B2 A8 09 TEST AL,RVRT+RHRZ ; IS HORIZONTAL OR VERTICAL RETRACE HIGH
981 03B4 74 FB JZ P42 ; WAIT UNTIL EITHER RETRACE ACTIVE
982 03B6 P43: MOV AX,BP ; GET THE CHARACTER SAVE IN (BP)
983 03B6 8B C5 ; PUT THE CHARACTER INTO REGEN BUFFER
984 03B8 AA STOSB ; BUMP POINTER PAST ATTRIBUTE
985 03B9 47 INC DI ; AS MANY TIMES AS REQUESTED
986 03BA E2 E6 LOOP P41
987
988 03BC E9 013D R JMP VIDEO_RETURN
989
990 03BF WRITE_C_CURRENT ENDP

```

```

991 PAGE
992 -----
993 | WRITE_STRING
994 | THIS ROUTINE WRITES A STRING OF CHARACTERS TO THE CRT.
995 |
996 | INPUT
997 | (AL) = WRITE STRING COMMAND 0 - 3
998 | (BH) = DISPLAY PAGE (ACTIVE PAGE)
999 | (CX) = COUNT OF CHARACTERS TO WRITE, IF (CX) = 0 THEN RETURN
1000 | (DX) = CURSOR POSITION FOR START OF STRING WRITE
1001 | (BL) = ATTRIBUTE OF CHARACTER TO WRITE IF (AL) = 0 OR (AL) = 1
1002 | (BP) = SOURCE STRING OFFSET
1003 | (OE) = SOURCE STRING SEGMENT (FOR USE IN (ES) IN STACK +14)
1004 |
1005 | OUTPUT
1006 | NONE
1007 |-----
1008 WRITE_STRING PROC NEAR
1009     PUSH BP
1010     MOV BP,SP
1011     MOV ES:[BP]+14+2
1012     POP BP
1013     CBW
1014     MOV DI,AX
1015     CMP AL,04
1016     JNB P59
1017     JCXZ P59
1018     MOV SI,BX
1019     MOV BL,BH
1020     XOR BH,BH
1021     XCHG SI,BX
1022     SAL SI,1
1023     PUSH [SI+OFFSET *CURSOR_POSN]
1024     MOV AX,0200H
1025     INT 10H
1026     P50:
1027     MOV AL,ES:[BP]
1028     INC BP
1029     I----- TEST FOR SPECIAL CHARACTER'S
1030
1031     CMP AL,08H
1032     JE P51
1033     CMP AL,09H
1034     JE P51
1035     CMP AL,0AH
1036     JE P51
1037     CMP AL,0DH
1038     JE P51
1039     CMP AL,0BH
1040     JNE P51
1041     MOV AH,0EH
1042     INT 10H
1043     MOV DX,[SI+OFFSET *CURSOR_POSN]
1044     JMP SHORT P54
1045
1046     P51:
1047     PUSH CX
1048     PUSH BX
1049     MOV CX,1
1050     CMP DI,2
1051     JB P53
1052     MOV BL,ES:[BP]
1053     INC BP
1054     P53:
1055     MOV AM,09H
1056     INT 10H
1057     POP BX
1058     POP DX
1059     INC DL
1060     CMP DL, BYTE PTR *CRT_COLS
1061     JB P54
1062     INC DH
1063     SUB DL,DL
1064     CMP DH,25
1065     JB P54
1066
1067     MOV AX,0E0AH
1068     INT 10H
1069     DEC DH
1070     P54:
1071     MOV AX,0200H
1072     INT 10H
1073     LOOP P50
1074
1075     POP DX
1076     XCHG AX,DI
1077     TEST AL,01H
1078     JNZ P59
1079     MOV AX,0200H
1080     INT 10H
1081     P59:
1082     JMP VIDEO_RETURN
1083
1084 WRITE_STRING ENDP
  
```

```

1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103 0443
1104 0443 E8 0477 R
1105 0446 26 8A 04
1106 0449 22 C4
1107 044B D2 E0
1108 044D 8A CE
1109 044F D2 C0
1110 0451 E9 013D R
1111 0454
1112
1113 0454
1114 0454 50
1115 0455 50
1116 0456 E8 0477 R
1117 0459 D2 E8
1118 045B 22 C4
1119 045D 26 8A 0C
1120 0460 5B
1121 0461 F6 C3 80
1122 0464 75 0D
1123 0466 F6 D4
1124 0468 22 CC
1125 046A 0A C1
1126 046C
1127 046C 26 88 04
1128 046F 58
1129 0470 E9 013D R
1130 0473
1131 0473 32 C1
1132 0475 EB F5
1133 0477
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147 0477
1148
1149
1150
1151
1152 0477 96
1153 0478 B0 28
1154 047A F6 E2
1155 047C A8 08
1156 047E 74 03
1157 0480 05 1FD8
1158 0483
1159 0483 96
1160 0484 8B D1
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170 0486 B8 02C0
1171 0489 B9 0302
1172 048C 8D 3E 0049 R 06
1173 0491 T2 06
1174 0493 B8 0180
1175 0496 B9 0703
1176
1177
1178 0499
1179 0499 22 EA
1180
1181
1182
1183 049B D3 EA
1184 049D D3 F2
1185 049F 8A F7
1186
1187
1188
1189 04A1 2A C9
1190 04A3
1191 04A3 D0 C8
1192 04A5 D2 CD
1193 04A7 FE CF
1194 04A9 75 F6
1195 04AB 8A E3
1196 04AD D2 EC
1197 04AF C3
1198 04B0

```

```

PAGE
-----
; READ DOT -- WRITE DOT
; THESE ROUTINES WILL WRITE A DOT, OR READ THE
; DOT AT THE INDICATED LOCATION
; ENTRY --
; DX = ROW (0-199) (THE ACTUAL VALUE DEPENDS ON THE MODE)
; CX = COLUMN (0-639) (THE VALUES ARE NOT RANGE CHECKED)
; AL = DOT VALUE TO WRITE (1,2 OR 4 BITS DEPENDING ON MODE,
; REQUIRED FOR WRITE DOT ONLY, RIGHT JUSTIFIED)
; BIT 7 OF AL = 1 INDICATES XOR THE VALUE INTO THE LOCATION
; DS = DATA SEGMENT
; ES = REGEN SEGMENT
; EXIT
; AL = DOT VALUE READ, RIGHT JUSTIFIED, READ ONLY
-----
ASSUME DS:DATA,ES:DATA
READ_DOT PROC NEAR
CALL R3
MOV AL,ES:[SI]
AND AL,AH
SHL AL,CL
MOV CL,DH
ROL AL,CL
JMP VIDEO_RETURN
ENDP
WRITE_DOT PROC NEAR
PUSH AX
PUSH AX
CALL R3
SHR AL,CL
AND AL,AH
MOV CL,ES:[SI]
POP BX
TEST BL,80H
JNZ R2
NOT AH
AND CL,AH
OR AL,CL
MOV ES:[SI],AL
POP AX
JMP VIDEO_RETURN
XOR AL,CL
JMP R1
ENDP
; THIS SUBROUTINE DETERMINES THE REGEN BYTE LOCATION OF THE
; INDICATED ROW COLUMN VALUE IN GRAPHICS MODE.
; ENTRY
; DX = ROW VALUE (0-199)
; CX = COLUMN VALUE (0-639)
; EXIT
; SI = OFFSET INTO REGEN BUFFER FOR BYTE OF INTEREST
; AH = MASK TO STRIP OFF THE BITS OF INTEREST
; CL = BITS TO SHIFT TO RIGHT JUSTIFY THE MASK IN AH
; DH = # BITS IN RESULT
; BX = MODIFIED
-----
R3 PROC NEAR
;----- DETERMINE 1ST BYTE IN INDICATED ROW BY MULTIPLYING ROW VALUE BY 40
; ( LOW BIT OF ROW DETERMINES EVEN/ODD, 80 BYTES/ROW )
XCHG SI,AX
MOV AX,40
MUL DX
TEST AL,008H
JZ R4
ADD AX,2000H-40
R4: XCHG SI,AX
MOV DX,CX
;----- DETERMINE GRAPHICS MODE CURRENTLY IN EFFECT
; SET UP THE REGISTERS ACCORDING TO THE MODE
; CH = MASK FOR LOW OF COLUMN ADDRESS ( 7/3 FOR HIGH/MED RES )
; CL = # OF ADDRESS BITS IN COLUMN VALUE ( 3/2 FOR H/M )
; BL = MASK TO SELECT BITS FROM POINTED BYTE ( 80H/COH FOR H/M )
; BH = NUMBER OF VALID BITS IN POINTED BYTE ( 1/2 FOR H/M )
MOV BX,2C0H
MOV CX,302H
CMP CRT_MODE,6
JC R5
MOV BX,180H
MOV CX,703H
;----- DETERMINE BIT OFFSET IN BYTE FROM COLUMN MASK
R5: AND CH,DL
;----- DETERMINE BYTE OFFSET FOR THIS LOCATION IN COLUMN
SHR DX,CL
ADD SI,DX
MOV DH,BH
;----- MULTIPLY BH (VALID BITS IN BYTE) BY CH (BIT OFFSET)
SUB CL,CL
R6: ROR AL,-1
ADD CL,CH
DEC BH
JNZ R6
MOV AH,BL
SHR AH,CL
RET
R3 ENDP

```

```

1199
1200 ; SCROLL UP
1201 ; THIS ROUTINE SCROLLS UP THE INFORMATION ON THE CRT
1202 ; ENTRY ---
1203 ; CH,CL = UPPER LEFT CORNER OF REGION TO SCROLL
1204 ; DH,DL = LOWER RIGHT CORNER OF REGION TO SCROLL
1205 ; BOTH OF THE ABOVE ARE IN CHARACTER POSITIONS
1206 ; BH = FILL VALUE FOR BLANKED LINES
1207 ; AL = # LINES TO SCROLL (AL=0 MEANS BLANK THE ENTIRE FIELD)
1208 ; DS = DATA SEGMENT
1209 ; ES = REGEN SEGMENT
1210 ; EXIT ---
1211 ; NOTHING, THE SCREEN IS SCROLLED
1212
1213 04B0 PROC NEAR
1214 04B0 8A D8 MOV BL,AL ; SAVE LINE COUNT IN BL
1215 04B2 8B C1 MOV AX,CX ; GET UPPER LEFT POSITION INTO AX REG
1216
1217 ;----- USE CHARACTER SUBROUTINE FOR POSITIONING
1218 ;----- ADDRESS RETURNED IS MULTIPLIED BY 2 FROM CORRECT VALUE
1219
1220 04B4 E8 06F0 R CALL GRAPH_POSN ; SAVE RESULT AS DESTINATION ADDRESS
1221 04B7 8B F8 MOV DI,AX
1222
1223 ;----- DETERMINE SIZE OF WINDOW
1224
1225 04B9 2B D1 SUB DX,CX
1226 04BB 81 C2 0101 ADD DX,101H ; ADJUST VALUES
1227 04BF D0 E6 SAL DH,1 ; MULTIPLY ROWS BY 4 AT 8 VERT DOTS/CHAR
1228 04C1 D0 E6 SAL DH,1 ; AND EVEN/ODD ROWS
1229
1230 ;----- DETERMINE CRT MODE
1231
1232 04C3 80 3E 0049 R 06 CMP @CRT_MODE,6 ; TEST FOR MEDIUM RES
1233 04C5 73 04 JNC RT ; FIND_SOURCE
1234
1235 ;----- MEDIUM RES UP
1236 04CA D0 E2 SAL DI,1 ; # COLUMNS * 2, SINCE 2 BYTES/CHAR
1237 04CC D1 E7 SAL DI,1 ; OFFSET *2 SINCE 2 BYTES/CHAR
1238
1239 ;----- DETERMINE THE SOURCE ADDRESS IN THE BUFFER
1240 04CE RT: ; FIND_SOURCE
1241 04CE 06 PUSH ES ; GET SEGMENTS BOTH POINTING TO REGEN
1242 04CF 1F POP DS
1243 04D0 2A ED SUB CH,CH
1244 04D2 D0 E3 SAL BL,1 ; MULTIPLY NUMBER OF LINES BY 4
1245 04D4 D0 E3 SAL BL,1
1246 04D6 74 2B JZ R11 ; IF ZERO, THEN BLANK ENTIRE FIELD
1247 04D8 80 50 MOV AL,80
1248 04DA F9 E3 MUL BL ; DETERMINE OFFSET TO SOURCE
1249 04DC 8B F7 MOV SI,DI ; SET UP SOURCE
1250 04DE 03 F0 ADD SI,AX ; ADD IN OFFSET TO IT
1251 04E0 3A E4 MOV AH,DH ; NUMBER OF ROWS IN FIELD
1252 04E2 2A E3 SUB AH,BL ; DETERMINE NUMBER TO MOVE
1253
1254 ;----- LOOP THROUGH, MOVING ONE ROW AT A TIME, BOTH EVEN AND ODD FIELDS
1255 04E4 R8: ; ROW_LOOP
1256 04E4 E8 0564 R CALL R17 ; MOVE ONE ROW
1257 04E7 81 EE 1FB0 SUB SI,2000H-80 ; MOVE TO NEXT ROW
1258 04E9 81 EF 1FB0 SUB DI,2000H-80
1259 04EF FE CC DEC AH ; NUMBER OF ROWS TO MOVE
1260 04F1 75 F1 JNZ R8 ; CONTINUE TILL ALL MOVED
1261
1262 ;----- FILL IN THE VACATED LINE(S)
1263 04F3 R9: ; CLEAR_ENTRY
1264 04F3 8A C7 MOV AL,BH ; ATTRIBUTE TO FILL WITH
1265 04F5 R10:
1266 04F5 E8 057D R CALL R18 ; CLEAR THAT ROW
1267 04F8 81 EF 1FB0 SUB DI,2000H-80 ; POINT TO NEXT LINE
1268 04FC FE CB DEC BL ; NUMBER OF LINES TO FILL
1269 04FE 75 F5 JNZ R10 ; CLEAR_LOOP
1270 0500 E9 013D R JMP VIDEO_RETURN ; EVERYTHING DONE
1271
1272 0503 R11: ; BLANK FIELD
1273 0503 8A DE MOV BL,DH ; SET BLANK COUNT TO EVERYTHING IN FIELD
1274 0505 EB EC JMP R9 ; CLEAR THE FIELD
1275 0507 GRAPHICS_UP ENDP
1276
1277 ; SCROLL DOWN
1278 ; THIS ROUTINE SCROLLS DOWN THE INFORMATION ON THE CRT
1279 ; ENTRY ---
1280 ; CH,CL = UPPER LEFT CORNER OF REGION TO SCROLL
1281 ; DH,DL = LOWER RIGHT CORNER OF REGION TO SCROLL
1282 ; BOTH OF THE ABOVE ARE IN CHARACTER POSITIONS
1283 ; BH = FILL VALUE FOR BLANKED LINES
1284 ; AL = # LINES TO SCROLL (AL=0 MEANS BLANK THE ENTIRE FIELD)
1285 ; DS = DATA SEGMENT
1286 ; ES = REGEN SEGMENT
1287 ; EXIT ---
1288 ; NOTHING, THE SCREEN IS SCROLLED
1289
1290
1291 0507 PROC NEAR
1292 0507 FD STD ; SET DIRECTION
1293 0508 8A D8 MOV BL,AL ; SAVE LINE COUNT IN BL
1294 050A 8B C2 MOV AX,DX ; GET LOWER RIGHT POSITION INTO AX REG
1295
1296 ;----- USE CHARACTER SUBROUTINE FOR POSITIONING
1297 ;----- ADDRESS RETURNED IS MULTIPLIED BY 2 FROM CORRECT VALUE
1298
1299 050C E8 06F0 R CALL GRAPH_POSN ; SAVE RESULT AS DESTINATION ADDRESS
1300 050F 8B F8 MOV DI,AX
1301
1302 ;----- DETERMINE SIZE OF WINDOW
1303
1304 0511 2B D1 SUB DX,CX
1305 0513 81 C2 0101 ADD DX,101H ; ADJUST VALUES
1306 0517 D0 E6 SAL DH,1 ; MULTIPLY ROWS BY 4 AT 8 VERT DOTS/CHAR
1307 0519 D0 E6 SAL DH,1 ; AND EVEN/ODD ROWS
1308
1309 ;----- DETERMINE CRT MODE
1310
1311 051B 80 3E 0049 R 06 CMP @CRT_MODE,6 ; TEST FOR MEDIUM RES
1312 0520 73 05 JNC R12 ; FIND_SOURCE_DOWN
    
```

SECTION 5

```

1313
1314
1315 0522 00 E2 ;----- MEDIUM RES DOWN
1316 0524 01 E7 SAL DL,1 ; # COLUMNS * 2, SINCE 2 BYTES/CHAR
1317 0526 47 INC DI ; OFFSET *2 SINCE 2 BYTES/CHAR
1318 ; POINT TO LAST BYTE
1319
1320 ;----- DETERMINE THE SOURCE ADDRESS IN THE BUFFER
1321 0527 R12: ; FIND_SOURCE_DOWN
1322 0527 06 PUSH ES ; BOTH SEGMENTS TO REGEN
1323 0528 1F POP DS
1324 0529 2A ED SUB CH,CH ; ZERO TO HIGH OF COUNT REGISTER
1325 052B 81 C7 00F0 ADD DI,240 ; POINT TO LAST ROW OF PIXELS
1326 052F 00 E3 SAL BL,1 ; MULTIPLY NUMBER OF LINES BY 4
1327 0531 00 E3 SAL BL,1
1328 0533 74 2B JZ R16 ; IF ZERO, THEN BLANK ENTIRE FIELD
1329 0535 90 50 MOV AL,80 ; 80 BYTES/ROW
1330 0537 F6 E3 MVL BL ; DETERMINE OFFSET TO SOURCE
1331 0539 8B F7 MOV SI,DI ; SET UP SOURCE
1332 053B 2B F0 SUB SI,AX ; SUBTRACT THE OFFSET
1333 053D 8A E6 MOV AH,DH ; NUMBER OF ROWS IN FIELD
1334 053F 2A E3 SUB AH,BL ; DETERMINE NUMBER TO MOVE
1335
1336 ;----- LOOP THROUGH, MOVING ONE ROW AT A TIME, BOTH EVEN AND ODD FIELDS
1337
1338 0541 R13: ; ROW LOOP DOWN
1339 0541 E6 0564 R ; MOVE ONE ROW
1340 0544 81 EE 2050 SUB SI,2000H+80 ; MOVE TO NEXT ROW
1341 0548 81 EF 2050 SUB DI,2000H+80
1342 054C FE CC DEC AH ; NUMBER OF ROWS TO MOVE
1343 054E 75 F1 JNZ R13 ; CONTINUE TILL ALL MOVED
1344
1345 ;----- FILL IN THE VACATED LINE(S)
1346 0550 R14: ; CLEAR_ENTRY_DOWN
1347 0550 8A C7 MOV AL,BH ; ATTRIBUTE TO FILL WITH
1348 0552 R15: ; CLEAR_LOOP_DOWN
1349 0552 E8 057D R CALL R18 ; CLEAR A ROW
1350 0555 81 EF 2050 SUB DI,2000H+80 ; POINT TO NEXT LINE
1351 0559 FE CB DEC BL ; NUMBER OF LINES TO FILL
1352 055B 75 F5 JNZ R15 ; CLEAR_LOOP_DOWN
1353
1354 0560 E9 013D R JMP VIDEO_RETURN ; EVERYTHING DONE
1355
1356 0560 R16: ; BLANK_FIELD_DOWN
1357 0560 8A DE MOV BL,DH ; SET BLANK COUNT TO EVERYTHING IN FIELD
1358 0562 EB EC JMP R14 ; CLEAR THE FIELD
1359 0564 GRAPHICS_DOWN ENDP
1360
1361 ;----- ROUTINE TO MOVE ONE ROW OF INFORMATION
1362
1363 0564 R17 PROC NEAR
1364 0564 8A CA MOV CL,DL ; NUMBER OF BYTES IN THE ROW
1365 0566 B6 PUSH SI
1366 0567 57 PUSH DI ; SAVE POINTERS
1367 0568 F3 A4 REP MOVSB ; MOVE THE EVEN FIELD
1368 056A 8F POP AH
1369 056B 5E POP SI
1370 056C 81 C6 2000 ADD SI,2000H ; POINT TO THE ODD FIELD
1371 0570 81 C7 2000 ADD DI,2000H
1372 0574 B6 PUSH SI
1373 0575 57 PUSH DI ; SAVE THE POINTERS
1374 0576 8A CA MOV CL,DL ; COUNT BACK
1375 057B F3 A4 REP MOVSB ; MOVE THE ODD FIELD
1376 057A 5F POP DI
1377 057B 5E POP SI ; POINTERS BACK
1378 057C C3 RET ; RETURN TO CALLER
1379 057D R17 ENDP
1380
1381 ;----- CLEAR A SINGLE ROW
1382
1383 057D R18 PROC NEAR
1384 057D 8A CA MOV CL,DL ; NUMBER OF BYTES IN FIELD
1385 057F 57 PUSH DI ; SAVE POINTER
1386 0580 F3 AA REP STOSB ; STORE THE NEW VALUE
1387 0582 5F POP DI ; POINTER BACK
1388 0583 81 C7 2000 ADD DI,2000H ; POINT TO ODD FIELD
1389 0587 57 PUSH DI
1390 0588 8A CA MOV CL,DL
1391 058A F3 AA REP STOSB ; FILL THE ODD FIELD
1392 058C 5F POP DI ; RETURN TO CALLER
1393 058D C3 RET
1394 058E R18 ENDP
1395
1396 ;-----
1397 ; GRAPHICS WRITE
1398 ; THIS ROUTINE WRITES THE ASCII CHARACTER TO THE CURRENT
1399 ; POSITION ON THE SCREEN.
1400 ; ENTRY --
1401 ; AL = CHARACTER TO WRITE
1402 ; BL = COLOR ATTRIBUTE TO BE USED FOR FOREGROUND COLOR
1403 ; IF BIT 7 IS SET, THE CHAR IS XOR'D INTO THE REGEN BUFFER
1404 ; (0 IS USED FOR THE BACKGROUND COLOR)
1405 ; CX = NUMBER OF CHARS TO WRITE
1406 ; DS = DATA SEGMENT
1407 ; ES = REGEN SEGMENT
1408 ; EXIT --
1409 ; NOTHING IS RETURNED
1410 ;
1411 ; GRAPHICS READ
1412 ; THIS ROUTINE READS THE ASCII CHARACTER AT THE CURRENT CURSOR
1413 ; POSITION ON THE SCREEN BY MATCHING THE DOTS ON THE SCREEN TO THE
1414 ; CHARACTER GENERATOR CODE POINTS
1415 ; ENTRY --
1416 ; NONE (0 IS ASSUMED AS THE BACKGROUND COLOR)
1417 ; EXIT --
1418 ; AL = CHARACTER READ AT THAT POSITION (0 RETURNED IF NONE FOUND)
1419 ;
1420 ; FOR BOTH ROUTINES, THE IMAGES USED TO FORM CHARS ARE CONTAINED IN ROM
1421 ; FOR THE 128 CHARS. TO ACCESS CHARS IN THE SECOND HALF, THE USER
1422 ; MUST INITIALIZE THE VECTOR AT INTERRUPT 1FH (LOCATION 0007H) TO
1423 ; POINT TO THE USER SUPPLIED TABLE OF GRAPHIC IMAGES (8X8 BOXES).
1424 ; FAILURE TO DO SO WILL CAUSE IN STRANGE RESULTS
1425 ;-----
1426

```

```

1427          ASSUME DS:DATA,ES:DATA
1428 058E      GRAPHICS WRITE PROC NEAR
1429 058E B4 00      MOV AH,0          ; ZERO TO HIGH OF CODE POINT
1430 0590 50      PUSH AX          ; SAVE CODE POINT VALUE
1431
1432          ;----- DETERMINE POSITION IN REGEN BUFFER TO PUT CODE POINTS
1433
1434 0591 E8 06ED R    CALL 526          ; FIND LOCATION IN REGEN BUFFER
1435 0594 8B F8      MOV DI,AX        ; REGEN POINTER IN DI
1436
1437          ;----- DETERMINE REGION TO GET CODE POINTS FROM
1438
1439 0596 58      POP AX          ; RECOVER CODE POINT
1440 0597 3C 80      CMP AL,80H      ; IS IT IN SECOND HALF
1441 0599 73 06      JAE S1         ; YES
1442
1443          ;----- IMAGE IS IN FIRST HALF, CONTAINED IN ROM
1444
1445 059B BE 0000 E    MOV SI,OFFSET CRT_CHAR_GEN ; OFFSET OF IMAGES
1446 059E 0E      PUSH CS        ; SAVE SEGMENT ON STACK
1447 059F EB 18      JMP SHORT S2   ; DETERMINE_MODE
1448
1449          ;----- IMAGE IS IN SECOND HALF, IN USER MEMORY
1450
1451 05A1          S1:          ; EXTEND CHAR
1452 05A1 2C 80      SUB AL,80H     ; ZERO ORIGIN FOR SECOND HALF
1453 05A3 1E      PUSH DS       ; SAVE DATA POINTER
1454 05A4 2B F6      SUB SI,S1     ; ESTABLISH VECTOR ADDRESSING
1455 05A6 8E DE      MOV DS,S1
1456          ASSUME DS:ABS0
1457 05A8 C5 36 007C R  LDS SI,0EXT_PTR ; GET THE OFFSET OF THE TABLE
1458 05AC 8C DA      MOV DX,DS     ; GET THE SEGMENT OF THE TABLE
1459          ASSUME DS:DATA
1460 05AE 1F      POP DS       ; RECOVER DATA SEGMENT
1461 05AF 52      PUSH DX      ; SAVE TABLE SEGMENT ON STACK
1462 05B0 0B 04      OR DX,S1     ; CHECK FOR VALID TABLE DEFINED
1463 05B2 75 05      JNZ S2       ; CONTINUE IF DS:SI NOT 0000:0000
1464
1465 05B4 58      POP AX      ; ELSE SET (AX)= 0000 FOR "NULL"
1466 05B5 8E 0000 E  MOV SI,OFFSET CRT_CHAR_GEN ; POINT TO DEFAULT TABLE OFFSET
1467 05B8 0E      PUSH CS     ; IN THE CODE SEGMENT
1468
1469          ;----- DETERMINE GRAPHICS MODE IN OPERATION
1470
1471 05B9          S2:          ; DETERMINE MODE
1472 05B9 D1 E0      SAL AX,1     ; MULTIPLY CODE POINT VALUE BY 8
1473 05BB D1 E0      SAL AX,1
1474 05BD D1 E0      SAL AX,1
1475 05BF 03 F0      ADD SI,AX    ; SI HAS OFFSET OF DESIRED CODES
1476 05C1 80 3E 0049 R 06  CMP 0CRT_MODE,6
1477 05C6 1F      POP DS      ; RECOVER TABLE POINTER SEGMENT
1478 05C7 72 2C      JC S7       ; TEST FOR MEDIUM RESOLUTION MODE
1479
1480          ;----- HIGH RESOLUTION MODE
1481
1482 05C9          S3:          ; HIGH_CHAR
1483 05C9 57      PUSH DI     ; SAVE REGEN POINTER
1484 05CA 56      PUSH SI     ; SAVE CODE POINTER
1485 05CB B6 04      MOV DH,4   ; NUMBER OF TIMES THROUGH LOOP
1486 05CD
1487 05CD AC      LODSB     ; GET BYTE FROM CODE POINTS
1488 05CE F6 C3 80    TEST BL,80H ; SHOULD WE USE THE FUNCTION
1489 05D1 75 16      JNZ S6     ; TO PUT CHAR IN
1490 05D3 AA      STOSB     ; STORE IN REGEN BUFFER
1491 05D4 AC      LODSB
1492 05D5
1493 05D5 26 88 85 1FFF  MOV ES:[DI+2000H-1],AL ; STORE IN SECOND HALF
1494 05DA 83 C7 4F    ADD DI,79   ; MOVE TO NEXT ROW IN REGEN
1495 05DD FE CE      DEC DH     ; DONE WITH LOOP
1496 05DF 75 EC      JNZ S4
1497 05E1 5E      POP SI
1498 05E2 5F      POP DI     ; RECOVER REGEN POINTER
1499 05E3 47      INC DI     ; POINT TO NEXT CHAR POSITION
1500 05E4 E2 E3      LOOP S3   ; MORE CHARS TO WRITE
1501 05E6 E9 013D R  JMP VIDEO_RETURN
1502
1503          S6:
1504 05E9 26 32 06    XOR AL,ESI[DI] ; EXCLUSIVE OR WITH CURRENT
1505 05EC AA      STOSB     ; STORE THE CODE POINT
1506 05ED AC      LODSB     ; AGAIN FOR ODD FIELD
1507 05EE 26 32 85 1FFF  XOR AL,ESI[DI+2000H-1]
1508 05F3 EB E0      JMP S6     ; BACK TO MAINSTREAM
1509
1510          ;----- MEDIUM RESOLUTION WRITE
1511
1512 05F5          S7:          ; MED_RES_WRITE
1513 05F5 8A D3      MOV DL,BL   ; SAVE HIGH COLOR BIT
1514 05F7 D1 E7      SAL DI,1   ; OFFSET*2 SINCE 2 BYTES/CHAR
1515          ; EXPAND BL TO FULL WORD OF COLOR
1516 05F9 80 E3 03    AND BL,3   ; ISOLATE THE COLOR BITS ( LOW 2 BITS )
1517 05FC B0 55      MOV AL,055H ; GET BIT CONVERSION MULTIPLIER
1518 05FE F6 E3      MUL BL     ; EXPAND 2 COLOR BITS TO 4 REPLICATIONS
1519 0600 8A D8      MOV BL,AL  ; PLACE BACK IN WORK REGISTER
1520 0602 8A F8      MOV BH,AL  ; EXPAND TO 8 REPLICATIONS OF COLOR BITS
1521 0604          S8:          ; MED_CHAR
1522 0604 57      PUSH DI   ; SAVE REGEN POINTER
1523 0605 56      PUSH SI   ; SAVE THE CODE POINTER
1524 0606 B6 04      MOV DH,4 ; NUMBER OF LOOPS
1525 0608
1526 0609 AC      LODSB     ; GET CODE POINT
1527 0609 E8 06C4 R  CALL S21  ; DOUBLE UP ALL THE BITS
1528 060C 23 C3      AND AX,BX ; CONVERT TO FOREGROUND COLOR ( 0 BACK )
1529 060E 86 E0      XCHG AH,AL ; SWAP HIGH/LOW BYTES FOR WORD MOVE
1530 0610 F6 C2 80    TEST DL,80H ; IS THIS XOR FUNCTION
1531 0613 74 03      JZ S10    ; NO, STORE IT IN AS IT IS
1532 0615 26 33 05    XOR AX,ESI[DI] ; DO FUNCTION WITH LOW/HIGH
1533 0618          S10:         ; STORE FIRST BYTE HIGH, SECOND LOW
1534 0618 26 89 05    MOV ES:[DI],AX ; GET CODE POINT
1535 061B AC      LODSB
1536 061C E8 06C4 R  CALL S21
1537 061F 23 C3      AND AX,BX ; CONVERT TO COLOR
1538 0621 86 E0      XCHG AH,AL ; SWAP HIGH/LOW BYTES FOR WORD MOVE
1539 0623 F6 C2 80    TEST DL,80H ; AGAIN, IS THIS XOR FUNCTION
1540 0625 74 05      JZ S11    ; NO, JUST STORE THE VALUES
1541 0628 26 33 85 2000  XOR AX,ESI[DI+2000H] ; FUNCTION WITH FIRST HALF LOW
    
```

SECTION 5

```

1641 062D 26 89 86 2000 S11: MOV ES:[DI+2000H],AX ; STORE SECOND PORTION HIGH
1642 062D 89 C7 50 ADD DI,80 ; POINT TO NEXT LOCATION
1643 0635 FE CE DEC S9
1644 0637 75 CF JNZ S9 ; KEEP GOING
1645 0639 5E POP SI ; RECOVER CODE POINTER
1646 063A 5F POP DI ; RECOVER REGEN POINTER
1647 063B 47 INC DI ; POINT TO NEXT CHAR POSITION
1648 063C 47 INC DI
1649 063D E2 C8 LOOP S8 ; MORE TO WRITE
1650 063F E9 013D R JMP VIDEO_RETURN
1651 0642 GRAPHICS_WRITE ENDP
1652 0642 GRAPHICS_READ -----
1653 0642 GRAPHICS_READ PROC NEAR
1654 0642 CALL S26 ; CONVERTED TO OFFSET IN REGEN
1655 0643 8B F0 MOV SI,AX ; SAVE IN SI
1656 0644 8B EC MOV BP,SP ; ALLOCATE SPACE FOR THE READ CODE POINT
1657 0645 8B EC MOV BP,SP ; POINTER TO SAVE AREA
1658 0646 DETERMINE GRAPHICS MODES
1659 0647 CMP CRT_MODE,6
1660 0648 PUSH ES
1661 0649 POP DS ; POINT TO REGEN SEGMENT
1662 064A JC S13 ; MEDIUM RESOLUTION
1663 064B HIGH RESOLUTION READ
1664 064C GET VALUES FROM REGEN BUFFER AND CONVERT TO CODE POINT
1665 064D MOV DH,4 ; NUMBER OF PASSES
1666 064E S12: MOV AL,[SI] ; GET FIRST BYTE
1667 064F MOV [BP],AL ; SAVE IN STORAGE AREA
1668 0650 INC BP ; NEXT LOCATION
1669 0651 MOV AL,[SI+2000H] ; GET LOWER REGION BYTE
1670 0652 MOV [BP],AL ; ADJUST AND STORE
1671 0653 INC BP
1672 0654 ADD SI,80 ; POINTER INTO REGEN
1673 0655 DEC DH ; LOOP CONTROL
1674 0656 JNZ S12 ; DO IT SOME MORE
1675 0657 JMP SHORT S16 ; GO MATCH THE SAVED CODE POINTS
1676 0658 MEDIUM RESOLUTION READ
1677 0659 S13: SAL SI,1 ; OFFSET*2 SINCE 2 BYTES/CHAR
1678 065A MOV DH,4 ; NUMBER OF PASSES
1679 065B S14: CALL S23 ; GET BYTES FROM REGEN INTO SINGLE SAVE
1680 065C ADD SI,2000H-2 ; GO TO LOWER REGION
1681 065D CALL S23 ; GET THIS PAIR INTO SAVE
1682 065E SUB SI,2000H-80+2 ; ADJUST POINTER BACK INTO UPPER
1683 065F DEC DH ; KEEP GOING UNTIL ALL 8 DONE
1684 0660 JNZ S14
1685 0661 SAVE AREA HAS CHARACTER IN IT, MATCH IT
1686 0662 S15: MOV DI,OFFSET CRT_CHAR_GEN ; FIND_CHAR
1687 0663 PUSH CS ; ESTABLISH ADDRESSING
1688 0664 POP ES ; CODE POINTS IN CS
1689 0665 SUB BP,8 ; ADJUST POINTER TO START OF SAVE AREA
1690 0666 MOV SI,BP ; CURRENT CODE POINT BEING MATCHED
1691 0667 MOV AL,0
1692 0668 S16: PUSH SS ; ESTABLISH ADDRESSING TO STACK
1693 0669 POP DS ; FOR THE STRING COMPARE
1694 066A MOV DX,128 ; NUMBER TO TEST AGAINST
1695 066B S17: PUSH SI ; SAVE SAVE AREA POINTER
1696 066C PUSH DI ; SAVE CODE POINTER
1697 066D MOV CX,4 ; NUMBER OF WORDS TO MATCH
1698 066E REPE CMPSW ; COMPARE THE 8 BYTES AS WORDS
1699 066F POP DI ; RECOVER THE POINTERS
1700 0670 POP SI
1701 0671 JZ S18 ; IF ZERO FLAG SET, THEN MATCH OCCURRED
1702 0672 INC AL ; NO MATCH, MOVE ON TO NEXT
1703 0673 ADD DI,8 ; NEXT CODE POINT
1704 0674 DEC DX ; LOOP CONTROL
1705 0675 JNZ S17 ; DO ALL OF THEM
1706 0676 CHAR NOT MATCHED, MIGHT BE IN USER SUPPLIED SECOND HALF
1707 0677 CMP AL,0 ; AL<= 0 IF ONLY 1ST HALF SCANNED
1708 0678 JE S18 ; IF = 0, THEN ALL HAS BEEN SCANNED
1709 0679 SUB AX,AX ; ESTABLISH ADDRESSING TO VECTOR
1710 067A MOV DS,AX
1711 067B ASSUME DS:ABS0
1712 067C LES DI,NEXT_PTR ; GET POINTER
1713 067D MOV AX,ES ; SEE IF THE POINTER REALLY EXISTS
1714 067E OR AX,DI ; IF ALL 0, THEN DOESN'T EXIST
1715 067F JZ S18 ; NO SENSE LOOKING
1716 0680 MOV AL,128 ; ORIGIN FOR SECOND HALF
1717 0681 JMP S16 ; GO BACK AND TRY FOR IT
1718 0682 ASSUME DS:DATA
1719 0683 CHARACTER IS FOUND ( AL=0 IF NOT FOUND )
1720 0684 S18: ADD SP,8 ; READJUST THE STACK, THROW AWAY SAVE
1721 0685 JMP VIDEO_RETURN ; ALL DONE
1722 0686 GRAPHICS_READ ENDP
1723 0686 EXPAND BYTE
1724 0687 THIS ROUTINE TAKES THE BYTE IN AL AND DOUBLES ALL
1725 0688 OF THE BITS, TURNING THE 8 BITS INTO 16 BITS.
1726 0689 THE RESULT IS LEFT IN AX
1727 068A S21: PROC NEAR
1728 068B PUSH CX ; SAVE REGISTER
1729 068C MOV CX,8 ; SHIFT COUNT REGISTER FOR ONE BYTE
1730 068D S22: ROR AL,1 ; SHIFT BITS, LOW BIT INTO CARRY FLAG
1731 068E RCR BP,1 ; MOVE CARRY FLAG (LOW BIT) INTO RESULTS
1732 068F SAR BP,1 ; SIGN EXTEND HIGH BIT (DOUBLE IT)
1733 0690 LOOP S22 ; REPEAT FOR ALL 8 BITS
1734 0691 XCHG AX,BP ; MOVE RESULTS TO PARAMETER REGISTER
1735 0692 POP CX ; RECOVER REGISTER
1736 0693 RET ; ALL DONE
1737 0694 S21: ENDP

```

```

1655 ; MED READ BYTE
1656 ; THIS ROUTINE WILL TAKE 2 BYTES FROM THE REGEN BUFFER,
1657 ; COMPARE AGAINST THE CURRENT FOREGROUND COLOR, AND PLACE
1658 ; THE CORRESPONDING ON/OFF BIT PATTERN INTO THE CURRENT
1659 ; POSITION IN THE SAVE AREA
1660 ; ENTRY ---
1661 ; SI,DS = POINTER TO REGEN AREA OF INTEREST
1662 ; BX = EXPANDED FOREGROUND COLOR
1663 ; BP = POINTER TO SAVE AREA
1664 ; EXIT --
1665 ; SI AND BP ARE INCREMENTED
1666 -----
1667
1668 06D3 PROC NEAR
1669 06D3 AD ; GET FIRST BYTE AND SECOND BYTES
1670 06D4 86 C4 ; SWAP FOR COMPARE
1671 06D5 89 C0D0 ; 2 BIT MASK TO TEST THE ENTRIES
1672 06D9 82 00 ; RESULT REGISTER
1673 06DB
1674 06DB 85 C1 ; IS THIS SECTION BACKGROUND?
1675 06DD 74 01 ; IF ZERO, IT IS BACKGROUND (CARRY=0)
1676 06DF 79 ; WASN'T, SO SET CARRY
1677 06E0
1678 06E0 DD D2 ; MOVE THAT BIT INTO THE RESULT
1679 06E2 D1 E9 ; SHR CX,1
1680 06E4 D1 E9 ; SHR CX,1
1681 06E6 73 F3 ; JNC S24
1682 06E8 88 56 00 ; MOV [BP],DL
1683 06EB 45 ; INC BP
1684 06EC C3 ; RET
1685 06ED
1686
1687 ; V4 POSITION
1688 ; THIS ROUTINE TAKES THE CURSOR POSITION CONTAINED IN
1689 ; THE MEMORY LOCATION, AND CONVERTS IT INTO AN OFFSET
1690 ; INTO THE REGEN BUFFER, ASSUMING ONE BYTE/CHAR.
1691 ; FOR MEDIUM RESOLUTION GRAPHICS, THE NUMBER MUST
1692 ; BE DOUBLED.
1693 ; ENTRY -- NO REGISTERS, MEMORY LOCATION @CURSOR_POSN IS USED
1694 ; EXIT --
1695 ; AX CONTAINS OFFSET INTO REGEN BUFFER
1696 -----
1697
1697 06ED PROC NEAR
1698 06ED A1 0050 R ; MOV AX,CURSOR_POSN
1699 06F0 LABEL NEAR
1700 06F0 53 ; PUSH BX
1701 06F1 8B D8 ; MOV BX,AX
1702 06F3 A0 004A R ; MOV AL,BYTE PTR @CRT_COLS
1703 06F6 F6 E4 ; MUL AH
1704 06F8 D1 E0 ; SHL AX,1
1705 06FA D1 E0 ; SHL AX,1
1706 06FC 2A FF ; SUB BH,BH
1707 06FE 03 C3 ; ADD AX,BX
1708 0700 5B ; POP BX
1709 0701 C3 ; RET
1710 0702
1711
1711 WRITE_TTY
1712 ;
1713 ; THIS INTERFACE PROVIDES A TELETYPE LIKE INTERFACE TO THE
1714 ; VIDEO CARDS. THE INPUT CHARACTER IS WRITTEN TO THE CURRENT
1715 ; CURSOR POSITION, AND THE CURSOR IS MOVED TO THE NEXT POSITION.
1716 ; IF THE CURSOR LEAVES THE LAST COLUMN OF THE FIELD, THE COLUMN
1717 ; IS SET TO ZERO, AND THE ROW VALUE IS INCREMENTED. IF THE ROW
1718 ; ROW VALUE LEAVES THE FIELD, THE CURSOR IS PLACED ON THE LAST ROW,
1719 ; FIRST COLUMN, AND THE ENTIRE SCREEN IS SCROLLED UP ONE LINE.
1720 ; WHEN THE SCREEN IS SCROLLED UP, THE ATTRIBUTE FOR FILLING THE
1721 ; NEWLY BLANKED LINE IS READ FROM THE CURSOR POSITION ON THE PREVIOUS
1722 ; LINE BEFORE THE SCROLL, IN CHARACTER MODE. IN GRAPHICS MODE,
1723 ; THE 0 COLOR IS USED.
1724 ; ENTRY --
1725 ; (AH) = CURRENT CRT MODE
1726 ; (AL) = CHARACTER TO BE WRITTEN
1727 ; NOTE THAT BACK SPACE, CARRIAGE RETURN, BELL AND LINE FEED ARE
1728 ; HANDLED AS COMMANDS RATHER THAN AS DISPLAY GRAPHICS CHARACTERS
1729 ; (BL) = FOREGROUND COLOR FOR CHAR WRITE IF CURRENTLY IN A GRAPHICS MODE
1730 ; EXIT --
1731 ; ALL REGISTERS SAVED THROUGH VIDEO_EXIT (INCLUDING (AX))
1732 -----
1733
1733 ASSUME DS:DATA
1734 0702 PROC NEAR
1735 0702 97 ; XCHG DI,AX
1736 0703 B4 03 ; MOV AH,03H
1737 0705 8A 3E 0062 R ; MOV BH,#ACTIVE_PAGE
1738 0709 CD 10 ; INT 10H
1739 070B 8B C7 ; MOV AX,DI
1740
1741 ;----- DX NOW HAS THE CURRENT CURSOR POSITION
1742
1743 070D 3C 0D ; CMP AL,CR
1744 070F 76 46 ; JBE UB
1745
1746 ;----- WRITE THE CHAR TO THE SCREEN
1747 0711 B4 0A ; UD: MOV AH,0AH
1748 0713 B9 00D1 ; MOV CX,1
1749 0716 CD 10 ; INT 10H
1750
1751 ;----- POSITION THE CURSOR FOR NEXT CHAR
1752
1753 0718 FE C2 ; INC DL
1754 071A 3A 16 004A R ; CMP DL,BYTE PTR @CRT_COLS
1755 071E 75 33 ; JNZ UT
1756 0720 82 00 ; MOV DL,0
1757 0722 90 FE 18 ; CMP DH,25-1
1758 0725 75 2A ; JNZ U6
1759
1760 ;----- SCROLL REQUIRED
1761 0727 B4 02 ; UI: MOV AH,02H
1762 0729 CD 10 ; INT 10H
1763
1764 ;----- DETERMINE VALUE TO FILL WITH DURING SCROLL
1765
1766 072B A0 0049 R ; MOV AL,@CRT_MODE
1767 072E 3C 04 ; CMP AL,4
1768 0730 72 06 ; JC U2
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
    
```

SECTION 5

```

1769 0732 3C 07      CMP     AL,7
1770 0734 87 00      MOV     BH,0          ; FILL WITH BACKGROUND
1771 0736 75 06      JNE     U3            ; SCROLL-UP
1772 0738          U2:   MOV     AH,08H        ; READ-CURSOR
1773 073B 84 08      INT     10H          ; GET READ CURSOR COMMAND
1774 073A CD 10      MOV     BH,AH        ; READ CHAR/ATTR AT CURRENT CURSOR
1775 073C 8A FC      MOV     BH,AH        ; STORE IN BH
1776 073E          U3:   MOV     AX,0601H      ; SCROLL-UP
1777 073E B8 0601    SUB     CX,CX         ; SCROLL ONE LINE
1778 0741 2B C9      MOV     DH,25-1      ; UPPER LEFT CORNER
1779 0743 B6 18      MOV     DL,BYTE PTR @CRT_COLS ; LOWER RIGHT ROW
1780 0745 8A 16 004A R MOV     DL,BYTE PTR @CRT_COLS ; LOWER RIGHT COLUMN
1781 0749 FE CA      DEC     DL           ;
1782 074B          U4:   INT     10H          ; VIDEO-CALL-RETURN
1783 074B CD 10      XCHG   AX,DI         ; SCROLL UP THE SCREEN
1784 074D          U5:   JMP     VIDEO_RETURN ; TTY-RETURN
1785 074D 97          ; RESTORE THE ENTRY CHARACTER FROM (DI)
1786 074E E9 013D R  JMP     VIDEO_RETURN ; RETURN TO CALLER
1787          U6:   INC     DH           ; SET-CURSOR-INC
1788 0751          U7:   MOV     AH,02H      ; NEXT ROW
1789 0751 FE C6      JMP     U4            ; SET-CURSOR
1790 0753          ; ESTABLISH THE NEW CURSOR
1791 0753 B4 02      ;
1792 0755 EB F4      ;
1793          ;
1794          ;----- CHECK FOR CONTROL CHARACTERS
1795 0757          U8:   JE      U9           ; WAS IT A CARRIAGE RETURN
1796 0757 74 13      CMP     AL,LF        ; IS IT A LINE FEED
1797 0759 3C 0A      JE      U10          ; GO TO LINE FEED
1798 075B 74 13      CMP     AL,U10       ; IS IT A BELL
1799 075D 3C 07      JE      U11          ; GO TO BELL
1800 075F 74 16      CMP     AL,U11       ; IS IT A BACKSPACE
1801 0761 3C 08      JNE     U0           ; IF NOT A CONTROL, DISPLAY IT
1802 0763 75 AC      ;
1803          ;----- BACK SPACE FOUND
1804          ;
1805          ;
1806 0765 0A D2      OR      DL,DL        ; IS IT ALREADY AT START OF LINE
1807 0767 74 EA      JE      U7           ; SET_CURSOR
1808 0769 4A        DEC     DX           ; NO -- JUST MOVE IT BACK
1809 076A EB E7      JMP     U7           ; SET_CURSOR
1810          ;
1811          ;----- CARRIAGE RETURN FOUND
1812 076C B2 00      U9:   MOV     DL,0        ; MOVE TO FIRST COLUMN
1813 076E EB E3      JMP     U7           ; SET_CURSOR
1814          ;
1815          ;----- LINE FEED FOUND
1816 0770 80 FE 18  U10:  CMP     DH,25-1     ; BOTTOM OF SCREEN
1817 0773 75 DC      JNE     U6           ; YES, SCROLL THE SCREEN
1818 0775 EB D0      JMP     U1           ; NO, JUST SET THE CURSOR
1819          ;
1820          ;----- BELL FOUND
1821 0777 B9 0593    U11:  MOV     CX,1331     ; DIVISOR FOR 8% HZ TONE
1822 077A B2 1F      MOV     DL,31       ; SET COUNT FOR 31/64 SECOND FOR BEEP
1823 077C E8 0000 E CALL    BEEP         ; SOUND THE POD BELL
1824 077F EB CC      JMP     U5           ; TTY_RETURN
1825 0781          ;
1826          ;----- WRITE_TTY
1827          ;
1828          ; THIS ROUTINE TESTS THE LIGHT PEN SWITCH AND THE LIGHT
1829          ; PEN TRIGGER. IF BOTH ARE SET, THE LOCATION OF THE LIGHT
1830          ; PEN IS DETERMINED. OTHERWISE, A RETURN WITH NO INFORMATION
1831          ; IS MADE.
1832          ;
1833          ; ON EXIT:
1834          ; (AH) = 0 IF NO LIGHT PEN INFORMATION IS AVAILABLE
1835          ; (BH,CX,DX) ARE DESTROYED
1836          ; (AH) = 1 IF LIGHT PEN IS AVAILABLE
1837          ; (DH,DL) = ROW,COLUMN OF CURRENT LIGHT PEN POSITION
1838          ; (CH) = RASTER POSITION
1839          ; (BX) = BEST GUESS AT PIXEL HORIZONTAL POSITION
1840          ;-----
1841 0781 03 03 05 03 03 V1  DB     3,3,6,6,3,3,3,4 ; SUBTRACT_TABLE
1842 0782 03 04          ;
1843          ;----- WAIT FOR LIGHT PEN TO BE DEPRESSED
1844          ;
1845 0789          READ_LPEN PROC NEAR
1846 0789 B4 00      MOV     AH,0        ; SET NO LIGHT PEN RETURN CODE
1847 078B 8B 16 0063 R MOV     DX,@ADDR_6845 ; GET BASE ADDRESS OF 6845
1848 078F 83 C2 06      ADD     DX,6        ; POINT TO STATUS REGISTER
1849 0792 EC          IN      AL,DX       ; GET STATUS REGISTER
1850 0793 A8 04      TEST    AL,004H     ; TEST LIGHT PEN SWITCH
1851 0795 74 03      JZ      V6_A        ; GO IF YES
1852 0797 E9 081C R  JMP     V6          ; NOT SET, RETURN
1853          ;
1854          ;----- NOW TEST FOR LIGHT PEN TRIGGER
1855          ;
1856 079A A8 02      V6_A:  TEST    AL,2        ; TEST LIGHT PEN TRIGGER
1857 079C 75 03      JNZ     V7A         ; RETURN WITHOUT RESETTING TRIGGER
1858 079E E9 0826 R  JMP     V7          ;
1859          ;
1860          ;----- TRIGGER HAS BEEN SET, READ THE VALUE IN
1861          ;
1862 07A1 B4 10      V7A:  MOV     AH,16       ; LIGHT PEN REGISTERS ON 6845
1863          ;
1864          ;----- INPUT REGISTERS POINTED TO BY AH, AND CONVERT TO ROW COLUMN IN (DX)
1865          ;
1866 07A3 8B 16 0063 R MOV     DX,@ADDR_6845 ; ADDRESS REGISTER FOR 6845
1867 07A7 8A C4      MOV     AL,AH        ; REGISTER TO READ
1868 07A9 EE          OUT     DX,AL        ; SET IT UP
1869 07AA EB 00      JMP     $+2          ; I/O DELAY
1870 07AC 42          INC     DX           ; DATA REGISTER
1871 07AD EC          IN      AL,DX       ; GET THE VALUE
1872 07AE 8A E8      MOV     CH,AL        ; SAVE IN CX
1873 07B0 4A        DEC     DX           ; ADDRESS REGISTER
1874 07B1 FE CA      INC     AH           ;
1875 07B3 8A C4      MOV     AL,AH        ; SECOND DATA REGISTER
1876 07B5 EE          OUT     DX,AL        ; POINT TO DATA REGISTER
1877 07B7 42          INC     DX           ; I/O DELAY
1878 07B7 EB 00      JMP     $+2          ;
1879 07B9 EC          IN      AL,DX       ; GET SECOND DATA VALUE
1880 07BA 8A E5      MOV     AH,CH        ; AX HAS INPUT VALUE
    
```

```

1881                                     PAGE
1882                                     ;----- AX HAS THE VALUE READ IN FROM THE 6845
1883
1884 07BC 8A 1E 0049 R                   MOV    BL,#CRT_MODE
1885 07C0 2A FF                          SUB    BH,BH
1886 07C2 2E 1A 9F 0781 R               MOV    BL,CS:VI[BX]
1887 07C7 2B C3                          SUB    AX,BX
1888 07C9 BB 1E 004E R                   MOV    BX,#CRT_START
1889 07CD D1 E8                          SHR    BX,1
1890 07CF 2B C3                          SUB    AX,BX
1891 07D1 79 02                          JNS   V2
1892 07D3 2B C0                          SUB    AX,AX
1893
1894                                     ;----- DETERMINE MODE OF OPERATION
1895
1896 07D5                                     V2:
1897 07D5 B1 03                          MOV    CL,3
1898 07D7 80 3E 0049 R 04                CMP    #CRT_MODE,4
1899 07DC 72 2A                          JB     V4
1900 07DE 80 3E 0049 R 07                CMP    #CRT_MODE,7
1901 07E3 74 23                          JE     V4
1902
1903                                     ;----- GRAPHICS MODE
1904
1905 07E5 B2 28                          MOV    DL,40
1906 07E7 F6 F2                          DIV    DL
1907
1908                                     ;----- DETERMINE GRAPHIC ROW POSITION
1909
1910 07E9 8A E8                          MOV    CH,AL
1911 07EB 02 ED                          ADD    CH,CH
1912 07ED 8A DC                          MOV    BH,AH
1913 07EF 2A FF                          SUB    BH,BH
1914 07F1 80 3E 0049 R 06                CMP    #CRT_MODE,6
1915 07F6 75 04                          JNE   V3
1916 07F8 B1 04                          MOV    CL,4
1917 07FA D0 E4                          SAL    AH,1
1918 07FC                                     V3:
1919 07FC D3 E3                          SHL   BX,CL
1920
1921                                     ;----- DETERMINE ALPHA CHAR POSITION
1922
1923 07FE 8A D4                          MOV    DL,AH
1924 0800 8A F0                          MOV    DH,AL
1925 0802 D0 EE                          SHR    DH,1
1926 0804 D0 EE                          SHR    DH,1
1927 0806 EB 12                          JMP   SHORT V5
1928
1929                                     ;----- ALPHA MODE ON LIGHT PEN
1930
1931 0808                                     V4:
1932 0808 F6 36 004A R                   DIV    BYTE PTR #CRT_COLS
1933 080C 8A F0                          MOV    DH,AL
1934 080E 8A D4                          MOV    DL,AH
1935 0810 D2 E0                          SAL    AL,CL
1936 0812 8A E8                          MOV    CH,AL
1937 0814 8A DC                          MOV    BH,AH
1938 0816 32 FF                          XOR    BH,BH
1939 0818 D3 E3                          SAL    BX,CL
1940
1941 081A                                     V5:
1942 081A B4 01                          MOV    AH,1
1943 081C                                     V6:
1944 081C 52                             PUSH  DX
1945 081D BB 16 0063 R                   MOV    DX,#ADDR_6845
1946 0821 83 C2 07                       ADD    DX,7
1947 0824 EE                             OUT   DX,AL
1948 0825 5A                             POP   DX
1949 0826 5D                             POP   BP
1950 0827 5F                             POP   DI
1951 0828 5E                             POP   SI
1952 0829 5D                             POP   DS
1953 082A 5F                             POP   DS
1954 082B 5F                             POP   DS
1955 082C 5F                             POP   DS
1956 082D 07                             POP   ES
1957 082E CF                             IRET
1958 082F                                     READ_LPEN ENDP
1959 082F                                     CODE     ENDS
1960
1961                                     END
    
```

SECTION 5

```

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  NM1_INT_1
10
11         EXTRN  C8042:NEAR      ; POST SEND 8042 COMMAND ROUTINE
12         EXTRN  CMOS_READ:NEAR ; READ CMOS LOCATION ROUTINE
13         EXTRN  D1:NEAR        ; "PARITY CHECK 1" MESSAGE
14         EXTRN  D2:NEAR        ; "PARITY CHECK 2" MESSAGE
15         EXTRN  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_CHAR: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_DETERMINE
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 1E      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
56
57         MEMORY_SIZE_DET_1 ENDP
58         ----- INT 11 H -----
59         EQUIPMENT_DETERMINATION
60         THIS ROUTINE ATTEMPTS TO DETERMINE WHAT OPTIONAL
61         DEVICES ARE ATTACHED TO THE SYSTEM.
62         INPUT
63         NO REGISTERS
64         THE #EQUIP_FLAG VARIABLE IS SET DURING THE POWER ON
65         DIAGNOSTICS USING THE FOLLOWING HARDWARE ASSUMPTIONS:
66         PORT 03FA = INTERRUPT ID REGISTER OF 8250 (PRIMARY)
67         02FA = INTERRUPT ID REGISTER OF 8250 (SECONDARY)
68         BITS 7-3 ARE ALWAYS 0
69         PORT 0378 = OUTPUT PORT OF PRINTER (PRIMARY)
70         0278 = OUTPUT PORT OF PRINTER (SECONDARY)
71         03BC = OUTPUT PORT OF PRINTER (MONOCHROME-PRINTER)
72         OUTPUT
73         (AX) IS SET, BIT SIGNIFICANT, TO INDICATE ATTACHED I/O
74         BIT 15,14 = NUMBER OF PRINTERS ATTACHED
75         BIT 13 = INTERNAL MODEM INSTALLED
76         BIT 12 NOT USED
77         BIT 11,10,9 = NUMBER OF RS232 CARDS ATTACHED
78         BIT 8 = NOT USED
79         BIT 7,6 = NUMBER OF DISKETTE DRIVES
80         00=1, 01=2 ONLY IF BIT 0 = 1
81         BIT 5,4 = INITIAL VIDEO MODE
82         00 - UNUSED
83         01 - 40X25 BW USING COLOR CARD
84         10 - 80X25 BW USING COLOR CARD
85         11 - 80X25 BW USING BW CARD
86
87         BIT 3 = NOT USED
88         BIT 2 = NOT USED
89         BIT 1 = MATH COPROCESSOR
90         BIT 0 = 1 (IPL DISKETTE INSTALLED)
91         NO OTHER REGISTERS AFFECTED
92         -----
93         000A      EQUIPMENT_1 PROC FAR
94         000A FB      STI          ; INTERRUPTS BACK ON
95         000B 1E      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
100        EQUIPMENT_1 ENDP

```

```

101 PAGE
102 |--- HARDWARE INT 02 H -- ( NMI LEVEL ) -----
103 | NON-MASKABLE INTERRUPT ROUTINE (REAL MODE)
104 | THIS ROUTINE WILL PRINT A *PARITY CHECK 1 OR 2* MESSAGE AND ATTEMPT
105 | TO FIND THE STORAGE LOCATION IN BASE 640K CONTAINING THE BAD PARITY.
106 | IF FOUND, THE SEGMENT ADDRESS WILL BE PRINTED. IF NO PARITY ERROR
107 | CAN BE FOUND (INTERMITTENT READ PROBLEM) ????? WILL BE DISPLAYED
108 | WHERE THE ADDRESS WOULD NORMALLY GO.
109 |
110 | PARITY CHECK 1 = PLANAR BOARD MEMORY FAILURE.
111 | PARITY CHECK 2 = OFF PLANAR BOARD MEMORY FAILURE.
112 |-----
113
114 0014 NMI_INT_1 PROC NEAR
115 0014 50 PUSH AX ; SAVE ORIGINAL CONTENTS OF (AX)
116
117 0015 E4 61 IN AL,PORT_B ; READ STATUS PORT
118 0017 A8 C0 TEST AL,PARITY_ERR ; PARITY CHECK OR I/O CHECK ?
119 0019 75 07 JNZ NMI_1 ; GO TO ERROR HALTS IF HARDWARE ERROR
120
121 001B B0 0F MOV AL,CMOS_SHUT_DOWN ; ELSE ?? - LEAVE NMI ON
122 001D E8 0000 E CALL CMOS_READ ; TOGGLE NMI USING COMMON READ ROUTINE
123 0020 58 POP AX ; RESTORE ORIGINAL CONTENTS OF (AX)
124 0021 CF IRET ; EXIT NMI HANDLER BACK TO PROGRAM
125
126
127 0022 NMI_1 ; HARDWARE ERROR
128 0022 50 PUSH AX ; SAVE INITIAL CHECK MASK IN (AL)
129 0023 B0 8F MOV AL,CMOS_SHUT_DOWN+NMI ; MASK TRAP (NMI) INTERRUPTS OFF
130 0025 E8 0000 E CALL CMOS_READ ; OPEN STANDBY LATCH BEFORE POWER DOWN
131 0028 B0 AD MOV AL,DS_KBD ; DISABLE THE KEYBOARD
132 002A E8 0000 E CALL CB042 ; SEND COMMAND TO ADAPTER
133 002D E8 0000 E CALL DDS ; ADDRESS DATA SEGMENT
134 0030 B4 00 MOV AH,0 ; INITIALIZE AND SET MODE FOR VIDEO
135 0032 AD 0049 R MOV AL,PORT_MODE ; GET CURRENT MODE
136 0035 CD 10 INT 10H ; CALL VIDEO_IO TO CLEAR SCREEN
137
138 |----- DISPLAY *PARITY CHECK ?* ERROR MESSAGES
139
140 0037 58 POP AX ; RECOVER INITIAL CHECK STATUS
141 0038 BE 0000 E MOV SI,OFFSET D1 ; PLANAR ERROR, ADDRESS *PARITY CHECK 1*
142 003B A5 10 TEST AL,PARITY_CHECK ; CHECK FOR PLANAR ERROR
143 003D 74 05 JZ NMI_2 ; SKIP IF NOT
144
145 003F 50 PUSH AX ; SAVE STATUS
146 0040 E8 0000 E CALL P_MSG ; DISPLAY *PARITY CHECK 1* MESSAGE
147 0043 58 POP AX ; AND RECOVER STATUS
148 0044
149 0044 BE 0000 E NMI_2: MOV SI,OFFSET D2 ; ADDRESS OF *PARITY CHECK 2* MESSAGE
150 0047 A5 10 TEST AL,I/O_CHECK ; I/O PARITY CHECK ?
151 0049 74 03 JZ NMI_3 ; SKIP IF CORRECT ERROR DISPLAYED
152 004B E8 0000 E CALL P_MSG ; DISPLAY *PARITY CHECK 2* ERROR
153
154 |----- TEST FOR HOT NMI ON PLANAR PARITY LINE
155
156 004E NMI_3:
157 004E E4 61 IN AL,PORT_B ; TOGGLE PARITY CHECK ENABLES
158 0050 0C 0C OR AL,RAM_PAR_OFF ; TO CLEAR THE PENDING CHECK
159 0052 E6 61 OUT PORT_B,AL
160 0054 24 F3 AND AL,RAM_PAR_ON
161 0056 E6 61 OUT PORT_B,AL
162
163 0058 FC CLD ; SET DIRECTION FLAG TO INCREMENT
164 0059 2B D2 SUB DX,DX ; POINT (DX) AT START OF REAL MEMORY
165 005B 2B F6 SUB SI,SI ; SET (SI) TO START OF (DS)
166 005D E4 61 IN AL,PORT_B ; READ CURRENT PARITY CHECK LATCH
167 005F A8 C0 TEST AL,PARITY_ERR ; CHECK FOR HOT NMI SOURCE
168 0061 75 19 JNZ NMI_5 ; SKIP IF ERROR NOT RESET (DISPLAY ???)
169
170 |----- SEE IF LOCATION THAT CAUSED PARITY CHECK CAN BE FOUND IN BASE MEMORY
171
172 0063 8B 1E 0013 R MOV BX,MEMORY_SIZE ; GET BASE MEMORY SIZE WORD
173 0067
174 0067 BE DA NMI_4: MOV DS,DX ; POINT TO 64K SEGMENT
175 0069 B9 B000 MOV CX,4000H*2 ; SET WORD COUNT FOR 64 KB SCAN
176 006C F3/ AD REP LODSW ; READ 64 KB OF MEMORY
177 006E E4 61 IN AL,PORT_B ; READ PARITY CHECK LATCHES
178 0070 A8 C0 TEST AL,PARITY_ERR ; CHECK FOR ANY PARITY ERROR PENDING
179 0072 75 10 JNZ NMI_6 ; GO PRINT SEGMENT ADDRESS IF ERROR
180
181 0074 80 C6 10 ADD DH,010H ; POINT TO NEXT 64K BLOCK
182 0077 83 EB 40 SUB BX,160H*4 ; DECREMENT COUNT OF 1024 BYTE SEGMENTS
183 007A 77 EB JA NMI_4 ; LOOP TILL ALL 64K SEGMENTS DONE
184 007C
185 007C BE 0000 E NMI_5: MOV SI,OFFSET D2A ; PRINT ROW OF ????? IF PARITY
186 007F E8 0000 E CALL P_MSG ; CHECK COULD NOT BE RE-CREATED
187 0082 FA CLI ; HALT SYSTEM
188 0083 F4 HLT
189
190 0084 NMI_6:
191 0084 E8 0000 E CALL PRT_SEG ; PRINT SEGMENT VALUE (IN DX)
192 0087 B0 2B MOV AL,'(' ; PRINT (S)
193 0089 E8 0000 E CALL PRT_HEX ; PRINT (S)
194 008C B0 53 MOV AL,'S'
195 008E E8 0000 E CALL PRT_HEX ; PRINT (S)
196 0091 B0 29 MOV AL,')'
197 0093 E8 0000 E CALL PRT_HEX ; PRINT (S)
198 0096 FA CLI ; HALT SYSTEM
199 0097 F4 HLT
200
201 0098 NMI_INT_1 ENDP
202
203 0098 CODE ENDS
204 END

```

SECTION 5

```

1 PAGE 118,121
2 TITLE BIOS1 ---- 06/10/85 INTERRUPT 15H BIOS ROUTINES
3 .286C
4 .LIST
5 0000
6 CODE SEGMENT BYTE PUBLIC
7 PUBLIC CASSETTE_IO_1
8 PUBLIC GATE_A20
9 PUBLIC SHUT9
10
11 EXTRN CMOS_READ:NEAR ; READ CMOS LOCATION ROUTINE
12 EXTRN CMOS_WRITE:NEAR ; WRITE CMOS LOCATION ROUTINE
13 EXTRN CONF_TBL:NEAR ; SYSTEM/BIOS CONFIGURATION TABLE
14 EXTRN DOS:NEAR ; LOAD (DS) WITH DATA SEGMENT SELECTOR
15 EXTRN PROC_SHUTDOWN:NEAR ; 80286 HARDWARE RESET ROUTINE
16
17
18 INT 15 H
19 INPUT - CASSETTE I/O FUNCTIONS
20
21 (AH) = 00H
22 (AH) = 01H
23 (AH) = 02H
24 (AH) = 03H
25 RETURNS FOR THESE FUNCTIONS ALWAYS (AH) = 86H, CY = 1)
26 IF CASSETTE PORT NOT PRESENT
27
28 INPUT - UNUSED FUNCTIONS
29 (AH) = 04H THROUGH 7FH
30 RETURNS FOR THESE FUNCTIONS ALWAYS (AH) = 86H, CY = 1)
31 (UNLESS INTERCEPTED BY SYSTEM HANDLERS)
32 NOTE: THE KEYBOARD INTERRUPT HANDLER INTERRUPTS WITH AH=4FH
33
34 EXTENSIONS
35 (AH) = 80H DEVICE OPEN
36 (BX) = DEVICE ID
37 (CX) = PROCESS ID
38
39 (AH) = 81H DEVICE CLOSE
40 (BX) = DEVICE ID
41 (CX) = PROCESS ID
42
43 (AH) = 82H PROGRAM TERMINATION
44 (BX) = DEVICE ID
45
46 (AH) = 83H EVENT WAIT
47
48 (AL) = 00H SET INTERVAL
49 (ES:BX) POINTER TO A BYTE IN CALLERS MEMORY
50 THAT WILL HAVE THE HIGH ORDER BIT SET
51 AS SOON AS POSSIBLE AFTER THE INTERVAL
52 EXPIRES.
53 (CX,DX) NUMBER OF MICROSECONDS TO ELAPSE BEFORE
54 POSTING.
55 (AL) = 01H CANCEL
56
57 RETURNS: CARRY IF AL NOT = 00H OR 01H
58 OR IF FUNCTION AL=0 ALREADY BUSY
59
60 (AH) = 84H JOYSTICK SUPPORT
61 (DX) = 00H - READ THE CURRENT SWITCH SETTINGS
62 RETURNS AL = SWITCH SETTINGS (BITS 7-4)
63 (DX) = 01H - READ THE RESISTIVE INPUTS
64 RETURNS AX = A(x) VALUE
65 BX = A(y) VALUE
66 CX = B(x) VALUE
67 DX = B(y) VALUE
68
69 (AH) = 85H SYSTEM REQUEST KEY PRESSED
70 (AL) = 00H MAKE OF KEY
71 (AL) = 01H BREAK OF KEY
72
73 (AH) = 86H WAIT
74 (CX,DX) NUMBER OF MICROSECONDS TO ELAPSE BEFORE
75 RETURN TO CALLER
76
77 (AH) = 87H MOVE BLOCK
78 (CX) NUMBER OF WORDS TO MOVE
79 (ES:SI) POINTER TO DESCRIPTOR TABLE
80
81 (AH) = 88H EXTENDED MEMORY SIZE DETERMINE
82
83 (AH) = 89H PROCESSOR TO VIRTUAL MODE
84
85 (AH) = 90H DEVICE BUSY LOOP
86 (AL) SEE TYPE CODE
87
88 (AH) = 91H INTERRUPT COMPLETE FLAG SET
89 (AL) TYPE CODE
90 00H -> 7FH
91 SERIALLY REUSABLE DEVICES
92 OPERATING SYSTEM MUST SERIALIZE ACCESS
93 80H -> BFH
94 REENRANT DEVICES; ES:BX IS USED TO
95 DISTINGUISH DIFFERENT CALLS (MULTIPLE I/O
96 CALLS ARE ALLOWED SIMULTANEOUSLY)
97 C0H -> FFH
98 WAIT ONLY CALLS -- THERE IS NO
99 COMPLEMENTARY 'POST' FOR THESE WAITS.
100 THESE ARE TIMEOUT ONLY. TIMES ARE
101 FUNCTION NUMBER DEPENDENT.
102
103 TYPE DESCRIPTION TIMEOUT
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 DW 8 LENGTH OF FOLLOWING TABLE
120 DB MODEL_BYTE SYSTEM MODEL BYTE
121 DB TYPE_BYTE SYSTEM MODEL TYPE BYTE
122 DB BIOS_LEVEL BIOS REVISION LEVEL
123 DB ? 10000000 = DMA CHANNEL 3 USE BY BIOS
124 01000000 = CASCADED INTERRUPT LEVEL 2
125 00100000 = REAL TIME CLOCK AVAILABLE
126 00010000 = KEYBOARD SCAN CODE HOOK (AH)
127 DB 0 RESERVED
128 DB 0 RESERVED
129 DB 0 RESERVED
130 DB 0 RESERVED
131
-----
132 ASSUME CS:CODE
133
134 CASSETTE_IO_1 PROC FAR
135 STI_1
136 0000
137 0000 FB ; ENABLE INTERRUPTS
138 0001 80 FC 80 ; CHECK FOR RANGE
139 0004 72 4C ; RETURN IF 00-7FH
140 0006 80 FC C0 ; CHECK FOR CONFIGURATION PARAMETERS
141 0009 74 4F ;
142 000B 80 EC 80 ; BASE ON 0
143 000E 74 48 ; DEVICE OPEN
144 0010 FE CC ;
145 0012 74 44 ; DEVICE CLOSE
146 0014 FE CC ;
147 0016 74 40 ; PROGRAM TERMINATION
148 0018 FE CC ;
149 001A 74 47 ; EVENT WAIT
150 001C FE CC ;
151 001E 75 03 ;
152 0020 E9 01D6 R ; JOYSTICK BIOS
153 0023
154 0023 FE CC NOT_JOYSTICK:
155 0025 74 31 ;
156 0027 FE CC ;
157 0029 74 07 ;
158 002B FE CC ;
159 002D 75 06 ;
160 002F E9 01D6 R ; MOVE BLOCK
161
162 0032 E9 0175 R ; WAIT
163
164 0035 FE CC ;
165
166 0037 75 03 ;
167 0039 E9 03FC R ; GO GET THE EXTENDED MEMORY
168
169 003C FE CC ;
170 003E 75 03 ;
171 0040 E9 0408 R ;
172
173 0043 80 EC 07 ;
174 0046 75 03 ;
175 0048 E9 0491 R ;
176
177 004B FE CC ;
178 004D 75 03 ;
179 004F E9 0495 R ;
180
181 0052 B4 86 ;
182 0054 F9 ;
183 0055 ;
184 0058 CA 0002 ;
185
186 ;
187 0058 ;
188 ;
189 0058 ;
190 ;
191 0058 ;
192 ;
193 0058 ;
194 0058 EB FB ;
195 005A ;
196 ;
197 005A ;
198 005A 0E ;
199 005B 07 ;
200 005C B8 0000 E ;
201 005F 3E E4 ;
202 0061 EB F2 ;
203 0063 ;
204 ;
205 0063 ;
206 ;
207 0063 1E ;
208 0064 EA 0000 E ;
209 0067 0A C0 ;
210 0069 74 08 ;
211 006B FE C8 ;
212 006D 74 45 ;
213 006F 1F ;
214 0070 F9 ;
215 0071 EB E2 ;
216 ;
217 0073 ;
218 0073 FA ;
219 0074 F6 06 00A0 R 01 ;
220 0079 74 05 ;
221 007B FB ;
222 007C 1F ;
223 007D F9 ;
224 007E EB D5 ;
225

```

SECTION 5

```

226 0080          EVENT_WAIT_1:
227 0080 E4 A1      IN      AL,INTB01          ; ENSURE INTERRUPT UNMASKED
228 0082 EB 00      JMP     $+2
229 0084 24 FE      AND     AL,0FEH
230 0086 E6 A1      OUT     INTB01,AL
231 0088 8C 06 009A R  MOV     @USER_FLAG_SEG,ES    ; SET UP TRANSFER TABLE
232 008C 89 1E 009B R  MOV     @USER_FLAG,BX
233 0090 89 0E 009E R  MOV     @RTC_HIGH,CX
234 0094 89 16 009C R  MOV     @RTC_LOW,DX
235 0098 C6 06 00A0 R 01 MOV     @RTC_WAIT_FLAG,01    ; SET ON FUNCTION ACTIVE SWITCH
236 009D B0 0B      MOV     AL,CMOS_REG_B        ; ENABLE PIE
237 009F E8 0000 E    CALL    CMOS_READ           ; READ CMOS LOCATION
238 00A2 24 7F      AND     AL,07FH             ; CLEAR SET
239 00A4 0C 40      OR     AL,040H              ; ENABLE PIE
240 00A6 50         PUSH    AX                   ; SAVE AH
241 00A7 8A E0      MOV     AH,AL                ; PLACE DATA INTO DATA REGISTER
242 00A9 B0 0B      MOV     AL,CMOS_REG_B        ; ADDRESS ALARM REGISTER
243 00AB E8 0000 E    CALL    CMOS_WRITE          ; PLACE DATA IN AH INTO ALARM REGISTER
244 00AE 58         POP     AX                   ; RESTORE AH
245 00AF 1F         POP     DS                    ; RESTORE DS
246 00B0 FB      STI     ; ENABLE INTERRUPTS
247 00B1 F8      CLC     ; CLEAR CARRY
248 00B2 EB A1      JMP     C1_F
249
250          ;----- CANCEL
251
252 00B4          EVENT_WAIT_3:
253 00B4 FA      CLI     ; DISABLE INTERRUPTS
254 00B5 F6 06 00A0 R 02 TEST    @RTC_WAIT_FLAG,02H    ; CHECK FOR "WAIT" IN PROGRESS
255 00BA 74 05      JZ     EVENT_WAIT_4          ; SKIP TO CANCEL CURRENT "EVENT_WAIT"
256
257 00BC FB      STI     ; ENABLE INTERRUPTS
258 00BD 1F      POP     DS                    ; CLEAR STACK
259 00BE F9      STC     ; AND SET CARRY FLAG FOR ERROR REQUEST
260 00BF EB 94      JMP     C1_F                 ; EXIT
261
262 00C1          EVENT_WAIT_4:
263 00C1 50         MOV     AX,X*CMOS_REG_B      ; SAVE (WITH INTERRUPTS DISABLED)
264 00C2 B8 0B0B    MOV     CMOS_READ            ; TURN OFF PIE
265 00C5 E8 0000 E    CALL    AL,0BFH             ; GET ALARM REGISTER
266 00C8 24 BF      AND     AH,AL                ; CLEAR PIE
267 00CA 86 E0      XCHG   AH,AL                ; PLACE INTO WRITE REGISTER
268 00CC E8 0000 E    CALL    CMOS_WRITE          ; WRITE BACK TO ALARM REGISTER
269 00CF 58         POP     AX                   ; RESTORE AH
270 00D0 C6 06 00A0 R 00 MOV     @RTC_WAIT_FLAG,0     ; SET FUNCTION ACTIVE FLAG OFF
271 00D5 FB      STI     ; ENABLE INTERRUPTS
272 00D6 1F      POP     DS                    ; RESTORE DATA SEGMENT
273 00D7 F8      CLC     ; SET CARRY OFF
274 00D8 E9 0055 R  JMP     C1_F                 ; RETURN
275
276 00DB          EVENT_WAIT  ENDP
277          ;--- JOY STICK
278          ;----- THIS ROUTINE WILL READ THE JOYSTICK PORT
279          ;-----
280          ;-----
281          ;----- INPUT READ THE CURRENT SWITCHES
282          ;----- (DX)=0 RETURNS (AL)= SWITCH SETTINGS IN BITS 7-4
283          ;-----
284          ;----- (DX)=1 READ THE RESISTIVE INPUTS
285          ;----- RETURNS (AX)=A(X) VALUE
286          ;----- (BX)=B(Y) VALUE
287          ;----- (CX)=B(X) VALUE
288          ;----- (DX)=B(Y) VALUE
289          ;-----
290          ;----- CY FLAG ON IF NO ADAPTER CARD OR INVALID CALL
291          ;-----
292
293 00DB          JOY_STICK  PROC  NEAR
294 00DB FB      STI     ; INTERRUPTS BACK ON
295 00DD 8B C2      MOV     AX,DX                ; GET SUB FUNCTION CODE
296 00DE B8 201H    MOV     DX,201H              ; ADDRESS OF PORT
297 00E1 0A C0      OR     AL,AL
298 00E3 74 0B      JZ     JOY_2                  ; READ SWITCHES
299 00E5 FE C8      DEC     AL                    ; READ RESISTIVE INPUTS
300 00E7 74 0C      JZ     JOY_3                  ; GO TO ERROR RETURN
301 00E9 E9 0052 R  JMP     C1                     ; GO TO COMMON RETURN
302 00EC
303 00EC FB      STI     ; INTERRUPTS BACK ON
304 00ED E9 0055 R  JMP     C1                     ; GO TO COMMON RETURN
305
306 00F0          JOY_2:
307 00F0 EC      IN      AL,DX                ; STRIP UNWANTED BITS OFF
308 00F1 24 F0      AND     AL,0F0H              ; FINISHED
309 00F3 EB F7      JMP     C1
310
311 00F5          JOY_3:
312 00F5 B3 01      MOV     BL,1
313 00F7 E8 0113 R  CALL    TEST_CORD            ; SAVE A(X) VALUE
314 00FA 51         PUSH    CX
315 00FB B3 02      MOV     BL,2
316 00FD E8 0113 R  CALL    TEST_CORD            ; SAVE A(Y) VALUE
317 0100 51         PUSH    CX
318 0101 B3 04      MOV     BL,4
319 0103 E8 0113 R  CALL    TEST_CORD            ; SAVE B(X) VALUE
320 0106 51         PUSH    CX
321 0107 B3 08      MOV     BL,8
322 0109 E8 0113 R  CALL    TEST_CORD            ; SAVE B(Y) VALUE
323 010C 8B D1      MOV     DX,CX
324 010E 59      POP     CX                    ; GET B(X) VALUE
325 0110 5B      POP     BX                    ; GET A(Y) VALUE
326 0112 5A      POP     AX                    ; GET A(X) VALUE
327 0114 EB D9      JMP     JOY_1                  ; FINISHED - RETURN
328
329 0113          TEST_CORD  PROC  NEAR
330 0113 52         PUSH    DX                    ; SAVE
331 0114 FA      CLI     ; BLOCK INTERRUPTS WHILE READING
332 0115 B0 00      MOV     AL,0                 ; SET UP TO LATCH TIMER 0
333 0117 E6 42      OUT     TIMER+3,AL
334 0119 EB 00      JMP     $+2
335 011B E4 40      IN      AL,TIMER             ; READ LOW BYTE OF TIMER 0
336 011D EB 00      JMP     $+2
337 011F 8A E0      MOV     AH,AL
338 0121 E4 40      IN      AL,TIMER             ; READ HIGH BYTE OF TIMER 0
339 0123 86 E0      XCHG   AH,AL                ; REARRANGE TO HIGH,LOW

```



```

449 PAGE
450 --- INT 15 H -- ( FUNCTION 87 H - BLOCK MOVE ) -----
451 |
452 | THIS BIOS FUNCTION PROVIDES A MEANS FOR A REAL MODE PROGRAM OR SYSTEM
453 | TO TRANSFER A BLOCK OF STORAGE TO AND FROM STORAGE ABOVE THE 1 MEG
454 | ADDRESS RANGE IN PROTECTED MODE SPACE BY SWITCHING TO PROTECTED MODE.
455 |
456 | ENTRY:
457 | (AH) = 87H (FUNCTION CALL) - BLOCK MOVE.
458 | (CX) = WORD COUNT OF STORAGE BLOCK TO BE MOVED.
459 | ES:SI = LOCATION OF A GDT TABLE BUILT BY ROUTINE USING THIS FUNCTION.
460 |
461 | (ES:SI) POINTS TO A DESCRIPTOR TABLE (GDT) BUILT BEFORE INTERRUPTING
462 | TO THIS FUNCTION. THE DESCRIPTORS ARE USED TO PERFORM THE BLOCK
463 | MOVE IN THE PROTECTED MODE. THE SOURCE AND TARGET DESCRIPTORS
464 | BUILT BY THE USER MUST HAVE A SEGMENT LENGTH = 2 * CX-1 OR GREATER.
465 | THE DATA ACCESS RIGHTS BYTE MUST BE SET TO CPL0-R/W (93H). THE
466 | 24 BIT ADDRESS (BYTE HI, WORD LOW) MUST BE SET TO THE TARGET/SOURCE.
467 |
468 | *** NO INTERRUPTS ARE ALLOWED DURING TRANSFER. LARGE BLOCK MOVES
469 | MAY CAUSE LOST INTERRUPTS.
470 |
471 |
472 | EXIT:
473 | (AH) = 00H IF SUCCESSFUL.
474 | (AH) = 01H IF MEMORY PARITY (PARITY ERROR REGISTERS ARE CLEARED)
475 | (AH) = 02H IF ANY OTHER EXCEPTION INTERRUPT ERROR OCCURRED
476 | (AH) = 03H IF GATE ADDRESS LINE 20 FAILED
477 | ALL REGISTERS ARE RESTORED EXCEPT (AH).
478 |
479 | IF SUCCESSFUL - CARRY FLAG = 0
480 | IF ERROR ----- CARRY FLAG = 1
481 |
482 | DESCRIPTION:
483 |
484 | 1. SAVE ENTRY REGISTERS AND SETUP FOR SHUTDOWN EXIT.
485 | 2. THE REQUIRED ENTRIES ARE BUILT IN THE GDT AT (ES:SI).
486 | 3. GATE ADDRESS LINE 20 ACTIVE, CLI AND SET SHUTDOWN CODES.
487 | 4. THE IDTR IS LOADED AND POINTS TO A ROM RESIDENT TABLE.
488 | 5. THE GDTR IS LOADED FROM THE OFFSET POINTER (ES:SI).
489 | 6. THE PROCESSOR IS PUT INTO PROTECTED MODE.
490 | 7. LOAD (DS) AND (ES) WITH SELECTORS FOR THE SOURCE AND TARGET.
491 | 8. DS:SI (SOURCE) (ES:DI) (TARGET) REP MOVSW IS EXECUTED.
492 | 9. CHECK MADE FOR PARITY ERRORS.
493 | 10. REAL MODE RESTORED WHEN SHUTDOWN 09H IS EXECUTED.
494 | 11. ERRORS ARE CHECKED FOR AND RETURN CODES ARE SET FOR (AH).
495 | 12. ADDRESS LINE 20 GATE IS DISABLED.
496 | 13. RETURN WITH REGISTERS RESTORED AND STATUS RETURN CODE.
497 | (FOR PC-AT COMPATIBILITY ZF=1 IF SUCCESSFUL, ZF=0 IF ERROR.)
498 |
499 |-----
500 | THE FOLLOWING DIAGRAM DEPICTS THE ORGANIZATION OF A BLOCK MOVE GDT.
501 |
502 | G D T
503 | (ES:SI)
504 |
505 | +00 DUMMY
506 |
507 | +06 GDTR LOC
508 |
509 | +10 SOURCE
510 | GDTR
511 |
512 | +18 TARGET
513 | GDTR
514 |
515 | +20 BIOS
516 | (CS)
517 |
518 | +28 (SS)
519 |
520 |
521 | 1. THE FIRST DESCRIPTOR IS THE REQUIRED DUMMY.
522 | (USER INITIALIZED TO 0)
523 |
524 | 2. THE SECOND DESCRIPTOR POINTS TO THE GDTR
525 | TABLE AS A DATA SEGMENT.
526 | (USER INITIALIZED TO 0 - MODIFIED BY BIOS)
527 |
528 | 3. THE THIRD DESCRIPTOR POINTS TO THE SOURCE
529 | TO BE MOVED. (FROM)
530 | (USER INITIALIZED)
531 |
532 | 4. THE FOURTH DESCRIPTOR POINTS TO THE
533 | DESTINATION SEGMENT. (TO)
534 | (USER INITIALIZED)
535 |
536 | 5. THE FIFTH IS A DESCRIPTOR THAT BIOS USES
537 | TO CREATE THE PROTECTED MODE CODE SEGMENT.
538 | (USER INITIALIZED TO 0 - MODIFIED BY BIOS)
539 |
540 | 6. THE SIXTH DESCRIPTOR IS USED BY BIOS TO
541 | CREATE A PROTECTED MODE STACK SEGMENT.
542 | (POINTS TO USER'S STACK)
543 |
544 |
545 | SAMPLE OF SOURCE OR TARGET DESCRIPTOR
546 | SOURCE_TARGET_DEF STRUC
547 |
548 | SEG_LIMIT DW ? ; SEGMENT LIMIT ((1-65536) BYTES)
549 | LO_WORD DW ? ; 24 BIT SEGMENT PHYSICAL
550 | HI_BYTE DB ? ; ADDRESS (0 TO (16M-1))
551 | DATA_ACC_RIGHTS DB 93H ; ACCESS RIGHTS BYTE (CPL0-R/W)
552 | RESERVED DW 0 ; RESERVED WORD (MUST BE ZERO)
553 |
554 | SOURCE_TARGET_DEF ENDS
555 |
556 |-----
557 | THE GLOBAL DESCRIPTOR TABLE (ACTUAL LOCATION POINTED TO BY ES:SI)
558 |
559 | BLOCKMOVE_GDT_DEF STRUC
560 | DQ ? ; FIRST DESCRIPTOR NOT ACCESSIBLE
561 | DQ ? ; LOCATION OF CALLING ROUTINE GDT
562 | DQ ? ; SOURCE DESCRIPTOR
563 | DQ ? ; TARGET DESCRIPTOR
564 | DQ ? ; BIOS CODE DESCRIPTOR
565 | DQ ? ; STACK DESCRIPTOR
566 | BLOCKMOVE_GDT_DEF ENDS
567 |
568 | BLOCKMOVE PROC NEAR
569 |
570 | CLD ; SET DIRECTION FORWARD
571 | PUSHA ; SAVE GENERAL PURPOSE REGISTERS
572 | PUSH ES ; SAVE USER'S EXTRA SEGMENT
573 | PUSH DS ; SAVE USER'S DATA SEGMENT
574 |
575 | I----- SAVE THE CALLING ROUTINE'S STACK
576 |
577 | 01DA E8 0000 E CALL DDS ; SET DS TO DATA AREA

```

```

563 01DD 8C 16 0069 R      MOV     @10_ROM_SEG,SS      ; SAVE USERS STACK SEGMENT
564 01E1 89 26 0067 R      MOV     @10_ROM_INIT,SP    ; SAVE USERS STACK POINTER
565
566 ;===== SET UP THE PROTECTED MODE DEFINITIONS =====
567 ;----- MAKE A 24 BIT ADDRESS OUT OF THE ES:SI FOR THE GDT POINTER
568
569
570 ASSUME DS:NOTHING        ; POINT (DS) TO USERS CONTROL BLOCK
571 MOV     AX,ES             ; GET THE GDT DATA SEGMENT
572 MOV     DS,AX             ; MOVE THE GDT SEGMENT POINTER TO (DS)
573 MOV     DH,AH             ; BUILD HIGH BYTE OF THE 24 BIT ADDRESS
574 SHR     DH,4              ; USE ONLY HIGH NIBBLE SHIFT - RIGHT 4
575 SHL     AX,4              ; STRIP HIGH NIBBLE FROM (AX)
576 ADD     AX,SI             ; ADD THE GDT OFFSET TO DEVELOP LOW WORD
577 ADC     DH,0              ; ADJUST HIGH BYTE IF CARRY FROM LOW
578
579 ;----- SET THE GDT_LOC
580
581 MOV     [SI].CGDT_LOC.SEG_LIMIT,MAX_SEG_LEN
582 MOV     [SI].CGDT_LOC.BASE_LO_WORD,AX  ; SET THE LOW WORD
583 MOV     [SI].CGDT_LOC.BASE_HI_BYTE,DH  ; SET THE HIGH BYTE
584 MOV     [SI].CGDT_LOC.DATA_RESERVED,0  ; RESERVED
585
586 ;----- SET UP THE CODE SEGMENT DESCRIPTOR
587
588 MOV     [SI].BIOS_CS.SEG_LIMIT,MAX_SEG_LEN
589 MOV     [SI].BIOS_CS.BASE_LO_WORD,CSEG@_LO  ; LOW WORD OF (CS) = 0
590 MOV     [SI].BIOS_CS.BASE_HI_BYTE,CSEG@_HI  ; HIGH BYTE OF (CS) = 0FH
591 MOV     [SI].BIOS_CS.DATA_ACC_RIGHTS,CPL0_CODE_ACCESS
592 MOV     [SI].BIOS_CS.DATA_RESERVED,0      ; RESERVED
593
594 ;----- MAKE A 24 BIT ADDRESS OUT OF THE (SS) - ( SP) REMAINS USER (SP)
595
596 MOV     AX,SS             ; GET THE CURRENT STACK SEGMENT
597 MOV     DH,AH             ; FORM HIGH BYTE OF 24 BIT ADDRESS
598 SHR     DH,4              ; FORM HIGH BYTE - SHIFT RIGHT 4
599 SHL     AX,4              ; STRIP HIGH NIBBLE FROM (AX)
600
601 ;----- SS IS NOW IN POSITION FOR A 24 BIT ADDRESS --> SETUP THE (SS) DESCRIPTOR
602
603 MOV     [SI].TEMP_SS.SEG_LIMIT,MAX_SEG_LEN  ; SET THE SS SEGMENT LIMIT
604 MOV     [SI].TEMP_SS.BASE_LO_WORD,AX        ; SET THE LOW WORD
605 MOV     [SI].TEMP_SS.BASE_HI_BYTE,DH        ; SET THE HIGH BYTE
606 MOV     [SI].TEMP_SS.DATA_ACC_RIGHTS,CPL0_DATA_ACCESS  ; SET CPL 0
607
608 ;----- GATE ADDRESS BIT 20 ON (DISABLE INTERRUPTS)
609
610 MOV     AH,ENABLE_BIT20  ; GET ENABLE MASK
611 CALL   GATE_A20         ; ENABLE A20 AND CLEAR INTERRUPTS
612 CMP     AL,0             ; WAS THE COMMAND ACCEPTED?
613 JZ      BL4              ; GO IF YES
614
615 MOV     AL,03H           ; SET THE ERROR FLAG IF NOT
616 OUT    MFG_PORT,AL     ; EARLY ERROR EXIT
617 JMP    SHORT SHUT9
618
619 ;----- SET SHUTDOWN RETURN ADDRESS AND DISABLE NMI
620
621 BL4:
622 MOV     AX,9*H+CMOS_SHUT_DOWN+NMI          ; SET THE SHUTDOWN BYTE LOCATION
623 CALL   CMOS_WRITE                          ; TO SHUT DOWN 9 AND DISABLE NMI
624
625 ;----- CLEAR EXCEPTION ERROR FLAG
626
627 SUB    AL,AL
628 OUT    MFG_PORT,AL  ; SET ERROR FLAG LOCATION TO 0
629
630 ;----- LOAD THE IDT AND GDT
631
632 MOV     BP,OFFSET ROM_IDT_LOC
633 MOV     SEG0V CS          ; LOAD THE IDT
634 DB     02E2 2E          +
635 LIDT  [BP]              + REGISTER FROM THIS AREA
636 DB     02E3 0F          +
637 LABEL BYTE             +
638 MOV     BX,WORD PTR [BP] +
639 LABEL BYTE             +
640 ORG    OFFSET CS:??0001 +
641 DB     02E4 01          +
642 ORG    OFFSET CS:??0002 +
643
644 LGDT  [SI].CGDT_LOC    ; LOAD GLOBAL DESCRIPTOR TABLE REGISTER
645 DB     02E7 0F          +
646 LABEL BYTE             +
647 MOV     DX,WORD PTR [SI].CGDT_LOC
648 LABEL BYTE             +
649 ORG    OFFSET CS:??0003 +
650 DB     02E8 01          +
651 ORG    OFFSET CS:??0004 +
652
653 ;----- SWITCH TO VIRTUAL MODE
654
655 MOV     AX,VIRTUAL_ENABLE  ; MACHINE STATUS WORD NEEDED TO
656 LMSW  AX                 ; SWITCH TO VIRTUAL MODE
657
658 DB     02E5 0F 01 F0H    +
659 DB     02E6 0E 0E 0E  ; PURGE PRE-FETCH QUEUE WITH FAR JUMP
660 DB     02E7 0E 0E 0E  ; TO OFFSET
661 DW     BIOS_CS          ; - IN SEGMENT -PROTECTED MODE SELECTOR
662
663 ;----- IN PROTECTED MODE - SETUP STACK SELECTOR AND SOURCE/TARGET SELECTORS
664
665 MOV     AX,TEMP_SS        ; USER'S SS+SP IS NOT A DESCRIPTOR
666 MOV     SS,AX            ; LOAD STACK SELECTOR
667 MOV     AX,SOURCE        ; GET THE SOURCE ENTRY
668 MOV     DS,AX            ; LOAD SOURCE SELECTOR
669 MOV     AX,TARGET        ; GET THE TARGET ENTRY
670 MOV     ES,AX            ; LOAD TARGET SELECTOR
671 SUB    SI,SI             ; SET SOURCE INDEX REGISTER TO ZERO
672 SUB    DI,DI             ; SET TARGET INDEX REGISTER TO ZERO
673
674 REP    MOVSW             ; MOVE THE BLOCK COUNT PASSED IN (CX)
675
676

```

SECTION 5

```

677                                     ;----- CHECK FOR MEMORY PARITY BEFORE SHUTDOWN
678
679 027B E4 61                          IN     AL,PORT_B           ; GET THE PARITY LATCHES
680 027D 24 C0                          AND    AL,PARITY_ERR     ; STRIP UNWANTED BITS
681 027F 74 12                          JZ     DONE1             ; GO IF NO PARITY ERROR
682
683                                     ;----- CLEAR PARITY BEFORE SHUTDOWN
684
685 0281 8B 05                          MOV    AX,DS:[DI]        ; FETCH CURRENT SOURCE DATA
686 0283 89 05                          MOV    DS:[DI],AX       ; WRITE IT BACK
687 0285 B0 01                          MOV    AL,01             ; SET PARITY CHECK ERROR = 01
688 0287 E6 80                          OUT    MFG_PORT,AL      ;
689 0289 E4 61                          IN     AL,PORT_B         ;
690 028B 0C 0C                          OR     AL,RAM_PAR_OFF    ; TOGGLE PARITY CHECK LATCHES
691 028D E6 61                          OUT    PORT_B,AL        ; TO CLEAR THE PENDING ERROR
692 028F 24 F3                          AND    AL,RAM_PAR_ON    ; AND ENABLE CHECKING
693 0291 E6 61                          OUT    PORT_B,AL        ;
694
695                                     ;----- CAUSE A SHUTDOWN
696
697 0293                                     DONE1:
698 0293 E9 0000 E                       JMP    PROC_SHUTDOWN    ; GO RESET PROCESSOR AND SHUTDOWN
699
700                                     ;----- RETURN FROM SHUTDOWN
701                                     ;-----
702                                     ;-----
703
704 0296                                     SHUT9:
705                                     ; RESTORE USERS STACK
706 0296 B8 ---- R                       ASSUME DS:DATA          ;
707 0299 8E 0B                          MOV    AX,DATA          ; SET DS TO DATA AREA
708 029B 8E 16 0069 R                   MOV    DS,AX            ;
709 029F 8B E6 0067 R                   MOV    SS,@10_ROM_SEG  ; GET USER STACK SEGMENT
710                                     ; GET USER STACK POINTER
711
712                                     ;----- GATE ADDRESS BIT 20 OFF
713 02A3 B4 0D                          MOV    AH,DISABLE_BIT20 ; DISABLE MASK
714 02A5 E8 03DA R                       CALL   GATE_A20         ; GATE ADDRESS 20 LINE OFF
715 02A8 3C 00                          CMP    AL,0             ; COMMAND ACCEPTED?
716 02AA 74 0A                          JZ     DONES            ; GO IF YES
717
718 02AC E4 80                          IN     AL,MFG_PORT      ; CHECK FOR ANY OTHER ERROR FIRST
719 02AE 3C 00                          CMP    AL,0             ; WAS THERE AN ERROR?
720 02B0 75 04                          JNZ   DONES3           ; REPORT FIRST ERROR IF YES
721 02B2 B0 03                          MOV    AL,03H          ; ELSE SET GATE A20 ERROR FLAG
722 02B4 E6 80                          OUT    MFG_PORT,AL     ;
723
724                                     ;----- RESTORE THE USERS REGISTERS AND SET RETURN CODES
725
726 02B6                                     DONES3:
727 02B6 B8 000F                       MOV    AX,CMOS_SHUT_DOWN ; CLEAR (AH) TO ZERO AND (AL) TO DEFAULT
728 02B9 E6 70                          OUT    CMOS_PORT,AL    ; ENABLE NM! INTERRUPTS
729 02BB 1F                             POP    DS               ; RESTORE USER DATA SEGMENT
730 02BC 07                             POP    ES               ; RESTORE USER EXTRA SEGMENT
731 02BD E4 71                          IN     AL,CMOS_DATA     ; OPEN CMOS STANDBY LATCH
732
733 02BF E4 80                          IN     AL,MFG_PORT      ; GET THE ENDING STATUS RETURN CODE
734 02C1 8B EC                          MOV    BP,SP            ; POINT TO REGISTERS IN THE STACK
735 02C3 88 46 0F                       MOV    [BP+15],AL       ; PLACE ERROR CODE INTO STACK AT (AH)
736 02C6 3A ED                          CMP    AH,AL            ; SET THE ZF & CY FLAGS WITH RETURN CODE
737 02C8 61                             POPA                    ; RESTORE THE GENERAL PURPOSE REGISTERS
738 02C9 FB                             STI                     ; TURN INTERRUPTS ON
739 02CA                                     DONE4:
740 02CA CA 0002                       RET     2                ; RETURN WITH FLAGS SET -- (AH)= CODE
741 02CD                                     DONE4:
742                                     ; (CF=0,ZF=1)= OK (CF=1,ZF=0)= ERROR
743
744                                     ;----- BLOCK MOVE EXCEPTION INTERRUPT HANDLER
745
746 02CD B0 02                          EX_INT:
747 02CF E6 80                          MOV    AL,02H          ; GET EXCEPTION ERROR CODE
748 02D1 E9 0000 E                       OUT    MFG_PORT,AL     ; SET EXCEPTION INTERRUPT OCCURRED FLAG
749                                     ; CAUSE A EARLY SHUTDOWN
750
751                                     ;----- ROM IDT LOCATION
752
753 02D4                                     ROM_IDT_LOC:
754 02D4 0100                          DW    ROM_IDT_END-ROM_IDT ; LENGTH OF ROM IDT TABLE
755 02D6 02DA R                        DW    ROM_IDT           ; LOW WORD OF BASE ADDRESS
756 02D8 0F                             DB    CSEG#_HI          ; HIGH BYTE OF BASE ADDRESS
757 02D9 00                             DB    0                 ; RESERVED
758
759                                     ;----- THE ROM EXCEPTION INTERRUPT VECTOR GATES FOR BLOCK MOVE
760
761 02DA                                     ROM_IDT:
762 02DA 02CD R                        DW    EX_INT           ; EXCEPTION 00
763 02DC 0020                          DW    BIOS_CS          ; DESTINATION OFFSET
764 02DE 00                             DB    0                ; DESTINATION SEGMENT SELECTOR
765 02DF 87                             DB    0                ; WORD COPY COUNT
766 02E0 0000                          DW    0                ; GATE TYPE - ACCESS RIGHTS BYTE
767                                     ; RESERVED
768 02E2 02CD R                        DW    EX_INT           ; EXCEPTION 01
769 02E4 0020                          DW    BIOS_CS          ; DESTINATION OFFSET
770 02E6 00                             DB    0                ; DESTINATION SEGMENT SELECTOR
771 02E7 87                             DB    0                ; WORD COPY COUNT
772 02E8 0000                          DW    0                ; GATE TYPE - ACCESS RIGHTS BYTE
773                                     ; RESERVED
774 02EA 02CD R                        DW    EX_INT           ; EXCEPTION 02
775 02EC 0020                          DW    BIOS_CS          ; DESTINATION OFFSET
776 02EE 00                             DB    0                ; DESTINATION SEGMENT SELECTOR
777 02EF 87                             DB    0                ; WORD COPY COUNT
778 02F0 0000                          DW    0                ; GATE TYPE - ACCESS RIGHTS BYTE
779                                     ; RESERVED
780 02F2 02CD R                        DW    EX_INT           ; EXCEPTION 03
781 02F4 0020                          DW    BIOS_CS          ; DESTINATION OFFSET
782 02F6 00                             DB    0                ; DESTINATION SEGMENT SELECTOR
783 02F7 87                             DB    0                ; WORD COPY COUNT
784 02F8 0000                          DW    0                ; GATE TYPE - ACCESS RIGHTS BYTE
785                                     ; RESERVED
786 02FA 02CD R                        DW    EX_INT           ; EXCEPTION 04
787 02FC 0020                          DW    BIOS_CS          ; DESTINATION OFFSET
788 02FE 00                             DB    0                ; DESTINATION SEGMENT SELECTOR
789 02FF 87                             DB    0                ; WORD COPY COUNT
790 0300 0000                          DW    0                ; GATE TYPE - ACCESS RIGHTS BYTE
791                                     ; RESERVED
792                                     ; EXCEPTION 05
    
```

```

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

```

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

```

955 PAGE
956
957 ; GATE_A20
958 ; THIS ROUTINE CONTROLS A SIGNAL WHICH GATES ADDRESS BIT 20.
959 ; THE GATE A20 SIGNAL IS AN OUTPUT OF THE 8042 SLAVE PROCESSOR.
960 ; ADDRESS BIT 20 SHOULD BE GATED ON BEFORE ENTERING PROTECTED MODE.
961 ; IT SHOULD BE GATED OFF AFTER ENTERING REAL MODE FROM PROTECTED
962 ; MODE. INTERRUPTS ARE LEFT DISABLED ON EXIT.
963
964 ; INPUT
965 ; (AH) = DDH ADDRESS BIT 20 GATE OFF. (A20 ALWAYS ZERO)
966 ; (AH) = DFH ADDRESS BIT 20 GATE ON. (A20 CONTROLLED BY 80286)
967
968 ; OUTPUT
969 ; (AL) = 00H OPERATION SUCCESSFUL. 8042 HAS ACCEPTED COMMAND.
970 ; (AL) = 02H FAILURE--8042 UNABLE TO ACCEPT COMMAND.
971
972 -----
973 GATE_A20 PROC
974     PUSH    CX                ; SAVE USERS (CX)
975     CLI                    ; DISABLE INTERRUPTS WHILE USING 8042
976     CALL    EMPTY_8042       ; INSURE 8042 INPUT BUFFER EMPTY
977     JNZ    GATE_A20_RETURN  ; EXIT IF 8042 UNABLE TO ACCEPT COMMAND
978     MOV    AL,0D1H          ; 8042 COMMAND TO WRITE OUTPUT PORT
979     OUT    STATUS_PORT,AL   ; OUTPUT COMMAND TO 8042
980     CALL    EMPTY_8042       ; WAIT FOR 8042 TO ACCEPT COMMAND
981     JNZ    GATE_A20_RETURN  ; EXIT IF 8042 UNABLE TO ACCEPT COMMAND
982     MOV    AL,AH            ; 8042 PORT DATA
983     OUT    PORT_A,AL        ; OUTPUT PORT DATA TO 8042
984     CALL    EMPTY_8042       ; WAIT FOR 8042 TO ACCEPT PORT DATA
985
986 ;----- 8042 OUTPUT WILL SWITCH WITHIN 20 MICRO SECONDS OF ACCEPTING PORT DATA
987
988 GATE_A20_RETURN:
989     POP     CX                ; RESTORE USERS (CX)
990     RET
991
992 -----
993 ; EMPTY_8042
994 ; THIS ROUTINE WAITS FOR THE 8042 INPUT BUFFER TO EMPTY.
995
996 ; INPUT
997 ; NONE
998
999 ; OUTPUT
1000 ; (AL) = 00H 8042 INPUT BUFFER EMPTY (ZERO FLAG SET)
1001 ; (AL) = 02H TIME OUT, 8042 INPUT BUFFER FULL (NON-ZERO FLAG SET)
1002 ; (CX) - MODIFIED
1003
1004 -----
1005 EMPTY_8042:
1006     SUB    CX,CX                ; (CX)=0, WILL BE USED AS TIME OUT VALUE
1007     IN    AL,STATUS_PORT       ; READ 8042 STATUS PORT
1008     AND    AL,INPT_BDF_FULL    ; TEST INPUT BUFFER FULL FLAG (BIT 1)
1009     LOOPNZ EMPTY_L            ; LOOP UNTIL BUFFER EMPTY OR TIME OUT
1010     RET
1011
1012 GATE_A20 ENDP
1013
1014
1015 ;--- INT 15 H --- ( FUNCTION 88 H - I/O MEMORY SIZE DETERMINE ) -----
1016 ; EXT_MEMORY
1017 ; THIS ROUTINE RETURNS THE AMOUNT OF MEMORY IN THE SYSTEM THAT IS
1018 ; LOCATED STARTING AT THE 1024K ADDRESSING RANGE, AS DETERMINED BY
1019 ; THE POST ROUTINES.
1020 ; NOTE THAT THE SYSTEM MAY NOT BE ABLE TO USE I/O MEMORY UNLESS THERE
1021 ; IS A FULL COMPLEMENT OF 512K OR 640 BYTES ON THE PLANAR. THIS SIZE
1022 ; SIZE IS STORED IN CMOS AT ADDRESS LOCATIONS 30H AND 31H.
1023
1024 ; INPUT
1025 ; AH = 88H
1026
1027 ; THE I/O MEMORY SIZE VARIABLE IS SET DURING POWER ON
1028 ; DIAGNOSTICS ACCORDING TO THE FOLLOWING ASSUMPTIONS:
1029 ;
1030 ; 1. ALL INSTALLED MEMORY IS FUNCTIONAL.
1031 ; 2. ALL MEMORY FROM 0 TO 640K MUST BE CONTIGUOUS.
1032
1033 ; OUTPUT
1034 ; (AX) = NUMBER OF CONTIGUOUS 1K BLOCKS OF MEMORY A
1035 ; AVAILABLE STARTING AT ADDRESS 1024K.
1036
1037 -----
1038 EXT_MEMORY PROC
1039     MOV    AX,CMOS_U_M_S_LO*H+CMOS_U_M_S_HI ; ADDRESS HIGH/LOW BYTES
1040     CALL    CMOS_READ                ; GET THE HIGH BYTE OF I/O MEMORY
1041     XCHG   AL,AR                      ; PUT HIGH BYTE IN POSITION (AH)
1042     CALL    CMOS_READ                ; GET THE LOW BYTE OF I/O MEMORY
1043     IRET                             ; RETURN TO USER
1044
1045 EXT_MEMORY ENDP
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500

```

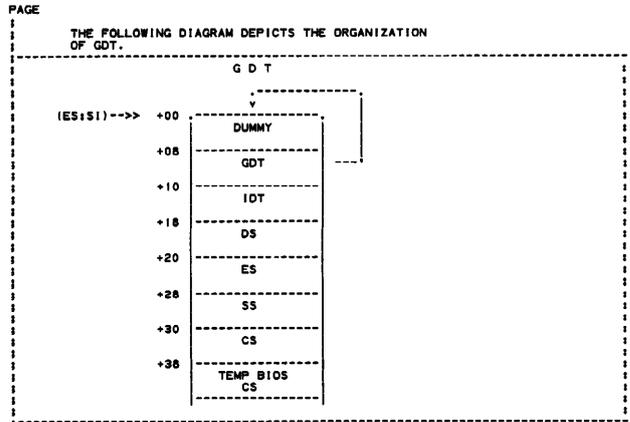
SECTION 5

```

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

```

1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189 0000 ??????????????????
1190 0008 ??????????????????
1191 0010 ??????????????????
1192 0018 ??????????????????
1193 0020 ??????????????????
1194 0028 ??????????????????
1195 0030 ??????????????????
1196 0038 ??????????????????
1197 0040
1198
1199
1200
1201 0408
1202 0408
1203
1204
1205
1206 0408 FA
1207 0409 B4 DF
1208 040B E8 03DA R
1209 040E 3C 00
1210 0410 74 04
1211 0412 B4 FF
1212 0414 F9
1213 0415 CF
1214
1215
1216 0416
1217 0416 06
1218 0417 1F
1219
1220
1221
1222
1223
1224 0418 B0 11
1225 041A E6 20
1226 041C EB 00
1227 041E 8A C7
1228 0420 E6 21
1229 0422 EB 00
1230 0424 B0 04
1231 0426 E6 21
1232 0428 EB 00
1233 042A B0 01
1234 042C E6 21
1235 042E EB 00
1236 0430 B0 FF
1237 0432 E6 21
1238
1239
1240
1241
1242
1243 0434 B0 11
1244 0436 E6 A0
1245 0438 EB 00
1246 043A 8A C9
1247 043C E6 A1
1248 043E B0 02
1249 0440 EB 00
1250 0442 E6 A1
1251 0444 EB 00
1252 0446 B0 01
1253 0448 E6 A1
1254 044A EB 00
1255 044C B0 FF
1256 044E E6 A1
    
```



THE GLOBAL DESCRIPTOR TABLE (ACTUAL LOCATION POINTED TO BY ES:SI)

```

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

```

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

```

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

```

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

```

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

SECTION 5

```

1257 PAGE
1258 ;-----
1259 ; SETUP BIOS CODE SEGMENT DESCRIPTOR ;
1260 ;-----
1261
1262 0450 C7 44 38 FFFF MOV [SI],BIO_CS.SEG_LIMIT,MAX_SEG_LEN ; SET LENGTH
1263 0455 C6 44 3C 0F MOV [SI],BIO_CS.BASE_HI_BYTE,CSEG_HI ; SET HIGH BYTE OF CS=0F
1264 0459 C7 44 3A 0000 MOV [SI],BIO_CS.BASE_LO_WORD,CSEG_LO ; SET LOW WORD OF CS=0
1265 045E C6 44 3D 9B MOV [SI],BIO_CS.DATA_ACE_RIGHTS,CPLD_CODE_ACCESS
1266 0462 C7 44 3E 0000 MOV [SI],BIO_CS.DATA_RESERVED,0 ; ZERO RESERVED AREA
1267
1268 ;-----
1269 ; ENABLE PROTECTED MODE ;
1270 ;-----
1271 LGDT [SI],GDTPTR ; LOAD GLOBAL DESCRIPTOR TABLE REGISTER
1272 0467 0F + DB 00FH
1273 0468 + ?77005 LABEL BYTE
1274 0468 8B 54 08 + MOV DX,WORD PTR [SI],GDTPTR
1275 0468 + ?77006 LABEL BYTE
1276 0468 + ORG OFFSET CS:??7005
1277 0468 01 + DB 001H
1278 0468 + ORG OFFSET CS:??7006
1279 LIDT [SI],IDTPTR ; INTERRUPT DESCRIPTOR TABLE REGISTER
1280 046B 0F + DB 00FH
1281 046C + ?77007 LABEL BYTE
1282 046C 8B 5C 10 + MOV BX,WORD PTR [SI],IDTPTR
1283 046F + ?77008 LABEL BYTE
1284 046C + ORG OFFSET CS:??7007
1285 046C 01 + DB 001H
1286 046F + ORG OFFSET CS:??7008
1287
1288 046F B8 0001 MOV AX,VIRTUAL_ENABLE ; MACHINE STATUS WORD NEEDED TO
1289 LMSW AX ; SWITCH TO VIRTUAL MODE
1290 0472 0F 01 F0 + DB 00FH,001H,0F0H
1291 0475 EA + DB 0EAH ; PURGE PRE-FETCH QUEUE WITH FAR JUMP
1292 0476 047A R + DW OFFSET VMODE ; - TO OFFSET
1293 0476 0038 + DW BIO_CS ; - IN SEGMENT -PROTECTED MODE SELECTOR
1294
1295 047A VMODE:
1296 ;-----
1297 ; SETUP USER SEGMENT REGISTERS ;
1298 ;-----
1299 047A B8 0018 MOV AX,USER_DS ; SETUP USER'S DATA SEGMENT
1300 047D 8E D8 MOV DS,AX ; TO PROTECTED MODE SELECTORS
1301 047F B8 0020 MOV AX,USER_ES ; SETUP USER'S EXTRA SEGMENT
1302 0482 8E C0 MOV ES,AX
1303 0484 B8 0028 MOV AX,USER_SS ; SETUP USER'S STACK SEGMENT
1304 0487 8E D0 MOV SS,AX
1305
1306 ;-----
1307 ; PUT TRANSFER ADDRESS ON STACK ;
1308 ; AND RETURN TO THE USER ;
1309 0489 5B POP BX ; GET RETURN IP FROM THE STACK
1310 048A 83 C4 04 ADD SP,4 ; NORMALIZE STACK POINTER
1311 048D 6A 30 PUSH USER_CS ; SET STACK FOR A RETURN FAR
1312 048F 53 PUSH BX
1313 0490 CB RET ; RETURN TO USER IN VIRTUAL MODE
1314
1315 0491 X_VIRTUAL ENDP
1316
1317 ;--- DEVICE BUSY AND INTERRUPT COMPLETE ---
1318 ;
1319 ; THIS ROUTINE IS A TEMPORARY HANDLER FOR DEVICE BUSY ;
1320 ; AND INTERRUPT COMPLETE ;
1321 ;
1322 ; INPUT - SEE PROLOGUE ;
1323 ;-----
1324
1325 0491 DEVICE_BUSY PROC NEAR
1326 0491 FB CLC ; TURN CARRY OFF
1327 0492 E9 0055 R JMP C1,F ; RETURN WITH CARRY FLAG
1328 0495 DEVICE_BUSY ENDP
1329
1330 0495 INT_COMPLETE PROC NEAR
1331 0495 CF IRET ; RETURN
1332 0496 INT_COMPLETE ENDP
1333
1334 0496 CODE ENDS
1335 END
    
```

```

1 PAGE 118,121
2 TITLE BIOS2 ---- 06/10/85 BIOS INTERRUPT ROUTINES
3 .286C
4 .LIST
5 0000 CODE SEGMENT BYTE PUBLIC
6
7 PUBLIC PRINT_SCREEN_1
8 PUBLIC RTC_INT
9 PUBLIC TIME_OF_DAY_1
10 PUBLIC TIMER_INT_1
11
12 EXTRN CMOS_READ:NEAR ; READ CMOS LOCATION ROUTINE
13 EXTRN CMOS_WRITE:NEAR ; WRITE CMOS LOCATION ROUTINE
14 EXTRN DDS:NEAR ; LOAD (DS) WITH DATA SEGMENT SELECTOR
15
16
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 ; 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 0FTH 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 TIME_OF_DAY_1 PROC FAR
82 0000 FB 08 ; INTERRUPTS BACK ON
83 0004 F5 ; CHECK IF COMMAND IN VALID RANGE (0-7)
84 0006 72 17 ; COMPLEMENT CARRY FOR ERROR EXIT
85 ; EXIT WITH CARRY = 1 IF NOT VALID
86
87 0007 1E ; SAVE USERS (DS) SEGMENT
88 0008 E8 0000 E ; GET DATA SEGMENT SELECTOR
89 0009 56 ; SAVE WORK REGISTER
90 000C C1 E8 08 ; CONVERT FUNCTION TO BYTE OFFSET
91 000F 03 C0 ; CONVERT FUNCTION TO WORD OFFSET (CY=0)
92 0011 8B F0 ; PLACE INTO ADDRESSING REGISTER
93 0013 FA ; NO INTERRUPTS DURING TIME FUNCTIONS
94 0014 2E: FF 94 0021 R ; VECTOR TO FUNCTION REQUESTED WITH CY=0
95 ; RETURN WITH CARRY FLAG SET FOR RESULT
96
97 0019 FB ; INTERRUPTS BACK ON
98 001A B4 00 ; CLEAR (AH) TO ZERO
99 001C 5E ; RECOVER USERS REGISTER
00 001D 1F ; RECOVER USERS SEGMENT SELECTOR
01 001E ; RETURN WITH CY= 0 IF NO ERROR
02
03 TIME_9:
04 RET 2
05
06
07
08
09
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
0001
0002
0003
0004
0005
0006
0007
0008
0009
000A
000B
000C
000D
000E
000F
0010
0011
0012
0013
0014
0015
0016
0017
0018
0019
001A
001B
001C
001D
001E
001F
0020
0021
0022
0023
0024
0025
0026
0027
0028
0029
002A
002B
002C
002D
002E
002F
0030
0031
0032
0033
0034
0035
0036
0037
0038
0039
003A
003B
003C
003D
003E
003F
0040
0041
0042
0043
0044
0045
0046
0047
0048
0049
004A
004B
004C
004D
004E
004F
0050
0051
0052
0053
0054
0055
0056
0057
0058
0059
005A
005B
005C
005D
005E
005F
0060
0061
0062
0063
0064
0065
0066
0067
0068
0069
006A
006B
006C
006D
006E
006F
0070
0071
0072
0073
0074
0075
0076
0077
0078
0079
007A
007B
007C
007D
007E
007F
0080
0081
0082
0083
0084
0085
0086
0087
0088
0089
008A
008B
008C
008D
008E
008F
0090
0091
0092
0093
0094
0095
0096
0097
0098
0099
009A
009B
009C
009D
009E
009F
00A0
00A1
00A2
00A3
00A4
00A5
00A6
00A7
00A8
00A9
00AA
00AB
00AC
00AD
00AE
00AF
00B0
00B1
00B2
00B3
00B4
00B5
00B6
00B7
00B8
00B9
00BA
00BB
00BC
00BD
00BE
00BF
00C0
00C1
00C2
00C3
00C4
00C5
00C6
00C7
00C8
00C9
00CA
00CB
00CC
00CD
00CE
00CF
00D0
00D1
00D2
00D3
00D4
00D5
00D6
00D7
00D8
00D9
00DA
00DB
00DC
00DD
00DE
00DF
00E0
00E1
00E2
00E3
00E4
00E5
00E6
00E7
00E8
00E9
00EA
00EB
00EC
00ED
00EE
00EF
00F0
00F1
00F2
00F3
00F4
00F5
00F6
00F7
00F8
00F9
00FA
00FB
00FC
00FD
00FE
00FF
0000
0001
0002
0003
0004
0005
0006
0007
0008
0009
000A
000B
000C
000D
000E
000F
0010
0011
0012
0013
0014
0015
0016
0017
0018
0019
001A
001B
001C
001D
001E
001F
0020
0021
0022
0023
0024
0025
0026
0027
0028
0029
002A
002B
002C
002D
002E
002F
0030
0031
0032
0033
0034
0035
0036
0037
0038
0039
003A
003B
003C
003D
003E
003F
0040
0041
0042
0043
0044
0045
0046
0047
0048
0049
004A
004B
004C
004D
004E
004F
0050
0051
0052
0053
0054
0055
0056
0057
0058
0059
005A
005B
005C
005D
005E
005F
0060
0061
0062
0063
0064
0065
0066
0067
0068
0069
006A
006B
006C
006D
006E
006F
0070
0071
0072
0073
0074
0075
0076
0077
0078
0079
007A
007B
007C
007D
007E
007F
0080
0081
0082
0083
0084
0085
0086
0087
0088
0089
008A
008B
008C
008D
008E
008F
0090
0091
0092
0093
0094
0095
0096
0097
0098
0099
009A
009B
009C
009D
009E
009F
00A0
00A1
00A2
00A3
00A4
00A5
00A6
00A7
00A8
00A9
00AA
00AB
00AC
00AD
00AE
00AF
00B0
00B1
00B2
00B3
00B4
00B5
00B6
00B7
00B8
00B9
00BA
00BB
00BC
00BD
00BE
00BF
00C0
00C1
00C2
00C3
00C4
00C5
00C6
00C7
00C8
00C9
00CA
00CB
00CC
00CD
00CE
00CF
00D0
00D1
00D2
00D3
00D4
00D5
00D6
00D7
00D8
00D9
00DA
00DB
00DC
00DD
00DE
00DF
00E0
00E1
00E2
00E3
00E4
00E5
00E6
00E7
00E8
00E9
00EA
00EB
00EC
00ED
00EE
00EF
00F0
00F1
00F2
00F3
00F4
00F5
00F6
00F7
00F8
00F9
00FA
00FB
00FC
00FD
00FE
00FF
0000
0001
0002
0003
0004
0005
0006
0007
0008
0009
000A
000B
000C
000D
000E
000F
0010
0011
0012
0013
0014
0015
0016
0017
0018
0019
001A
001B
001C
001D
001E
001F
0020
0021
0022
0023
0024
0025
0026
0027
0028
0029
002A
002B
002C
002D
002E
002F
0030
0031
0032
0033
0034
0035
0036
0037
0038
0039
003A
003B
003C
003D
003E
003F
0040
0041
0042
0043
0044
0045
0046
0047
0048
0049
004A
004B
004C
004D
004E
004F
0050
0051
0052
0053
0054
0055
0056
0057
0058
0059
005A
005B
005C
005D
005E
005F
0060
0061
0062
0063
0064
0065
0066
0067
0068
0069
006A
006B
006C
006D
006E
006F
0070
0071
0072
0073
0074
0075
0076
0077
0078
0079
007A
007B
007C
007D
007E
007F
0080
0081
0082
0083
0084
0085
0086
0087
0088
0089
008A
008B
008C
008D
008E
008F
0090
0091
0092
0093
0094
0095
0096
0097
0098
0099
009A
009B
009C
009D
009E
009F
00A0
00A1
00A2
00A3
00A4
00A5
00A6
00A7
00A8
00A9
00AA
00AB
00AC
00AD
00AE
00AF
00B0
00B1
00B2
00B3
00B4
00B5
00B6
00B7
00B8
00B9
00BA
00BB
00BC
00BD
00BE
00BF
00C0
00C1
00C2
00C3
00C4
00C5
00C6
00C7
00C8
00C9
00CA
00CB
00CC
00CD
00CE
00CF
00D0
00D1
00D2
00D3
00D4
00D5
00D6
00D7
00D8
00D9
00DA
00DB
00DC
00DD
00DE
00DF
00E0
00E1
00E2
00E3
00E4
00E5
00E6
00E7
00E8
00E9
00EA
00EB
00EC
00ED
00EE
00EF
00F0
00F1
00F2
00F3
00F4
00F5
00F6
00F7
00F8
00F9
00FA
00FB
00FC
00FD
00FE
00FF
0000
0001
0002
0003
0004
0005
0006
0007
0008
0009
000A
000B
000C
000D
000E
000F
0010
0011
0012
0013
0014
0015
0016
0017
0018
0019
001A
001B
001C
001D
001E
001F
0020
0021
0022
0023
0024
0025
0026
0027
0028
0029
002A
002B
002C
002D
002E
002F
0030
0031
0032
0033
0034
0035
0036
0037
0038
0039
003A
003B
003C
003D
003E
003F
0040
0041
0042
0043
0044
0045
0046
0047
0048
0049
004A
004B
004C
004D
004E
004F
0050
0051
0052
0053
0054
0055
0056
0057
0058
0059
005A
005B
005C
005D
005E
005F
0060
0061
0062
0063
0064
0065
0066
0067
0068
0069
006A
006B
006C
006D
006E
006F
0070
0071
0072
0073
0074
0075
0076
0077
0078
0079
007A
007B
007C
007D
007E
007F
0080
0081
0082
0083
0084
0085
0086
0087
0088
0089
008A
008B
008C
008D
008E
008F
0090
0091
0092
0093
0094
0095
0096
0097
0098
0099
009A
009B
009C
009D
009E
009F
00A0
00A1
00A2
00A3
00A4
00A5
00A6
00A7
00A8
00A9
00AA
00AB
00AC
00AD
00AE
00AF
00B0
00B1
00B2
00B3
00B4
00B5
00B6
00B7
00B8
00B9
00BA
00BB
00BC
00BD
00BE
00BF
00C0
00C1
00C2
00C3
00C4
00C5
00C6
00C7
00C8
00C9
00CA
00CB
00CC
00CD
00CE
00CF
00D0
00D1
00D2
00D3
00D4
00D5
00D6
00D7
00D8
00D9
00DA
00DB
00DC
00DD
00DE
00DF
00E0
00E1
00E2
00E3
00E4
00E5
00E6
00E7
00E8
00E9
00EA
00EB
00EC
00ED
00EE
00EF
00F0
00F1
00F2
00F3
00F4
00F5
00F6
00F7
00F8
00F9
00FA
00FB
00FC
00FD
00FE
00FF
0000
0001
0002
0003
0004
0005
0006
0007
0008
0009
000A
000B
000C
000D
000E
000F
0010
0011
0012
0013
0014
0015
0016
0017
0018
0019
001A
001B
001C
001D
001E
001F
0020
0021
0022
0023
0024
0025
0026
0027
0028
0029
002A
002B
002C
002D
002E
002F
0030
0031
0032
0033
0034
0035
0036
0037
0038
0039
003A
003B
003C
003D
003E
003F
0040
0041
0042
0043
0044
0045
0046
0047
0048
0049
004A
004B
004C
004D
004E
004F
0050
0051
0052
0053
0054
0055
0056
0057
0058
0059
005A
005B
005C
005D
005E
005F
0060
0061
0062
0063
0064
0065
0066
0067
0068
0069
006A
006B
006C
006D
006E
006F
0070
0071
0072
0073
0074
0075
0076
0077
0078
0079
007A
007B
007C
007D
007E
007F
0080
0081
0082
0083
0084
0085
0086
0087
0088
0089
008A
008B
008C
008D
008E
008F
0090
0091
0092
0093
0094
0095
0096
0097
0098
0099
009A
009B
009C
009D
009E
009F
00A0
00A1
00A2
00A3
00A4
00A5
00A6
00A7
00A8
00A9
00AA
00AB
00AC
00AD
00AE
00AF
00B0
00B1
00B2
00B3
00B4
00B5
00B6
00B7
00B8
00B9
00BA
00BB
00BC
00BD
00BE
00BF
00C0
00C1
00C2
00C3
00C4
00C5
00C6
00C7
00C8
00C9
00CA
00CB
00CC
00CD
00CE
00CF
00D0
00D1
00D2
00D3
00D4
00D5
00D6
00D7
00D8
00D9
00DA
00DB
00DC
00DD
00DE
00DF
00E0
00E1
00E2
00E3
00E4
00E5
00E6
00E7
00E8
00E9
00EA
00EB
00EC
00ED
00EE
00EF
00F0
00F1
00F2
00F3
00F4
00F5
00F6
00F7
00F8
00F9
00FA
00FB
00FC
00FD
00FE
00FF
0000
0001
0002
0003
0004
0005
0006
0007
0008
0009
000A
000B
000C
000D
000E
000F
0010
0011
0012
0013
0014
0015
0016
0017
0018
0019
001A
001B
001C
001D
001E
001F
0020
0021
0022
0023
0024
0025
0026
0027
0028
0029
002A
002B
002C
002D
002E
002F
0030
0031
0032
0033
0034
0035
0036
0037
0038
0039
003A
003B
003C
003D
003E
003F
0040
0041
0042
0043
0044
0045
0046
0047
0048
0049
004A
004B
004C
004D
004E
004F
0050
0051
0052
0053
0054
0055
0056
0057
0058
0059
005A
005B
005C
005D
005E
005F
0060
0061
0062
0063
0064
0065
0066
0067
0068
0069
006A
006B
006C
006D
006E
006F
0070
0071
0072
0073
0074
0075
0076
0077
0078
0079
007A
007B
007C
007D
007E
007F
0080
0081
0082
0083
0084
0085
0086
0087
0088
0089
008A
008B
008C
008D
008E
008F
0090
0091
0092
0093
0094
0095
0096
0097
0098
0099
009A
009B
009C
009D
009E
009F
00A0
00A1
00A2
00A3
00A4
00A5
00A6
00A7
00A8
00A9
00AA
00AB
00AC
00AD
00AE
00AF
00B0
00B1
00B2
00B3
00B4
00B5
00B6
00B7
00B8
00B9
00BA
00BB
00BC
00BD
00BE
00BF
00C0
00C1
00C2
00C3
00C4
00C5
00C6
00C7
00C8
00C9
00CA
00CB
00CC
00CD
00CE
00CF
00D0
00D1
00D2
00D3
00D4
00D5
00D6
00D7
00D8
00D9
00DA
00DB
00DC
00DD
00DE
00DF
00E0
00E1
00E2
00E3
00E4
00E5
00E6
00E7
00E8
00E9
00EA
00EB
00EC
00ED
00EE
00EF
00F0
00F1
00F2
00F3
00F4
00F5
00F6
00F7
00F8
00F9
00FA
00FB
00FC
00FD
00FE
00FF
0000
0001
0002
0003
0004
0005
0006
0007
0008
0009
000A
000B
000C
000D
000E
000F
0010
0011
0012
0013
0014
0015
0016
0017
0018
0019
001A
001B
001C
001D
001E
001F
0020
0021
0022
0023
0024
0025
0026
0027
0028
0029
002A
002B
002C
002D
002E
002F
0030
0031
0032
0033
0034
0035
0036
0037
0038
0039
003A
003B
003C
003D
003E
003F
0040
0041
0042
0043
0044
0045
0046
0047
0048
0049
004A
004B
004C
004D
004E
004F
0050
0051
0052
0053
0054
0055
0056
0057
0058
0059
005A
005B
005C
005D
005E
005F
0060
0061
0062
0063
0064
0065
0066
0067
0068
0069
006A
006B
006C
006D
006E
006F
0070
0071
0072
0073
0074
0075
0076
0077
0078
0079
007A
007B
007C
007D
007E
007F
0080
0081
0082
0083
0084
0085
0086
0087
0088
0089
008A
008B
008C
008D
008E
008F
0090
0091
0092
0093
0094
0095
0096
0097
0098
0099
009A
009B
009C
009D
009E
009F
00A0
00A1
00A2
00A3
00A4
00A5
00A6
00A7
00A8
00A9
00AA
00AB
00AC
00AD
00AE
00AF
00B0
00B1
00B2
00B3
00B4
00B5
00B6
00B7
00B8
00B9
00BA
00BB
00BC
00BD
00BE
00BF
00C0
00C1
00C2
00C3
00C4
00C5
00C6
00C7
00C8
00C9
00CA
00CB
00CC
00CD
00CE
00CF
00D0
00D1
00D2
00D3
00D4
00D5
00D6
00D7
00D8
00D9
00DA
00DB
00DC
00DD
00DE
00DF
00E0
00E1
00E2
00E3
00E4
00E5
00E6
00E7
00E8
00E9
00EA
00EB
00EC
00ED
00EE
00EF
00F0
00F1
00F2
00F3
00F4
00F5
00F6
00F7
00F8
00F9
00FA
00FB
00FC
00FD
00FE
00FF
0000
0001
0002
0003
0004
0005
0006
0007
0008
0009
000A
000B
000C
000D
000E
000F
0010
0011
0012
0013
0014
0015
0016
0017
0018
0019
001A
001B
001C
001D
001E
001F
0020
0021
0022
0023
0024
0025
0026
0027
0028
0029
002A
002B
002C
002D
002E
002F
0030
0031
0032
0033
0034
0035
0036
0037
0038
0039
003A
003B
003C
003D
003E
003F
0040
0041
0042
0043
0044
0045
0046
0047
0048
0049
004A
004B
004C
004D
004E
004F
0050
0051
0052
0053
0054
0055
0056
0057
0058
0059
005A
005B
005C
005D
005E
005F
0060
0061
0062
0063
0064
0065
0066
0067
0068
0069
006A
006B
006C
006D
006E
006F
0070
0071
0072
0073
0074
0075
0076
0077
0078
0079
007A
007B
007C
007D
007E
007F
0080
0081
0082
0083
0084
0085
0086
0087
0088
0089
008A
008B
008C
008D
008E
008F
0090
0091
0092
0093
0094
0095
0096
0097
0098
0099
009A
009B
009C
009D
009E
009F
00A0
00A1
00A2
00A3
00A4
00A5
00A6
00A7
00A8
00A9
00AA
00AB
00AC
00AD
00AE
00AF
00B0
00B1
00B2
00B3
00B4
00B5
00B6
00B7
00B8
00B9
00BA
00BB
00BC
00BD
00BE
00BF
00C0
00C1
00C2
00C3
00C4
00C5
00C6
00C7
00C8
00C9
00CA
00CB
00CC
00CD
00CE
00CF
00D0
00D1
00D2
00D3
00D4
00D5
00D6
00D7
00D8
00D9
00DA
00DB
00DC
00DD
00DE
00DF
00E0
00E1
00E2
00E3
00E4
00E5
00E6
00E7
00E8
00E9
00EA
00EB
00EC
00ED
00EE
00EF
00F0
00F1
00F2
00F3
00F4
00F5
00F6
00F7
00F8
00F9
00FA
00FB
00FC
00FD
00FE
00FF
0000
0001
0002
0003
0004
0005
0006
0007
0008
0009
000A
000B
000C
000D
000E
000F
0010
0011
0012
0013
0014
0015
0016
0017
0018
0019
001A
001B
001C
001D
001E
001F
0020
0021
0022
0023
0024
0025
0026
0027
0028
0029
002A
002B
002C
002D
002E
002F
0030
0031
0032
0033
0034
0035
0036
0037
0038
0039
003A
003B
003C
003D
003E
003F
0040
0041
0042
0043
0044
0045
0046
0047
004
```

```

114                                     PAGE
115 0031 A0 0070 R                       RTC_00 PROC    NEAR
116 0034 C6 06 0070 R 00                 MOV     AL,®TIMER_OFL
117 0039 BB 0E 006E R 00                 MOV     ®TIMER_OFL,0
118 003D BB 16 006C R 00                 MOV     CX,®TIMER_HIGH
119 0041 C3                               MOV     DX,®TIMER_LOW
120 0041 C3                               RET
121
122 0042                                     RTC_10:
123 0042 89 16 006C R                     MOV     ®TIMER_LOW,DX
124 0046 89 0E 006E R                     MOV     ®TIMER_HIGH,CX
125 004A C6 06 0070 R 00                 MOV     ®TIMER_OFL,0
126 004F C3                               RET
127
128 0050                                     RTC_20:
129 0050 E8 016B R                         CALL    UPD_IPR
130 0053 72 1F                             JC      RTC_29
131
132 0055 B0 00                             MOV     AL,CMOS_SECONDS
133 0057 EB 0000 E                         CALL    CMOS_READ
134 005A 8A F0                             MOV     DH,AL
135 005C B0 08                             MOV     AL,CMOS_REG_B
136 005E E8 0000 E                         CALL    CMOS_READ
137 0061 24 01                             AND     AL,0000001B
138 0063 8A D0                             MOV     DL,AL
139 0065 B0 02                             MOV     AL,CMOS_MINUTES
140 0067 EB 0000 E                         CALL    CMOS_READ
141 006A 8A C8                             MOV     CL,AL
142 006C B0 04                             MOV     AL,CMOS_HOURS
143 006E E8 0000 E                         CALL    CMOS_READ
144 0071 8A E8                             MOV     CH,AL
145 0073 F8                             CLC
146 0074                                     RTC_29:
147 0074 C3                               RET
148
149 0075                                     RTC_30:
150 0075 E8 016B R                         CALL    UPD_IPR
151 0078 73 03                             JNC    RTC_35
152 007A E8 0154 R                         CALL    RTC_STA
153
154 007D 8A E6                             MOV     AH,DH
155 007F B0 00                             MOV     AL,CMOS_SECONDS
156 0081 EB 0000 E                         CALL    CMOS_WRITE
157 0084 8A E1                             MOV     AH,CL
158 0086 B0 02                             MOV     AL,CMOS_MINUTES
159 0088 EB 0000 E                         CALL    CMOS_WRITE
160 008B 8A C8                             MOV     AH,CH
161 008D B0 04                             MOV     AL,CMOS_HOURS
162 008F E8 0000 E                         CALL    CMOS_WRITE
163 0092 BB 080B E                         MOV     AX,®CMOS_REG_B
164 0095 E8 0000 E                         CALL    CMOS_READ
165 0098 24 62                             AND     AL,0100010B
166 009A 0C 02                             OR      AL,00000010B
167 009C B0 E2 01                         AND     DL,00000001B
168 009F 0A C2                             OR      AL,DL
169 00A1 86 E0                             XCHG   AH,AL
170 00A3 E8 0000 E                         CALL    CMOS_WRITE
171 00A6 F8                             CLC
172 00A7 C3                               RET
173
174 00A8                                     RTC_40:
175 00A8 E8 016B R                         CALL    UPD_IPR
176 00AB 72 1D                             JC      RTC_49
177
178 00AD B0 07                             MOV     AL,CMOS_DAY_MONTH
179 00AF EB 0000 E                         CALL    CMOS_READ
180 00B2 8A D0                             MOV     DL,AL
181 00B4 B0 08                             MOV     AL,CMOS_MONTH
182 00B6 EB 0000 E                         CALL    CMOS_READ
183 00B9 8A F0                             MOV     DH,AL
184 00BB B0 09                             MOV     AL,CMOS_YEAR
185 00BD EB 0000 E                         CALL    CMOS_READ
186 00C0 8A C8                             MOV     CL,AL
187 00C2 B0 32                             MOV     AL,CMOS_CENTURY
188 00C4 E8 0000 E                         CALL    CMOS_READ
189 00C7 8A E8                             MOV     CH,AL
190 00C9 F8                             CLC
191 00CA                                     RTC_49:
192 00CA C3                               RET
193
194 00CB                                     RTC_50:
195 00CB E8 016B R                         CALL    UPD_IPR
196 00CE 73 03                             JNC    RTC_55
197 00D0 E8 0154 R                         CALL    RTC_STA
198
199 00D3 BB 0006 E                         MOV     AX,CMOS_DAY_WEEK
200 00D6 EB 0000 E                         CALL    CMOS_WRITE
201 00D9 8A E2                             MOV     AH,DL
202 00DB B0 07                             MOV     AL,CMOS_DAY_MONTH
203 00DD EB 0000 E                         CALL    CMOS_WRITE
204 00E0 8A E6                             MOV     AH,DH
205 00E2 B0 08                             MOV     AL,CMOS_MONTH
206 00E4 EB 0000 E                         CALL    CMOS_WRITE
207 00E7 8A E1                             MOV     AH,CL
208 00E9 B0 09                             MOV     AL,CMOS_YEAR
209 00EB EB 0000 E                         CALL    CMOS_WRITE
210 00EE 8A E5                             MOV     AH,CH
211 00F0 B0 32                             MOV     AL,CMOS_CENTURY
212 00F2 EB 0000 E                         CALL    CMOS_WRITE
213 00F5 BB 080B E                         MOV     AX,®CMOS_REG_B
214 00F8 EB 0000 E                         CALL    CMOS_READ
215 00FB 24 7F                             AND     AL,07FH
216 00FD B6 E0                             XCHG   AH,AL
217 00FF E8 0000 E                         CALL    CMOS_WRITE
218 0102 F8                             CLC
219 0103 C3                               RET

```

```

220 PAGE
221 RTC_60:
222 0104 MOV AL,CMOS_REG_B ; SET RTC ALARM
223 0106 EB 0B CALL CMOS_READ ; ADDRESS ALARM
224 0109 A8 20 TEST AL,20H ; READ ALARM REGISTER
225 010B F9 STC ; CHECK FOR ALARM ALREADY ENABLED
226 010C 75 33 JNZ RTC_69 ; SET CARRY IN CASE OF ERROR
227 ; ERROR EXIT IF ALARM SET
228 010E EB 016B R CALL UPD_IPR ; CHECK FOR UPDATE IN PROCESS
229 0111 73 03 JNC RTC_65 ; SKIP INITIALIZATION IF NO ERROR
230 0113 EB 0154 R CALL RTC_STA ; ELSE INITIALIZE CLOCK
231 0116
232 0116 8A E6 MOV AH,DH ; GET SECONDS BYTE
233 0118 B0 01 MOV AL,CMOS_SEC_ALARM ; ADDRESS THE SECONDS ALARM REGISTER
234 011A EB 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 EB 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 EB 0000 E CALL CMOS_WRITE ; INSERT HOURS
241 012B E4 A1 IN AL,INTB01 ; READ SECOND INTERRUPT MASK REGISTER
242 012D 24 FE AND AL,0FEH ; ENABLE ALARM TIMER BIT (CY= 0)
243 012F E6 A1 OUT INTB01,AL ; WRITE UPDATED MASK
244 0131 B8 0B0B MOV AX,X*CMOS_REG_B ; ADDRESS ALARM REGISTER
245 0134 EB 0000 E CALL CMOS_READ ; READ CURRENT ALARM REGISTER
246 0137 24 7F AND AL,07FH ; ENSURE SET BIT TURNED OFF
247 0139 0C 20 OR AL,20H ; TURN ON ALARM ENABLE
248 013B 86 E0 XCHG AH,AL ; MOVE MASK TO OUTPUT REGISTER
249 013D EB 0000 E CALL CMOS_WRITE ; WRITE NEW ALARM MASK
250 0140 F8 CLC ; SET CY= 0
251 0141
252 0141 B8 0000 RTC_69: MOV AX,0 ; CLEAR AX REGISTER
253 0144 C3 RET ; RETURN WITH RESULTS IN CARRY FLAG
254
255 0148 RTC_70: ; RESET ALARM
256 0148 B8 0B0B MOV AX,X*CMOS_REG_B ; ADDRESS ALARM REGISTER (TO BOTH AH,AL)
257 014B EB 0000 E CALL CMOS_READ ; READ ALARM REGISTER
258 014B 24 57 AND AL,57H ; TURN OFF ALARM ENABLE
259 014D 86 E0 XCHG AH,AL ; SAVE DATA AND RECOVER ADDRESS
260 014F EB 0000 E CALL CMOS_WRITE ; RESTORE NEW VALUE
261 0152 F8 CLC ; SET CY= 0
262 0153 C3 RET ; RETURN WITH NO CARRY
263
264 0154 RTC_00 ENDP
265
266 0154 RTC_STA PROC NEAR ; INITIALIZE REAL TIME CLOCK
267 0154 B8 260A AX,26H*H+CMOS_REG_A ; ADDRESS REGISTER A AND LOAD DATA MASK
268 0157 EB 0000 E CALL CMOS_WRITE ; INITIALIZE STATUS REGISTER A
269 015A B8 820B AX,82H*H+CMOS_REG_B ; SET *SET BIT* FOR CLOCK INITIALIZATION
270 015D EB 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 EB 0000 E CALL CMOS_READ ; READ REGISTER C TO INITIALIZE
273 0165 B0 0D MOV AL,CMOS_REG_D ; ADDRESS REGISTER D
274 0167 EB 0000 E CALL CMOS_READ ; READ REGISTER D TO INITIALIZE
275 016A C3 RET
276
277 016B RTC_STA ENDP
278
279 016B UPD_IPR PROC NEAR ; WAIT TILL UPDATE NOT IN PROGRESS
280 016B 51 PUSH CX ; SAVE CALLERS REGISTER
281 016C B9 0320 MOV CX,800 ; SET TIMEOUT LOOP COUNT
282 016F
283 016F B0 0A UPD_10: MOV AL,CMOS_REG_A ; ADDRESS STATUS REGISTER A
284 0171 FA CLI ; NO TIMER INTERRUPTS DURING UPDATES
285 0172 EB 0000 E CALL CMOS_READ ; READ UPDATE IN PROCESS FLAG
286 0175 A8 80 TEST AL,80H ; IF UIP BIT IS ON ( CANNOT READ TIME )
287 0177 74 06 JZ UPD_90 ; EXIT WITH CY= 0 IF CAN READ CLOCK NOW
288 0179 F9 STI ; ALLOW INTERRUPTS WHILE WAITING
289 017A E2 F3 LOOP UPD_10 ; LOOP TILL READY OR TIMEOUT
290 017C 33 C0 XOR AX,AX ; CLEAR RESULTS IF ERROR
291 017E F9 STC ; SET CARRY FOR ERROR
292 017F
293 017F 59 UPD_90: POP CX ; RESTORE CALLERS REGISTER
294 0180 FA CLI ; INTERRUPTS OFF DURING SET
295 0181 C3 RET ; RETURN WITH CY FLAG SET
296
297 0182 UPD_IPR ENDP
    
```

SECTION 5

```

298 PAGE
299 ;--- HARDWARE INT TO H-- ( IRQ LEVEL 8 ) -----
300 ; ALARM INTERRUPT HANDLER (RTC)
301 ; THIS ROUTINE HANDLES THE PERIODIC AND ALARM INTERRUPTS FROM THE CMOS
302 ; TIMER. INPUT FREQUENCY IS 1.024 KHZ OR APPROXIMATELY 1024 INTERRUPTS
303 ; EVERY SECOND FOR THE PERIODIC INTERRUPT. FOR THE ALARM FUNCTION,
304 ; THE INTERRUPT WILL OCCUR AT THE DESIGNATED TIME.
305 ;
306 ; INTERRUPTS ARE ENABLED WHEN THE EVENT OR ALARM FUNCTION IS ACTIVATED.
307 ; FOR THE EVENT INTERRUPT, THE HANDLER WILL DECREMENT THE WAIT COUNTER
308 ; AND WHEN IT EXPIRES WILL SET THE DESIGNATED LOCATION TO 80H. FOR
309 ; THE ALARM INTERRUPT, THE USER MUST PROVIDE A ROUTINE TO INTERCEPT
310 ; THE CORRECT ADDRESS FROM THE VECTOR TABLE INVOKED BY INTERRUPT 4AH
311 ; PRIOR TO SETTING THE REAL TIME CLOCK ALARM (INT 1AH, AH= 06H).
312 ;-----
313
314 0182 RTC_INT PROC FAR ; ALARM INTERRUPT
315 0182 1E PUSH DS ; LEAVE INTERRUPTS DISABLED
316 0183 90 PUSH AX ; SAVE REGISTERS
317 0184 57 PUSH DI
318
319 0185 RTC_I_1 ; CHECK FOR SECOND INTERRUPT
320 0185 88 88BC MOV AX, (CMOS_REG_B+NMI)*H+CMOS_REG_C+NMI ; ALARM AND STATUS
321 0188 E6 70 OUT CMOS_PORT,AL ; WRITE ALARM FLAG MASK ADDRESS
322 018A 90 NOP ; I/O DELAY
323 018B E4 71 IN AL,CMOS_DATA ; READ AND RESET INTERRUPT REQUEST FLAGS
324 018D A8 60 TEST AL,0100000B ; CHECK FOR EITHER INTERRUPT PENDING
325 018F 74 5C JZ RTC_I_9 ; EXIT IF NOT A VALID RTC INTERRUPT
326
327 0191 86 E0 XCHG AH,AL ; SAVE FLAGS AND GET ENABLE ADDRESS
328 0193 E6 70 OUT CMOS_PORT,AL ; WRITE ALARM ENABLE MASK ADDRESS
329 0195 90 NOP ; I/O DELAY
330 0196 E4 71 IN AL,CMOS_DATA ; READ CURRENT ALARM ENABLE MASK
331 0198 22 C4 AND AL,AH ; ALLOW ONLY SOURCES THAT ARE ENABLED
332 019A A8 40 TEST AL,0100000B ; CHECK FOR PERIODIC INTERRUPT
333 019C 74 3F JZ RTC_I_5 ; SKIP IF NOT A PERIODIC INTERRUPT
334
335 ;----- DECREMENT WAIT COUNT BY INTERRUPT INTERVAL
336
337 019E E8 0000 E CALL DDS ; ESTABLISH DATA SEGMENT ADDRESSABILITY
338 01A1 81 2E 009C R 03DD SUB @RTC_LOW,0976 ; DECREMENT COUNT LOW BY 1/1024
339 01A7 83 1E 009E R 00 SBB @RTC_HIGH,0 ; ADJUST HIGH WORD FOR LOW WORD BORROW
340 01AC 73 2F JNC RTC_I_5 ; SKIP TILL 32 BIT WORD LESS THAN ZERO
341
342 ;----- TURN OFF PERIODIC INTERRUPT ENABLE
343
344 01AE 50 PUSH AX ; SAVE INTERRUPT FLAG MASK
345 01AF 88 88BB MOV AX,*(CMOS_REG_B+NMI) ; INTERRUPT ENABLE REGISTER
346 01B2 E6 70 OUT CMOS_PORT,AL ; WRITE ADDRESS TO CMOS CLOCK
347 01B4 90 NOP ; I/O DELAY
348 01B5 E4 71 IN AL,CMOS_DATA ; READ CURRENT ENABLER
349 01B7 24 8F AND AL,0BFH ; TURN OFF PIE
350 01B9 86 C4 XCHG AL,AH ; GET CMOS ADDRESS AND SAVE VALUE
351 01BB E6 70 OUT CMOS_PORT,AL ; ADDRESS REGISTER B
352 01BD 86 C4 XCHG AL,AH ; GET NEW INTERRUPT ENABLE MASK
353 01BF E6 71 OUT CMOS_DATA,AL ; SET MASK IN INTERRUPT ENABLE REGISTER
354 01C1 88 POP AX ; GET INTERRUPT SOURCE BACK
355 01C2 F6 06 00A0 R 02 TEST @RTC_WAIT_FLAG,02H ; CHECK FOR "WAIT" FUNCTION ACTIVE
356 01C7 C6 06 00A0 R 00 MOV @RTC_WAIT_FLAG,0 ; SET FUNCTION ACTIVE FLAGS OFF
357 01CC 05 3E 0098 R LDS DI,WORD PTR @USER_FLAG ; SET UP (DS:DI) TO POINT TO USER FLAG
358 01D0 C6 05 80 MOV BYTE PTR [DI],80H ; TURN ON USERS POSTED FLAG
359 01D3 74 08 JZ RTC_I_5 ; SKIP IF "EVENT_WAIT" FUNCTION
360
361 01D5 E8 0000 E CALL DDS ; ESTABLISH DATA SEGMENT ADDRESSABILITY
362 01D8 C6 06 00A0 R 83 MOV @RTC_WAIT_FLAG,083H ; AND SET "WAIT" BACK TO BUSY & POSTED
363 01DD JZ
364 01DD A8 20 TEST AL,00100000B ; TEST FOR ALARM INTERRUPT
365 01DF 74 0A JZ RTC_I_7 ; SKIP USER INTERRUPT CALL IF NOT ALARM
366
367 01E1 80 0F MOV AL,CMOS_SHUT_DOWN ; POINT TO DEFAULT READ ONLY REGISTER
368 01E3 E6 70 OUT CMOS_PORT,AL ; ENABLE NMI AND CMOS ADDRESS TO DEFAULT
369 01E5 FB STI ; INTERRUPTS BACK ON NOW
370 01E6 62 PUSH DX
371 01E7 CD 4A INT 4AH ; TRANSFER TO USER ROUTINE
372 01E9 5A POP DX
373 01EA FA CLI ; BLOCK INTERRUPT FOR RETRY
374 01EB JZ
375 01EB EB 98 RTC_I_7 ; RESTART ROUTINE TO HANDLE DELAYED
376 ; ENTRY AND SECOND EVENT BEFORE DONE
377
378 01ED RTC_I_9 ; EXIT - NO PENDING INTERRUPTS
379 01ED 80 0F MOV AL,CMOS_SHUT_DOWN ; POINT TO DEFAULT READ ONLY REGISTER
380 01EF E6 70 OUT CMOS_PORT,AL ; ENABLE NMI AND CMOS ADDRESS TO DEFAULT
381 01F1 90 NOP ; I/O DELAY
382 01F2 E4 71 IN AL,CMOS_DATA ; OPEN STANDBY LATCH
383 01F4 80 20 MOV AL,E01 ; END OF INTERRUPT MASK TO 8259 - 2
384 01F6 E6 A0 OUT INTB00,AL ; TO 8259 - 2
385 01F8 E4 20 OUT INTA00,AL ; TO 8259 - 2
386 01FA 5F POP DI ; RESTORE REGISTERS
387 01FB 58 POP AX
388 01FC 1F POP DS
389 01FD CF IRET ; END OF INTERRUPT
390
391 01FE RTC_INT ENDP

```

```

392 PAGE
393 ;--- INT 05H -----
394 ; PRINT_SCREEN
395 ; THIS LOGIC WILL BE INVOKED BY INTERRUPT 05H TO PRINT THE SCREEN.
396 ; THE CURSOR POSITION AT THE TIME THIS ROUTINE IS INVOKED WILL BE
397 ; SAVED AND RESTORED UPON COMPLETION. THE ROUTINE IS INTENDED TO
398 ; RUN WITH INTERRUPTS ENABLED. IF A SUBSEQUENT PRINT_SCREEN KEY
399 ; IS DEPRESSED WHILE THIS ROUTINE IS PRINTING IT WILL BE IGNORED.
400 ; THE BASE PRINTERS STATUS IS CHECKED FOR NOT BUSY AND NOT OUT OF
401 ; PAPER. AN INITIAL STATUS ERROR WILL ABEND THE PRINT REQUEST.
402 ; ADDRESS 0050:0000 CONTAINS THE STATUS OF THE PRINT_SCREEN:
403 ;
404 ; 50:0 = 0 PRINT_SCREEN HAS NOT BEEN CALLED OR UPON RETURN
405 ; FROM A CALL THIS INDICATES A SUCCESSFUL OPERATION.
406 ; = 1 PRINT_SCREEN IS IN PROGRESS IGNORE THIS REQUEST.
407 ; = 255 ERROR ENCOUNTERED DURING PRINTING.
408 ;-----
409
410 PRINT_SCREEN_1 PROC FAR ; DELAY INTERRUPT ENABLE TILL FLAG SET
411
412 01FE 1E PUSH DS ; SAVE WORK REGISTERS
413 01FF 50 PUSH AX
414 0200 53 PUSH BX
415 0201 51 PUSH CX
416 0202 52 PUSH DX ; USE 0040:0100 FOR STATUS AREA STORAGE
417 0203 E8 0000 E CALL DDS ; GET STATUS BYTE DATA SEGMENT
418 0206 80 3E 0100 R 01 CMP #STATUS_BYTE,1 ; SEE IF PRINT ALREADY IN PROGRESS
419 020B 74 74 JE PRI90 ; EXIT IF PRINT ALREADY IN PROGRESS
420 020D C6 06 0100 R 01 MOV #STATUS_BYTE,1 ; INDICATE PRINT NOW IN PROGRESS
421 0212 FB STI ; MUST RUN WITH INTERRUPTS ENABLED
422 0213 94 0F MOV AH,0FH ; WILL REQUEST THE CURRENT SCREEN MODE
423 0215 CD 10 INT 10H ; (AL)= MODE
424 ; (AH)= NUMBER COLUMNS/LINE
425 ; (BH)= VISUAL PAGE
426 ; WILL MAKE USE OF (CX) REGISTER TO
427 0217 8A CC MOV CL,AH ; CONTROL ROWS ON SCREEN & COLUMNS
428 0219 8A 2E 0084 R MOV CH,#ROWS ; ADJUST ROWS ON DISPLAY COUNT
429 021D FE C5 INC CH ; (CL)= NUMBER COLUMNS/LINE
430 ; (CH)= NUMBER OF ROWS ON DISPLAY
431
432 ;-----
433 ; AT THIS POINT WE KNOW THE COLUMNS/LINE COUNT IS IN (CL) ;
434 ; AND THE NUMBER OF ROWS ON THE DISPLAY IS IN (CH) ;
435 ; THE PAGE IF APPLICABLE IS IN (BH). THE STACK HAS ;
436 ; (DS), (AX), (BX), (CX), (DX) PUSHED. ;
437 ;-----
438 021F 33 D2 XOR DX,DX ; FIRST PRINTER
439 0221 B4 02 MOV AH,02H ; SET PRINTER STATUS REQUEST COMMAND
440 0223 CD 17 INT 17H ; REQUEST CURRENT PRINTER STATUS
441 0228 F6 C4 A0 XOR AH,080H ; CHECK FOR PRINTER BUSY (NOT CONNECTED)
442 022B 75 4E JNZ AH,0A0H ; OR OUT OF PAPER
443 ; ERROR EXIT IF PRINTER STATUS ERROR
444 022D E8 0287 R CALL CRLF ; CARRIAGE RETURN LINE FEED TO PRINTER
445
446 0230 51 PUSH CX ; SAVE SCREEN BOUNDS
447 0231 B4 03 MOV AH,03H ; NOW READ THE CURRENT CURSOR POSITION
448 0233 CD 10 INT 10H ; AND RESTORE AT END OF ROUTINE
449 0235 59 POP CX ; RECALL SCREEN BOUNDS
450 0236 52 PUSH DX ; PRESERVE THE ORIGINAL POSITION
451 0237 33 D2 XOR DX,DX ; INITIAL CURSOR (0,0) AND FIRST PRINTER
452 ;-----
453 ; THIS LOOP IS TO READ EACH CURSOR POSITION FROM THE ;
454 ; SCREEN AND PRINT IT. (BH)= VISUAL PAGE (CH)= ROWS ;
455 ;-----
456
457 0239 PRI10: MOV AH,02H ; INDICATE CURSOR SET REQUEST
458 023B CD 10 INT 10H ; NEW CURSOR POSITION ESTABLISHED
459 023D B4 08 MOV AH,08H ; INDICATE READ CHARACTER FROM DISPLAY
460 023F CD 10 INT 10H ; CHARACTER NOW IN (AL)
461 0241 0A C0 OR AL,AL ; SEE IF VALID CHAR
462 0243 75 02 JNZ PRI20 ; JUMP IF VALID CHAR
463 0245 B0 50 MOV AL,' ' ; ELSE MAKE IT A BLANK
464 0247 PRI20:
465 0247 52 PUSH DX ; SAVE CURSOR POSITION
466 0248 33 D2 XOR DX,DX ; INDICATE FIRST PRINTER (DX= 0)
467 024A 32 E4 XOR AH,AH ; INDICATE PRINT CHARACTER IN (AL)
468 024C CD 17 INT 17H ; PRINT THE CHARACTER
469 024E 5A POP DX ; RECALL CURSOR POSITION
470 024F F6 C4 29 TEST AH,29H ; TEST FOR PRINTER ERROR
471 0252 75 22 JNZ PRI70 ; EXIT IF ERROR DETECTED
472 0254 FE C2 INC DL ; ADVANCE TO NEXT COLUMN
473 0256 3A CA CMP CL,DL ; SEE IF AT END OF LINE
474 0258 75 DF JNZ PRI10 ; IF NOT LOOP FOR NEXT COLUMN
475 025A 32 D2 XOR DL,DL ; BACK TO COLUMN 0
476 025C 8A E2 MOV AH,DL ; (AH)=0
477 025E 52 POP DX ; SAVE NEW CURSOR POSITION
478 025F E8 0287 R CALL CRLF ; LINE FEED CARRIAGE RETURN
479 0262 5A POP DX ; RECALL CURSOR POSITION
480 0263 FE C6 INC DH ; ADVANCE TO NEXT LINE
481 0265 3A EE CMP CH,DH ; FINISHED?
482 0267 75 D0 JNZ PRI10 ; IF NOT LOOP FOR NEXT LINE
483
484 0269 5A POP DX ; GET CURSOR POSITION
485 026A B4 02 MOV AH,02H ; INDICATE REQUEST CURSOR SET
486 026C CD 10 INT 10H ; CURSOR POSITION RESTORED
487 026E FA CLD ; BLOCK INTERRUPTS TILL STACK CLEARED
488 026F C6 06 0100 R 00 MOV #STATUS_BYTE,0 ; MOVE OK RESULTS FLAG TO STATUS_BYTE
489 0274 EB 0B JMP SHORT PRI90 ; EXIT PRINTER ROUTINE
490
491 0276 PRI70: ; ERROR EXIT
492 0276 5A POP DX ; GET CURSOR POSITION
493 0277 B4 02 MOV AH,02H ; INDICATE REQUEST CURSOR SET
494 0279 CD 10 INT 10H ; CURSOR POSITION RESTORED
495 027B FA CLD ; BLOCK INTERRUPTS TILL STACK CLEARED
496 027C C6 06 0100 R FF MOV #STATUS_BYTE,0FFH ; SET ERROR FLAG
497 0281 PRI90:
498 0281 5A POP DX ; EXIT ROUTINE
499 0282 59 POP CX ; RESTORE ALL THE REGISTERS USED
500 0283 5B POP BX
501 0284 55 POP AX
502 0285 57 POP DS
503 0286 5F POP SI
504 0287 CF IRET ; RETURN WITH INITIAL INTERRUPT MASK
505 0287 PRINT_SCREEN_1 ENDP

```

SECTION 5

```

506
507
508
509 0287          CRLF      PROC      NEAR
510
511 0287 33 D2          XOR      DX,DX          ; SEND CR,LF TO FIRST PRINTER
512 0289 B8 000D      MOV      AX,OR          ; ASSUME FIRST PRINTER (DX=0)
513 028C CD 17          INT      17H           ; GET THE PRINT CHARACTER COMMAND AND
514 028E B8 000A      MOV      AX,LF         ; THE CARRIAGE RETURN CHARACTER
515 0291 CD 17          INT      17H           ; NOW GET THE LINE FEED AND
516 0293 C3          RET                      ; SEND IT TO THE BIOS PRINTER ROUTINE
517 0294          CRLF      ENDP
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537 0294          TIMER_INT_1  PROC      FAR
538 0294 FB          STI                      ; INTERRUPTS BACK ON
539 0295 IE          PUSH     DS
540 0296 50          PUSH     AX
541 0297 52          PUSH     DX
542 0298 EB 0000 E  CALL     DOS
543 029B FF 06 006C R INC      @TIMER_LOW
544 029F 75 04      JNZ     T4
545 02A1 FF 06 006E R INC      @TIMER_HIGH
546 02A5          T4:          CMP      @TIMER_HIGH,010H
547 02A5 83 3E 006E R 18 JNZ     T5
548 02AA 75 15      JNZ     T5
549 02AC 81 3E 006C R 00B0 CMP      @TIMER_LOW,0B0H
550 02B2 75 0D      JNZ     T5
551
552
553
554 02B4 2B C0      SUB      AX,AX
555 02B6 A3 006E R  MOV      @TIMER_HIGH,AX
556 02B9 A3 006C R  MOV      @TIMER_LOW,AX
557 02BC C6 06 0070 R 01 MOV      @TIMER_OFL,1
558
559
560
561 02C1          T5:          TEST    FOR DISKETTE TIME OUT
562 02C1 FE 0E 0040 R DEC      @MOTOR_COUNT
563 02C5 75 0B      JNZ     T6
564 02C7 80 26 003F R F0 AND      @MOTOR_STATUS,0F0H
565 02CC 80 0C      MOV      AL,0CH
566 02CE BA 03F2     MOV      DX,03F2H
567 02D1 EE          OUT     DX,AL
568
569 02D2          T6:          ; TIMER TICK INTERRUPT
570 02D2 CD 1C      INT     1CH
571
572 02D4 5A          POP     DX
573 02D5 B0 20      MOV     AL,ED1
574 02D7 FA          CLI
575 02D8 E6 20      OUT    INTA00,AL
576 02DA 58          POP     AX
577 02DB 1F          POP     DS
578 02DC CF          IRET
579
580 02DD          TIMER_INT_1  ENDP
581
582 02DD          CODE   ENDS
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600

```

```

1 PAGE 118,121
2 TITLE ORGS ----- 04/21/86 COMPATIBILITY MODULE
3 .LIST
4 0000 CODE SEGMENT BYTE PUBLIC
5
6 PUBLIC A1
7 PUBLIC CONF_TBL
8 PUBLIC CRT_CHAR_GEN
9 PUBLIC D1
10 PUBLIC D2
11 PUBLIC D2A
12 PUBLIC DISK_BASE
13 PUBLIC DUMMY_RETURN
14 PUBLIC E101
15 PUBLIC E102
16 PUBLIC E103
17 PUBLIC E104
18 PUBLIC E105
19 PUBLIC E106
20 PUBLIC E107
21 PUBLIC E108
22 PUBLIC E109
23 PUBLIC E161
24 PUBLIC E162
25 PUBLIC E163
26 PUBLIC E164
27 PUBLIC E201
28 PUBLIC E202
29 PUBLIC E203
30 PUBLIC E301
31 PUBLIC E302
32 PUBLIC E303
33 PUBLIC E304
34 PUBLIC E401
35 PUBLIC E501
36 PUBLIC E601
37 PUBLIC E602
38 PUBLIC F1780
39 PUBLIC F1781
40 PUBLIC F1782
41 PUBLIC F1790
42 PUBLIC F1791
43 PUBLIC F301
44 PUBLIC F302
45 PUBLIC F303
46 PUBLIC FD_TBL
47 PUBLIC FLOPPY
48 PUBLIC HRD
49 PUBLIC K6
50 PUBLIC K6L
51 PUBLIC K7
52 PUBLIC K8
53 PUBLIC K10
54 PUBLIC K11
55 PUBLIC K12
56 PUBLIC K14
57 PUBLIC K15
58 PUBLIC M4
59 PUBLIC M5
60 PUBLIC M6
61 PUBLIC NT
62 PUBLIC NMI_INT
63 PUBLIC PRINT_SCREEN
64 PUBLIC PDR
65 PUBLIC SEGRS_I
66 PUBLIC SLAVE_VECTOR_TABLE
67 PUBLIC TUTOR
68 PUBLIC VECTOR_TABLE
69 PUBLIC VIDEO_FARMS
70
71 EXTRN BOOT_STRAP_I:NEAR
72 EXTRN CASSETTE_IO_I:NEAR
73 EXTRN D11:NEAR
74 EXTRN DISK_INT_I:NEAR
75 EXTRN DISK_SETUP:NEAR
76 EXTRN DISKETTE_IO_I:NEAR
77 EXTRN DSKETTE_SETUP:NEAR
78 EXTRN EQUIPMENT_I:NEAR
79 EXTRN INT_281:NEAR
80 EXTRN K16:NEAR
81 EXTRN KEYBOARD_IO_I:NEAR
82 EXTRN KB_INT_I:NEAR
83 EXTRN MEMORY_SIZE_DET_I:NEAR
84 EXTRN NMI_INT_I:NEAR
85 EXTRN PRINT_SCREEN_I:NEAR
86 EXTRN PRINTER_IO_I:NEAR
87 EXTRN RE_DIRECT:NEAR
88 EXTRN RS232_IO_I:NEAR
89 EXTRN RTC_INT:NEAR
90 EXTRN SEC:NEAR
91 EXTRN START_I:NEAR
92 EXTRN TIME_OF_DAY_I:NEAR
93 EXTRN TIMER_INT_I:NEAR
94 EXTRN VIDEO_IO_T:NEAR
95
96 ASSUME CS:CODE,DS:DATA
97
98 -----
99
100 THIS MODULE HAS BEEN ADDED TO FACILITATE THE EXPANSION OF THIS PROGRAM.
101 IT ALLOWS FOR THE FIXED ORG STATEMENT ENTRY POINTS THAT HAVE TO REMAIN
102 AT THE SAME ADDRESSES. THE USE OF ENTRY POINTS AND TABLES WITHIN THIS
103 MODULE SHOULD BE AVOIDED AND ARE INCLUDED ONLY TO SUPPORT EXISTING CODE
104 THAT VIOLATE THE STRUCTURE AND DESIGN OF BIOS. ALL BIOS ACCESS SHOULD
105 USE THE DOCUMENTED INTERRUPT VECTOR INTERFACE FOR COMPATIBILITY.
106
107

```

SECTION 5

```

108                                     PAGE
109                                     |-----|
110                                     | COPYRIGHT NOTICE |
111                                     |-----|
112                                     | ORG 0E000H |
113 0000                                | ORG 00000H |
114                                     |-----|
115 0000 37 38 58 37 34 36             DB '78X7462 COPR. IBM 1981, 1986 '
116                                     32 20 43 4F 50 52
117                                     2E 20 49 42 40 20
118                                     31 39 38 31 2C 20
119                                     31 39 38 36 20 20
120                                     20 20
121
122                                     |-----|
123                                     | PARITY ERROR MESSAGES |
124                                     |-----|
125
126 0020 60 41 52 49 54 59             D1 DB 'PARITY CHECK 1',CR,LF ; PLANAR BOARD PARITY CHECK LATCH SET
127                                     20 43 48 45 43 4B
128                                     20 31 0D 0A
129 0030 50 41 52 49 54 59             D2 DB 'PARITY CHECK 2',CR,LF ; I/O CHANNEL CHECK LATCH SET
130                                     20 43 48 45 43 4B
131                                     20 3E 0D 0A
132 0040 3F 3F 3F 3F 3F 0D             D2A DB '?????',CR,LF
133                                     0A
134 = 0047                                IP = $
135                                     | ORG 0E05BH |
136 005B                                | ORG 0005BH |
137 005B                                RESET: JUMP START_1 ; RESET START
138 005B E9 0000 E                       ; VECTOR ON TO THE MOVED POST CODE
139
140                                     |-----|
141                                     | POST ERROR MESSAGES |
142                                     |-----|
143
144 005E 20 31 30 31 2D 53             E101 DB ' 101-System Board Error',CR,LF ; INTERRUPT FAILURE
145                                     79 73 74 65 6D 20
146                                     42 6F 61 72 64 20
147                                     45 72 72 6F 72 0D
148                                     0A
149 0077 20 31 30 32 2D 53             E102 DB ' 102-System Board Error',CR,LF ; TIMER FAILURE
150                                     79 73 74 65 6D 20
151                                     42 6F 61 72 64 20
152                                     45 72 72 6F 72 0D
153                                     0A
154 0090 20 31 30 33 2D 53             E103 DB ' 103-System Board Error',CR,LF ; TIMER INTERRUPT FAILURE
155                                     79 73 74 65 6D 20
156                                     42 6F 61 72 64 20
157                                     45 72 72 6F 72 0D
158                                     0A
159 00A9 20 31 30 34 2D 53             E104 DB ' 104-System Board Error',CR,LF ; PROTECTED MODE FAILURE
160                                     79 73 74 65 6D 20
161                                     42 6F 61 72 64 20
162                                     45 72 72 6F 72 0D
163                                     0A
164 00C2 20 31 30 35 2D 53             E105 DB ' 105-System Board Error',CR,LF ; LAST 8042 COMMAND NOT ACCEPTED
165                                     79 73 74 65 6D 20
166                                     42 6F 61 72 64 20
167                                     45 72 72 6F 72 0D
168                                     0A
169 00DB 20 31 30 36 2D 53             E106 DB ' 106-System Board Error',CR,LF ; CONVERTING LOGIC TEST
170                                     79 73 74 65 6D 20
171                                     42 6F 61 72 64 20
172                                     45 72 72 6F 72 0D
173                                     0A
174 00F4 20 31 30 37 2D 53             E107 DB ' 107-System Board Error',CR,LF ; HOT NMI TEST
175                                     79 73 74 65 6D 20
176                                     42 6F 61 72 64 20
177                                     45 72 72 6F 72 0D
178                                     0A
179 010D 20 31 30 38 2D 53             E108 DB ' 108-System Board Error',CR,LF ; TIMER BUS TEST
180                                     79 73 74 65 6D 20
181                                     42 6F 61 72 64 20
182                                     45 72 72 6F 72 0D
183                                     0A
184 0126 20 31 30 39 2D 53             E109 DB ' 109-System Board Error',CR,LF ; LOW MEG CHIP SELECT TEST
185                                     79 73 74 65 6D 20
186                                     42 6F 61 72 64 20
187                                     45 72 72 6F 72 0D
188                                     0A
189 013F 20 31 36 31 2D 53             E161 DB ' 161-System Options Not Set-(Run SETUP)',CR,LF ; DEAD BATTERY
190                                     79 73 74 65 6D 20
191                                     4F 70 74 69 6F 6E
192                                     73 20 4E 6F 74 20
193                                     58 6B 74 2D 28 52
194                                     7B 6E 20 53 4B 64
195                                     65 60 29 0D 0A
196 0168 20 31 36 32 2D 53             E162 DB ' 162-System Options Not Set-(Run SETUP)',CR,LF ;CHECKSUM/CONFIG
197                                     79 73 74 65 6D 20
198                                     4F 70 74 69 6F 6E
199                                     73 20 4E 6F 74 20
200                                     58 65 74 2D 28 52
201                                     75 6E 20 53 4B 64
202                                     55 50 29 0D 0A
203 0191 20 31 36 33 2D 54             E163 DB ' 163-Time & Date Not Set-(Run SETUP)',CR,LF ;CLOCK NOT UPDATING
204                                     69 6D 65 20 26 20
205                                     44 61 74 65 20 4E
206                                     6F 74 20 53 65 74
207                                     2D 28 52 75 6E 20
208                                     53 45 54 65 50 29
209                                     0D 0A
210 01B7 20 31 36 34 2D 4D             E164 DB ' 164-Memory Size Error-(Run SETUP)',CR,LF ; CMOS DOES NOT MATCH
211                                     65 6D 6F 72 79 20
212                                     53 69 7A 65 20 45
213                                     72 72 6F 72 2D 28
214                                     52 75 6E 20 53 45
215                                     54 55 50 29 0D 0A
216 01DB 20 32 30 31 2D 4D             E201 DB ' 201-Memory Error',CR,LF
217                                     65 6D 6F 72 79 20
218                                     45 72 72 6F 72 0D
219                                     0A
220 01EE 20 32 30 32 2D 4D             E202 DB ' 202-Memory Address Error',CR,LF ; LINE ERROR 00->16
221                                     65 6D 6F 72 79 20
  
```

```

222      41 64 64 72 65 73
223      73 20 45 72 72 6F
224      72 0D 0A
225 0209 20 32 30 33 2D 4D E203 DB ' 203-Memory Address Error',CR,LF ; LINE ERROR 16->23
226      65 6D 6F 72 79 20
227      41 64 64 72 65 73
228      73 20 45 72 72 6F
229      72 0D 0A
230 0224 20 33 30 31 2D 4B E301 DB ' 301-Keyboard Error',CR,LF ; KEYBOARD ERROR
231      65 79 62 6F 61 72
232      64 20 45 72 72 6F
233      72 0D 0A
234 0239 20 33 30 32 2D 53 E302 DB ' 302-System Unit Keylock is Locked',CR,LF ; KEYBOARD LOCK ON
235      65 79 62 6F 61 72
236      55 6E 69 74 20 4B
237      65 79 6C 6F 63 6B
238      20 69 73 20 4C 6F
239      63 6B 65 64 0D 0A
240 025D 20 28 52 45 53 55 F3D DB ' (RESUME = "F1" KEY)',CR,LF
241      4D 45 20 3D 20 22
242      46 31 22 20 4B 45
243      59 29 0D 0A
244
245 ;----- NMI ENTRY
246
247 = 0273
248
249 02C3
250 = 02C3
251 02C3 E9 0000 E
252
253 02C6 20 33 30 33 2D 4B E303 DB ' 303-Keyboard Or System Unit Error',CR,LF
254      65 79 62 6F 61 72
255      64 20 4F 72 20 53
256      79 73 74 65 6D 20
257      55 6E 69 74 20 45
258      72 72 6F 72 0D 0A
259
260 02EA 20 33 30 34 2D 4B E304 DB ' 304-Keyboard Or System Unit Error',CR,LF ; KEYBOARD CLOCK HIGH
261      65 79 62 6F 61 72
262      64 20 4F 72 20 53
263      79 73 74 65 6D 20
264      55 6E 69 74 20 45
265      72 72 6F 72 0D 0A
266 030E 20 34 30 31 2D 43 E401 DB ' 401-CRT Error',CR,LF ; MONOCHROME
267      52 54 20 45 72 72
268      6F 72 0D 0A
269 031E 20 35 31 2D 43 E501 DB ' 501-CRT Error',CR,LF ; COLOR
270      52 54 20 45 72 72
271      6F 72 0D 0A
272 032E 20 36 30 31 2D 44 E601 DB ' 601-Diskette Error',CR,LF ; DISKETTE ERROR
273      69 73 6B 65 74 74
274      65 20 45 72 72 6F
275      72 0D 0A
276
277 0343 20 36 30 32 2D 44 E602 DB ' 602-Diskette Boot Record Error',CR,LF
278      69 73 6B 65 74 74
279      65 20 42 6F 6F 74
280      20 52 65 63 6F 72
281      64 20 45 72 72 6F
282      72 0D 0A
283
284 0364 31 37 38 30 2D 44 F1780 DB '1780-Disk 0 Failure',CR,LF
285      69 73 6B 20 30 20
286      46 61 69 6C 75 72
287      65 0D 0A
288 0379 31 37 38 31 2D 44 F1781 DB '1781-Disk 1 Failure',CR,LF
289      69 73 6B 20 31 20
290      46 61 69 6C 75 72
291      65 0D 0A
292 038E 31 37 38 32 2D 44 F1782 DB '1782-Disk Controller Failure',CR,LF
293      69 73 6B 20 43 6F
294      6E 74 72 6F 6C 6C
295      65 72 20 46 61 69
296      6C 75 72 65 0D 0A
297 03AC 31 37 39 30 2D 44 F1790 DB '1790-Disk 0 Error',CR,LF
298      69 73 6B 20 30 20
299      45 72 72 6F 72 0D
300      0A
301 03BF 31 37 39 31 2D 44 F1791 DB '1791-Disk 1 Error',CR,LF
302      69 73 6B 20 31 20
303      45 72 72 6F 72 0D
304      0A
305
306 03D2 52 4F 4D 20 2D 45 F3A DB 'ROM Error ',CR,LF ; ROM CHECKSUM
307      72 72 6F 72 20 0D
308      0A
309 03DF 20 20 20 2D 55 F3D1 DB ' -Unlock System Unit Keylock ',CR,LF
310      6E 6C 6F 63 6B 20
311      53 79 73 74 65 6D
312      20 58 6E 69 74 20
313      48 65 79 6C 6F 63
314      6B 20 0D 0A

```

SECTION 5

```

315                                     PAGE
316                                     |-----|
317                                     | INITIALIZE DRIVE CHARACTERISTICS |
318                                     |-----|
319                                     | FIXED DISK PARAMETER TABLE |
320                                     |-----|
321                                     | - THE TABLE IS COMPOSED OF A BLOCK DEFINED AS: |
322                                     |-----|
323                                     | +0 (1 WORD) - MAXIMUM NUMBER OF CYLINDERS |
324                                     | +2 (1 BYTE) - MAXIMUM NUMBER OF HEADS |
325                                     | +3 (1 WORD) - NOT USED/SEE PC-XT |
326                                     | +5 (1 WORD) - STARTING WRITE PRECOMPENSATION CYL |
327                                     | +7 (1 BYTE) - NOT USED/SEE PC-XT |
328                                     | +8 (1 BYTE) - CONTROL BYTE |
329                                     | BIT 7 DISABLE RETRIES -OR- |
330                                     | BIT 6 DISABLE RETRIES |
331                                     | BIT 3 MORE THAN 8 HEADS |
332                                     | +9 (3 BYTES) - NOT USED/SEE PC-XT |
333                                     | +12 (1 WORD) - LANDING ZONE |
334                                     | +14 (1 BYTE) - NUMBER OF SECTORS/TRACK |
335                                     | +15 (1 BYTE) - RESERVED FOR FUTURE USE |
336                                     |-----|
337                                     | - TO DYNAMICALLY DEFINE A SET OF PARAMETERS |
338                                     | BUILD A TABLE FOR UP TO 15 TYPES AND PLACE |
339                                     | THE CORRESPONDING VECTOR INTO INTERRUPT 41 |
340                                     | FOR DRIVE 0 AND INTERRUPT 46 FOR DRIVE 1. |
341                                     |-----|
342
343
344 0401                                FD_TBL:
345
346
347
348 0401 0132                            |-----|
349 0403 04                              | DRIVE TYPE 01 |
350 0404 0000                            | DW 0306D     | CYLINDERS
351 0406 0080                            | DB 04D       | HEADS
352 0408 00                              | DW 0128D     | WRITE PRE-COMPENSATION CYLINDER
353 0409 00                              | DB 0         |
354 040A 00 00 00                        | DB 0,0,0    | CONTROL BYTE
355 040D 0131                            | DW 0305D     | LANDING ZONE
356 040F 11                              | DB 17D       | SECTORS/TRACK
357 0410 00                              | DB 0         |
358
359
360
361 0411 0267                            |-----|
362 0413 04                              | DRIVE TYPE 02 |
363 0414 0000                            | DW 0615D     | CYLINDERS
364 0416 012C                            | DB 04D       | HEADS
365 0418 00                              | DW 0300D     | WRITE PRE-COMPENSATION CYLINDER
366 0419 00                              | DB 0         |
367 041A 00 00 00                        | DB 0,0,0    | CONTROL BYTE
368 041D 0267                            | DW 0615D     | LANDING ZONE
369 041F 11                              | DB 17D       | SECTORS/TRACK
370 0420 00                              | DB 0         |
371
372
373
374 0421 0267                            |-----|
375 0423 06                              | DRIVE TYPE 03 |
376 0424 0000                            | DW 0615D     | CYLINDERS
377 0426 012C                            | DB 06D       | HEADS
378 0428 00                              | DW 0300D     | WRITE PRE-COMPENSATION CYLINDER
379 0429 00                              | DB 0         |
380 042A 00 00 00                        | DB 0,0,0    | CONTROL BYTE
381 042D 0267                            | DW 0615D     | LANDING ZONE
382 042F 11                              | DB 17D       | SECTORS/TRACK
383 0430 00                              | DB 0         |
384
385
386
387 0431 03AC                            |-----|
388 0433 08                              | DRIVE TYPE 04 |
389 0434 0000                            | DW 0940D     | CYLINDERS
390 0436 0200                            | DB 08D       | HEADS
391 0438 00                              | DW 0512D     | WRITE PRE-COMPENSATION CYLINDER
392 0439 00                              | DB 0         |
393 043A 00 00 00                        | DB 0,0,0    | CONTROL BYTE
394 043D 03AC                            | DW 0940D     | LANDING ZONE
395 043F 11                              | DB 17D       | SECTORS/TRACK
396 0440 00                              | DB 0         |
397
398
399
400 0441 03AC                            |-----|
401 0443 06                              | DRIVE TYPE 05 |
402 0444 0000                            | DW 0940D     | CYLINDERS
403 0446 0200                            | DB 06D       | HEADS
404 0448 00                              | DW 0512D     | WRITE PRE-COMPENSATION CYLINDER
405 0449 00                              | DB 0         |
406 044A 00 00 00                        | DB 0,0,0    | CONTROL BYTE
407 044D 03AC                            | DW 0940D     | LANDING ZONE
408 044F 11                              | DB 17D       | SECTORS/TRACK
409 0450 00                              | DB 0         |
410
411
412
413 0451 0267                            |-----|
414 0453 04                              | DRIVE TYPE 06 |
415 0454 0000                            | DW 0615D     | CYLINDERS
416 0456 FFFF                            | DB 04D       | HEADS
417 0458 00                              | DW 0         | NO WRITE PRE-COMPENSATION
418 0459 00                              | DB 0         |
419 045A 00 00 00                        | DB 0,0,0    | CONTROL BYTE
420 045D 0267                            | DW 0615D     | LANDING ZONE
421 045F 11                              | DB 17D       | SECTORS/TRACK
422 0460 00                              | DB 0         |

```

```

423                                     PAGE
424                                     |----- DRIVE TYPE 07
425
426 0461 01CE                            DW 0462D                | CYLINDERS
427 0463 08                               DB 08D                  | HEADS
428 0464 0000                             DW 0                      |
429 0466 0100                             DW 0266D                | WRITE PRE-COMPENSATION CYLINDER
430 0468 00                               DB 0                      |
431 0469 00                               DB 0                      |
432 046A 00 00 00                         DB 0,0,0                | CONTROL BYTE
433 046D 01FF                             DW 0511D                | LANDING ZONE
434 046F 11                               DB 17D                  | SECTORS/TRACK
435 0470 00                               DB 0                      |
436
437                                     |----- DRIVE TYPE 08
438
439 0471 02DD                             DW 0733D                | CYLINDERS
440 0473 05                               DB 05D                  | HEADS
441 0474 0000                             DW 0                      |
442 0476 FFFF                             DW 0FFFFH               | NO WRITE PRE-COMPENSATION
443 0478 00                               DB 0                      |
444 0479 00                               DB 0                      | CONTROL BYTE
445 047A 00 00 00                         DB 0,0,0                |
446 047D 02DD                             DW 0733D                | LANDING ZONE
447 047F 11                               DB 17D                  | SECTORS/TRACK
448 0480 00                               DB 0                      |
449
450                                     |----- DRIVE TYPE 09
451
452 0481 0384                             DW 0900D                | CYLINDERS
453 0483 0F                               DB 16D                  | HEADS
454 0484 0000                             DW 0                      |
455 0486 FFFF                             DW 0FFFFH               | NO WRITE PRE-COMPENSATION
456 0488 00                               DB 0                      |
457 0489 08                               DB 008H                 | CONTROL BYTE
458 048A 00 00 00                         DB 0,0,0                |
459 048D 0385                             DW 0901D                | LANDING ZONE
460 048F 11                               DB 17D                  | SECTORS/TRACK
461 0490 00                               DB 0                      |
462
463                                     |----- DRIVE TYPE 10
464
465 0491 0334                             DW 0820D                | CYLINDERS
466 0493 03                               DB 03D                  | HEADS
467 0494 0000                             DW 0                      |
468 0496 FFFF                             DW 0FFFFH               | NO WRITE PRE-COMPENSATION
469 0498 00                               DB 0                      |
470 0499 00                               DB 0                      | CONTROL BYTE
471 049A 00 00 00                         DB 0,0,0                |
472 049D 0334                             DW 0820D                | LANDING ZONE
473 049F 11                               DB 17D                  | SECTORS/TRACK
474 04A0 00                               DB 0                      |
475
476                                     |----- DRIVE TYPE 11
477
478 04A1 0357                             DW 0855D                | CYLINDERS
479 04A3 05                               DB 05D                  | HEADS
480 04A4 0000                             DW 0                      |
481 04A6 FFFF                             DW 0FFFFH               | NO WRITE PRE-COMPENSATION
482 04A8 00                               DB 0                      |
483 04A9 00                               DB 0                      | CONTROL BYTE
484 04AA 00 00 00                         DB 0,0,0                |
485 04AD 0357                             DW 0855D                | LANDING ZONE
486 04AF 11                               DB 17D                  | SECTORS/TRACK
487 04B0 00                               DB 0                      |
488
489                                     |----- DRIVE TYPE 12
490
491 04B1 0357                             DW 0855D                | CYLINDERS
492 04B3 07                               DB 07D                  | HEADS
493 04B4 0000                             DW 0                      |
494 04B6 FFFF                             DW 0FFFFH               | NO WRITE PRE-COMPENSATION
495 04B8 00                               DB 0                      |
496 04B9 00                               DB 0                      | CONTROL BYTE
497 04BA 00 00 00                         DB 0,0,0                |
498 04BD 0357                             DW 0855D                | LANDING ZONE
499 04BF 11                               DB 17D                  | SECTORS/TRACK
500 04C0 00                               DB 0                      |
501
502                                     |----- DRIVE TYPE 13
503
504 04C1 0132                             DW 0306D                | CYLINDERS
505 04C3 08                               DB 08D                  | HEADS
506 04C4 0000                             DW 0                      |
507 04C6 0080                             DW 0128D                | WRITE PRE-COMPENSATION CYLINDER
508 04C8 00                               DB 0                      |
509 04C9 00                               DB 0                      | CONTROL BYTE
510 04CA 00 00 00                         DB 0,0,0                |
511 04CD 013F                             DW 0319D                | LANDING ZONE
512 04CF 11                               DB 17D                  | SECTORS/TRACK
513 04D0 00                               DB 0                      |
514
515                                     |----- DRIVE TYPE 14
516
517 04D1 02DD                             DW 0733D                | CYLINDERS
518 04D3 07                               DB 07D                  | HEADS
519 04D4 0000                             DW 0                      |
520 04D6 FFFF                             DW 0FFFFH               | NO WRITE PRE-COMPENSATION
521 04D8 00                               DB 0                      |
522 04D9 00                               DB 0                      | CONTROL BYTE
523 04DA 00 00 00                         DB 0,0,0                |
524 04DD 02DD                             DW 0733D                | LANDING ZONE
525 04DF 11                               DB 17D                  | SECTORS/TRACK
526 04E0 00                               DB 0                      |

```

SECTION 5

```

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

```

```

631 PAGE
632 ;----- DRIVE TYPE 23
633
634 0561 0132 DW 0306D ; CYLINDERS
635 0563 04 DB 04D ; HEADS
636 0564 0000 DW 0 ;
637 0566 0000 DW 0000D ; WRITE PRE-COMPENSATION ALL CYL
638 0568 00 DB 0 ;
639 0569 00 DB 0 ; CONTROL BYTE
640 056A 00 00 00 DB 0,0,0 ;
641 056D 0150 DW 0336D ; LANDING ZONE
642 056F 11 DB 17D ; SECTORS/TRACK
643 0570 00 DB 0 ;
644
645 ;----- DRIVE TYPE 24
646
647 0571 0264 DW 0612D ; CYLINDERS
648 0573 04 DB 04D ; HEADS
649 0574 0000 DW 0 ;
650 0576 0131 DW 0305D ; WRITE PRE-COMPENSATION CYL
651 0578 00 DB 0 ;
652 0579 00 DB 0 ; CONTROL BYTE
653 057A 00 00 00 00 DB 0,0,0,0 ;
654 057D 0297 DW 0663D ; LANDING ZONE
655 057F 11 DB 17D ; SECTORS/TRACK
656 0580 00 DB 0 ;
657
658 ;----- DRIVE TYPE 25 *** RESERVED***
659
660 0581 0000 DW 0000D ; CYLINDERS
661 0583 00 DB 00D ; HEADS
662 0584 0000 DW 0 ;
663 0586 0000 DW 0000D ; WRITE PRE-COMPENSATION CYL
664 0588 00 DB 0 ;
665 0589 00 DB 0 ; CONTROL BYTE
666 058A 00 00 00 00 DB 0,0,0,0 ;
667 058D 0000 DW 0000D ; LANDING ZONE
668 058F 00 DB 00D ; SECTORS/TRACK
669 0590 00 DB 0 ;
670
671 ;----- DRIVE TYPE 26 *** RESERVED***
672
673 0591 0000 DW 0000D ; CYLINDERS
674 0593 00 DB 00D ; HEADS
675 0594 0000 DW 0 ;
676 0596 0000 DW 0000D ; WRITE PRE-COMPENSATION CYL
677 0598 00 DB 0 ;
678 0599 00 DB 0 ; CONTROL BYTE
679 059A 00 00 00 00 DB 0,0,0,0 ;
680 059D 0000 DW 0000D ; LANDING ZONE
681 059F 00 DB 00D ; SECTORS/TRACK
682 05A0 00 DB 0 ;
683
684 ;----- DRIVE TYPE 27 *** RESERVED***
685
686 05A1 0000 DW 0000D ; CYLINDERS
687 05A3 00 DB 00D ; HEADS
688 05A4 0000 DW 0 ;
689 05A6 0000 DW 0000D ; WRITE PRE-COMPENSATION CYL
690 05A8 00 DB 0 ;
691 05A9 00 DB 0 ; CONTROL BYTE
692 05AA 00 00 00 00 DB 0,0,0,0 ;
693 05AD 0000 DW 0000D ; LANDING ZONE
694 05AF 00 DB 00D ; SECTORS/TRACK
695 05B0 00 DB 0 ;
696
697 ;----- DRIVE TYPE 28 *** RESERVED***
698
699 05B1 0000 DW 0000D ; CYLINDERS
700 05B3 00 DB 00D ; HEADS
701 05B4 0000 DW 0 ;
702 05B6 0000 DW 0000D ; WRITE PRE-COMPENSATION CYL
703 05B8 00 DB 0 ;
704 05B9 00 DB 0 ; CONTROL BYTE
705 05BA 00 00 00 00 DB 0,0,0,0 ;
706 05BD 0000 DW 0000D ; LANDING ZONE
707 05BF 00 DB 00D ; SECTORS/TRACK
708 05C0 00 DB 0 ;
709
710 ;----- DRIVE TYPE 29 *** RESERVED***
711
712 05C1 0000 DW 0000D ; CYLINDERS
713 05C3 00 DB 00D ; HEADS
714 05C4 0000 DW 0 ;
715 05C6 0000 DW 0000D ; WRITE PRE-COMPENSATION CYL
716 05C8 00 DB 0 ;
717 05C9 00 DB 0 ; CONTROL BYTE
718 05CA 00 00 00 00 DB 0,0,0,0 ;
719 05CD 0000 DW 0000D ; LANDING ZONE
720 05CF 00 DB 00D ; SECTORS/TRACK
721 05D0 00 DB 0 ;
722
723 ;----- DRIVE TYPE 30 *** RESERVED***
724
725 05D1 0000 DW 0000D ; CYLINDERS
726 05D3 00 DB 00D ; HEADS
727 05D4 0000 DW 0 ;
728 05D6 0000 DW 0000D ; WRITE PRE-COMPENSATION CYL
729 05D8 00 DB 0 ;
730 05D9 00 DB 0 ; CONTROL BYTE
731 05DA 00 00 00 00 DB 0,0,0,0 ;
732 05DD 0000 DW 0000D ; LANDING ZONE
733 05DF 00 DB 00D ; SECTORS/TRACK
734 05E0 00 DB 0 ;

```

SECTION 5

```

735          PAGE
736          |----- DRIVE TYPE 31 *** RESERVED***
737
738 05E1 0000      DW 0000D      | CYLINDERS
739 05E2 00        DB 00D        | HEADS
740 05E4 0000      DW 0         |
741 05E6 0000      DW 0000D      | WRITE PRE-COMPENSATION CYL
742 05E8 00        DB 0         |
743 05E9 00        DB 0         | CONTROL BYTE
744 05EA 00 00 00  DB 0,0,0      |
745 05ED 0000      DW 0000D      | LANDING ZONE
746 05EF 00        DB 00D        | SECTORS/TRACK
747 05F0 00        DB 0         |
748
749          |----- DRIVE TYPE 32 *** RESERVED***
750
751 05F1 0000      DW 0000D      | CYLINDERS
752 05F3 00        DB 00D        | HEADS
753 05F4 0000      DW 0         |
754 05F6 0000      DW 0000D      | WRITE PRE-COMPENSATION CYL
755 05F8 00        DB 0         |
756 05F9 00        DB 0         | CONTROL BYTE
757 05FA 00 00 00  DB 0,0,0      |
758 05FD 0000      DW 0000D      | LANDING ZONE
759 05FF 00        DB 00D        | SECTORS/TRACK
760 0600 00        DB 0         |
761
762          |----- DRIVE TYPE 33 *** RESERVED***
763
764 0601 0000      DW 0000D      | CYLINDERS
765 0603 00        DB 00D        | HEADS
766 0604 0000      DW 0         |
767 0606 0000      DW 0000D      | WRITE PRE-COMPENSATION CYL
768 0608 00        DB 0         |
769 0609 00        DB 0         | CONTROL BYTE
770 060A 00 00 00  DB 0,0,0      |
771 060D 0000      DW 0000D      | LANDING ZONE
772 060F 00        DB 00D        | SECTORS/TRACK
773 0610 00        DB 0         |
774
775          |----- DRIVE TYPE 34 *** RESERVED***
776
777 0611 0000      DW 0000D      | CYLINDERS
778 0613 00        DB 00D        | HEADS
779 0614 0000      DW 0         |
780 0616 0000      DW 0000D      | WRITE PRE-COMPENSATION CYL
781 0618 00        DB 0         |
782 0619 00        DB 0         | CONTROL BYTE
783 061A 00 00 00  DB 0,0,0      |
784 061D 0000      DW 0000D      | LANDING ZONE
785 061F 00        DB 00D        | SECTORS/TRACK
786 0620 00        DB 0         |
787
788          |----- DRIVE TYPE 35 *** RESERVED***
789
790 0621 0000      DW 0000D      | CYLINDERS
791 0623 00        DB 00D        | HEADS
792 0624 0000      DW 0         |
793 0626 0000      DW 0000D      | WRITE PRE-COMPENSATION CYL
794 0628 00        DB 0         |
795 0629 00        DB 0         | CONTROL BYTE
796 062A 00 00 00  DB 0,0,0      |
797 062D 0000      DW 0000D      | LANDING ZONE
798 062F 00        DB 00D        | SECTORS/TRACK
799 0630 00        DB 0         |
800
801          |----- DRIVE TYPE 36 *** RESERVED***
802
803 0631 0000      DW 0000D      | CYLINDERS
804 0633 00        DB 00D        | HEADS
805 0634 0000      DW 0         |
806 0636 0000      DW 0000D      | WRITE PRE-COMPENSATION CYL
807 0638 00        DB 0         |
808 0639 00        DB 0         | CONTROL BYTE
809 063A 00 00 00  DB 0,0,0      |
810 063D 0000      DW 0000D      | LANDING ZONE
811 063F 00        DB 00D        | SECTORS/TRACK
812 0640 00        DB 0         |
813
814          |----- DRIVE TYPE 37 *** RESERVED***
815
816 0641 0000      DW 0000D      | CYLINDERS
817 0643 00        DB 00D        | HEADS
818 0644 0000      DW 0         |
819 0646 0000      DW 0000D      | WRITE PRE-COMPENSATION CYL
820 0648 00        DB 0         |
821 0649 00        DB 0         | CONTROL BYTE
822 064A 00 00 00  DB 0,0,0      |
823 064D 0000      DW 0000D      | LANDING ZONE
824 064F 00        DB 00D        | SECTORS/TRACK
825 0650 00        DB 0         |
826
827          |----- DRIVE TYPE 38 *** RESERVED***
828
829 0651 0000      DW 0000D      | CYLINDERS
830 0653 00        DB 00D        | HEADS
831 0654 0000      DW 0         |
832 0656 0000      DW 0000D      | WRITE PRE-COMPENSATION CYL
833 0658 00        DB 0         |
834 0659 00        DB 0         | CONTROL BYTE
835 065A 00 00 00  DB 0,0,0      |
836 065D 0000      DW 0000D      | LANDING ZONE
837 065F 00        DB 00D        | SECTORS/TRACK
838 0660 00        DB 0         |

```

```

839          PAGE
840          |----- DRIVE TYPE 39      *** RESERVED***
841
842          DW 0661 0000          ; CYLINDERS
843          DB 0663 00           ; HEADS
844          DW 0664 0000          ; WRITE PRE-COMPENSATION CYL
845          DW 0666 0000          ;
846          DB 0668 00           ; CONTROL BYTE
847          DB 0669 00           ;
848          DB 066A 00 00 00      ;
849          DW 066D 0000          ; LANDING ZONE
850          DB 066F 00           ; SECTORS/TRACK
851          DB 0670 00           ;
852
853          |----- DRIVE TYPE 40      *** RESERVED***
854
855          DW 0671 0000          ; CYLINDERS
856          DB 0673 00           ; HEADS
857          DW 0674 0000          ;
858          DW 0676 0000          ; WRITE PRE-COMPENSATION CYL
859          DB 0678 00           ;
860          DB 0679 00           ; CONTROL BYTE
861          DB 067A 00 00 00      ;
862          DW 067D 0000          ; LANDING ZONE
863          DB 067F 00           ; SECTORS/TRACK
864          DB 0680 00           ;
865
866          |----- DRIVE TYPE 41      *** RESERVED***
867
868          DW 0681 0000          ; CYLINDERS
869          DB 0683 00           ; HEADS
870          DW 0684 0000          ;
871          DW 0686 0000          ; WRITE PRE-COMPENSATION CYL
872          DB 0688 00           ;
873          DB 0689 00           ; CONTROL BYTE
874          DB 068A 00 00 00      ;
875          DW 068D 0000          ; LANDING ZONE
876          DB 068F 00           ; SECTORS/TRACK
877          DB 0690 00           ;
878
879          |----- DRIVE TYPE 42      *** RESERVED***
880
881          DW 0691 0000          ; CYLINDERS
882          DB 0693 00           ; HEADS
883          DW 0694 0000          ;
884          DW 0696 0000          ; WRITE PRE-COMPENSATION CYL
885          DB 0698 00           ;
886          DB 0699 00           ; CONTROL BYTE
887          DB 069A 00 00 00      ;
888          DW 069D 0000          ; LANDING ZONE
889          DB 069F 00           ; SECTORS/TRACK
890          DB 06A0 00           ;
891
892          |----- DRIVE TYPE 43      *** RESERVED***
893
894          DW 06A1 0000          ; CYLINDERS
895          DB 06A3 00           ; HEADS
896          DW 06A4 0000          ;
897          DW 06A6 0000          ; WRITE PRE-COMPENSATION CYL
898          DB 06A8 00           ;
899          DB 06A9 00           ; CONTROL BYTE
900          DB 06AA 00 00 00      ;
901          DW 06AD 0000          ; LANDING ZONE
902          DB 06AF 00           ; SECTORS/TRACK
903          DB 06B0 00           ;
904
905          |----- DRIVE TYPE 44      *** RESERVED***
906
907          DW 06B1 0000          ; CYLINDERS
908          DB 06B3 00           ; HEADS
909          DW 06B4 0000          ;
910          DW 06B6 0000          ; WRITE PRE-COMPENSATION CYL
911          DB 06B8 00           ;
912          DB 06B9 00           ; CONTROL BYTE
913          DB 06BA 00 00 00      ;
914          DW 06BD 0000          ; LANDING ZONE
915          DB 06BF 00           ; SECTORS/TRACK
916          DB 06C0 00           ;
917
918          |----- DRIVE TYPE 45      *** RESERVED***
919
920          DW 06C1 0000          ; CYLINDERS
921          DB 06C3 00           ; HEADS
922          DW 06C4 0000          ;
923          DW 06C6 0000          ; WRITE PRE-COMPENSATION CYL
924          DB 06C8 00           ;
925          DB 06C9 00           ; CONTROL BYTE
926          DB 06CA 00 00 00      ;
927          DW 06CD 0000          ; LANDING ZONE
928          DB 06CF 00           ; SECTORS/TRACK
929          DB 06D0 00           ;
930
931          |----- DRIVE TYPE 46      *** RESERVED***
932
933          DW 06D1 0000          ; CYLINDERS
934          DB 06D3 00           ; HEADS
935          DW 06D4 0000          ;
936          DW 06D6 0000          ; WRITE PRE-COMPENSATION CYL
937          DB 06D8 00           ;
938          DB 06D9 00           ; CONTROL BYTE
939          DB 06DA 00 00 00      ;
940          DW 06DD 0000          ; LANDING ZONE
941          DB 06DF 00           ; SECTORS/TRACK
942          DB 06E0 00           ;

```

SECTION 5

```

943 PAGE
944 |----- DRIVE TYPE 47 *** RESERVED***
945
946 04E1 0000 DW 0000D ; CYLINDERS
947 04E2 00 DB 00D ; HEADS
948 04E4 0000 DW 0 ;
949 04E6 0000 DW 0000D ; WRITE PRE-COMPENSATION CYL
950 04E8 00 DB 0 ;
951 04E9 00 DB 0 ; CONTROL BYTE
952 04EA 00 00 00 DB 0,0,0 ;
953 04ED 0000 DW 0000D ; LANDING ZONE
954 04EF 00 DB 00D ; SECTORS/TRACK
955 04F0 00 DB 0 ;
956
957
958 |----- BOOT LOADER INTERRUPT
959
960 = 06F1 IP = $
961 06F7 FC II- ORG 06F2H
962 06F2 ORG 006F2H
963 = 06F2 BOOT_STRAP EQU $
964 06F2 E9 0000 E JMP BOOT_STRAP_1 ; VECTOR ON TO MOVED BOOT CODE
965
966
967 06F5 CONF_TBL:
968 06F5 0008 DW CONF_E-CONF_TBL-2 ; USE INT 15 H AH: 00H
969 06F7 FC DW MODEL_BYTE ; CONFIGURATION TABLE FOR THIS SYSTEM
970 06F8 02 DB SUB_MODEL_BYTE ; LENGTH OF FOLLOWING TABLE
971 06F9 00 DB BIOS_LEVEL ; SYSTEM MODEL BYTE
972 06FA 70 DB 0110000B ; SYSTEM SUB_MODEL_TYPE_BYTE
973 ; BIOS REVISION LEVEL
974 ; 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 FC II- ORG 0E729H
988 072B 0300 A1 DW 1047 ; 110 BAUD ; TABLE OF VALUES
989 072D 0180 DW 768 ; FOR INITIALIZATION
990 072F 00C0 DW 384 ; 300
991 0731 0040 DW 192 ; 600
992 0733 0030 DW 96 ; 1200
993 0735 0018 DW 48 ; 2400
994 0737 000C DW 24 ; 4800
995 0739 0006 DW 12 ; 9600
996
997 |----- RS232
998
999 0739 FC II- ORG 0E739H
1000 = 0739 RS232_IO EQU $
1001 0739 E9 0000 E JMP RS232_IO_1 ; VECTOR ON TO MOVED RS232 CODE
1002
1003 |----- KEYBOARD
1004
1005 II- ORG 0E82EH
1006 082E ORG 0082EH
1007 = 082E KEYBOARD_IO EQU $
1008 082E E9 0000 E JMP KEYBOARD_IO_1 ; VECTOR ON TO MOVED KEYBOARD CODE
1009
1010 |----- KEY IDENTIFICATION SCAN TABLES
1011
1012 II- ORG 0E87EH
1013 087E ORG 0087EH
1014 087E
1015 |----- TABLE OF SHIFT KEYS AND MASK VALUES
1016
1017 087E KEY_TABLE:
1018 087E 52 K6 LABEL BYTE
1019 087F 3A 45 44 38 1D DB INS_KEY ; INSERT KEY
1020 0880 2A 36 DB CAPS_KEY,NUM_KEY,SCROLL_KEY,ALT_KEY,CTL_KEY
1021 = 0008 K6L EQU $-K6 DB LEFT_KEY,RIGHT_KEY
1022
1023 |----- MASK_TABLE
1024
1025 0886 K7 LABEL BYTE
1026 0887 40 20 10 08 04 DB INS_SHIFT ; INSERT MODE SHIFT
1027 088C 02 01 DB CAPS_SHIFT,NUM_SHIFT,SCROLL_SHIFT,ALT_SHIFT,CTL_SHIFT
1028 088E DB LEFT_SHIFT,RIGHT_SHIFT
1029
1030 |----- TABLES FOR CTRL CASE
1031
1032 088E K8 LABEL BYTE
1033 088E 1B FF 00 FF FF FF DW 27,-1,00,-1,-1,-1 ; Eee, 1, 2, 3, 4, 5
1034 0891 1E FF FF FF FF IF DW 30,-1,-1,-1,-1,31 ; 6, 7, 8, 9, 0, -
1035 089A FF 7F 94 11 17 05 DW -1,127,148,17,23,6 ; a, Bkap, Tab, Q, W, E
1036 08A0 12 14 19 15 09 0F DW 18,20,25,21,09,16 ; R, T, F, U, I, O
1037 08A6 10 1B 1D 0A FF 01 DW 16,27,29,10,-1,01 ; P, [, ], Enter, Ctrl, A
1038 08AC 13 04 06 07 08 0A DW 19,04,06,07,08,10 ; S, D, F, G, H, J
1039 08B2 08 0C FF FF FF FF DW 11,12,-1,-1,-1,-1 ; K, L, ; ', ', LShift
1040 08B8 1C 1A 18 03 16 02 DW 28,26,24,09,22,02 ; Backash, Z, X, C, V, B
1041 08C4 96 FF 20 FF DW 14,13,-1,-1,-1,-1 ; N, M, , , / RShift
1042 08C8 5E 5F 60 61 62 63 DW 160,-1,-1,-1,-1 ; , Alt, Space, CL
1043 08CC 5E 5F 60 61 62 63 DW 94,96,96,97,98,99 ; F1 - F6
1044 08CE 64 65 66 67 FF FF DW 100,101,102,103,-1,-1 ; F7 - F10, NL, SL
1045 08D4 77 8D 84 8E 73 8F DW 119,141,132,142,115,143 ; Home, Up, PgUp, -, Left, Pad5
1046 08DA 74 90 75 91 76 92 DW 116,144,117,146,118,148 ; Right, *, End, Down, PgDn, Ins
1047 08E0 93 FF FF FF 89 8A DW 147,-1,-1,-1,137,138 ; Del, SysReq, Underf, Wt, F11, F12

```

```

1048                                     PAGE
1049                                     ;----- TABLES FOR LOWER CASE -----
1050
1051 08E6                                     K10 LABEL BYTE
1052 08E6 1B 31 32 33 34 35                DB 27,'12345'
1053 08E6 36 37 38 39 30 2D                DB '67890-'
1054 08F2 3D 08 09 71 77 65                DB '=,08,09,'qwe'
1055 08F8 72 74 79 75 69 6F                DB 'rtyui0'
1056 08FE 70 5B 50 0D FF 61                DB 'p[]',ODH,-1,'a'          ; LETTERS, Return, Ctrl
1057 0904 73 64 66 67 68 6A                DB 'sd fghj'
1058 090A 68 6C 3B 27 60 FF                DB 'kl;',126,-1             ; LETTERS, L Shift
1059 0910 5C 7A 78 63 76 62                DB '2;',230VB'
1060 0916 6E 6D 2C 2E 2F                DB 'nm,/'
1061 091B FF 2A FF 20 FF                DB -1,'*',-1,' ',-1        ; R Shift,*, Alt, Space, CL
1062
1063                                     ;----- LC TABLE SCAN
1064 0920 3B 3C 3D 3E 3F                DB 59,60,61,62,63          ; BASE STATE OF F1 - F10
1065 0925 40 41 42 43 44                DB 64,65,66,67,68        ; NL, SL
1066 092A FF FF
1067
1068                                     ;----- KEYPAD TABLE
1069 092C                                     K15 LABEL BYTE
1070 092C 47 48 49 FF 4B FF                DB 71,72,73,-1,75,-1      ; BASE STATE OF KEYPAD KEYS
1071 0932 4D FF 4F 50 51 52                DB 77,-1,79,80,81,82
1072 0938 53                                DB 83
1073 0939 FF FF 5C 85 86                DB -1,-1,92,133,134       ; SysRq, Underf, WT, F11, F12
1074
1075                                     ;----- KEYBOARD INTERRUPT
1076
1077                                     ;-
1078 0987                                     ORG 0E987H
1079 = 0987                                 EQU $
1080 0987 E9 0000 E                         JMP KB_INT_1               ; VECTOR ON TO MOVED KEYBOARD HANDLER
1081
1082                                     ;----- TABLES FOR UPPER CASE -----
1083
1084 098A                                     K11 LABEL BYTE
1085 098A 1B 21 40 23 24 25                DB 27,'1@#%&'
1086 0990 5E 26 2A 28 29 5F                DB '94,'4*()&'
1087 0996 2B 08 00 51 57 45                DB '+,08,00,'QWE'
1088 099C 52 54 59 55 49 4F                DB 'RTYUI0'
1089 09A2 50 7B 7D 0D FF 41                DB 'P[]',ODH,-1,'A'          ; LETTERS, Return, Ctrl
1090 09A8 53 44 46 47 48 4A                DB 'SDFGHJ'
1091 09AE 4B 4C 3A 22 7E FF                DB 'KL;',126,-1             ; LETTERS, L Shift
1092 09B4 7C 5A 58 43 56 42                DB '2;',230VB'
1093 09BA 4E 4D 3C 3E 3F                DB 'NM<?>'
1094 09BF FF 2A FF 20 FF                DB -1,'*',-1,' ',-1        ; R Shift,*, Alt, Space, CL
1095
1096                                     ;----- LC TABLE SCAN
1097 09C4                                     K12 LABEL BYTE
1098 09C4 54 55 56 57 58                DB 84,85,86,87,88          ; SHIFTED STATE OF F1 - F10
1099 09C9 59 5A 5B 5C 5D                DB 89,90,91,92,93        ; NL, SL
1100 09CE FF FF
1101
1102                                     ;----- NUM STATE TABLE
1103 09D0                                     K14 LABEL BYTE
1104 09D0 37 38 39 2D 34 35                DB '789-456+1230.'         ; NUMLOCK STATE OF KEYPAD KEYS
1105 36 2B 31 32 33 30
1106 2E
1107 09DD FF FF 7C 87 88                DB -1,-1,124,135,136       ; SysRq, Underf, WT, F11, F12
  
```

SECTION 5

```

1108 PAGE
1109 I----- DISKETTE I/O
1110
1111
1112 0C59 ORG 0EC59H
1113 = 0C59 DISKETTE_IO EQU $
1114 0C59 E9 0000 E JMP DISKETTE_IO_1 ; VECTOR ON TO MOVED DISKETTE CODE
1115
1116 I----- DISKETTE INTERRUPT
1117
1118
1119 0F57 ORG 0EF57H
1120 = 0F57 DISK_INT EQU $
1121 0F57 E9 0000 E JMP DISK_INT_1 ; VECTOR ON TO MOVED DISKETTE HANDLER
1122
1123 I----- DISKETTE PARAMETERS
1124
1125 0FC7 ORG 0EFC7H
1126 0FC7 ORG 0DFC7H
1127
1128 I-----
1129 | DISK_BASE |
1130 | THIS IS THE SET OF PARAMETERS REQUIRED FOR |
1131 | DISKETTE OPERATION. THEY ARE POINTED AT BY THE |
1132 | DATA VARIABLE @DISK_POINTER. TO MODIFY THE PARAMETERS, |
1133 | BUILD ANOTHER PARAMETER BLOCK AND POINT AT IT |
1134 |-----|
1135
1136 0FC7 DISK_BASE LABEL BYTE
1137
1138 0FC7 0F DB 11011111B ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
1139 0FC8 02 DB 2 ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
1140 0FC9 26 DB MOTOR_WAIT ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
1141 0FCA 02 DB 2 ; 512 BYTES/SECTOR
1142 0FCB 0F DB 15 ; EOT ( LAST SECTOR ON TRACK)
1143 0FCC 1B DB 01BH ; GAP LENGTH
1144 0FCD FF DB 0FFH ; DTL
1145 0FCE 54 DB 054H ; GAP LENGTH FOR FORMAT
1146 0FCF F6 DB 0F6H ; FILL BYTE FOR FORMAT
1147 0FD0 0F DB 15 ; HEAD SETTLE TIME (MILLISECONDS)
1148 0FD1 06 DB 6 ; MOTOR START TIME (1/8 SECONDS)
1149
1150 I----- PRINTER I/O
1151
1152 0FD2 ORG 0EFD2H
1153 0FD2 ORG 0DFD2H
1154 = 0FD2 PRINTER_IO EQU $
1155 0FD2 E9 0000 E JMP PRINTER_IO_1 ; VECTOR ON TO MOVED PRINTER CODE
1156
1157 I----- FOR POSSIBLE COMPATIBILITY ENTRY POINTS
1158
1159 1045 ORG 0F045H
1160 1045 ORG 01045H
1161 1045 ASSUME CS:CODE,DS:DATA
1162
1163 EXTRN SET_MODE:NEAR
1164 EXTRN SET_CTYPE:NEAR
1165 EXTRN SET_CPOS:NEAR
1166 EXTRN READ_CURSOR:NEAR
1167 EXTRN READ_LPEN:NEAR
1168 EXTRN ACT_DISP_PAGE:NEAR
1169 EXTRN SCROLL_UP:NEAR
1170 EXTRN SCROLL_DOWN:NEAR
1171 EXTRN READ_AC_CURRENT:NEAR
1172 EXTRN WRITE_AC_CURRENT:NEAR
1173 EXTRN WRITE_C_CURRENT:NEAR
1174 EXTRN SET_COLOR:NEAR
1175 EXTRN WRITE_DOT:NEAR
1176 EXTRN READ_DOT:NEAR
1177 EXTRN WRITE_TTY:NEAR
1178 EXTRN VIDEO_STATE:NEAR
1179
1180 1045 0000 E MI DW OFFSET SET_MODE ; TABLE OF ROUTINES WITHIN VIDEO I/O
1181 1047 0000 E DW OFFSET SET_CTYPE ; EXIT STACK VALUES MAY BE
1182 1049 0000 E DW OFFSET SET_CPOS ; DIFFERENT DEPENDING ON THE
1183 104B 0000 E DW OFFSET READ_CURSOR ; SYSTEM AND MODEL
1184 104D 0000 E DW OFFSET READ_LPEN
1185 104F 0000 E DW OFFSET ACT_DISP_PAGE
1186 1051 0000 E DW OFFSET SCROLL_UP
1187 1053 0000 E DW OFFSET SCROLL_DOWN
1188 1055 0000 E DW OFFSET READ_AC_CURRENT
1189 1057 0000 E DW OFFSET WRITE_AC_CURRENT
1190 1059 0000 E DW OFFSET WRITE_C_CURRENT
1191 105B 0000 E DW OFFSET SET_COLOR
1192 105D 0000 E DW OFFSET WRITE_DOT
1193 105F 0000 E DW OFFSET READ_DOT
1194 1061 0000 E DW OFFSET WRITE_TTY
1195 1063 0000 E DW OFFSET VIDEO_STATE
1196 = 0020 MIL EQU $-MI
1197
1198
1199 1065 ORG 0F065H
1200 = 1065 ORG 01065H
1201 1065 E9 0000 E VIDEO_IO JMP VIDEO_IO_1 ; VECTOR ON TO MOVED VIDEO CODE
1202
1203 I----- VIDEO PARAMETERS --- INIT_TABLE
1204
1205 10A4 ORG 0F0A4H
1206 10A4 ORG 010A4H
1207
1208 10A4 VIDEO_PARMS LABEL BYTE
1209 10A4 38 28 2D 0A 1F 06 DB 38H,28H,2DH,0AH,1FH,6,19H ; SET UP FOR 40X25
1210 10A5 19 DB 1CH,2,7,6,7
1211 10A6 1C 02 07 06 07 DB 0,0,0,0
1212 10A8 00 00 00 00 DB $-VIDEO_PARMS
1213 = 0010 M4 EQU $-VIDEO_PARMS
1214
1215 10B4 71 50 5A 0A 1F 06 DB 71H,50H,5AH,0AH,1FH,6,19H ; SET UP FOR 80X25
1216 10B5 19 DB 1CH,2,7,6,7
1217 10B6 1C 02 07 06 07 DB 0,0,0,0
1218 10C0 00 00 00 00 DB 38H,28H,2DH,0AH,7FH,6,64H ; SET UP FOR GRAPHICS
1219
1220 10C4 38 28 2D 0A 7F 06 DB
1221 64

```


1336	1B6E	00 00 00 00 00 00	DB	000H,000H,000H,000H,000H,000H,000H,000H ; D_20	SPACE
1337		00 00			
1338	1B76	30 78 78 30 30 00	DB	030H,078H,078H,030H,030H,000H,030H,000H ; D_21	!
1339		30 00			EXCLAMATION
1340	1B7E	6C 6C 6C 00 00 00	DB	06CH,06CH,06CH,000H,000H,000H,000H,000H ; D_22	"
1341		00 00			QUOTATION
1342	1B86	6C 6C 6C FE 6C FE 6C	DB	06CH,06CH,0F6H,06CH,0F6H,06CH,06CH,000H ; D_23	#
1343		6C 00			LB.
1344	1B8E	30 7C 00 78 0C F8	DB	030H,07CH,0C0H,078H,00CH,0F8H,030H,000H ; D_24	\$
1345		30 00			DOLLAR SIGN
1346	1B96	00 C6 CC 18 30 66	DB	000H,0C6H,0CCH,018H,030H,066H,0C6H,000H ; D_25	%
1347		C6 00			PERCENT
1348	1B9E	38 6C 38 76 DC CC	DB	038H,06CH,038H,076H,0DCH,0CCH,076H,000H ; D_26	&
1349		76 00			AMPERSAND
1350	1BA6	60 60 C0 00 00 00	DB	060H,060H,0C0H,000H,000H,000H,000H,000H ; D_27	'
1351		00 00			APOSTROPHE
1352	1BAE	18 30 60 60 60 30	DB	018H,030H,060H,060H,060H,030H,018H,000H ; D_28	(
1353		18 00			L. PARENTHESIS
1354	1BB6	60 30 18 18 18 30	DB	060H,030H,018H,018H,018H,030H,060H,000H ; D_29)
1355		60 00			R. PARENTHESIS
1356	1BBE	00 66 3C FF 3C 66	DB	000H,066H,03CH,0FFH,03CH,066H,000H,000H ; D_2A	*
1357		00 00			ASTERISK
1358	1BC6	00 30 30 FC 30 30	DB	000H,030H,030H,0FCH,030H,030H,000H,000H ; D_2B	+
1359		00 00			PLUS
1360	1BCE	00 00 00 00 00 30	DB	000H,000H,000H,000H,000H,030H,030H,060H ; D_2C	,
1361		30 60			COMMA
1362	1BD6	00 00 00 FC 00 00	DB	000H,000H,000H,0FCH,000H,000H,000H,000H ; D_2D	-
1363		00 00			DASH
1364	1BDE	00 00 00 00 00 30	DB	000H,000H,000H,000H,000H,030H,030H,000H ; D_2E	.
1365		30 00			PERIOD
1366	1BE6	06 0C 18 30 60 C0	DB	066H,00CH,018H,030H,060H,0C0H,060H,000H ; D_2F	/
1367		60 00			SLASH
1368					
1369	1BEE	7C C6 CE DE F6 E6	DB	07CH,0C6H,0CEH,0DEH,0F6H,0E6H,07CH,000H ; D_30	0
1370		7C 00			
1371	1BF6	30 70 30 30 30 30	DB	030H,070H,030H,030H,030H,030H,0FCH,000H ; D_31	1
1372		FC 00			
1373	1BFE	78 CC 0C 38 60 CC	DB	078H,0CCH,0CCH,038H,060H,0CCH,0FCH,000H ; D_32	2
1374		FC 00			
1375	1C06	78 CC 0C 38 0C CC	DB	078H,0CCH,0CCH,038H,00CH,0CCH,078H,000H ; D_33	3
1376		78 00			
1377	1C0E	1C 3C 6C 0C FE 0C	DB	01CH,03CH,06CH,0CCH,0F6H,00CH,01EH,000H ; D_34	4
1378		15 00			
1379	1C16	FC C0 F8 0C 0C CC	DB	0FCH,0C0H,0F8H,00CH,00CH,0CCH,078H,000H ; D_35	5
1380		78 00			
1381	1C1E	38 60 C0 F8 CC CC	DB	038H,060H,0C0H,0F8H,0CCH,0CCH,078H,000H ; D_36	6
1382		78 00			
1383	1C26	FC CC 0C 18 30 30	DB	0FCH,0CCH,0CCH,018H,030H,030H,030H,000H ; D_37	7
1384		30 00			
1385	1C2E	78 CC 0C 78 CC CC	DB	078H,0CCH,0CCH,078H,0CCH,0CCH,078H,000H ; D_38	8
1386		78 00			
1387	1C36	78 CC CC 7C 0C 18	DB	078H,0CCH,0CCH,07CH,00CH,018H,070H,000H ; D_39	9
1388		70 00			
1389	1C3E	00 30 30 00 00 30	DB	000H,030H,030H,000H,000H,030H,030H,000H ; D_3A	:
1390		30 00			COLON
1391	1C46	00 30 30 00 00 30	DB	000H,030H,030H,000H,000H,030H,030H,060H ; D_3B	;
1392		30 60			SEMICOLON
1393	1CAE	18 30 60 C0 60 30	DB	018H,030H,060H,0C0H,060H,030H,018H,000H ; D_3C	<
1394		18 00			LESS THAN
1395	1C66	00 00 FC 00 00 FC	DB	000H,000H,0FCH,000H,000H,0FCH,000H,000H ; D_3D	=
1396		00 00			EQUAL
1397	1C86	60 30 18 0C 18 30	DB	060H,030H,018H,00CH,018H,030H,060H,000H ; D_3E	>
1398		60 00			GREATER THAN
1399	1C66	78 CC 0C 18 30 00	DB	078H,0CCH,0CCH,018H,030H,000H,030H,000H ; D_3F	?
1400		30 00			QUESTION MARK
1401					
1402	1C6E	7C C6 DE DE DE 00	DB	07CH,0C6H,0DEH,0DEH,0DEH,0C0H,078H,000H ; D_40	@
1403		78 00			AT
1404	1C76	30 78 CC CC FC CC	DB	030H,078H,0CCH,0CCH,0FCH,0CCH,0CCH,000H ; D_41	A
1405		CC 00			
1406	1C7E	6C 66 7C 66 66	DB	0FCH,066H,066H,07CH,066H,066H,0FCH,000H ; D_42	B
1407		FC 00			
1408	1C86	3C 66 C0 C0 C0 66	DB	03CH,066H,0C0H,0C0H,0C0H,066H,03CH,000H ; D_43	C
1409		3C 00			
1410	1C8E	F8 6C 66 66 66 6C	DB	0F8H,06CH,066H,066H,066H,06CH,0F8H,000H ; D_44	D
1411		F8 00			
1412	1C96	FE 62 68 78 68 62	DB	0F6H,062H,068H,078H,068H,062H,0F6H,000H ; D_45	E
1413		FE 00			
1414	1C9E	FE 62 68 78 68 60	DB	0F6H,062H,068H,078H,068H,060H,0F0H,000H ; D_46	F
1415		F0 00			
1416	1CA6	3C 66 C0 C0 CE 66	DB	03CH,066H,0C0H,0C0H,0CEH,066H,03EH,000H ; D_47	G
1417		3E 00			
1418	1CAE	CC CC CC FC CC CC	DB	0CCH,0CCH,0CCH,0FCH,0CCH,0CCH,0CCH,000H ; D_48	H
1419		CC 00			
1420	1CB6	78 30 30 30 30 30	DB	078H,030H,030H,030H,030H,030H,078H,000H ; D_49	I
1421		78 00			
1422	1CBE	1E 0C 0C 0C CC CC	DB	01EH,00CH,00CH,00CH,0CCH,0CCH,078H,000H ; D_4A	J
1423		78 00			
1424	1CC6	E6 66 6C 78 6C 66	DB	0E6H,066H,06CH,078H,06CH,066H,0E6H,000H ; D_4B	K
1425		E6 00			
1426	1CCE	F0 60 60 60 62 66	DB	0F0H,060H,060H,060H,062H,066H,0F6H,000H ; D_4C	L
1427		FE 00			
1428	1CD6	C6 EE FE FE D6 C6	DB	0C6H,0EEH,0F6H,0F6H,0D6H,0C6H,0C6H,000H ; D_4D	M
1429		C6 00			
1430	1CDE	C6 E6 F6 DE CE C6	DB	0C6H,0E6H,0F6H,0DEH,0CEH,0C6H,0C6H,000H ; D_4E	N
1431		C6 00			
1432	1CE6	38 6C C6 C6 C6 6C	DB	038H,06CH,0C6H,0C6H,0C6H,06CH,038H,000H ; D_4F	O
1433		38 00			
1434					
1435	1CEE	FC 66 66 7C 60 60	DB	0FCH,066H,066H,07CH,060H,060H,0F0H,000H ; D_50	P
1436		F0 00			
1437	1CF6	78 CC CC CC DC 78	DB	078H,0CCH,0CCH,0CCH,0DCH,078H,01CH,000H ; D_51	Q
1438		1C 00			
1439	1CFE	6C 66 7C 6C 66	DB	0FCH,066H,066H,07CH,06CH,066H,0E6H,000H ; D_52	R
1440		E6 00			
1441	1D06	78 CC E0 70 1C CC	DB	078H,0CCH,0E0H,070H,01CH,0CCH,078H,000H ; D_53	S
1442		78 00			
1443	1D0E	FC 84 30 30 30 30	DB	0FCH,0B4H,030H,030H,030H,030H,078H,000H ; D_54	T
1444		78 00			
1445	1D16	CC CC CC CC CC CC	DB	0CCH,0CCH,0CCH,0CCH,0CCH,0CCH,0FCH,000H ; D_55	U
1446		FC 00			
1447	1D1E	CC CC CC CC CC 78	DB	0CCH,0CCH,0CCH,0CCH,0CCH,078H,030H,000H ; D_56	V
1448		30 00			
1449	1D26	C6 C6 C6 D6 FE EE	DB	0C6H,0C6H,0C6H,0D6H,0F6H,0EEH,0C6H,000H ; D_57	W

```

1450 C6 00
1451 ID2E C6 C6 6C 3B 3B 6C DB 0C6H,0C6H,06CH,038H,038H,06CH,0C6H,000H ; D_58 X
1452 C6 00 DB
1453 ID36 CC CC 78 30 30 DB 0CCH,0CCH,0CCH,078H,030H,030H,078H,000H ; D_59 Y
1454 78 00 DB
1455 ID3E FE C6 8C 18 32 66 DB 0FEH,0C6H,08CH,018H,032H,066H,0FEH,000H ; D_5A Z
1456 FE 00 DB
1457 ID46 78 60 60 60 60 60 DB 078H,060H,060H,060H,060H,060H,078H,000H ; D_5B [ LEFT BRACKET
1458 78 00 DB
1459 ID4E C0 60 30 18 0C 06 DB 0C0H,060H,030H,018H,00CH,066H,002H,000H ; D_5C $ BACKSLASH
1460 02 00 DB
1461 ID56 78 18 18 18 18 18 DB 078H,018H,018H,018H,018H,018H,078H,000H ; D_5D ] RIGHT BRACKET
1462 78 00 DB
1463 ID5E 10 38 6C C6 00 00 DB 010H,038H,06CH,0C6H,000H,000H,000H,000H ; D_5E ¢ CIRCUMFLEX
1464 00 00 DB
1465 ID66 00 00 00 00 00 00 DB 000H,000H,000H,000H,000H,000H,0FFH ; D_5F _ UNDERSCORE
1466 00 FF DB
1467
1468 ID6E 30 30 18 00 00 00 DB 030H,030H,018H,000H,000H,000H,000H ; D_60 ' APOSTROPHE REV
1469 00 00 DB
1470 ID76 00 00 78 0C 7C CC DB 000H,000H,078H,00CH,07CH,0CCH,076H,000H ; D_61 ¢
1471 78 00 DB
1472 ID7E E0 60 60 7C 66 66 DB 0E0H,060H,060H,07CH,066H,066H,0DCH,000H ; D_62 b
1473 DC 00 DB
1474 ID86 00 00 78 CC C0 CC DB 000H,000H,078H,0CCH,0C0H,0CCH,078H,000H ; D_63 c
1475 78 00 DB
1476 ID8E 1C 0C 0C 7C CC CC DB 01CH,00CH,00CH,07CH,0CCH,0CCH,076H,000H ; D_64 d
1477 76 00 DB
1478 ID96 00 00 78 CC FC C0 DB 000H,000H,078H,0CCH,0FCH,0C0H,078H,000H ; D_65 e
1479 78 00 DB
1480 ID9E 38 6C 60 F0 60 60 DB 038H,06CH,060H,0F0H,060H,060H,0F0H,000H ; D_66 f
1481 F0 00 DB
1482 IDA6 00 00 76 CC CC 7C DB 000H,000H,076H,0CCH,0CCH,07CH,00CH,0F8H ; D_67 g
1483 0C F8 DB
1484 IDAE E0 60 6C 76 66 66 DB 0E0H,060H,06CH,076H,066H,066H,0E6H,000H ; D_68 h
1485 E6 00 DB
1486 IDB6 30 00 70 30 30 30 DB 030H,000H,070H,030H,030H,030H,078H,000H ; D_69 i
1487 78 00 DB
1488 IDBE 0C 00 0C 0C 0C CC DB 00CH,000H,00CH,00CH,00CH,0CCH,0CCH,078H ; D_6A j
1489 CC 78 DB
1490 IDC6 E0 60 66 6C 78 6C DB 0E0H,060H,066H,06CH,078H,06CH,0E6H,000H ; D_6B k
1491 E6 00 DB
1492 IDCE 70 30 30 30 30 30 DB 070H,030H,030H,030H,030H,030H,078H,000H ; D_6C l
1493 78 00 DB
1494 IDD6 00 00 CC FE FE D6 DB 000H,000H,0CCH,0FEH,0FEH,0D6H,0C6H,000H ; D_6D m
1495 C6 00 DB
1496 IDDE 00 00 F8 CC CC CC DB 000H,000H,0F8H,0CCH,0CCH,0CCH,0CCH,000H ; D_6E n
1497 CC 00 DB
1498 IDE6 00 00 78 CC CC CC DB 000H,000H,078H,0CCH,0CCH,0CCH,078H,000H ; D_6F o
1499 78 00 DB
1500
1501 IDEE 00 00 DC 66 66 7C DB 000H,000H,0DCH,066H,066H,07CH,060H,0F0H ; D_70 p
1502 60 F0 DB
1503 IDF6 00 00 76 CC CC 7C DB 000H,000H,076H,0CCH,0CCH,07CH,00CH,01EH ; D_71 q
1504 0C 1E DB
1505 IDFE 00 00 DC 76 66 60 DB 000H,000H,0DCH,076H,066H,060H,0F0H,000H ; D_72 r
1506 F0 00 DB
1507 IE06 00 00 7C C0 78 0C DB 000H,000H,07CH,0C0H,078H,00CH,0F8H,000H ; D_73 s
1508 F8 00 DB
1509 IE0E 10 30 7C 30 30 34 DB 010H,030H,07CH,030H,030H,034H,018H,000H ; D_74 t
1510 18 00 DB
1511 IE16 00 00 CC CC CC CC DB 000H,000H,0CCH,0CCH,0CCH,0CCH,076H,000H ; D_75 u
1512 76 00 DB
1513 IE1E 00 00 CC CC CC 78 DB 000H,000H,0CCH,0CCH,0CCH,078H,030H,000H ; D_76 v
1514 30 00 DB
1515 IE26 00 00 C6 D6 FE FE DB 000H,000H,0C6H,0D6H,0FEH,0FEH,06CH,000H ; D_77 w
1516 6C 00 DB
1517 IE2E 00 00 C6 6C 38 6C DB 000H,000H,0C6H,06CH,038H,06CH,0C6H,000H ; D_78 x
1518 C6 00 DB
1519 IE36 00 00 CC CC CC 7C DB 000H,000H,0CCH,0CCH,0CCH,07CH,00CH,0F8H ; D_79 y
1520 0C F8 DB
1521 IE3E 00 00 FC 98 30 64 DB 000H,000H,0FCH,098H,030H,064H,0FCH,000H ; D_7A z
1522 FC 00 DB
1523 IE46 1C 30 30 E0 30 30 DB 01CH,030H,030H,0E0H,030H,030H,01CH,000H ; D_7B { LEFT BRACE
1524 1C 00 DB
1525 IE4E 18 18 18 00 18 18 DB 018H,018H,018H,000H,018H,018H,018H,000H ; D_7C | BROKEN STROKE
1526 18 00 DB
1527 IE56 E0 30 30 1C 30 30 DB 0E0H,030H,030H,01CH,030H,030H,0E0H,000H ; D_7D } RIGHT BRACE
1528 E0 00 DB
1529 IE5E 76 DC 00 00 00 00 DB 076H,0DCH,000H,000H,000H,000H,000H ; D_7E ¤ TILDE
1530 00 00 DB
1531 IE66 00 10 38 6C C6 C6 DB 000H,010H,038H,06CH,0C6H,0C6H,0FEH,000H ; D_7F DELTA
1532 FE 00 DB
1533
1534
1535
1536
1537 IE6E
1538 = IE6E
1539 IE6E E9 0000 E
1540
1541
1542
1543
1544 IEA5
1545 = IEA5
1546 IEA5 E9 0000 E

```

```

;----- TIME OF DAY
;-- ORG 0FE6EH
ORG 01E6EH
TIME_OF_DAY EQU $
;----- TIMER INTERRUPT
;-- ORG 0FEA5H
ORG 01EA5H
TIMER_INT EQU $
; VECTOR ON TO MOVED BIOS CODE

```

SECTION 5

```

1547 PAGE
1548 |----- VECTOR TABLE
1549
1550 |
1551 |11- ORG 0FEF3H
1552 | ORG 01EF3H
1553 | LABEL WORD
1554 | VECTOR_TABLE LABEL WORD
1555 | DW OFFSET TIMER_INT
1556 | DW OFFSET KB_INT
1557 | DW OFFSET DT1
1558 | DW OFFSET D11
1559 | DW OFFSET D11
1560 | DW OFFSET D11
1561 | DW OFFSET DISK_INT
1562 | DW OFFSET D11
1563
1564 |----- SOFTWARE INTERRUPTS ( BIOS CALLS AND POINTERS )
1565 |
1566 | DW OFFSET VIDEO_IO
1567 | DW OFFSET EQUIPMENT
1568 | DW OFFSET MEMORY_SIZE_DET
1569 | DW OFFSET DISKETTE_IO
1570 | DW OFFSET RS232_IO
1571 | DW OFFSET CASSETTE_IO
1572 | DW OFFSET KEYBOARD_IO
1573 | DW OFFSET PRINTER_IO
1574 | DW 00000H
1575 | DW OFFSET BOOT_STRAP
1576 | DW OFFSET TIME_OF_DAY
1577 | DW OFFSET DUMMY_RETURN
1578 | DW OFFSET DUMMY_RETURN
1579 | DW OFFSET VIDEO_PARAMS
1580 | DW OFFSET DISK_BASE
1581 | DW 00000H
1582
1583 | SLAVE_VECTOR_TABLE LABEL WORD
1584 | DW OFFSET RTC_INT
1585 | DW OFFSET RE_DIRECT
1586 | DW OFFSET DT1
1587 | DW OFFSET D11
1588 | DW OFFSET D11
1589 | DW OFFSET INT_287
1590 | DW OFFSET D11
1591 | DW OFFSET D11
1592
1593 |----- DUMMY INTERRUPT HANDLER
1594 |
1595 |11- ORG 0FF53H
1596 | ORG 01F53H
1597 | EQU $
1598 | DUMMY_RETURN EQU $
1599 | IRET
1600
1601 |----- PRINT SCREEN
1602 |
1603 |11- ORG 0FF54H
1604 | ORG 01F54H
1605 | EQU $
1606 | PRINT_SCREEN EQU $
1607 | .LIST JUMP PRINT_SCREEN_1
1608 |
1609 |----- POWER ON RESET VECTOR
1610 |
1611 |-----
1612 |11- ORG 0FF50H
1613 | ORG 01FF50H
1614 |
1615 |----- POWER ON RESET
1616 |
1617 |
1618 | P_O_R LABEL FAR
1619 | DB 0EAH
1620 | DW OFFSET RESET
1621 | DW 0F000H
1622 |
1623 | DB '04/21/86'
1624 |
1625 | ORG 01FFE0H
1626 | DB MODEL_BYTE
1627 |
1628 | CODE EQU $
1629 | END
1630
1631

```

SECTION 6. INSTRUCTION SET

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

80286 Instruction Set

Data Transfer

MOV = move

Register to Register/Memory

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

Register/Memory to Register

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

Immediate to Register/Memory

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

Immediate to Register

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

Memory to Accumulator

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

Accumulator to Memory

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

Register/Memory to Segment Register

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

Segment Register to Register/Memory

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

PUSH = Push

Memory

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

Register

01010reg

Segment Register

000reg110

Immediate

011010s0	data	data if s = 0
----------	------	---------------

PUSHA = Push All

01100000

POP = Pop

Memory

10001111	mod000 r/m
----------	------------

Register

01011reg

Segment Register

000reg111	reg ≠ 01
-----------	----------

POPA = Pop All

01100001

XCHG = Exchange

Register/Memory with Register

1000011w	mod reg r/m
----------	-------------

Register with Accumulator

10010reg

IN = Input From

Fixed Port

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

Variable Port

1110110w

OUT = Output To

Fixed Port

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

Variable Port

1110111w

XLAT = Translate Byte to AL

11010111

LEA = Load EA to Register

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

LDS = Load Pointer to DS

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

LES = Load Pointer to ES

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

LAHF = Load AH with Flags

10011111

SAHF = Store AH with Flags

10011110

PUSHF = Push Flags

10011100

POPF = Pop Flags

10011101

Arithmetic

ADD = Add

Register/Memory with Register to Either

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

Immediate to Register Memory

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

Immediate to Accumulator

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

ADC = Add with Carry

Register/Memory with Register to Either

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

Immediate to Register/Memory

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

Immediate to Accumulator

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

INC = Increment

Register/Memory

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

Register

01000reg

SUB = Subtract

Register/Memory with Register to Either

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

Immediate from Register/Memory

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

Immediate from Accumulator

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

SBB = Subtract with Borrow

Register/Memory with Register to Either

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

Immediate to Register/Memory

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

Immediate to Accumulator

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

DEC = Decrement

Register/Memory

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

Register

01001reg

CMP = Compare

Register/Memory with Register

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

Register with Register/Memory

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

Immediate with Register/Memory

10000sw	mod111 r/m	data	data if sw = 01
---------	------------	------	-----------------

Immediate with Accumulator

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

NEG = Change Sign

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

AAA = ASCII Adjust for Add

00110111

DAA = Decimal Adjust for Add

00100111

AAS = ASCII Adjust for Subtract

00111111

DAS = Decimal Adjust for Subtract

00110111

MUL = Multiply (Unsigned)

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

IMUL = Integer Multiply (Signed)

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

IIMUL = Integer Immediate Multiply (Signed)

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

DIV = Divide (Unsigned)

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

IDIV = Integer Divide (Signed)

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

AAM = ASCII Adjust for Multiply

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

AAD = ASCII Adjust for Divide

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

CBW = Convert Byte to Word

10011000

CWD = Convert Word to Double Word

10011001

Logic

Shift/Rotate Instructions

Register/Memory by 1

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

Register/Memory by CL

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

Register/Memory by Count

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

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

AND = And

Register/Memory and Register to Either

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

Immediate to Register/Memory

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

Immediate to Accumulator

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

TEST = AND Function to Flags; No Result

Register/Memory and Register

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

Immediate Data and Register/Memory

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

Immediate to Accumulator

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

Or = Or

Register/Memory and Register to Either

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

Immediate to Register/Memory

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

Immediate to Accumulator

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

XOR = Exclusive OR

Register/Memory and Register to Either

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

Immediate to Register/Memory

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

Immediate to Accumulator

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

NOT = Invert Register/Memory

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

String Manipulation

MOVS = Move Byte Word

1010010w

CMPS = Compare Byte Word

1010011w

SCAS = Scan Byte Word

1010111w

LODS = Load Byte Word to AL/AX

1010110w

STOS = Store Byte Word from AL/AX

1010101w

INS = Input Byte from DX Port

0110110w

OUTS = Output Byte Word to DX Port

0110111w

REP/REPNE, REPZ/REPNZ = Repeat String

Repeat Move String

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

Repeat Compare String (z/Not z)

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

Repeat Scan String (z/Not z)

1111001z	1010111w
----------	----------

Repeat Load String

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

Repeat Store String

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

Repeat Input String

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

Repeat Output String

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

Control Transfer

CALL = Call

Direct Within Segment

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

Register/Memory Indirect Within Segment

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

Direct Intersegment

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

Indirect Intersegment

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

JMP = Unconditional Jump

Short/Long

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

Direct within Segment

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

Register/Memory Indirect Within Segment

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

Direct Intersegment

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

Indirect Intersegment

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

RET = Return from Call

Within Segment

11000011

Within Segment Adding Immediate to SP

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

Intersegment

11001011

Intersegment Adding Immediate to SP

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

JE/JZ = Jump on Equal/Zero

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

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

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

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

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

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

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

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

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

JP/JPE = Jump on Parity/Parity Even

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

JO = Jump on Overflow

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

JS = Jump on Sign

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

JNE/JNZ = Jump on Not Equal/Not Zero

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

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

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

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

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

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

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

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

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

JNP/JPO = Jump on Not Parity/Parity Odd

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

JNO = Jump on Not Overflow

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

JNS = Jump on Not Sign

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

LOOP = Loop CX Times

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

LOOPZ/LOOPE = Loop while Zero/Equal

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

LOOPNZ/LOOPNE = Loop while Not Zero/Not Equal

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

JCXZ = Jump on CX Zero

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

ENTER = Enter Procedure

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

LEAVE = Leave Procedure

11001001

INT = Interrupt

Type Specified

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

Type 3

11001100

INTO = Interrupt on Overflow

11001110

IRET = Interrupt Return

11001111

BOUND = Detect Value Out of Range

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

Processor Control

CLC = Clear Carry

11111000

CMC = Complement Carry

11110101

STC = Set Carry

11111001

CLD = Clear Direction

11111100

STD = Set Direction

11111101

CLI Clear Interrupt

11111010

STI = Set Interrupt

11111011

HLT = Halt

11110100

WAIT = Wait

10011011

LOCK = Bus Lock Prefix

11110000

CTS = Clear Task Switched Flag

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

ESC = Processor Extension Escape

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

Protection Control

LGDT = Load Global Descriptor Table Register

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

SGDT = Store Global Descriptor Table Register

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

LIDT = Load Interrupt Descriptor Table Register

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

SIDT = Store Interrupt Descriptor Table Register

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

LLDT = Load Local Descriptor Table Register from Register/Memory

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

SLDT = Store Local Descriptor Table Register from Register/Memory

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

LTR = Load Task Register from Register/Memory

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

STR = Store Task Register to Register/Memory

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

LMSW = Load Machine Status Word from Register/Memory

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

SMSW = Store Machine Status Word

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

LAR = Load Access Rights from Register/Memory

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

LSL = Load Segment Limit from Register/Memory

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

ARPL = Adjust Requested Privilege Level from Register/Memory

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

VERR = Verify Read Access; Register/Memory

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

VERR = Verify Write Access

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

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

If mod = 11, then r/m is treated as a reg field.
If mod = 00, then disp = 0, disp-low and disp-high are absent.
If mod = 01, then disp = disp-low sign-extended to 16 bits, disp-high is absent.
If mod = 10, then disp = disp-high:disp-low.

If r/m = 000, then EA = (BX) + (SI) + DISP
If r/m = 001, then EA = (BX) + (SI) + DISP
If r/m = 010, then EA = (BP) + (SI) + DISP
If r/m = 011, then EA = (BP) + (DI) + DISP
If r/m = 100, then EA = (SI) + DISP
If r/m = 101, then EA = (DI) + DISP
If r/m = 110, then EA = (BP) + DISP
If r/m = 111, then EA = (BX) + DISP

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

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

Segment Override Prefix

001reg001

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

2-Bit reg Field

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

3-Bit reg Field

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

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

80287 Coprocessor Instruction Set

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

Data Transfer

FLD = Load

Integer/Real Memory to ST(0)

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

Long Integer Memory to ST(0)

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

Temporary Real Memory to ST(0)

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

BCD Memory to ST(0)

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

ST(i) to ST(0)

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

FST = Store

ST(0) to Integer/Real Memory

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

ST(0) to ST(i)

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

FSTP = Store and Pop

ST(0) to Integer/Real Memory

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

ST(0) to Long Integer Memory

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

ST(0) to Temporary Real Memory

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

ST(0) to BCD Memory

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

ST(0) to ST(i)

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

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

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

Comparison

FCOM = Compare

Integer/Real Memory to ST(0)

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

ST(i) to ST(0)

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

FCOMP = Compare and Pop

Integer/Real Memory to ST(0)

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

ST(i) to ST(0)

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

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

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

FTST = Test ST(0)

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

FXAM = Examine ST(0)

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

Constants

FLDZ = Load + 0.0 into ST(0)

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

FLD1 = Load + 1.0 into ST(0)

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

FLDP1 = Load π into ST(0)

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

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

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

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

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

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

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

Arithmetic

FADD = Addition

Integer/Real Memory with ST(0)

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

ST(i) and ST(0)

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

FSUB = Subtraction

Integer/Real Memory with ST(0)

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

ST(i) and ST(0)

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

FMUL = Multiplication

Integer/Real Memory with ST(0)

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

ST(i) and ST(0)

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

FDIV = Division

Integer/Real Memory with ST(0)

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

ST(i) and ST(0)

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

FSQRT = Square Root of ST(0)

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

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

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

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

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

FRNDINT = Round ST(0) to Integer

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

FXTRACT = Extract Components of ST(0)

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

FABS = Absolute Value of ST(0)

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

FCHS = Change Sign of ST(0)

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

Transcendental

FPTAN = Partial Tangent of ST(0)

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

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

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

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

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

FYL2X = ST(1) x Log_2 [ST(0)]

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

FYL2XP1 = ST(1) x 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

Character Codes 7-3
Quick Reference 7-14



7-2 Characters, Keystrokes, and Colors

Character Codes

Value		As Characters			As Text Attributes		
					Color/Graphics Monitor Adapter		IBM Monochrome Display Adapter
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground	
00	0	Blank (Null)	Ctrl 2		Black	Black	Non-Display
01	1	☺	Ctrl A		Black	Blue	Underline
02	2	☹	Ctrl B		Black	Green	Normal
03	3	♥	Ctrl C		Black	Cyan	Normal
04	4	♦	Ctrl D		Black	Red	Normal
05	5	♣	Ctrl E		Black	Magenta	Normal
06	6	♠	Ctrl F		Black	Brown	Normal
07	7	●	Ctrl G		Black	Light Grey	Normal
08	8	•	Ctrl H, Backspace, Shift Backspace		Black	Dark Grey	Non-Display
09	9	○	Ctrl I		Black	Light Blue	High Intensity Underline
0A	10	◐	Ctrl J, Ctrl ←		Black	Light Green	High Intensity
0B	11	♂	Ctrl K		Black	Light Cyan	High Intensity
0C	12	♀	Ctrl L		Black	Light Red	High Intensity
0D	13	♪	Ctrl M, ←, Shift ←		Black	Light Magenta	High Intensity
0E	14	♫	Ctrl N		Black	Yellow	High Intensity
0F	15	⚙	Ctrl O		Black	White	High intensity
10	16	▶	Ctrl P		Blue	Black	Normal
11	17	◀	Ctrl Q		Blue	Blue	Underline
12	18	↕	Ctrl R		Blue	Green	Normal
13	19	!!	Ctrl S		Blue	Cyan	Normal
14	20	¶	Ctrl T		Blue	Red	Normal
15	21	§	Ctrl U		Blue	Magenta	Normal
16	22	■	Ctrl V		Blue	Brown	Normal
17	23	↕	Ctrl W		Blue	Light Grey	Normal

Value		As Characters			As Text Attributes		
					Color/Graphics Monitor Adapter		IBM Monochrome Display Adapter
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground	
18	24	↑	Ctrl X		Blue	Dark Grey	High Intensity
19	25	↓	Ctrl Y		Blue	Light Blue	High Intensity Underline
1A	26	→	Ctrl Z		Blue	Light Green	High Intensity
1B	27	←	Ctrl [, Esc, Shift Esc, Ctrl Esc		Blue	Light Cyan	High Intensity
1C	28	⌞	Ctrl \		Blue	Light Red	High Intensity
1D	29	↔	Ctrl]		Blue	Light Magenta	High Intensity
1E	30	▲	Ctrl 6		Blue	Yellow	High Intensity
1F	31	▼	Ctrl —		Blue	White	High Intensity
20	32	Blank Space	Space Bar, Shift, Space, Ctrl Space, Alt Space		Green	Black	Normal
21	33	!	!	Shift	Green	Blue	Underline
22	34	”	”	Shift	Green	Green	Normal
23	35	#	#	Shift	Green	Cyan	Normal
24	36	\$	\$	Shift	Green	Red	Normal
25	37	%	%	Shift	Green	Magenta	Normal
26	38	&	&	Shift	Green	Brown	Normal
27	39	'	'		Green	Light Grey	Normal
28	40	((Shift	Green	Dark Grey	High Intensity
29	41))	Shift	Green	Light Blue	High Intensity Underline
2A	42	*	*	Note 1	Green	Light Green	High Intensity
2B	43	+	+	Shift	Green	Light Cyan	High Intensity
2C	44	,	,		Green	Light Red	High Intensity
2D	45	-	-		Green	Light Magenta	High Intensity
2E	46	.	.	Note 2	Green	Yellow	High Intensity

7-4 Characters, Keystrokes, and Colors

Value		As Characters			As Text Attributes		
					Color/Graphics Monitor Adapter		IBM Monochrome Display Adapter
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground	
2F	47	/	/		Green	White	High Intensity
30	48	0	0	Note 3	Cyan	Black	Normal
31	49	1	1	Note 3	Cyan	Blue	Underline
32	50	2	2	Note 3	Cyan	Green	Normal
33	51	3	3	Note 3	Cyan	Cyan	Normal
34	52	4	4	Note 3	Cyan	Red	Normal
35	53	5	5	Note 3	Cyan	Magenta	Normal
36	54	6	6	Note 3	Cyan	Brown	Normal
37	55	7	7	Note 3	Cyan	Light Grey	Normal
38	56	8	8	Note 3	Cyan	Dark Grey	High Intensity
39	57	9	9	Note 3	Cyan	Light Blue	High Intensity Underline
3A	58	:	:	Shift	Cyan	Light Green	High Intensity
3B	59	;	;		Cyan	Light Cyan	High Intensity
3C	60	<	<	Shift	Cyan	Light Red	High Intensity
3D	61	=	=		Cyan	Light Magenta	High Intensity
3E	62	>	>	Shift	Cyan	Yellow	High Intensity
3F	63	?	?	Shift	Cyan	White	High Intensity
40	64	@	@	Shift	Red	Black	Normal
41	65	A	A	Note 4	Red	Blue	Underline
42	66	B	B	Note 4	Red	Green	Normal
43	67	C	C	Note 4	Red	Cyan	Normal
44	68	D	D	Note 4	Red	Red	Normal
45	69	E	E	Note 4	Red	Magenta	Normal
46	70	F	F	Note 4	Red	Brown	Normal
47	71	G	G	Note 4	Red	Light Grey	Normal
48	72	H	H	Note 4	Red	Dark Grey	High Intensity
49	73	I	I	Note 4	Red	Light Blue	High Intensity Underline
4A	74	J	J	Note 4	Red	Light Green	High Intensity

Value		As Characters			As Text Attributes		
					Color/Graphics Monitor Adapter		IBM Monochrome Display Adapter
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground	
4B	75	K	K	Note 4	Red	Light Cyan	High Intensity
4C	76	L	L	Note 4	Red	Light Red	High Intensity
4D	77	M	M	Note 4	Red	Light Magenta	High Intensity
4E	78	N	N	Note 4	Red	Yellow	High Intensity
4F	79	O	O	Note 4	Red	White	High Intensity
50	80	P	P	Note 4	Magenta	Black	Normal
51	81	Q	Q	Note 4	Magenta	Blue	Underline
52	82	R	R	Note 4	Magenta	Green	Normal
53	83	S	S	Note 4	Magenta	Cyan	Normal
54	84	T	T	Note 4	Magenta	Red	Normal
55	85	U	U	Note 4	Magenta	Magenta	Normal
56	86	V	V	Note 4	Magenta	Brown	Normal
57	87	W	W	Note 4	Magenta	Light Grey	Normal
58	88	X	X	Note 4	Magenta	Dark Grey	High Intensity
59	89	Y	Y	Note 4	Magenta	Light Blue	High Intensity Underline
5A	90	Z	Z	Note 4	Magenta	Light Green	High Intensity
5B	91	[[Magenta	Light Cyan	High Intensity
5C	92	\	\		Magenta	Light Red	High Intensity
5D	93]]		Magenta	Light Magenta	High Intensity
5E	94	^	^	Shift	Magenta	Yellow	High Intensity
5F	95	-	-	Shift	Magenta	White	High Intensity
60	96	'	'		Brown	Black	Normal
61	97	a	a	Note 5	Brown	Blue	Underline
62	98	b	b	Note 5	Brown	Green	Normal
63	99	c	c	Note 5	Brown	Cyan	Normal
64	100	d	d	Note 5	Brown	Red	Normal
65	101	e	e	Note 5	Brown	Magenta	Normal
66	102	f	f	Note 5	Brown	Brown	Normal

7-6 Characters, Keystrokes, and Colors

Value		As Characters			As Text Attributes		
					Color/Graphics Monitor Adapter		IBM Monochrome Display Adapter
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground	
67	103	g	g	Note 5	Brown	Light Grey	Normal
68	104	h	h	Note 5	Brown	Dark Grey	High Intensity
69	105	i	i	Note 5	Brown	Light Blue	High Intensity Underline
6A	106	j	j	Note 5	Brown	Light Green	High Intensity
6B	107	k	k	Note 5	Brown	Light Cyan	High Intensity
6C	108	l	l	Note 5	Brown	Light Red	High Intensity
6D	109	m	m	Note 5	Brown	Light Magenta	High Intensity
6E	110	n	n	Note 5	Brown	Yellow	High Intensity
6F	111	o	o	Note 5	Brown	White	High Intensity
70	112	p	p	Note 5	Light Grey	Black	Reverse Video
71	113	q	q	Note 5	Light Grey	Blue	Underline
72	114	r	r	Note 5	Light Grey	Green	Normal
73	115	s	s	Note 5	Light Grey	Cyan	Normal
74	116	t	t	Note 5	Light Grey	Red	Normal
75	117	u	u	Note 5	Light Grey	Magenta	Normal
76	118	v	v	Note 5	Light Grey	Brown	Normal
77	119	w	w	Note 5	Light Grey	Light Grey	Normal
78	120	x	x	Note 5	Light Grey	Dark Grey	Reverse Video
79	121	y	y	Note 5	Light Grey	Light Blue	High Intensity Underline
7A	122	z	z	Note 5	Light Grey	Light Green	High Intensity
7B	123	{	{	Shift	Light Grey	Light Cyan	High Intensity
7C	124			Shift	Light Grey	Light Red	High Intensity
7D	125	}	}	Shift	Light Grey	Light Magenta	High Intensity
7E	126	~	~	Shift	Light Grey	Yellow	High Intensity
7F	127	Δ	Ctrl ←		Light Grey	White	High Intensity

Value		As Characters			As Text Attributes		
					Color/Graphics Monitor Adapter		IBM Monochrome Display Adapter
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground	
* * * * 80 to FF Hex are Flashing in both Color & IBM Monochrome * * * *							
80	128	Ç	Alt 128	Note 6	Black	Black	Non-Display
81	129	ü	Alt 129	Note 6	Black	Blue	Underline
82	130	é	Alt 130	Note 6	Black	Green	Normal
83	131	â	Alt 131	Note 6	Black	Cyan	Normal
84	132	ã	Alt 132	Note 6	Black	Red	Normal
85	133	à	Alt 133	Note 6	Black	Magenta	Normal
86	134	â	Alt 134	Note 6	Black	Brown	Normal
87	135	ç	Alt 135	Note 6	Black	Light Grey	Normal
88	136	ê	Alt 136	Note 6	Black	Dark Grey	Non-Display
89	137	ë	Alt 137	Note 6	Black	Light Blue	High Intensity Underline
8A	138	è	Alt 138	Note 6	Black	Light Green	High Intensity
8B	139	ï	Alt 139	Note 6	Black	Light Cyan	High Intensity
8C	140	î	Alt 140	Note 6	Black	Light Red	High Intensity
8D	141	ì	Alt 141	Note 6	Black	Light Magenta	High Intensity
8E	142	Ä	Alt 142	Note 6	Black	Yellow	High Intensity
8F	143	Å	Alt 143	Note 6	Black	White	High Intensity
90	144	É	Alt 144	Note 6	Blue	Black	Normal
91	145	æ	Alt 145	Note 6	Blue	Blue	Underline
92	146	Æ	Alt 146	Note 6	Blue	Green	Normal
93	147	ô	Alt 147	Note 6	Blue	Cyan	Normal
94	148	ö	Alt 148	Note 6	Blue	Red	Normal
95	149	ò	Alt 149	Note 6	Blue	Magenta	Normal
96	150	û	Alt 150	Note 6	Blue	Brown	Normal
97	151	ù	Alt 151	Note 6	Blue	Light Grey	Normal
98	152	ÿ	Alt 152	Note 6	Blue	Dark Grey	High Intensity
99	153	Ö	Alt 153	Note 6	Blue	Light Blue	High Intensity Underline
9A	154	Ü	Alt 154	Note 6	Blue	Light Green	High Intensity

7-8 Characters, Keystrokes, and Colors

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

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

7-10 Characters, Keystrokes, and Colors

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

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

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	å	û	à				μ	÷
7	7	ç	ù	ó				γ	≈
8	8	ê	ÿ	¿				Φ	°
9	9	ë	Ö	⌋				Θ	•
10	A	è	Ü	⌋				Ω	•
11	B	ï	¢	½				δ	√
12	C	î	£	¼				∞	n
13	D	ì	¥	¡				φ	²
14	E	Ä	℞	«				€	■
15	F	Å	f	»				∩	BLANK 'FF'

Notes:

—

—

—

SECTION 8. IBM PERSONAL COMPUTER COMPATIBILITY

Hardware Considerations	8-3
System Board	8-3
Fixed Disk Drive	8-5
Diskette Drive Compatibility	8-5
Copy Protection	8-5
Bypassing BIOS	8-6
Diskette Drive Controls	8-6
Write Current Control	8-6
Application Guidelines	8-7
High-Level Language Considerations	8-7
Assembler Language Programming Considerations ..	8-8
Multitasking Provisions	8-15
Interfaces	8-15
Classes	8-17
Time-Outs	8-18
Machine-Sensitive Code	8-19

SECTION 8

|

)

)

)

8-2 Compatibility

This section describes the differences among the members of the IBM Personal Computer family. It also contains information necessary to design hardware and programs that will be compatible with all members of the IBM Personal Computer family.

Hardware Considerations

To design compatible hardware or programs, you must consider hardware differences among the IBM Personal Computers. The following are hardware features of the IBM Personal Computer XT Model 286 that are not supported by all of the IBM Personal Computer family.

System Board

The IBM Personal Computer XT Model 286 system board uses an Intel 80286-6 Microprocessor. This microprocessor uses the 80287 Math Coprocessor and is generally compatible with the Intel 8088 Microprocessor used in other IBM Personal Computers.

The following table identifies the microprocessor and describes the I/O channel used with each type of IBM Personal Computer.

System Name	System Unit Microprocessor	I/O Channel Description
Personal Computer	8088	5 62-Pin
PCjr	8088	Not Compatible
Personal Computer XT	8088	8 62-Pin
Portable Personal Computer	8088	8 62-Pin
Personal Computer XT Model 286	80286-6	3 62-pin 5 98-Pin (62 Pin + 36 Pin)
Personal Computer AT	80286(-6 or -8)	2 62-pin 6 98-Pin (62 Pin + 36 Pin)

System Hardware Identification Chart

The faster processing capability of the 80286, compared to the 8088, creates special programming considerations, which are discussed later in this section under "Application Guidelines."

Adapters designed to use all 98 pins on the 98-pin connectors are not compatible with all members of the IBM Personal Computer family. Some options not supported by the IBM Personal Computer XT Model 286 are:

- Expansion Unit
- AT 128KB Memory Expansion
- AT 512KB Memory Expansion
- AT 128/640KB Memory Expansion
- AT Fixed Disk And Diskette Drive Adapter
- 256KB Memory Expansion
- 64/256KB Memory Expansion
- 64KB Memory Module Kit
- Full-high diskette drives
- AT 30MB Fixed Disk Drive
- AT 20MB Fixed Disk Drive
- 10MB Fixed Disk Drive
- AT Prototype Card
- Diskette Drive Adapter
- Fixed Disk Adapter
- 8087 Math Coprocessor
- Professional Graphics Adapter and Display
- Game Control Adapter
- Color Printer
- Other keyboards

On the I/O channel:

- The system clock signal should be used only for synchronization and not for applications requiring a fixed frequency.
- The 14.31818-MHz oscillator is not synchronous with the system clock.
- The ALE signal remains high during DMA cycles.
- Pin B04 supports IRQ 9.

Fixed Disk Drive

Reading from and writing to this drive is initiated in the same way as with other IBM Personal Computers; however, the Fixed Disk and Diskette Drive Adapter may be addressed from different BIOS locations.

Diskette Drive Compatibility

The following chart shows the read, write, and format capabilities for each of the diskette drives used by IBM Personal Computers.

Diskette Drive Name	160/180K Mode	320/360K Mode	1.2M Mode	720K Mode
5-1/4 In. Diskette Drive				
Type 1	R W F	---	---	---
Type 2	R W F	R W F	---	---
Type 3	R W F	R W F	---	---
Slimline Diskette Drive	R W F	R W F	---	---
Double Sided Diskette Drive	R W F	R W F	---	---
High Capacity Diskette Drive	R W*	R W*	R W F	---
3-1/2 In.- 720K Drive	---	---	---	R W F

R=Read W=Write F=Format W*= If a diskette is formatted in either 160/180K mode or 320/360K mode and written on by a High Capacity Drive, that diskette may be read by only a High Capacity Drive.

Diskette Drive Compatibility Chart

Note: Diskettes designed for the 1.2M mode may not be used in either a 160/180K or a 320/360K diskette drive.

Copy Protection

The following methods of copy protection may not work on systems using the High Capacity Diskette Drive:

- Bypassing BIOS
- Diskette drive controls
- Write current control

Bypassing BIOS

Copy protection that tries to bypass the following BIOS routines will not work on the High Capacity Diskette Drive:

Track Density: The High Capacity Diskette Drive records tracks at a density of 96 tracks per inch (TPI). This drive has to double-step in the 48 TPI mode, which is performed by BIOS.

Data Transfer Rate: BIOS selects the proper data transfer rate for the media being used.

Disk __ Base: Copy protection, which creates its own disk __ base will not work on the High Capacity Diskette Drive.

Diskette Drive Controls

Copy protection that uses the following will not work on the High Capacity Diskette Drive:

Rotational Speed: The time between two events on a diskette is controlled by the Fixed Disk and Diskette Drive Adapter.

Access Time: Diskette BIOS routines must set the track-to-track access time for the different types of media used on the IBM Personal Computer XT Model 286.

Head Geometry: See "Diskette Drive Compatibility" on page 8-5.

Diskette Change Signal: Copy protection may not be able to reset this signal.

Write Current Control

Copy protection that uses write current control will not work because the Fixed Disk and Diskette Drive Adapter selects the proper write current for the media being used.

Application Guidelines

The following information should be used to develop application programs for the IBM Personal Computer family.

High-Level Language Considerations

The IBM-supported languages of BASIC, FORTRAN, COBOL, Pascal, and APL are the best choices for writing compatible programs.

If a program uses specific features of the hardware, that program may not be compatible with all IBM Personal Computers. Specifically, the use of assembler language subroutines or hardware-specific commands (In, Out, Peek, Poke, ...) must follow the assembler language rules (see "Assembler Language Programming Considerations" on page 8-8).

Any program that requires precise timing information should obtain it through a DOS or language interface; for example, TIME\$ in BASIC. If greater precision is required, the assembler techniques in "Assembler Language Programming Considerations" are available. The use of programming loops may prevent a program from being compatible with other IBM Personal Computers.

Assembler Language Programming Considerations

The following OP codes work differently on systems using the 80286 microprocessor than they do on systems using the 8088 microprocessor.

- If the system microprocessor executes a POPF instruction in either the real or the virtual address mode with $CPL \leq IOPL$, then a pending maskable interrupt (the INTR pin active) may be improperly recognized after executing the POPF instruction even if maskable interrupts were disabled before the POPF instruction and the value popped had $IF=0$. If the interrupt is improperly recognized, the interrupt is still correctly executed. This errata has no effect when interrupts are enabled in either real or virtual address mode. This errata has no effect in the virtual address mode when $CPL > IOPL$.

The POPF instruction may be simulated with the following code macro:

```
POPFF      Macro      ; use POPFF instead of POPF
              ; simulate popping flags
              ; using IRET
EB 01      JMP $+3    ; jump around IRET
CF         IRET       ; POP CS, IP, flags
OE         PUSH CS
E8 FB FF   CALL $-2  ; CALL within segment
              ; program will continue here
```

- **PUSH SP**
 - 80286 microprocessor pushes the current stack pointer.
 - 8088 microprocessor pushes the new stack pointer.
- **Single step interrupt (when $TF=1$) on the interrupt instruction (OP code hex CC,CD):**
 - 80286 microprocessor does **not** interrupt on the INT instruction.

8088 microprocessor does interrupt on the INT instruction.

- The divide error exception (interrupt 0):

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

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

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

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

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

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

Exception #NP(0) = Exception #11 = Not-present Fault

Exception #SS(0) = Exception #12 = Stack Fault

Exception #GP(0) = Exception #13 = General Protection Fault

- Exception #GP(0) from attempted access to data segment or extra segment when the corresponding segment register holds a null selector.
- Exception #GP(0) from attempted data read from code segment when code segment has the "execute only" attribute.
- Exception #GP(0) from attempted write to code segment (code segments are not writable in protected mode), or to data segment of extra segment if the data or extra segment has the read only attribute.
- Exception #GP(0) from attempted load of a selector referencing the local descriptor table into CS, DS, ES or SS, when the LDT is not present.

- Exception #GP(0) from attempted input or output instruction when $CPL > IOPL$.
- Exception #GP(selector) from attempted access to a descriptor is GDT, LDT, or IDT, beyond the defined limit of the descriptor table.
- Exception #GP(0) from attempted read or write (except for "PUSH" onto stack) beyond the defined limit of segment.
- Exception #SS(0) from attempted "PUSH" below the defined limit of the stack segment.

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

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

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

The following correctly describes the operation of all 80286 parts:

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

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

- Program interrupts are used for access to these functions. This practice removes the absolute addressing from the program. Only the interrupt number is required.

8-10 Compatibility

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

The power-on-self-test code in the system ROM enables hardware IRQ 13 and sets up its vector to point to a routine in ROM. The ROM routine clears the 'busy' signal latch and then transfers control to the address pointed to by the NMI interrupt vector. This allows code written for any IBM Personal Computer to work on an IBM Personal Computer XT Model 286. The NMI interrupt handler should read the coprocessor's status to determine if the NMI was caused by the coprocessor. If the interrupt was not generated by the coprocessor, control should be passed to the original NMI interrupt handler.

- Back to back I/O commands to the same I/O ports will not permit enough recovery time for I/O chips. To ensure enough time, a `JMP SHORT $+2` must be inserted between IN/OUT instructions to the same I/O chip.

Note: `MOV AL,AH` type instruction does not allow enough recovery time. An example of the correct procedure follows:

```
OUT  IO_ADD,AL
JMP  SHORT $+2
MOV  AL,AH
OUT  IO_ADD,AL
```

- In systems using the 80286 microprocessor, IRQ 9 is redirected to INT hex 0A (hardware IRQ 2). This insures that hardware designed to use IRQ 2 will operate in the IBM Personal Computer XT Model 286.
- The system can mask hardware sensitivity. New devices can change the ROM BIOS to accept the same programming interface on the new device.

- In cases where BIOS provides parameter tables, such as for video or diskette, a program may substitute new parameter values by building a new copy of the table and changing the vector to point to that table. However, the program should copy the current table, using the current vector, and then modify those locations in the table that need to be changed. In this way, the program will not inadvertently change any values that should be left the same.
- **Disk__Base** consists of 11 parameters required for diskette operation. They are pointed at by the data variable, **Disk__Pointer**, at absolute address 0:78. It is strongly recommended that the values supplied in ROM be used. If it becomes necessary to modify any of the parameters, build another parameter block and modify the address in **Disk__Pointer** to point to the new block.

The parameters were established to operate both the High Capacity Diskette Drive and the Double Sided Diskette Drive. Three of the parameters in this table are under control of BIOS in the following situations.

The Gap Length Parameter is no longer retrieved from the parameter block.

The gap length used during diskette read, write, and verify operations is derived from within diskette BIOS.

The gap length for format operations is still obtained from the parameter block.

Special considerations are required for formatting operations. See the prolog of Diskette BIOS for the required details. If a parameter block contains a head settle time parameter value of 0 milliseconds, and a write operation is being performed, at least 15 milliseconds of head settle time will be enforced for a High Capacity Diskette Drive and 20 milliseconds will be enforced for a Double Sided Diskette Drive. If a parameter block contains a motor start wait parameter of less than 1 second for a write or format operation of 625 milliseconds for a read or verify operation, Diskette BIOS will enforce those times listed above.

- The following procedure is used to determine the type of media inserted in the High Capacity Diskette Drive:
 1. Read Track 0, Head 0, Sector 1 to allow diskette BIOS to establish the media/drive combination. If this is successful, continue with the next step.
 2. Read Track 0, Sector 15. If an error occurs, a double sided diskette is in the drive.

Note: Refer to the *DOS Technical Reference* manual for the File Allocation Table (FAT) parameters for single- and double-sided diskettes.
If a successful read occurs, a high capacity diskette is in the drive.
 3. If Step 1 fails, issue the reset function (AH=0) to diskette BIOS and retry. If a successful read cannot be done, the media needs to be formatted or is defective.

ROM BIOS and DOS do not provide for all functions. The following are the allowable I/O operations with which IBM will maintain compatibility in future systems.

- Control of the sound, using port hex 61, and the sound channel of the timer/counter. A program can control timer/counter channels 0 and 2, ports hex 40, 42, and 43. A program must not change the value in port hex 41, because this port controls the dynamic-memory refresh. Channel 0 provides the time-of-day interrupt, and can also be used for timing short intervals. Channel 2 of the timer/counter is the output for the speaker and cassette ports. This channel may also be used for timing short intervals, although it cannot interrupt at the end of the period.

Note: Programs should use the timer for delay on the paddle input rather than a program loop.
- Interrupt Mask Register (IMR), port hex 21, can be used to selectively mask and unmask the hardware features.

The following information pertains to absolute memory locations.

- **Interrupt Vectors Segment (hex 0)**--A program may change these to point at different processing routines. When an interrupt vector is modified, the original value should be retained. If the interrupt, either hardware or program, is not directed toward this device handler, the request should be passed to the next item in the list.
- **Video Display Buffers (hex B000 and B8000)**-- For each mode of operation defined in the video display BIOS, the memory map will remain the same. For example, the bit map for the 320 x 200 medium-resolution graphics mode of the Color/Graphics Monitor adapter will be retained on any future adapter that supports that mode. If the bit map is modified, a different mode number will be used.
- **ROM BIOS Data Area (hex 40:0)**--Any variables in this area will retain their current definition, whenever it is reasonable to do so. IBM may use these data areas for other purposes when the variable no longer has meaning in the system. In general, ROM BIOS data variables should be read or modified through BIOS calls whenever possible, and not with direct access to the variable.

A program that requires timing information should use either the time-of-day clock or the timing channels of the timer/counter. The input frequency to the timer will be maintained at 1.19 MHz, providing a constant time reference. Program loops should be avoided.

Programs that use copy protection schemes should use the ROM BIOS diskette calls to read and verify the diskette and should not be timer dependent. Any method can be used to create the diskette, although manufacturing capability should be considered. The verifying program can look at the diskette controller's status bytes in the ROM BIOS data area for additional information about embedded errors. More information about copy protection may be found on page 8-5 under "Copy Protection".

Any DOS program must be relocatable and insensitive to the size of DOS or its own load addresses. A program's memory requirement should be identified and contiguous with the load module. A program should not assume that all of memory is available to it.

There are several 80286 instructions that, when executed, lock out external bus signals. DMA requests are not honored during the execution of these instructions. Consecutive instructions of this type prevent DMA activity from the start of the first instruction to the end of the last instruction. To allow for necessary DMA cycles, as required by the diskette controller in a multitasking system, multiple lock-out instructions must be separated by JMP SHORT \$+2.

Multitasking Provisions

The IBM Personal Computer XT Model 286 BIOS contains a feature to assist multitasking implementation. "Hooks" are provided for a multitasking dispatcher. Whenever a busy (wait) loop occurs in the BIOS, a hook is provided for the program to break out of the loop. Also, whenever BIOS services an interrupt, a corresponding wait loop is exited, and another hook is provided. Thus a program may be written that employs the bulk of the device driver code. The following is valid only in the microprocessor's real address mode and must be taken by the code to allow this support.

The program is responsible for the serialization of access to the device driver. The BIOS code is not reentrant.

The program is responsible for matching corresponding wait and post calls.

Interfaces

There are four interfaces to be used by the multitasking dispatcher:

Startup

First, the startup code hooks interrupt hex 15. The dispatcher is responsible to check for function codes of AH= hex 90 or 91. The "Wait" and "Post" sections describe these codes. The dispatcher must pass all other functions to the previous user of interrupt hex 15. This can be done by a JMP or a CALL. If the function code is hex 90 or 91, the dispatcher should do the appropriate processing and return by the IRET instruction.

Serialization

It is up to the multitasking system to ensure that the device driver code is used serially. Multiple entries into the code can result in serious errors.

Wait (Busy)

Whenever the BIOS is about to enter a busy loop, it first issues an interrupt hex 15 with a function code of hex 90 in AH. This signals a wait condition. At this point, the dispatcher should save the task status and dispatch another task. This allows overlapped execution of tasks when the hardware is busy. The following is an outline of the code that has been added to the BIOS to perform this function.

```
MOV AX, 90XXH      ; wait code in AH and
                   ; type code in AL
INT 15H           ; issue call
JC  TIMEOUT       ; optional: for time-out or
                   ; if carry is set, time-out
                   ; occurred
NORMAL TIMEOUT LOGIC ; normal time-out
```

Post (Interrupt)

Whenever the BIOS has set an interrupt flag for a corresponding busy loop, an interrupt 15 occurs with a function code of hex 91 in AH. This signals a post condition. At this point, the dispatcher should set the task status to "ready to run" and return to the interrupt routine. The following is an outline of the code added to BIOS that performs this function.

```
MOV AX, 91XXH      ; post code AH and
                   ; type code AL
INT 15H           ; issue call
```

Classes

The following types of wait loops are supported:

- The class for hex 0 to 7F is serially reusable. This means that for the devices that use these codes, access to the BIOS must be restricted to only one task at a time.
- The class for hex 80 to BF is reentrant. There is no restriction on the number of tasks that may access the device.
- The class for hex C0 to FF is non-interrupt. There is no corresponding interrupt for the wait loop. Therefore, it is the responsibility of the dispatcher to determine what satisfies this condition to exit the loop.

Function Code Classes

Type Code (AL)	Description
00H->7FH	Serially reusable devices; operating system must serialize access
80H->0BFH	Reentrant devices; ES:BX is used to distinguish different calls (multiple I/O calls are allowed simultaneously)

0C0H->0FH Wait only calls; there is no complementary POST for these waits--these are time-out only. Times are function-number dependent.

Function Code Assignments

The following are specific assignments for the IBM Personal Computer XT Model 286 BIOS. Times are approximate. They are grouped according to the classes described under "Function Code Classes".

Type Code (AL)	Time-out	Description
00H	yes (6 second)	fixed disk
01H	yes (2 second)	diskette
02H	no (2 second)	keyboard
0FDH	yes (1 second-write)	diskette motor start
--	(625 ms-read)	--
0FEH	yes (18 second)	printer

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

Time-Outs

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

Machine-Sensitive Code

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

First Byte	Second Byte	Machine Identification
FF	00	IBM Personal Computer
FE	00	IBM Personal Computer XT
FE	00	IBM Portable Personal Computer
FD	00	IBM PCjr
FC	00	IBM Personal Computer AT
FC	02	IBM Personal Computer XT Model 286
FB	00	IBM Personal Computer XT with 256/640 system board

Machine Identification Code

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

Notes:

Glossary

This glossary includes terms and definitions from the *IBM Vocabulary for Data Processing, Telecommunications, and Office Systems*, GC20-1699.

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 alphameric.

alternating current (ac). A current that periodically reverses its direction of flow.

American National Standard Code for Information Interchange (ASCII). The standard code, using a coded character set consisting of 7-bit coded characters (8 bits including parity check), used for information exchange between data processing systems, data communication systems, and associated equipment. The ASCII set consists of control characters and graphic characters.

ampere (A). The basic unit of electric current.

A/N. Alphanumeric

analog. (1) Pertaining to data in the form of continuously variable physical quantities. (2) Contrast with digital.

AND. A logic operator having the property that if P is a statement, Q is a statement, R is a statement,..., then the AND of P, Q, R,... is true if all statements are true, false if any statement is false.

AND gate. A logic gate in which the output is 1 only if all inputs are 1.

AND operation. The boolean operation whose result has the boolean value 1, if and only if, each operand has the boolean value 1. Synonymous with conjunction.

APA. All points addressable.

ASCII. American National Standard Code for Information Interchange.

assemble. To translate a program expressed in an assembler language into a computer language.

assembler. A computer program used to assemble.

assembler language. A computer-oriented language whose instructions are usually in one-to-one correspondence with computer instructions.

asynchronous transmission. (1) Transmission in which the time of occurrence of the start of each character, or block of characters, is arbitrary; once started, the time of occurrence of each signal representing a bit within a character, or block, has the same relationship to significant instants of a fixed time frame. (2) Transmission in which each information character is individually transmitted (usually timed by the use of start elements and stop elements).

audio frequencies. Frequencies that can be heard by the human ear (approximately 15 hertz to 20,000 hertz).

auxiliary storage. (1) A storage device that is not main storage. (2) Data storage other than main storage; for example, storage on magnetic disk. (3) Contrast with main storage.

BASIC. Beginner's all-purpose symbolic instruction code.

basic input/output system (BIOS). The feature of the IBM Personal Computer that provides the level control of the major I/O devices, and relieves the programmer from concern about hardware device characteristics.

baud. (1) A unit of signaling speed equal to the number of discrete conditions or signal events per second. For example, one baud equals one bit per second in a train of binary signals, one-half dot cycle per second in Morse code, and one 3-bit value per second in a train of signals each of which can assume one of eight different states. (2) In asynchronous transmission, the unit of modulation rate corresponding to one unit of interval per second; that is, if the duration of the unit interval is 20 milliseconds, the modulation rate is 50 baud.

BCC. Block-check character.

beginner's all-purpose symbolic instruction code (BASIC). A programming language with a small repertoire of commands and a simple syntax, primarily designed for numeric applications.

binary. (1) Pertaining to a selection, choice, or condition that has two possible values or states. (2) Pertaining to a fixed radix numeration system having a radix of 2.

binary digit. (1) In binary notation, either of the characters 0 or 1. (2) Synonymous with bit.

binary notation. Any notation that uses two different characters, usually the binary digits 0 and 1.

binary synchronous communications (BSC). A uniform procedure, using a standardized set of control characters and control character sequences for synchronous transmission of binary-coded data between stations.

BIOS. Basic input/output system.

bit. Synonym for binary digit

bits per second (bps). A unit of measurement representing the number of discrete binary digits transmitted by a device in one second.

block. (1) A string of records, a string of words, or a character string formed for technical or logic reasons to be treated as an entity. (2) A set of things, such as words, characters, or digits, treated as a unit.

block-check character (BCC). In cyclic redundancy checking, a character that is transmitted by the sender after each message block and is compared with a block-check character computed by the receiver to determine if the transmission was successful.

boolean operation. (1) Any operation in which each of the operands and the result take one of two values. (2) An operation that follows the rules of boolean algebra.

bootstrap. A technique or device designed to bring itself into a desired state by means of its own action; for example, a machine routine whose first few instructions are sufficient to bring the rest of itself into the computer from an input device.

bps. Bits per second.

BSC. Binary synchronous communications.

buffer. (1) An area of storage that is temporarily reserved for use in performing an input/output operation, into which data is read or from which data is written. Synonymous with I/O area. (2) A portion of storage for temporarily holding input or output data.

bus. One or more conductors used for transmitting signals or power.

byte. (1) A sequence of eight adjacent binary digits that are operated upon as a unit. (2) A binary character operated upon as a unit. (3) The representation of a character.

C. Celsius.

capacitor. An electronic circuit component that stores an electric charge.

Cartesian coordinates. A system of coordinates for locating a point on a plane by its distance from each of two intersecting lines, or in space by its distance from each of three mutually perpendicular planes.

CAS. Column address strobe.

cathode ray tube (CRT). A vacuum tube in which a stream of electrons is projected onto a fluorescent screen producing a luminous spot. The location of the spot can be controlled.

cathode ray tube display (CRT display). (1) A CRT used for displaying data. For example, the electron beam can be controlled to form alphanumeric data by use of a dot matrix. (2) Synonymous with monitor.

CCITT. International Telegraph and Telephone Consultative Committee.

Celsius (C). A temperature scale. Contrast with Fahrenheit (F).

central processing unit (CPU). Term for processing unit.

channel. A path along which signals can be sent; for example, data channel, output channel.

character generator. (1) In computer graphics, a functional unit that converts the coded representation of a graphic character into the shape of the character for display. (2) In word processing, the means within equipment for generating visual characters or symbols from coded data.

character set. (1) A finite set of different characters upon which agreement has been reached and that is considered complete for some purpose. (2) A set of unique representations called characters. (3) A defined collection of characters.

characters per second (cps). A standard unit of measurement for the speed at which a printer prints.

check key. A group of characters, derived from and appended to a data item, that can be used to detect errors in the data item during processing.

clipping. In computer graphics, removing parts of a display image that lie outside a window.

closed circuit. A continuous unbroken circuit; that is, one in which current can flow. Contrast with open circuit.

CMOS. Complementary metal oxide semiconductor.

code. (1) A set of unambiguous rules specifying the manner in which data may be represented in a discrete form. Synonymous with coding scheme. (2) A set of items, such as abbreviations, representing the members of another set. (3) To represent data or a computer program in a symbolic form that can be accepted by a data processor. (4) Loosely, one or more computer programs, or part of a computer program.

coding scheme. Synonym for code.

collector. An element in a transistor toward which current flows.

color cone. An arrangement of the visible colors on the surface of a double-ended cone where lightness varies along the axis of

the cone, and hue varies around the circumference. Lightness includes both the intensity and saturation of color.

column address strobe (CAS). A signal that latches the column addresses in a memory chip.

compile. (1) To translate a computer program expressed in a problem-oriented language into a computer-oriented language. (2) To prepare a machine-language program from a computer program written in another programming language by making use of the overall logic structure of the program, or generating more than one computer instruction for each symbolic statement, or both, as well as performing the function of an assembler.

complement. A number that can be derived from a specified number by subtracting it from a second specified number.

complementary metal oxide semiconductor (CMOS). A logic circuit family that uses very little power. It works with a wide range of power supply voltages.

computer. A functional unit that can perform substantial computation, including numerous arithmetic operations or logic operations, without human intervention during a run.

computer instruction code. A code used to represent the instructions in an instruction set. Synonymous with machine code.

computer program. A sequence of instructions suitable for processing by a computer.

computer word. A word stored in one computer location and capable of being treated as a unit.

configuration. (1) The arrangement of a computer system or network as defined by the nature, number, and the chief characteristics of its functional units. More specifically, the term configuration may refer to a hardware configuration or a software configuration. (2) The devices and programs that make up a system, subsystem, or network.

conjunction. Synonym for AND operation.

contiguous. Touching or joining at the edge or boundary; adjacent.

control character. A character whose occurrence in a particular context initiates, modifies, or stops a control operation.

control operation. An action that affects the recording, processing, transmission, or interpretation of data; for example, starting or stopping a process, carriage return, font change, rewind, and end of transmission.

control storage. A portion of storage that contains microcode.

coordinate space. In computer graphics, a system of Cartesian coordinates in which an object is defined.

cps. Characters per second.

CPU. Central processing unit.

CRC. Cyclic redundancy check.

CRT. Cathode ray tube.

CRT display. Cathode ray tube display.

CTS. Clear to send. Associated with modem control.

cursor. (1) In computer graphics, a movable marker that is used to indicate position on a display. (2) A displayed symbol that acts as a marker to help the user locate a point in text, in a system command, or in storage. (3) A movable spot of light on the screen of a display device, usually indicating where the next character is to be entered, replaced, or deleted.

cyclic redundancy check (CRC). (1) A redundancy check in which the check key is generated by a cyclic algorithm. (2) A system of error checking performed at both the sending and receiving station after a block-check character has been accumulated.

cylinder. (1) The set of all tracks with the same nominal distance from the axis about which the disk rotates. (2) The

tracks of a disk storage device that can be accessed without repositioning the access mechanism.

daisy-chained cable. A type of cable that has two or more connectors attached in series.

data. (1) A representation of facts, concepts, or instructions in a formalized manner suitable for communication, interpretation, or processing by human or automatic means. (2) Any representations, such as characters or analog quantities, to which meaning is, or might be assigned.

data base. A collection of data that can be immediately accessed and operated upon by a data processing system for a specific purpose.

data processing system. A system that performs input, processing, storage, output, and control functions to accomplish a sequence of operations on data.

data transmission. Synonym for transmission.

dB. Decibel.

dBa. Adjusted decibels.

dc. Direct current.

debounce. (1) An electronic means of overcoming the make/break bounce of switches to obtain one smooth change of signal level. (2) The elimination of undesired signal variations caused by mechanically generated signals from contacts.

decibel. (1) A unit that expresses the ratio of two power levels on a logarithmic scale. (2) A unit for measuring relative power.

decoupling capacitor. A capacitor that provides a low impedance path to ground to prevent common coupling between circuits.

Deutsche Industrie Norm (DIN). (1) German Industrial Norm. (2) The committee that sets German dimension standards.

digit. (1) A graphic character that represents an integer; for example, one of the characters 0 to 9. (2) A symbol that

represents one of the non-negative integers smaller than the radix. For example, in decimal notation, a digit is one of the characters 0 to 9.

digital. (1) Pertaining to data in the form of digits.
(2) Contrast with analog.

DIN. Deutsche Industrie Norm.

DIN connector. One of the connectors specified by the DIN committee.

DIP. Dual in-line package.

DIP switch. One of a set of small switches mounted in a dual in-line package.

direct current (dc). A current that always flows in one direction.

direct memory access (DMA). A method of transferring data between main storage and I/O devices that does not require processor intervention.

disable. To stop the operation of a circuit or device.

disabled. Pertaining to a state of a processing unit that prevents the occurrence of certain types of interruptions. Synonymous with masked.

disk. Loosely, a magnetic disk.

diskette. A thin, flexible magnetic disk and a semirigid protective jacket, in which the disk is permanently enclosed. Synonymous with flexible disk.

diskette drive. A device for storing data on and retrieving data from a diskette.

display. (1) A visual presentation of data. (2) A device for visual presentation of information on any temporary character imaging device. (3) To present data visually. (4) See cathode ray tube display.

display attribute. In computer graphics, a particular property that is assigned to all or part of a display; for example, low intensity, green color, blinking status.

display element. In computer graphics, a basic graphic element that can be used to construct a display image; for example, a dot, a line segment, a character.

display group. In computer graphics, a collection of display elements that can be manipulated as a unit and that can be further combined to form larger groups.

display image. In computer graphics, a collection of display elements or display groups that are represented together at any one time in a display space.

display space. In computer graphics, that portion of a display surface available for a display image. The display space may be all or part of a display surface.

display surface. In computer graphics, that medium on which display images may appear; for example, the entire screen of a cathode ray tube.

DMA. Direct memory access.

dot matrix. (1) In computer graphics, a two-dimensional pattern of dots used for constructing a display image. This type of matrix can be used to represent characters by dots. (2) In word processing, a pattern of dots used to form characters. This term normally refers to a small section of a set of addressable points; for example, a representation of characters by dots.

dot printer. Synonym for matrix printer.

dot-matrix character generator. In computer graphics, a character generator that generates character images composed of dots.

drawing primitive. A group of commands that draw defined geometric shapes.

DSR. Data set ready. Associated with modem control.

DTR. In the IBM Personal Computer, data terminal ready. Associated with modem control.

dual in-line package (DIP). A widely used container for an integrated circuit. DIPs have pins in two parallel rows. The pins are spaced 1/10 inch apart. See also DIP switch.

duplex. (1) In data communication, pertaining to a simultaneous two-way independent transmission in both directions. (2) Contrast with half-duplex.

duty cycle. In the operation of a device, the ratio of on time to idle time. Duty cycle is expressed as a decimal or percentage.

dynamic memory. RAM using transistors and capacitors as the memory elements. This memory requires a refresh (recharge) cycle every few milliseconds. Contrast with static memory.

EBCDIC. Extended binary-coded decimal interchange code.

ECC. Error checking and correction.

edge connector. A terminal block with a number of contacts attached to the edge of a printed-circuit board to facilitate plugging into a foundation circuit.

EIA. Electronic Industries Association.

electromagnet. Any device that exhibits magnetism only while an electric current flows through it.

enable. To initiate the operation of a circuit or device.

end of block (EOB). A code that marks the end of a block of data.

end of file (EOF). An internal label, immediately following the last record of a file, signaling the end of that file. It may include control totals for comparison with counts accumulated during processing.

end-of-text (ETX). A transmission control character used to terminate text.

end-of-transmission (EOT). A transmission control character used to indicate the conclusion of a transmission, which may have included one or more texts and any associated message headings.

end-of-transmission-block (ETB). A transmission control character used to indicate the end of a transmission block of data when data is divided into such blocks for transmission purposes.

EOB. End of block.

EOF. End of file.

EOT. End-of-transmission.

EPROM. Erasable programmable read-only memory.

erasable programmable read-only memory (EPROM). A PROM in which the user can erase old information and enter new information.

error checking and correction (ECC). The detection and correction of all single-bit errors, plus the detection of double-bit and some multiple-bit errors.

ESC. The escape character.

escape character (ESC). A code extension character used, in some cases, with one or more succeeding characters to indicate by some convention or agreement that the coded representations following the character or the group of characters are to be interpreted according to a different code or according to a different coded character set.

ETB. End-of-transmission-block.

ETX. End-of-text.

extended binary-coded decimal interchange code (EBCDIC). A set of 256 characters, each represented by eight bits.

F. Fahrenheit.

Fahrenheit (F). A temperature scale. Contrast with Celsius (C).

falling edge. Synonym for negative-going edge.

FCC. Federal Communications Commission.

fetch. To locate and load a quantity of data from storage.

FF. The form feed character.

field. (1) In a record, a specified area used for a particular category of data. (2) In a data base, the smallest unit of data that can be referred to.

field-programmable logic sequencer (FPLS). An integrated circuit containing a programmable, read-only memory that responds to external inputs and feedback of its own outputs.

FIFO (first-in-first out). A queuing technique in which the next item to be retrieved is the item that has been in the queue for the longest time.

fixed disk drive. In the IBM Personal Computer, a unit consisting of nonremovable magnetic disks, and a device for storing data on and retrieving data from the disks.

flag. (1) Any of various types of indicators used for identification. (2) A character that signals the occurrence of some condition, such as the end of a word. (3) Deprecated term for mark.

flexible disk. Synonym for diskette.

flip-flop. A circuit or device containing active elements, capable of assuming either one of two stable states at a given time.

font. A family or assortment of characters of a given size and style; for example, 10 point Press Roman medium.

foreground. (1) In multiprogramming, the environment in which high-priority programs are executed. (2) On a color display screen, the characters as opposed to the background.

form feed. (1) Paper movement used to bring an assigned part of a form to the printing position. (2) In word processing, a

function that advances the typing position to the same character position on a predetermined line of the next form or page.

form feed character. A control character that causes the print or display position to move to the next predetermined first line on the next form, the next page, or the equivalent.

format. The arrangement or layout of data on a data medium.

FPLS. Field-programmable logic sequencer.

frame. (1) In SDLC, the vehicle for every command, every response, and all information that is transmitted using SDLC procedures. Each frame begins and ends with a flag. (2) In data transmission, the sequence of contiguous bits bracketed by and including beginning and ending flag sequences.

g. Gram.

G. (1) Prefix giga; 1,000,000,000. (2) When referring to computer storage capacity, 1,073,741,824. ($1,073,741,824 = 2$ to the 30th power.)

gate. (1) A combinational logic circuit having one output channel and one or more input channels, such that the output channel state is completely determined by the input channel states. (2) A signal that enables the passage of other signals through a circuit.

Gb. 1,073,741,824 bytes.

general-purpose register. A register, usually explicitly addressable within a set of registers, that can be used for different purposes; for example, as an accumulator, as an index register, or as a special handler of data.

giga (G). Prefix 1,000,000,000.

gram (g). A unit of weight (equivalent to 0.035 ounces).

graphic. A symbol produced by a process such as handwriting, drawing, or printing.

graphic character. A character, other than a control character, that is normally represented by a graphic.

half-duplex. (1) In data communication, pertaining to an alternate, one way at a time, independent transmission. (2) Contrast with duplex.

hardware. (1) Physical equipment used in data processing, as opposed to programs, procedures, rules, and associated documentation. (2) Contrast with software.

head. A device that reads, writes, or erases data on a storage medium; for example, a small electromagnet used to read, write, or erase data on a magnetic disk.

hertz (Hz). A unit of frequency equal to one cycle per second.

hex. Common abbreviation for hexadecimal. Also, hexadecimal can be noted as X' '.

hexadecimal. (1) Pertaining to a selection, choice, or condition that has 16 possible different values or states. These values or states are usually symbolized by the ten digits 0 through 9 and the six letters A through F. (2) Pertaining to a fixed radix numeration system having a radix of 16.

high impedance state. A state in which the output of a device is effectively isolated from the circuit.

highlighting. In computer graphics, emphasizing a given display group by changing its attributes relative to other display groups in the same display field.

high-order position. The leftmost position in a string of characters. See also most-significant digit.

hither plane. In computer graphics, a plane that is perpendicular to the line joining the viewing reference point and the view point and that lies between these two points. Any part of an object between the hither plane and the view point is not seen. See also yon plane.

housekeeping. Operations or routines that do not contribute directly to the solution of the problem but do contribute directly to the operation of the computer.

Hz. Hertz

image. A fully processed unit of operational data that is ready to be transmitted to a remote unit; when loaded into control storage in the remote unit, the image determines the operations of the unit.

immediate instruction. An instruction that contains within itself an operand for the operation specified, rather than an address of the operand.

index register. A register whose contents may be used to modify an operand address during the execution of computer instructions.

indicator. (1) A device that may be set into a prescribed state, usually according to the result of a previous process or on the occurrence of a specified condition in the equipment, and that usually gives a visual or other indication of the existence of the prescribed state, and that may in some cases be used to determine the selection among alternative processes; for example, an overflow indicator. (2) An item of data that may be interrogated to determine whether a particular condition has been satisfied in the execution of a computer program; for example, a switch indicator, an overflow indicator.

inhibited. (1) Pertaining to a state of a processing unit in which certain types of interruptions are not allowed to occur. (2) Pertaining to the state in which a transmission control unit or an audio response unit cannot accept incoming calls on a line.

initialize. To set counters, switches, addresses, or contents of storage to 0 or other starting values at the beginning of, or at prescribed points in, the operation of a computer routine.

input/output (I/O). (1) Pertaining to a device or to a channel that may be involved in an input process, and, at a different time, in an output process. In the English language, "input/output" may be used in place of such terms as "input/output data," "input/output signal," and "input/output terminals," when such usage is clear in a given context. (2) Pertaining to a device

whose parts can be performing an input process and an output process at the same time. (3) Pertaining to either input or output, or both.

instruction. In a programming language, a meaningful expression that specifies one operation and identifies its operands, if any.

instruction set. The set of instructions of a computer, of a programming language, or of the programming languages in a programming system.

intensity. In computer graphics, the amount of light emitted at a display point

interface. A device that alters or converts actual electrical signals between distinct devices, programs, or systems.

interleave. To arrange parts of one sequence of things or events so that they alternate with parts of one or more other sequences of the same nature and so that each sequence retains its identity.

interrupt. (1) A suspension of a process, such as the execution of a computer program, caused by an event external to that process, and performed in such a way that the process can be resumed. (2) In a data transmission, to take an action at a receiving station that causes the transmitting station to terminate a transmission. (3) Synonymous with interruption.

I/O. Input/output.

I/O area. Synonym for buffer.

irrecoverable error. An error that makes recovery impossible without the use of recovery techniques external to the computer program or run.

joystick. In computer graphics, a lever that can pivot in all directions and that is used as a locator device.

k. Prefix kilo; 1000.

K. When referring to storage capacity, 1024. ($1024 = 2$ to the 10th power.)

KB. 1024 bytes.

key lock. A device that deactivates the keyboard and locks the cover on for security.

kg. Kilogram; 1000 grams.

kHz. Kilohertz; 1000 hertz.

kilo (k). Prefix 1000

kilogram (kg). 1000 grams.

kilohertz (kHz). 1000 hertz

latch. (1) A simple logic-circuit storage element. (2) A feedback loop in sequential digital circuits used to maintain a state.

least-significant digit. The rightmost digit. See also low-order position.

LED. Light-emitting diode.

light-emitting diode (LED). A semiconductor device that gives off visible or infrared light when activated.

load. In programming, to enter data into storage or working registers.

look-up table (LUT). (1) A technique for mapping one set of values into a larger set of values. (2) In computer graphics, a table that assigns a color value (red, green, blue intensities) to a color index.

low power Schottky TTL. A version (LS series) of TTL giving a good compromise between low power and high speed. See also transistor-transistor logic and Schottky TTL.

low-order position. The rightmost position in a string of characters. See also least-significant digit.

luminance. The luminous intensity per unit projected area of a given surface viewed from a given direction.

LUT. Look-up table.

m. (1) Prefix milli; 0.001. (2) Meter.

M. (1) Prefix mega; 1,000,000. (2) When referring to computer storage capacity, 1,048,576. ($1,048,576 = 2$ to the 20th power.)

mA. Milliampere; 0.001 ampere.

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

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

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

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

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

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

masked. Synonym for disabled.

matrix. (1) A rectangular array of elements, arranged in rows and columns, that may be manipulated according to the rules of matrix algebra. (2) In computers, a logic network in the form of an array of input leads and output leads with logic elements connected at some of their intersections.

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

MB. 1,048,576 bytes.

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

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

memory. Term for main storage.

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

MFM. Modified frequency modulation.

MHz. Megahertz; 1,000,000 hertz.

micro (μ). Prefix 0.000,001.

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

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

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

microsecond (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 (RGBO). The description of a direct-drive color monitor that accepts input signals of red, green, blue, and intensity.

redundancy check. A check that depends on extra characters attached to data for the detection of errors. See cyclic redundancy check.

register. (1) A storage device, having a specified storage capacity such as a bit, a byte, or a computer word, and usually intended for a special purpose. (2) A storage device in which specific data is stored.

retry. To resend the current block of data (from the last EOB or ETB) a prescribed number of times, or until it is entered correctly or accepted.

reverse video. A form of highlighting a character, field, or cursor by reversing the color of the character, field, or cursor with its background; for example, changing a red character on a black background to a black character on a red background.

RF. Radio frequency.

RF modulator. The device used to convert the composite video signal to the antenna level input of a home TV.

RGBO. Red-green-blue-intensity.

rising edge. Synonym for positive-going edge.

ROM. Read-only memory.

ROM/BIOS. The ROM resident basic input/output system, which provides the level control of the major I/O devices in the computer system.

row address strobe (RAS). A signal that latches the row address in a memory chip.

RS-232C. A standard by the EIA for communication between computers and external equipment.

RTS. Request to send. Associated with modem control.

run. A single continuous performance of a computer program or routine.

saturation. In computer graphics, the purity of a particular hue. A color is said to be saturated when at least one primary color (red, blue, or green) is completely absent.

scaling. In computer graphics, enlarging or reducing all or part of a display image by multiplying the coordinates of the image by a constant value.

schematic. The representation, usually in a drawing or diagram form, of a logical or physical structure.

Schottky TTL. A version (S series) of TTL with faster switching speed, but requiring more power. See also transistor-transistor logic and low power Schottky TTL.

SDL. Shielded Data Link

SDLC. Synchronous Data Link Control.

sector. That part of a track or band on a magnetic drum, a magnetic disk, or a disk pack that can be accessed by the magnetic heads in the course of a predetermined rotational displacement of the particular device.

SERDES. Serializer/deserializer.

serial. (1) Pertaining to the sequential performance of two or more activities in a single device. In English, the modifiers serial and parallel usually refer to devices, as opposed to sequential and consecutive, which refer to processes. (2) Pertaining to the sequential or consecutive occurrence of two or more related activities in a single device or channel. (3) Pertaining to the sequential processing of the individual parts of a whole, such as the bits of a character or the characters of a word, using the same facilities for successive parts. (4) Contrast with parallel.

serializer/deserializer (SERDES). A device that serializes output from, and deserializes input to, a business machine.

setup. (1) In a computer that consists of an assembly of individual computing units, the arrangement of interconnections between the units, and the adjustments needed for the computer to operate. (2) The preparation of a computing system to perform a job or job step. Setup is usually performed by an operator and often involves performing routine functions, such as mounting tape reels. (3) The preparation of the system for normal operation.

short circuit. A low-resistance path through which current flows, rather than through a component or circuit.

signal. A variation of a physical quantity, used to convey data.

sink. A device or circuit into which current drains.

software. (1) Computer programs, procedures, and rules concerned with the operation of a data processing system. (2) Contrast with hardware.

source. The origin of a signal or electrical energy.

square wave. An alternating or pulsating current or voltage whose waveshape is square.

square wave generator. A signal generator delivering an output signal having a square waveform.

SS. Start-stop.

start bit. (1) A signal to a receiving mechanism to get ready to receive data or perform a function. (2) In a start-stop system, a signal preceding a character or block that prepares the receiving device for the reception of the code elements.

start-of-text (STX). A transmission control character that precedes a text and may be used to terminate the message heading.

start-stop system. A data transmission system in which each character is preceded by a start bit and is followed by a stop bit.

start-stop (SS) transmission. (1) Asynchronous transmission such that a group of signals representing a character is preceded by a start bit and followed by a stop bit. (2) Asynchronous transmission in which a group of bits is preceded by a start bit that prepares the receiving mechanism for the reception and registration of a character and is followed by at least one stop bit that enables the receiving mechanism to come to an idle condition pending the reception of the next character.

static memory. RAM using flip-flops as the memory elements. Data is retained as long as power is applied to the flip-flops. Contrast with dynamic memory.

stop bit. (1) A signal to a receiving mechanism to wait for the next signal. (2) In a start-stop system, a signal following a character or block that prepares the receiving device for the reception of a subsequent character or block.

storage. (1) A storage device. (2) A device, or part of a device, that can retain data. (3) The retention of data in a storage device. (4) The placement of data into a storage device.

strobe. An instrument that emits adjustable-rate flashes of light. Used to measure the speed of rotating or vibrating objects.

STX. Start-of-text.

symbol. (1) A conventional representation of a concept. (2) A representation of something by reason of relationship, association, or convention.

synchronization. The process of adjusting the corresponding significant instants of two signals to obtain the desired phase relationship between these instants.

Synchronous Data Link Control (SDLC). A protocol for management of data transfer over a data link.

synchronous transmission. (1) Data transmission in which the time of occurrence of each signal representing a bit is related to a fixed time frame. (2) Data transmission in which the sending and receiving devices are operating continuously at substantially the same frequency and are maintained, by means of correction, in a desired phase relationship.

syntax. (1) The relationship among characters or groups of characters, independent of their meanings or the manner of their interpretation and use. (2) The structure of expressions in a language. (3) The rules governing the structure of a language. (4) The relationships among symbols.

text. In ASCII and data communication, a sequence of characters treated as an entity if preceded and terminated by one STX and one ETX transmission control character, respectively.

time-out. (1) A parameter related to an enforced event designed to occur at the conclusion of a predetermined elapsed time. A time-out condition can be cancelled by the receipt of an appropriate time-out cancellation signal. (2) A time interval allotted for certain operations to occur; for example, response to polling or addressing before system operation is interrupted and must be restarted.

track. (1) The path or one of the set of paths, parallel to the reference edge on a data medium, associated with a single reading or writing component as the data medium moves past the component. (2) The portion of a moving data medium such as a drum, or disk, that is accessible to a given reading head position.

transistor-transistor logic (TTL). A popular logic circuit family that uses multiple-emitter transistors.

translate. To transform data from one language to another.

transmission. (1) The sending of data from one place for reception elsewhere. (2) In ASCII and data communication, a series of characters including headings and text. (3) The dispatching of a signal, message, or other form of intelligence by wire, radio, telephone, or other means. (4) One or more blocks or messages. For BSC and start-stop devices, a transmission is terminated by an EOT character. (5) Synonymous with data transmission.

TTL. Transistor-transistor logic.

typematic key. A keyboard key that repeats its function when held pressed.

V. Volt.

vector. In computer graphics, a directed line segment.

video. Computer data or graphics displayed on a cathode ray tube, monitor, or display.

view point. In computer graphics, the origin from which angles and scales are used to map virtual space into display space.

viewing reference point. In computer graphics, a point in the modeling coordinate space that is a defined distance from the view point.

viewing transformation. Operations on the coordinates of an object (usually matrix multiplications) that cause the view of the object to be rotated about any axis, translated (moved without rotating), and/or scaled (changed in size along any or all dimensions). Viewing transformation differs from modeling transformation in that perspective is considered. See also modeling transformation.

viewplane. The visible plane of a CRT display screen that completely contains a defined window.

viewport. In computer graphics, a predefined part of the CRT display space.

volt. The basic practical unit of electric pressure. The potential that causes electrons to flow through a circuit.

W. Watt.

watt. The practical unit of electric power.

window. (1) A predefined part of the virtual space. (2) The visible area of a viewplane.

word. (1) A character string or a bit string considered as an entity. (2) See computer word.

write. To make a permanent or transient recording of data in a storage device or on a data medium.

write precompensation. The varying of the timing of the head current from the outer tracks to the inner tracks of the diskette to keep a constant 'write' signal.

yon plane. In computer graphics, a plane that is perpendicular to the line joining the viewing reference point and the view point, and that lies beyond the viewing reference point. Any part of an object beyond the yon plane is not seen. See also hither plane.

Bibliography

- **Microprocessor and Peripheral Handbook**
 - INTEL Corporation. *210844.001*
- **Introduction to the iAPX 286**
 - INTEL Corporation. *210308.001*
- **iAPX 286 Operating Systems Writer's Guide**
 - INTEL Corporation. *121960.001*
- **iAPX 286 Programmer's Reference Manual**
 - INTEL Corporation. *210498.001*
- **iAPX 286 Hardware Reference Manual**
 - INTEL Corporation. *210760.001*
- **Numeric Processor Extension Data Sheet**
 - INTEL Corporation. *210920*
- **80287 Support Library Reference Manual**
 - INTEL Corporation. *122129*
- **National Semiconductor Corporation. *NS16450***
- **Motorola Microprocessor's Data Manual**
 - Motorola Inc. *Series B*

Notes:

—

—

—

Index

A

AAA 6-8
AAD 6-9
AAM 6-9
AAS 6-8
access time,
 track-to-track 8-6
ACK command 4-13
Acknowledge (ACK)
 command 4-13
ADC 6-6
ADD 6-6
address generation,
 DMA 1-9
address latch enable 1-35
address latch enable,
 buffered 1-32
address mode
 real 1-4
 virtual 1-4
address space, I/O 1-24
address, segment 1-4
addresses, CMOS RAM 1-59
addresses, page register 1-11
AEN 1-35
ALE 8-4
alternate key 4-37
AND 6-10
APL 8-7
application guidelines 8-7
arithmetic instructions 6-6,
 6-25
ARPL 6-19
ASCII characters 7-3
ASCII, extended 4-30

B

BALE 1-32
bandwidth 1-7
BASIC 8-7
basic assurance test 4-5
BASIC interrupts 5-6
BAT (basic assurance
 test) 4-5
BAT Completion Code
 command 4-13
BAT Failure Code
 command 4-13
battery connector 1-75
BHE 1-9
BIOS fixed disk
 parameters 1-65
BIOS memory map 5-10
BIOS programming
 hints 5-10
BIOS quick reference 5-14
block diagram
 keyboard interface 1-54
 system xiv
 system board 1-6
 system timer 1-22
board, system 1-3
BOUND 6-16
break code 4-4
break key 4-38
buffer, keyboard 4-4
buffered address latch
 enable 1-32
buffers, video display 8-14

bus cycle 1-7
busy loop 8-17
bypassing BIOS 8-6
byte high enable 1-9

C

cabling 4-3
CALL 6-13
caps lock key 4-37
CBW 6-9
channel, I/O 1-24
 connectors 1-25
 pin assignments 1-28
 signals 1-31
channels, DMA 1-9
character codes 4-30
characters 7-3
characters keystrokes, and
 colors 7-14
classes, wait loop 8-17
CLC 6-17
CLD 6-17
CLEX 6-27
CLI 6-17
CLK 1-31
clock and data signals 4-27
 data input 4-29
 data output 4-28
 data stream 4-27
clock cycle 1-7
clock line, keyboard 1-58
clock, real-time 1-59
clock, system 1-7
CMC 6-17
CMOS RAM 1-59
CMOS RAM addresses 1-59
CMOS RAM
 configuration 1-63
CMOS RAM I/O
 operations 1-70

CMP 6-7
CMPS 6-11
COBOL 8-7
code
 device driver 8-15
 machine
 identification 8-19
 machine-sensitive 8-19
codes
 character 4-30
 extended 4-34
 multitasking
 function 8-17
command codes, DMA
 controller 1 1-10
command codes, DMA
 controller 2 1-11
commands
 I/O 8-11
 keyboard controller 1-56
commands from the
 system 4-6
 Default Disable 4-7
 Echo 4-7
 Enable 4-7
 Read ID 4-7
 Resend 4-8
 Reset 4-8
 Select Alternate Scan
 Codes 4-8
 Set All Keys 4-9
 Set Default 4-9
 Set Key Type 4-9
 Set Typematic
 Rate/Delay 4-11
 Set/Reset Status
 Indicators 4-10
commands to the system
 ACK (acknowledge) 4-13
 BAT (basic assurance test)
 Completion Code 4-13
 BAT Failure 4-13
commands to the
 system 4-13

- Echo 4-13
- Key Detection Error 4-14
- Keyboard ID 4-14
- Overrun 4-14
- Resend 4-14
- comparison instructions 6-23
- compatibility, hardware 8-3
- condition, wait 8-16
- configuration record 1-60
- configuration, CMOS
 - RAM 1-63
- connectors
 - battery 1-75
 - I/O channel 1-25
 - J-1 through J-8 1-27
 - J-1, J-7 and J-8 1-26
 - keyboard 1-75
 - power supply 1-74
 - power supply output 3-6
 - speaker 1-75
 - system board 1-74
- constants instructions 6-24
- control key 4-36
- control transfer
 - instructions 6-13
- control, sound 8-13
- controller, keyboard 1-44
- controllers
 - DMA 1-7, 1-9, 1-10
 - interrupt 1-12
 - refresh 1-7
- Coprocessor controls 1-42
- coprocessor
 - programming 2-3
- coprocessor, math 2-3
- copy protection 8-5, 8-14
- Ctrl state 4-34
- CTS 6-18
- CWD 6-9
- cycle
 - bus 1-7
 - clock 1-7
 - microprocessor 1-7

D

- DACK 0-3 and 5-7 1-35
- DAS 6-8
- data area, ROM BIOS 8-14
- data input 4-29
- data line, keyboard 1-58
- data output 4-28
- data stream 4-27
- data transfer
 - instructions 6-3, 6-22
- data transfer rate,
 - diskette 8-6
- DEC 6-7, 6-8
- decodes, memory 1-12
- DECSTP 6-28
- Default Disable
 - command 4-7
- default segment
 - workspace 5-10
- delay, typematic 4-4
- description 4-3
 - buffer 4-4
 - cabling 4-3
 - key-code scanning 4-4
 - keys 4-4
 - sequencing key-code scanning 4-4
- descriptors 1-5
- device driver code 8-15
- diagnostic checkpoint
 - port 1-42
- direct memory access 1-9
- disk pointer 8-12
- disk_base 8-6, 8-12
- diskette change signal 8-6
- diskette data transfer
 - rate 8-6
- diskette rotational speed 8-6
- diskette track density 8-6

INDEX

diskette write current 8-6
DIV 6-9
divide error exception 8-9
DMA address generation 1-9
DMA channels 1-9
DMA controller 1-7
DMA controller 1 1-9
DMA controller 1 command codes 1-10
DMA controller 2 1-10
DMA controller 2 command codes 1-11
DMA controllers 1-9
DOS 8-7
DOS function calls 8-10
DOS interrupts 5-6
DRQ0-DRQ3 1-34
DRQ5-DRQ7 1-34

E

Echo command 4-7, 4-13
Enable command 4-7
enable NMI 1-39
encoding, keyboard 4-30
ENTER 6-16
ESC 6-18
exception, divide error 8-9
extended ASCII 4-30
extended codes 4-34

F

FABS 6-26
FADD 6-25
FCHS 6-26
FCOM 6-23
FCOMP 6-23

FCOMPP 6-23
FDIV 6-25
FIFO 4-4
FLD 6-22
FLDLG2 6-24
FLDLN2 6-24
FLDL2T 6-24
FLDP1 6-24
FLDZ 6-24
FLD1 6-24
FMUL 6-25
FORTRAN 8-7
FPATAN 6-26
FPREM 6-26
FREE 6-28
French keyboard 4-41
FRNDINT 6-26
FSCALE 6-26
FSQRT 6-25
FST 6-22
FSTP 6-22
FSUB 6-25
FTST 6-24
function calls, DOS 8-10
function codes, multitasking 8-17
FXAM 6-24
FXCH 6-23
EXTRACT 6-26

G

gap length parameter 8-12
generator, refresh request 1-22
German keyboard 4-42
graphics modes 5-8
guidelines, application 8-7

H

hard code 5-10
hardware compatibility 8-3
hardware interrupts 5-6
HLT 6-17
hooks 8-15

I

I/O address map 1-38
I/O address space 1-24
I/O addresses 1-38
I/O CH CK 1-32, 1-39
I/O CH RDY 1-33
I/O channel 1-24
 connectors 1-25
 pin assignments 1-28
 signals 1-31
I/O channel check 1-32
I/O channel connectors 1-28
I/O channel ready 1-33
I/O chip select 1-36
I/O commands 8-11
I/O CS16 1-36
I/O port (read/write) 1-40
I/O ports, keyboard
 controller 1-58
I/O read 1-33
I/O write 1-33
IDIV 6-9
IIMUL 6-9
IMR 8-13
IMUL 6-8
IN 6-5
INC 6-6
INCSTP 6-28
input buffer, keyboard
 controller 1-56

input port, keyboard
 controller 1-58
input requirements 3-3
input, keyboard 4-29
inputs, power supply 3-3
INS 6-12
instructions
 arithmetic 6-6, 6-25
 comparison 6-23
 constants 6-24
 control transfer 6-13
 data transfer 6-3, 6-22
 logic 6-9
 processor control 6-17
 protection control 6-18
 rotate 6-9
 shift 6-9
 string manipulation 6-11
INT 6-16, 6-27
interfaces, multitasking 8-15
interrupt controller 1-12
interrupt mask register 8-13
interrupt service routine 1-33
interrupt sharing 1-14
interrupt vectors 8-14
interrupt, single step 8-8
interrupts
 BASIC 5-6
 DOS 5-6
 hardware 5-6
 program 5-3
 program interrupt listing
 (real mode) 5-5
 sharing 1-14
 system 1-12
interrupts, program (real
 mode) 5-5
INTO 6-16
IOR 1-33
IOW 1-33
IRET 6-16
IRQ 2 8-11
IRQ 9 8-4, 8-11

IRQ3-IRQ15 1-33
Italian keyboard 4-43

J

JB/JNAE 6-14
JBE/JNA 6-14
JCXZ 6-16
JE/JZ 6-14
JL/JNGE 6-14
JLE/JNG 6-14
JMP 6-13
JNB/JAE 6-15
JNBE/JA 6-15
JNE/JNZ 6-15
JNL/JGE 6-15
JNLE/JG 6-15
JNO 6-15
JNP/JPO 6-15
JNS 6-15
JO 6-14
joystick support 5-6
JP/JPE 6-14
JS 6-14
jumper, RAM 1-43

K

Key Detection Error
command 4-14
key-code scanning 4-4
keyboard 4-48
clock line 1-58
connector 1-75
controller 1-44
controller
commands 1-56
controller I/O ports 1-58
controller input
buffer 1-56

controller input port 1-58
controller output
buffer 1-56
controller output
port 1-58
controller status
register 1-54
controller test inputs 1-58
data line 1-58
encoding 4-30
interface block
diagram 1-54
layout 1-47, 4-31
routine 4-39
keyboard buffer 4-4
keyboard data input 4-29
keyboard data output 4-28
Keyboard ID command 4-14
keyboard layouts
French 4-41
German 4-42
Italian 4-43
keyboard layouts 4-40
Spanish 4-44
U.K. English 4-45
U.S. English 4-46
keyboard logic diagrams 4-48
keyboard scan codes 4-15
keyboard scan-code outputs
scan code set 1 4-16
scan code set 2 4-20
scan code set 3 4-24
keyboard, French 4-41
keyboard, German 4-42
keyboard, Italian 4-43
keyboard, Spanish 4-44
keyboard, U.K. English 4-45
keyboard, U.S. English 4-46
keys 4-4
alternate 4-37
break 4-38
caps lock 4-37
combinations 4-37

- control 4-36
- number lock 4-37
- pause 4-38
- print screen 4-38
- scroll lock 4-37
- shift 4-36
- system request 4-38, 5-6

keys, typematic 4-4

L

- LAHF 6-5
- LAR 6-19
- layout system board 1-76
- layout, keyboard 1-47, 4-31
- layouts
 - French 4-41
 - German 4-42
 - Italian 4-43
 - layouts 4-40
 - Spanish 4-44
 - U.K. English 4-45
 - U.S. English 4-46
- LA17-LA23 1-31
- LDCW 6-27
- LDENV 6-27
- LDS 6-5
- LEA 6-5
- LEAVE 6-16
- LES 6-5
- LGDT 6-18
- LIDT 6-18
- line contention 4-28
- line protocol 4-6
- LLDT 6-18
- LMSW 6-19
- load current 3-3
- LOCK 6-17
- LODS 6-11
- logic diagrams
 - system board 1-77
- logic instructions 6-9

- LOOP 6-15
- loop, busy 8-17
- LOOPNZ/LOOPNE 6-16
- loops, program 8-14
- LOOPZ/LOOPE 6-15
- LSL 6-19
- LTR 6-18

M

- machine identification
 - code 8-19
- machine-sensitive code 8-19
- make code 4-4
- make/break 4-4
- mask on and off 1-39
- MASTER (I) 1-36
- math coprocessor 2-3, 8-11
- math coprocessor
 - controls 1-42
- MEM chip select 1-36
- MEM CS16 1-36
- memory 1-4
- memory decodes 1-12
- memory locations,
 - reserved 5-9
- memory map, BIOS 5-10
- memory module
 - packages 1-24
- MEMR 1-34
- MEMW 1-34
- microprocessor 1-4, 1-7
- microprocessor cycle 1-7
- mode, data stream 1-45, 4-6, 4-27
- modes, graphic 5-8
- modules, RAM 1-24
- modules,
 - ROM/EPROM 1-23
- MOV 6-3
- MOVS 6-11

MUL 6-8
multi-tasking
 function codes 8-17
 interfaces 8-15
 provisions 8-15
 serialization 8-16
 startup 8-16

N

NEG 6-8
NMI 1-12, 1-39
NMI controls 1-39
no load protection 3-4
non-maskable interrupt 1-39
NOP 6-26, 6-28
NOT 6-11
Num Lock state 4-34
number lock key 4-37

O

operations, CMOS RAM
 I/O 1-70
OR 6-10
OSC 1-36
oscillator 1-36
OUT 6-5
output buffer, keyboard
 controller 1-56
output port, keyboard
 controller 1-58
output voltage sense
 levels 3-5
output voltage
 sequencing 3-4
output, keyboard 4-28
outputs, power supply 3-3
OUTS 6-12

Overrun command 4-14

P

packages, memory
 module 1-24
page register addresses 1-11
parameter
 gap length 8-12
 passing 5-4
 tables 8-12
parameters, BIOS fixed
 disk 1-65
Pascal 8-7
pause key 4-38
performance, system 1-7
POP 6-4
POPA 6-4
POPF 6-6, 8-8
POR 4-5
port, diagnostic
 checkpoint 1-42
post 8-17
power good signal 3-4
power requirements 4-47
power supply
 connectors 1-74
 inputs 3-3
 output connectors 3-6
 outputs 3-3
power-on reset 4-5
power-on routine 4-5
 basic assurance test 4-5
 BAT (basic assurance
 test) 4-5
 POR (power-on
 reset) 4-5
 power-on reset 4-5
print screen key 4-38
priorities, shift key 4-37
processor control
 instructions 6-17

program interrupts 5-3
 program loops 8-14
 programming hints,
 BIOS 5-10
 programming,
 coprocessor 2-3
 protected mode 1-5, 5-6
 protection control
 instructions 6-18
 protection, no load 3-4
 protocol 4-6
 provisions, multitasking 8-15
 PTAN 6-26
 PUSH 6-3
 PUSH SP 8-8
 PUSHA 6-4
 PUSHF 6-6

Q

quick reference charts
 BIOS 5-14
 characters keystrokes, and
 colors 7-14

R

R/W memory 1-24
 RAM jumper 1-43
 RAM modules 1-24
 RAM subsystem 1-24
 RAM, CMOS 1-59
 rate, typematic 4-4, 4-11
 Read ID command 4-7
 real address mode 1-4, 2-5
 real mode 5-3
 real-time clock 1-59, 1-60
 record, configuration 1-60
 REFRESH 1-35

refresh controller 1-7
 refresh request
 generator 1-22
 regulation tolerance 3-3
 REP/REPNE,
 REPZ/REPNZ 6-12
 requirements, input 3-3
 Resend command 4-8, 4-14
 reserved memory
 locations 5-9
 reserved scan codes 1-52
 Reset command 4-8
 RESET DRV 1-32
 reset, power-on 4-5
 reset, system 4-38
 RET 6-13
 ROM BIOS 8-10
 ROM BIOS data area 8-14
 ROM scan codes 4-30
 ROM subsystem 1-23
 ROM/EPROM
 modules 1-23
 rotate instructions 6-9
 rotational, speed 8-6
 routine, interrupt
 service 1-33
 routine, keyboard 4-39
 RSTOR 6-28

S

SAHF 6-5
 SAVE 6-28
 SA0-SA19 1-31
 SBB 6-7
 SBHE 1-35
 scan code set 1 4-16
 scan code set 2 4-20
 scan code set 3 4-24
 scan code tables (set 1) 4-16
 scan code tables (set 2) 4-20
 scan code tables (set 3) 4-24

INDEX

scan code translation 1-46
 scan codes, keyboard 4-15
 scan codes, ROM 4-30
 scanning, key-code
 sequencing 4-4
 SCAS 6-11
 scroll lock key 4-37
 SD0-SD15 1-32
 segment address 1-4
 segments 1-4
 Select Alternate Scan Codes
 command 4-8
 sense levels, output
 voltage 3-5
 sequencing key-code
 scanning 4-4
 sequencing, output
 voltage 3-4
 serialization,
 multitasking 8-16
 Set All Keys commands 4-9
 Set Default command 4-9
 Set Key Type commands 4-9
 Set Typematic Rate/Delay
 command 4-11
 Set/Reset Status Indicators
 command 4-10
 SETPM 6-27
 SGDT 6-18
 shift counts 8-9
 shift instructions 6-9
 shift key 4-36
 shift key priorities 4-37
 Shift state 4-34
 shift states 4-36
 SIDT 6-18
 signals
 diskette change 8-6
 I/O channels 1-31
 power good 3-4
 system clock 8-4
 signals, clock and data 4-27
 single step interrupt 8-8
 SLDT 6-18
 SMEMR 1-34
 SMEMW 1-34
 SMSW 6-19
 sound control 8-13
 Spanish keyboard 4-44
 speaker 1-43
 speaker connector 1-75
 speaker tone generation 1-22
 special vectors 5-6
 specifications 4-47
 power requirements 4-47
 size 4-47
 system unit 1-72
 weight 4-47
 startup, multitasking 8-16
 states
 Ctrl 4-34
 Num Lock 4-34
 Shift 4-34, 4-36
 status register, keyboard
 controller 1-54
 STC 6-17
 STCW 6-27
 STD 6-17
 STENV 6-27
 STI 6-17
 STOS 6-12
 STR 6-19
 stream, data 4-27
 string manipulation
 instructions 6-11
 STSW 6-27
 STSWAX 6-27
 SUB 6-7
 subsystem, RAM 1-24
 subsystem, ROM 1-23
 support joystick 5-6
 switch, display 1-44
 system BIOS usage 5-3
 system block diagram xiv
 system board 1-3
 system board block
 diagram 1-6

system board
 connectors 1-74
 system board layout 1-76
 system board logic
 diagrams 1-77
 system bus high enable 1-35
 system clock 1-7
 system clock signal 8-4
 system interrupts 1-12
 system performance 1-7
 system request key 4-38, 5-6
 system reset 4-38
 system timer block
 diagram 1-22
 system timers 1-22

T

T/C 1-35
 table, translation 1-49
 tables, parameter 8-12
 terminal count 1-35
 TEST 6-10
 test inputs, keyboard
 controller 1-58
 time-outs 8-18
 timer/counter 1-22
 timer/counters 1-22
 timers, system 1-22
 tone generation,
 speaker 1-22
 track density, diskette 8-6
 track-to-track access
 time 8-6
 translation table 1-49
 translation, scan code 1-46
 tri-state 1-36
 type of display switch 1-44
 typematic delay 4-4
 typematic keys 4-4
 typematic rate 4-4, 4-11

U

U.K. English keyboard 4-45
 U.S. English keyboard 4-46

V

vectors, special 5-6
 VERR 6-19
 video display buffers 8-14
 virtual address mode 1-4, 2-5

W

WAIT 6-17
 wait condition 8-16
 wait loop classes 8-17
 workspace, default
 segment 5-10
 write current, diskette 8-6

X

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

Y

YL2XP1 6-27

INDEX

Z

zero wait state 1-37

Numerics

OWS 1-37
2XM1 6-26
80286 1-4
8042 1-44
8237A-5 1-9
8254-2 1-22
8259A Interrupt 1-12

Notes:

⌋

⌋

⌋

Notes:

—
)

)

)