IBM

Personal Computer Hardware Reference Library

Technical Reference

1502494

Personal Computer Hardware Reference Library

Technical Reference

First Edition (March 1984)

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 manual "as is," without warranty of any kind, either expressed or implied, including, but not limited to, the particular purpose. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this manual at any time.

This product 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.

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

Products are not stocked at the address below. Requests for copies of this product and for technical information about the system should be made to your authorized IBM Personal Computer dealer.

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

All specifications subject to change without notice.

FEDERAL COMMUNICATIONS COMMISSION RADIO FREQUENCY INTERFERENCE STATEMENT

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

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

Notes:

Preface

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

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

This manual consists of nine sections, four of which describe the hardware aspects of the IBM Personal Computer AT including signal charts and register information. Section 5 contains information about the usage of BIOS and a system BIOS listing. Section 6 contains instruction sets for the 80286 microprocessor and the 80287 math coprocessor. Section 7 provides information about characters, keystrokes, and colors. Section 8 has general communications information. Section 9 contains information about the compatibility of the IBM Personal Computer AT and the rest of the IBM Personal Computer family.

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

Prerequisite Publications

Guide to Operations for the IBM Personal Computer AT

Suggested Reading

- *BASIC* for the IBM Personal Computer
- *Disk Operating System (DOS)*
- *Hardware Maintenance and Service* for the IBM Personal Computer AT
- *MACRO Assembler* for the IBM Personal Computer

Contents

Notes:

INDEX TAB LISTING

SECTION 1

SECTION 2

SECTION 3

SECTION 4

SECTION 5

SECTION 6

Notes:

SECTION 7

SECTION 8

SECTION 9

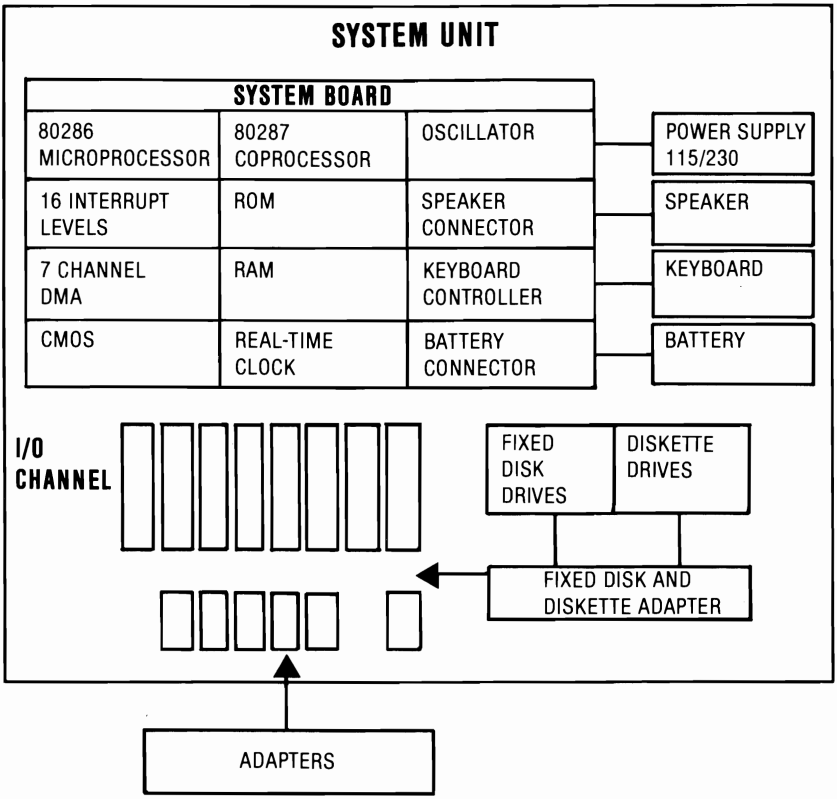
GLOSSARY

BIBLIOGRAPHY

INDEX

Notes:

System Block Diagram



Notes:

SECTION 1. SYSTEM BOARD

Contents

Description

The system board is approximately 30.5 by 33 centimeters (12 by 13 inches) and uses very large scale integration (VLSI) technology. It has the following components:

- Intel 80286 Microprocessor
- System support function:
  - 7-Channel Direct Memory Access (DMA)
  - 16-level interrupt
  - System clock
  - Three programmable timers
- 64Kb read-only memory (ROM) subsystem, expandable to 128Kb
- Either a 256Kb or a 512Kb random-access memory (RAM) Subsystem
- Speaker 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
- Keyboard attachment
- 8 input/output (I/O) slots:
  - 6 with a 36- and a 62-pin card-edge socket.
  - 2 with only the 62-pin card-edge socket.

Memory

The system board has two banks of memory sockets, each supporting 18 128K by 1 modules for a total maximum memory size of 512Kb, with parity checking.

Microprocessor

The Intel 80286 Microprocessor has a 24-bit address, 16-bit memory interface[1], an extensive instruction set, DMA and interrupt support capabilities, a hardware fixed-point multiply and divide, integrated memory management, four-level memory protection, 1-gigabyte (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 64Kb in size and may be read, written, or executed. An exception or interrupt can occur if data operands or instructions attempt to wrap around the end of a segment; for example, a word with its low-order byte at offset FFFF and its high-order byte at 0000. If, in the real-address mode, the information contained in the segment does

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

not use the full 64Kb, the unused end of the segment may be overlayed by another segment to reduce physical memory requirements.

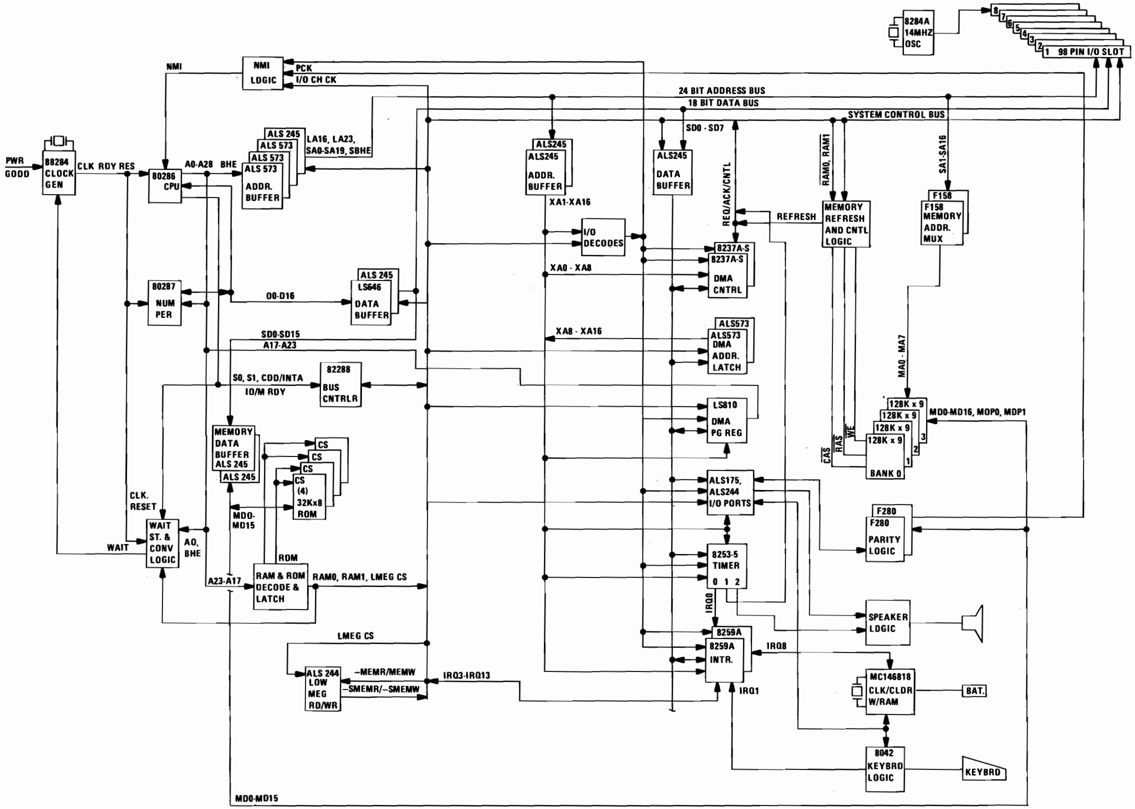
Protected Mode

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

The protected mode provides a 1-gigabyte virtual address space per task mapped into a 16-megabyte 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 tables are automatically referenced by the microprocessor whenever a segment register is loaded with a selector. All instructions that load a segment register will refer to the memory based-tables without additional program support. The memory-based tables contain 8-byte values called *descriptors*.

Following is a block diagram of the system board.



System Board Block Diagram

System Performance

The 80286 Microprocessor operates at 6 MHz, which results in a clock cycle time of 167 nanoseconds.

A bus cycle requires three clock cycles (which includes 1 wait state) so that a 500-nanosecond, 16-bit, microprocessor cycle time is achieved. 8-bit bus operations to 8-bit devices take 6 clock cycles (which include 4 wait states), resulting in a 1000-nanosecond microprocessor cycle. 16-bit bus operations to 8-bit devices take 12 clock cycles (which include 10 I/O wait states) resulting in a 2000 nanosecond microprocessor cycle.

The refresh controller operates at 6 MHz. Each refresh cycle requires 5 clock cycles to refresh all of the system's dynamic memory; 256 refresh cycles are required every 4 milliseconds. The following formula determines the percent of bandwidth used for refresh.

```
% Bandwidth used     5 cycles X 256    1280
   for Refresh     = -------------- = ----- = 5.3%
                      4 ms/167 ns      24000
```

The DMA controller operates at 3 MHz, which results in a clock cycle time of 333 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.

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

The following figure is a system memory map.

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

System Memory Map

System Timers

The system has three programmable timer/counters controlled by an Intel 8254-2 timer/counter chip and defined as Channels 0 through 2 as follows:

Channel 0	**System Timer**
GATE 0	Tied on
CLK IN 0	1.190 MHz OSC
CLK OUT 0	8259A IRQ 0
Channel 1	**Refresh Request Generator**

GATE 1 Tied on

CLK IN 1 1.190 MHz OSC

CLK OUT 1 Request Refresh Cycle

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

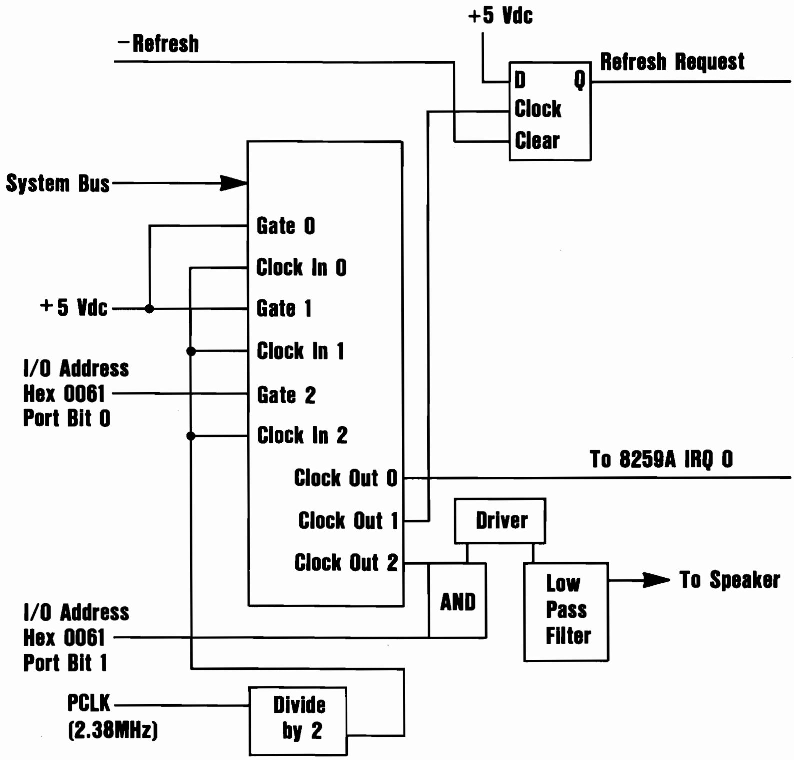
Channel 2 **Tone Generation for Speaker**

GATE 2 Controlled by bit 0 of port hex 61 PPI bit

CLK IN 2 1.190 MHz OSC

CLK OUT 2 Used to drive the speaker

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



System Interrupts

The 80286 Microprocessor NMI and two 8259A Interrupt Controller chips provide 16 levels of system interrupts. The following shows the interrupt-level assignments in decreasing priority.

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

Level		Function
MicroProcessor NMI		Parity or I/O Channel Check
Interrupt Controllers		
CTLR 1	CTLR 2	
IRQ 0		Timer Output 0
IRQ 1		Keyboard (Output Buffer Full)
IRQ 2		Interrupt from CTLR 2
	IRQ 8	Realtime Clock Interrupt
	IRQ 9	Software Redirected to INT ØAH (IRQ 2)
	IRQ 10	Reserved
	IRQ 11	Reserved
	IRQ 12	Reserved
	IRQ 13	Coprocessor
	IRQ 14	Fixed Disk Controller
	IRQ 15	Reserved
IRQ 3		Serial Port 2
IRQ 4		Serial Port 1
IRQ 5		Parallel Port 2
IRQ 6		Diskette Controller
IRQ 7		Parallel Port 1

ROM Subsystem

The system board's ROM subsystem consists of two 32K by 8-bit ROM/EPROM modules or four 16K 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 (hex 0F0000 and hex FF0000). ROM is not parity-checked. Its access time is 150 nanoseconds and its cycle time is 230 nanoseconds.

RAM Subsystem

The system board's RAM subsystem starts at address hex 000000 of the 16M address space. It consists of either 256Kb or 512Kb of 128K by 1-bit RAM modules. 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.

Direct Memory Access (DMA)

The system supports seven DMA channels. Two Intel 8237A-5 DMA Controller Chips are used, with four channels for each chip. The DMA channels are assigned as follows:

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

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 16-megabyte system-address space in 64Kb blocks.

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 16-megabyte system-address space in 128Kb blocks. Channels 5, 6, and 7 cannot transfer data on odd byte boundaries.

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

The following figures show address generation for the DMA channels.

Source	DMA Page Registers	8237A-5
Address	A23<---------->A16	A15<---------->A0

Address Generation for DMA Channels 3 through 0

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

Source	DMA Page Registers	8237A-5
Address	A23<---------->A17	A16<---------->A1

Address Generation for DMA Channels 7 through 5

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

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

Programming the 16-Bit DMA Channels

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 (8237A-5), which controls these channels, is through I/O addresses 0C0 through 0DF. The command codes for the DMA controller are as follows:

Hex Address	Command Codes
0C0	CH0 base and current address
0C2	CH0 base and current word count
0C4	CH1 base and current address
0C6	CH1 base and current word count
0C8	CH2 base and current address
0CA	CH2 base and current word count
0CC	CH3 base and current address
0CE	CH3 base and current word count
0D0	Read Status Register/Write Command Register
0D2	Write Request Register
0D4	Write Single Mask Register Bit
0D6	Write Mode Register
0D8	Clear Byte Pointer Flip-Flop
0DA	Read Temporary Register/Write Master Clear
0DC	Clear Mask Register
0DE	Write All Mask Register Bits

DMA Controller Registers

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 that is 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 16Mb memory space into 128Kb pages. When the DMA page registers for channels 5 through 7 are programmed, data bits D7 through D1 should 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.

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

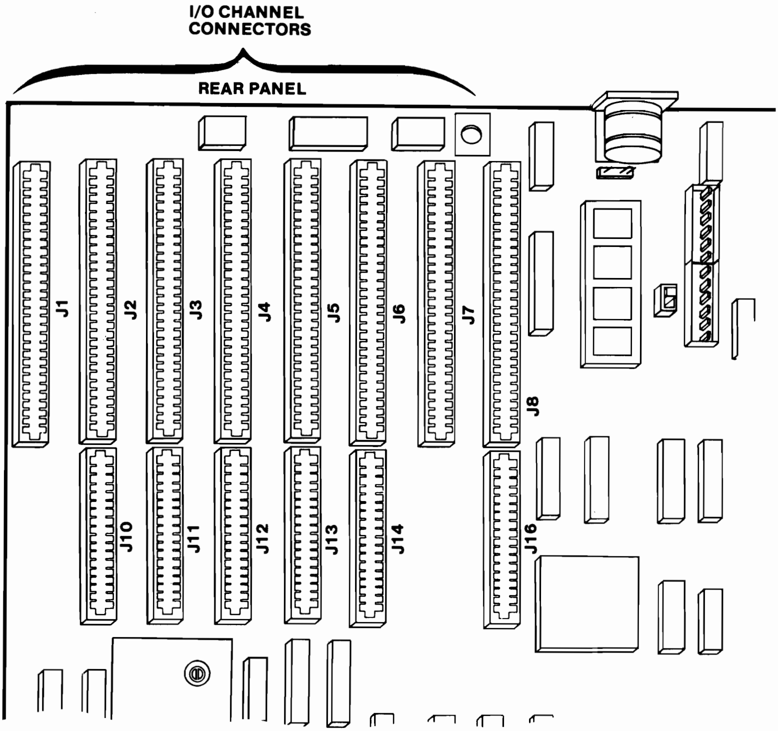
I/O Channel

The I/O channel supports:

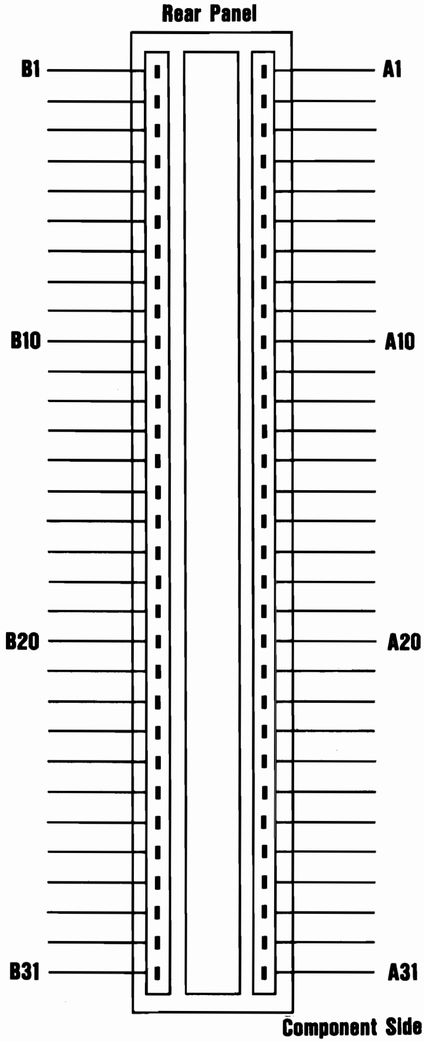
- I/O address space hex 100 to hex 3FF
- 24-bit memory addresses (16Mb)
- Selection of data accesses (either 8- or 16-bit)
- Interrupts
- DMA channels
- I/O wait-state generation
- Open-bus structure (allowing multiple microprocessors to share the system's resources, including memory)
- Refresh of system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

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

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

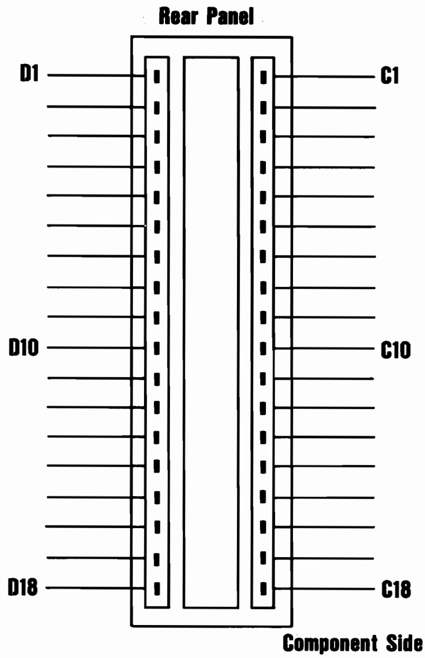

I/O CHANNEL
CONNECTORS
REAR PANEL
J1
J2
J3
J4
J5
J6
J7
J8
J10
J11
J12
J13
J14
J16

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



I/O Channel Pin Numbering (J1-J8)

The following figure shows the pin numbering for I/O channel connectors J12 through J16 and J18.



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

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

I/O Pin	Signal Name	I/O
A 1	-I/O CH CK	I
A 2	SD7	I/O
A 3	SD6	I/O
A 4	SD5	I/O
A 5	SD4	I/O
A 6	SD3	I/O
A 7	SD2	I/O
A 8	SD1	I/O
A 9	SD0	I/O
A 10	-I/O CH RDY	I
A 11	AEN	O
A 12	SA19	I/O
A 13	SA18	I/O
A 14	SA17	I/O
A 15	SA16	I/O
A 16	SA15	I/O
A 17	SA14	I/O
A 18	SA13	I/O
A 19	SA12	I/O
A 20	SA11	I/O
A 21	SA10	I/O
A 22	SA9	I/O
A 23	SA8	I/O
A 24	SA7	I/O
A 25	SA6	I/O
A 26	SA5	I/O
A 27	SA4	I/O
A 28	SA3	I/O
A 29	SA2	I/O
A 30	SA1	I/O
A 31	SA0	I/O

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

I/O Pin	Signal Name	I/O
B 1	GND	Ground
B 2	RESET DRV	O
B 3	+5 Vdc	Power
B 4	IRQ 9	I
B 5	-5 Vdc	Power
B 6	DRQ2	I
B 7	-12 Vdc	Power
B 8	OWS	I
B 9	+12 Vdc	Power
B 10	GND	Ground
B 11	-SMEMW	O
B 12	-SMEMR	O
B 13	-IOW	I/O
B 14	-IOR	I/O
B 15	-DACK3	O
B 16	DRQ3	I
B 17	-DACK1	O
B 18	DRQ1	I
B 19	-Refresh	I/O
B 20	CLK	O
B 21	IRQ7	I
B 22	IRQ6	I
B 23	IRQ5	I
B 24	IRQ4	I
B 25	IRQ3	I
B 26	-DACK2	O
B 27	T/C	O
B 28	BALE	O
B 29	+5 Vdc	Power
B 30	OSC	O
B 31	GND	Ground

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

I/O Pin	Signal Name	I/O
C 1	SBHE	I/O
C 2	LA23	I/O
C 3	LA22	I/O
C 4	LA21	I/O
C 5	LA20	I/O
C 6	LA19	I/O
C 7	LA18	I/O
C 8	LA17	I/O
C 9	-MEMR	I/O
C 10	-MEMW	I/O
C 11	SD08	I/O
C 12	SD09	I/O
C 13	SD10	I/O
C 14	SD11	I/O
C 15	SD12	I/O
C 16	SD13	I/O
C 17	SD14	I/O
C 18	SD15	I/O

I/O Channel (C-Side J10 through J14 and J16)

I/O Pin	Signal Name	I/O
D 1	-MEM CS16	I
D 2	-I/O CS16	I
D 3	IRQ10	I
D 4	IRQ11	I
D 5	IRQ12	I
D 6	IRQ15	I
D 7	IRQ14	I
D 8	-DACK0	O
D 9	DRQ0	I
D 10	-DACK5	O
D 11	DRQ5	I
D 12	-DACK6	O
D 13	DRQ6	I
D 14	-DACK7	O
D 15	DRQ7	I
D 16	+5 Vdc	Power
D 17	-MASTER	I
D 18	GND	Ground

I/O Channel (D-Side, J10 through J14 and J16)

I/O Channel Signal Description

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

SA0 through SA19 (I/O)

Address bits 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 16Mb of memory. SA0 through SA19 are gated on the system bus when '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 16Mb of addressability. These signals are valid when 'BALE' is high. LA17 through LA23 are not latched during microprocessor cycles and therefore do not stay valid for the whole cycle. Their purpose is to generate memory decodes for 1 wait-state memory cycles. These decodes should be latched by I/O adapters on the falling edge of 'BALE.' These signals also may be driven by other microprocessors or DMA controllers that reside on the I/O channel.

CLK (0)

This is the 6-MHz system clock. It is a synchronous microprocessor cycle clock with a cycle time of 167 nanoseconds. The clock has a 50% duty cycle. This signal should only be used for synchronization. It is not intended for uses requiring a fixed frequency.

SECTION 1

RESET DRV (0)

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

SD0 through SD15 (I/O)

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

BALE (0) (buffered)

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

-I/O CH CK (I)

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

I/O CH RDY (I)

'I/O channel ready' 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 and IRQ 14 through 15 (I)

Interrupt Requests 3 through 7, 9 through 12, and 14 through 15 are used to signal the microprocessor that an I/O device needs attention. The interrupt requests are prioritized, with IRQ9 through IRQ12 and IRQ14 through 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 must be held high until the microprocessor acknowledges the interrupt request (Interrupt Service routine). Interrupt 13 is used on the system board and is not available on the I/O channel. Interrupt 8 is used for the real-time clock.

-IOR (I/O)

'-I/O Read' instructs an I/O device to drive its data onto the data bus. It 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)

'-I/O Write' instructs an I/O device to read the data on 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 1Mb 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 1Mb 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 system clock period 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 1Mb of the memory space. '-MEMW' is active on all memory read 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 1Mb 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 system clock period before driving '-MEMW' active. Both signals are active low.

DRQ0-DRQ3 and DRQ5-DRQ7 (I)

DMA Requests 0 through 3 and 5 through 7 are asynchronous channel requests used by peripheral devices and the I/O channel microprocessors to gain DMA service (or control of the system). They are prioritized, with 'DRQ0' having the highest priority and 'DRQ7' having the lowest. A request is generated by bringing a DRQ line to an active level. A DRQ line must be held high until the corresponding 'DMA Request Acknowledge' (DACK) line goes active. 'DRQ0' through 'DRQ3' will perform 8-bit DMA transfers; 'DRQ5' through 'DRQ7' will 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 to 3 and 5 to 7 are used to acknowledge DMA requests (DRQ0 through DRQ7). They are active low.

AEN (O)

'Address Enable' 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).

-REFRESH (I/O)

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

T/C (O)

'Terminal Count' provides a pulse when the terminal count for any DMA channel is reached.

SBHE (I/O)

'Bus High Enable' (system) 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.

-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', an I/O microprocessor may pull '-MASTER' low, which will allow it to

UPDATE NUMBER 4

IBM TECHNICAL REFERENCE UPDATE

Note: All previous Technical Reference updates must be installed before installing this update.

This package contains updates to the IBM Personal Computer AT *Technical Reference* manual.

IBM PERSONAL COMPUTER AT TECHNICAL REFERENCE UPDATE

The following pages update the *Technical Reference* for the IBM Personal Computer AT.

Replace or add the update pages as instructed below.

1-27 and 1-28	Replace
1-28.1 and 1-28.2	Replace

Discard this instruction sheet.

control the system address, data, and control lines (a condition known as *tri-state*). After '-MASTER' is low, the I/O microprocessor must wait one system clock period before driving the address and data lines, and two clock periods before issuing a Read and Write command. If this signal is held low for more than 15 microseconds, system memory may be lost because of a lack of refresh.

-MEM CS16 (I)

'-MEM 16 Chip Select' signals the system board if 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' should be driven with an open collector or tri-state driver capable of sinking 20 mA.

-I/O CS16 (I)

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

OSC (O)

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

0WS (I)

The 'Zero Wait State' (0WS) signal tells the microprocessor that it can complete the present bus cycle without inserting any additional wait cycles. In order to run a memory cycle to a 16-bit device without wait cycles, '0WS' is derived from an address decode gated with a Read or Write command. In order to run a memory cycle to an 8-bit device with a minimum of two wait states, '0WS' should be driven active one system clock after the

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

The following figure is an I/O address map.

Hex Range*	Usage
000-01F	DMA controller 1, 8237A-5
020-03F	Interrupt controller 1, 8259A, Master
02E1	GPIB (Adapter 0)
02E2 & 02E3	Data Acquisition (Adapter 0)
040-05F	Timer 8254.2
060-06F	8042 (Keyboard)
06E2 & 06E3	Data Acquisition (Adapter 1)
070-07F	Real-time clock, NMI (non-maskable interrupt) mask
080-09F	DMA page registers, 74LS612
0A0-0BF	Interrupt controller 2, 8259A
0AE2 & 0AE3	Data Acquisition (Adapter 2)
0C0-0DF	DMA controller 2,8237A-5
0EE2 & 0EE3	Data Acquisition (Adapter 3)
0F0	Clear Math Coprocessor Busy
0F1	Reset Math Coprocessor
0F8-0FF	Math Coprocessor
1F0-1F8	Fixed Disk
200-207	Game I/O
22E1	GPIB (Adapter 1)
278-27F	Parallel printer port 2
2B0-2DF	Alternate Enhanced Graphics Adapter
2F8-27F	Serial port 2

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 base addresses for GPIB and Data Acquisition are shown.

I/O Address Map (Part 1 of 2)

Hex Range*	Usage
300-31F	Prototype card
360-36F	P C Network
378-37F	Parallel printer port 1
380-38F	SDLC, bisynchronous 2
390-393	Cluster
3A0-3AF	Bisynchronous 1
3B0-3BF	Monochrome Display and Printer Adapter
3C0-3CF	Enhanced Graphics Adapter
3D0-3DF	Color/Graphics Monitor Adapter
3F0-3F7	Diskette controller
3F8-3FF	Serial port 1
42E1	GPIB (Adapter 2)
62E1	GPIB (Adapter 3)
790-793	Cluster (Adapter 1)
82E1	GPIB (Adapter 4)
A2E1	GPIB (Adapter 5)
B90-B93	Cluster (Adapter 2)
C2E1	GPIB (Adapter 6)
E2E1	GPIB (Adapter 7)
1390-1393	Cluster (Adapter 3)
2390-2393	Cluster (Adapter 4)

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 base addresses for GPIB and Data Acquisition are shown.

I/O Address Map (Part 2 of 2)

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

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

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

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

The following is a description of the Math Coprocessor controls.

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

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

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

The '-I/O channel check signal' (-I/O CH CK) is used to report uncorrectable errors on RAM adapters on the I/O channel. This check will create a non-maskable interrupt (NMI) if enabled (see the figure, "I/O Address Map," for enable control). At power-on time, the NMI is masked off and check is disabled. Before check or NMI is enabled, the following steps should be taken.

1. Write data in all I/O RAM-adapter memory locations; this will establish good parity at all locations.

2. Enable I/O channel check.

3. Enable NMI.

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

When a check occurs, an interrupt (NMI) will result. Check the status bits to determine the source of the NMI (see the figure, "I/O Address Map"). To determine the location of the failing adapter, write to any memory location within a given adapter. If the parity check was from that adapter, '-I/O CH CK' will be inactive.

Other Circuits

Speaker

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

- The I/O-port output bit
- The timer/counter's clock out
- Both

Jumper

The system board has a 3-pin, Berg-strip connector. The placement of a jumper across the pins of the connector determines whether the system board's 2nd 256Kb of RAM is enabled or disabled. Following are the pin assignments for the connector.

Pin	Assignments
1	No connection
2	Ground
3	A8 (28S42)

RAM Jumper Connector(J18)

The following shows how the jumper affects RAM.

Jumper Positions	Function
1 and 2	Enable 2nd 256Kb of system board ram
2 and 3	Disable 2nd 256Kb of system board ram

RAM Jumper

Note: The normal mode is the enable mode. The disable mode permits the 2nd 256Kb of RAM to reside on adapters plugged into the I/O bus.

Type of Display Adapter Switch

The system board has a slide switch, the purpose of which is to tell the system into which display adapter the primary display is attached. Its positions are assigned as follows:

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

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

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

Variable Capacitor

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

Keyboard Controller

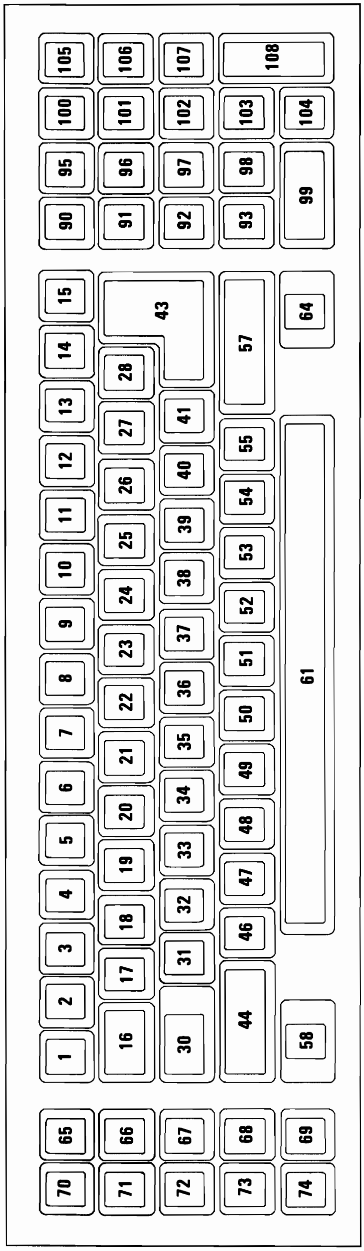
The keyboard controller is a single-chip microcomputer (Intel 8042) that is programmed to support the IBM Personal Computer AT Keyboard serial interface. The keyboard controller receives serial data from the keyboard, checks the parity of the data, translates scan codes, and presents the data to the system as a byte of data in its output buffer. The controller will interrupt the system when data is placed in its output buffer. The status register contains bits that indicate if an error was detected while receiving the data. Data may be sent to the keyboard by writing to the keyboard controller's input buffer. The byte of data will be sent to the keyboard serially with an odd parity bit automatically inserted. The keyboard is required to acknowledge all data transmissions. No transmission should be sent to the keyboard until acknowledgment is received for the previous byte sent.

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, a hex FF is placed in its output buffer, and the parity bit in the status register is set to 1, indicating a receive parity error. The keyboard controller will also time a byte of data from the keyboard. If a keyboard transmission does not end within two milliseconds, a hex FF is placed in the keyboard controller's output buffer, and the receive time-out bit in the status register is set. No retries will be attempted on a receive time-out error.

Scan Code Translation

Scan codes, which are received from the keyboard, are converted by the keyboard controller before they are put into the controller's output buffer. The following figure shows the keyboard layout with key numbers.



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
30
31
32
33
34
35
36
37
38
39
40
41
43
44
46
47
48
49
50
51
52
53
54
55
57
58
61
64
65
66
67
68
69
70
71
72
73
74
90
91
92
93
95
96
97
98
99
100
101
102
103
104
105
106
107
108

The following figure is the scan-code translation table.

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

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

Keyboard Scan Code	Key	System Scan Code
32	50	30
31	51	31
3A	52	32
41	53	33
49	54	34
4A	55	35
59	57	36
7C	106	37
11	58	38
29	61	39
58	64	3A
05	70	3B
06	65	3C
04	71	3D
0C	66	3E
03	72	3F
0B	67	40
02 or 83	73	41
0A	68	42
01	74	43
09	69	44
77	95	45
7E	100	46
6C	91	47
75	96	48
7D	101	49
7B	107	4A
6B	92	4B
73	97	4C
74	102	4D
79	108	4E
69	93	4F
72	98	50
7A	103	51
70	99	52
71	104	53
7F or 84	105	54

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

The following scan codes are reserved.

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

Scan-Code Translation Table

Sending Data to the Keyboard

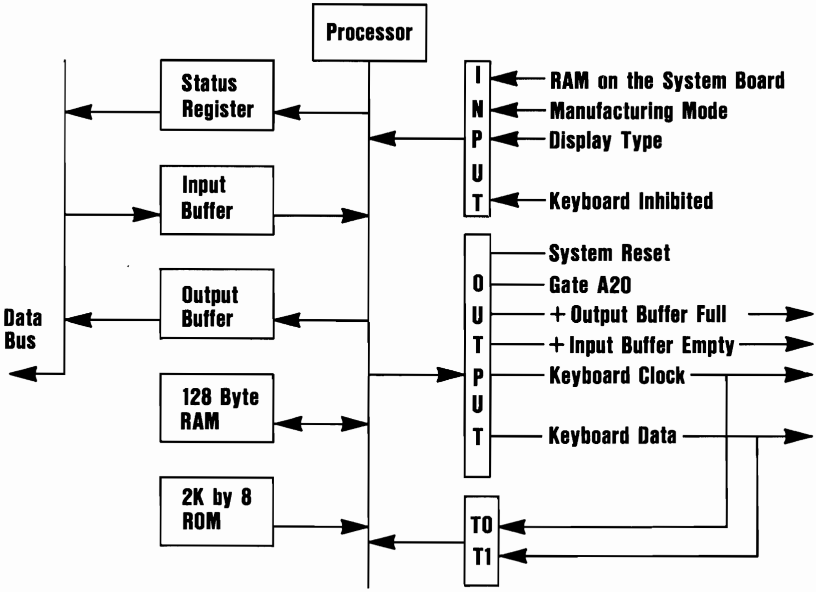
Data is sent to the keyboard in the same serial format used to receive data from the keyboard. A parity bit is automatically inserted by the keyboard controller. If the keyboard does not start clocking the data out of the keyboard controller within 15 milliseconds or complete that clocking within 2 milliseconds, 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. If the response contains a parity error, a hex FE is placed in the keyboard controller's 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 time limit for the keyboard to respond. If 25 milliseconds are exceeded, the keyboard controller places a hex FE in its output buffer and sets the transmit and receive time-out error bits in the status register. No retries will be made by the keyboard controller for any transmission error.

Inhibit

The keyboard interface may be inhibited by a key-controlled hardware switch, although all transmissions to the keyboard will be allowed, regardless of the state of the switch. The keyboard controller tests data received from the keyboard to determine if the byte received is a command response or a scan code. If the byte is a command response, it is placed in the keyboard controller's output buffer. If the byte is a scan code, it is ignored.

Keyboard Controller System Interface

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



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 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.

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 2 System Flag—This bit may be set to 0 or 1 by writing to the system's flag bit in the keyboard controller's command byte. It is set to 0 after a power on reset.

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

Bit 5 Transmit Time-Out—A 1 indicates that a transmission started by the keyboard controller was not properly completed. If the transmit byte was not clocked out within the specified time limit, this will be the only error. If the transmit byte was clocked out but a response was not received within the programmed time limit, the transmit time-out and receive time-out error bits are set On. 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 On.

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 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 with odd parity.

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's 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, that 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 equal to 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 to 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 two-byte break sequence to the one-byte IBM Personal Computer format.

Bit 5 IBM Personal Computer Mode—Writing a 1 to this bit programs the keyboard to support the IBM Personal Computer keyboard interface. In this mode the controller does not check parity or convert scan codes.

Bit 4 Disable Keyboard—Writing a 1 to this bit disables the keyboard interface by driving the 'clock' line low. Data is not sent or received.

Bit 3 Inhibit Override—Writing a 1 to this bit disables the keyboard inhibit function.

Bit 2 System Flag—The value written to this bit is placed in the system flag bit of the controller's status register.

Bit 1 Reserved—Should be written to 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 data lines. The test result is placed in the output buffer as follows:

00 No error detected.

01 The 'keyboard clock' line is stuck low.

02 The 'keyboard clock' line is stuck high.

03 The 'keyboard data' line is stuck low.

04 The 'keyboard data' line is stuck high.

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

AD Disable Keyboard Feature—This command sets bit 4 of the controller's command byte. This disables the keyboard interface by driving the clock line low. Data will not be sent or received.

AE Enable Keyboard Interface—This command clears bit 4 of the command byte, which releases the keyboard interface.

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

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

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

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

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 8-bit I/O ports and two test inputs. One of the ports is assigned for input and the other for output. The controller uses the test inputs to read the state of the keyboard's 'clock' line and the keyboard's 'data' line.

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

Bit 0	Undefined
Bit 1	Undefined
Bit 2	Undefined
Bit 3	Undefined
Bit 4	RAM on the system board 0 = Disable 2nd 256Kb of system board RAM 1 = Enable 2nd 256Kb of system board RAM
Bit 5	Manufacturing jumper 0 = Manufacturing jumper installed 1 = Jumper not installed
Bit 6	Display type switch 0 = Primary display attached to Color/Graphics adapter 1 = Primary display attached to Monochrome adapter
Bit 7	Keyboard inhibit switch 0 = Keyboard inhibited 1 = Keyboard not inhibited

Input-Port Definitions

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

Output-Port Bit Definitions

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

Test-Input Port Bit Definitions

Real-time Clock/Complementary Metal Oxide Semiconductor (RT/CMOS) RAM Information

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

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

CMOS RAM Address Map

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

Real-time Clock Information

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

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

Real-Time Clock Information (addresses 00–0D)

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

Status Register A

Bit 7 Update in Progress (UIP)—A 1 indicates the time update cycle is in progress. A 0 indicates the current date and time is 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 32.768kHz 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 1.024kHz 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 establishes 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.

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' interrupts are enabled in register B.

Bit 3–Bit 0 Reserved

Register D

Bit 7 Valid RAM Bit (VRB)—This bit is read only and indicates the condition of the contents of the CMOS RAM through the power sense pin. A low state of the power sense pin indicates that the real-time clock has lost its power (battery dead). A 1 on the VRB indicates power on the real-time clock and a 0 indicates that the real-time clock has lost power.

Bits 6–Bit 0 Reserved

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 Real-time clock chip has lost power. A 0 indicates that the chip has not lost power, and a 1 indicates that the chip lost power.

Bit 6	Configuration Record—Checksum Status Indicator—A 0 indicates that checksum is good, and a 1 indicates it is bad.
Bit 5	Incorrect Configuration Information—This is a check, at power on time, of the equipment byte of the configuration record. A 0 indicates that the configuration information is valid, and a 1 indicates it is invalid. Power-on checks require: • At least one diskette drive to be installed (bit 0 of the equipment byte set to 1). • The primary display adapter setting in configuration matches the system board's display switch setting and the actual display hardware in the system.
Bit 4	Memory Size Miscompare—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 the time is invalid.
Bit 1–Bit 0	Reserved

Shutdown Status Byte (Hex 0F)

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

Diskette Drive Type Byte (Hex 10)

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

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

  Note: 0011 through 1111 are reserved.

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

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

  Note: 0011 through 1111 are reserved.

Hex address 11 contains a reserved byte.

Fixed Disk Type Byte (Hex 12)

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

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

0001 through 1111 define type 1 through type 15 (see BIOS listing at label FD__TBL).

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

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

0001 through 1111 define type 1 through type 15 (see BIOS listing at label FD__TBL).

The following figure shows the BIOS fixed disk parameters.

Type	Cylinders	Heads	Write Pre-comp	Landing Zone
1	306	4	128	305
2	615	4	300	615
3	615	6	300	615
4	940	8	512	940
5	940	6	512	940
6	615	4	no	615
7	462	8	256	511
8	733	5	no	733
9	900	15	no8	901
10	820	3	no	820
11	855	5	no	855
12	855	7	no	855
13	306	8	128	319
14	733	7	no	733
15	Reserved--set to zeros			

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 Reserved

01 Primary display is attached to the Color/Graphics Monitor Adapter in the 40-column mode.

10 Primary display is attached to the Color/Graphics Monitor Adapter in the 80-column mode.

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 The set condition of this bit indicates that diskette drives are installed.

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

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

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

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

Valid Sizes:

0100H 256Kb system-board RAM

0200H 512Kb system-board RAM

0280H 640Kb 512Kb system board RAM and the IBM Personal Computer AT 128KB Memory Expansion Option

Low and High Memory Expansion 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	512Kb I/O adapter
0400H	1024Kb I/O adapter (2 adapters)
600H	1536Kb I/O adapter (3 adapters)
to	
3C00H	15360Kb I/O adapter (15Mb maximum)

Hex addresses 19 through 2D are reserved.

Checksum (Hex 2E and 2F)

Address hex 2E	High byte of checksum
Address hex 2F	Low byte of checksum

Note: Checksum is on addresses hex 10-20.

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	512Kb I/O adapter
0400H	1024Kb I/O adapter

0600H	1536Kb I/O adapter
to	
3C00H	15360Kb I/O adapter (15Mb maximum)

Note: This word reflects the total expansion memory above the 1Mb 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.

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	Set if the IBM Personal Computer AT 128KB Memory Expansion Option is installed.
Bit 6	This bit is used by the Setup utility to put out a first user message after initial setup.
Bit 5–Bit 0	Reserved

Note: Hex addresses 34 through 3F are reserved.

I/O Operations

Writing to CMOS RAM involves two steps:

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

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

Reading CMOS RAM also requires two steps:

1. OUT to port hex 70 with the CMOS address that is to be read from.

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

Specifications

System Unit

Size

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

Weight

- 19.05 kilograms (42 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)
- Humidity
  - System On: 8% to 80%
  - System Off: 20% to 80%
- Altitude
  - Maximum altitude: 2133.6 meters (7000 feet)

Heat Output

- 1229 British Thermal Units per hour

Noise Level

- Meets Class 3; 42 decibels average-noise rating

Electrical

- VA — 450
- Range 1
  - Nominal - 115 Vac
  - Minimum - 100 Vac

- Maximum - 125 Vac

- Range 2
  - Nominal - 230 Vac
  - Minimum - 200 Vac
  - Maximum - 240 Vac

Connectors

The system board has the following connectors:

- Speaker connector (J19)
- Two power-supply connectors (PS8 and PS9)
- Keyboard connector (J9)
- Power LED and keylock connector (J20)
- Battery connector (J21)

The speaker connector is a 4-pin, keyed, Berg strip. The pin assignments follow.

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

Speaker Connector (J19)

The pin assignments for power-supply connectors, P8 and P9, are as follows:

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

Power Supply Connectors

The keyboard connector is a 5-pin, 90-degree Printed Circuit Board (PCB) mounting, DIN connector. The pin assignments are as follows:

Pin	Assignments
1	Keyboard clock
2	Keyboard data
3	Spare
4	Ground
5	+5 Vdc

Keyboard Connector (J22)

The power LED and keylock connector is a 5-pin Berg strip. Its pin assignments follow:

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

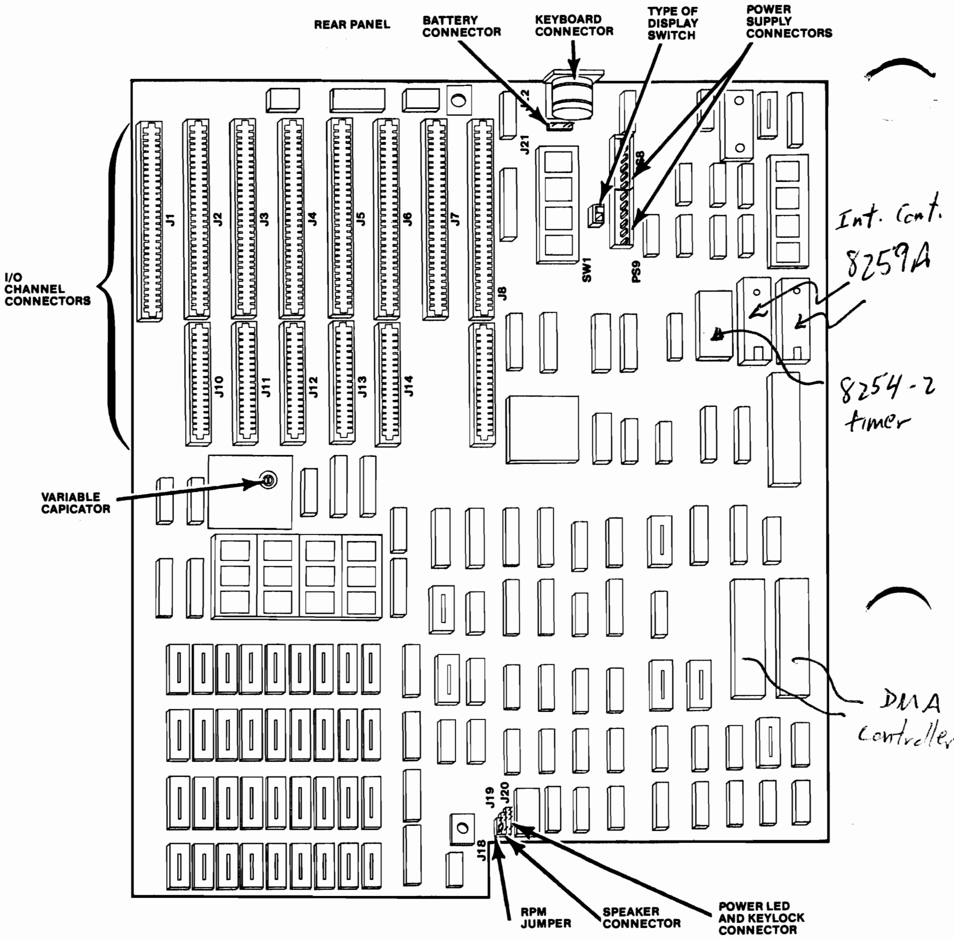
Power LED and Keylock Connector (J20)

The battery connector is a 4-pin, keyed, Berg strip. The pin assignments follow:

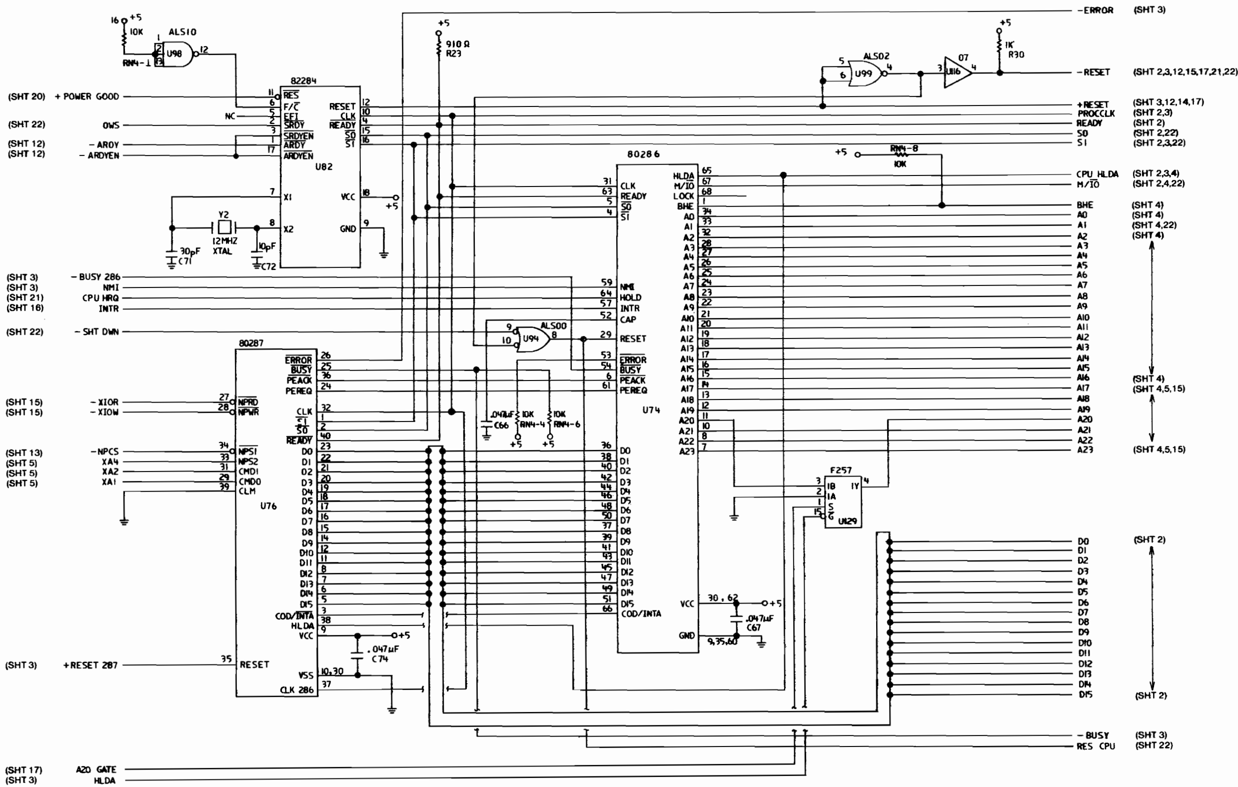
Pin	Assignments
1	Ground
2	Not Used
3	Not Used
4	6 Vdc.

Battery Connector (J21)

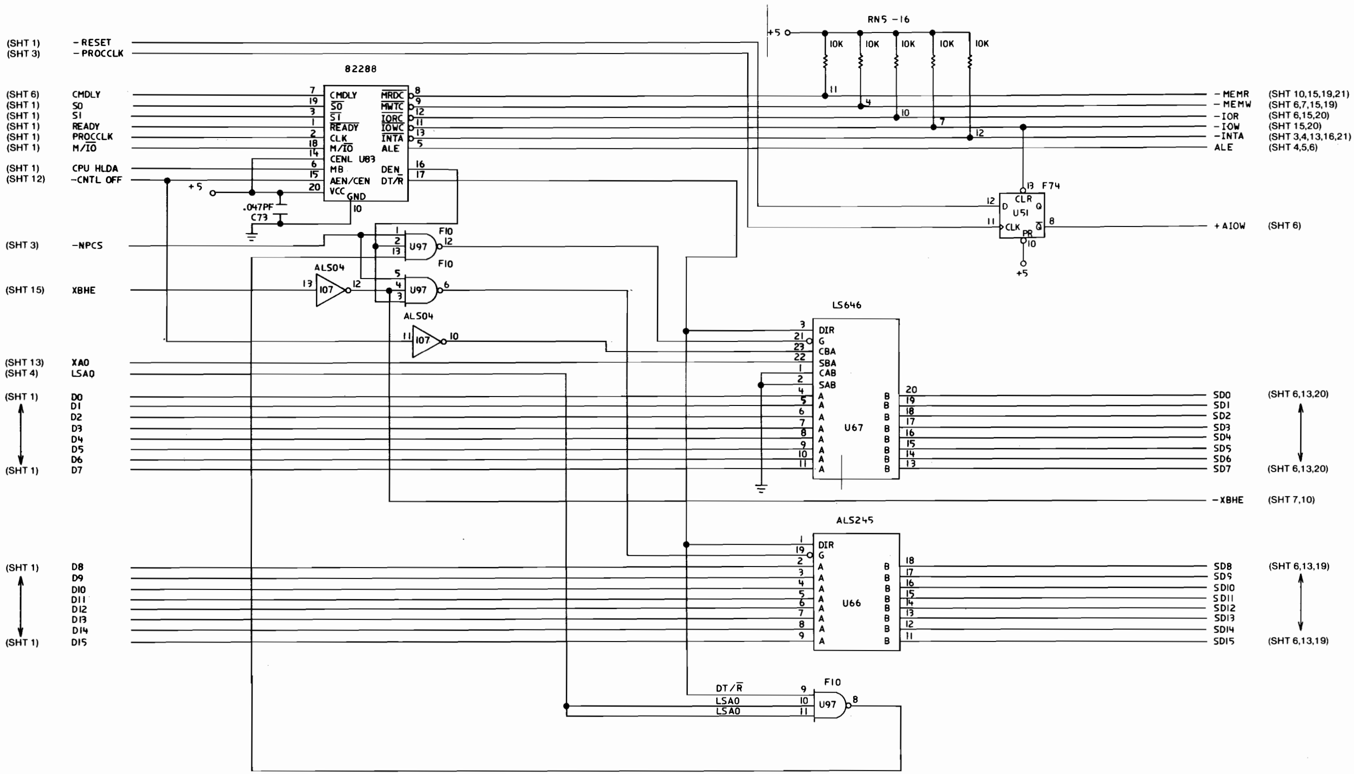
The following figure shows the layout of the system board.



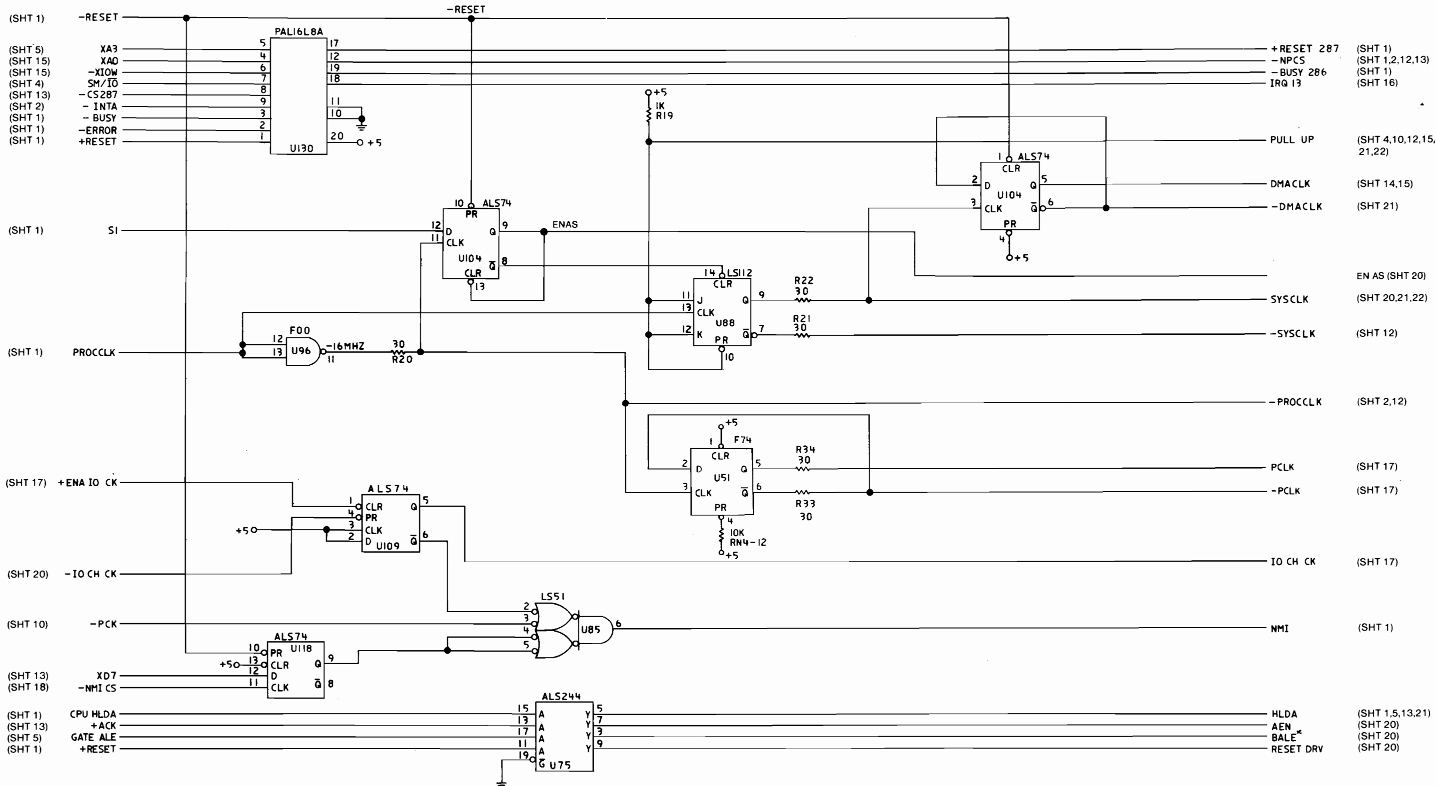
System Board Layout



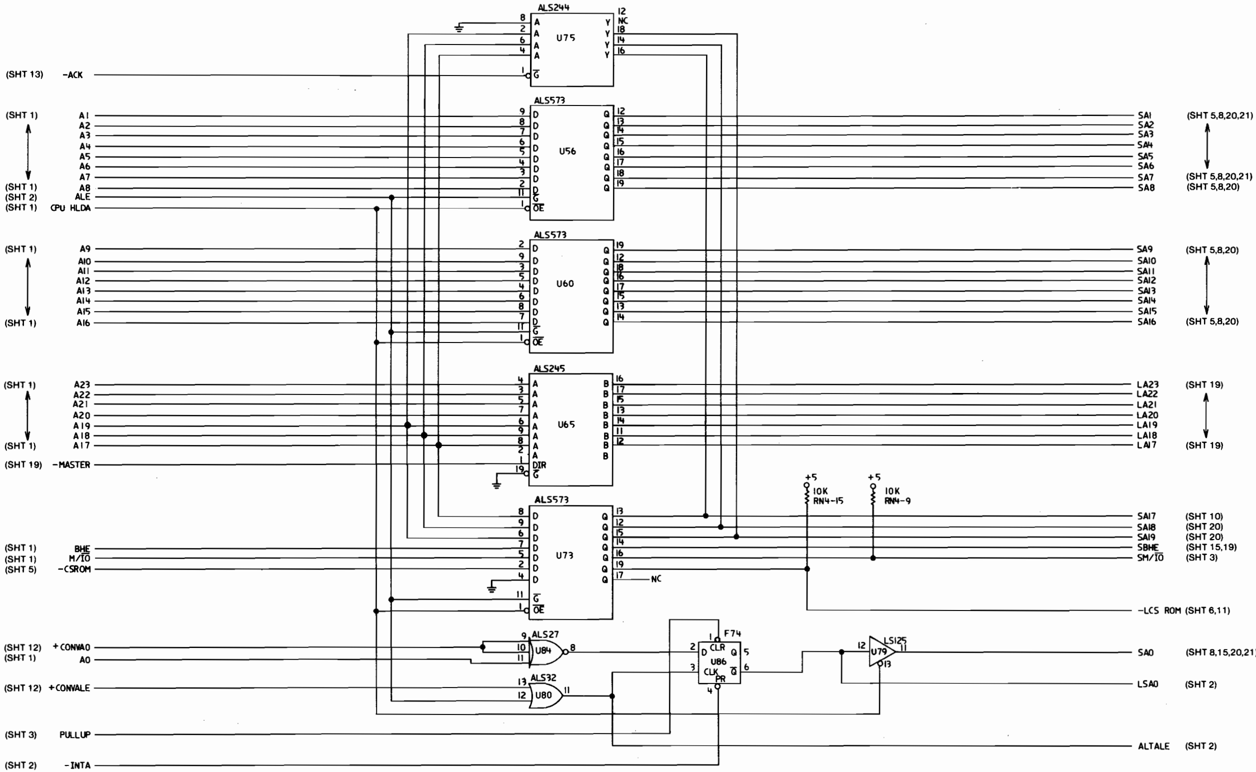
System Board (Sheet 1 of 22)



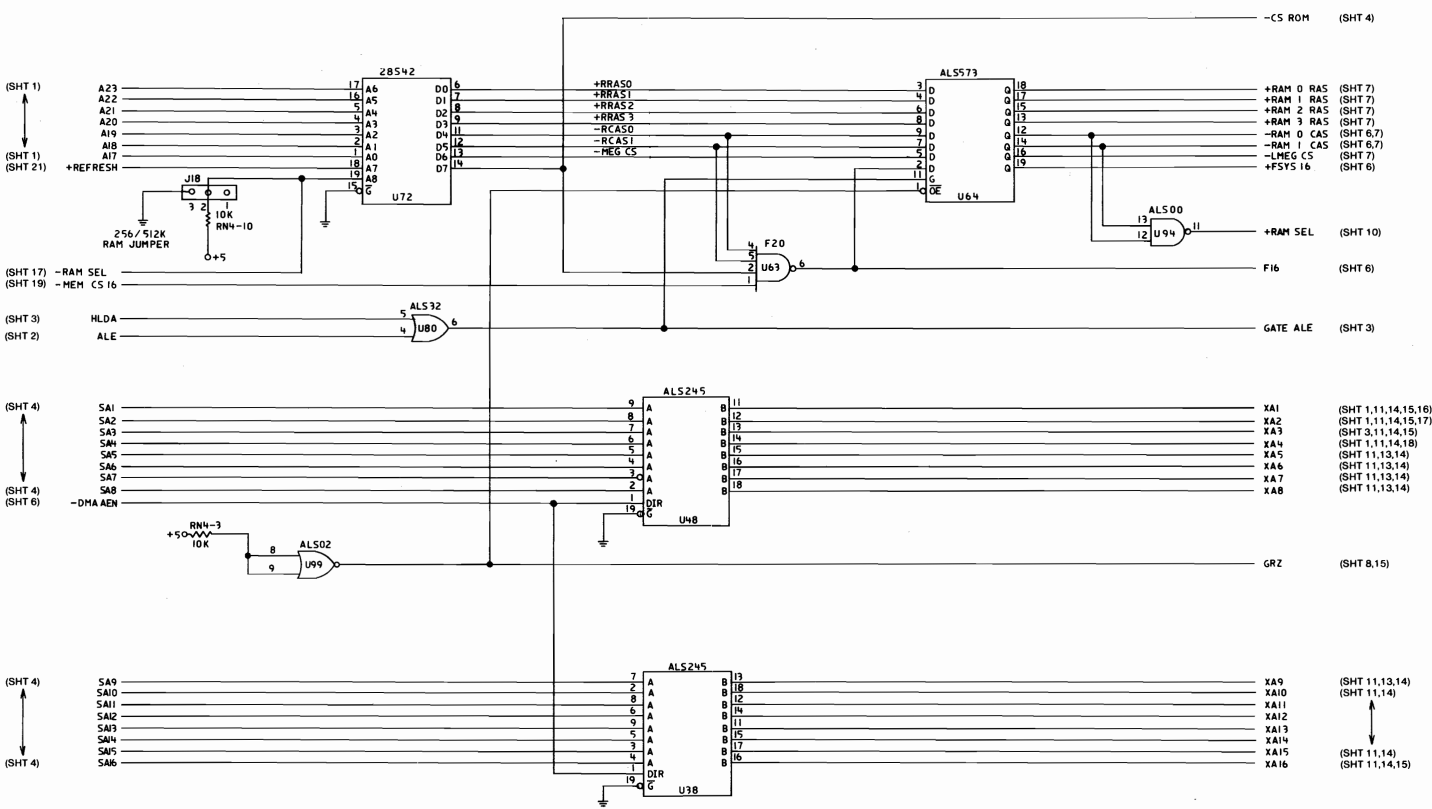
System Board (Sheet 2 of 22)



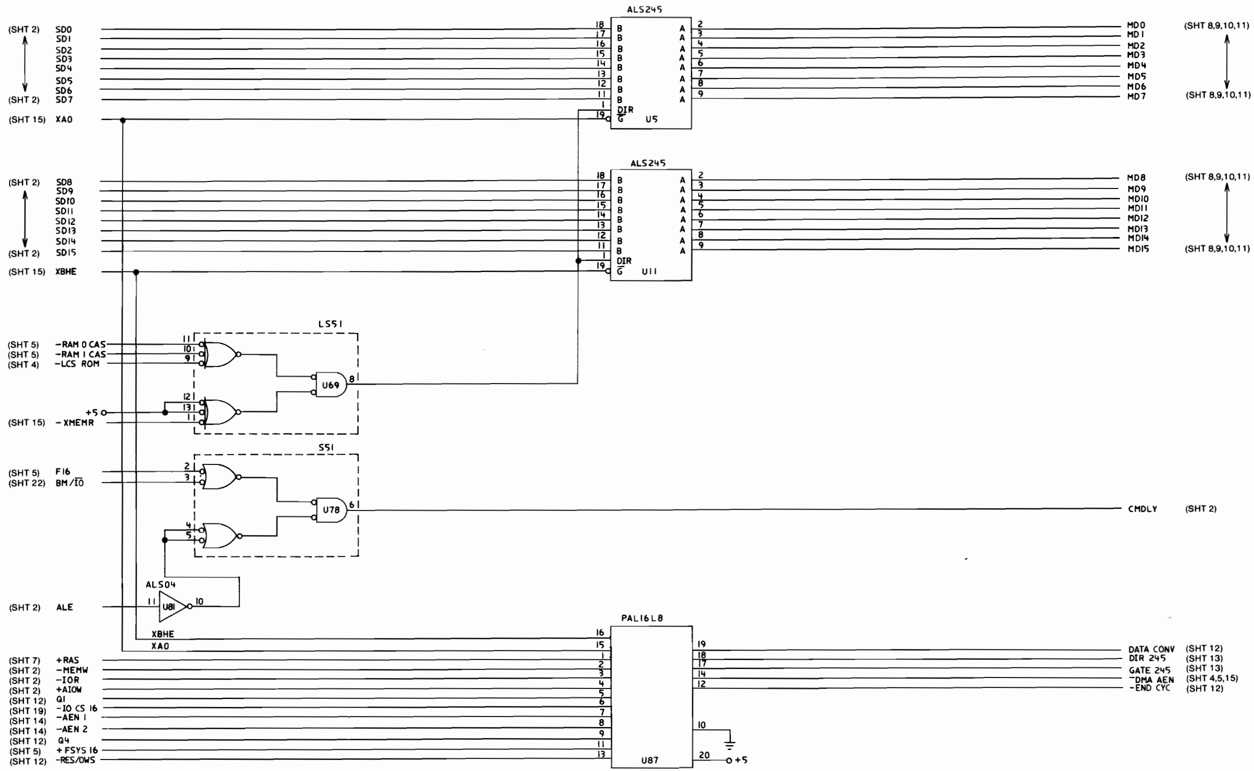
System Board (Sheet 3 of 22)



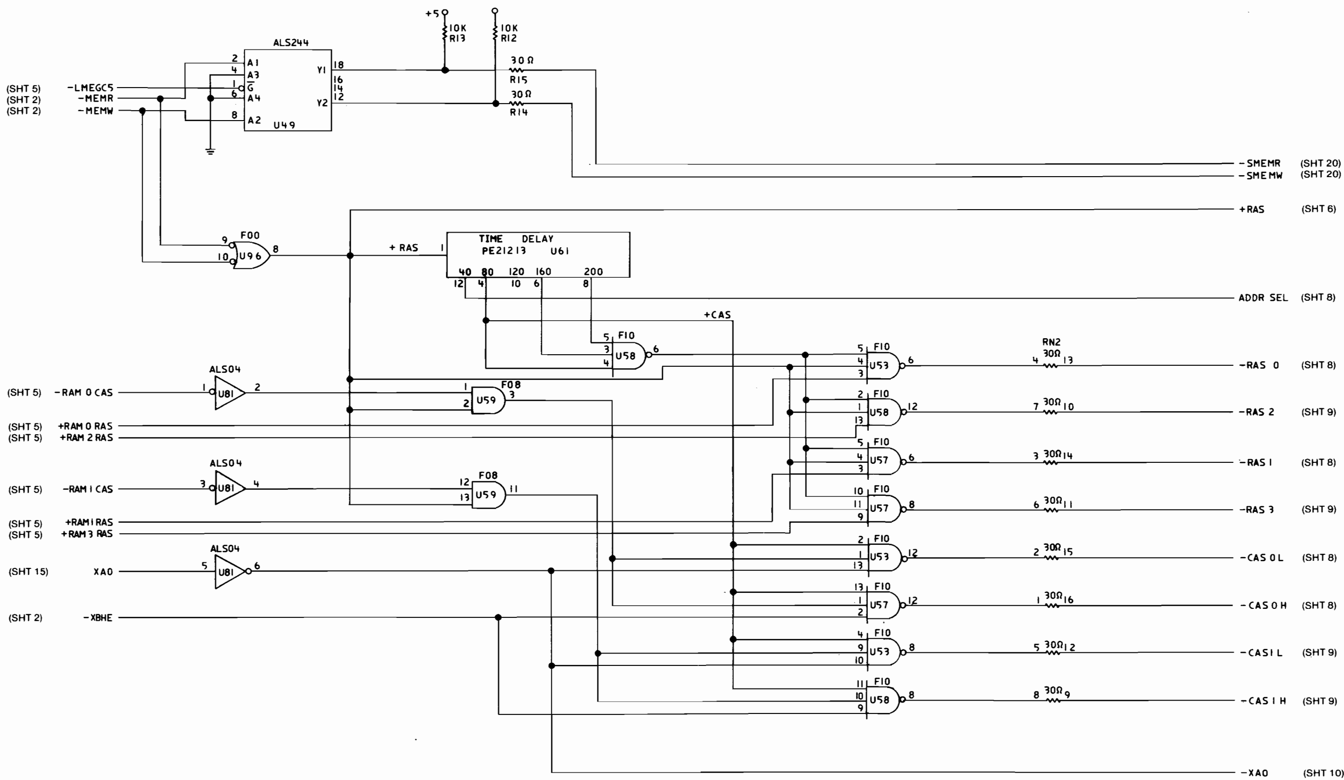
System Board (Sheet 4 of 22)



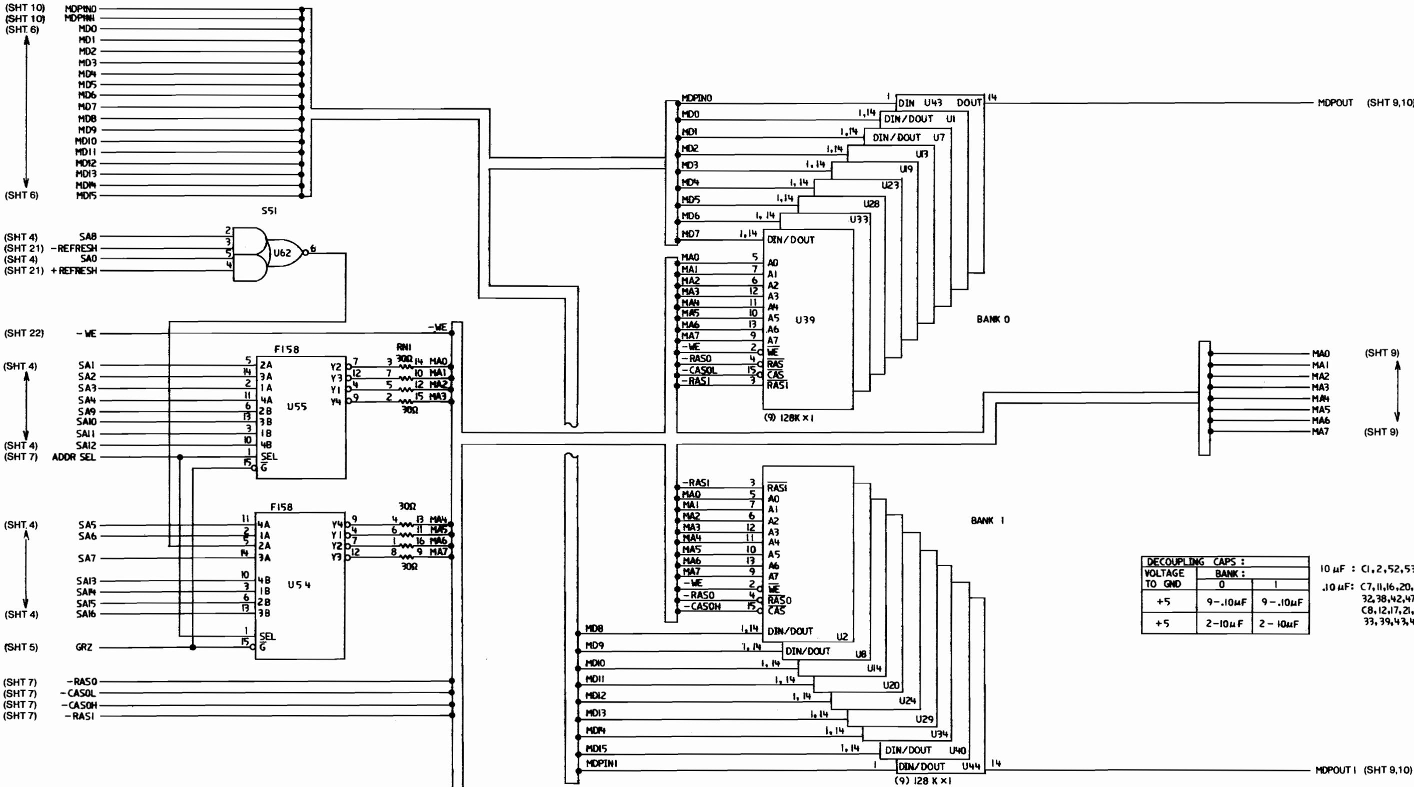
System Board (Sheet 5 of 22)



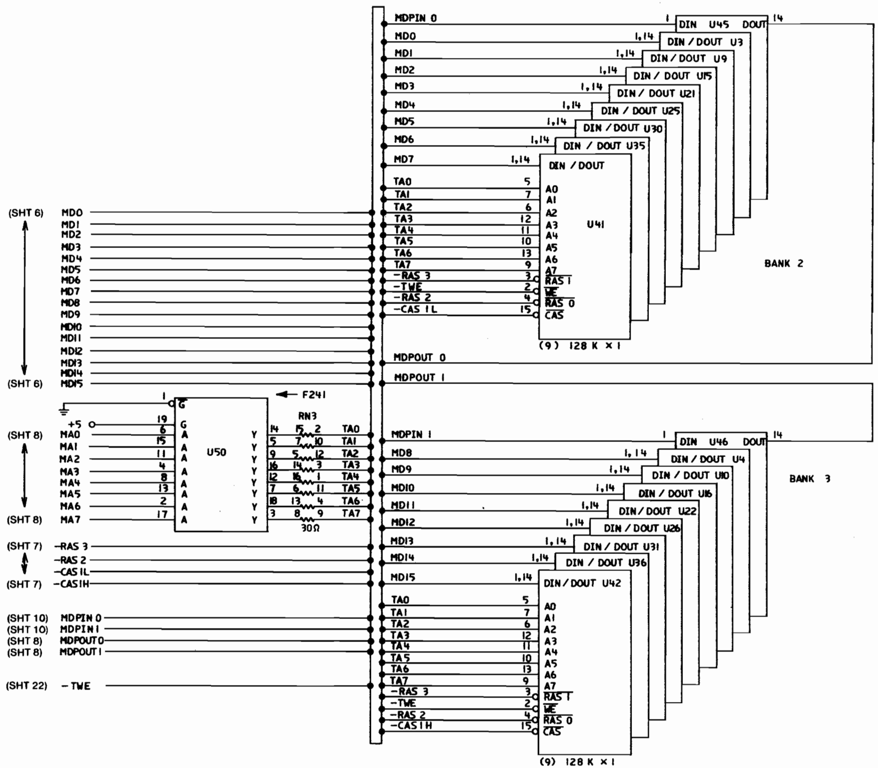
System Board (Sheet 6 of 22)



System Board (Sheet 7 of 22)



System Board (Sheet 8 of 22)

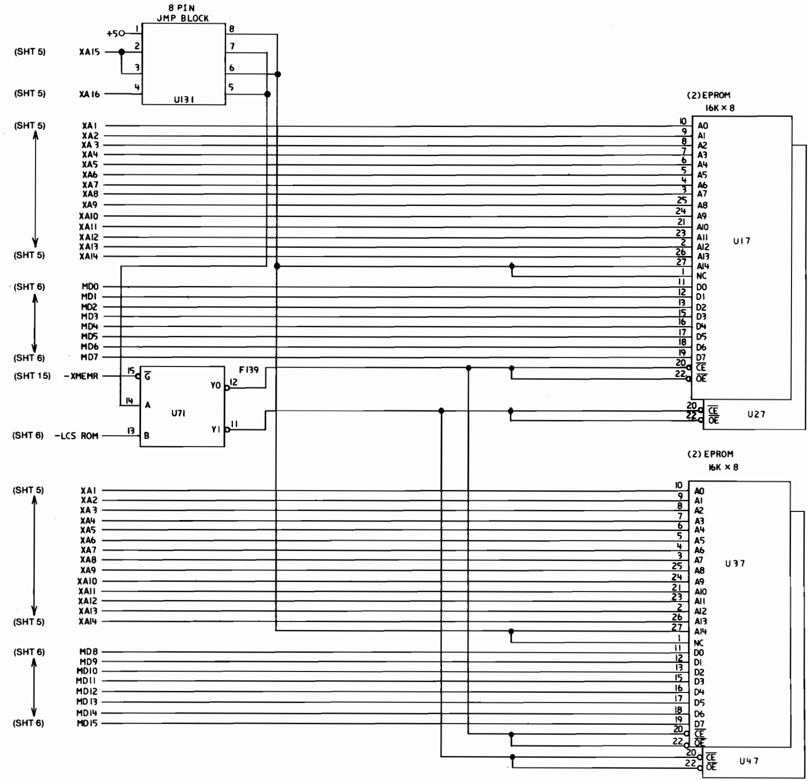


DECOUPLING CAP:		
VOLTAGE TO GND	BANK 2	BANK 3
+ 5	9 − .10µF	9 − .10µF
+ 5	2 − 10µF	2 − 10µF

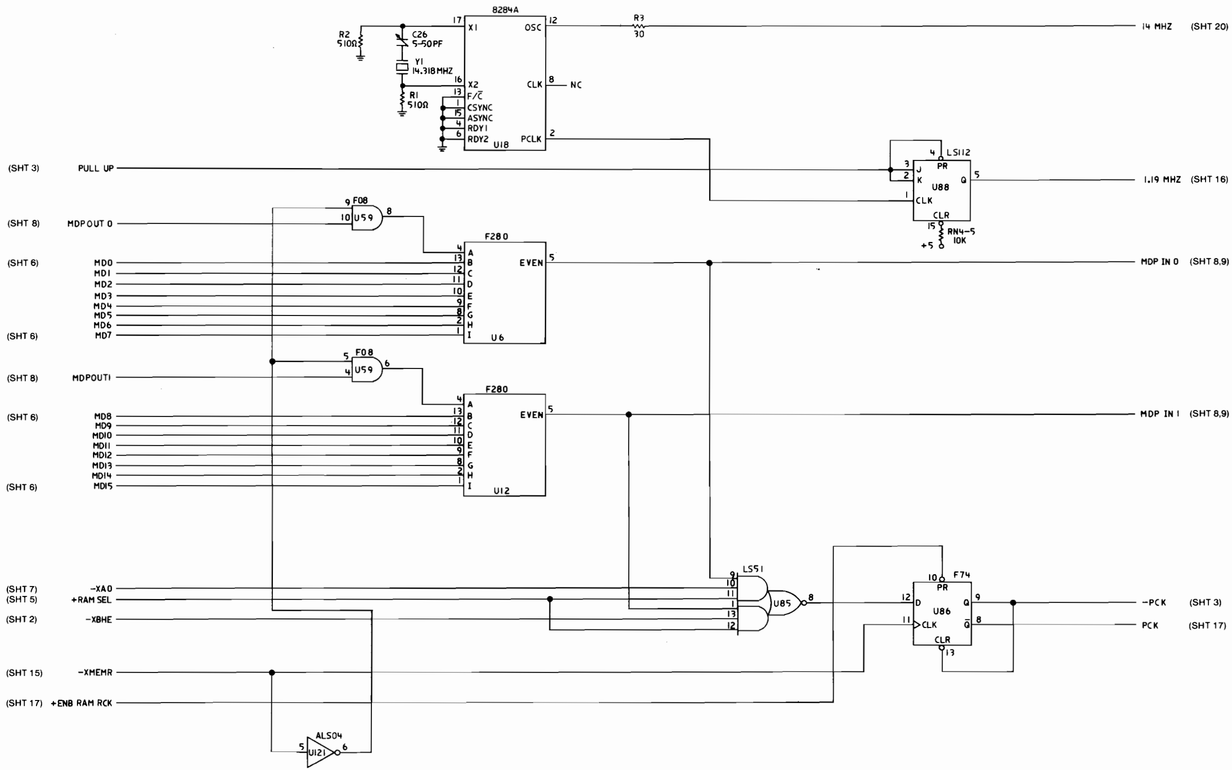
10µF : C3,4,54,55

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

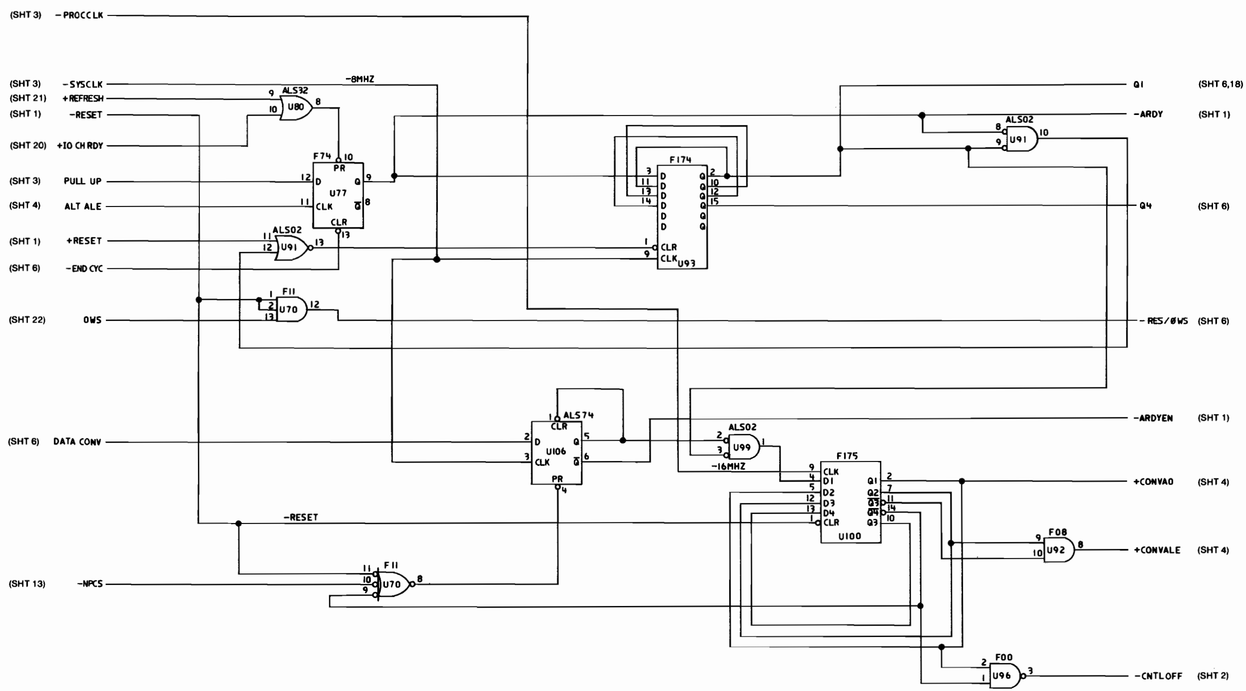
System Board (Sheet 9 of 22)



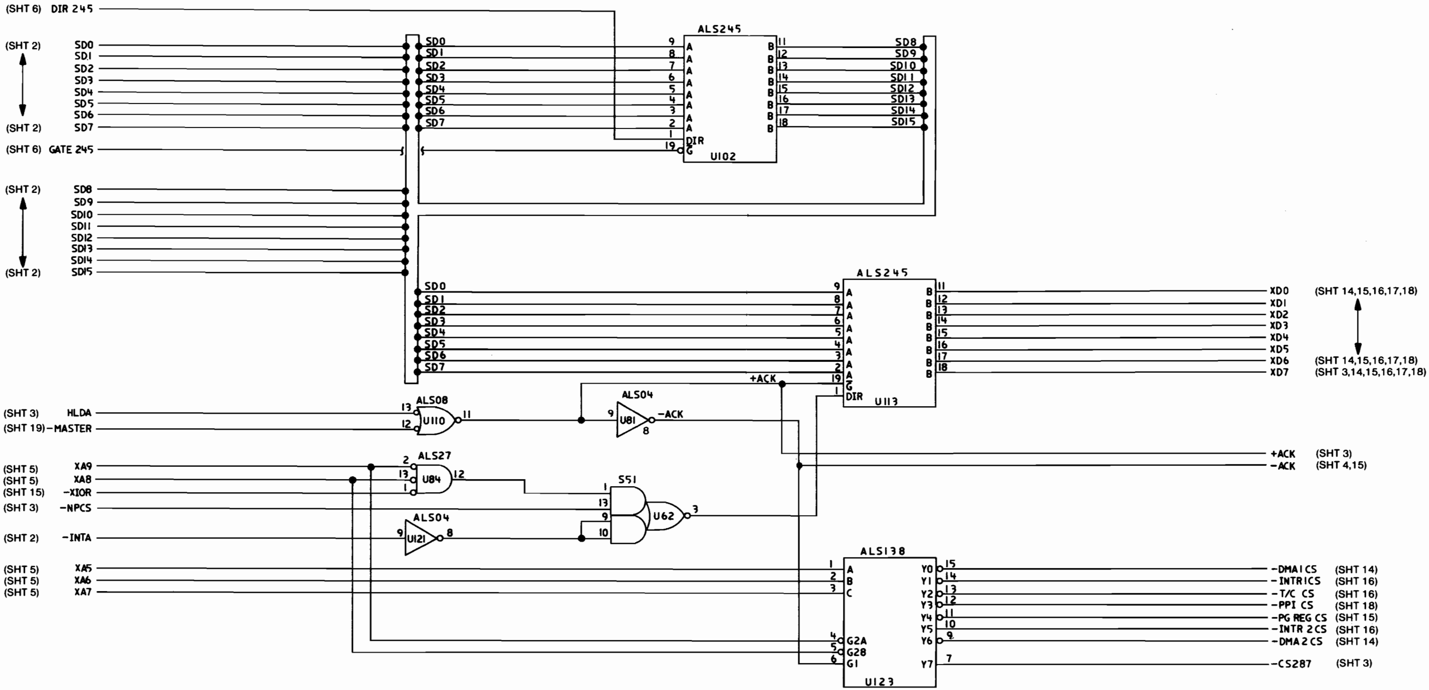
System Board (Sheet 10 of 22)



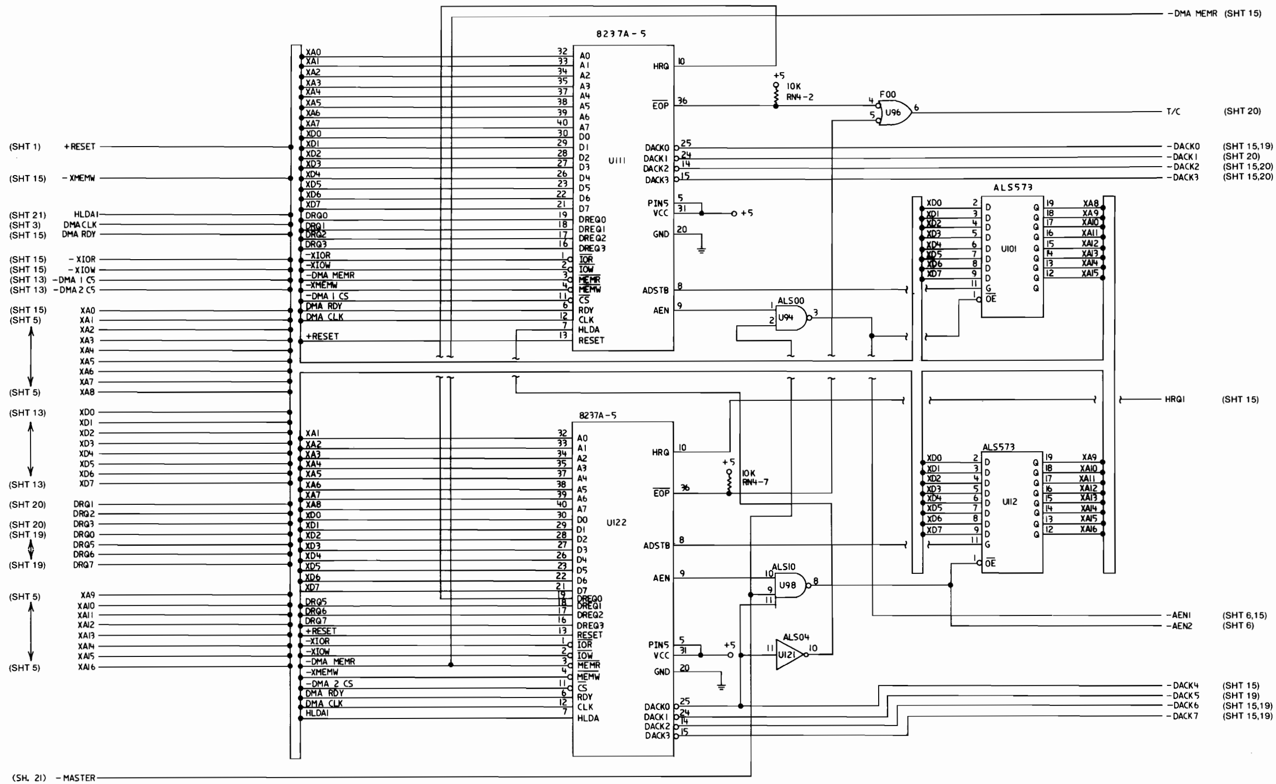
System Board (Sheet 11 of 22)



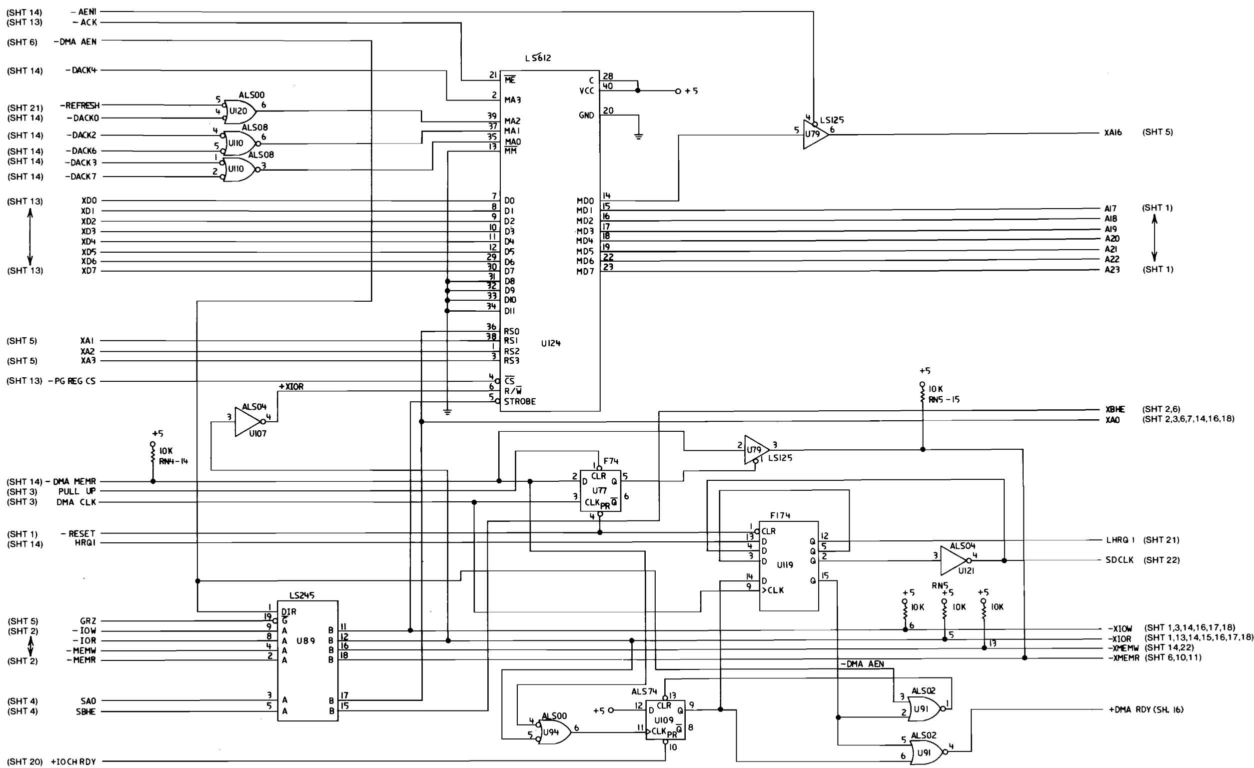
System Board (Sheet 12 of 22)



System Board (Sheet 13 of 22)

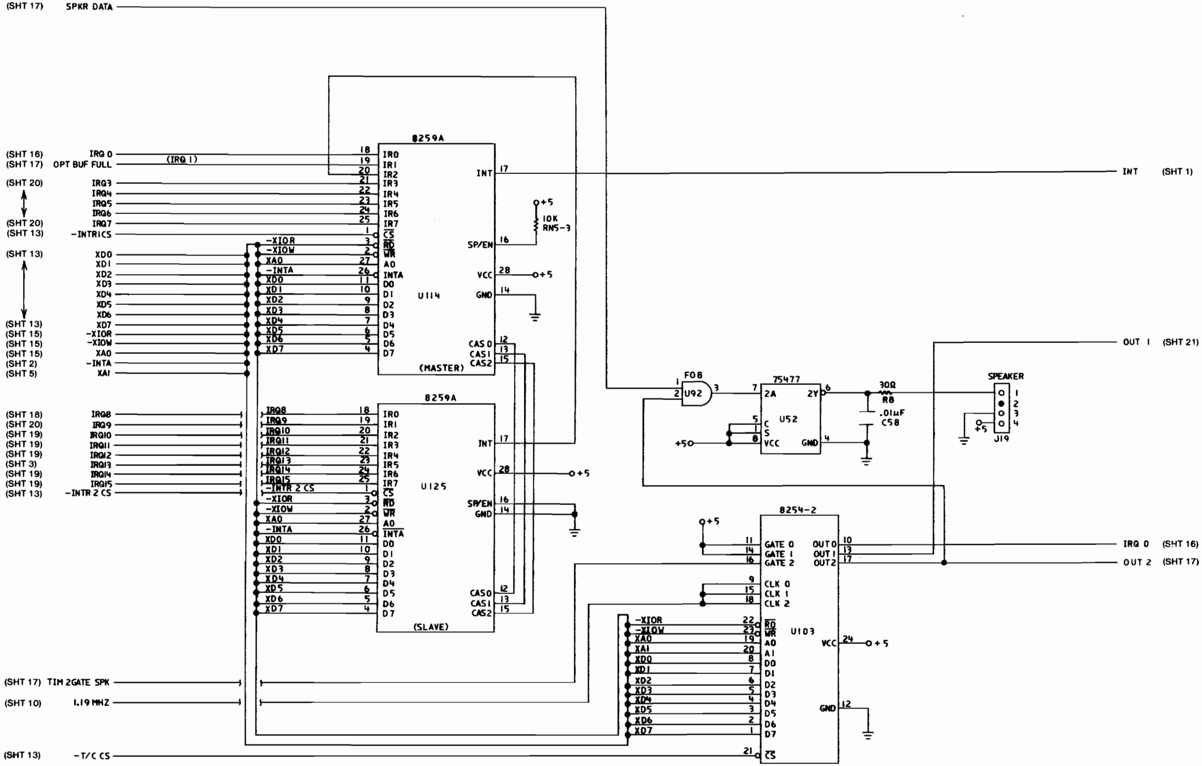


System Board (Sheet 14 of 22)

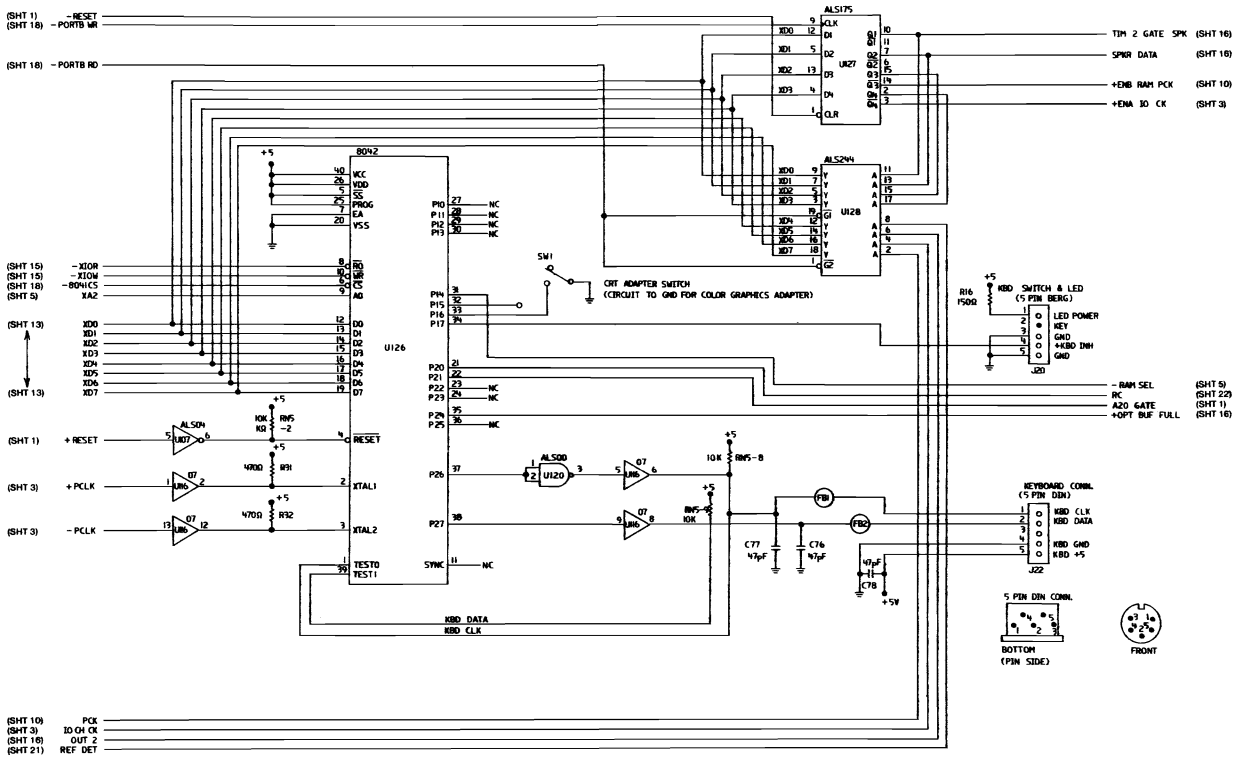


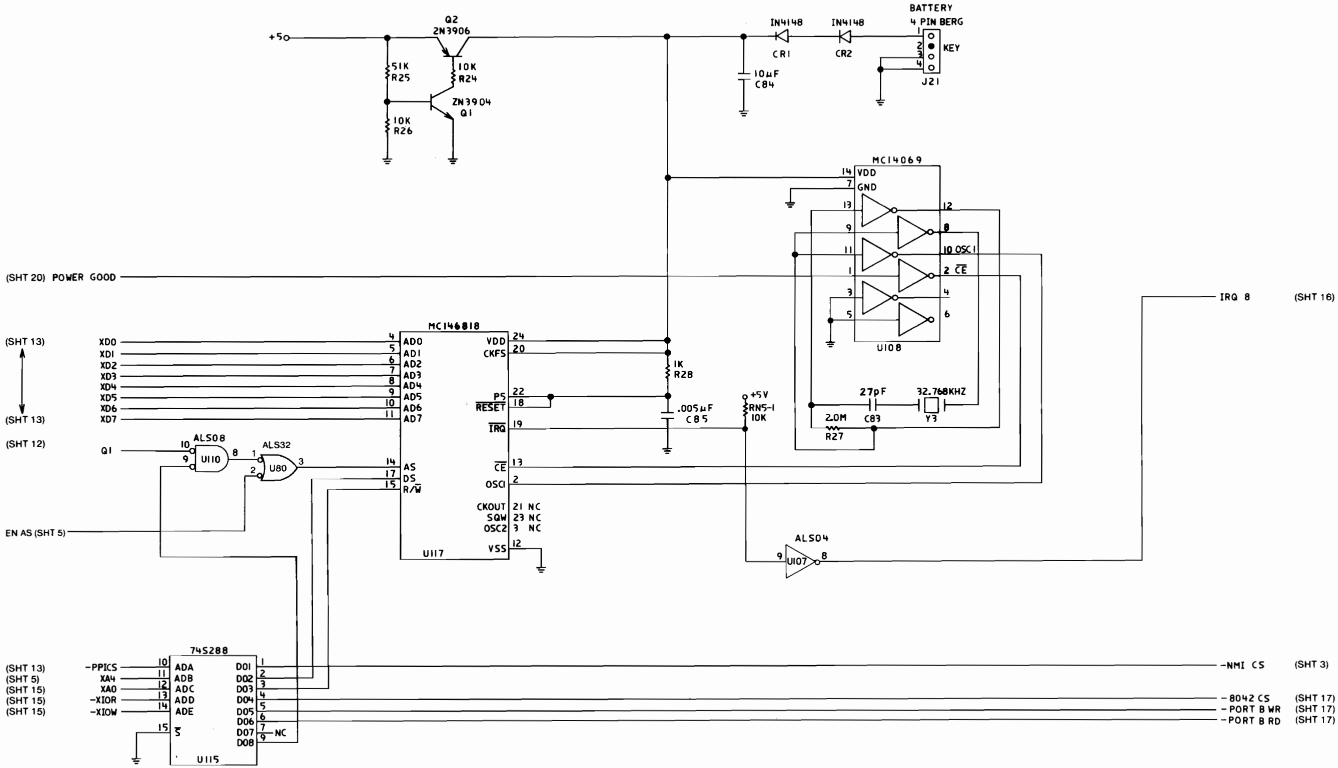
System Board (Sheet 15 of 22)

System Board (Sheet 16 of 22)

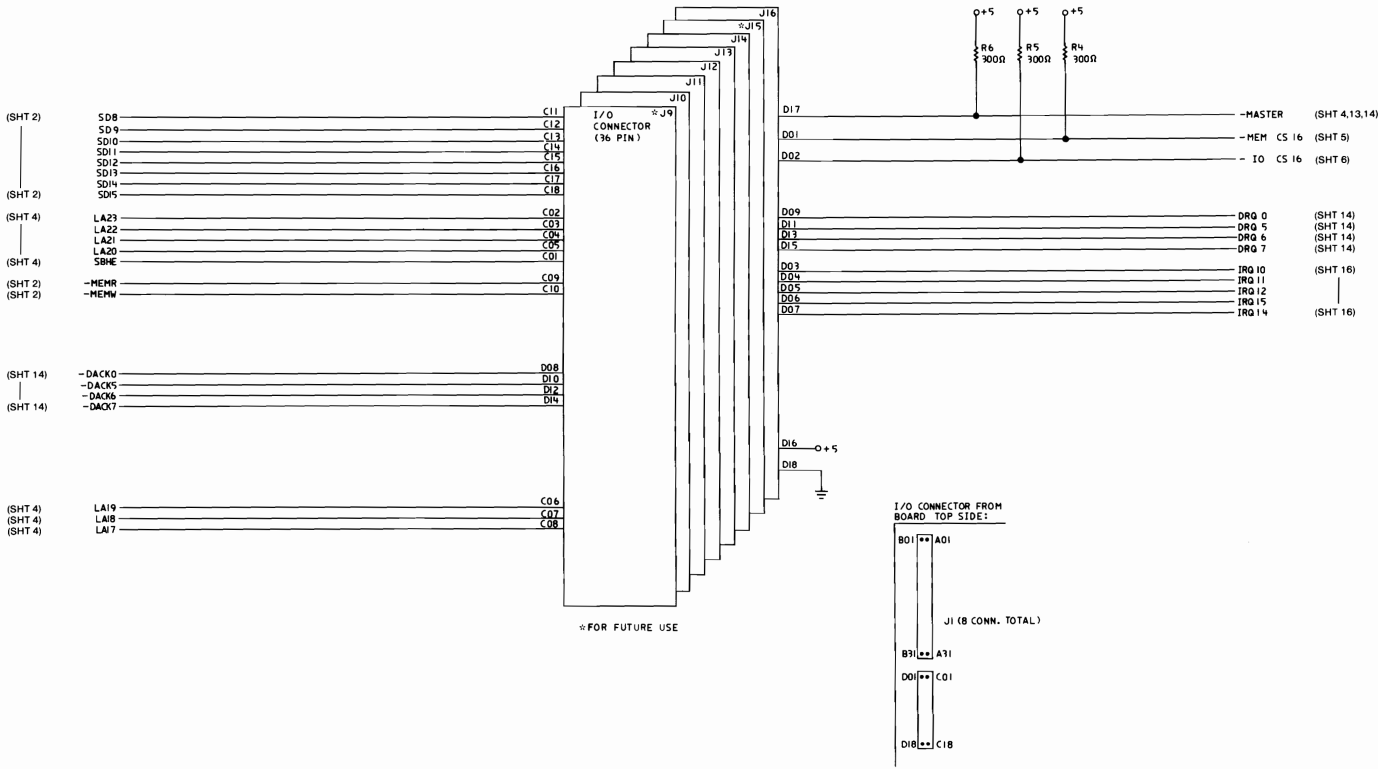


System Board (Sheet 17 of 22)

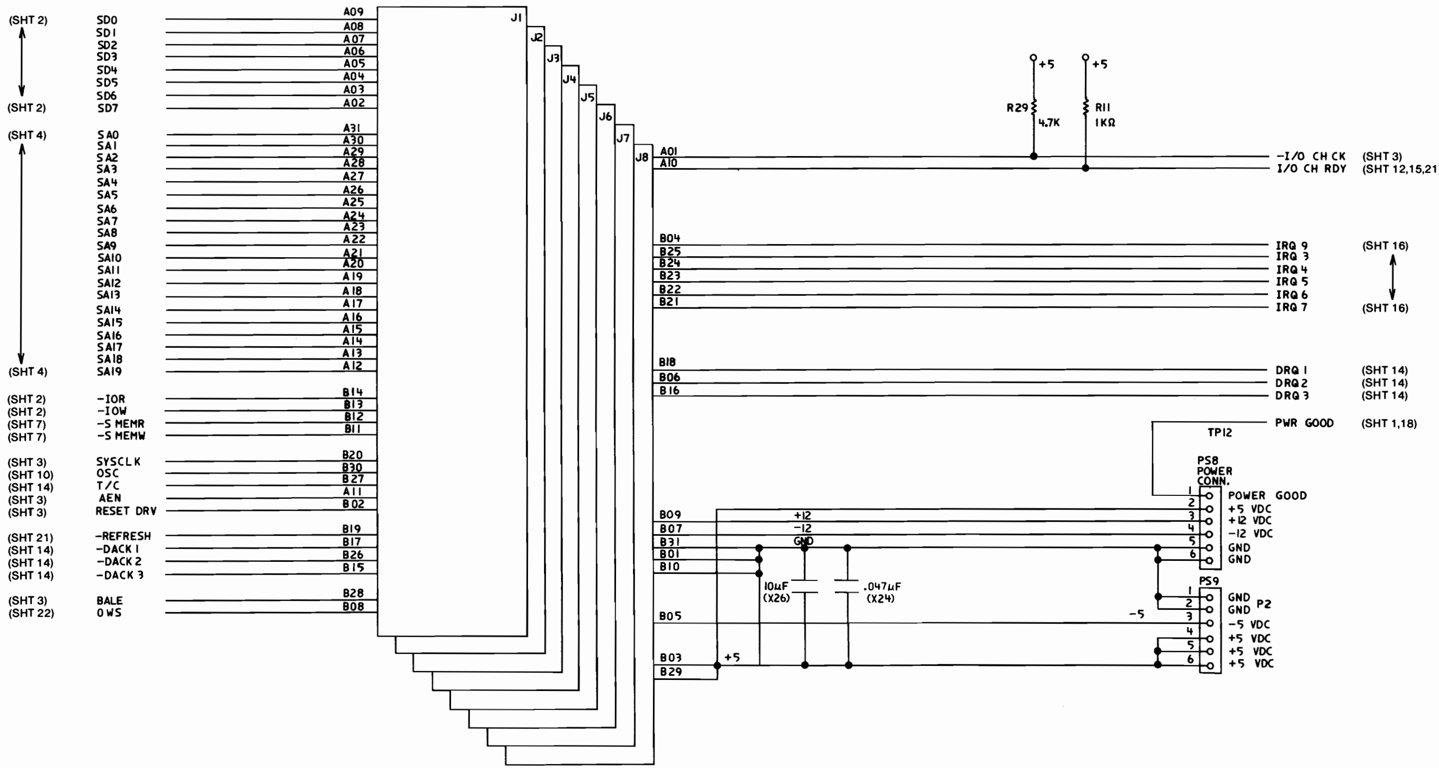




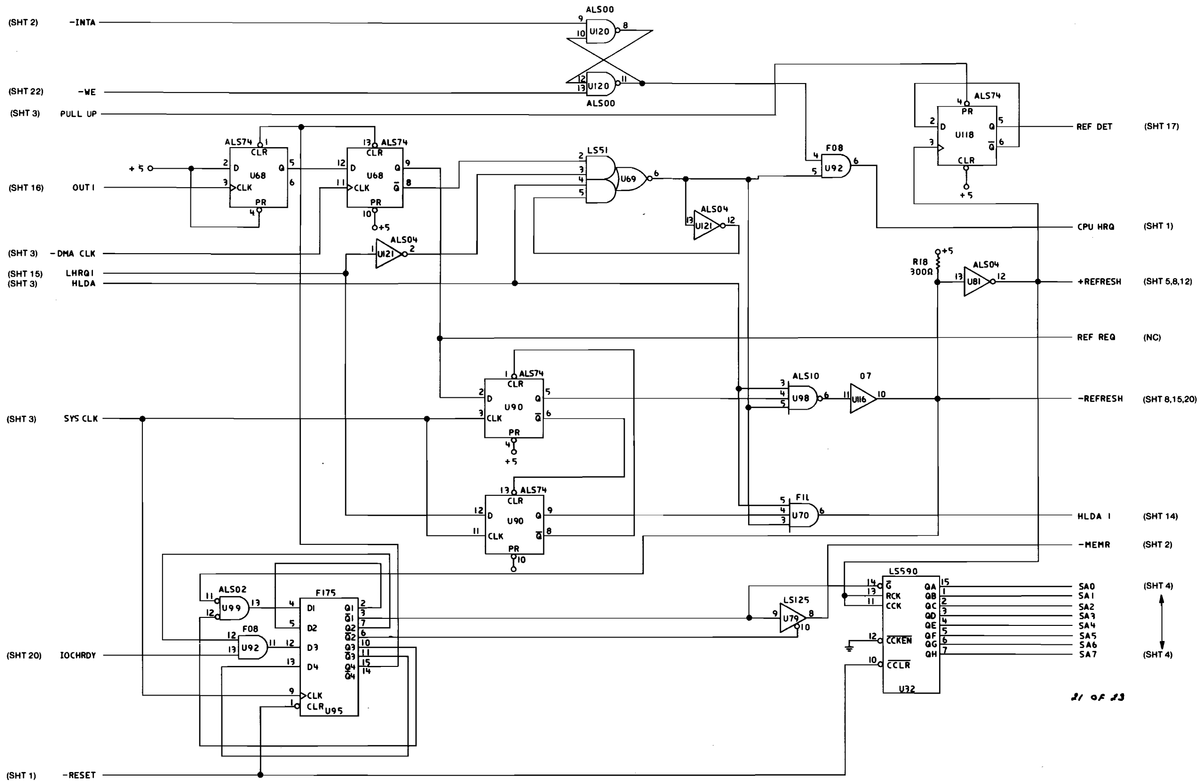
System Board (Sheet 18 of 22)



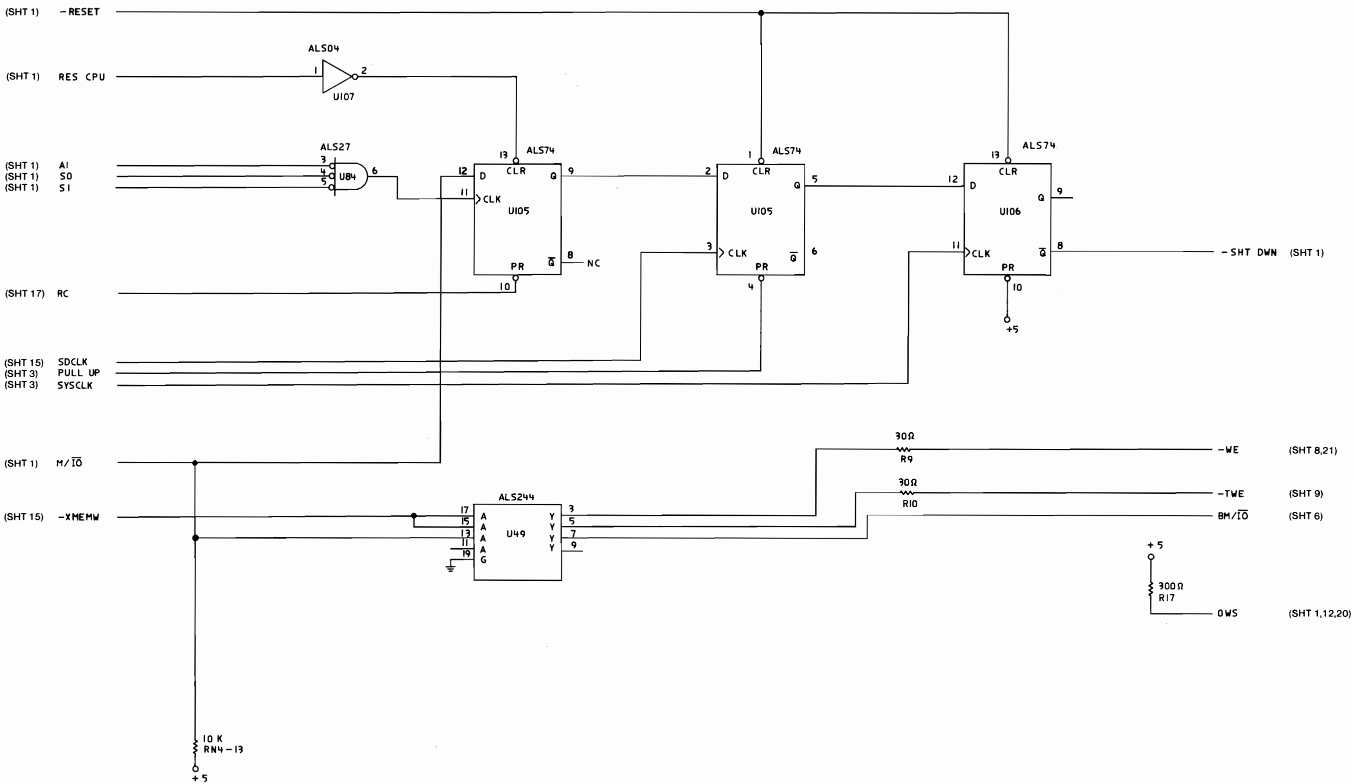
System Board (Sheet 19 of 22)



System Board (Sheet 20 of 22)



System Board (Sheet 21 of 22)



System Board (Sheet 22 of 22)

SECTION 2. COPROCESSOR

Contents

SECTION 2

Notes:

Description

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

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 the 40 16–bit registers in the microprocessor. This register space allows constants and temporary results to be held in registers during calculations, thus reducing memory access and improving speed as well as bus availability. The register space can be used as a stack or as a fixed register set. When used as a stack, only the top two stack elements are operated on. The following figure shows representations of large and small numbers in each data type.

Data Type	Bits	Significant Digits (Decimal)	Approximate Range (Decimal)
Word Integer	16	4	$-32,768 \leq x \leq +32,767$
Short Integer	32	9	$-2 \times 10^9 \leq x \leq +2 \times 10^9$
Long Integer	64	19	$-9 \times 10^{18} \leq x \leq +9 \times 10^{18}$
Packed Decimal	80	18	$-99...99 \leq x \leq +99...99$ (18 digits)
Short Real *	32	6-7	$8.43 \times 10^{-37} \leq x \leq 3.37 \times 10^{38}$
Long Real *	64	15-16	$4.19 \times 10^{-307} \leq x \leq 1.67 \times 10^{308}$
Temporary Real	80	19	$3.4 \times 10^{-4932} \leq x \leq 1.2 \times 10^{4932}$

Data Types

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

Hardware Interface

The math coprocessor uses the same clock generator as the microprocessor. It works at one-third the frequency of the system microprocessor clock. 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 hardware interrupt 13 and sets up its vector to point to a routine in ROM. The ROM routine clears the 'BUSY' signal's latch and then transfers control to the address pointed to by the NMI interrupt vector. This allows code written for any IBM Personal Computer to work on an IBM Personal Computer AT. The NMI interrupt handler should read the coprocessor's status to determine if the NMI was caused by the coprocessor. If the interrupt was not generated by the coprocessor, control should be passed to the original NMI interrupt handler.

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

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

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

Notes:

SECTION 3. POWER SUPPLY

Contents

SECTION 3

Notes:

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

Inputs

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

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

Input Requirements

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

Outputs

The power supply provides +5, -5, +12, and -12 Vdc. The following figure shows the load current and regulation tolerance for the voltages.

Note: The power supply also supplies either 115 Vac or 230 Vac for the IBM Monochrome Display.

SECTION 3

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

DC Load Requirements

Output Protection

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

Dummy Load

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

Output Voltage Sequencing

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

No-Load Operation

No damage or hazardous conditions occur when primary power is applied with no load on any output level. In such cases, the power supply may switch off, and a power/on cycle 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 1 second and then switched on, the 'power-good' signal is generated, assuming there are no problems. This signal is a logical AND of the dc output-voltage sense signal and the ac input-voltage sense signal. The power-good signal is also a TTL-compatible high level for normal operation, or a low level for fault conditions. The ac fail signal causes power-good to go to a low level at least 1 millisecond before any output voltage falls below the regulation limits. The operating point used as a reference for measuring the 1 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. The following figure shows the minimum sense levels for the output voltages.

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

Sense Levels

Fan-Out

Fan-out is the number of inputs that one output can drive. The 'power-good' signal can drive six standard TTL loads.

Connectors

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

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

DC Load Distribution

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

Notes:

SECTION 4. KEYBOARD

Contents

SECTION 4

Description

The keyboard is a low-profile, 84 key, detachable unit.

Interface

The keyboard uses a bidirectional serial interface to carry signals between the keyboard and system unit.

Sequencing Key Code Scanning

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

Keyboard Buffer

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

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

Keys

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

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

Functions Performed at Power-On Time

Power-On Reset

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

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

Basic Assurance Test

Immediately following the POR, the keyboard executes a basic assurance test (BAT). This test consists of a checksum of all read-only memory (ROM), and a stuck-bit and addressing test of all random-access memory (RAM) in the keyboard's microprocessor. The mode indicators—three light emitting diodes

(LEDs) on the upper right-hand corner of the keyboard—are turned on then off, and must be observed to ensure they are operational.

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

The BAT can also be started by a Reset command.

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

Commands from the System

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

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

SECTION 4

Reset (Hex FF)

The system issues a Reset command to start a program reset and a keyboard internal self-test. The keyboard acknowledges the command with an 'acknowledge' signal (ACK) and ensures the system accepts the 'ACK' before executing the command. The system signals acceptance of the 'ACK' by raising the clock and data for a minimum of 500 microseconds. The keyboard is disabled from the time it receives the Reset command until the 'ACK' is accepted or until another command overrides the previous one. Following acceptance of the 'ACK', the keyboard begins the reset operation, which is similar to a power-on reset. The keyboard clears the output buffer and sets up default values for typematic and delay rates.

Resend (Hex FE)

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

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

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

Set Default (Hex F6)

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

Default Disable (Hex F5)

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

Enable (Hex F4)

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

Set Typematic Rate/Delay (Hex F3)

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

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

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

Bit	Rate	Bit	Rate
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	8.6	11110	2.1
01111	8.0	11111	2.0

Typematic Rate

The keyboard responds to the Set Typematic Rate Delay command with an 'ACK', stops scanning, and waits for the rate parameter. The keyboard responds to the rate parameter with another 'ACK', sets the rate and delay, and continues scanning (if the keyboard was previously enabled). If a command is received instead of the rate parameter, the set-typematic-rate

SECTION 4

function ends with no change to the existing rate, and the new command is processed. However, the keyboard will not resume scanning unless instructed to do so by an Enable command.

The default rate for the system keyboard is as follows:

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

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

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

Echo (Hex EE)

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

Set/Reset Mode Indicators (Hex ED)

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

It is up to the using system to remember the previous state of an indicator. This is in case its setting does not change when a command sequence is issued to change the state of another indicator.

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

The command has the following format:

Set/Reset Command

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

11101101 – hex ED

The second byte is an option byte. It has a list of the indicators to be acted upon. The format of the option byte is as follows:

Bit 7 Reserved

Bit 6 Reserved

Bit 5 Reserved

Bit 4 Reserved

Bit 3 Reserved

Bit 2 Caps Lock indicator

Bit 1 Numeric Lock indicator

Bit 0 Scroll Lock indicator

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

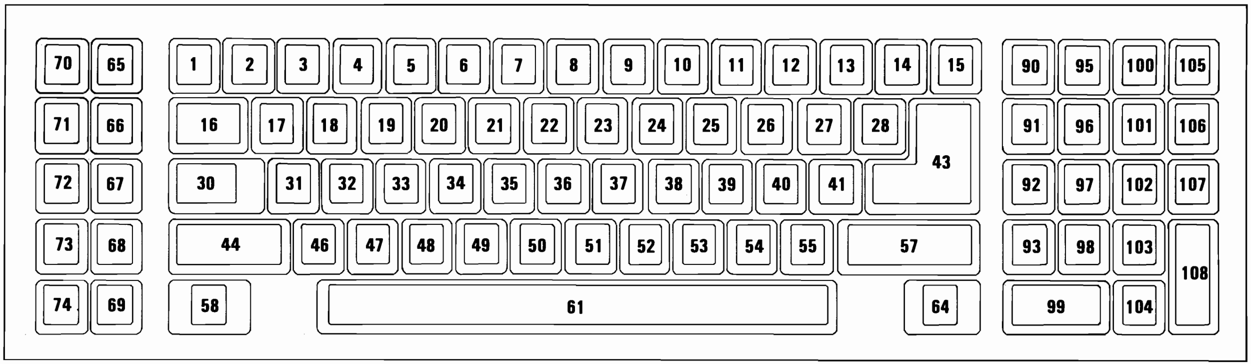
The keyboard will respond to the Set/Reset Mode Indicators command with an 'ACK', discontinue scanning, and wait for the option byte. The keyboard will respond to the option byte with an Ack, set the indicators, and continue scanning if the keyboard was previously enabled. If another command is received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

Keyboard Outputs

Key Scan Codes

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

The typematic scan code for a key is the same as the key's make code. The following figure is a keyboard layout.


70 65 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 90 95 100 105
71 66 16 17 18 19 20 21 22 23 24 25 26 27 28 43 91 96 101 106
72 67 30 31 32 33 34 35 36 37 38 39 40 41 92 97 102 107
73 68 44 46 47 48 49 50 51 52 53 54 55 57 93 98 103 108
74 69 58 61 64 99 104

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

Key Positions and Their Make Codes				
1--DE	18--1D	36--33	55--4A	90--76
2--16	19--24	37--3B	56--51	91--6C
3--1E	20--2D	38--42	57--59	92--6B
4--26	21--2C	39--4B	58--11	93--69
5--25	22--35	40--4C	60--19	94--77
6--2E	23--3C	41--52	61--29	96--75
7--36	24--43	43--5A	64--58	97--73
8--3D	25--44	44--12	65--D6	98--72
9--3E	26--4D	46--1A	66--DC	99--70
10--46	27--54	47--22	67--0B	100--7E
11--45	28--5B	48--21	68--0A	101--7D
12--4E	30--14	49--2A	69--09	102--74
13--55	31--1C	50--32	70--05	103-7A
14--5D	32--1B	51--31	71--04	104--71
15--66	33--23	52--3A	72--D3	105--84
16--0D	34--2B	53--41	73--83	106--7C
17--15	35--34	54--49	74--01	107--7B

Make Scan Codes

Command Codes to the System

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

Resend (Hex FE)

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

ACK (Hex FA)

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

Overrun (Hex 00)

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

Diagnostic Failure (Hex FD)

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

Break Code Prefix (Hex F0)

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

BAT Completion Code (Hex AA)

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

ECHO Response (Hex EE)

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

Clock and Data Signals

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

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

Bit	Function
1st bit	0 start bit
2nd bit	Data bit 0 (least-significant)
3rd bit	Data bit 1
4th bit	Data bit 2
5th bit	Data bit 3
6th bit	Data bit 4
7th bit	Data bit 5
8th bit	Data bit 6
9th bit	Data bit 7 (most-significant)
10th bit	Parity bit (odd parity)
11th bit	Stop bit

Transmission Data Stream

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

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

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

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

Keyboard Data Output

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

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

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

Keyboard Data Input

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

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

SECTION 4

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

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

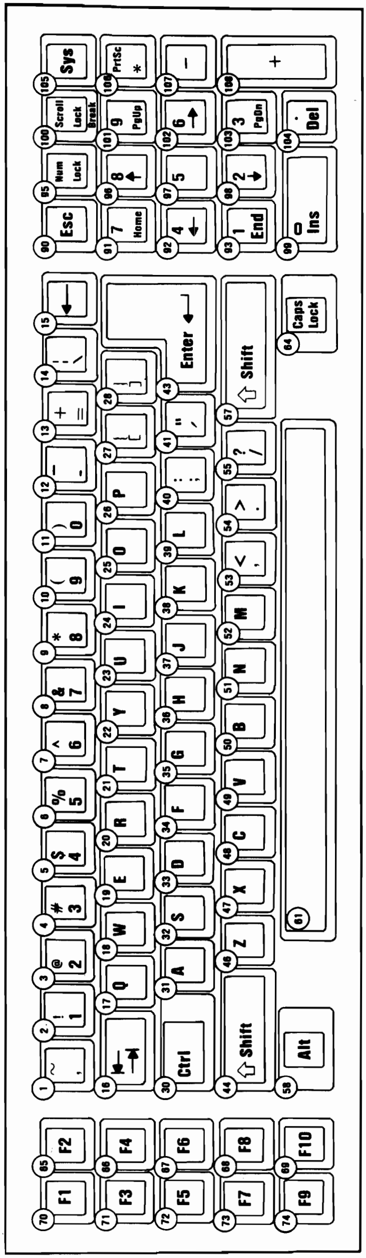
Keyboard Layout

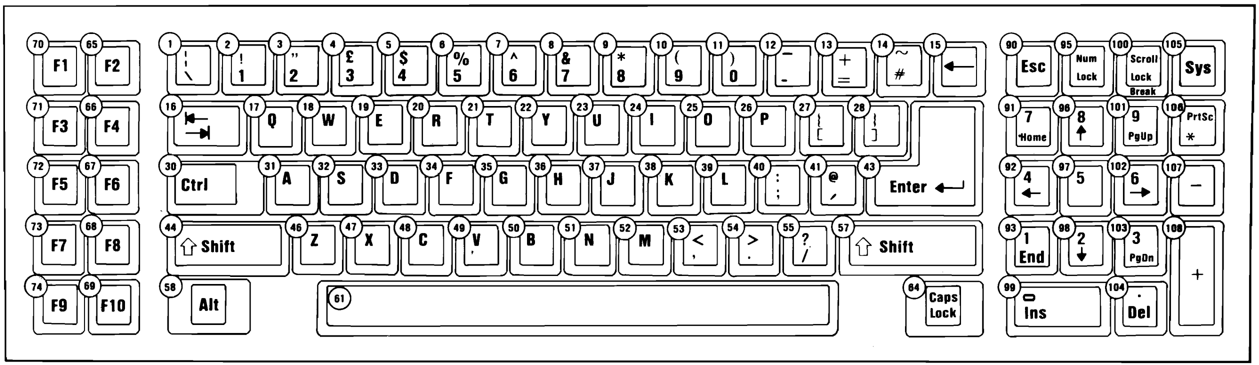
The IBM Personal Computer AT Keyboard is available in six different layouts:

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

The following pages show all six possible keyboard layouts.

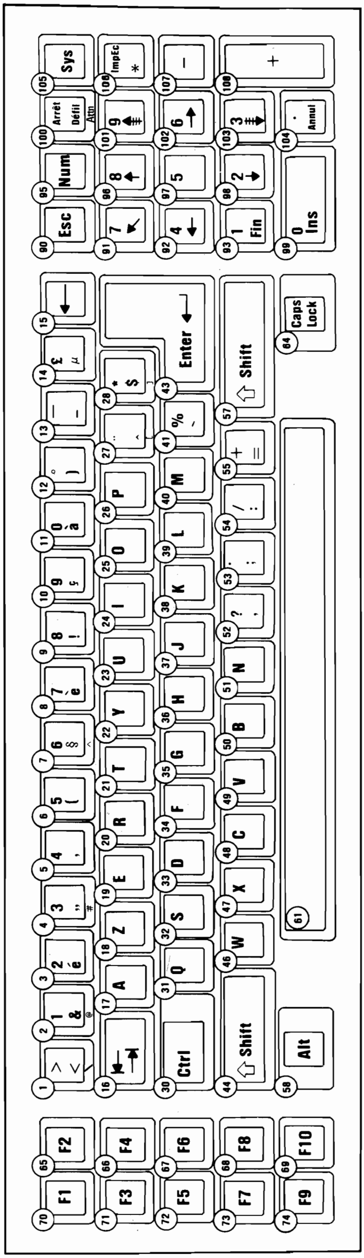
U.S. English Keyboard



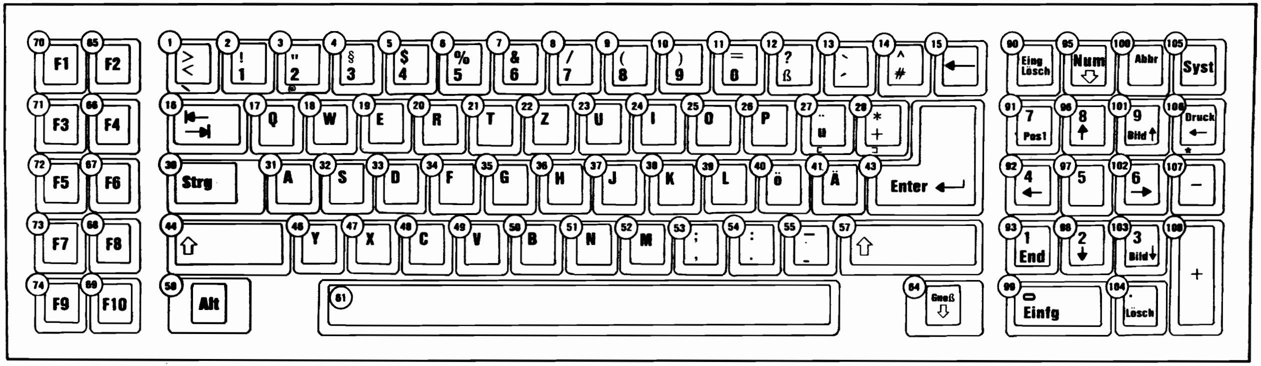


F1 F2 F3 F4 F5 F6 F7 F8 F9 F10
` \ ! 1 " 2 £ 3 $ 4 % 5 ^ 6 & 7 * 8 (9) 0 _ - + = ~ #
Q W E R T Y U I O P { [}]
Ctrl A S D F G H J K L : ; @ ' Enter
Shift Z X C V B N M < , > . ? / Shift
Alt Caps Lock
Esc Num Lock Scroll Lock Break Sys
7 Home 8 9 PgUp PrtSc *
4 5 6 −
1 End 2 3 PgDn +
Ins . Del

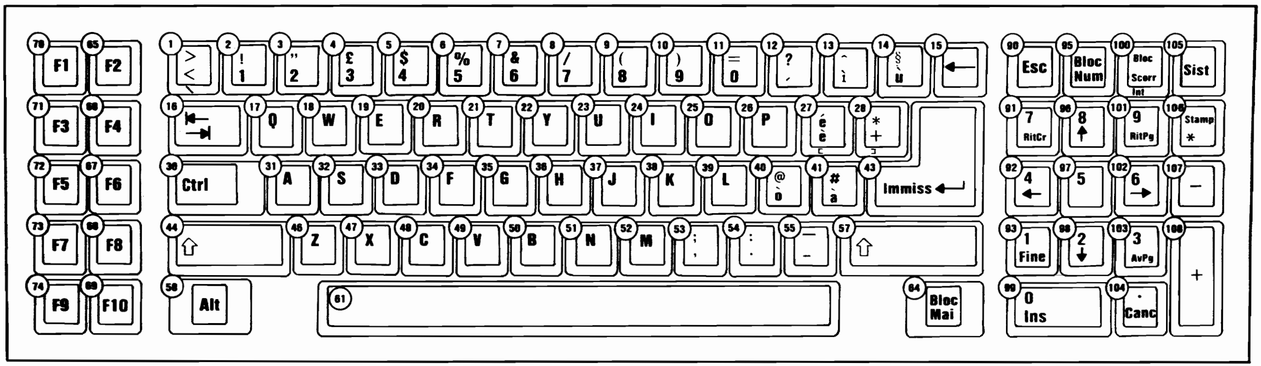
French Keyboard

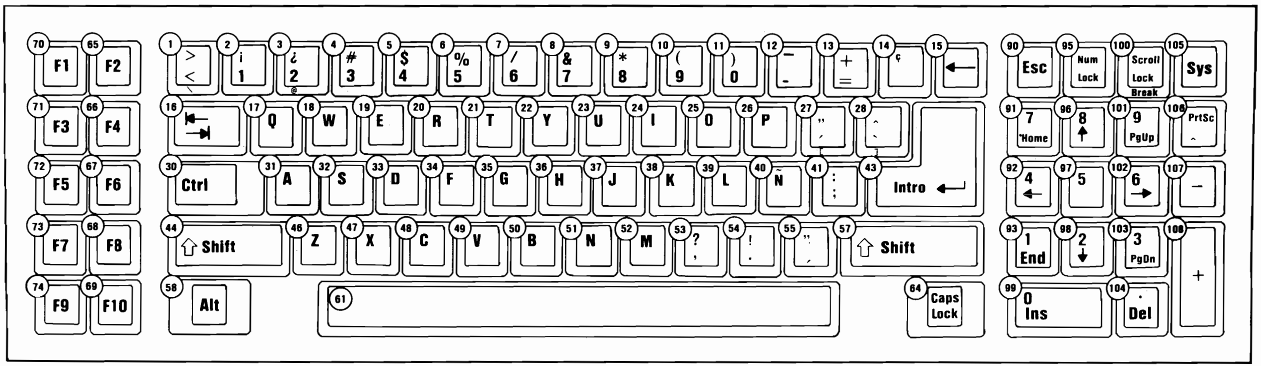


SECTION 4


70 F1
65 F2
1 > <
2 ! 1
3 " 2 @
4 § 3
5 $ 4
6 % 5
7 & 6
8 / 7
9 (8
10) 9
11 = 0
12 ? ß
13 ` ´
14 ^ #
15
90 Eing Lösch
95 Num
100 Abbr
105 Syst
71 F3
66 F4
16
17 Q
18 W
19 E
20 R
21 T
22 Z
23 U
24 I
25 O
26 P
27 ü
28 * +
91 7 Pos1
96 8
101 9 Bild
106 Druck
72 F5
67 F6
30 Strg
31 A
32 S
33 D
34 F
35 G
36 H
37 J
38 K
39 L
40 ö
41 Ä
43 Enter
92 4
97 5
102 6
107 −
73 F7
68 F8
44
46 Y
47 X
48 C
49 V
50 B
51 N
52 M
53 ; ,
54 : .
55 _ -
57
93 1 End
98 2
103 3 Bild
108 +
74 F9
69 F10
58 Alt
61
64 Groß
99 Einfg
104 Lösch

Italian Keyboard




F1
F2
F3
F4
F5
F6
F7
F8
F9
F10
Ctrl
Shift
Alt
Intro
Shift
Caps Lock
Esc
Num Lock
Scroll Lock Break
Sys
7 Home
8
9 PgUp
PrtSc
4
5
6
1 End
2
3 PgDn
0 Ins
. Del
+
–
Q
W
E
R
T
Y
U
I
O
P
A
S
D
F
G
H
J
K
L
Ñ
Z
X
C
V
B
N
M

Specifications

Size

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

Weight

- 2.8 kilograms (6.2 pounds)

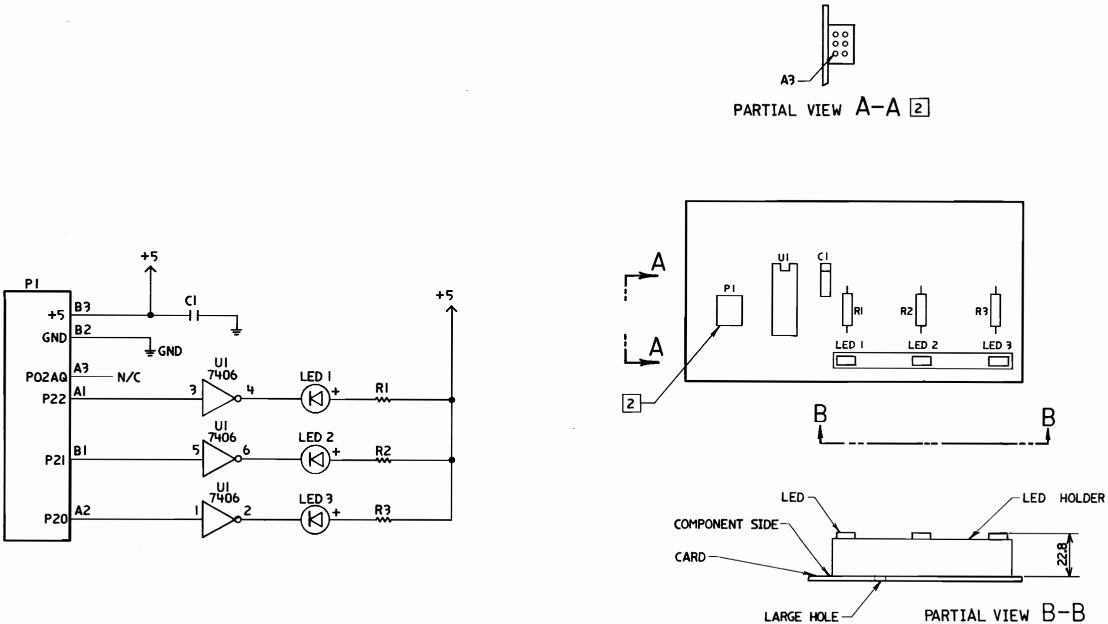
Keyboard Connector

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

Connector Pin	Signal Name
1	Clock
2	Data
3	Spare
4	Ground
5	+5 Vdc

Keyboard Connector

SECTION 4



NOTES:

1 SOLDERING: THIS MANUFACTURING PROCESS MAY INCLUDE HAZARDOUS OPERATIONS AND REQUIRE SPECIAL HEALTH AND/OR SAFETY PRECAUTIONS.

2 POLARIZE P1 AT POSITION A3

Enhancement Logic Card Assembly

SECTION 5. SYSTEM BIOS

Contents

SECTION 5

Notes:

System BIOS

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

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

The goal of the ROM 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 protects 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 become transparent to user programs.

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

System BIOS Usage

Access to BIOS is through program interrupts of the 80286 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 80286 in the real mode, INT 12H will invoke the BIOS routine for determining the memory size and return 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 1Kb 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.

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

80286 Program Interrupt Listing (Real Mode Only)

SECTION 5

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

Address	Interrupt	Function
80-83	20	DOS program terminate
84-87	21	DOS function call
88-8B	22	DOS terminate address
8c-8F	23	DOS Ctrl Break exit address
90-93	24	DOS fatal error vector
94-97	25	DOS absolute disk read
98-9B	26	DOS absolute disk write
9C-9F	27	DOS terminate, fix in storage
A0-FF	28-3F	Reserved for DOS
100-17F	40-5F	Reserved
180-19F	60-67	Reserved for user program interrupts
1A0-1BF	68-6F	Not used
1C0-1C3	70	IRQ 8 Realtime clock INT (BIOS entry RTC__INT)
1C4-1C7	71	IRQ 9 (BIOS entry RE__DIRECT)
1C8-1CB	72	IRQ 10 (BIOS entry D11)
1CC-1CF	73	IRQ 11 (BIOS entry D11)
1D0-1D3	74	IRQ 12 (BIOS entry D11)
1D4-1D7	75	IRQ 13 BIOS Redirect to NMI interrupt (BIOS entry INT__287)
1D8-1DB	76	IRQ 14 (BIOS entry D11)
1DC-1DF	77	IRQ 15 (BIOS entry D11)
1E0-1FF	78-7F	Not used
200-217	80-85	Reserved by BASIC
218-3C3	86-F0	Used by BASIC interpreter while BASIC is running
3C4-3FF	F1-FF	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 will be executed when the Ctrl and Break keys are pressed on the keyboard. The vector is invoked while responding to 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.

Control may be retained by this routine with the following problems:

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

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

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

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

Interrupt 1F--Graphics Character Extensions: When operating in graphics modes 320 x 200 or 640 x 200, the read/write character interface will form a character from the ASCII code point, using a set of dot patterns. ROM contains the dot patterns for the first 128 code points. For access to the second 128 code points, this vector must be established to point at a table of up to 1Kb, 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 an IBM Personal Computer AT Fixed Disk and Diskette Drive Adapter is installed, the BIOS routines use interrupt 40 to revector the diskette pointer.

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

Other Read/Write Memory Usage

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

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

The following figure shows the reserved memory locations.

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

Reserved Memory Locations

If you do a DEF SEG (default workspace segment):

SECTION 5

Offset	Length	
2E	2	Line number of current line being executed
347	2	Line number of last error
30	2	Offset into segment of start of program text
358	2	Offset into segment of start of variables (end of program text 1-1)
6A	1	Keyboard buffer contents 0=No characters in buffer 1=Characters in buffer
4E	1	Character color in graphics mode*

BASIC Workspace Variables

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

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

L	H
Hex 64	Hex 00

The following is a BIOS memory map.

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

BIOS Memory Map

BIOS Programming Hints

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

If an error is reported by the disk or diskette code, you should reset the drive adapter and retry the operation. A specified number of retries should be required for diskette reads to ensure that 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 programmer should restore the original environment. Failure to adhere to this practice may cause incompatibility with present and future applications.

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

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 via 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 far call to byte 3 of the ROM, which should be executable code. The adapter may now perform its power-on initialization tasks. The adapter's ROM should now return control to the BIOS routines by executing a far return.

System Board Additional ROM Modules

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

The absolute addresses hex E0000 through EFFFF are scanned in a 64K block in search of a valid checksum. Valid ROM is defined as follows:

Byte 0 Hex 55

Byte 1 Hex AA

Byte 2 Not used

Byte 3 Entry via a CALL FAR

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

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

Keyboard Encoding and Usage

Encoding

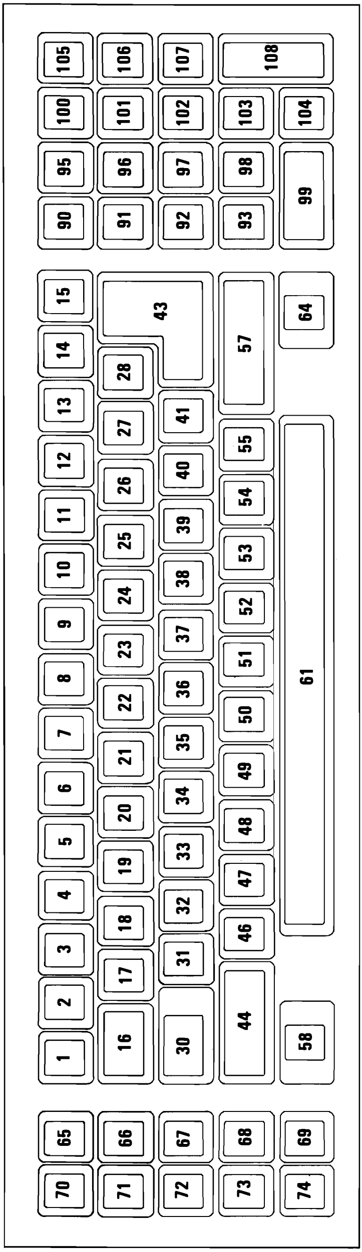
The keyboard routine provided by IBM in the ROM scan codes into what will be termed *Extended ASCII*

Extended ASCII encompasses one-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 following character codes 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 Section 7 for the exact codes.

The following figure is a keyboard layout showing the key positions.



Key	Base Case	Upper Case	Ctrl	Alt
90	Esc	Esc	Esc	-1
2	1	!	-1	Note 1
3	2	@	Nul(000) Note 1	Note 1
4	3	#	-1	Note 1
5	4	$	-1	Note 1
6	5	%	-1	Note 1
7	6	¬	RS(030)	Note 1
8	7	&	-1	Note 1
9	8	*	-1	Note 1
10	9	(	-1	Note 1
11	0	)	-1	Note 1
12	-	_	US(031)	Note 1
13	=	+	-1	Note 1
15	Backspace(008)	Backspace(008)	Del(127)	-1
16	→(009)	←(Note 1)	-1	-1
17	q	Q	DC1(017)	Note 1
18	w	W	ETB(023)	Note 1
19	e	E	ENQ(005)	Note 1
20	r	R	DC2(018)	Note 1
21	t	T	DC4(020)	Note 1
22	y	Y	EM(025)	Note 1
23	u	U	NAK(021)	Note 1
24	i	I	HT(009)	Note 1
25	o	O	SI(015)	Note 1
26	p	P	DLE(016)	Note 1
27	[	{	Esc(027)	Note 1
28	]	}	GS(029)	-1
43	CR	CR	LF(010)	-1
30 Ctrl	-1	-1	-1	-1
31	a	A	SOH(001)	Note 1
32	s	S	DC3(019)	Note 1
33	d	D	EOT(004)	Note 1
34	f	F	ACK(006)	Note 1
35	g	G	BEL(007)	Note 1
36	h	H	BS(008)	Note 1
37	j	J	LF(010)	Note 1
38	k	K	VT(011)	Note 1
39	l	L	FF(012)	Note 1
40	;	:	-1	-1
41	'	■	-1	-1
1	Ω	°	-1	-1
44 Shift	-1	-1	-1	-1
14	\	|	FS(028)	-1
46	z	Z	SUB(026)	Note 1
47	x	X	CAN(024)	Note 1
48	c	C	ETX(003)	Note 1
49	v	V	SYN(022)	Note 1
50	b	B	STX(022)	Note 1

(Part 1 of 1). **Character Codes**

SECTION 5

Key	Base Case	Upper Case	Ctrl	Alt
51	n	N	SO(014)	Note 1
52	m	M	CR(013)	Note 1
53	,	<	-1	-1
54	.	>	-1	-1
55	/	?	-1	-1
57 Shift	-1	-1	-1	-1
106	*	Note 2	Note 1	-1
56 Alt	-1	-1	-1	-1
61	SP	SP	SP	SP
64 Caps Lock Lock	-1	-1	-1	-1
70	Nul(Note 1)	Nul(Note 1)	Nul(Note 1)	Nul(Note 1)
65	Nul(Note 1)	Nul(Note 1)	Nul(Note 1)	Nul(Note 1)
71	Nul(Note 1)	Nul(Note 1)	Nul(Note 1)	Nul(Note 1)
66	Nul(Note 1)	Nul(Note 1)	Nul(Note 1)	Nul(Note 1)
72	Nul(Note 1)	Nul(Note 1)	Nul(Note 1)	Nul(Note 1)
67	Nul(Note 1)	Nul(Note 1)	Nul(Note 1)	Nul(Note 1)
73	Nul(Note 1)	Nul(Note 1)	Nul(Note 1)	Nul(Note 1)
68	Nul(Note 1)	Nul(Note 1)	Nul(Note 1)	Nul(Note 1)
74	Nul(Note 1)	Nul(Note 1)	Nul(Note 1)	Nul(Note 1)
69	Nul(Note 1)	Nul(Note 1)	Nul(Note 1)	Nul(Note 1)
95 Num Lock	-1	-1	Pause(Note 2)	-1
100 Scroll Lock	-1	-1	Break(Note 2)	-1

Notes:
1. Refer to Extended Codes in this section.
2. Refer to Special Handling in this section.

(Part 2 of 2) Character Codes

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

Key	Num Lock	Base Case	Alt	Ctrl
91	7	Home(Note 1)	-1	Clear Screen
96	8	↑(Note 1)	-1	-1
101	9	Page Up(Note 1)	-1	Top of Text and Home
107	-	-	-1	-1
92	4	←(Note 1)	-1	Reverse Word(Note 1)
97	5	-1	-1	-1
102	6	→(Note 1)	-1	Advance Word(Note 1)
108	+	+Note 1)	-1	-1
94	1	End(Note 1)	-1	Erase to EOL(Note 1)
98	2	↓(Note 1)	-1	-1
102	3	Page Down(Note 1)	-1	Erase to EOS(Note 1)
99	0	Ins	-1	-1
104	.	Del(Notes 1,2)	Note 2	Note 2

Notes:
Refer to Extended Codes in this section.
Refer to Special Handling in this section

Special Character Codes

Extended Codes

Extended Functions

For certain functions that cannot be represented by the 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.

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

Keyboard Extended Functions

Shift States

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

Shift: This key temporarily shifts keys 1-14, 16-28, 31-41, 46-55, 106, and 65-74 to uppercase (base case if in Caps lock state). Also, the Shift temporarily reverses the Num Lock or non-Num Lock state of keys 91-93, 96, 98, and 101-103.

Ctrl: This key temporarily shifts keys 3, 7, 13, 15, 17-28, 31-39, 46-52, 106, 65-74, 42, 101, 92, 102, 91, 93, 95, 100, and 103 to the Ctrl state. The Ctrl key is also used with the Alt and Del keys to cause the system-reset function; with the Scroll Lock key to cause the break function; and with the Num Lock key to cause the pause function. The system-reset, break, and pause functions are described under "Special Handling" later in this section.

Alt: This key temporarily shifts keys 1-13, 17-26, 31-39, 46-52, and 65-74 to the Alt state. The Alt key is also used with the Ctrl and Del keys to cause the system reset function.

The Alt key also allows the user to enter any character code from 0-255 into the system from the keyboard. The user holds down the Alt key and types the decimal value of the characters desired on the numeric keypad (keys 91-93, 96-98, and 101-103). The Alt key is then released. If more than three digits are typed, a modulo-256 result is created. These three digits are interpreted as a character code and are sent through the keyboard routine to the system or application program. Alt is handled internal to the keyboard routine.

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

Pause: The combination of the Ctrl and Num Lock keys causes the keyboard interrupt routine to loop, waiting for any key except Num Lock to be pressed. This provides a system- or application-transparent method of temporarily suspending list, print, etc. and then resuming the operation. The key used to resume operation is thrown away. Pause is handled internal to the keyboard routine.

Print Screen: The combination of the Shift and PrtSc keys results in an interrupt invoking the print screen routine. This routine works in the alphanumeric or graphics mode, with unrecognizable characters printing as blanks.

Caps Lock: This key shifts keys 17-26, 31-39, and 46-52 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 toggles 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: This key is interpreted by appropriate application programs as indicating that the use of cursor control keys should cause windowing over the text rather than cursor movement. 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 toggles 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 90-93 and 95-104 to upper case. 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 toggles 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.

Sys Req

When the Sys key is pressed, a hex 8500 is placed in AX, and an interrupt 15 is executed. When the Sys key is released, a hex 8501 is placed in AX, and another interrupt 15 is executed. If an application is to use the Sys key, 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

It is the responsibility of the application to preserve the value in all registers, except AX, upon return. Sys 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.

Special Handling

System Reset

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

Notes:

```
Warning: No STACK segment

 Start  Stop   Length  Name
 00000H 0FFFEH FFFFH  CODE

 Origin   Group
  Address         Publics by Name

 0000:E729        A1
 0000:3792        ACT_DISP_PAGE
 0000:E137        ADERR
 0000:E11C        ADERR1
 0000:17AA        BEEP
 0000:0000        BEGIN
 0000:16B9        BLINK_INT
 0000:E372        BOOT_INVA
 0000:E6F2        BOOT_STRAP
 0000:1B66        BOOT_STRAP_1
 0000:E05E        C1
 0000:0222        C11
 0000:E060        C2
 0000:0C3F        C21
 0000:0454        C30
 0000:0405        C8042
 0000:E062        C8042A
 0000:E066        C8042B
 0000:E068        C8042C
 0000:F859        CASSETTE_IO
 0000:3FE2        CASSETTE_IO_1
 0000:09FB        CHK_VIDEO
 0000:E234        CM1
 0000:E25D        CM2
 0000:E286        CM3
 0000:E0D0        CM4
 0000:E2C6        CM4_A
 0000:E2DF        CM4_B
 0000:E2F8        CM4_C
 0000:E311        CM4_D
 0000:FA6E        CRT_CHAR_GEN
 0000:E164        D1
 0000:1805        D11
 0000:E174        D2
 0000:E184        D2A
 0000:17FD        DDS
 0000:EC59        DISKETTE_IO
 0000:20A5        DISKETTE_IO_1
 0000:EFC7        DISK_BASE
 0000:EF57        DISK_INT
 0000:260E        DISK_INT_1
 0000:2A71        DISK_IO
 0000:28DA        DISK_SETUP
 0000:2816        DSKETTE_SETUP
 0000:FF53        DUMMY_RETURN
 0000:1851        DUMMY_RETURN_1
 0000:E06C        E0
 0000:E085        E0_A
 0000:E09E        E0_B
 0000:E0E9        E1
 0000:E32A        E1_A
 0000:E0FC        E1_B
 0000:E10C        E1_C
 0000:03E5        E30B
 0000:03EB        E30C
 0000:F84D        EQUIPMENT
 0000:3E6C        EQUIPMENT_1
 0000:177A        ERR_BEEP
 0000:187F        EXC_00
 0000:1884        EXC_01
 0000:1889        EXC_02
 0000:188E        EXC_03
 0000:1893        EXC_04
 0000:1898        EXC_05
 0000:18B1        EXC_06
 0000:18B6        EXC_07
 0000:18BB        EXC_08
 0000:18C0        EXC_09
 0000:18C5        EXC_10
 0000:18CA        EXC_11
 0000:18CF        EXC_12
 0000:18D4        EXC_13
 0000:18D9        EXC_14
 0000:18DE        EXC_15
 0000:18E3        EXC_16
 0000:18E8        EXC_17
 0000:18ED        EXC_18
 0000:18F2        EXC_19
 0000:18F7        EXC_20
 0000:18FC        EXC_21
 0000:1901        EXC_22
 0000:1906        EXC_23
 0000:190B        EXC_24
 0000:1910        EXC_25
 0000:1915        EXC_26
 0000:191A        EXC_27
 0000:191F        EXC_28
 0000:1924        EXC_29
 0000:1929        EXC_30
 0000:192E        EXC_31
 0000:1753        E_MSG
 0000:E1C2        F1
 0000:E393        F1780
 0000:E3A8        F1781
 0000:E3BD        F1782
 0000:E3DB        F1790
 0000:E3EE        F1791
 0000:E1FB        F1_A
 0000:E34E        F1_B
 0000:E21F        F3
 0000:E152        F3A
 0000:E15D        F3B
 0000:E18B        F3D
 0000:E1A1        F3D1
 0000:E2AC        F4
 0000:E2B2        F4E
 0000:E401        FD_TBL
 0000:4752        FILL
 0000:4392        GATE_A20
 0000:1FF0        GDT_BLD
 0000:1BC6        H5
 0000:2FA4        HD_INT
 0000:1852        INT_287
 0000:E8E1        K10
 0000:E91B        K11
 0000:E955        K12
 0000:E95F        K13
 0000:E969        K14
 0000:E976        K15
 0000:30A9        K16
 0000:E87E        K6
 0000:0008  Abs   K6L
 0000:E886        K7
 0000:E88E        K8
 0000:E8C8        K9
 0000:17D2        KBD_RESET
 0000:E987        KB_INT
 0000:3054        KB_INT_1
 0000:E82E        KEYBOARD_IO
 0000:2FC8        KEYBOARD_IO_1
 0000:E1D7        LOCK
 0000:0010  Abs   M4
 0000:F0E4        M5
 0000:F0EC        M6
 0000:F0F4        M7
 0000:F841        MEMORY_SIZE_DETERMINE
 0000:3E62        MEMORY_SIZE_DETERMINE_1
 0000:E2C3        NMI_INT
 0000:3E76        NMI_INT_1
 0000:0411        OBF_42
 0000:E064        OBF_42A
 0000:E06A        OBF_42B
 0000:002C        POST1
 0000:0C3F        POST2
 0000:16AD        POST3
 0000:1753        POST4
 0000:187F        POST5
 0000:199C        POST6
 0000:1C2D        POST7
 0000:EFD2        PRINTER_IO
 0000:346F        PRINTER_IO_1
 0000:FF54        PRINT_SCREEN
 0000:46CC        PRINT_SCREEN_1
 0000:174C        PROC_SHUTDOWN
 0000:1720        PROT_PRT_HEX
 0000:1719        PRT_HEX
 0000:186A        PRT_SEG
 0000:176C        P_MSG
 0000:FFF0        P_O_R
 0000:38F5        READ_AC_CURRENT
 0000:377B        READ_CURSOR
 0000:3A3B        READ_DOT
 0000:3DBC        READ_LPEN
 0000:1861        RE_DIRECT
```

SECTION 5

```
0000:16D0        ROM_CHECK
0000:1AF9        ROM_ERR
0000:16AD        ROS_CHECKSUM
0000:E739        RS232_IO
0000:34F5        RS232_IO_1
0000:462A        RTC_INT
0000:38A3        SCROLL_DOWN
0000:37FF        SCROLL_UP
0000:24C1        SEEK
0000:37B6        SET_COLOR
0000:3751        SET_CPOS
0000:372A        SET_CTYPE
0000:364E        SET_MODE
0000:3F2F        SET_TOD
0000:1197        SHUT2
0000:114A        SHUT3
0000:169B        SHUT4
0000:11BC        SHUT6
0000:119A        SHUT7
0000:4252        SHUT9
0000:1FF9        SIDT_BLD
0000:FF23        SLAVE_VECTOR_TABLE
0000:E05B        START
0000:00A6        START_1
0000:199C        STGTST_CNT
0000:1F1A        SYSINIT1
0000:1933        SYS_32
0000:1938        SYS_33
0000:193D        SYS_34
0000:1942        SYS_35
0000:1947        SYS_36
0000:194C        SYS_37
0000:1951        SYS_38
0000:FEA5        TIMER_INT
0000:4684        TIMER_INT_1
0000:FE6E        TIME_OF_DAY
0000:445C        TIME_OF_DAY_1
0000:03C7        TST4_B
0000:03D3        TST4_C
0000:03F7        TST4_D
0000:FEF3        VECTOR_TABLE
0000:F065        VIDEO_IO
0000:3605        VIDEO_IO_1
0000:F0A4        VIDEO_PARMS
0000:37DC        VIDEO_STATE
0000:E0B7        VIR_ERR
0000:393B        WRITE_AC_CURRENT
0000:396E        WRITE_C_CURRENT
0000:3A4C        WRITE_DOT
0000:3D38        WRITE_TTY
0000:1713        XLAT_PR
0000:1B25        XMIT_8042
0000:1708        XPC_BYTE

  Address          Publics by Value

0000:0000        BEGIN
0000:0008   Abs  K6L
0000:0010   Abs  M4
0000:002C        POST1
0000:00A6        START_1
0000:0222        C11
0000:03C7        TST4_B
0000:03D3        TST4_C
0000:03E5        E30B
0000:03EB        E30C
0000:03F7        TST4_D
0000:0405        C8042
0000:0411        OBF_42
0000:0454        C30
0000:09FB        CHK_VIDEO
0000:0C3F        POST2
0000:0C3F        C21
0000:114A        SHUT3
0000:1197        SHUT2
0000:119A        SHUT7
0000:11BC        SHUT6
0000:169B        SHUT4
0000:16AD        ROS_CHECKSUM
0000:16AD        POST3
0000:16B9        BLINK_INT
0000:16D0        ROM_CHECK
0000:1708        XPC_BYTE
0000:1713        XLAT_PR
0000:1719        PRT_HEX
0000:1720        PROT_PRT_HEX
0000:174C        PROC_SHUTDOWN
0000:1753        POST4
0000:1753        E_MSG
0000:176C        P_MSG
0000:177A        ERR_BEEP
0000:17AA        BEEP
0000:17D2        KBD_RESET
0000:17FD        DDS
0000:1805        D11
0000:1851        DUMMY_RETURN_1
0000:1852        INT_287
0000:1861        RE_DIRECT
0000:186A        PRT_SEG
0000:187F        EXC_00
0000:187F        POST5
0000:1884        EXC_01
0000:1889        EXC_02
0000:188E        EXC_03
0000:1893        EXC_04
0000:1898        EXC_05
0000:18B1        EXC_06
0000:18B6        EXC_07
0000:18BB        EXC_08
0000:18C0        EXC_09
0000:18C5        EXC_10
0000:18CA        EXC_11
0000:18CF        EXC_12
0000:18D4        EXC_13
0000:18D9        EXC_14
0000:18DE        EXC_15
0000:18E3        EXC_16
0000:18E8        EXC_17
0000:18ED        EXC_18
0000:18F2        EXC_19
0000:18F7        EXC_20
0000:18FC        EXC_21
0000:1901        EXC_22
0000:1906        EXC_23
0000:190B        EXC_24
0000:1910        EXC_25
0000:1915        EXC_26
0000:191A        EXC_27
0000:191F        EXC_28
0000:1924        EXC_29
0000:1929        EXC_30
0000:192E        EXC_31
0000:1933        SYS_32
0000:1938        SYS_33
0000:193D        SYS_34
0000:1942        SYS_35
0000:1947        SYS_36
0000:194C        SYS_37
0000:1951        SYS_38
0000:199C        POST6
0000:199C        STGTST_CNT
0000:1AF9        ROM_ERR
0000:1B25        XMIT_8042
0000:1B66        BOOT_STRAP_1
0000:1BC6        H5
0000:1C2D        POST7
0000:1F1A        SYSINIT1
0000:1FF0        GDT_BLD
0000:1FF9        SIDT_BLD
0000:20A5        DISKETTE_IO_1
0000:24C1        SEEK
0000:260E        DISK_INT_1
0000:2816        DSKETTE_SETUP
0000:28DA        DISK_SETUP
0000:2A71        DISK_IO
0000:2FA4        HD_INT
0000:2FC8        KEYBOARD_IO_1
0000:3054        KB_INT_1
0000:30A9        K16
0000:346F        PRINTER_IO_1
0000:34F5        RS232_IO_1
0000:3605        VIDEO_IO_1
0000:364E        SET_MODE
0000:372A        SET_CTYPE
0000:3751        SET_CPOS
0000:377B        READ_CURSOR
0000:3792        ACT_DISP_PAGE
0000:37B6        SET_COLOR
0000:37DC        VIDEO_STATE
0000:37FF        SCROLL_UP
0000:38A3        SCROLL_DOWN
0000:38F5        READ_AC_CURRENT
0000:393B        WRITE_AC_CURRENT
```

0000:396E	WRITE_C_CURRENT
0000:3A3B	READ_DOT
0000:3A4C	WRITE_DOT
0000:3D38	WRITE_TTY
0000:3DBC	READ_LPEN
0000:3E62	MEMORY_SIZE_DETERMINE_1
0000:3E6C	EQUIPMENT_1
0000:3E76	NMI_INT_1
0000:3F2F	SET_TOD
0000:3FE2	CASSETTE_IO_1
0000:4252	SHUT9
0000:4392	GATE_A20
0000:445C	TIME_OF_DAY_1
0000:462A	RTC_INT
0000:4684	TIMER_INT_1
0000:46CC	PRINT_SCREEN_1
0000:4752	FILL
0000:E05B	START
0000:E05E	C1
0000:E060	C2
0000:E062	C8042A
0000:E064	OBF_42A
0000:E066	C8042B
0000:E068	C8042C
0000:E06A	OBF_42B
0000:E06C	E0
0000:E085	E0_A
0000:E09E	E0_B
0000:E0B7	VIR_ERR
0000:E0D0	CM4
0000:E0E9	E1
0000:E0FC	E1_B
0000:E10C	E1_C
0000:E11C	ADERR1
0000:E137	ADERR
0000:E152	F3A
0000:E15D	F3B
0000:E164	D1
0000:E174	D2
0000:E184	D2A
0000:E18B	F3D
0000:E1A1	F3D1
0000:E1C2	F1
0000:E1D7	LOCK
0000:E1FB	F1_A
0000:E21F	F3
0000:E234	CM1
0000:E25D	CM2
0000:E286	CM3
0000:E2AC	F4
0000:E2B2	F4E
0000:E2C3	NMI_INT
0000:E2C6	CM4_A
0000:E2DF	CM4_B
0000:E2F8	CM4_C
0000:E311	CM4_D
0000:E32A	E1_A
0000:E34E	F1_B
0000:E372	BOOT_INVA
0000:E393	F1780
0000:E3A8	F1781
0000:E3BD	F1782
0000:E3DB	F1790
0000:E3EE	F1791
0000:E401	FD_TBL
0000:E6F2	BOOT_STRAP
0000:E729	A1
0000:E739	RS232_IO
0000:E82E	KEYBOARD_IO
0000:E87E	K6
0000:E886	K7
0000:E88E	K8
0000:E8C8	K9
0000:E8E1	K10
0000:E91B	K11
0000:E955	K12
0000:E95F	K13
0000:E969	K14
0000:E976	K15
0000:E987	KB_INT
0000:EC59	DISKETTE_IO
0000:EF57	DISK_INT
0000:EFC7	DISK_BASE
0000:EFD2	PRINTER_IO
0000:F065	VIDEO_IO
0000:F0A4	VIDEO_PARMS
0000:F0E4	M5
0000:F0EC	M6
0000:F0F4	M7
0000:F841	MEMORY_SIZE_DETERMINE
0000:F84D	EQUIPMENT
0000:F859	CASSETTE_IO
0000:FA6E	CRT_CHAR_GEN
0000:FE6E	TIME_OF_DAY
0000:FEA5	TIMER_INT
0000:FEF3	VECTOR_TABLE
0000:FF23	SLAVE_VECTOR_TABLE
0000:FF53	DUMMY_RETURN
0000:FF54	PRINT_SCREEN
0000:FFF0	P_O_R

```
                    TITLE TEST1 11/28/83 ROM POST

                    ;--------------------------------------------------------------------
                    ;                                                                  :
                    ; BIOS I/O INTERFACE                                               :
                    ;                                                                  :
                    ;       THESES INTERFACE LISTINGS, PROVIDE ACCESS TO BIOS ROUTINES :
                    ;       THESE BIOS ROUTINES ARE MEANT TO BE ACCESSED THROUGH       :
                    ;       SOFTWARE INTERRUPTS ONLY.  ANY ADDRESSES PRESENT IN        :
                    ;       THE LISTINGS  ARE INCLUDED  ONLY FOR  COMPLETENESS,        :
                    ;       NOT FOR  REFERENCE.   APPLICATIONS WHICH  REFERENCE        :
                    ;       ABSOLUTE   ADDRESSES   WITHIN   THE   CODE  SEGMENT        :
                    ;       VIOLATE THE STRUCTURE AND DESIGN OF BIOS.                  :
                    ;                                                                  :
                    ;--------------------------------------------------------------------
                    PAGE
                    ;----------------------------------------------------------------
                    ; MODULES REQUIRED
                    ;       DATA.SRC        -->     DATA AREA
                    ;       TEST1.SRC       -->     TEST.01 THRU TEST.16
                    ;       TEST2.SRC       -->     TEST.17 THRU TEST.22
                    ;       TEST3.SRC       -->     PROCEDURES
                    ;                                 ROS_CHECKSUM
                    ;                                 BLINK_INT
                    ;                                 ROM_CHECK
                    ;                                 XPC_BYTE
                    ;                                 PRT_HEX
                    ;                                 PROT_PRT_HEX
                    ;                                 PROC_SHUTDOWN
                    ;       TEST4.SRC       -->       E_MSG
                    ;                                 P_MSG
                    ;                                 BEEP
                    ;                                 ERR_BEEP
                    ;                                 KBD_RESET
                    ;                                 D11 DUMMY INT HANDLER
                    ;                                 INT13 - X287 HANDLER
                    ;                                 PRT_SEG
                    ;                                 DDS
                    ;                                 HARDWARE INT 9 HANDLER (TYPE 71)
                    ;       TEST5.SRC       -->     EXCEPTION INTERRUPTS
                    ;       TEST6.SRC       -->       STGTST_CNT
                    ;                                 ROM_ERR
                    ;                                 XMIT_8042
                    ;                                 BOOT_STRAP
                    ;       TEST7.SRC       -->     PROTECTED MODE TEST
                    ;       SYSINIT1.SRC    -->     BUILD PROTECTED MODE DESCRIPTORS
                    ;         GDT_BLD.SRC
                    ;         SIDT_BLD.SRC
                    ;       DSKETTE.SRC     -->     DISKETTE BIOS
                    ;       DISK.SRC        -->     HARD FILE BIOS
                    ;       KYBD.SRC        -->     KEYBOARD BIOS
                    ;       PRT.SRC         -->     PRINTER BIOS
                    ;       RS232.SRC       -->     RS232 BIOS
                    ;       VIDEO1.SRC      -->     VIDEO BIOS
                    ;       BIOS.SRC        -->       MEM_SIZE
                    ;                                 EQUIP_DET
                    ;                                 NMI
                    ;                                 SET_TOD
                    ;       BIOS1.SRC       -->     DUMMY_CASSETTE (INT 15)
                    ;                                 DEVICE OPEN
                    ;                                 DEVICE CLOSE
                    ;                                 PROGRAM TERMINATION
                    ;                                 EVENT WAIT
                    ;                                 JOYSTICK SUPPORT
                    ;                                 SYSTEM REQUEST KEY
                    ;                                 WAIT
                    ;                                 MOVE BLOCK
                    ;                                 EXTENDED MEMORY SIZE DETERMINE
                    ;                                 PROCESSOR TO VIRTUAL MODE
                    ;       BIOS2.SRC       -->       TIME OF DAY
                    ;                                 TIMER1 INT
                    ;                                 PRINT_SCREEN
                    ;       ORGS.SRC        -->     PC COMPATABILITY AND TABLES
                    ;                                 POST ERROR MESSAGES
                    ;----------------------------------------------------------------
                  C INCLUDE POSTEQU.SRC
                  C ;--------------------------------------------------------------------------
                  C ;               EQUATES                 :
                  C ;--------------------------------------------------------------------------
 = 0000           C TEST            EQU     0               ; CONDITIONAL ASM (TEST2.SRC)
 = 0000           C KY_LOCK         EQU     0               ; CONDITIONAL ASM (TEST2.SRC)
 = 0000           C KEY_NUMS        EQU     0               ; CONDITIONAL ASM (KYBD.SRC)
                  C ;--------------------------------------------------------------------------
 = 00F0           C X287            EQU     0F0H            ; MATH PROCESSOR
                  C ;--------------------------------------------------------------------------
 = 0020           C LOOP_POST       EQU     020H            ; MFG LOOP POST JUMPER
                  C ;--------------------------------------------------------------------------
 = 0010           C REFRESH_BIT     EQU     010H            ; REFRESH TEST BIT
                  C ;--------------------------------------------------------------------------
 = 0000           C POST_SS         EQU     0H              ; POST STACK SEGMENT
 = 8000           C POST_SP         EQU     8000H           ; POST STACK POINTER
 = FFFF           C TEMP_STACK_LO   EQU     0FFFFH          ;
 = 0000           C TEMP_STACK_HI   EQU     0               ; SET PROTECTED MODE TEMP_SS
                  C                                         ; 0:FFFFH
                  C ;--------------------------------------------------------------------------
 = 0060           C PORT_A          EQU     60H             ; 8042 KEYBOARD SCAN/DIAG OUTPUTS
 = 0061           C PORT_B          EQU     61H             ; 8042 READ WRITE REGISTER
 = 00C0           C PARITY_ERR      EQU     0C0H            ; RAM/IO CHANNEL PARITY ERROR
 = 00F3           C RAM_PAR_ON      EQU     11110011B       ; AND THIS VALUE
 = 000C           C RAM_PAR_OFF     EQU     00001100B       ; OR THIS VALUE
 = 0040           C IO_CHK          EQU     01000000B       ; IO CHECK?
 = 0080           C PRTY_CHK        EQU     10000000B       ; PARITY CHECK?
                  C
 = 0064           C STATUS_PORT     EQU     64H             ;8042 STATUS PORT
 = 0001           C OUT_BUF_FULL    EQU     01H             ; 0 = +OUTPUT BUFFER FULL
 = 0002           C INPT_BUF_FULL   EQU     02H             ; 1 = +INPUT BUFFER FULL
 = 0004           C SYS_FLAG        EQU     04H             ; 2 = -SYSTEM FLAG -POR/-SELF TEST
 = 0008           C CMD_DATA        EQU     08H             ; 3 = -COMMAND/+DATA
 = 0010           C KYBD_INH        EQU     10H             ; 4 = +KEYBOARD INHIBITED
 = 0020           C TRANS_TMOUT     EQU     20H             ; 5 = +TRANSMIT TIMEOUT
 = 0040           C RCV_TMOUT       EQU     40H             ; 6 = +RECEIVE TIME OUT
 = 0080           C PARITY_EVEN     EQU     80H             ; 7 = +PARITY IS EVEN
 = 00FE           C SHUT_CMD        EQU     0FEH            ; CAUSE A SHUTDOWN COMMAND
 = 00AB           C INTR_FACE_CK    EQU     0ABH            ; CHECK 8042 INTERFACE CMD
 = 00E0           C KYBD_CLK_DATA   EQU     0E0H            ; GET KYBD CLOCK AND DATA CMD
 = 0001           C KYBD_CLK        EQU     001H            ; KEYBOARD CLOCK BIT 0
                  C ;----------MANUFACTURING PORT--------------------------------------------
 = 0080           C MFG_PORT        EQU     80H             ; MANUFACTURING CHECKPOINT PORT
                  C ;----------MANUFACTURING BIT DEFINITION FOR MFG_ERR_FLAG+1------------------
 = 0001           C MEM_FAIL        EQU     00000001B      ; STORAGE TEST FAILED (ERROR 20X)
 = 0002           C PRO_FAIL        EQU     00000010B      ; VIRTUAL MODE TEST FAILED (ERROR 104)
 = 0004           C LMCS_FAIL       EQU     00000100B      ; LOW MEG CHIP SELECT FAILED (ERROR 109)
 = 0008           C KYCLK_FAIL      EQU     00001000B      ; KEYBOARD CLOCK TEST FAILED (ERROR 304)
 = 0010           C KY_SYS_FAIL     EQU     00010000B      ; KEYBOARD OR SYSTEM FAILED (ERROR 303)
 = 0020           C KYBD_FAIL       EQU     00100000B      ; KEYBOARD FAILED (ERROR 301)
 = 0040           C DSK_FAIL        EQU     01000000B      ; DISKETTE TEST FAILED (ERROR 601)
```

SECTION 5

```
= 0080                  C   KEY_FAIL        EQU     10000000B       ; KEYBOARD LOCKED (ERROR 302)
                        C   ;----------8042 INPUT PORT BIT DEFINITION-----------------------------------
= 0010                  C   BASE_RAM        EQU     10H             ;BASE R/W MEMORY
= 0020                  C   MFG_JMP         EQU     20H             ;LOOP POST JUMPER
= 0040                  C   DSP_JMP         EQU     40H             ;DISPLAY TYPE JUMPER
= 0080                  C   KEY_BD_INHIB    EQU     80H             ;KEYBOARD INHIBIT SWITCH
                        C   ;----------8042 RAM DEFINITION-----------------------------------------------
= 0010                  C   INH_KEYBOARD    EQU     10H             ;BYTE 0 BIT 4 OF 8042 RAM
                        C   ;-------------- COMMANDS ----------------------------------------------------
= 0020                  C   READ_8042_RAM   EQU     20H             ; BITS 0-4 = ADDRESS (20-3F)
= 0060                  C   WRITE_8042_RAM  EQU     60H             ;
= 00AA                  C   SELF_8042_TEST  EQU     0AAH            ; 8042 SELF TEST
= 00C0                  C   READ_8042_INPUT EQU     0C0H            ; READ 8042 INPUT PORT
= 00AE                  C   ENA_KBD         EQU     0AEH            ; ENABLE KEYBOARD COMMAND
= 00AD                  C   DIS_KBD         EQU     0ADH            ; DISABLE KEYBOARD COMMAND
= 00DF                  C   ENABLE_BIT20    EQU     0DFH            ; ENABLE ADDR LINE BIT 20
= 00DD                  C   DISABLE_BIT20   EQU     0DDH            ; DISABLE ADDR LINE BIT 20
                        C   ;-------------- KEYBOARD/LED COMMANDS ---------------------------------------
= 00F1                  C   KB_MENU         EQU     0F1H            ; SELECT MENU COMMAND
= 00F4                  C   KB_ENABLE       EQU     0F4H            ; KEYBOARD ENABLE
= 00F7                  C   KB_MAKE_BREAK   EQU     0F7H            ; TYPAMATIC
= 00FE                  C   KB_ECHO         EQU     0FEH            ; ECHO COMMAND
= 00FF                  C   KB_RESET        EQU     0FFH            ; SELF DIAGNOSTIC COMMAND
= 00ED                  C   LED_CMD         EQU     0EDH            ; LED WRITE COMMAND
                        C   ;--------------- KEYBOARD RESPONSE ------------------------------------------
= 00AA                  C   KB_OK           EQU     0AAH            ; RESPONSE FROM SELF DIAG
= 00FA                  C   KB_ACK          EQU     0FAH            ; ACKNOWLEDGE FROM TRANSMISSION
= 00FF                  C   KB_OVER_RUN     EQU     0FFH            ; OVER RUN
= 00FE                  C   KB_RESEND       EQU     0FEH            ; RESEND REQUEST
= 00F0                  C   KB_BREAK        EQU     0F0H            ; KEYBOARD BREAK CODE
= 0010                  C   KB_FA           EQU     010H            ; ACK RECEIVED
= 0020                  C   KB_FE           EQU     020H            ; RESEND RECEIVED FLAG
= 0040                  C   KB_PR_LED       EQU     040H            ; MODE INDICATOR UPDATE
                        C   ;--------------- CMOS EQUATES -----------------------------------------------
= 0070                  C   CMOS_PORT       EQU     070H            ; IO ADDRESS OF CMOS PORT
= 008A                  C   CLK_UP          EQU     08AH            ; CLOCK UPDATE STATUS
= 008B                  C   CMOS_ALARM      EQU     08BH            ;
= 0090                  C   CMOS_BEGIN      EQU     090H            ;
= 00AD                  C   CMOS_END        EQU     0ADH            ;
= 008F                  C   SHUT_DOWN       EQU     08FH            ; SHUTDOWN OFFSET
= 008D                  C   BATTERY_COND_STATUS EQU 08DH            ; BATTERY STATUS
= 00B1                  C   M_SIZE_HI       EQU     0B1H            ; IO MEMORY SIZE HIGH BYTE (POST)
= 00B0                  C   M_SIZE_LO       EQU     0B0H            ; IO MEMORY SIZE LO BYTE   (POST)
= 0096                  C   M1_SIZE_HI      EQU     096H            ; 0->640K CONFIG MEMORY SIZE (SETUP)
= 0095                  C   M1_SIZE_LO      EQU     095H            ;   LOW BYTE (SETUP)
= 0098                  C   M2_SIZE_HI      EQU     098H            ; 640K->UP CONFIG MEMORY SIZE (SETUP)
= 0097                  C   M2_SIZE_LO      EQU     097H            ;   LOW BYTE (SETUP)
= 0094                  C   C_EQUIP         EQU     094H            ; CMOS EQUIPMENT FLAG
= 0092                  C   HD_FILE_TYPE    EQU     092H            ; HARD FILE TYPE BYTE
                        C   PAGE
                        C   ;--------------- CMOS DIAG_STATUS ERROR FLAGS-------------------------------
= 008E                  C   DIAG_STATUS     EQU     08EH            ; CMOS ADDRESS OF DIAG_STATUS
= 0080                  C   BAD_BAT         EQU     080H            ; DEAD BATTERY
= 0040                  C   BAD_CKSUM       EQU     040H            ; CHECKSUM ERROR
= 0020                  C   BAD_CONFIG      EQU     020H            ; MINIMUM CONFIG USED INSTEAD OF CMOS
= 0010                  C   W_MEM_SIZE      EQU     010H            ; MEMORY SIZE NOT EQUAL TO CONFIG
= 0008                  C   HF_FAIL         EQU     008H            ; HARD FILE FAILURE ON INIT
= 0004                  C   CMOS_CLK_FAIL   EQU     004H            ; CMOS CLK NOT UPDATING OR NOT VALID
                        C   ;--------------- CMOS INFORMATION FLAGS-------------------------------------
= 00B3                  C   INFO_STATUS     EQU     0B3H            ; CMOS ADDRESS OF INFO BYTE
= 0080                  C   M640K           EQU     080H            ; 512K -> 640K CARD INSTALLED
= 0040                  C   NEW_INST        EQU     040H            ; FLAG USED BY CMOS SETUP UTILITY
= 0020                  C   HF_BOOT         EQU     020H            ; BOOT HARD FILE FLAG
                        C   ;--------------- INTERRUPT EQUATES ------------------------------------------
= 0020                  C   INTA00          EQU     20H             ; 8259 PORT
= 0021                  C   INTA01          EQU     21H             ; 8259 PORT
= 0020                  C   EOI             EQU     20H
= 00A0                  C   INTB00          EQU     0A0H            ; 2ND 8259
= 00A1                  C   INTB01          EQU     0A1H            ;
= 0070                  C   INT_TYPE        EQU     070H            ; START OF 8259 INTERRUPT TABLE LOCATION
= 0010                  C   INT_VIDEO       EQU     010H            ; VIDEO VECTOR
                        C   ;---------------------------------------------------------------------------
= 0040                  C   TIMER           EQU     40H
= 0043                  C   TIM_CTL         EQU     43H             ; 8253 TIMER CONTROL PORT ADDR
= 0040                  C   TIMER0          EQU     40H             ; 8253 TIMER/CNTER 0 PORT ADDR
= 0001                  C   TMINT           EQU     01              ; TIMER 0 INTR RECVD MASK
                        C   ;---------------------------------------------------------------------------
= 0008                  C   DMA08           EQU     08              ; DMA STATUS REG PORT ADDR
= 0000                  C   DMA             EQU     00              ; DMA CH.0 ADDR. REG PORT ADDR
                        C   ;---------------------------------------------------------------------------
= 00D0                  C   DMA18           EQU     0D0H            ; 2ND DMA STATUS PORT ADDR
= 00C0                  C   DMA1            EQU     0C0H            ; 2ND DMA CH.0 ADDR. REG PORT ADDR
                        C   ;---------------------------------------------------------------------------
= 0081                  C   DMA_PAGE        EQU     81H             ; START OF DMA PAGE REGISTERS
= 008F                  C   LAST_DMA_PAGE   EQU     8FH             ; LAST DMA PAGE REGISTER
                        C   ;---------------------------------------------------------------------------
= 0540                  C   MAX_PERIOD      EQU     540H
= 0410                  C   MIN_PERIOD      EQU     410H
= 0060                  C   KBD_IN          EQU     60H             ; KEYBOARD DATA IN ADDR PORT
= 0002                  C   KBDINT          EQU     02              ; KEYBOARD INTR MASK
= 0060                  C   KB_DATA         EQU     60H             ; KEYBOARD SCAN CODE PORT
= 0061                  C   KB_CTL          EQU     61H             ; CONTROL BITS FOR KEYBOARD SENSE DATA
= 0080                  C   KB_ERR          EQU     80H             ; KEYBOARD TRANSMIT ERROR FLAG
                        C   ;----- SHIFT FLAG EQUATES WITHIN KB_FLAG
= 0080                  C   INS_STATE       EQU     80H             ; INSERT STATE IS ACTIVE
= 0040                  C   CAPS_STATE      EQU     40H             ; CAPS LOCK STATE HAS BEEN TOGGLED
= 0020                  C   NUM_STATE       EQU     20H             ; NUM LOCK STATE HAS BEEN TOGGLED
= 0010                  C   SCROLL_STATE    EQU     10H             ; SCROLL LOCK STATE HAS BEEN TOGGLED
= 0008                  C   ALT_SHIFT       EQU     08H             ; ALTERNATE SHIFT KEY DEPRESSED
= 0004                  C   CTL_SHIFT       EQU     04H             ; CONTROL SHIFT KEY DEPRESSED
= 0002                  C   LEFT_SHIFT      EQU     02H             ; LEFT SHIFT KEY DEPRESSED
= 0001                  C   RIGHT_SHIFT     EQU     01H             ; RIGHT SHIFT KEY DEPRESSED
= 0080                  C   INS_SHIFT       EQU     80H             ; INSERT KEY IS DEPRESSED
= 0040                  C   CAPS_SHIFT      EQU     40H             ; CAPS LOCK KEY IS DEPRESSED
= 0020                  C   NUM_SHIFT       EQU     20H             ; NUM LOCK KEY IS DEPRESSED
= 0010                  C   SCROLL_SHIFT    EQU     10H             ; SCROLL LOCK KEY IS DEPRESSED
= 0008                  C   HOLD_STATE      EQU     08H             ; SUSPEND KEY HAS BEEN TOGGLED
= 0004                  C   SYS_SHIFT       EQU     04H             ; SYSTEM KEY DEPRESSED AND HELD
= 0045                  C   NUM_KEY         EQU     69              ; SCAN CODE FOR NUMBER LOCK
= 0046                  C   SCROLL_KEY      EQU     70              ; SCROLL LOCK KEY
= 0038                  C   ALT_KEY         EQU     56              ; ALTERNATE SHIFT KEY SCAN CODE
= 001D                  C   CTL_KEY         EQU     29              ; SCAN CODE FOR CONTROL KEY
= 003A                  C   CAPS_KEY        EQU     58              ; SCAN CODE FOR SHIFT LOCK
= 002A                  C   LEFT_KEY        EQU     42              ; SCAN CODE FOR LEFT SHIFT
= 0036                  C   RIGHT_KEY       EQU     54              ; SCAN CODE FOR RIGHT SHIFT
= 0052                  C   INS_KEY         EQU     82              ; SCAN CODE FOR INSERT KEY
= 0053                  C   DEL_KEY         EQU     83              ; SCAN CODE FOR DELETE KEY
= 0054                  C   SYS_KEY         EQU     54H             ; SCAN CODE FOR SYSTEM KEY
                        C   ;-------------- DISKETTE EQUATES
= 0080                  C   INT_FLAG        EQU     080H            ; INTERRUPT OCCURRENCE FLAG
= 0025                  C   MOTOR_WAIT      EQU     37              ; 2 SECS OF COUNTS FOR MOTOR TURN OFF
= 0080                  C   TIME_OUT        EQU     80H             ; ATTACHMENT FAILED TO RESPOND
= 0040                  C   BAD_SEEK        EQU     40H             ; SEEK OPERATION FAILED
= 0020                  C   BAD_NEC         EQU     20H             ; NEC CONTROLLER HAS FAILED
= 0010                  C   BAD_CRC         EQU     10H             ; BAD CRC ON DISKETTE READ
= 0009                  C   DMA_BOUNDARY    EQU     09H             ; ATTEMPT TO DMA ACROSS 64K BOUNDARY
```

```
= 0008                 C  BAD_DMA         EQU     08H           ; DMA OVERRUN ON OPERATION
= 0006                 C  MEDIA_CHANGE    EQU     06H           ; MEDIA REMOVED ON DUAL ATTACH CARD
= 0004                 C  RECORD_NOT_FND  EQU     04H           ; REQUESTED SECTOR NOT FOUND
= 0003                 C  WRITE_PROTECT   EQU     03H           ; WRITE ATTEMPTED ON WRITE PROT DISK
= 0002                 C  BAD_ADDR_MARK   EQU     02H           ; ADDRESS MARK NOT FOUND
= 0001                 C  BAD_CMD         EQU     01H           ; BAD COMMAND PASSED TO DISKETTE I/O
                       C
= 0002                 C  XRATE           EQU     02H           ; 250KBS DATA TRANSFER RATE
= 0001                 C  DUAL            EQU     01H           ; DUAL ATTACH CARD PRESENT FLAG
                       C
= 0080                 C  DSK_CHG         EQU     080H          ; DISKETTE CHANGE FLAG MASK BIT
= 0007                 C  STATE_MSK       EQU     007H          ; USED TO STRIP OFF STATE OF MEDIA
= 00F8                 C  REV_STATE       EQU     0F8H          ; USED AS MASK FOR STATE BITS
= 0010                 C  DETERMINED      EQU     010H          ; SET STATE DETERMINED IN STATE BITS
= 0003                 C  TRAN_MSK        EQU     03H           ; ISOLATE SHIFTED TRANSFER RATE BITS
= 0020                 C  DOUBLE_STEP     EQU     020H          ; MASK TO TURN ON DOUBLE STEPPING
= 00F0                 C  MOTOR_MSK       EQU     0F0H          ; MASK TO CLEAR MOTOR ON BITS
= 0002                 C  MAX_DRV         EQU     002H          ; MAX NUMBER OF DRIVES
= 0010                 C  HOME            EQU     010H          ; TRACK 0 MASK
= 0004                 C  SENSE_DRV_ST    EQU     004H          ; SENSE DRIVE STATUS COMMAND
= 0001                 C  ONE             EQU     001H          ; SEEK ONE TRACK
= 0030                 C  TRK_SLAP        EQU     030H          ; CRASH STOP (48 TPI DRIVES)
= 000A                 C  QUIET_SEEK      EQU     00AH          ; SEEK TO TRACK 10
= 000F                 C  HD12_SETTLE     EQU     015D          ; 1.2 M HEAD SETTLE TIME
= 0014                 C  HD320_SETTLE    EQU     020D          ; 320 K HEAD SETTLE TIME
= 0080                 C  WRITE_OP        EQU     080H          ; WRITE OPERATION FLAG
                       C  PAGE
                       C  ;------ DISK CHANGE LINE EQUATES
= 0001                 C  NOCHGLN         EQU     001H          ; NO DISK CHANGE LINE AVAILABLE
= 0002                 C  CHGLN           EQU     002H          ; DISK CHANGE LINE AVAILABLE
                       C  ;------ MEDIA/DRIVE STATE INDICATORS
= 0093                 C  M326D326        EQU     093H          ; STATE MACHINE - 320/360 MEDIA/DRIVE
= 0074                 C  M326D12         EQU     074H          ; STATE MACHINE - 320/360 MEDIA,1.2DRIVE
= 0015                 C  M12D12          EQU     015H          ; STATE MACHINE - 1.2 MEDIA/DRIVE
= 0061                 C  POA_DUAL        EQU     061H          ; 300K DATA TRANSFER RATE & STATE 1
= 0080                 C  POA_START       EQU     080H          ; 250K DATA TRANSFER RATE & STATE 0
                       C  ;------ CMOS NON-VOLATILE RAM EQUATES
= 000E                 C  CMOSDSB_ADDR    EQU     00EH          ; DISKETTE STATUS BYTE ADDRESS
= 0070                 C  CADR_PRT        EQU     070H          ; CMOS ADDRESS PORT ADDRESS
= 0071                 C  CDATA_PRT       EQU     071H          ; CMOS DATA PORT ADDRESS
= 00C0                 C  CMOS_GOOD       EQU     0C0H          ; BATTERY AND CHECKSUM INDICATOR
= 0010                 C  CMOSDSK_BYTE    EQU     010H          ; DISKETTE BYTE ADDRESS
= 000F                 C  LOWNIB          EQU     00FH          ; ISOLATE LOW NIBBLE IN REGISTER MASK
= 0002                 C  INVALID_DRV     EQU     002H          ; FIRST INVALID DISKETTE TYPE
                       C  ;----------------------------------------
                       C  ;        TIMER DATA AREA               :
                       C  ;----------------------------------------
                       C  ; COUNTS_SEC    EQU     18
                       C  ; COUNTS_MIN    EQU     1092
                       C  ; COUNTS_HOUR   EQU     65543
                       C  ; COUNTS_DAY    EQU     1573040 = 1800B0H
                       C  PAGE
                       C
                       C  INCLUDE DSEG.SRC
                       C  ;----------------------------------------
                       C  ;        0286 INTERRUPT LOCATIONS (READ):
                       C  ;----------------------------------------
0000                   C  ABS0    SEGMENT AT 0
0000                   C  STG_LOC0                LABEL   BYTE
0008                   C          ORG 2*4
0008                   C  NMI_PTR                 LABEL   WORD
0014                   C          ORG 5*4
0014                   C  INT5_PTR                LABEL   WORD
0020                   C          ORG 8*4
0020                   C  INT_ADDR                LABEL   WORD
0020                   C  INT_PTR                 LABEL   DWORD
0040                   C          ORG 10H*4
0040                   C  VIDEO_INT               LABEL   WORD
004C                   C          ORG 13H*4                     ; NEW FDISK
004C                   C  ORG_VECTOR              LABEL   DWORD
0060                   C          ORG 18H*4
0060                   C  BASIC_PTR               LABEL   WORD
0064                   C          ORG 19H*4
0064                   C  BOOT_VEC                LABEL   DWORD
0064                   C  BOOT_VECTOR             LABEL   DWORD
0074                   C          ORG 1DH*4
0074                   C  PARM_PTR                LABEL   DWORD   ; POINTER TO VIDEO PARMS
0078                   C          ORG 1EH*4
0078                   C  DISK_POINTER            LABEL   DWORD
007C                   C          ORG 01FH*4
007C                   C  EXT_PTR                 LABEL   DWORD
0100                   C          ORG 40H*4                     ; DISKETTE POINTER
0100                   C  DISK_VECTOR             LABEL   DWORD
0104                   C         ORG 41H*4
0104                   C  HF_TBL_VEC              LABEL   DWORD
0118                   C          ORG 46H*4
0118                   C  HF1_TBL_VEC             LABEL   DWORD
01C0                   C          ORG 70H*4
01C0                   C  SLAVE_INT_PTR           LABEL   DWORD
01C0                   C  RTC_INT_VEC             LABEL   DWORD   ; REAL TIME CLOCK INT
01D8                   C          ORG 76H*4                     ; FIXED DISK INTERRUPT VECTOR
01D8                   C  HDISK_INT               LABEL   DWORD
0400                   C          ORG     400H
0400                   C  DATA_AREA                LABEL BYTE   ;ABSOLUTE LOCATION OF DATA SEGMENT
0400                   C  DATA_WORD                LABEL WORD
0500                   C          ORG     0500H
0500                   C  MFG_TEST_RTN             LABEL   FAR
7C00                   C          ORG     7C00H
7C00                   C  BOOT_LOCN                LABEL   FAR
7C00                   C  ABS0    ENDS
                       C  PAGE
                       C  ;------------------------------------------------
                       C  ; STACK -- USED DURING INITIALIZATION ONLY      :
                       C  ;------------------------------------------------
0000                   C  STACK   SEGMENT AT 30H
0000      80 [         C          DW      128 DUP(?)
             ????      C
                  ]    C
                       C
0100                   C  TOS     LABEL   WORD
0100                   C  STACK   ENDS
                       C  ;----------------------------------------
                       C  ;        ROM BIOS DATA AREAS           :
                       C  ;----------------------------------------
0000                   C  DATA    SEGMENT AT 40H
0000                   C  DATA_BASE       LABEL   BYTE
0000      04 [         C  RS232_BASE      DW      4 DUP(?)      ; ADDRESSES OF RS232 ADAPTERS
             ????      C
                  ]    C
                       C
0008      04 [         C  PRINTER_BASE    DW      4 DUP(?)      ; ADDRESSES OF PRINTERS
             ????      C
                  ]    C
                       C
0010      01 [         C  EQUIP_FLAG      DW      1 DUP(?)      ; INSTALLED HARDWARE
             ????      C
```

SECTION 5

```
                        ]       C
                                C
0012      01 [                  C  MFG_TST        DB      1 DUP(?)        ; INITIALIZATION FLAG
                 ??             C
                      ]         C
                                C
0013      01 [                  C  MEMORY_SIZE    DW      1 DUP(?)        ; MEMORY SIZE IN K BYTES
                 ????           C
                        ]       C
                                C
0015      01 [                  C  MFG_ERR_FLAG   DB      1 DUP(?)        ; SCRATCHPAD FOR MANUFACTURING
                 ??             C
                      ]         C
                                C
0016      01 [                  C                 DB      1 DUP(?)        ; ERROR CODES
                 ??             C
                      ]         C
                                C
                                C  PAGE
                                C  ;----------------------------------------
                                C  ;       KEYBOARD DATA AREAS            :
                                C  ;----------------------------------------
0017      01 [                  C  KB_FLAG        DB      1 DUP(?)
                 ??             C
                      ]         C
                                C
0018      01 [                  C  KB_FLAG_1      DB      1 DUP(?)        ; SECOND BYTE OF KEYBOARD STATUS
                 ??             C
                      ]         C
                                C
0019      01 [                  C  ALT_INPUT      DB      1 DUP(?)        ; STORAGE FOR ALTERNATE KEYPAD ENTRY
                 ??             C
                      ]         C
                                C
001A      01 [                  C  BUFFER_HEAD    DW      1 DUP(?)        ; POINTER TO HEAD OF KEYBOARD BUFFER
                 ????           C
                        ]       C
                                C
001C      01 [                  C  BUFFER_TAIL    DW      1 DUP(?)        ; POINTER TO TAIL OF KEYBOARD BUFFER
                 ????           C
                        ]       C
                                C
001E      10 [                  C  KB_BUFFER      DW      16 DUP(?)       ; ROOM FOR 15 ENTRIES
                 ????           C
                        ]       C
                                C
003E                            C  KB_BUFFER_END  LABEL   WORD
                                C
                                C  ;------ HEAD = TAIL INDICATES THAT THE BUFFER IS EMPTY
                                C
                                C  ;----------------------------------------
                                C  ;       DISKETTE DATA AREAS            :
                                C  ;----------------------------------------
003E      01 [                  C  SEEK_STATUS    DB      1 DUP(?)        ; DRIVE RECALIBRATION STATUS
                 ??             C
                      ]         C
                                C
                                C                                         ; BIT 3-0 = DRIVE 3-0 NEEDS RECAL
                                C                                         ; BEFORE NEXT SEEK IF BIT IS = 0
003F      01 [                  C  MOTOR_STATUS   DB      1 DUP(?)        ; MOTOR STATUS
                 ??             C
                      ]         C
                                C
                                C                                         ; BIT 3-0 = DRIVE 3-0 IS CURRENTLY
                                C                                         ;   RUNNING
                                C                                         ; BIT 7 = CURRENT OPERATION IS A WRITE,
                                C                                         ;   REQUIRES DELAY
0040      01 [                  C  MOTOR_COUNT    DB      1 DUP(?)        ; TIME OUT COUNTER FOR DRIVE TURN OFF
                 ??             C
                      ]         C
                                C
0041      01 [                  C  DISKETTE_STATUS DB     1 DUP(?)        ; RETURN CODE STATUS BYTE
                 ??             C
                      ]         C
                                C
0042                            C  CMD_BLOCK      LABEL   BYTE
0042                            C  HD_ERROR       LABEL   BYTE
0042      07 [                  C  NEC_STATUS     DB      7 DUP(?)        ; STATUS BYTES FROM NEC
                 ??             C
                      ]         C
                                C
                                C  PAGE
                                C  ;----------------------------------------
                                C  ;       VIDEO DISPLAY DATA AREA        :
                                C  ;----------------------------------------
0049      01 [                  C  CRT_MODE       DB      1 DUP(?)        ; CURRENT CRT MODE
                 ??             C
                      ]         C
                                C
004A      01 [                  C  CRT_COLS       DW      1 DUP(?)        ; NUMBER OF COLUMNS ON SCREEN
                 ????           C
                        ]       C
                                C
004C      01 [                  C  CRT_LEN        DW      1 DUP(?)        ; LENGTH OF REGEN IN BYTES
                 ????           C
                        ]       C
                                C
004E      01 [                  C  CRT_START      DW      1 DUP(?)        ; STARTING ADDRESS IN REGEN BUFFER
                 ????           C
                        ]       C
                                C
0050      08 [                  C  CURSOR_POSN    DW      8 DUP(?)        ; CURSOR FOR EACH OF UP TO 8 PAGES
                 ????           C
                        ]       C
                                C
0060      01 [                  C  CURSOR_MODE    DW      1 DUP(?)        ; CURRENT CURSOR MODE SETTING
                 ????           C
                        ]       C
                                C
0062      01 [                  C  ACTIVE_PAGE    DB      1 DUP(?)        ; CURRENT PAGE BEING DISPLAYED
                 ??             C
                      ]         C
                                C
0063      01 [                  C  ADDR_6845      DW      1 DUP(?)        ; BASE ADDRESS FOR ACTIVE DISPLAY CARD
                 ????           C
                        ]       C
                                C
0065      01 [                  C  CRT_MODE_SET   DB      1 DUP(?)        ; CURRENT SETTING OF THE 3X8 REGISTER
                 ??             C
                      ]         C
                                C
0066      01 [                  C  CRT_PALLETTE   DB      1 DUP(?)        ; CURRENT PALLETTE SETTING COLOR CARD
                 ??             C
                      ]         C
                                C
                                C  PAGE
```

```
                                C   ;----------------------------------------
                                C   ;        POST DATA AREA                 :
                                C   ;----------------------------------------
0067      01 [                  C   IO_ROM_INIT     DW      1 DUP(?)        ; PNTR TO OPTIONAL I/O ROM INIT ROUTINE
                ????            C
                        ]       C
                                C
0069      01 [                  C   IO_ROM_SEG      DW      1 DUP(?)        ; POINTER TO IO ROM SEGMENT
                ????            C
                        ]       C
                                C
006B      01 [                  C   INTR_FLAG       DB      1 DUP(?)        ; FLAG TO INDICATE AN INTERRUPT HAPPEND
                ??              C
                    ]           C
                                C
                                C
                                C   ;----------------------------------------
                                C   ;        TIMER DATA AREA                :
                                C   ;----------------------------------------
006C      01 [                  C   TIMER_LOW       DW      1 DUP(?)        ; LOW WORD OF TIMER COUNT
                ????            C
                        ]       C
                                C
006E      01 [                  C   TIMER_HIGH      DW      1 DUP(?)        ; HIGH WORD OF TIMER COUNT
                ????            C
                        ]       C
                                C
0070      01 [                  C   TIMER_OFL       DB      1 DUP(?)        ; TIMER HAS ROLLED OVER SINCE LAST READ
                ??              C
                    ]           C
                                C
                                C
                                C   ;----------------------------------------
                                C   ;        SYSTEM DATA AREA               :
                                C   ;----------------------------------------
0071      01 [                  C   BIOS_BREAK      DB      1 DUP(?)        ; BIT 7=1 IF BREAK KEY HAS BEEN HIT
                ??              C
                    ]           C
                                C
0072      01 [                  C   RESET_FLAG      DW      1 DUP(?)        ; WORD=1234H IF KEYBOARD RESET UNDERWAY
                ????            C
                        ]       C
                                C
                                C   PAGE
                                C   ;----------------------------------------
                                C   ;        HARD FILE DATA AREAS           :
                                C   ;----------------------------------------
0074      01 [                  C   DISK_STATUS1    DB      1 DUP(?)
                ??              C
                    ]           C
                                C
0075      01 [                  C   HF_NUM          DB      1 DUP(?)
                ??              C
                    ]           C
                                C
0076      01 [                  C   CONTROL_BYTE    DB      1 DUP(?)
                ??              C
                    ]           C
                                C
0077      01 [                  C   PORT_OFF        DB      1 DUP(?)
                ??              C
                    ]           C
                                C
                                C
                                C   ;-------------------------------------------------------------
                                C   ;        PRINTER AND RS232 TIME-OUT VARIABLES               :
                                C   ;-------------------------------------------------------------
0078      04 [                  C   PRINT_TIM_OUT   DB      4 DUP(?)
                ??              C
                    ]           C
                                C
007C      04 [                  C   RS232_TIM_OUT   DB      4 DUP(?)
                ??              C
                    ]           C
                                C
                                C
                                C   ;----------------------------------------
                                C   ;        ADDITIONAL KEYBOARD DATA AREA  :
                                C   ;----------------------------------------
0080      01 [                  C   BUFFER_START    DW      1 DUP(?)
                ????            C
                        ]       C
                                C
0082      01 [                  C   BUFFER_END      DW      1 DUP(?)
                ????            C
                        ]       C
                                C
                                C
                                C   ;----------------------------------------
                                C   ;        ADDITIONAL FLOPPY DATA         ;
                                C   ;----------------------------------------
008B                            C           ORG     8BH
008B      01 [                  C   LASTRATE        DB      1 DUP(?)        ; LAST DATA RATE SELECTED
                ??              C
                    ]           C
                                C
                                C   PAGE
                                C   ;----------------------------------------
                                C   ;        ADDITIONAL HARD FILE DATA      :
                                C   ;----------------------------------------
008C                            C           ORG     8CH
008C      01 [                  C   HF_STATUS       DB      1 DUP(?)        ; STATUS REGISTER
                ??              C
                    ]           C
                                C
008D      01 [                  C   HF_ERROR        DB      1 DUP(?)        ; ERROR REGISTER
                ??              C
                    ]           C
                                C
008E      01 [                  C   HF_INT_FLAG     DB      1 DUP(?)        ; HARD FILE INTERRUPT FLAG
                ??              C
                    ]           C
                                C
008F      01 [                  C   HF_CNTRL        DB      1 DUP(?)        ; COMBO HARD FILE/FLOPPY CARD BIT 0=1
                ??              C
                    ]           C
                                C
                                C
                                C   ;----------------------------------------
                                C   ;        ADDITIONAL DISKETTE AREA       :
                                C   ;----------------------------------------
0090                            C           ORG     90H
0090                            C   DSK_STATE       LABEL   BYTE
0090      01 [                  C                   DB      1 DUP(?)        ; DRIVE 0 MEDIA STATE
                ??              C
                    ]           C
                                C
0091      01 [                  C                   DB      1 DUP(?)        ; DRIVE 1 MEDIA STATE
                ??              C
                    ]           C
                                C
0092      01 [                  C                   DB      1 DUP(?)        ; DRIVE 0 OPERATION START STATE
```

SECTION 5

```
                ??              C
                     ]          C
                                C
0093     01 [                   C                  DB       1 DUP(?)         ; DRIVE 1 OPERATION START STATE
                ??              C
                     ]          C
                                C
0094     01 [                   C  DSK_TRK         DB       1 DUP(?)         ; DRIVE 0 PRESENT CYLINDER
                ??              C
                     ]          C
                                C
0095     01 [                   C                  DB       1 DUP(?)         ; DRIVE 1 PRESENT CYLINDER
                ??              C
                     ]          C
                                C
0096     01 [                   C                  DB       1 DUP(?)         ; RESERVED
                ??              C
                     ]          C
                                C
                                C  ;----------------------------------------
                                C  ;       ADDITIONAL KEYBOARD LED FLAG     :
                                C  ;----------------------------------------
0097                            C          ORG     97H
0097     01 [                   C  KB_FLAG_2       DB       1 DUP(?)
                ??              C
                     ]          C
                                C
                                C  PAGE
                                C  ;----------------------------------------
                                C  ;       REAL TIME CLOCK DATA AREA        :
                                C  ;----------------------------------------
0098                            C          ORG     98H
0098     01 [                   C  USER_FLAG       DW       1 DUP(?)         ; OFFSET ADDR OF USERS WAIT FLAG
                ????            C
                       ]        C
                                C
009A     01 [                   C  USER_FLAG_SEG   DW       1 DUP(?)         ; SEG ADDR OF USER WAIT FLAG
                ????            C
                       ]        C
                                C
009C     01 [                   C  RTC_LOW         DW       1 DUP(?)         ; LOW WORD OF USER WAIT FLAG
                ????            C
                       ]        C
                                C
009E     01 [                   C  RTC_HIGH        DW       1 DUP(?)         ; HIGH WORD OF USER WAIT FLAG
                ????            C
                       ]        C
                                C
00A0     01 [                   C  RTC_WAIT_FLAG   DB       1 DUP(?)         ; WAIT ACTIVE FLAG
                ??              C
                     ]          C
                                C
00A1                            C  DATA    ENDS
                                C  ;----------------------------------------
                                C  ;       EXTRA DATA AREA                  :
                                C  ;----------------------------------------
0000                            C  XXDATA  SEGMENT AT 50H
0000     01 [                   C  STATUS_BYTE     DB       1 DUP(?)
                ??              C
                     ]          C
                                C
0001                            C  XXDATA  ENDS
                                C  ;----------------------------------------
                                C  ;       VIDEO DISPLAY BUFFER             :
                                C  ;----------------------------------------
0000                            C  VIDEO_RAM       SEGMENT AT 0B800H
0000                            C  REGEN   LABEL   BYTE
0000                            C  REGENW  LABEL   WORD
0000  4000 [                    C          DB      16384 DUP(?)
                ??              C
                     ]          C
                                C
4000                            C  VIDEO_RAM       ENDS
                                C
                                   .LIST
                                C  INCLUDE SEGMENT.SRC
0000                            C  CODE SEGMENT BYTE PUBLIC
                                C
                                   EXTRN   VIDEO_PARMS:BYTE
                                   EXTRN   POST2:NEAR
                                   EXTRN   DDS:NEAR
                                   EXTRN   D11:NEAR
                                   EXTRN   VECTOR_TABLE:NEAR
                                   EXTRN   KBD_RESET:NEAR
                                   EXTRN   DUMMY_RETURN:NEAR
                                   EXTRN   STGTST_CNT:NEAR
                                   EXTRN   ERR_BEEP:NEAR
                                   EXTRN   ROM_CHECK:NEAR
                                   EXTRN   ROS_CHECKSUM:NEAR
                                   EXTRN   SYSINIT1:NEAR
                                   EXTRN   SHUT2:NEAR
                                   EXTRN   SHUT3:NEAR
                                   EXTRN   SHUT4:NEAR
                                   EXTRN   SHUT6:NEAR
                                   EXTRN   SHUT7:NEAR
                                   EXTRN   SHUT9:NEAR
                                   EXTRN   PROC_SHUTDOWN:NEAR
                                   EXTRN   C1:NEAR
                                   EXTRN   C2:NEAR
                                   EXTRN   C8042A:NEAR
                                   EXTRN   OBF_42A:NEAR
                                   EXTRN   C8042B:NEAR
                                   EXTRN   C8042C:NEAR
                                   EXTRN   OBF_42B:NEAR
                                   EXTRN   F3B:NEAR
                                   EXTRN   SLAVE_VECTOR_TABLE:NEAR
                                   EXTRN   NMI_INT:NEAR
                                   EXTRN   PRINT_SCREEN:NEAR
                                   EXTRN   GATE_A20:NEAR

                                           ASSUME  CS:CODE,SS:CODE,ES:ABS0,DS:DATA

                                   PUBLIC  POST1
                                   PUBLIC  BEGIN
                                   PUBLIC  CHK_VIDEO
                                   PUBLIC  START_1
                                   PUBLIC  C8042
                                   PUBLIC  OBF_42
                                   PUBLIC  C11
                                   PUBLIC  C30
                                   PUBLIC  TST4_B
                                   PUBLIC  TST4_C
                                   PUBLIC  TST4_D
                                   PUBLIC  E30B
                                   PUBLIC  E30C
```

```
 = 0000                             BEGIN   EQU     $

                                    ;        6 1 8 1 0 2 8   C O P R .   I B M   1 9 8 4  ;EVEN
                                    ;         6 1 8 1 0 2 9   C O P R .   I B M   1 9 8 4  ;ODD
 0000  36 36 31 31 38 38            DB      '66118811002289  CCOOPPRR..  IIBBMM  11998844' ;COPYRIGHT NOTICE
       31 31 30 30 32 32
       38 39 20 20 43 43
       4F 4F 50 50 52 52
       2E 2E 20 20 49 49
       42 42 4D 4D 20 20
       31 31 39 39 38 38
       34 34

                                    ;--------------------------------------------
                                    ;       INITIAL RELIABILITY TESTS -- PHASE 1    :
                                    ;--------------------------------------------
 002C                               POST1   PROC    NEAR
                                    ;-----------------------------------------------------------------
                                    ;       LOAD A BLOCK OF TEST CODE THROUGH THE KEYBOARD PORT       :
                                    ;       FOR MANUFACTUING TEST.                                     :
                                    ;       THIS ROUTINE WILL LOAD A TEST (MAX LENGTH=FAFFH) THROUGH  :
                                    ;       THE KEYBOARD PORT. CODE WILL BE LOADED AT LOCATION        :
                                    ;       0000:0500. AFTER LOADING, CONTROL WILL BE TRANSFERED      :
                                    ;       TO LOCATION 0000:0500. STACK WILL BE LOCATED AT 30:100    :
                                    ;       THIS ROUTINE ASSUMES THAT THE FIRST 2                      :
                                    ;       BYTES TRANSFERED CONTAIN THE COUNT OF BYTES TO BE LOADED  :
                                    ;       (BYTE 1=COUNT LOW, BYTE 2=COUNT HI.)                       :
                                    ;-----------------------------------------------------------------
                                    ;
 002C                               MFG_BOOT:
 002C  FA                                   CLI                             ; NO INTERRUPTS

                                    ;------- DEGATE ADDRESS LINE 20

 002D  B4 DD                                MOV     AH,DISABLE_BIT20        ; DEGATE COMMAND
 002F  E8 0000 E                            CALL    GATE_A20                ; ISSUE THE COMMAND

                                    ;------- SETUP HARDWARE INT VECTOR TABLE LVL 0-7

 0032  2B C0                                SUB     AX,AX                   ;
 0034  8E C0                                MOV     ES,AX
 0036  B9 0008                              MOV     CX,08                   ; GET VECTOR CNT
 0039  0E                                   PUSH    CS                      ; SETUP DS SEG REG
 003A  1F                                   POP     DS
 003B  BE 0000 E                            MOV     SI,OFFSET VECTOR_TABLE
 003E  BF 0020 R                            MOV     DI,OFFSET INT_PTR
 0041  A5                           MFG_B:  MOVSW
 0042  47                                   INC     DI                      ; SKIP OVER SEGMENT
 0043  47                                   INC     DI
 0044  E2 FB                                LOOP    MFG_B

                                    ;------- SETUP HARDWARE INT VECTOR TABLE LVL 8-15 (VECTORS START AT INT 70H)

 0046  2B C0                                SUB     AX,AX                   ;
 0048  8E C0                                MOV     ES,AX
 004A  B9 0008                              MOV     CX,08                   ; GET VECTOR CNT
 004D  0E                                   PUSH    CS                      ; SETUP DS SEG REG
 004E  1F                                   POP     DS
 004F  BE 0000 E                            MOV     SI,OFFSET SLAVE_VECTOR_TABLE
 0052  BF 01C0 R                            MOV     DI,OFFSET SLAVE_INT_PTR
 0055  A5                           MFG_C:  MOVSW
 0056  47                                   INC     DI                      ; SKIP OVER SEGMENT
 0057  47                                   INC     DI
 0058  E2 FB                                LOOP    MFG_C                   ;

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

                                            ASSUME  DS:ABS0
                                            ASSUME  ES:ABS0
 005A  2B C0                                SUB     AX,AX                           ; DS=0
 005C  8E D8                                MOV     DS,AX
 005E  8E C0                                MOV     ES,AX                           ; ES=0
 0060  C7 06 0008 R 0000 E                  MOV     NMI_PTR,OFFSET NMI_INT          ; NMI INTERRUPT
 0066  C7 06 0014 R 0000 E                  MOV     INT5_PTR,OFFSET PRINT_SCREEN    ; PRINT SCREEN
 006C  C7 06 0062 R F600                    MOV     BASIC_PTR+2,0F600H              ; SEGMENT FOR CASSETTE BASIC

                                    ;------- ENABLE KEYBOARD PORT

 0072  B0 60                                MOV     AL,60H                  ; WRITE 8042 RAM 0
 0074  E8 0405 R                            CALL    C8042                   ; ISSUE THE COMMAND
 0077  B0 09                                MOV     AL,00001001B            ; SET INHIBIT OVERIDE/ENABLE OBF INT
 0079  E6 60                                OUT     PORT_A,AL               ;   AND NOT PC COMP

 007B  E8 009D R                            CALL    MFG_2                   ; GET COUNT LOW
 007E  8A F8                                MOV     BH,AL                   ; SAVE IT
 0080  E8 009D R                            CALL    MFG_2                   ; GET COUNT HI
 0083  8A E8                                MOV     CH,AL                   ;
 0085  8A CF                                MOV     CL,BH                   ; CX NOW HAS COUNT
 0087  FC                                   CLD                             ; SET DIR. FLAG TO INCRIMENT
 0088  BF 0500                              MOV     DI,0500H                ; SET TARGET OFFSET (DS=0000)
 008B                               MFG_1:
 008B  E4 64                                IN      AL,STATUS_PORT          ; GET 8042 STATUS PORT
 008D  A8 01                                TEST    AL,OUT_BUF_FULL         ; KB REQUEST PENDING?
 008F  74 FA                                JZ      MFG_1                   ; LOOP TILL DATA PRESENT
 0091  E4 60                                IN      AL,PORT_A               ; GET DATA
 0093  AA                                   STOSB                           ; STORE IT

 0094  E6 80                                OUT     MFG_PORT,AL             ; DISPLAY CHAR AT MFG PORT

 0096  E2 F3                                LOOP    MFG_1                   ; LOOP TILL ALL BYTES READ

 0098  EA 0500 ---- R                       JMP     MFG_TEST_RTN            ; FAR JUMP TO CODE THAT WAS JUST
                                                                            ; LOADED
 009D  E4 64                        MFG_2:  IN      AL,STATUS_PORT          ; CHECK FOR OUTPUT BUFF FULL
 009F  A8 01                                TEST    AL,OUT_BUF_FULL         ;   HANG HERE IF NO DATA AVAILABLE
 00A1  E1 FA                                LOOPZ   MFG_2                   ;

 00A3  E4 60                                IN      AL,PORT_A               ; GET THE COUNT
 00A5  C3                                   RET                             ;

                                    ;--------------------------------------
                                    ; TEST.01                               :
                                    ;       X286 PROCESSOR TEST (REAL MODE) :
                                    ; DESCRIPTION                           :
                                    ;       VERIFY FLAGS, REGISTERS         :
                                    ;       AND CONDITIONAL JUMPS           :
                                    ;--------------------------------------
                                            ASSUME  CS:CODE,DS:DATA,ES:NOTHING,SS:NOTHING
 00A6  FA                           START_1: CLI                            ; DISABLE INTERRUPTS
 00A7  B4 D5                                MOV     AH,0D5H                 ; SET SF, CF, ZF, AND AF FLAGS ON
 00A9  9E                                   SAHF
 00AA  73 2A                                JNC     ERR02                   ; GO TO ERR ROUTINE IF CF NOT SET
 00AC  75 28                                JNZ     ERR02                   ; GO TO ERR ROUTINE IF ZF NOT SET
 00AE  7B 26                                JNP     ERR02                   ; GO TO ERR ROUTINE IF PF NOT SET
 00B0  79 24                                JNS     ERR02                   ; GO TO ERR ROUTINE IF SF NOT SET
 00B2  9F                                   LAHF                            ; LOAD FLAG IMAGE TO AH
```

SECTION 5

```
00B3  B1 05                      MOV    CL,5                    ; LOAD CNT REG WITH SHIFT CNT
00B5  D2 EC                      SHR    AH,CL                   ; SHIFT AF INTO CARRY BIT POS
00B7  73 1D                      JNC    ERR02                   ; GO TO ERR ROUTINE IF AF NOT SET
00B9  B0 40                      MOV    AL,40H                  ; SET THE OF FLAG ON
00BB  D0 E0                      SHL    AL,1                    ; SETUP FOR TESTING
00BD  71 17                      JNO    ERR02                   ; GO TO ERR ROUTINE IF OF NOT SET
00BF  32 E4                      XOR    AH,AH                   ; SET AH = 0
00C1  9E                         SAHF                           ; CLEAR SF, CF, ZF, AND PF
00C2  76 12                      JBE    ERR02                   ; GO TO ERR ROUTINE IF CF ON
                                                                ; GO TO ERR ROUTINE IF ZF ON
00C4  78 10                      JS     ERR02                   ; GO TO ERR ROUTINE IF SF ON
00C6  7A 0E                      JP     ERR02                   ; GO TO ERR ROUTINE IF PF ON
00C8  9F                         LAHF                           ; LOAD FLAG IMAGE TO AH
00C9  B1 05                      MOV    CL,5                    ; LOAD CNT REG WITH SHIFT CNT
00CB  D2 EC                      SHR    AH,CL                   ; SHIFT `AF' INTO CARRY BIT POS
00CD  72 07                      JC     ERR02                   ; GO TO ERR ROUTINE IF ON
00CF  D0 E4                      SHL    AH,1                    ; CHECK THAT `OF' IS CLEAR
00D1  70 03                      JO     ERR02                   ; GO TO ERR ROUTINE IF ON
00D3  EB 04 90                   JMP    C7A                     ; CONTINUE
00D6  E9 01AC R         ERR02:   JMP    ERR01                   ; ERROR EXIT

00D9                    C7A:
00D9  B8 ---- R                  MOV    AX,DATA                 ; SET DATA SEGMENT
00DC  8E D8                      MOV    DS,AX

                        ;----- CHECK FOR PROCESSOR SHUTDOWN

00DE  E4 64                      IN     AL,STATUS_PORT          ; CHECK FOR SHUTDOWN
00E0  A8 04                      TEST   AL,SYS_FLAG             ;
00E2  75 03                      JNZ    C7B                     ; GO IF YES
00E4  E9 0181 R                  JMP    C7                      ;

                        ;------ CHECK FOR SHUTDOWN 9
00E7                    C7B:
00E7  B0 8F                      MOV    AL,SHUT_DOWN            ; CMOS ADDR FOR SHUTDOWN BYTE
00E9  E6 70                      OUT    CMOS_PORT,AL
00EB  EB 00                      JMP    SHORT $+2               ; IO DELAY
00ED  E4 71                      IN     AL,CMOS_PORT+1          ; GET WHO
00EF  86 C4                      XCHG   AL,AH                   ; SAVE THE SHUTDOWN REQUEST
00F1  80 FC 09                   CMP    AH,09H                  ; WAS IT SHUTDOWN REQUEST 9?
00F4  74 3C                      JZ     C7C                     ; BYPASS INIT OF INT CHIPS
                        ;----------------------------------------------------------
                        ;    RE-INITIALIZE THE 8259 INTERRUPT #1 CONTROLLER CHIP :
                        ;----------------------------------------------------------

00F6  2A C0                      SUB    AL,AL                   ; INSURE MATH PROCESSOR RESET
00F8  E6 F1                      OUT    X287+1,AL               ;
00FA  B0 11                      MOV    AL,11H                  ; ICW1 - EDGE, MASTER, ICW4
00FC  E6 20                      OUT    INTA00,AL
00FE  EB 00                      JMP    SHORT $+2               ; WAIT STATE FOR IO
0100  B0 08                      MOV    AL,8                    ; SETUP ICW2 - INT TYPE 8 (8-F)
0102  E6 21                      OUT    INTA01,AL
0104  EB 00                      JMP    SHORT $+2               ; WAIT STATE FOR IO

0106  B0 04                      MOV    AL,04H                  ; SETUP ICW3 - MASTER LV 2
0108  E6 21                      OUT    INTA01,AL               ;
010A  EB 00                      JMP    SHORT $+2               ; IO WAIT STATE
010C  B0 01                      MOV    AL,01H                  ; SETUP ICW4 - MASTER,8086 MODE
010E  E6 21                      OUT    INTA01,AL               ;
0110  EB 00                      JMP    SHORT $+2               ; WAIT STATE FOR IO
0112  B0 FF                      MOV    AL,0FFH                 ; MASK ALL INTS. OFF
0114  E6 21                      OUT    INTA01,AL               ; (VIDEO ROUTINE ENABLES INTS.)
                        ;----------------------------------------------------------
                        ;    RE-INITIALIZE THE 8259 INTERRUPT #2 CONTROLLER CHIP  :
                        ;----------------------------------------------------------

0116  B0 11                      MOV    AL,11H                  ; ICW1 - EDGE, SLAVE ICW4
0118  E6 A0                      OUT    INTB00,AL
011A  EB 00                      JMP    SHORT $+2               ; WAIT STATE FOR IO
011C  B0 70                      MOV    AL,INT_TYPE             ; SETUP ICW2 - INT TYPE 50 (50-5F)
011E  E6 A1                      OUT    INTB01,AL
0120  B0 02                      MOV    AL,02H                  ; SETUP ICW3 - SLAVE LV 2
0122  EB 00                      JMP    SHORT $+2               ;
0124  E6 A1                      OUT    INTB01,AL               ;
0126  EB 00                      JMP    SHORT $+2               ; IO WAIT STATE
0128  B0 01                      MOV    AL,01H                  ; SETUP ICW4 - 8086 MODE, SLAVE
012A  E6 A1                      OUT    INTB01,AL               ;
012C  EB 00                      JMP    SHORT $+2               ; WAIT STATE FOR IO
012E  B0 FF                      MOV    AL,0FFH                 ; MASK ALL INTS. OFF
0130  E6 A1                      OUT    INTB01,AL               ;
                        ;----------------------------------------------------------------
                        ;  SHUTDOWN                                                      :
                        ;       RETURN CONTROL AFTER A SHUTDOWN COMMAND IS ISSUED        :
                        ;  DESCRIPTION                                                   :
                        ;       A TEST IS MADE FOR THE SYSTEM FLAG BEING SET.  IF        :
                        ;       THE SYSTEM FLAG IS SET, THE SHUTDOWN BYTE IN CMOS        :
                        ;       IS USED TO DETERMINE WHERE CONTROL IS RETURNED.          :
                        ;                                                                :
                        ;       CMOS = 0     SOFT RESET OR UNEXPECTED SHUTDOWN           :
                        ;       CMOS = 1     SHUT DOWN AFTER MEMORY SIZE                 :
                        ;       CMOS = 2     SHUT DOWN AFTER MEMORY TEST                 :
                        ;       CMOS = 3     SHUT DOWN WITH MEMORY ERROR                 :
                        ;       CMOS = 4     SHUT DOWN WITH BOOT LOADER REQUEST          :
                        ;       CMOS = 5     JMP DWORD REQUEST (WITH INT INIT)           :
                        ;       CMOS = 6     PROTECTED MODE TEST7 PASSED                 :
                        ;       CMOS = 7     PROTECTED MODE TEST7 FAILED                 :
                        ;       CMOS = 8     PROTECTED MODE TEST1 FAILED                 :
                        ;       CMOS = 9     BLOCK MOVE SHUTDOWN REQUEST                 :
                        ;       CMOS = A     JMP DWORD REQUEST (W/O INT INIT)            :
                        ;----------------------------------------------------------------
                        ;----- CHECK FROM WHERE

0132  B0 8F             C7C:     MOV    AL,SHUT_DOWN            ; CLEAR CMOS BYTE
0134  E6 70                      OUT    CMOS_PORT,AL            ;
0136  EB 00                      JMP    SHORT $+2               ; IO DELAY
0138  2A C0                      SUB    AL,AL                   ; SET BYTE TO 0
013A  E6 71                      OUT    CMOS_PORT+1,AL          ;
013C  86 E0                      XCHG   AH,AL                   ;
013E  3C 0A                      CMP    AL,0AH                  ; MAX TABLE ENTRYS
0140  77 2C                      JA     SHUT0                   ; GO IF GREATER THAN MAX
0142  BE 0158 R                  MOV    SI,OFFSET BRANCH      ; GET THE START OF BRANCH TABLE
0145  03 F0                      ADD    SI,AX                   ;
0147  03 F0                      ADD    SI,AX                   ; POINT TO BRANCH ADDRESS
0149  2E: 8B 1C                  MOV    BX,CS:[SI]              ; GET BRANCH TO BX
014C  FA                         CLI
014D  B8 ---- R                  MOV    AX,STACK                ; SET STACK
0150  8E D0                      MOV    SS,AX                   ;
0152  BC 0100 R                  MOV    SP,OFFSET TOS           ;
0155  FB                         STI                            ;
0156  FF E3                      JMP    BX                      ; JUMP BACK

0158  016E R            BRANCH:  DW     SHUT0                   ; NORMAL POWER UP/UNEXPECTED SHUTDOWN
015A  09B0 R                     DW     SHUT1                   ; SHUT DOWN AFTER MEMORY SIZE
015C  0000 E                     DW     SHUT2                   ; SHUT DOWN AFTER MEMORY TEST
```

```
015E  0000 E                         DW      SHUT3                 ; SHUT DOWN WITH MEMORY ERROR
0160  0000 E                         DW      SHUT4                 ; SHUT DOWN WITH BOOT_LOADER REQUEST
0162  0171 R                         DW      SHUT5                 ; JMP DWORD REQUEST (WITH INTERRUPT INIT)
0164  0000 E                         DW      SHUT6                 ; PROTECTED MODE TEST7 PASSED
0166  0000 E                         DW      SHUT7                 ; PROTECTED MODE TEST7 FAILED
0168  07F7 R                         DW      SHUT8                 ; PROTECTED MODE TEST1 FAILED
016A  0000 E                         DW      SHUT9                 ; BLOCK MOVE SHUTDOWN REQUEST
016C  017D R                         DW      SHUTA                 ; JMP DWORD REQUEST (W/O INTERRUPT INIT)
016E  EB 11 90               SHUT0:  JMP     C7

                             ;------- IO_ROM_INIT MUST BE INITIALIZED BY THE USER
                                                                   ; FLUSH THE KEYBOARD BUFFER
0171  E4 64                  SHUT5:  IN      AL,STATUS_PORT        ; CHECK IF OUTPUT BUFFER FULL
0173  A8 01                          TEST    AL,OUT_BUF_FULL
0175  74 02                          JZ      SHUT5B                ; GO IF NOT
0177  E4 60                          IN      AL,PORT_A             ; FLUSH
0179  B0 20                  SHUT5B: MOV     AL,EOI                ; FLUSH LAST TIMER TICK
017B  E6 20                          OUT     INTA00,AL             ; -TO ALLOW TIMER INTERRUPTS

017D  FF 2E 0067 R           SHUTA:  JMP     DWORD PTR DS:IO_ROM_INIT;

                             ;----- CHECKPOINT 1

0181  B0 01                  C7:     MOV     AL,01H                ; <><><><><><><><><><><>
0183  E6 80                          OUT     MFG_PORT,AL           ; <><><>CHECKPOINT 1<><><>

                             ;----- READ/WRITE THE X286 GENERAL AND SEGMENTATION REGISTERS
                             ;       WITH ALL ONE'S AND ZEROES'S.

0185  B8 FFFF                        MOV     AX,0FFFFH             ; SETUP ONE'S PATTERN IN AX
0188  F9                             STC                           ; SET CARRY FLAG
0189  73 21                          JNC     ERR01                 ; GO IF NO CARRY
018B  8E D8                  C8:     MOV     DS,AX                 ; WRITE PATTERN TO ALL REGS
018D  8C DB                          MOV     BX,DS
018F  8E C3                          MOV     ES,BX
0191  8C C1                          MOV     CX,ES
0193  8E D1                          MOV     SS,CX
0195  8C D2                          MOV     DX,SS
0197  8B E2                          MOV     SP,DX
0199  8B EC                          MOV     BP,SP
019B  8B F5                          MOV     SI,BP
019D  8B FE                          MOV     DI,SI
019F  73 07                          JNC     C9                    ;
01A1  33 C7                          XOR     AX,DI                 ; PATTERN MAKE IT THRU ALL REGS
01A3  75 07                          JNZ     ERR01                 ; NO - GO TO ERR ROUTINE
01A5  F8                             CLC                           ; CLEAR CARRY FLAG
01A6  EB E3                          JMP     C8
01A8                         C9:                                   ; TST1A
01A8  0B C7                          OR      AX,DI                 ; ZERO PATTERN MAKE IT THRU?
01AA  74 01                          JZ      C10A                  ; YES - GO TO NEXT TEST
01AC  F4                     ERR01:  HLT                           ; HALT SYSTEM
                             ;------- INSURE THAT CMOS CLOCK INTERRUPTS ARE DISABLED

01AD  B0 8B                  C10A:   MOV     AL,CMOS_ALARM         ;
01AF  E6 70                          OUT     CMOS_PORT,AL          ;
01B1  EB 00                          JMP     SHORT $+2             ;
01B3  E4 71                          IN      AL,CMOS_PORT+1        ; GET THE CURRENT CONTROL REG
01B5  86 C4                          XCHG    AL,AH                 ; SAVE IT
01B7  80 E4 07                       AND     AH,07H                ; CLEAR SET,PIE,AIE, AND SQWE BITS
01BA  B0 8B                          MOV     AL,CMOS_ALARM         ;
01BC  E6 70                          OUT     CMOS_PORT,AL          ;
01BE  86 C4                          XCHG    AL,AH                 ;
01C0  EB 00                          JMP     SHORT $+2             ; IO DELAY
01C2  E6 71                          OUT     CMOS_PORT+1,AL        ;

01C4  EB 00                          JMP     SHORT $+2             ; IO DELAY
01C6  B0 8C                          MOV     AL,CMOS_ALARM+1       ; CLEAR PENDING INTERRUPT
01C8  E6 70                          OUT     CMOS_PORT,AL          ;
01CA  EB 00                          JMP     SHORT $+2             ; IO DELAY
01CC  E4 71                          IN      AL,CMOS_PORT+1        ;

                             ;-------- RESET VIDEO

                                     ASSUME  DS:DATA
01CE  B8 ---- R                      MOV     AX,DATA
01D1  8E D8                          MOV     DS,AX                 ; SET DATA SEGMENT
01D3  81 3E 0072 R 1234              CMP     RESET_FLAG,1234H      ; SOFT RESET?
01D9  74 0B                          JZ      SFT_RST               ; GO IF YES

01DB  2A C0                          SUB     AL,AL                 ;
01DD  BA 03D8                        MOV     DX,3D8H
01E0  EE                             OUT     DX,AL                 ; DISABLE COLOR VIDEO
01E1  FE C0                          INC     AL
01E3  B2 B8                          MOV     DL,0B8H
01E5  EE                             OUT     DX,AL                 ; DISABLE B/W VIDEO,EN HIGH RES
01E6  B0 FC                  SFT_RST:MOV     AL,11111100B          ; DISABLE PARITY CHECKERS
01E8  E6 61                          OUT     PORT_B,AL

                             ;----------------------------------------
                             ; TEST.02                               :
                             ;       VERIFY CMOS SHUTDOWN BYTE       :
                             ; DESCRIPTION                           :
                             ;       ROLLING BIT WRITTEN AND VERIFIED :
                             ;       AT SHUTDOWN ADDRESS             :
                             ;----------------------------------------

                             ;----- VERIFY AND CLEAR SHUTDOWN FLAG

01EA  B0 02                          MOV     AL,2                  ;<><><><><><><><><><><>
01EC  E6 80                          OUT     MFG_PORT,AL           ;<><><>CHECKPOINT 2<><>

01EE  B9 0009                        MOV     CX,09H                ; LOOP COUNT
01F1  B4 01                          MOV     AH,1                  ; START WITH BIT 0
01F3  B0 8F                  C10B:   MOV     AL,SHUT_DOWN          ;
01F5  E6 70                          OUT     CMOS_PORT,AL          ;
01F7  8A C4                          MOV     AL,AH                 ; OUTPUT ROLLING BIT
01F9  EB 00                          JMP     SHORT $+2             ; IO DELAY
01FB  E6 71                          OUT     CMOS_PORT+1,AL        ;
01FD  B0 8F                          MOV     AL,SHUT_DOWN          ; READ CMOS
01FF  EB 00                          JMP     SHORT $+2             ; IO DELAY
0201  E6 70                          OUT     CMOS_PORT,AL          ;
0203  EB 00                          JMP     SHORT $+2             ; IO DELAY
0205  E4 71                          IN      AL,CMOS_PORT+1        ;
0207  3A C4                          CMP     AL,AH                 ; MUST BE THE SAME
0209  75 A1                          JNZ     ERR01                 ; ERROR IF NOT
020B  D0 D4                          RCL     AH,1                  ; ROLL A BIT THRU SHUT DOWN
020D  E2 E4                          LOOP    C10B                  ; LOOP TILL DONE

                             ;----------------------------------------
                             ; TEST.03                               :
                             ;       ROS CHECKSUM TEST I             :
                             ; DESCRIPTION                           :
                             ;       A CHECKSUM IS DONE FOR THE 32K  :
                             ;       ROS MODULES CONTAINING POD AND  :
```

```
                              ;       BIOS.                                :
                              ;---------------------------------------------
020F                          C10:
                              ;----- CHECKPOINT 3

020F  B0 03                          MOV    AL,03H                    ;<><><><><><><><><><><><><><>
0211  E6 80                          OUT    MFG_PORT,AL               ;<><><>CHECKPOINT 3<><><>
                                                                      ;
0213  8C C8                          MOV    AX,CS                     ; SETUP SS SEG REG
0215  8E D0                          MOV    SS,AX
0217  8E D8                          MOV    DS,AX                     ; SET UP DATA SEG TO POINT TO  ========
                                                                      ; ROM ADDRESS
                                     ASSUME SS:CODE
0219  BB 0000 R                      MOV    BX,OFFSET BEGIN           ; SETUP STARTING ROS ADDR
021C  BC 0000 E                      MOV    SP,OFFSET C1              ; SETUP RETURN ADDRESS
021F  E9 0000 E                      JMP    ROS_CHECKSUM
0222                          C11:
0222  74 01                          JZ     C11A                      ; HALT SYSTEM IF ERROR
0224  F4                             HLT
                              ;---------------------------------------------
                              ; TEST.04                                      :
                              ;       8253 CHECK TIMER 1 ALL BITS ON         ;
                              ; DESCRIPTION                                  ;
                              ;       SET TIMER COUNT                        :
                              ;       CHECK THAT TIMER 1 ALL BITS ON         :
                              ;----------------------------------------------:
                                     ASSUME DS:DATA
0225  B8 ---- R               C11A:  MOV    AX,DATA                   ; SET DATA SEGMENT
0228  8E D8                          MOV    DS,AX                     ;
022A  B0 04                          MOV    AL,04H                    ;<><><><><><><><><><><><><><><><><><>
022C  E6 80                          OUT    MFG_PORT,AL               ;<><><><>CHECKPOINT 4<><><><><>

                              ;----- DISABLE DMA CONTROLLER

                              ;      MOV    AL,04                     ; AL ALREADY = 04H
022E  E6 08                          OUT    DMA08,AL                  ; DISABLE DMA CONTROLLER 1
0230  E6 D0                          OUT    DMA18,AL                  ; DISABLE DMA CONTROLLER 2

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

0232  8B 16 0072 R                   MOV    DX,RESET_FLAG             ; SAVE RESET FLAG WHILE REFRESH IS OFF
0236  B0 54                          MOV    AL,54H                    ; SEL TIMER 1,LSB,MODE 2
0238  E6 43                          OUT    TIMER+3,AL
023A  EB 00                          JMP    SHORT $+2                 ; WAIT STATE FOR IO
023C  8A C1                          MOV    AL,CL                     ; SET INITIAL TIMER CNT TO 0
023E  E6 41                          OUT    TIMER+1,AL
0240  B7 05                          MOV    BH,05H                    ; LOOP COUNT
0242                          C12:                                    ; TIMER1_BITS_ON
0242  B0 40                          MOV    AL,40H                    ; LATCH TIMER 1 COUNT
0244  EB 00                          JMP    SHORT $+2                 ; IO DELAY
0246  E6 43                          OUT    TIMER+3,AL
0248  80 FB FF                       CMP    BL,0FFH                   ; YES - SEE IF ALL BITS GO OFF
024B  74 0B                          JE     C13                       ; TIMER1_BITS_OFF
024D  E4 41                          IN     AL,TIMER+1                ; READ TIMER 1 COUNT
024F  0A D8                          OR     BL,AL                     ; ALL BITS ON IN TIMER
0251  E2 EF                          LOOP   C12                       ; TIMER1_BITS_ON
0253  FE CF                          DEC    BH
0255  75 EB                          JNZ    C12                       ;  TRY AGAIN
0257  F4                             HLT                              ; TIMER 1 FAILURE, HALT SYS
                                                                      ; TIMER1_BITS_OFF
                              ;-----------------------------------------------------------
                              ; TEST.05                                                   :
                              ;       8253 CHECK TIMER 1 ALL BIT OFF                      :
                              ; DESCRIPTION                                               :
                              ;       SET TIMER COUNT                                     :
                              ;       CHECK THAT TIMER 1 ALL BITS OFF                     :
                              ;-----------------------------------------------------------

                              ;------ CHECKPOINT 5

0258  B0 05                   C13:   MOV    AL,05H                    ; <><><><><><><><><><><><><><>
025A  E6 80                          OUT    MFG_PORT,AL               ; <><><>CHECKPOINT 5<><><>

025C  8A C3                          MOV    AL,BL                     ; SET TIMER 1 CNT
025E  2B C9                          SUB    CX,CX
0260  E6 41                          OUT    TIMER+1,AL
0262  B7 05                          MOV    BH,05H                    ; SET TRY AGAIN COUNT
0264                          C14:                                    ; TIMER_LOOP
0264  EB 00                          JMP    SHORT $+2                 ;  IO DELAY
0266  B0 40                          MOV    AL,40H                    ; LATCH TIMER 1 COUNT
0268  E6 43                          OUT    TIMER+3,AL
026A  EB 00                          JMP    SHORT $+2                 ; DELAY FOR TIMER
026C  EB 00                          JMP    SHORT $+2                 ; ADDED DELAY FOR TIMER
026E  E4 41                          IN     AL,TIMER+1                ; READ TIMER 1 COUNT
0270  22 D8                          AND    BL,AL
0272  74 07                          JZ     C15                       ; WRAP_DMA_REG
0274  E2 EE                          LOOP   C14                       ; TIMER_LOOP
0276  FE CF                          DEC    BH                        ;
0278  75 EA                          JNZ    C14                       ;
027A  F4                             HLT                              ; HALT SYSTEM
                              ;-----------------------------------------------------------
                              ; TEST.06                                                   :
                              ;       8237 DMA 0 INITIALIZATION CHANNEL REGISTER TEST     :
                              ; DESCRIPTION                                               :
                              ;       DISABLE THE 8237 DMA CONTROLLER.                    :
                              ;       WRITE/READ THE CURRENT                              :
                              ;       ADDRESS AND WORD COUNT REGISTERS FOR ALL            :
                              ;       CHANNELS.                                           :
                              ;-----------------------------------------   --------------

                              ;----- CHECKPOINT 6

027B                          C15:
027B  B8 ---- R                      MOV    AX,DATA                   ; SET DATA SEGMENT
027E  8E D8                          MOV    DS,AX
0280  B0 06                          MOV    AL,06H                    ;<><><><><><><><><><><><><><>
0282  E6 80                          OUT    MFG_PORT,AL               ;<><><>CHECKPOINT 6<><><>
0284  89 16 0072 R                   MOV    RESET_FLAG,DX             ; RESTORE SOFT RESET FLAG
0288  E6 0D                          OUT    DMA+0DH,AL                ; SEND MASTER CLEAR TO DMA
                              ;----- WRAP DMA 0 CHANNEL ADDRESS AND COUNT REGISTERS

028A  B0 FF                          MOV    AL,0FFH                   ; WRITE PATTERN FF TO ALL REGS
028C  8A D8                   C16:   MOV    BL,AL                     ; SAVE PATTERN FOR COMPARE
028E  8A F8                          MOV    BH,AL
0290  B9 0008                        MOV    CX,8                      ; SETUP LOOP CNT
0293  BA 0000                        MOV    DX,DMA                    ; SETUP I/O PORT ADDR OF REG
0296  EE                      C17:   OUT    DX,AL                     ; WRITE PATTERN TO REG, LSB
0297  EB 00                          JMP    SHORT $+2                 ; WAIT STATE FOR IO
0299  EE                             OUT    DX,AL                     ; MSB OF 16 BIT REG
029A  B0 01                          MOV    AL,01H                    ; AL TO ANOTHER PAT BEFORE RD
029C  EB 00                          JMP    SHORT $+2                 ; WAIT STATE FOR IO
029E  EC                             IN     AL,DX                     ; READ 16-BIT DMA CH REG, LSB  2ST DMA
029F  EB 00                          JMP    SHORT $+2                 ; WAIT STATE FOR IO
02A1  8A E0                          MOV    AH,AL                     ; SAVE LSB OF 16-BIT REG
02A3  EC                             IN     AL,DX                     ; READ MSB OF DMA CH REG
```

```
02A4  3B D8                    CMP     BX,AX              ; PATTERN READ AS WRITTEN?
02A6  74 01                    JE      C18                ; YES - CHECK NEXT REG
02A8  F4                       HLT                        ; NO - HALT THE SYSTEM
02A9                   C18:                               ; NXT_DMA_CH
02A9  42                       INC     DX                 ; SET IO PORT TO NEXT CH REG
02AA  E2 EA                    LOOP    C17                ; WRITE PATTERN TO NEXT REG
02AC  FE C0                    INC     AL                 ; SET PATTERN TO 0
02AE  74 DC                    JZ      C16                ; YES CONTINUE

                       ;------- WRITE DMA WITH 55 PATTERN

02B0  80 FB 55                 CMP     BL,55H             ; CHECK IF 55 PATTERN DONE
02B3  74 09                    JZ      C19                ; GO IF YES
02B5  80 FB AA                 CMP     BL,0AAH            ; CHECK IF AA PATTERN DONE
02B8  74 08                    JZ      C20                ; GO IF YES
02BA  B0 55                    MOV     AL,55H             ;
02BC  EB CE                    JMP     C16

                       ;------- WRITE DMA WITH AA PATTERN

02BE  B0 AA            C19:    MOV     AL,0AAH            ;
02C0  EB CA                    JMP     C16                ;
                       ;--------------------------------------------------
                       ; TEST.07                                          :
                       ;       8237 DMA 1 INITIALIZATION CHANNEL REGISTER TEST :
                       ; DESCRIPTION                                      :
                       ;       DISABLE THE 8237 DMA CONTROLLER 1.         :
                       ;       WRITE/READ THE CURRENT DMA 1               :
                       ;       ADDRESS AND WORD COUNT REGISTERS FOR ALL   :
                       ;       CHANNELS.                                  :
                       ;--------------------------------------------------

                       ;------- CHECKPOINT 7 DMA 1

02C2  B0 07            C20:    MOV     AL,07H             ; <><><><><><><><><><><><>
02C4  E6 80                    OUT     MFG_PORT,AL        ; <><><>CHECKPOINT 7<><><>
02C6  E6 DA                    OUT     DMA1+0DH*2,AL      ; SEND MASTER CLEAR TO 2ND DMA

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

02C8  B0 FF                    MOV     AL,0FFH            ; WRITE PATTERN FF TO ALL REGS
02CA  8A D8            C16A:   MOV     BL,AL              ; SAVE PATTERN FOR COMPARE
02CC  8A F8                    MOV     BH,AL
02CE  B9 0008                  MOV     CX,8               ; SETUP LOOP CNT
02D1  BA 00C0                  MOV     DX,DMA1            ; SETUP I/O PORT ADDR OF REG
02D4  EE               C17A:   OUT     DX,AL              ; WRITE PATTERN TO REG, LSB
02D5  EB 00                    JMP     SHORT $+2          ; WAIT STATE FOR IO
02D7  EE                       OUT     DX,AL              ; MSB OF 16 BIT REG
02D8  B0 01                    MOV     AL,01H             ; AL TO ANOTHER PAT BEFORE RD
02DA  EB 00                    JMP     SHORT $+2          ; WAIT STATE FOR IO
02DC  EC                       IN      AL,DX              ; READ 16-BIT DMA CH REG, LSB  2ST DMA
02DD  EB 00                    JMP     SHORT $+2          ; WAIT STATE FOR IO
02DF  8A E0                    MOV     AH,AL              ; SAVE LSB OF 16-BIT REG
02E1  EC                       IN      AL,DX              ; READ MSB OF DMA CH REG
02E2  3B D8                    CMP     BX,AX              ; PATTERN READ AS WRITTEN?
02E4  74 01                    JE      C18A               ; YES - CHECK NEXT REG
02E6  F4                       HLT                        ; NO - HALT THE SYSTEM
02E7                   C18A:                              ; NXT_DMA_CH
02E7  83 C2 02                 ADD     DX,2               ; SET IO PORT TO NEXT CH REG
02EA  E2 E8                    LOOP    C17A               ; WRITE PATTERN TO NEXT REG
02EC  FE C0                    INC     AL                 ; SET PATTERN TO 0
02EE  74 DA                    JZ      C16A               ; YES CONTINUE
                       ;------- WRITE DMA WITH 55 PATTERN

02F0  80 FB 55                 CMP     BL,55H             ; CHECK IF 55 PATTERN DONE
02F3  74 09                    JZ      C20A               ; GO IF YES
02F5  80 FB AA                 CMP     BL,0AAH            ; CHECK IF AA PATTERN DONE
02F8  74 08                    JZ      C21                ; GO IF YES
02FA  B0 55                    MOV     AL,55H             ;
02FC  EB CC                    JMP     C16A

                       ;------- WRITE DMA WITH AA PATTERN

02FE  B0 AA            C20A:   MOV     AL,0AAH            ;
0300  EB C8                    JMP     C16A               ;

                       ;----- INITIALIZE AND START MEMORY REFRESH.

0302  8B 1E 0072 R     C21:    MOV     BX,RESET_FLAG      ; GET THE RESET FLAG

0306  A3 0010 R                MOV     EQUIP_FLAG,AX      ; DO A DUMMY WRITE RAM BEFORE REFRESH
0309  B0 12                    MOV     AL,18              ; START TIMER
030B  E6 41                    OUT     TIMER+1,AL

                       ;------- SET DMA COMMAND
030D                   C21Z:
030D  2A C0                    SUB     AL,AL              ; DACK SENSE LOW,DREQ SENSE HIGH
030F  E6 08                    OUT     DMA+8,AL           ; LATE WRITE,FIXED PRIORITY,NORMAL TIMING
                                                          ; CONTROLLER ENABLE,CH0 ADDR HOLD DISABLE
                                                          ; MEMORY TO MEM DISABLE
0311  E6 D0                    OUT     DMA18,AL           ; SAME TO SECOND CONTROLLER

                       ;-------- MODE SET ALL DMA CHANNELS

0313  B0 40                    MOV     AL,40H             ; SET MODE FOR CHANNEL 0
0315  E6 0B                    OUT     DMA+0BH,AL         ;
0317  B0 C0                    MOV     AL,0C0H            ; SET CASCADE MODE ON CHANNEL 4
0319  E6 D6                    OUT     DMA18+06H,AL
031B  EB 00                    JMP     SHORT $+2          ; WAIT STATE FOR IO
031D  B0 41                    MOV     AL,41H             ; SET MODE FOR CHANNEL 1
031F  E6 0B                    OUT     DMA+0BH,AL
0321  E6 D6                    OUT     DMA18+06H,AL       ; SET MODE FOR CHANNEL 5
0323  EB 00                    JMP     SHORT $+2          ; WAIT STATE FOR IO
0325  B0 42                    MOV     AL,42H             ; SET MODE FOR CHANNEL 2
0327  E6 0B                    OUT     DMA+0BH,AL
0329  E6 D6                    OUT     DMA18+06H,AL       ; SET MODE FOR CHANNEL 6
032B  EB 00                    JMP     SHORT $+2          ; WAIT STATE FOR IO
032D  B0 43                    MOV     AL,43H             ; SET MODE FOR CHANNEL 3
032F  E6 0B                    OUT     DMA+0BH,AL
0331  E6 D6                    OUT     DMA18+06H,AL       ; SET MODE FOR CHANNEL 7

                       ;------- RESTORE RESET FLAG

0333  89 1E 0072 R             MOV     RESET_FLAG,BX      ;

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

                       ;----- CHECK POINT 8

0337  B0 08                    MOV     AL,08H             ; <><><><><><><><><><><><>
```

SECTION 5

```
0339  E6 80                 OUT    MFG_PORT,AL           ; <><><>CHECKPOINT 8<><><>
033B  2A C0                 SUB    AL,AL                 ;
033D  BA 0081               MOV    DX,DMA_PAGE           ;
0340  B9 00FF               MOV    CX,0FFH               ; DO ALL DATA PATTERNS
0343  EE           C22A:    OUT    DX,AL                 ;
0344  42                    INC    DX                    ;
0345  FE C0                 INC    AL                    ;
0347  81 FA 008F            CMP    DX,8FH                ; TEST DMA PAGES 81 THUR 8EH
034B  75 F6                 JNZ    C22A                  ;
034D  86 E0                 XCHG   AH,AL                 ; SAVE CURRENT DATA PATTERN
034F  FE CC                 DEC    AH                    ; CHECK LAST WRITTEN
0351  4A                    DEC    DX                    ;
0352  2A C0        C22B:    SUB    AL,AL                 ; CHANGE DATA BEFORE READ
0354  EC                    IN     AL,DX                 ;
0355  3A C4                 CMP    AL,AH                 ; DATA AS WRITTEN?
0357  75 30                 JNZ    C26                   ; GO ERROR HALT IF NOT
0359  FE CC                 DEC    AH                    ;
035B  4A                    DEC    DX                    ;
035C  81 FA 0080            CMP    DX,MFG_PORT           ; CONTINUE TILL PORT 80
0360  75 F0                 JNZ    C22B                  ;
0362  FE C4                 INC    AH                    ; NEXT PATTERN TO RIPPLE
0364  8A C4                 MOV    AL,AH                 ;
0366  E2 DB                 LOOP   C22A                  ;

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

0368  B0 CC                 MOV    AL,0CCH               ; WRITE AN CC TO PAGE REGISTERS
036A  BA 008F      C22:     MOV    DX,LAST_DMA_PAGE      ;
036D  8A E0                 MOV    AH,AL                 ; SAVE THE DATA PATTERN
036F  EE           C23:     OUT    DX,AL                 ; OUTPUT PAGE REG
                   ;-------- VERIFY PAGE REGISTER 8F

0370  2A C0        C24:     SUB    AL,AL                 ; CHANGE DATA PATTERN BEFORE READ
0372  EC                    IN     AL,DX                 ; GET THE DATA FROM PAGE REG
0373  3A C4                 CMP    AL,AH                 ;
0375  75 12                 JNZ    C26                   ; GO IF ERROR
0377  80 FC CC              CMP    AH,0CCH               ;
037A  75 04                 JNZ    C25                   ; GO IF ERROR
037C  B0 33                 MOV    AL,033H               ; SET UP DATA PATTERN OF 33
037E  EB EA                 JMP    C22                   ; DO DATA 33
0380  80 FC 00     C25:     CMP    AH,0                  ; CHECK DONE
0383  74 05                 JZ     C27                   ; GO IF YES
0385  2A C0                 SUB    AL,AL                 ; SET UP FOR DATA PATTERN 00
0387  EB E1                 JMP    C22                   ; DO DATA 0

                   ;------- ERROR HALT

0389  F4           C26:     HLT                          ; HALT SYSTEM
                   ;-------------------------------------------------------
                   ; TEST.09                                               :
                   ;        STORAGE REFRESH TEST                           :
                   ; DESCRIPTION                                           :
                   ;        VERIFY STORAGE REFRESH IS OCCURRING            :
                   ;-------------------------------------------------------

                   ;------- CHECKPOINT 9 TEST MEMORY REFRESH

038A  B0 09        C27:     MOV    AL,09H                ; <><><><><><><><><><><><>
038C  E6 80                 OUT    MFG_PORT,AL           ; <><><>CHECKPOINT 9<><><>

038E  2B C9                 SUB    CX,CX                 ;
0390  E4 61        C28:     IN     AL,PORT_B             ; INSURE REFRESH BIT IS TOGGLING
0392  A8 10                 TEST   AL,REFRESH_BIT        ;
0394  E1 FA                 LOOPZ  C28                   ; INSURE REFRESH IS OFF
0396  74 F1                 JZ     C26                   ; GO IF NOT
0398  2B C9                 SUB    CX,CX                 ;
039A  E4 61        C29:     IN     AL,PORT_B             ;
039C  A8 10                 TEST   AL,REFRESH_BIT        ; INSURE REFRESH IS ON
039E  E0 FA                 LOOPNZ C29                   ;
03A0  75 E7                 JNZ    C26                   ; GO IF NO REFRESH
                   ;---------------------------------------------
                   ; TEST.10                                     :
                   ;        8042 TEST AND CONFIGURATION JUMPERS  :
                   ; DESCRIPTION                                 :
                   ;        ISSUE A SELF TEST TO THE 8042        :
                   ;        INSURE A 55H IS RECEIVED             :
                   ;        GET MANUFACTURING/DISPLAY TYPE JUMPER:
                   ;         INPUT PORT INFO SAVED IN MFG_TEST   :
                   ;---------------------------------------------

                   ;------- CHECKPOINT 0A

03A2  B0 0A                 MOV    AL,0AH                ; <><><><><><><><><><><><>
03A4  E6 80                 OUT    MFG_PORT,AL           ; <><>CHECPOINT 0A<><><>

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

03A6  2B C9                 SUB    CX,CX                 ; 100 MSEC FOR THIS LOOP
03A8  E4 64        TST1:    IN     AL,STATUS_PORT        ; CHECK FOR INPUT BUFFER FULL
03AA  8A E0                 MOV    AH,AL                 ;
03AC  F6 C4 01              TEST   AH,OUT_BUF_FULL       ;
03AF  74 02                 JZ     TST2                  ; GO IF NOT
03B1  E4 60                 IN     AL,PORT_A             ; FLUSH
03B3  F6 C4 02     TST2:    TEST   AH,INPT_BUF_FULL      ; IS THE OUTPUT BUFFER ALSO FULL?
03B6  E0 F0                 LOOPNZ TST1                  ; TRY AGAIN
03B8  74 01                 JZ     TST4                  ; CONTINUE IF OK

03BA  F4           ERR0:    HLT                          ; HALT SYSTEM IF BUFFER FULL

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

03BB  B0 0B        TST4:    MOV    AL,0BH                ; <><><><><><><><><><><><>
03BD  E6 80                 OUT    MFG_PORT,AL           ; <><><>CHECKPOINT 0B <><>

03BF  B0 AA                 MOV    AL,0AAH               ; SELF TEST COMMAND
03C1  BC 0000 E             MOV    SP,OFFSET C8042A      ; SET RETURN ADDR
03C4  EB 3F 90              JMP    C8042                 ;
03C7  A8 01        TST4_B:  TEST   AL,OUT_BUF_FULL       ; IS THE OUTPUT BUFFER FULL?
03C9  74 02                 JZ     TST4_A                ; GO IF NOT
03CB  E4 60                 IN     AL,PORT_A             ; FLUSH
03CD  BC 0000 E    TST4_A:  MOV    SP,OFFSET OBF_42A     ; SET RETURN ADDR
03D0  EB 3F 90              JMP    OBF_42                ; GO WAIT FOR BUFFER
03D3  E4 60        TST4_C:  IN     AL,PORT_A             ; GET THE ENDING RESPONSE
03D5  3C 55                 CMP    AL,55H                ;

03D7  B0 0C                 MOV    AL,0CH                ; <><><><><><><><><><><><>
03D9  E6 80                 OUT    MFG_PORT,AL           ; <><><>CHECKPOINT 0C <><>

03DB  75 DD                 JNZ    ERR0                  ; GO IF NOT OK

                   ;------- GET THE SWITCH SETTINGS

03DD  B0 C0                 MOV    AL,0C0H               ; READ INPUT COMMAND
03DF  BC 0000 E             MOV    SP,OFFSET C8042C      ; SET RETURN ADDRESS
03E2  EB 21 90              JMP    C8042                 ; ISSUE COMMAND
03E5  BC 0000 E    E30B:    MOV    SP,OFFSET OBF_42B     ; SET RETURN ADDRESS
```

```
03E8  EB 27 90                       JMP     OBF_42                  ; GO WAIT FOR RESPONSE
03EB  E4 60                  E30C:   IN      AL,PORT_A               ; GET THE SWITCH
03ED  E6 82                          OUT     DMA_PAGE+1,AL           ; SAVE TEMP

                             ;------ WRITE BYTE 0 OF 8042 RAM

03EF  B0 60                          MOV     AL,60H                  ; WRITE BYTE COMMAND
03F1  BC 0000 E                      MOV     SP,OFFSET C8042B        ; SET RETURN ADDR
03F4  EB 0F 90                       JMP     C8042                   ; ISSUE THE COMMAND
03F7  74 05                  TST4_D: JZ      TST4_D1                 ; CONTINUE IF COMMAND ACCEPTED

03F9  B0 0D                          MOV     AL,0DH                  ; <><><><><><><><><><><>
03FB  E6 80                          OUT     MFG_PORT,AL             ; <><><>CHECKPOINT 0D <><>
03FD  F4                             HLT
03FE  B0 5D                  TST4_D1:MOV     AL,5DH                  ; ENABLE OUTPUT BUFF FULL INT - DISABLE KEYBOARD
0400  E6 60                          OUT     PORT_A,AL               ; SET SYS FLAG - PC 1 COMP - INH OVERRIDE
0402  EB 1E 90                       JMP     E30A                    ; CONTINUE

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

0405  FA                     C8042:  CLI                             ; NO INTERRUPTS ALLOWED
0406  E6 64                          OUT     STATUS_PORT,AL          ; SEND COMMAND IN AL REG

0408  2B C9                          SUB     CX,CX                   ; LOOP COUNT
040A  E4 64                  C42_1:  IN      AL,STATUS_PORT          ; WAIT FOR THE COMMAND ACCEPTED
040C  A8 02                          TEST    AL,INPT_BUF_FULL        ;
040E  E0 FA                          LOOPNZ  C42_1
0410  C3                             RET                             ;

                             ;------- WAIT FOR 8042 RESPONSE

0411  2B C9                  OBF_42: SUB     CX,CX                   ;
0413  B3 06                          MOV     BL,6                    ; 200MS/PER LOOP * 6  =1200 MS +
0415  E4 64                  C42_2:  IN      AL,STATUS_PORT          ; CHECK FOR RESPONSE
0417  A8 01                          TEST    AL,OUT_BUF_FULL         ;
0419  75 06                          JNZ     C42_3                   ; GO IF RESPONSE
041B  E2 F8                          LOOP    C42_2                   ; TRY AGAIN
041D  FE CB                          DEC     BL                      ; DECREMENT LOOP COUNT
041F  75 F4                          JNZ     C42_2                   ;
0421  C3                     C42_3:  RET                             ; RETURN TO CALLER
                             ;-----------------------------------------------
                             ; TEST.11                                       :
                             ;       BASE 64K READ/WRITE STORAGE TEST        :
                             ; DESCRIPTION                                   :
                             ;       WRITE/READ/VERIFY DATA PATTERNS         :
                             ;       AA,55,FF,01, AND 00 TO 1ST 64K OF       :
                             ;       STORAGE. VERIFY STORAGE ADDRESSABILITY. :
                             ;-----------------------------------------------

                             ;----- FILL MEMORY WITH DATA

0422  B0 0E                  E30A:   MOV     AL,0EH                  ;SET CHECKPOINT (E)
0424  E6 80                          OUT     MFG_PORT,AL             ;<><><><><><><><><><><>

0426  B8 ---- R                      MOV     AX,DATA                 ; GET THE SYSTEM SEGMENT
0429  8E D8                          MOV     DS,AX                   ;  OF DATA
042B  8B 1E 0072 R                   MOV     BX,RESET_FLAG           ; SAVE RESET_FLAG IN BX
042F  FC                             CLD                             ; SET DIR FLAG TO INC.
0430  B9 8000                        MOV     CX,2000H*4              ; SET FOR 32K WORDS
0433  2B FF                          SUB     DI,DI                   ; FIRST 16K
0435  2B F6                          SUB     SI,SI                   ;
0437  2B C0                          SUB     AX,AX                   ;
0439  8E D8                          MOV     DS,AX                   ;
043B  8E C0                          MOV     ES,AX                   ;
043D  81 FB 1234                     CMP     BX,1234H                ; WARM START?
0441  75 03                          JNZ     E30A_0                  ; GO IF NOT
0443  E9 05E6 R                      JMP     CLR_STG

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

0446  B0 0F                  E30A_0: MOV     AL,0FH                  ; <><><><><><><><><><><>
0448  E6 80                          OUT     MFG_PORT,AL             ; <><><>CHECKPOINT F<><><>

044A  B0 80                          MOV     AL,PRTY_CHK             ; SET BASE RAM PARITY
044C  E6 87                          OUT     DMA_PAGE+6,AL           ; USE AS TEMP SAVE
044E  BC 0000 E                      MOV     SP,OFFSET C2            ; SET RETURN ADDRESS
0451  E9 0000 E                      JMP     STGTST_CNT
0454  8B D8                  C30:    MOV     BX,AX                   ; SAVE FAILING BIT PATTERN
0456  75 03                          JNZ     C31
0458  E9 05F1 R                      JMP     C33                     ; STORAGE OK, CONTINUE
                             ;-------------------------------------------
                             ; BASE 64K STORAGE FAILURE
                             ;   DISPLAY THE CHECKPOINT (MFG CHECKPOINT)
                             ;     AND XOR EXPECTED WITH READ IN MFG_PORT
                             ;   DISPLAY CHECKPOINT IN MFG_PORT+3
                             ;   DISPLAY XOR'D DATA HIGH BYTE MFG_PORT+1
                             ;     LOW BYTE IN MFG_PORT+2
                             ;   A READ/WRITE SCOPE LOOP OF THE FIRST
                             ;   WORD FOR POSSIBLE ADDRESS LINE FAILURES
                             ;-------------------------------------------

045B                         C31:
045B  8A C7                          MOV     AL,BH                   ; SAVE HIGH BYTE
045D  E6 81                          OUT     MFG_PORT+1,AL           ;
045F  8A C3                          MOV     AL,BL                   ; SAVE LOW BYTE
0461  E6 82                          OUT     MFG_PORT+2,AL

                             ;------- CHECK FOR VIDEO ROM

0463  B9 C000                        MOV     CX,0C000H               ; START OF IO ROM
0466  8E D9                  M1:     MOV     DS,CX                   ;
0468  2B DB                          SUB     BX,BX                   ; GET THE FIRST 2 LOCATIONS
046A  8B 07                          MOV     AX,[BX]                 ;
046C  EB 00                          JMP     SHORT $+2               ; BUS SETTLE
046E  3D AA55                        CMP     AX,0AA55H               ; IS THE VIDEO ROM PRESENT?
0471  74 0C                          JZ      Z5                      ; GO IF YES
0473  81 C1 0080                     ADD     CX,080H                 ; POINT TO NEXT 2K BLOCK
0477  81 F9 C800                     CMP     CX,0C800H               ; TOP OF VIDEO ROM AREA YET?
047B  7C E9                          JL      M1                      ; TRY AGAIN
047D  23 C9                          AND     CX,CX                   ; SET NON ZERO FLAG

047F  75 03                  Z5:     JNZ     C32                     ; GO IF NOT
0481  E9 0573 R                      JMP     C31_0                   ; BYPASS ERROR DISPLAY IF VIDEO ROM
                             ;-----------------------------------------------
                             ; SET VIDEO MODE TO DISPLAY MEMORY ERROR
                             ;       THIS ROUTINE INITIALIZES THE ATTACHMENT TO
                             ;       TO DISPLAY FIRST 64K STORAGE ERRORS.
                             ; BOTH COLOR AND MONO ATTACHMENTS ARE INITIALIZED.
                             ;-----------------------------------------------
= 0010                       M4      EQU     10H
                             ;------- INIT COLOR/MONO

0484  BA 03D8                C32:    MOV     DX,3D8H                 ; CONTROL REG ADDRESS OF COLOR CARD
0487  2A C0                          SUB     AL,AL                   ; MODE SET
0489  EE                             OUT     DX,AL                   ;
```

SECTION 5

```
048A  BA 03B8                 MOV   DX,03B8H                ; CONTROL REG ADDRESS OF BW CARD
048D  B0 01                   MOV   AL,1                    ; MODE SET FOR CARD
048F  EE                      OUT   DX,AL                   ; RESET VIDEO
0490  83 EA 04                SUB   DX,4                    ; BACK TO BASE REGISTER

0493  BB 0030 E               MOV   BX,OFFSET VIDEO_PARMS+M4*3  ; POINT TO VIDEO PARMS
                              ASSUME DS:CODE
0496  B9 0010           Z_2:  MOV   CX,M4                   ; COUNT OF MONO VIDEO PARMS

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

0499  32 E4                   XOR   AH,AH           ; AH WILL SERVE AS REGISTER NUMBER DURING LOOP

                        ;----- LOOP THROUGH TABLE, OUTPUTTTING REG ADDRESS, THEN VALUE FROM TABLE

049B  8A C4             M10:  MOV   AL,AH                   ; GET 6845 REGISTER NUMBER
049D  EE                      OUT   DX,AL
049E  42                      INC   DX                      ; POINT TO DATA PORT
049F  FE C4                   INC   AH                      ; NEXT REGISTER VALUE
04A1  2E: 8A 07               MOV   AL,CS:[BX]              ; GET TABLE VALUE
04A4  EE                      OUT   DX,AL                   ; OUT TO CHIP
04A5  43                      INC   BX                      ; NEXT IN TABLE
04A6  4A                      DEC   DX                      ; BACK TO POINTER REGISTER
04A7  E2 F2                   LOOP  M10                     ; DO THE WHOLE TABLE
04A9  8A E2                   MOV   AH,DL                   ; CHECK IF COLOR CARD DONE
04AB  80 E4 F0                AND   AH,0F0H                 ; STRIP UNWANTED BITS
04AE  80 FC D0                CMP   AH,0D0H                 ; IS IT THE COLOR CARD?
04B1  74 08                   JZ    Z_3                     ; CONTINUE IF COLOR
04B3  BB 0000 E               MOV   BX,OFFSET VIDEO_PARMS   ; POINT TO VIDEO PARMS
04B6  BA 03D4                 MOV   DX,3D4H                 ; COLOR BASE
04B9  EB DB                   JMP   Z_2                     ; CONTINUE

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

04BB  33 FF             Z_3:  XOR   DI,DI                   ; SET UP POINTER FOR REGEN
04BD  B8 B000                 MOV   AX,0B000H               ; SET UP ES TO VIDEO REGEN
04C0  8E C0                   MOV   ES,AX                   ;

04C2  B9 0800                 MOV   CX,2048                 ; NUMBER OF WORDS IN MONO CARD
04C5  B8 0720                 MOV   AX,' '+7*256            ; FILL CHAR FOR ALPHA
04C8  F3/ AB                  REP   STOSW                   ; FILL THE REGEN BUFFER WITH BLANKS

04CA  33 FF                   XOR   DI,DI                   ; CLEAR COLOR VIDEO RAM
04CC  BB B800                 MOV   BX,0B800H               ; SET UP ES TO COLOR VIDEO RAM
04CF  8E C3                   MOV   ES,BX                   ;
04D1  B9 2000                 MOV   CX,8192                 ;
04D4  F3/ AB                  REP   STOSW                   ; FILL WITH BLANKS

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

04D6  BA 03B8                 MOV   DX,3B8H
04D9  B0 29                   MOV   AL,29H                  ;
04DB  EE                      OUT   DX,AL                   ; SET VIDEO ENABLE PORT

                        ;----- SET UP OVERSCAN REGISTER

04DC  42                      INC   DX                      ; SET OVERSCAN PORT TO A DEFAULT
04DD  B0 30                   MOV   AL,30H                  ; VALUE OF 30H FOR ALL MODES EXCEPT 640X200
04DF  EE                      OUT   DX,AL                   ; OUTPUT THE CORRECT VALUE TO 3D9 PORT

                        ;----- ENABLE COLOR VIDEO AND CORRECT PORT SETTING
04E0  BA 03D8                 MOV   DX,3D8H
04E3  B0 28                   MOV   AL,28H                  ;
04E5  EE                      OUT   DX,AL                   ; SET VIDEO ENABLE PORT

                        ;----- SET UP OVERSCAN REGISTER

04E6  42                      INC   DX                      ; SET OVERSCAN PORT TO A DEFAULT
04E7  B0 30                   MOV   AL,30H                  ; VALUE OF 30H FOR ALL MODES EXCEPT 640X200
04E9  EE                      OUT   DX,AL                   ; OUTPUT THE CORRECT VALUE TO 3D9 PORT

                        ;------- DISPLAY FAILING CHECKPOINT AND

04EA  8C C8                   MOV   AX,CS                   ; SET STACK SEGMENT TO CODE SEGMENT
04EC  8E D0                   MOV   SS,AX                   ;

04EE  BB B000                 MOV   BX,0B000H               ;
04F1  8E DB                   MOV   DS,BX                   ; SET DS TO BW CRT BUFFER

04F3  B0 30             Z_0:  MOV   AL,'0'                  ; DISPLAY BANK 000000
04F5  B9 0006                 MOV   CX,6                    ;
04F8  2B FF                   SUB   DI,DI                   ; START AT 0
04FA  88 05             Z:    MOV   DS:[DI],AL              ; WRITE TO CRT BUFFER
04FC  47                      INC   DI                      ; POINT TO NEXT POSTITON
04FD  47                      INC   DI                      ;
04FE  E2 FA                   LOOP  Z

0500  80 FF B8                CMP   BH,0B8H                 ; CHECK THAT COLOR BUFFER WRITTEN
0503  74 0C                   JZ    Z_1
0505  2B FF                   SUB   DI,DI                   ; POINT TO START OF BUFFER

0507  B7 B0                   MOV   BH,0B0H                 ;
0509  8E C3                   MOV   ES,BX                   ; ES = MONO
050B  B7 B8                   MOV   BH,0B8H                 ; SET SEGMENT TO COLOR
050D  8E DB                   MOV   DS,BX                   ; DS = COLOR
050F  EB E2                   JMP   Z_0                     ;

                        ;-------- PRINT FAILING BIT PATTERN

0511  B0 20             Z_1:  MOV   AL,' '                  ; DISPLAY A BLANK
0513  88 05                   MOV   DS:[DI],AL              ; WRITE TO COLOR BUFFER
0515  26: 88 05               MOV   ES:[DI],AL              ; WRITE TO MONO BUFFER
0518  47                      INC   DI                      ; POINT TO NEXT POSTITON
0519  47                      INC   DI                      ;
051A  E4 81                   IN    AL,MFG_PORT+1           ; GET THE HIGH BYTE OF FALING PATTERN
051C  B1 04                   MOV   CL,4                    ; SHIFT COUNT
051E  D2 E8                   SHR   AL,CL                   ; NIBBLE SWAP
0520  BC 05DE R               MOV   SP,OFFSET Z1_0
0523  EB 1E 90                JMP   PR

0526  E4 81             Z1:   IN    AL,MFG_PORT+1           ;
0528  24 0F                   AND   AL,0FH                  ; ISOLATE TO LOW NIBBLE
052A  BC 05E0 R               MOV   SP,OFFSET Z2_0
052D  EB 14 90                JMP   PR              ;
0530  E4 82             Z2:   IN    AL,MFG_PORT+2           ; GET THE HIGH BYTE OF FALING PATTERN
0532  B1 04                   MOV   CL,4                    ; SHIFT COUNT
0534  D2 E8                   SHR   AL,CL                   ; NIBBLE SWAP
0536  BC 05E2 R               MOV   SP,OFFSET Z3_0
0539  EB 08 90                JMP   PR
053C  E4 82             Z3:   IN    AL,MFG_PORT+2           ;
053E  24 0F                   AND   AL,0FH                  ; ISOLATE TO LOW NIBBLE
0540  BC 05E4 R               MOV   SP,OFFSET Z4_0          ; RETURN TO Z4:

                        ;------- CONVERT AND PRINT
```

```
                                                    ;       CONVERT 00-0F TO ASCII CHARACTER
0543  04 90             PR:     ADD     AL,090H     ; ADD FIRST CONVERSION FACTOR
0545  27                        DAA                 ; ADJUST FOR NUMERIC AND ALPHA RANGE
0546  14 40                     ADC     AL,040H     ; ADD CONVERSION AND ADJUST LOW NIBBLE
0548  27                        DAA                 ; ADJUST HIGH NIBBLE TO ASCHI RANGE

0549  88 05                     MOV     DS:[DI],AL  ; WRITE TO COLOR BUFFER
054B  26: 88 05                 MOV     ES:[DI],AL  ; WRITE TO MONO BUFFER
054E  47                        INC     DI          ; POINT TO NEXT POSTITON
054F  47                        INC     DI          ;
0550  C3                        RET

                        ;------- DISPLAY 201 ERROR

0551  B0 20             Z4:     MOV     AL,' '      ; DISPLAY A BLANK
0553  88 05                     MOV     DS:[DI],AL  ; WRITE TO CRT BUFFER
0555  26: 88 05                 MOV     ES:[DI],AL  ; WRITE TO MONO BUFFER
0558  47                        INC     DI          ; POINT TO NEXT POSTITON
0559  47                        INC     DI          ;
055A  B0 32                     MOV     AL,'2'      ; DISPLAY 201 ERROR
055C  88 05                     MOV     DS:[DI],AL  ; WRITE TO CRT BUFFER
055E  26: 88 05                 MOV     ES:[DI],AL  ; WRITE TO MONO BUFFER
0561  47                        INC     DI          ; POINT TO NEXT POSTITON
0562  47                        INC     DI          ;
0563  B0 30                     MOV     AL,'0'      ;
0565  88 05                     MOV     DS:[DI],AL  ; WRITE TO CRT BUFFER
0567  26: 88 05                 MOV     ES:[DI],AL  ; WRITE TO MONO BUFFER
056A  47                        INC     DI          ; POINT TO NEXT POSTITON
056B  47                        INC     DI          ;
056C  B0 31                     MOV     AL,'1'      ;
056E  88 05                     MOV     DS:[DI],AL  ; WRITE TO CRT BUFFER
0570  26: 88 05                 MOV     ES:[DI],AL  ; WRITE TO MONO BUFFER

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

0573  B0 DD             C31_0:  MOV     AL,0DDH     ; <><><><><><><><><><><><>
0575  E6 80                     OUT     MFG_PORT,AL ; <><><>CHECKPOINT DD <><><>
0577  E6 83                     OUT     MFG_PORT+3,AL ; ALSO DISPLAY CHECK POINT IN PORT 83
0579  2B C9                     SUB     CX,CX       ;
057B                    C31_A:

057B  2B C0                     SUB     AX,AX       ; SETUP SEGMENT
057D  8E D8                     MOV     DS,AX       ;
057F  B8 AA55                   MOV     AX,0AA55H   ; WRITE AN AA55
0582  2B FF                     SUB     DI,DI       ;
0584  89 05                     MOV     DS:[DI],AX  ;
0586  8B 05                     MOV     AX,DS:[DI]  ; READ THE FIRST WORD
0588  E2 F1                     LOOP    C31_A       ; DISPLAY CHKPT LONGER
058A                    C31_B:
058A  89 05                     MOV     DS:[DI],AX  ;
058C  8B 05                     MOV     AX,DS:[DI]  ;
058E  E2 FA                     LOOP    C31_B
0590                    C31_C:
0590  89 05                     MOV     DS:[DI],AX  ;
0592  8B 05                     MOV     AX,DS:[DI]  ;
0594  E2 FA                     LOOP    C31_C       ;
0596                    C31_D:
0596  89 05                     MOV     DS:[DI],AX  ;
0598  8B 05                     MOV     AX,DS:[DI]  ;
059A  E2 FA                     LOOP    C31_D       ;
059C                    C31_E:
059C  89 05                     MOV     DS:[DI],AX  ;
059E  8B 05                     MOV     AX,DS:[DI]  ;
05A0  E2 FA                     LOOP    C31_E

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

05A2  E4 81                     IN      AL,MFG_PORT+1 ; XOR OF FAILING BIT PATTERN
05A4  E6 80                     OUT     MFG_PORT,AL ;  HIGH BYTE
05A6                    C31_G:
05A6  B8 AA55                   MOV     AX,0AA55H   ; WRITE AN AA55
05A9  89 05                     MOV     DS:[DI],AX  ;
05AB  8B 05                     MOV     AX,DS:[DI]  ; READ THE FIRST WORD
05AD  E2 F7                     LOOP    C31_G       ;
05AF                    C31_H:
05AF  89 05                     MOV     DS:[DI],AX  ;
05B1  8B 05                     MOV     AX,DS:[DI]  ;
05B3  E2 FA                     LOOP    C31_H       ;
05B5                    C31_I:
05B5  89 05                     MOV     DS:[DI],AX  ;
05B7  8B 05                     MOV     AX,DS:[DI]  ;
05B9  E2 FA                     LOOP    C31_I

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

05BB  E4 82                     IN      AL,MFG_PORT+2 ;  LOW BYTE
05BD  E6 80                     OUT     MFG_PORT,AL ;
05BF  B8 AA55                   MOV     AX,0AA55H   ; WRITE AN AA55
05C2  2B FF             C31_K:  SUB     DI,DI       ;
05C4  89 05                     MOV     DS:[DI],AX  ;
05C6  8B 05                     MOV     AX,DS:[DI]  ; READ THE FIRST WORD
05C8  E2 F8                     LOOP    C31_K       ;
05CA                    C31_L:
05CA  89 05                     MOV     DS:[DI],AX  ;
05CC  8B 05                     MOV     AX,DS:[DI]  ;
05CE  E2 FA                     LOOP    C31_L       ;
05D0                    C31_M:
05D0  89 05                     MOV     DS:[DI],AX  ;
05D2  8B 05                     MOV     AX,DS:[DI]  ;
05D4  E2 FA                     LOOP    C31_M       ;
05D6                    C31_N:
05D6  89 05                     MOV     DS:[DI],AX  ;
05D8  8B 05                     MOV     AX,DS:[DI]  ;
05DA  E2 FA                     LOOP    C31_N       ;
05DC  EB 95                     JMP     C31_0       ; DO AGAIN

05DE  0526 R            Z1_0    DW      Z1          ; TEMP STACK
05E0  0530 R            Z2_0    DW      Z2          ; TEMP STACK
05E2  053C R            Z3_0    DW      Z3          ; TEMP STACK
05E4  0551 R            Z4_0    DW      Z4          ; TEMP STACK

                        ;------- CLEAR STORAGE ENTRY

05E6                    CLR_STG:
                                ASSUME  DS:DATA
05E6  F3/ AB                    REP     STOSW       ; STORE 32K WORDS OF 0000
05E8  B8 ---- R                 MOV     AX,DATA     ; RESTORE DATA SEGMENT
05EB  8E D8                     MOV     DS,AX       ;
05ED  89 1E 0072 R              MOV     RESET_FLAG,BX ; RESTORE RESET FLAG

                        ;----- SETUP STACK SEG AND SP

05F1  B8 ---- R         C33:    MOV     AX,DATA     ; SET DATA SEGMENT
05F4  8E D8                     MOV     DS,AX       ;
05F6  BC 0000                   MOV     SP,POST_SS  ; GET STACK VALUE
05F9  8E D4                     MOV     SS,SP       ; SET THE STACK UP
```

SECTION 5

```
05FB  BC 8000                      MOV    SP,POST_SP            ; STACK IS READY TO GO

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

05FE  B0 11             C37:       MOV    AL,11H                ; <><><><><><><><><><><><>
0600  E6 80                        OUT    MFG_PORT,AL           ; <><><>CHECKPOINT 11 <><>

0602  E4 82                        IN     AL,DMA_PAGE+1         ; GET THE SWITCH SETTINGS
0604  24 F0                        AND    AL,0F0H               ; STRIP UNUSED BITS
0606  A2 0012 R                    MOV    MFG_TST,AL            ; SAVE SETTINGS
0609  2A C0                        SUB    AL,AL                 ; RESET DMA_PAGE
060B  E6 82                        OUT    DMA_PAGE+1,AL         ;
                        ;---------------------------------------------------
                        ; TEST.11A                                         :
                        ;          VERIFY 286 LGDT/SGDT LIDT/SIDT          :
                        ;          INSTRUCTIONS                            :
                        ; DESCRIPTION                                      :
                        ;          LOAD GDT AND IDT REGISTERS WITH         :
                        ;          AA,55,00 AND VERIFY CORRECT             :
                        ;---------------------------------------------------

                        ;-------- VERIFY STATUS INDICATE COMPABILITY (REAL) MODE

                                   SMSW   AX                    ; GET THE CURRENT STATUS WORD
060D  0F                     +     DB     00FH
060E                         + ??0000 LABEL BYTE
060E  D1 E0                  +     SHL    AX,1
0610                         + ??0001 LABEL BYTE
060E                         +     ORG    OFFSET CS:??0000
060E  01                     +     DB     001H
0610                         +     ORG    OFFSET CS:??0001
0610  A9 000F                      TEST   AX,0FH                ; PE/MP/EM/TS BITS SHOULD BE ZERO
0613  75 37                        JNZ    ERR_PROT              ; GO IF STATUS NOT REAL MODE

                        ;-------- TEST PROTECTED MODE REGISTERS

0615  B0 12                        MOV    AL,12H                ;SET CHECK POINT 12
0617  E6 80                        OUT    MFG_PORT,AL           ;<><><><><><><><><><><><>

0619  1E                           PUSH   DS                    ; SET ES TO SAME SEGMENT AS DS
061A  07                           POP    ES                    ;
061B  BF D0A0                      MOV    DI,SYS_IDT_LOC        ; USE THIS AREA TO BUILD TEST PATTERN
061E  B9 0003                      MOV    CX,3                  ;
0621  B8 AAAA                      MOV    AX,0AAAAH             ; FIRST PATTERN
0624  E8 064F R                    CALL   WRT_PAT               ;
0627  B8 5555                      MOV    AX,05555H             ;
062A  E8 064F R                    CALL   WRT_PAT               ; WRITE NEXT PATTERN
062D  2B C0                        SUB    AX,AX                 ; WRITE 0
062F  E8 064F R                    CALL   WRT_PAT               ;
0632  2B ED                        SUB    BP,BP                 ; RESTORE BP REG

                        ;-------- TEST 286 CONTROL FLAGS

0634  FD                           STD                          ; SET DIRECTION FLAG FOR DECREMENT
0635  9C                           PUSHF                        ; GET THE FLAGS
0636  58                           POP    AX                    ;
0637  A9 0200                      TEST   AX,0200H              ; INTERRUPT FLAG SHOULD BE OFF
063A  75 10                        JNZ    ERR_PROT              ; GO IF NOT
063C  A9 0400                      TEST   AX,0400H              ; CHECK DIRECTION FLAG
063F  74 0B                        JZ     ERR_PROT              ; GO IF NOT SET
0641  FC                           CLD                          ; CLEAR DIRECTION FLAG
0642  9C                           PUSHF                        ; INSURE DIRECTION FLAG IS RESET
0643  58                           POP    AX                    ;
0644  A9 0400                      TEST   AX,0400H              ;
0647  75 03                        JNZ    ERR_PROT              ; GO IF NOT

0649  EB 3E 90                     JMP    C37A                  ; TEST OK CONTINUE
064C                    ERR_PROT:
064C  F4                           HLT                          ; PROTECTED MODE REGISTER FAILURE
064D  EB FD                        JMP    SHORT ERR_PROT        ; INSURE NO BREAKOUT OF HALT

                        ;------- WRITE TO 286 REGISTERS

064F  B9 0003           WRT_PAT:MOV   CX,3                  ;
0652  F3/ AB            REP        STOSW                        ; STORE 6 BYTES OF PATTERN
0654  BD D0A0                      MOV    BP,SYS_IDT_LOC        ;
                                   SEGOV  ES                    ; LOAD THE IDT
0657  26                     +     DB     026H
                                   LIDT   [BP]                  ;  REGISTER FROM THIS AREA
0658  0F                     +     DB     00FH
0659                         + ??0003 LABEL BYTE
0659  8B 5E 00               +     MOV    BX,WORD PTR [BP]
065C                         + ??0004 LABEL BYTE
0659                         +     ORG    OFFSET CS:??0003
0659  01                     +     DB     001H
065C                         +     ORG    OFFSET CS:??0004
065C  BD D0A0                      MOV    BP,SYS_IDT_LOC        ;
                                   SEGOV  ES                    ; LOAD THE GDT
065F  26                     +     DB     026H
                                   LGDT   [BP]                  ;  FROM THE SAME AREA
0660  0F                     +     DB     00FH
0661                         + ??0006 LABEL BYTE
0661  8B 56 00               +     MOV    DX,WORD PTR [BP]
0664                         + ??0007 LABEL BYTE
0661                         +     ORG    OFFSET CS:??0006
0661  01                     +     DB     001H
0664                         +     ORG    OFFSET CS:??0007
                        ;------- READ AND VERIFY 286 REGISTERS

0664  BD D8A0                      MOV    BP,GDT_LOC            ; STORE THE REGISTERS HERE
                                   SEGOV  ES
0667  26                     +     DB     026H
                                   SIDT   [BP]                  ; GET THE IDT REGS
0668  0F                     +     DB     00FH
0669                         + ??0009 LABEL BYTE
0669  8B 4E 00               +     MOV    CX,[BP]
066C                         + ??000A LABEL BYTE
0669                         +     ORG    OFFSET CS:??0009
0669  01                     +     DB     001H
066C                         +     ORG    OFFSET CS:??000A
066C  BD D8A5                      MOV    BP,GDT_LOC+5          ;
                                   SEGOV  ES                    ;
066F  26                     +     DB     026H
                                   SGDT   [BP]                  ; GET THE GDT REGS
0670  0F                     +     DB     00FH
0671                         + ??000C LABEL BYTE
0671  03 46 00               +     ADD    AX,[BP]
0674                         + ??000D LABEL BYTE
0671                         +     ORG    OFFSET CS:??000C
0671  01                     +     DB     001H
0674                         +     ORG    OFFSET CS:??000D
0674  BF D0A0                      MOV    DI,SYS_IDT_LOC        ;
0677  8B 05                        MOV    AX,DS:[DI]            ; GET THE PATTERN WRITTEN
0679  B9 0005                      MOV    CX,5                  ; CHECK ALL REGISTERS
067C  BE D8A0                      MOV    SI,GDT_LOC            ; POINT TO THE BEGINNING
067F  26: 3B 04         C37B:      CMP    AX,ES:[SI]            ;
```

```
0682  75 C8                 JNZ     ERR_PROT                ; HALT IF ERROR
0684  46                    INC     SI                      ; POINT TO NEXT WORD
0685  46                    INC     SI                      ;
0686  E2 F7                 LOOP    C37B                    ; CONTINUE TILL DONE
0688  C3                    RET
                    ;--------------------------------------------------------
                    ;       INITIALIZE THE 8259 INTERRUPT #1 CONTROLLER CHIP :
                    ;--------------------------------------------------------

0689  2A C0         C37A:   SUB     AL,AL                   ; RESET MATH PROCESSOR
068B  E6 F1                 OUT     X287+1,AL               ;
068D  B0 11                 MOV     AL,11H                  ; ICW1 - EDGE, MASTER, ICW4
068F  E6 20                 OUT     INTA00,AL
0691  EB 00                 JMP     SHORT $+2               ; WAIT STATE FOR IO
0693  B0 08                 MOV     AL,8                    ; SETUP ICW2 - INT TYPE 8 (8-F)
0695  E6 21                 OUT     INTA01,AL
0697  EB 00                 JMP     SHORT $+2               ; WAIT STATE FOR IO

0699  B0 04                 MOV     AL,04H                  ; SETUP ICW3 - MASTER LV 2
069B  E6 21                 OUT     INTA01,AL               ;
069D  EB 00                 JMP     SHORT $+2               ; IO WAIT STATE
069F  B0 01                 MOV     AL,01H                  ; SETUP ICW4 - MASTER,8086 MODE
06A1  E6 21                 OUT     INTA01,AL               ;
06A3  EB 00                 JMP     SHORT $+2               ; WAIT STATE FOR IO
06A5  B0 FF                 MOV     AL,0FFH                 ; MASK ALL INTS. OFF
06A7  E6 21                 OUT     INTA01,AL               ; (VIDEO ROUTINE ENABLES INTS.)

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

06A9  B0 13                 MOV     AL,13H                  ; <><><><><><><><><><><><>
06AB  E6 80                 OUT     MFG_PORT,AL             ; <><><>CHECKPOINT 13 <><>

06AD  B0 11                 MOV     AL,11H                  ; ICW1 - EDGE, SLAVE ICW4
06AF  E6 A0                 OUT     INTB00,AL
06B1  EB 00                 JMP     SHORT $+2               ; WAIT STATE FOR IO
06B3  B0 70                 MOV     AL,INT_TYPE             ; SETUP ICW2 - INT TYPE 50 (50-5F)
06B5  E6 A1                 OUT     INTB01,AL               ;
06B7  B0 02                 MOV     AL,02H                  ; SETUP ICW3 - SLAVE LV 2
06B9  EB 00                 JMP     SHORT $+2               ;
06BB  E6 A1                 OUT     INTB01,AL               ;
06BD  EB 00                 JMP     SHORT $+2               ; IO WAIT STATE
06BF  B0 01                 MOV     AL,01H                  ; SETUP ICW4 - 8086 MODE, SLAVE
06C1  E6 A1                 OUT     INTB01,AL               ;
06C3  EB 00                 JMP     SHORT $+2               ; WAIT STATE FOR IO
06C5  B0 FF                 MOV     AL,0FFH                 ; MASK ALL INTS. OFF
06C7  E6 A1                 OUT     INTB01,AL               ;
                    ;------- SET UP THE INTERRUPT VECTORS TO TEMP INTERRUPT

06C9  B0 14                 MOV     AL,14H                  ; <><><><><><><><><><><><>
06CB  E6 80                 OUT     MFG_PORT,AL             ; <><><>CHECKPOINT 14 <><>

06CD  B9 0078               MOV     CX,78H                  ; FILL ALL INTERRUPT LOCATIONS
06D0  2B FF                 SUB     DI,DI                   ; FIRST INTERRUPT LOCATION
06D2  8E C7                 MOV     ES,DI                   ; SET ES ALSO
06D4  B8 0000 E     D3:     MOV     AX,OFFSET D11           ; MOVE ADDRESS OF INT OFFSET
06D7  AB                    STOSW
06D8  8C C8                 MOV     AX,CS                   ; GET THE SEGMENT
06DA  AB                    STOSW
06DB  E2 F7                 LOOP    D3                      ;

                    ;------- ESTABLISH BIOS SUBROUTINE CALL INTERRUPT VECTORS
06DD  B0 15                 MOV     AL,15H                  ; <><><><><><><><><><><><>
06DF  E6 80                 OUT     MFG_PORT,AL             ; <><><>CHECKPOINT 15 <><>

06E1  BF 0040 R             MOV     DI,OFFSET VIDEO_INT     ; SET VIDIO INT AREA
06E4  0E                    PUSH    CS                      ;
06E5  1F                    POP     DS                      ; SET UP ADDRESS OF VECTOR TABLE
06E6  8C D8                 MOV     AX,DS                   ; SET AX=SEGMENT
06E8  BE 0010 E             MOV     SI,OFFSET VECTOR_TABLE+16 ;START WITH VIDEO ENTRY
06EB  B9 0010               MOV     CX,16                   ;

06EE  A5            D3A:    MOVSW                           ; MOVE VECTOR TABLE TO RAM
06EF  47                    INC     DI
06F0  47                    INC     DI                      ; SKIP SEGMENT POINTER
06F1  E2 FB                 LOOP    D3A

                    ;----------------------------------------------
                    ; TEST.12                                     :
                    ;       VERIFY CMOS CHECKSUM/BATTERY GOOD     ;
                    ; DESCRIPTION                                 :
                    ;       DETERMINE IF CONFIG RECORD SHOULD BE  :
                    ;       USED FOR INITIALIZATION               :
                    ;----------------------------------------------
                            ASSUME  DS:DATA
06F3  E8 0000 E     CMOS:   CALL    DDS                     ; SET THE DATA SEGMENT

06F6  B0 16                 MOV     AL,16H                  ; <><><><><><><><><><><><>
06F8  E6 80                 OUT     MFG_PORT,AL             ; <><><>CHECKPOINT 16 <><>

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

06FA  B0 8D                 MOV     AL,BATTERY_COND_STATUS  ; CHECK BATTERY CONDITION
06FC  E6 70                 OUT     CMOS_PORT,AL            ;  POINT TO BATTERY STATUS
06FE  EB 00                 JMP     SHORT $+2               ; WAIT STATE FOR IO
0700  E4 71                 IN      AL,CMOS_PORT+1          ;
0702  A8 80                 TEST    AL,80H                  ; IS THE BATTERY LOW?
0704  74 0F                 JZ      CMOS1A                  ; GO IF YES
0706  B0 8E                 MOV     AL,DIAG_STATUS          ; GET THE OLD STATUS
0708  E6 70                 OUT     CMOS_PORT,AL            ;
070A  EB 00                 JMP     SHORT $+2               ;
070C  E4 71                 IN      AL,CMOS_PORT+1          ;
070E  A8 80                 TEST    AL,BAD_BAT              ; HAS CUSTOMER SETUP BEEN EXECUTED?
0710  74 21                 JZ      CMOS1                   ; GO CHECK CHECKSUM IF YES

0712  E9 07A1 R             JMP     CMOS4                   ; CONTINUE WITHOUT CONFIG

                    ;------ SET DEFECTIVE BATTERY FLAG

0715  B0 17         CMOS1A: MOV     AL,17H                  ; <><><><><><><><><><><><>
0717  E6 80                 OUT     MFG_PORT,AL             ; <><><>CHECKPOINT 17 <><>

0719  B0 8E                 MOV     AL,DIAG_STATUS          ; CMOS DIAGNOSTIC STATUS BYTE
071B  E6 70                 OUT     CMOS_PORT,AL            ;
071D  EB 00                 JMP     SHORT $+2               ;
071F  E4 71                 IN      AL,CMOS_PORT+1          ; GET THE CURRENT STATUS
0721  86 C4                 XCHG    AL,AH                   ; SAVE
0723  80 CC 80              OR      AH,BAD_BAT              ; SET THE DEAD BATTERY FLAG
0726  B0 8E                 MOV     AL,DIAG_STATUS          ;
0728  E6 70                 OUT     CMOS_PORT,AL            ;
072A  86 C4                 XCHG    AL,AH                   ; OUTPUT THE STATUS
072C  EB 00                 JMP     SHORT $+2               ;
072E  E6 71                 OUT     CMOS_PORT+1,AL          ; SET FLAG IN CMOS
```

SECTION 5

```
0730  EB 6F 90                         JMP    CMOS4                  ; GO TO MINIMUM CONFIG
                               ;----- VERIFY CHECKSUM

0733  B0 8E                    CMOS1:  MOV    AL,DIAG_STATUS         ; CLEAR OLD STATUS
0735  E6 70                            OUT    CMOS_PORT,AL           ;
0737  EB 00                            JMP    SHORT $+2              ; IO DELAY
0739  E4 71                            IN     AL,CMOS_PORT+1         ; GET THE CURRENT STATUS
073B  EB 00                            JMP    SHORT $+2              ; IO DELAY
073D  86 C4                            XCHG   AL,AH                  ; SAVE THE CURRENT STATUS
073F  B0 8E                            MOV    AL,DIAG_STATUS         ;
0741  E6 70                            OUT    CMOS_PORT,AL           ;
0743  81 3E 0072 R 1234                CMP    RESET_FLAG,1234H       ; IS THIS A SOFT RESET
0749  75 07                            JNZ    CMOS1_A                ; GO IF NOT
074B  86 E0                            XCHG   AH,AL                  ; RESTORE THE STATUS
074D  24 10                            AND    AL,W_MEM_SIZE          ; CLEAR ALL BUT THE CMOS/POR MEMORY SIZE
                                                                     ;   MISCOMPARE
074F  EB 03 90                         JMP    CMOS1_B                ;

0752                           CMOS1_A:
0752  2A C0                            SUB    AL,AL                  ;
0754  E6 71                    CMOS1_B:OUT    CMOS_PORT+1,AL         ;

0756  2B DB                            SUB    BX,BX
0758  2B C9                            SUB    CX,CX                  ;
075A  B1 90                            MOV    CL,CMOS_BEGIN          ; SET START OF CMOS
075C  B5 AE                            MOV    CH,CMOS_END+1          ; SET END OF CMOS

075E  8A C1                    CMOS2:  MOV    AL,CL                  ;
0760  E6 70                            OUT    CMOS_PORT,AL           ; ADDRESS THE BEGINNING
0762  EB 00                            JMP    SHORT $+2              ; WAIT STATE FOR IO
0764  E4 71                            IN     AL,CMOS_PORT+1         ;
0766  2A E4                            SUB    AH,AH                  ; INSURE AH=0
0768  13 D8                            ADC    BX,AX                  ; ADD TO CURRENT VALUE
076A  FE C1                            INC    CL                     ; POINT TO NEXT WORD
076C  3A E9                            CMP    CH,CL                  ; FINISHED?
076E  75 EE                            JNZ    CMOS2                  ; GO IF NOT
0770  0B DB                            OR     BX,BX                  ; BX MUST NOT BE 0
0772  74 16                            JZ     CMOS3                  ; CMOS BAD IF CKSUM=0
0774  B0 AE                            MOV    AL,CMOS_END+1          ; GET THE CHECK SUM
0776  E6 70                            OUT    CMOS_PORT,AL           ;
0778  EB 00                            JMP    SHORT $+2              ;
077A  E4 71                            IN     AL,CMOS_PORT+1         ; FIRST BYTE OF CHECKSUM
077C  8A E0                            MOV    AH,AL                  ; SAVE IT
077E  B0 AF                            MOV    AL,CMOS_END+2          ; SECOND BYTE OF CHECKSUM
0780  E6 70                            OUT    CMOS_PORT,AL           ;
0782  EB 00                            JMP    SHORT $+2              ;
0784  E4 71                            IN     AL,CMOS_PORT+1         ;
0786  3B C3                            CMP    AX,BX                  ; IS THE CHECKSUM OK
0788  74 17                            JZ     CMOS4                  ; GO IF YES

                               ;------- SET CMOS CHECKSUM ERROR

078A  B0 8E                    CMOS3:  MOV    AL,DIAG_STATUS         ; SET BAD CHECKSUM FLAG
078C  E6 70                            OUT    CMOS_PORT,AL           ;
078E  EB 00                            JMP    SHORT $+2              ; IO DELAY
0790  E4 71                            IN     AL,CMOS_PORT+1         ; GET THE CURRENT STATUS
0792  86 C4                            XCHG   AL,AH                  ; SAVE IT
0794  80 CC 40                         OR     AH,BAD_CKSUM           ; SET BAD CHECKSUM FLAG
0797  B0 8E                            MOV    AL,DIAG_STATUS         ;
0799  E6 70                            OUT    CMOS_PORT,AL           ;
079B  EB 00                            JMP    SHORT $+2              ; IO DELAY
079D  86 C4                            XCHG   AL,AH                  ; SET FLAG
079F  E6 71                            OUT    CMOS_PORT+1,AL         ;
07A1  B0 18                    CMOS4:  MOV    AL,18H                 ; <><><><><><><><><><><><><>
07A3  E6 80                            OUT    MFG_PORT,AL            ; <><><>CHECKPOINT 18 <><>
                               ;--------------------------------------------------------
                               ;                                                       :
                               ;       ENABLE PROTECTED MODE                           :
                               ;                                                       :
                               ;--------------------------------------------------------
07A5  E4 61                            IN     AL,PORT_B              ; DISABLE IO/RAM PARITY CHK
07A7  0C 0C                            OR     AL,RAM_PAR_OFF         ;
07A9  EB 00                            JMP    SHORT $+2              ; IO DELAY
07AB  E6 61                            OUT    PORT_B,AL              ;

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

07AD  B0 19                            MOV    AL,19H                 ; <><><><><><><><><><><><><>
07AF  E6 80                            OUT    MFG_PORT,AL            ; <><><>CHECKPOINT 19 <><>

07B1  B0 8F                            MOV    AL,SHUT_DOWN           ; SET THE RETURN ADDR
07B3  E6 70                            OUT    CMOS_PORT,AL           ;
07B5  EB 00                            JMP    SHORT $+2              ; IO DELAY
07B7  B0 01                            MOV    AL,01H                 ; FIRST SHUTDOWN RETN ADDR
07B9  E6 71                            OUT    CMOS_PORT+1,AL         ;

07BB  BC 0000                          MOV    SP,POST_SS             ; SET STACK FOR SYSINIT1
07BE  8E D4                            MOV    SS,SP                  ;
07C0  BC 8000                          MOV    SP,POST_SP             ;
07C3  E8 0000 E                        CALL   SYSINIT1               ; CALL THE DESCRIPTOR TABLE BUILDER
                                                                     ;  AND REAL-TO-PROTECTED MODE SWITCHER

07C6  B0 1A                            MOV    AL,1AH                 ; <><><><><><><><><><><><><>
07C8  E6 80                            OUT    MFG_PORT,AL            ; <><><>CHECKPOINT 1A <><>

                               ;-------- SET TEMPORY STACK

07CA  B8 0008                          MOV    AX,GDT_PTR             ;
07CD  8E D8                            MOV    DS,AX                  ;
07CF  C7 06 005A 0000                  MOV    DS:SS_TEMP.BASE_LO_WORD,0
07D5  C6 06 005C 00                    MOV    BYTE PTR DS:(SS_TEMP.BASE_HI_BYTE),TEMP_STACK_HI
07DA  BE 0058                          MOV    SI,SS_TEMP             ;
07DD  8E D6                            MOV    SS,SI                  ;
07DF  BC FFFD                          MOV    SP,MAX_SEG_LEN-2       ;
                               ;-----------------------------------------
                               ; TEST.13                                :
                               ;  PROTECTED MODE TEST                   :
                               ;    CHECK MSW FOR PROTECTED MODE        :
                               ;  MEMORY SIZE DETERMINE (RAM -> 640K)   :
                               ; DESCRIPTION                            :
                               ;  THIS ROUTINE RUNS IN PROTECTED MODE IN :
                               ;  ORDER TO ADDRESS ALL STORAGE          :
                               ; MEMORY SIZE IS SAVED AT MEMORY_SIZE    :
                               ; CMOS DIAGNOSTIC BYTE BIT 4 = 512 -> 640K :
                               ;-----------------------------------------

                               ;------- INSURE PROTECTED MODE

                                       SMSW   AX                            ; GET THE MACHINE STATUS WORD
07E2  0F                      +        DB     00FH
07E3                          + ??000E LABEL  BYTE
07E3  D1 E0                   +        SHL    AX,1
07E5                          + ??000F LABEL  BYTE
07E3                          +        ORG    OFFSET CS:??000E
07E3  01                      +        DB     001H
```

```
07E5                                  +         ORG     OFFSET CS:??000F
07E5  A9 0001                                   TEST    AX,VIRTUAL_ENABLE           ; ARE WE IN PROTECTED MODE
07E8  75 10                                     JNZ     VIR_OK

07EA  B0 8F                           SHUT_8:  MOV     AL,SHUT_DOWN                ; SET THE RETURN ADDR
07EC  E6 70                                     OUT     CMOS_PORT,AL                ;
07EE  EB 00                                     JMP     SHORT $+2                   ; IO DELAY
07F0  B0 08                                     MOV     AL,08H                      ; SET SHUTDOWN 8
07F2  E6 71                                     OUT     CMOS_PORT+1,AL              ;
07F4  E9 0000 E                                 JMP     PROC_SHUTDOWN               ; CAUSE A SHUTDOWN

                                      ;-------- VIRTUAL MODE ERROR HALT

07F7  F4                              SHUT8:   HLT
07F8  EB FL                                     JMP     SHUT8                       ; ERROR HALT

                                      ;------- 64K SEGMENT LIMIT

07FA  C7 06 0048 FFFF                 VIR_OK:  MOV     DS:ES_TEMP.SEG_LIMIT,MAX_SEG_LEN

                                      ;------- CPLO, DATA ACCESS RIGHTS

0800  C6 06 004D 93                             MOV     BYTE PTR DS:(ES_TEMP.DATA_ACC_RIGHTS),CPLO_DATA_ACCESS

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

0805  C6 06 004C 01                             MOV     BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE),01H
080A  C7 06 004A 0000                           MOV     DS:ES_TEMP.BASE_LO_WORD,0H

0810  B0 1B                                     MOV     AL,1BH                  ; <><><><><><><><><><><><>
0812  E6 80                                     OUT     MFG_PORT,AL             ; <><><>CHECKPOINT 1B <><>

0814  BB 0040                                   MOV     BX,16*4                 ; SET THE FIRST 64K DONE


                                      ;------- START STORAGE SIZE/CLEAR

0817                                  NOT_DONE:
0817  B8 0048                                   MOV     AX,ES_TEMP              ; POINT ES TO DATA
081A  8E C0                                     MOV     ES,AX                   ; POINT TO SEGMENT TO TEST
081C  E8 0838 R                                 CALL    HOW_BIG                 ; DO THE FIRST 64K
081F  74 03                                     JZ      NOT_FIN                 ; CHECK IF TOP OF RAM
0821  E9 08B7 R                                 JMP     DONE                    ;
0824                                  NOT_FIN:
0824  83 C3 40                                  ADD     BX,16*4                 ; BUMP MEMORY COUNT BY 64K

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

0827  FE 06 004C                                INC     BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE)

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

082B  80 3E 004C 0A                             CMP     BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE),0AH
0830  75 E5                                     JNZ     NOT_DONE                                ; GO IF NOT
0832  E8 088B R                                 CALL    HOW_BIG_END                             ; GO SET MEMORY SIZE
0835  E9 08B7 R                                 JMP     DONE

                                      ;------- FILL/CHECK LOOP

0838                                  HOW_BIG:
0838  2B FF                                     SUB     DI,DI
083A  B8 AA55                                   MOV     AX,0AA55H               ; TEST PATTERN
083D  8B C8                                     MOV     CX,AX                   ; SAVE PATTERN
083F  26: 89 05                                 MOV     ES:[DI],AX              ; SEND PATTERN TO MEM.
0842  B0 0F                                     MOV     AL,0FH                  ; PUT SOMETHING IN AL
0844  26: 8B 05                                 MOV     AX,ES:[DI]              ; GET PATTERN
0847  26: 89 05                                 MOV     ES:[DI],AX              ; INSURE NO PARITY IO CHECK
084A  33 C1                                     XOR     AX,CX                   ; COMPARE PATTERNS
084C  75 3D                                     JNZ     HOW_BIG_END             ; GO END IF NO COMPARE

084E  1E                                        PUSH    DS                      ;
084F  B8 0018                                   MOV     AX,RSDA_PTR             ; POINT TO SYSTEM DATA AREA
0852  8E D8                                     MOV     DS,AX                   ;
0854  81 3E 0072 R 1234                         CMP     RESET_FLAG,1234H        ; SOFT RESET
085A  1F                                        POP     DS                      ; RESTORE DS
085B  75 26                                     JNZ     HOW_BIG_2               ; GO IF NOT SOFT RESET

085D  26: C7 05 0101                            MOV     WORD PTR ES:[DI],0101H  ; TURN OFF BOTH PARITY BITS

0862  E4 61                                     IN      AL,PORT_B
0864  EB 00                                     JMP     SHORT $+2               ; IO DELAY
0866  0C 0C                                     OR      AL,RAM_PAR_OFF          ; TOGGLE PARITY CHECK ENABLES
0868  E6 61                                     OUT     PORT_B,AL
086A  EB 00                                     JMP     SHORT $+2               ; IO DELAY
086C  24 F3                                     AND     AL,RAM_PAR_ON
086E  E6 61                                     OUT     PORT_B,AL

0870  B8 FFFF                                   MOV     AX,0FFFFH
0873  50                                        PUSH    AX
0874  58                                        POP     AX                      ; DELAY
0875  26: 8B 05                                 MOV     AX,ES:[DI]              ; CHECK PARITY

0878  E4 61                                     IN      AL,PORT_B               ; CHECK FOR PARITY/IO CHECK
087A  24 C0                                     AND     AL,PARITY_ERR           ;

087C  26: C7 05 0000                            MOV     WORD PTR ES:[DI],0      ; INSURE NO PARITY IO CHECK
0881  75 08                                     JNZ     HOW_BIG_END             ; GO IF PARITY/IO CHECK
0883                                  HOW_BIG_2:
0883  2B C0                                     SUB     AX,AX                   ; WRITE ZEROS

0885  B9 8000                                   MOV     CX,2000H*4              ; SET COUNT FOR 32K WORDS
0888  F3/ AB                                    REP     STOSW                   ; FILL 32K WORDS
088A  C3                                        RET                             ;

088B                                  HOW_BIG_END:
088B  9C                                        PUSHF                           ; SAVE THE CURRENT FLAGS
088C  B0 1C                                     MOV     AL,1CH                  ; <><><><><><><><><><><><>
088E  E6 80                                     OUT     MFG_PORT,AL             ; <><><>CHECKPOINT 1C <><><>

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

0890  B0 B3                                     MOV     AL,INFO_STATUS          ; SET/RESET 640K STATUS FLAG
0892  E6 70                                     OUT     CMOS_PORT,AL            ;
0894  EB 00                                     JMP     SHORT $+2               ; IO DELAY
0896  E4 71                                     IN      AL,CMOS_PORT+1          ; GET THE DIAGNOSTIC STATUS
0898  0C 80                                     OR      AL,M640K                ;
089A  86 C4                                     XCHG    AL,AH                   ; SAVE THE STATUS
089C  B0 B3                                     MOV     AL,INFO_STATUS          ;
089E  E6 70                                     OUT     CMOS_PORT,AL            ;
08A0  86 C4                                     XCHG    AL,AH                   ; RESTORE THE STATUS
08A2  81 FB 0200                                CMP     BX,512                  ; CHECK MEMORY SIZE
08A6  77 02                                     JA      K640                    ; SET FLAG FOR 512 -> 640 INSTALLED
08A8  24 7F                                     AND     AL,NOT M640K            ;
08AA  E6 71                           K640:    OUT     CMOS_PORT+1,AL          ;
```

SECTION 5

```
08AC  B8 0018                    MOV   AX,RSDA_PTR                  ; RESTORE THE DATA SEGMENT
08AF  8E D8                      MOV   DS,AX                        ;
08B1  89 1E 0013 R               MOV   MEMORY_SIZE,BX               ; SAVE MEMORY SIZE
08B5  9D                         POPF                               ; RESTORE THE FLAG REG
08B6  C3                         RET
                           ;-------------------------------------------
                           ; TEST.13A                                 :
                           ;  MEMORY SIZE DETERMINE (RAM ABOVE 1024K) :
                           ; DESCRIPTION                              :
                           ;  THIS ROUTINE RUNS IN PROTECTED MODE     :
                           ; MEMORY SIZE ABOVE 1MEG ADDRESSING IS     :
                           ; SAVED IN CMOS                            :
                           ;-------------------------------------------

08B7                       DONE:

08B7  B8 0008                    MOV   AX,GDT_PTR                   ; POINT DS TO THE DESCRIPTER TABLE
08BA  8E D8                      MOV   DS,AX                        ;

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

08BC  C6 06 004C 10              MOV   BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE),10H
08C1  C7 06 004A 0000            MOV   DS:ES_TEMP.BASE_LO_WORD,0H

08C7  B0 1D                      MOV   AL,1DH                       ; <><><><><><><><><><><><>
08C9  E6 80                      OUT   MFG_PORT,AL                  ; <><><>CHECKPOINT 1D <><>

08CB  2B DB                      SUB   BX,BX                        ; START WITH COUNT 0

                           ;------- START STORAGE SIZE/CLEAR

08CD                       NOT_DONE1:
08CD  B8 0048                    MOV   AX,ES_TEMP                   ; POINT ES TO DATA
08D0  8E C0                      MOV   ES,AX                        ; POINT TO SEGMENT TO TEST
08D2  E8 08EE R                  CALL  HOW_BIG1                     ; DO THE FIRST 64K

08D5  74 03                      JZ    DONEA                        ; CHECK IF TOP
08D7  EB 75 90                   JMP   DONE1                        ; GO IF TOP

08DA  83 C3 40             DONEA: ADD  BX,16*4                      ; BUMP MEMORY COUNT BY 64K

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

08DD  FE 06 004C                 INC   BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE)

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

08E1                       NOT_END_BASE:
08E1  80 3E 004C FE              CMP   BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE),0FEH        ; LAST OF POSSIBLE RAM?
08E6  75 E5                      JNZ   NOT_DONE1                                      ; GO IF NOT
08E8  E8 0933 R                  CALL  HOW_BIG_END1                                   ; GO SET MEMORY SIZE
08EB  EB 61 90                   JMP   DONE1
                           ;------- FILL/CHECK LOOP

08EE                       HOW_BIG1:
08EE  2B FF                      SUB   DI,DI
08F0  B8 AA55                    MOV   AX,0AA55H                    ; TEST PATTERN
08F3  8B C8                      MOV   CX,AX                        ; SAVE PATTERN
08F5  26: 89 05                  MOV   ES:[DI],AX                   ; SEND PATTERN TO MEM.
08F8  B0 0F                      MOV   AL,0FH                       ; PUT SOMETHING IN AL
08FA  26: 8B 05                  MOV   AX,ES:[DI]                   ; GET PATTERN
08FD  26: 89 05                  MOV   ES:[DI],AX                   ; INSURE NO PARITY IO CHECK
0900  33 C1                      XOR   AX,CX                        ; COMPARE PATTERNS
0902  75 2F                      JNZ   HOW_BIG_END1                 ; GO END IF NO COMPARE

0904  1E                         PUSH  DS                           ;
0905  B8 0018                    MOV   AX,RSDA_PTR                  ; POINT TO SYSTEM DATA AREA
0908  8E D8                      MOV   DS,AX                        ;
090A  81 3E 0072 R 1234          CMP   RESET_FLAG,1234H             ; SOFT RESET
0910  1F                         POP   DS                           ; RESTORE DS
0911  75 18                      JNZ   HOW_BIG_2A                   ; GO IF NOT SOFT RESET

0913  26: C7 05 0101             MOV   WORD PTR ES:[DI],0101H       ; TURN OFF BOTH PARITY BITS
0918  B8 FFFF                    MOV   AX,0FFFFH                    ;
091B  50                         PUSH  AX
091C  58                         POP   AX                           ; DELAY
091D  26: 8B 05                  MOV   AX,ES:[DI]                   ; CHECK PARITY
0920  E4 61                      IN    AL,PORT_B                    ; CHECK FOR IO CHECK
0922  A8 40                      TEST  AL,IO_CHK                    ;

0924  26: C7 05 0000             MOV   WORD PTR ES:[DI],0           ; INSURE NO PARITY IO CHECK
0929  75 08                      JNZ   HOW_BIG_END1                 ; GO IF IO CHECK
092B                       HOW_BIG_2A:
092B  2B C0                      SUB   AX,AX                        ; WRITE ZEROS
092D  B9 8000                    MOV   CX,2000H*4                   ; SET COUNT FOR 32K WORDS
0930  F3/ AB                     REP   STOSW                        ; FILL 32K WORDS
0932  C3                         RET                                ;

0933                       HOW_BIG_END1:
0933  B0 1E                      MOV   AL,1EH                       ; <><><><><><><><><><><><>
0935  E6 80                      OUT   MFG_PORT,AL                  ; <><>CHECKPOINT 1E <><><>

                           ;------- SET IO RAM SIZE IN CMOS

0937  B0 B0                      MOV   AL,M_SIZE_LO                 ;
0939  E6 70                      OUT   CMOS_PORT,AL                 ; ADDRESS LO BYTE
093B  EB 00                      JMP   SHORT $+2                    ; IO DELAY
093D  8A C3                      MOV   AL,BL                        ; SET LOW MEMORY SIZE
093F  E6 71                      OUT   CMOS_PORT+1,AL               ;   IN CMOS
0941  EB 00                      JMP   SHORT $+2                    ; IO DELAY
0943  B0 B1                      MOV   AL,M_SIZE_HI                 ; ADDRESS HI BYTE
0945  E6 70                      OUT   CMOS_PORT,AL                 ;
0947  EB 00                      JMP   SHORT $+2                    ; IO DELAY
0949  8A C7                      MOV   AL,BH                        ; SET THE HIGH MEMORY SIZE
094B  E6 71                      OUT   CMOS_PORT+1,AL               ;   IN CMOS

094D  C3                         RET

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

094E  B0 1F                DONE1: MOV  AL,1FH                       ; <><><><><><><><><><><><>
0950  E6 80                      OUT   MFG_PORT,AL                  ; <><><>CHECKPOINT 1F <><>
0952  C6 06 004C 00              MOV   BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE),00H
0957  BA FFFF                    MOV   DX,0FFFFH                    ; WRITE FFFF AT ADDRESS 0
095A  E8 098A R                  CALL  SDO
095D  2B D2                      SUB   DX,DX                        ; WRITE 0

095F  C6 06 004C 08              MOV   BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE),08H
0964  E8 098A R                  CALL  SDO                          ;
0967  C6 06 004C 10              MOV   BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE),10H
096C  E8 098A R                  CALL  SDO                          ;
096F  C6 06 004C 20              MOV   BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE),20H
0974  E8 098A R                  CALL  SDO                          ;
0977  C6 06 004C 40              MOV   BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE),40H
097C  E8 098A R                  CALL  SDO                          ;
```

```
097F  C6 06 004C 80                   MOV    BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE),80H
0984  E8 098A R                       CALL   SD0                       ;

0987  EB 20 90                        JMP    SD2                       ; TEST PASSED CONTINUE

098A  2B FF                  SD0:     SUB    DI,DI                     ;
098C  B8 0048                         MOV    AX,ES_TEMP                ; POINT ES TO DATA
098F  8E C0                           MOV    ES,AX                     ; POINT TO SEGMENT TO TEST
0991  26: 89 15                       MOV    ES:[DI],DX                ; WRITE THE PATTERN

0994  C6 06 004C 00                   MOV    BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE),00H

0999  B8 0048                         MOV    AX,ES_TEMP                ; POINT ES TO DATA
099C  8E C0                           MOV    ES,AX                     ; POINT TO SEGMENT TO TEST
099E  26: 81 3D FFFF                  CMP    WORD PTR ES:[DI],0FFFFH ; DID LOCATION 0 CHANGE?
09A3  74 03                           JZ     SD1                       ; CONTINUE IF NOT
09A5  E9 07EA R                       JMP    SHUT_8                    ; GO HALT IF YES
09A8  C3                     SD1:     RET

                             ;------- CAUSE A SHUTDOWN

09A9  B0 20                  SD2:     MOV    AL,20H                    ; <><><><><><><><><><><><>
09AB  E6 80                           OUT    MFG_PORT,AL               ; <><><>CHECKPOINT 20 <><>
09AD  E9 0000 E                       JMP    PROC_SHUTDOWN             ; CAUSE A SHUTDON (RETURN VIA JUMP
                             ;-----------------------------------------------------
                             ; RETURN 1 FROM SHUTDOWN                             :
                             ;-----------------------------------------------------

09B0  B0 21                  SHUT1:   MOV    AL,21H                    ; <><><><><><><><><><><><>
09B2  E6 80                           OUT    MFG_PORT,AL               ; <><><>CHECKPOINT 21 <><>
09B4  BC ---- R                       MOV    SP,STACK                  ; SET REAL MODE STACK
09B7  8E D4                           MOV    SS,SP                     ;
09B9  BC 0100 R                       MOV    SP,OFFSET TOS             ;

09BC  B8 ---- R                       MOV    AX,DATA                   ; SET UP THE REAL DATA AREA
09BF  8E D8                           MOV    DS,AX                     ;

                             ;-------- GET THE CONFIGURATION FROM CMOS

09C1  B0 8E                           MOV    AL,DIAG_STATUS            ; CHECK CMOS GOOD
09C3  E6 70                           OUT    CMOS_PORT,AL              ;
09C5  EB 00                           JMP    SHORT $+2
09C7  E4 71                           IN     AL,CMOS_PORT+1            ; GET THE STATUS
09C9  A8 C0                           TEST   AL,0C0H                   ; OK?
09CB  74 03                           JZ     M_OK                      ; GO IF YES
09CD  EB 77 90                        JMP    BAD_MOS                   ; GO IF NOT
09D0  8A E0                  M_OK:    MOV    AH,AL                     ; SAVE THE CMOS STATUS
09D2  B0 8E                           MOV    AL,DIAG_STATUS            ; ADDRESS THE DIAG STATUS
09D4  E6 70                           OUT    CMOS_PORT,AL
09D6  86 C4                           XCHG   AL,AH                     ; RESTORE THE STATUS BYTE
09D8  24 DF                           AND    AL,0DFH                   ; CLEAR THE MIN CONFIG BIT
09DA  E6 71                           OUT    CMOS_PORT+1,AL            ;
09DC  B0 94                           MOV    AL,C_EQUIP                ; GET THE EQUIPMENT BYTE
09DE  EB 00                           JMP    SHORT $+2                 ;
09E0  E6 70                           OUT    CMOS_PORT,AL              ;
09E2  EB 00                           JMP    SHORT $+2                 ; IO DELAY
09E4  E4 71                           IN     AL,CMOS_PORT+1

                             ;-------- INSURE CONFIGURATION HAS CORRECT VIDEO TYPE

09E6  8A E0                           MOV    AH,AL                     ; SAVE VIDEO TYPE
09E8  A8 30                           TEST   AL,030H                   ; ANY VIDEO?
09EA  75 2E                           JNZ    MOS_OK_1                  ; CONTINUE
09EC  E8 09FB R                       CALL   CHK_VIDEO                 ; INSURE VIDEO ROM PRESENT
09EF  74 4A                           JZ     MOS_OK                    ; CONTINUE

09F1  F6 06 0012 R 20                 TEST   MFG_TST,MFG_JMP           ; EXCEPT IF MFG JUMPER IS INSTALLED
09F6  74 7A                           JZ     NORMAL_CONFIG             ; GO IF INSTALLED

09F8  EB 4C 90                        JMP    BAD_MOS                   ; GO DEFAULT
                             ;-------- ROUTINE CHECK FOR VIDEO ROM PRESENT

09FB                         CHK_VIDEO:
09FB  B9 C000                         MOV    CX,0C000H                 ; START OF IO ROM
09FE                         CHK_VIDEO1:
09FE  50                              PUSH   AX                        ; SAVE THE CONFIG
09FF  1E                              PUSH   DS                        ; SAVE THE DATA SEGMENT
0A00  8E D9                           MOV    DS,CX                     ;
0A02  2B DB                           SUB    BX,BX                     ; GET THE FIRST 2 LOCATIONS
0A04  8B 07                           MOV    AX,[BX]                   ;
0A06  1F                              POP    DS                        ; RESTORE DATA SEG AND BUS SETTLE
0A07  3D AA55                         CMP    AX,0AA55H                 ; IS THE VIDEO ROM PRESENT?
0A0A  58                              POP    AX                        ; GET THE CONFIG
0A0B  74 0C                           JZ     CHK_VIDEO2                ; GO IF VIDEO ROM INSTALLED
0A0D  81 C1 0080                      ADD    CX,080H                   ; POINT TO NEXT 2K BLOCK
0A11  81 F9 C800                      CMP    CX,0C800H                 ; TOP OF VIDEO ROM AREA YET?
0A15  7C E7                           JL     CHK_VIDEO1                ; TRY AGAIN
0A17  23 C9                           AND    CX,CX                     ; SET NON ZERO FLAG
0A19                         CHK_VIDEO2:
0A19  C3                              RET                              ; RETURN TO CALLER

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

0A1A                         MOS_OK_1:
0A1A  E8 09FB R                       CALL   CHK_VIDEO                 ; IS THE VIDEO ROM INSTALLED?
0A1D  74 27                           JZ     BAD_MOS                   ; WRONG CONFIGURATION IN CONFIG BYTE

0A1F  8A C4                           MOV    AL,AH                     ; RESTORE CONFIGURATION
0A21  F6 06 0012 R 40                 TEST   MFG_TST,DSP_JMP           ; CHECK FOR DISPLAY JUMPER
0A26  74 0B                           JZ     MOS_OK_2                  ; GO IF COLOR CARD IS PRIMARY DISPLAY

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

0A28  24 30                           AND    AL,30H                    ; INSURE MONO IS PRIMARY
0A2A  3C 30                           CMP    AL,30H                    ; CONFIG OK?
0A2C  75 18                           JNZ    BAD_MOS                   ; GO IF NOT
0A2E  8A C4                           MOV    AL,AH                     ; RESTORE CONFIGURATION
0A30  EB 09 90                        JMP    MOS_OK                    ; USE THE CONFIG BYTE FOR CRT

                             ;-------- COLOR CARD

0A33                         MOS_OK_2:
0A33  24 30                           AND    AL,30H                    ; STRIP UNWANTED BITS
0A35  3C 30                           CMP    AL,30H                    ; MUST NOT BE MONO WITH JUMPER INSTALLED
0A37  8A C4                           MOV    AL,AH                     ; RESTORE CONFIGURATION
0A39  74 0B                           JZ     BAD_MOS                   ; GO IF YES

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

0A3B  A8 01                  MOS_OK:  TEST   AL,01H                    ; MUST HAVE AT LEAST ON DISKETTE
0A3D  75 33                           JNZ    NORMAL_CONFIG             ; GO SET CONFIGURATION IF OK
0A3F  F6 06 0012 R 20                 TEST   MFG_TST,MFG_JMP           ; EXCEPT IF MFG JUMPER IS INSTALLED
0A44  74 2C                           JZ     NORMAL_CONFIG             ; GO IF INSTALLED
```

SECTION 5

```
                                 ;-------- MINIMUM CONFIG WITH BAD CMOS OR NON VALID VIDEO

0A46                             BAD_MOS:
0A46  B0 8E                              MOV     AL,DIAG_STATUS          ; GET THE DIAGNOSTIC STATUS
0A48  E6 70                              OUT     CMOS_PORT,AL            ;
0A4A  EB 00                              JMP     SHORT $+2               ;
0A4C  E4 71                              IN      AL,CMOS_PORT+1          ;
0A4E  A8 C0                              TEST    AL,0C0H                 ; WAS THE BATTERY DEFECTIVE OR BAD CKSUM
0A50  75 0E                              JNZ     BAD_MOS1                ; GO IF YES
0A52  86 C4                              XCHG    AL,AH                   ; SAVE THE STATUS
0A54  B0 8E                              MOV     AL,DIAG_STATUS          ; CHECK CMOS GOOD
0A56  E6 70                              OUT     CMOS_PORT,AL            ;
0A58  EB 00                              JMP     SHORT $+2
0A5A  86 C4                              XCHG    AL,AH                   ; RESTORE THE STATUS
0A5C  0C 20                              OR      AL,20H                  ; SET THE MIN CONFG FLAG

0A5E  E6 71                              OUT     CMOS_PORT+1,AL          ; STORE THE STATUS
0A60                             BAD_MOS1:
0A60  E8 09FB R                          CALL    CHK_VIDEO               ; CHECK FOR VIDEO ROM
0A63  B0 01                              MOV     AL,01H                  ; DISKETTE ONLY
0A65  74 0B                              JZ      NORMAL_CONFIG           ; GO IF VIDEO ROM PRESENT

0A67  F6 06 0012 R 40                    TEST    MFG_TST,DSP_JMP         ; CHECK FOR DISPLAY JUMPER
0A6C  B0 11                              MOV     AL,11H                  ; DEFAULT TO 40X25 COLOR
0A6E  74 02                              JZ      NORMAL_CONFIG           ; GO IF JUMPER IS INSTALLED

0A70  B0 31                              MOV     AL,31H                  ; DISKETTE / BW CRT 80X25
                                 ;-----------------------------------------------------
                                 ;       CONFIGURATION AND MFG. MODE            :
                                 ;-----------------------------------------------------
0A72                             NORMAL_CONFIG:

0A72  F6 06 0012 R 20                    TEST    MFG_TST,MFG_JMP         ; IS THE MANUFACTURING JUMPER INSTALLED
0A77  75 02                              JNZ     NORM1                   ; GO IF NOT
0A79  24 3E                              AND     AL,03EH                 ; STRIP DISKETTE FOR MFG TEST

0A7B  2A E4                      NORM1:  SUB     AH,AH                   ;
0A7D  A3 0010 R                          MOV     EQUIP_FLAG,AX           ; SAVE SWITCH INFO
0A80  81 3E 0072 R 1234                  CMP     RESET_FLAG,1234H        ; BYPASS IF SOFT RESET
0A86  74 2C                              JZ      E6                      ;

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

0A88  B0 60                              MOV     AL,60H                  ; ENABLE KEYBOARD
0A8A  E8 0405 R                          CALL    C8042                   ; ISSUE WRITE BYTE COMMNAD
0A8D  B0 4D                              MOV     AL,4DH                  ; ENABLE OUT BUFF FULL INT
                                                                         ; SYS FLAG - PC 1 COMP - INH OVERRIDE
                                                                         ; ENABLE KEYBOARD
0A8F  E6 60                              OUT     PORT_A,AL               ;

0A91  2B C9                              SUB     CX,CX                   ; WAIT FOR COMMAND ACCEPTED
0A93  E8 040A R                          CALL    C42_1                   ;

0A96  B9 7FFF                            MOV     CX,07FFFH               ; SET LOOP COUNT FOR APPROX 100 MS
                                                                         ;  TO RESPOND
0A99  E4 64                      TST6:   IN      AL,STATUS_PORT          ; WAIT FOR OUTPUT BUFF FULL
0A9B  A8 01                              TEST    AL,OUT_BUF_FULL         ;
0A9D  E1 FA                              LOOPZ   TST6                    ; TRY AGAIN IF NOT

0A9F  9C                                 PUSHF                           ; SAVE FLAGS
0AA0  B0 AD                              MOV     AL,DIS_KBD              ; DISABLE KEYBOARD
0AA2  E8 0405 R                          CALL    C8042                   ; ISSUE THE COMMAND
0AA5  9D                                 POPF                            ; RESTORE FLAGS
0AA6  74 0C                              JZ      E6                      ; CONTINUE WITHOUT RESULTS
0AA8  E4 60                              IN      AL,PORT_A               ; GET INPUT FROM KEY BOARD
0AAA  A2 0072 R                          MOV     BYTE PTR RESET_FLAG,AL  ; TEMP SAVE FOR AA RECIEVED

                                 ;-------- CHECK FOR MFG REQUEST

0AAD  3C 65                              CMP     AL,065H                 ; LOAD MFG. TEST REQUEST?
0AAF  75 03                              JNE     E6
0AB1  E9 002C R                          JMP     MFG_BOOT                ; GO TO BOOTSTRAP IF SO
                                 ;----------------------------------------------------------
                                 ; 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 STG. CHECK STG         :
                                 ;       ADDRESSABILITY.                                    :
                                 ; ERROR = 1 LONG AND 2 SHORT BEEPS                         :
                                 ;----------------------------------------------------------

0AB4                             E6:
0AB4  A1 0010 R                          MOV     AX,EQUIP_FLAG           ; GET SENSE INFO
0AB7  50                                 PUSH    AX                      ; SAVE IT
0AB8  B0 30                              MOV     AL,30H
0ABA  A3 0010 R                          MOV     EQUIP_FLAG,AX
0ABD  2A E4                              SUB     AH,AH
0ABF  CD 10                              INT     INT_VIDEO               ; SEND INIT TO B/W CARD
0AC1  B0 20                              MOV     AL,20H
0AC3  A3 0010 R                          MOV     EQUIP_FLAG,AX
0AC6  2A E4                              SUB     AH,AH                   ; AND INIT COLOR CARD
0AC8  CD 10                              INT     INT_VIDEO
0ACA  B8 0001                            MOV     AX,0001H                ; SET COLOR 40X25 MODE
0ACD  CD 10                              INT     INT_VIDEO               ;
0ACF  58                                 POP     AX                      ; RECOVER REAL SWITCH INFO
0AD0  A3 0010 R                          MOV     EQUIP_FLAG,AX           ; RESTORE IT
0AD3  24 30                              AND     AL,30H                  ; ISOLATE VIDEO SWS
0AD5  75 12                              JNZ     E7                      ; VIDEO SWS SET TO 0?
0AD7  1E                                 PUSH    DS                      ; SAVE THE DATA SEGMENT
0AD8  50                                 PUSH    AX
0AD9  2B C0                              SUB     AX,AX                   ; SET DATA SEGMENT TO 0
0ADB  8E D8                              MOV     DS,AX
0ADD  BF 0040 R                          MOV     DI,OFFSET VIDEO_INT     ; SET INT 10H TO DUMMY
0AE0  C7 05 0000 E                       MOV     WORD PTR [DI],OFFSET DUMMY_RETURN   ; RETURN IF NO VIDEO CARD
0AE4  58                                 POP     AX                      ; RESTORE REGISTERS
0AE5  1F                                 POP     DS
0AE6  E9 0B68 R                          JMP     E18_1                   ; BYPASS VIDEO TEST
0AE9                             E7:                                     ; TEST_VIDEO:
0AE9  3C 30                              CMP     AL,30H                  ; B/W CARD ATTACHED?
0AEB  74 08                              JE      E8                      ; YES - SET MODE FOR B/W CARD
0AED  FE C4                              INC     AH                      ; SET COLOR MODE FOR COLOR CD
0AEF  3C 20                              CMP     AL,20H                  ; 80X25 MODE SELECTED?
0AF1  75 02                              JNE     E8                      ; NO - SET MODE FOR 40X25
0AF3  B4 03                              MOV     AH,3                    ; SET MODE FOR 80X25
0AF5  86 E0                      E8:     XCHG    AH,AL                   ; SET_MODE:
0AF7  50                                 PUSH    AX                      ; SAVE VIDEO MODE ON STACK
0AF8  2A E4                              SUB     AH,AH                   ; INITIALIZE TO ALPHANUMERIC MD
0AFA  CD 10                              INT     INT_VIDEO               ; CALL VIDEO_IO
0AFC  58                                 POP     AX                      ; RESTORE VIDEO SENSE SWS IN AH
0AFD  50                                 PUSH    AX                      ; RESAVE VALUE
0AFE  BB B000                            MOV     BX,0B000H               ; BEG VIDEO RAM ADDR B/W CD
0B01  BA 03B8                            MOV     DX,3B8H                 ; MODE REG FOR B/W
```

```
0B04  B9 0800                      MOV   CX,2048           ; RAM WORD CNT FOR B/W CD
0B07  B0 01                        MOV   AL,1              ; SET MODE FOR BW CARD
0B09  80 FC 30                     CMP   AH,30H            ; B/W VIDEO CARD ATTACHED?
0B0C  74 09                        JE    E9                ; YES - GO TEST VIDEO STG
0B0E  B7 B8                        MOV   BH,0B8H           ; BEG VIDEO RAM ADDR COLOR CD
0B10  BA 03D8                      MOV   DX,3D8H           ; MODE REG FOR COLOR CD
0B13  B5 20                        MOV   CH,20H            ; RAM WORD CNT FOR COLOR CD
0B15  FE C8                        DEC   AL                ; SET MODE TO 0 FOR COLOR CD
0B17                         E9:                           ; TEST_VIDEO_STG:
0B17  EE                           OUT   DX,AL             ; DISABLE VIDEO FOR COLOR CD
0B18  8E C3                        MOV   ES,BX             ; POINT ES TO VIDEO RAM
0B1A  8E DB                        MOV   DS,BX             ; POINT DS TO VIDEO RAM
0B1C  D1 C9                        ROR   CX,1              ; DIVIDE BY 2 FOR WORD COUNT
0B1E  E8 0000 E                    CALL  STGTST_CNT        ; GO TEST VIDEO R/W STG
0B21  75 6F                        JNE   E17               ; R/W STG FAILURE - BEEP SPK
                             ;----------------------------------------------
                             ; TEST.15                                      :
                             ;        SETUP VIDEO DATA ON SCREEN FOR VIDEO  :
                             ;        LINE TEST.                            :
                             ; DESCRIPTION                                  :
                             ;        ENABLE VIDEO SIGNAL AND SET MODE.     :
                             ;        DISPLAY A HORIZONTAL BAR ON SCREEN.   :
                             ;----------------------------------------------
0B23                         E10:
0B23  B0 22                        MOV   AL,22H            ; <><><><><><><><><><><><>
0B25  E6 80                        OUT   MFG_PORT,AL       ; <><><>CHECKPOINT 22 <><>

0B27  58                           POP   AX                ; GET VIDEO SENSE SWS (AH)
0B28  50                           PUSH  AX                ; SAVE IT
0B29  B4 00                        MOV   AH,0              ; ENABLE VIDEO AND SET MODE
0B2B  CD 10                        INT   INT_VIDEO         ; VIDEO
0B2D  B8 7020                      MOV   AX,7020H          ; WRT BLANKS IN REVERSE VIDEO
0B30  2B FF                        SUB   DI,DI             ; SETUP STARTING LOC
0B32  B9 0028                      MOV   CX,40             ; NO. OF BLANKS TO DISPLAY
0B35  F3/ AB                       REP   STOSW             ; WRITE VIDEO STORAGE
                             ;----------------------------------------
                             ; TEST.16                                :
                             ;        CRT INTERFACE LINES TEST        :
                             ; DESCRIPTION                            :
                             ;        SENSE ON/OFF TRANSITION OF THE  :
                             ;        VIDEO ENABLE AND HORIZONTAL     :
                             ;        SYNC LINES.                     :
                             ;----------------------------------------
0B37  58                           POP   AX                ; GET VIDEO SENSE SW INFO
0B38  50                           PUSH  AX                ; SAVE IT
0B39  80 FC 30                     CMP   AH,30H            ; B/W CARD ATTACHED?
0B3C  BA 03BA                      MOV   DX,03BAH          ; SETUP ADDR OF BW STATUS PORT
0B3F  74 03                        JE    E11               ; YES - GO TEST LINES
0B41  BA 03DA                      MOV   DX,03DAH          ; COLOR CARD IS ATTACHED
0B44                         E11:                          ; LINE_TST:
0B44  B4 08                        MOV   AH,8
0B46                         E12:                          ; OFLOOP_CNT:
0B46  2B C9                        SUB   CX,CX
0B48  EC                     E13:  IN    AL,DX             ; READ CRT STATUS PORT
0B49  22 C4                        AND   AL,AH             ; CHECK VIDEO/HORZ LINE
0B4B  75 04                        JNZ   E14               ; ITS ON - CHECK IF IT GOES OFF
0B4D  E2 F9                        LOOP  E13               ; LOOP TILL ON OR TIMEOUT
0B4F  EB 41                        JMP   SHORT E17         ; GO PRINT ERROR MSG
0B51  2B C9                  E14:  SUB   CX,CX
0B53  EC                     E15:  IN    AL,DX             ; READ CRT STATUS PORT
0B54  22 C4                        AND   AL,AH             ; CHECK VIDEO/HORZ LINE
0B56  74 05                        JZ    E16               ; ITS ON - CHECK NEXT LINE
0B58  E2 F9                        LOOP  E15               ; LOOP IF ON TILL IT GOES OFF
0B5A  EB 36 90                     JMP   E17               ; GO ERROR BEEP

                             ;------- CHECK HORIZONTAL LINE

0B5D  B1 03                  E16:  MOV   CL,3              ; GET NEXT BIT TO CHECK
0B5F  D2 EC                        SHR   AH,CL
0B61  75 E3                        JNZ   E12               ; CONTINUE
0B63                         E18:                          ; DISPLAY_CURSOR:
0B63  58                           POP   AX                ; GET VIDEO SENSE SWS (AH)
0B64  B4 00                        MOV   AH,0              ; SET MODE AND DISPLAY CURSOR
0B66  CD 10                        INT   INT_VIDEO         ; CALL VIDEO I/O PROCEDURE

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

0B68  BA C000                E18_1: MOV  DX,0C000H         ; SET THE LOW SEGMENT VALUE
0B6B  B0 23                  E18A: MOV   AL,23H            ; <><><><><><><><><><><><>
0B6D  E6 80                        OUT   MFG_PORT,AL       ; <><><>CHECKPOINT 23 <><>
0B6F  8E DA                        MOV   DS,DX
0B71  2B DB                        SUB   BX,BX
0B73  8B 07                        MOV   AX,[BX]           ; GET FIRST 2 LOCATIONS
0B75  53                           PUSH  BX
0B76  5B                           POP   BX                ; LET BUS SETTLE
0B77  3D AA55                      CMP   AX,0AA55H         ; PRESENT?
0B7A  75 05                        JNZ   E18B              ; NO? GO LOOK FOR OTHER MODULES
0B7C  E8 0000 E                    CALL  ROM_CHECK         ; GO SCAN MODULE
0B7F  EB 04                        JMP   SHORT E18C
0B81  81 C2 0080             E18B: ADD   DX,0080H          ; POINT TO NEXT 2K BLOCK
0B85  81 FA C800             E18C: CMP   DX,0C800H         ; TOP OF VIDEO ROM AREA YET?
0B89  7C E0                        JL    E18A              ; GO SCAN FOR ANOTHER MODULE
0B8B  B0 24                        MOV   AL,24H            ; <><><><><><><><><><><><><><>
0B8D  E6 80                        OUT   MFG_PORT,AL       ; <><><>CHECKPOINT 24 <><><><>
0B8F  E9 0000 E                    JMP   POST2             ; GO TO NEXT TEST

                             ;-------- CRT ERROR SET MFG CKPT AND ERR BEEP

0B92  E8 0000 E              E17:  CALL  DDS               ; POINT TO DATA

                             ;-------- CHECKPOINT 0C = MONO FAILED

0B95  C6 06 0015 R 0C              MOV   MFG_ERR_FLAG,0CH     ;<><><><>CRT ERR CHKPT. 0C<><>
0B9A  80 3E 0072 R 64              CMP   BYTE PTR RESET_FLAG,064H ; IS THIS A MFG REQUEST?
0B9F  74 0D                        JZ    E19               ; BY PASS ERROR BEEP IF YES
0BA1  F6 06 0012 R 20              TEST  MFG_TST,MFG_JMP   ; IS THE MFG LOOP JUMPER INSTALLED?
0BA6  74 06                        JZ    E19               ; BY PASS ERROR BEEP IF YES
0BA8  BA 0102                      MOV   DX,102H
0BAB  E8 0000 E                    CALL  ERR_BEEP          ; GO BEEP SPEAKER
0BAE  1E                     E19:  PUSH  DS
0BAF  A1 0010 R                    MOV   AX,EQUIP_FLAG     ; GET THE CURRENT VIDEO
0BB2  24 30                        AND   AL,30H            ; STRIP OTHER BITS
0BB4  3C 30                        CMP   AL,30H            ; IS IT MONO?
0BB6  74 31                        JZ    TRY_COLOR         ; GO IF YES
                             ;------- COLOR FAILED TRY MONO

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

0BB8  C6 06 0015 R 0D              MOV   MFG_ERR_FLAG,0DH     ;<><><><>CRT ERR CHKPT. 0D<><>

0BBD  BA 03B8                      MOV   DX,3B8H           ; DISABLE B/W
0BC0  B0 01                        MOV   AL,1              ;
0BC2  EE                           OUT   DX,AL             ; OUTPUT THE DISABLE
0BC3  BB B000                      MOV   BX,0B000H         ; CHECK FOR MONO VIDEO RAM
0BC6  8E DB                        MOV   DS,BX             ;
0BC8  B8 AA55                      MOV   AX,0AA55H         ; WRITE AN AA55
```

SECTION 5

```
0BCB  2B DB                       SUB    BX,BX                 ;  TO THE FIRST LOCATION
0BCD  89 07                       MOV    [BX],AX               ;
0BCF  EB 00                       JMP    SHORT $+2             ; ALLOW BUS TO SETTLE
0BD1  8B 07                       MOV    AX,[BX]               ; READ THE FIRST LOCATION
0BD3  3D AA55                     CMP    AX,0AA55H             ; IS THE MONO VIDEO CARD THERE?
0BD6  1F                          POP    DS                    ; RESTORE THE DATA SEGMENT
0BD7  75 56                       JNZ    E17_3                 ; GO IF NOT
0BD9  81 0E 0010 R 0030           OR     EQUIP_FLAG,30H        ; TURN ON MONO BITS IN EQUIP FLAG
0BDF  A1 0010 R                   MOV    AX,EQUIP_FLAG         ; ENABLE VIDEO
0BE2  2A E4                       SUB    AH,AH                 ;
0BE4  CD 10                       INT    INT_VIDEO             ;
0BE6  EB 35 90                    JMP    E17_1                 ; CONTINUE

                          ;-------- MONO FAILED TRY COLOR

0BE9                      TRY_COLOR:
0BE9  B0 01                       MOV    AL,01H                ; SET MODE COLOR 40X25
0BEB  2A E4                       SUB    AH,AH                 ;
0BED  CD 10                       INT    INT_VIDEO             ;
0BEF  BA 03D8                     MOV    DX,3D8H               ; DISABLE COLOR
0BF2  B0 00                       MOV    AL,0                  ;
0BF4  EE                          OUT    DX,AL                 ; OUTPUT THE DISABLE
0BF5  BB B800                     MOV    BX,0B800H             ; CHECK FOR COLOR VIDEO RAM
0BF8  8E DB                       MOV    DS,BX                 ;
0BFA  B8 AA55                     MOV    AX,0AA55H             ; WRITE AN AA55
0BFD  2B DB                       SUB    BX,BX                 ;  TO THE FIRST LOCATION
0BFF  89 07                       MOV    [BX],AX               ;
0C01  EB 00                       JMP    SHORT $+2             ; ALLOW BUS TO SETTLE
0C03  8B 07                       MOV    AX,[BX]               ; READ THE FIRST LOCATION
0C05  3D AA55                     CMP    AX,0AA55H             ; IS THE COLOR VIDEO CARD THERE?
0C08  1F                          POP    DS                    ; RESTORE THE DATA SEGMENT
0C09  75 24                       JNZ    E17_3                 ; GO IF NOT
0C0B  81 26 0010 R FFCF           AND    EQUIP_FLAG,0FFCFH     ; TURN OFF VIDEO BITS
0C11  81 0E 0010 R 0010           OR     EQUIP_FLAG,10H        ; SET COLOR 40X24
0C17  B0 01                       MOV    AL,01H                ;
0C19  2A E4                       SUB    AH,AH                 ;
0C1B  CD 10                       INT    INT_VIDEO             ;
0C1D                      E17_1:
0C1D  58                          POP    AX                    ; SET NEW VIDEO TYPE ON STACK
0C1E  A1 0010 R                   MOV    AX,EQUIP_FLAG         ;
0C21  24 30                       AND    AL,30H                ;
0C23  3C 30                       CMP    AL,30H                ; IS IT THE B/W?
0C25  2A C0                       SUB    AL,AL                 ;
0C27  74 02                       JZ     E17_2                 ; GO IF YES
0C29  FE C0                       INC    AL                    ; INIT FOR 40X25
0C2B  50                  E17_2:  PUSH   AX
0C2C  E9 0B63 R           E17_4:  JMP    E18

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

0C2F                      E17_3:
0C2F  1E                          PUSH   DS                    ;
0C30  2B C0                       SUB    AX,AX                 ; SET DS SEGMENT TO 0
0C32  8E D8                       MOV    DS,AX
0C34  BF 0040 R                   MOV    DI,OFFSET VIDEO_INT   ; SET INT 10H TO DUMMY
0C37  C7 05 0000 E                MOV    WORD PTR [DI],OFFSET DUMMY_RETURN  ; RETURN IF NO VIDEO CARD
0C3B  1F                          POP    DS
0C3C  E9 0B68 R                   JMP    E18_1                 ; BYPASS REST OF VIDEO TEST
0C3F                      POST1   ENDP
0C3F                      CODE    ENDS
                                  END
```

```
                                 TITLE 01/03/84 TEST2 POWER ON SELF TEST
                                 .LIST
                                 PUBLIC  C21
                                 PUBLIC  SHUT2
                                 PUBLIC  SHUT3
                                 PUBLIC  SHUT4
                                 PUBLIC  SHUT6
                                 PUBLIC  SHUT7
                                 PUBLIC  POST2

                               C INCLUDE SEGMENT.SRC
 0000                          C CODE SEGMENT BYTE PUBLIC
                               C

                                 EXTRN   H5:NEAR
                                 EXTRN   POST7:NEAR
                                 EXTRN   SET_TOD:NEAR
                                 EXTRN   E0:NEAR                          ; 101 ERROR CODE
                                 EXTRN   E0_A:NEAR                        ; 102 ERROR CODE
                                 EXTRN   E0_B:NEAR                        ; 103 ERROR CODE

                                 EXTRN   VIR_ERR:NEAR                     ; 104 ERROR CODE
                                 EXTRN   CM4:NEAR                         ; 105 ERROR CODE
                                 EXTRN   CM4_A:NEAR                       ; 106 ERROR CODE
                                 EXTRN   CM4_B:NEAR                       ; 107 ERROR CODE
                                 EXTRN   CM4_C:NEAR                       ; 108 ERROR CODE
                                 EXTRN   CM4_D:NEAR                       ; 109 ERROR CODE
                                 EXTRN   CM1:NEAR                         ; 161 ERROR CODE
                                 EXTRN   CM2:NEAR                         ; 162 ERROR CODE
                                 EXTRN   CM3:NEAR                         ; 163 ERROR CODE
                                 EXTRN   E1_A:NEAR                        ; 164 ERROR CODE

                                 EXTRN   E1:NEAR                          ; 201 ERROR CODE
                                 EXTRN   ADERR1:NEAR                      ; 202 ERROR CODE
                                 EXTRN   ADERR:NEAR                       ; 203 ERROR CODE


                                 EXTRN   F1:NEAR                          ; 301 ERROR CODE
                                 EXTRN   LOCK:NEAR                        ; 302 ERROR CODE
                                 EXTRN   F1_A:NEAR                        ; 303 ERROR CODE
                                 EXTRN   F1_B:NEAR                        ; 304 ERROR CODE

                                 EXTRN   E1_B:NEAR                        ; 401 ERROR CODE
                                 EXTRN   E1_C:NEAR                        ; 501 ERROR CODE

                                 EXTRN   F3:NEAR                          ; 601 ERROR CODE

                                 EXTRN   KBD_RESET:NEAR
                                 EXTRN   GATE_A20:NEAR

                                 EXTRN   E_MSG:NEAR
                                 EXTRN   XPC_BYTE:NEAR
                                 EXTRN   VECTOR_TABLE:NEAR
                                 EXTRN   SLAVE_VECTOR_TABLE:NEAR
                                 EXTRN   NMI_INT:NEAR
                                 EXTRN   PRINT_SCREEN:NEAR
                                 EXTRN   BLINK_INT:NEAR
                                 EXTRN   PRT_HEX:NEAR
                                 EXTRN   F3B:NEAR
                                 EXTRN   PRT_SEG:NEAR
                                 EXTRN   XPC_BYTE:NEAR

                                 EXTRN   ROM_CHECK:NEAR
                                 EXTRN   ROS_CHECKSUM:NEAR
                                 EXTRN   SEEK:NEAR
                                 EXTRN   ERR_BEEP:NEAR
                                 EXTRN   P_MSG:NEAR
                                 EXTRN   START_1:NEAR
                                 EXTRN   F4:NEAR
                                 EXTRN   F4E:NEAR
                                 EXTRN   F3A:NEAR
                                 EXTRN   DISK_BASE:NEAR
                                 EXTRN   F3D:NEAR
                                 EXTRN   F3D1:NEAR
                                 EXTRN   PROC_SHUTDOWN:NEAR
                                 EXTRN   SYSINIT1:NEAR
                                 EXTRN   PROT_PRT_HEX:NEAR
                                 EXTRN   DISK_IO:NEAR
                                 EXTRN   HD_INT:NEAR
                                 EXTRN   C8042:NEAR
                                 EXTRN   OBF_42:NEAR
                                 EXTRN   STGTST_CNT:NEAR
                                 EXTRN   BOOT_STRAP_1:NEAR
                                 EXTRN   XMIT_8042:NEAR
                                 EXTRN   ROM_ERR:NEAR
                                 EXTRN   DDS:NEAR
                                 EXTRN   DISK_SETUP:NEAR
                                 EXTRN   DSKETTE_SETUP:NEAR
                                 ;---------------------------------------------------------
                                 ; TEST.17                                                 :
                                 ;       8259 INTERRUPT CONTROLLER TEST                    :
                                 ; DESCRIPTION                                             :
                                 ;       READ/WRITE THE INTERRUPT MASK REGISTER (IMR)      :
                                 ;       WITH ALL ONES AND ZEROES. ENABLE SYSTEM           :
                                 ;       INTERRUPTS.  MASK DEVICE INTERRUPTS OFF. CHECK    :
                                 ;       FOR HOT INTERRUPTS (UNEXPECTED).                  :
                                 ;---------------------------------------------------------

                                         ASSUME  CS:CODE
                                         ASSUME  DS:DATA

 0000                            POST2   PROC    NEAR

 0000  B0 0A                     C21:    MOV     AL,10                    ; LINE FEED ON CRT
 0002  E8 0000 E                         CALL    PRT_HEX                  ;
 0005  E8 0000 E                         CALL    DDS                      ;SET DATA SEGMENT

                                 ;----- TEST THE IMR REGISTERS

 0008  FA                        C21A:   CLI                              ; TURN OFF INTERRUPTS
 0009  B0 00                             MOV     AL,0                     ; SET IMR TO ZERO
 000B  E6 21                             OUT     INTA01,AL
 000D  E6 A1                             OUT     INTB01,AL                ; SEND TO 2ND INT
 000F  EB 00                             JMP     SHORT $+2                ;
 0011  E4 21                             IN      AL,INTA01                ; READ IMR
 0013  8A E0                             MOV     AH,AL                    ; SAVE RESULTS
 0015  E4 A1                             IN      AL,INTB01                ; READ 2ND IMR

 0017  0A E0                             OR      AH,AL                    ; BOTH IMR = 0?
 0019  75 2C                             JNZ     D6                       ; GO TO ERR ROUTINE IF NOT 0

 001B  B0 25                             MOV     AL,25H                   ;<><><><><><><><><><><><>
 001D  E6 80                             OUT     MFG_PORT,AL              ;<><><>CHECKPOINT 25 <><><><>

 001F  B0 FF                             MOV     AL,0FFH                  ; DISABLE DEVICE INTERRUPTS
 0021  E6 21                             OUT     INTA01,AL                ; WRITE TO IMR
```

SECTION 5

```
0023  E6 A1                     OUT     INTB01,AL               ; WRITE TO 2ND IMR
0025  EB 00                     JMP     SHORT $+2               ; IO DELAY
0027  E4 21                     IN      AL,INTA01               ; READ IMR
0029  8A E0                     MOV     AH,AL                   ; SAVE RESULTS
002B  E4 A1                     IN      AL,INTB01               ; READ 2ND IMR

002D  05 0001                   ADD     AX,1                    ; ALL IMR BIT ON?
0030  75 15                     JNZ     D6                      ; NO - GO TO ERR ROUTINE

                        ;----- CHECK FOR HOT INTERRUPTS

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

0032  A2 006B R                 MOV     INTR_FLAG,AL            ; CLEAR INTERRUPT FLAG

0035  B0 26                     MOV     AL,26H                  ;<><><><><><><><><><><><>
0037  E6 80                     OUT     MFG_PORT,AL             ;<><><>CHECKPOINT 26 <><><>

0039  FB                        STI                             ; ENABLE EXTERNAL INTERRUPTS
003A  2B C9                     SUB     CX,CX                   ; WAIT 1 SEC FOR ANY INTRS THAT
003C  E2 FE             D4:     LOOP    D4                      ; MIGHT OCCUR
003E  E2 FE             D5:     LOOP    D5
0040  80 3E 006B R 00           CMP     INTR_FLAG,00H           ; DID ANY INTERRUPTS OCCUR?
0045  74 0D                     JZ      D7                      ; NO - GO TO NEXT TEST

0047  C6 06 0015 R 05   D6:     MOV     MFG_ERR_FLAG,05H        ; <><><><><><><><><><><><>
                                                                ; <><>CHECKPOINT 5<><><><><>
004C  BE 0000 E                 MOV     SI,OFFSET E0            ; DISPLAY 101 ERROR
004F  E8 0000 E         D6A:    CALL    E_MSG
0052  FA                        CLI
0053  F4                        HLT                             ; HALT THE SYSTEM

                        ;-------CHECK THE CONVERTING LOGIC

0054  B0 27             D7:     MOV     AL,27H                  ;<><><><><><><><><><><><>
0056  E6 80                     OUT     MFG_PORT,AL             ;<><><>CHECKPOINT 27 <><><>

0058  B8 AA55                   MOV     AX,0AA55H               ;
005B  E7 82                     OUT     MFG_PORT+2,AX           ; WRITE A WORD
005D  E4 82                     IN      AL,MFG_PORT+2           ;  GET THE FIRST BYTE
005F  86 C4                     XCHG    AL,AH                   ;  SAVE IT
0061  EB 00                     JMP     SHORT $+2               ; IO DELAY
0063  E4 83                     IN      AL,MFG_PORT+3           ;  GET THE SECOND BYTE
0065  3D 55AA                   CMP     AX,55AAH                ; IS IT OK?
0068  74 05                     JZ      D7_A                    ; GO IF YES

006A  BE 0000 E                 MOV     SI,OFFSET CM4_A         ; DISPLAY 106 ERROR
006D  EB E0                     JMP     D6A                     ;

                        ;------- CHECK FOR HOT NMI INTERRUPTS WITHOUT IO/RAM PARITY ENABLED

006F                    D7_A:
006F  2A C0                     SUB     AL,AL                   ; SET FLAG TO ZERO
0071  E6 80                     OUT     MFG_PORT,AL             ; SAVE IT

0073  B0 0F                     MOV     AL,0FH                  ; TURN ON NMI
0075  E6 70                     OUT     CMOS_PORT,AL            ;
0077  B9 00FF                   MOV     CX,00FFH                ; DELAY
007A  E2 FE             D7_B:   LOOP    D7_B                    ;
007C  B0 8F                     MOV     AL,8FH                  ; TURN OFF NMI
007E  E6 70                     OUT     CMOS_PORT,AL            ;
0080  E4 80                     IN      AL,MFG_PORT             ; ANY NMI?
0082  0A C0                     OR      AL,AL                   ;
0084  74 09                     JZ      D7_C                    ; CONTINUE IF NOT

0086  B0 28                     MOV     AL,28H                  ;<><><><><><><><><><><><>
0088  E6 80                     OUT     MFG_PORT,AL             ;<><><>CHECKPOINT 28 <><><>

008A  BE 0000 E                 MOV     SI,OFFSET CM4_B         ; DISPLAY 107 ERROR
008D  EB C0                     JMP     D6A                     ;

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

008F  B0 29             D7_C:   MOV     AL,29H                  ;<><><><><><><><><><><><>
0091  E6 80                     OUT     MFG_PORT,AL             ;<><><>CHECKPOINT 29 <><><>
0093  E4 61                     IN      AL,PORT_B                       ; GET CURRENT SETTING OF PORT
0095  8A E0                     MOV     AH,AL                           ; SAVE THAT SETTING
0097  EB 00                     JMP     SHORT $+2                       ; IO DELAY
0099  24 FC                     AND     AL,0FCH                         ; INSURE SPEAKER OFF
009B  E6 61                     OUT     PORT_B,AL

009D  B0 B0                     MOV     AL,10110000B                    ; SEL TIM 2,LSB,MSB,BINARY,MODE 0
009F  E6 43                     OUT     TIMER+3,AL                      ; WRITE THE TIMER MODE REG
00A1  EB 00                     JMP     SHORT $+2                       ; IO DELAY
00A3  B8 AA55                   MOV     AX,0AA55H                       ; WRITE AN AA55
00A6  E6 42                     OUT     TIMER+2,AL                      ; WRITE TIMER 2 CNT - LSB
00A8  EB 00                     JMP     SHORT $+2                       ; IO DELAY
00AA  8A C4                     MOV     AL,AH
00AC  E6 42                     OUT     TIMER+2,AL                      ; WRITE TIMER 2 CNT - MSB

00AE  EB 00                     JMP     SHORT $+2                       ; IO DELAY
00B0  E4 42                     IN      AL,TIMER+2                      ; GET THE LSB
00B2  86 E0                     XCHG    AH,AL                           ; SAVE IT

00B4  EB 00                     JMP     SHORT $+2                       ; IO DELAY
00B6  E4 42                     IN      AL,TIMER+2                      ; GET THE MSB
00B8  3D 55AA                   CMP     AX,055AAH                       ; BUS OK?
00BB  74 05                     JZ      D7_D                            ; GO IF OK

00BD  BE 0000 E                 MOV     SI,OFFSET CM4_C         ; DISPLAY 108 ERROR
00C0  EB 8D                     JMP     D6A                     ;

                        ;--------------------------------------------------------
                        ; TEST.18                                               :
                        ;       8253 TIMER CHECKOUT                             :
                        ; DESCRIPTION                                           :
                        ;       VERIFY THAT THE SYSTEM TIMER (0) DOESN'T COUNT  :
                        ;       TOO FAST OR TOO SLOW.                           :
                        ;--------------------------------------------------------

00C2  B0 2A             D7_D:   MOV     AL,2AH                  ;<><><><><><><><><><><><>
00C4  E6 80                     OUT     MFG_PORT,AL             ;<><><>CHECKPOINT 2A <><><>
00C6  FA                        CLI                             ;
00C7  B0 FE                     MOV     AL,0FEH                 ; MASK ALL INTRS EXCEPT LVL 0
00C9  E6 21                     OUT     INTA01,AL               ; WRITE THE 8259 IMR
00CB  B0 10                     MOV     AL,00010000B            ; SEL TIM 0, LSB, MODE 0, BINARY
00CD  E6 43                     OUT     TIM_CTL,AL              ; WRITE TIMER CONTROL MODE REG
00CF  B9 002C                   MOV     CX,16H*2                ; SET PGM LOOP CNT

00D2  EB 00                     JMP     SHORT $+2               ; IO DELAY
00D4  8A C1                     MOV     AL,CL                   ; SET TIMER 0 CNT REG
00D6  E6 40                     OUT     TIMER0,AL               ; WRITE TIMER 0 CNT REG
00D8  FB                        STI                             ;
00D9  F6 06 006B R 01   D8:     TEST    INTR_FLAG,01H
```

```
                                                                ; DID TIMER 0 INTERRUPT OCCUR?
00DE  75 0D                      JNZ     D9                     ; YES - CHECK TIMER OP FOR SLOW TIME
00E0  E2 F7                      LOOP    D8                     ; WAIT FOR INTR FOR SPECIFIED TIME

00E2  C6 06 0015 R 02            MOV     MFG_ERR_FLAG,02H       ;<><><><><><><><><><><><><><>
                                                                ;<><>TIMER CHECKPOINT (2)<>

00E7  BE 0000 E          D8_A:   MOV     SI,OFFSET E0_A         ; DISPLAY 102 ERROR
00EA  E9 004F R                  JMP     D6A                    ; TIMER 0 INTR DIDN'T OCCUR - ERR

00ED  B0 2B              D9:     MOV     AL,2BH                 ;<><><><><><><><><><><><><><>
00EF  E6 80                      OUT     MFG_PORT,AL            ;<><><>CHECKPOINT 2B <><><><>

00F1  FA                         CLI                            ;
00F2  B1 0C                      MOV     CL,12                  ; SET PGM LOOP CNT
00F4  B0 FF                      MOV     AL,0FFH                ; WRITE TIMER 0 CNT REG
00F6  E6 40                      OUT     TIMER0,AL
00F8  C6 06 006B R 00            MOV     INTR_FLAG,0            ; RESET INTR RECEIVED FLAG
00FD  B0 FE                      MOV     AL,0FEH                ; REENABLE TIMER 0 INTERRUTS
00FF  E6 21                      OUT     INTA01,AL
0101  FB                         STI                            ;
0102  F6 06 006B R 01    D10:    TEST    INTR_FLAG,01H          ; DID TIMER 0 INTERRUPT OCCUR?
0107  75 DE                      JNZ     D8_A                   ; YES - TIMER CNTING TOO FAST, ERR
0109  E2 F7                      LOOP    D10                    ; WAIT FOR INTR FOR SPECIFIED TIME

                         ;------- WAIT FOR INTERRUPT

010B  2B C9                      SUB     CX,CX                  ;

010D  B0 2C                      MOV     AL,2CH                 ;<><><><><><><><><><><><><><>
010F  E6 80                      OUT     MFG_PORT,AL            ;<><><>CHECKPOINT 2C <><><><>

0111  F6 06 006B R 01    D11:    TEST    INTR_FLAG,01H          ; DID TIMER 0 INTERRUPT OCCUR?
0116  75 08                      JNZ     D12                    ; GO IF YES
0118  E2 F7                      LOOP    D11                    ; TRY AGAIN

011A  BE 0000 E                  MOV     SI,OFFSET E0_B         ; DISPLAY 103 ERROR
011D  E9 004F R                  JMP     D6A                    ; ERROR IF NOT

                         ;------ SETUP TIMER 0 TO MODE 3

0120  FA                 D12:    CLI                            ;
0121  B0 FF                      MOV     AL,0FFH                ; DISABLE ALL DEVICE INTERRUPTS
0123  E6 21                      OUT     INTA01,AL
0125  B0 36                      MOV     AL,36H                 ; SEL TIM 0,LSB,MSB,MODE 3
0127  E6 43                      OUT     TIMER+3,AL             ; WRITE TIMER MODE REG
0129  EB 00                      JMP     SHORT $+2              ; IO DELAY
012B  B0 00                      MOV     AL,0
012D  E6 40                      OUT     TIMER,AL               ; WRITE LSB TO TIMER 0 REG
012F  EB 00                      JMP     SHORT $+2              ; IO DELAY
0131  E6 40                      OUT     TIMER,AL               ; WRITE MSB TO TIMER 0 REG
                                 ;==================================
                         ;------- CHECK 8042 FOR LAST COMMAND ACCEPTED
                                 ;==================================
0133  2B C9                      SUB     CX,CX                  ; SET WAIT TIME
0135  B0 2D                      MOV     AL,2DH                 ;<><><><><><><><><><><><><><>
0137  E6 80                      OUT     MFG_PORT,AL            ;<><><>CHECKPOINT 2D <><><><>
0139  E4 64              D13:    IN      AL,STATUS_PORT         ; GET THE 8042 STATUS
013B  A8 02                      TEST    AL,INPT_BUF_FULL       ; HAS THE LAST COMMAND BEEN ACCEPTED?
013D  74 08                      JZ      E19                    ; GO IF YES
013F  E2 F8                      LOOP    D13                    ; TRY AGAIN
                         ;-------- ERROR EXIT (MSG 105)

0141  BE 0000 E                  MOV     SI,OFFSET CM4          ; PRINT 105 ERROR
0144  E9 004F R                  JMP     D6A                    ; GO ERROR HALT

                         ;----------------------------------------------------
                         ; TEST.19                                            :
                         ;        ADDITIONAL READ/WRITE STORAGE TEST          :
                         ;        ++++ MUST RUN IN PROTECTED MODE ++++        :
                         ; DESCRIPTION                                        :
                         ;        WRITE/READ DATA PATTERNS TO ANY READ/WRITE  :
                         ;        STORAGE AFTER THE FIRST 64K.  STORAGE       :
                         ;        ADDRESSABILITY IS CHECKED.                  :
                         ;----------------------------------------------------
                                 ASSUME  DS:DATA
0147                     E19:
0147  E8 0000 E                  CALL    DDS                            ; SET DATA SEGMENT
014A  B0 2F                      MOV     AL,2FH                         ;<><><><><><><><><><><><><><>
014C  E6 80                      OUT     MFG_PORT,AL                    ;<><><>CHECKPOINT 2F <><><><>

014E  81 3E 0072 R 1234          CMP     RESET_FLAG,1234H               ; WARM START?
0154  75 03                      JNE     E19A                           ; GO IF NOT
0156  E9 0558 R                  JMP     SHUT2                          ; GO TO NEXT TEST IF WARM START

                         ;------- SET SHUTDOWN RETURN 2

0159  B0 30              E19A:   MOV     AL,30H                         ;<><><><><><><><><><><><><><>
015B  E6 80                      OUT     MFG_PORT,AL                    ;<><><>CHECKPOINT 30 <><><><>

015D  B0 8F                      MOV     AL,SHUT_DOWN                   ; ADDR FOR SHUTDOWN BYTE
015F  E6 70                      OUT     CMOS_PORT,AL                   ;
0161  B0 02                      MOV     AL,2                           ; SECOND ENTRY INTO TABLE
0163  EB 00                      JMP     SHORT $+2                      ; IO DELAY
0165  E6 71                      OUT     CMOS_PORT+1,AL                 ;
                                 ;--------------------
                         ;------- ENABLE PROTECTED MODE
                                 ;--------------------
0167  BC 0000                    MOV     SP,POST_SS                     ; SET STACK FOR SYSINIT1
016A  8E D4                      MOV     SS,SP                          ;
016C  BC 8000                    MOV     SP,POST_SP                     ;

016F  E8 0000 E                  CALL    SYSINIT1                        ; GO ENABLE PROTECTED MODE

0172  B0 31                      MOV     AL,31H                 ;<><><><><><><><><><><><><><>
0174  E6 80                      OUT     MFG_PORT,AL            ;<><><>CHECKPOINT 31 <><><><>

                         ;------- SET TEMPORY STACK

0176  B8 0008                    MOV     AX,GDT_PTR                     ;
0179  8E C0                      MOV     ES,AX                          ;
017B  26: C7 06 005A 0000        MOV     ES:SS_TEMP.BASE_LO_WORD,0
0182  26: C6 06 005C 00          MOV     BYTE PTR ES:(SS_TEMP.BASE_HI_BYTE),0
0188  BE 0058                    MOV     SI,SS_TEMP
018B  8E D6                      MOV     SS,SI
018D  BC FFFD                    MOV     SP,MAX_SEG_LEN-2

                         ;------- DATA SEGMENT TO SYSTEM DATA AREA

0190  B8 0018                    MOV     AX,RSDA_PTR                    ; POINT TO DATA AREA
0193  8E D8                      MOV     DS,AX                          ;

0195  B0 80                      MOV     AL,PRTY_CHK                    ; SET CHECK PARITY
```

SECTION 5

```
0197  E6 87                   OUT   DMA_PAGE+6,AL              ; SAVE WHICH CHECK TO USE

                        ;------- PRINT 64 K BYTES OK

0199  B8 0040           E20A:   MOV   AX,16*4                    ; STARTING AMT. OF MEMORY OK
019C  50                        PUSH  AX                         ; SAVE MEMORY OK SIZE
019D  E9 0347 R                 JMP   PRT_SIZ                    ; POST MESSAGE

                        ;-------- IS CMOS GOOD?

01A0  B0 8E             E20B:   MOV   AL,DIAG_STATUS        ; DETERMINE THE CONDITION OF CMOS
01A2  E6 70                     OUT   CMOS_PORT,AL          ;
01A4  EB 00                     JMP   SHORT $+2             ; IO DELAY
01A6  E4 71                     IN    AL,CMOS_PORT+1        ; GET THE CMOS STATUS
01A8  50                        PUSH  AX                    ; SAVE CMOS STATUS

                        ;-------- GET THE MEMORY SIZE DETERMINED (PREPARE BX FOR BAD CMOS)

01A9  B0 B1                     MOV   AL,M_SIZE_HI          ; GET THE HIGH BYTE
01AB  E6 70                     OUT   CMOS_PORT,AL          ;
01AD  EB 00                     JMP   SHORT $+2             ; IO DELAY
01AF  E4 71                     IN    AL,CMOS_PORT+1        ; HIGH BYTE
01B1  86 E0                     XCHG  AH,AL                 ; SAVE HIGH BYTE
01B3  B0 B0                     MOV   AL,M_SIZE_LO          ; GET LOW BYTE
01B5  E6 70                     OUT   CMOS_PORT,AL          ;
01B7  EB 00                     JMP   SHORT $+2             ; IO DELAY
01B9  E4 71                     IN    AL,CMOS_PORT+1        ; LOW BYTE
01BB  8B 1E 0013 R              MOV   BX,MEMORY_SIZE        ; PRE LOAD THE MEMORY SIZE
01BF  03 D8                     ADD   BX,AX                 ; SET TOTAL MEMORY SIZE
01C1  89 1E 0017 R              MOV   WORD PTR KB_FLAG,BX   ; SAVE THE TOTAL SIZE
01C5  58                        POP   AX                    ; RESTORE CMOS STATUS

01C6  A8 C0                     TEST  AL,0C0H               ; CMOS OK?
01C8  74 03                     JZ    E20B0                 ; GO IF YES
01CA  E9 026E R                 JMP   E20C                  ; DEFAULT IF NOT
01CD                    E20B0:
                        ;-------- GET THE BASE 0->640K MEMORY SIZE FROM CONFIG IN CMOS

01CD  B0 96                     MOV   AL,M1_SIZE_HI         ; GET THE HIGH BYTE
01CF  E6 70                     OUT   CMOS_PORT,AL          ;
01D1  EB 00                     JMP   SHORT $+2             ; IO DELAY
01D3  E4 71                     IN    AL,CMOS_PORT+1        ; HIGH BYTE
01D5  86 E0                     XCHG  AH,AL                 ; SAVE HIGH BYTE
01D7  B0 95                     MOV   AL,M1_SIZE_LO         ; GET LOW BYTE
01D9  E6 70                     OUT   CMOS_PORT,AL          ;
01DB  EB 00                     JMP   SHORT $+2             ; IO DELAY
01DD  E4 71                     IN    AL,CMOS_PORT+1        ; LOW BYTE
01DF  39 06 0013 R              CMP   MEMORY_SIZE,AX        ; IS MEMORY SIZE GREATER THAN CONFIG?
01E3  74 1C                     JZ    E20B1                 ; GO IF EQUAL

                        ;------- SET MEMERY SIZE DETERMINE NOT EQUAL TO CONFIG

01E5  50                        PUSH  AX                    ; SAVE AX
01E6  B0 8E                     MOV   AL,DIAG_STATUS        ;
01E8  E6 70                     OUT   CMOS_PORT,AL          ; ADDRESS THE STATUS BYTE
01EA  EB 00                     JMP   SHORT $+2             ; IO DELAY
01EC  E4 71                     IN    AL,CMOS_PORT+1        ; GET THE STATUS
01EE  0C 10                     OR    AL,W_MEM_SIZE         ; SET CMOS FLAG
01F0  86 C4                     XCHG    AH                  ; SAVE AL
01F2  B0 8E                     MOV     DIAG_STATUS         ;
01F4  E6 70                     OUT   CMOS_PORT,AL          ;
01F6  86 C4                     XCHG  AL,AH                 ; RESTORE AL
01F8  EB 00                     JMP   SHORT $+2             ; IO DELAY
01FA  E6 71                     OUT   CMOS_PORT+1,AL        ;
01FC  58                        POP   AX                    ; RESTORE AX
01FD  39 06 0013 R              CMP   MEMORY_SIZE,AX        ; IS MEMORY SIZE GREATER THAN CONFIG?
0201  77 6B             E20B1:  JA    E20C                  ; DEFAULT TO MEM SIZE DET IF YES
0203  8B D8                     MOV   BX,AX                 ; SET BASE MEMORY SIZE
0205  3D 0201                   CMP   AX,513                ; CHECK IF BASE RAM LESS 512K
0208  72 16                     JB    NO_640                ; GO IF YES
020A  B0 B3                     MOV   AL,INFO_STATUS        ; SET 640K BASE RAM BIT
020C  E6 70                     OUT   CMOS_PORT,AL          ;
020E  EB 00                     JMP   SHORT $+2             ; IO DELAY
0210  E4 71                     IN    AL,CMOS_PORT+1        ; GET THE CURRENT STATUS
0212  0C 80                     OR    AL,M640K              ; TURN ON 640K BIT IF NOT ALREADY ON
0214  86 C4                     XCHG  AL,AH                 ; SAVE THE CURRENT DIAG STATUS
0216  B0 B3                     MOV   AL,INFO_STATUS        ;
0218  E6 70                     OUT   CMOS_PORT,AL          ; ADDR THE STATUS BYTE
021A  86 C4                     XCHG  AL,AH                 ; RESTORE THE STATUS
021C  EB 00                     JMP   SHORT $+2             ; IO DELAY
021E  E6 71                     OUT   CMOS_PORT+1,AL        ;

                        ;-------- CHECK MEMORY SIZE ABOVE 640K FROM CONFIG

0220                    NO_640:
0220  B0 98                     MOV   AL,M2_SIZE_HI         ; GET THE HIGH BYTE
0222  E6 70                     OUT   CMOS_PORT,AL          ;
0224  EB 00                     JMP   SHORT $+2             ; IO DELAY
0226  E4 71                     IN    AL,CMOS_PORT+1        ; HIGH BYTE
0228  86 E0                     XCHG  AH,AL                 ; SAVE HIGH BYTE
022A  B0 97                     MOV   AL,M2_SIZE_LO         ; GET LOW BYTE
022C  E6 70                     OUT   CMOS_PORT,AL          ;
022E  EB 00                     JMP   SHORT $+2             ; IO DELAY
0230  E4 71                     IN    AL,CMOS_PORT+1        ; LOW BYTE
0232  8B C8                     MOV   CX,AX                 ; SAVE THE ABOVE 640K RAM SIZE
                        ;------- ABOVE 640K SIZE FROM MEMORY SIZE DETERMINE
                        ;------- CX=CONFIG  AX=MEMORY SIZE DETERMINE
0234  B0 B1                     MOV   AL,M_SIZE_HI          ; GET THE HIGH BYTE
0236  E6 70                     OUT   CMOS_PORT,AL          ;
0238  EB 00                     JMP   SHORT $+2             ; IO DELAY
023A  E4 71                     IN    AL,CMOS_PORT+1        ; HIGH BYTE
023C  86 E0                     XCHG  AH,AL                 ; SAVE HIGH BYTE
023E  B0 B0                     MOV   AL,M_SIZE_LO          ; GET LOW BYTE
0240  E6 70                     OUT   CMOS_PORT,AL          ;
0242  EB 00                     JMP   SHORT $+2             ; IO DELAY
0244  E4 71                     IN    AL,CMOS_PORT+1        ; LOW BYTE
                        ;------- WHICH IS GREATER
                        ;------- AX=MEMORY SIZE DETERMINE CX=CONFIG (ABOVE 640)  BX=SIZE (BELOW 640)
0246  3B C8                     CMP   CX,AX                 ; IS CONFIG EQUAL TO DETERMINED?
0248  74 18                     JZ    SET_MEM1              ; GO IF EQUAL
                        ;------- SET MEMERY SIZE DETERMINE NOT EQUAL TO CONFIG
024A  50                        PUSH  AX                    ; SAVE AX
024B  B0 8E                     MOV   AL,DIAG_STATUS        ;
024D  E6 70                     OUT   CMOS_PORT,AL          ; ADDRESS THE STATUS BYTE
024F  EB 00                     JMP   SHORT $+2             ; IO DELAY
0251  E4 71                     IN    AL,CMOS_PORT+1        ; GET THE STATUS
0253  0C 10                     OR    AL,W_MEM_SIZE         ; SET CMOS FLAG
0255  86 C4                     XCHG  AL,AH                 ; SAVE AL
0257  B0 8E                     MOV   AL,DIAG_STATUS        ;
0259  E6 70                     OUT   CMOS_PORT,AL          ;
025B  86 C4                     XCHG  AL,AH                 ; RESTORE AL
025D  EB 00                     JMP   SHORT $+2             ; IO DELAY
025F  E6 71                     OUT   CMOS_PORT+1,AL        ;
0261  58                        POP   AX                    ; RESTORE AX

0262                    SET_MEM1:
```

```
0262  3B C8                        CMP    CX,AX                    ; IS CONFIG GREATER THAN DETERMINED?
0264  77 02                        JA     SET_MEM                  ;  GO IF YES
0266  8B C8                        MOV    CX,AX                    ; USE MEMORY SIZE DETERMINE IF NOT
0268                       SET_MEM:
0268  03 D9                        ADD    BX,CX                    ; SET TOTAL MEMORY SIZE
026A  89 1E 0017 R                 MOV    WORD PTR KB_FLAG,BX      ; SAVE TOTAL SIZE FOR LATER TESTING
026E                       E20C:
026E  83 EB 40             E20D:   SUB    BX,16*4                          ; 1ST 64K ALREADY DONE
0271  B1 06                        MOV    CL,06H
0273  D3 EB                        SHR    BX,CL                            ; DIVIDE BY 54
0275  53                           PUSH   BX                               ; SAVE COUNT OF 64K BLOCKS
                                   ;=======================
                           ;------- MODIFY DESCRIPTOR TABLES
                                   ;=======================
                                                                           ;
0276  B8 0008                      MOV    AX,GDT_PTR                       ; MODIFY THE DESCRIPTER TABLE
0279  8E C0                        MOV    ES,AX                            ;
                                   ;---------------------------------------
                           ;------- SET TEMP ES DESCRIPTOR 64K SEGMENT LIMIT
                                   ;---------------------------------------

027B  26: C7 06 0048 FFFF          MOV    ES:ES_TEMP.SEG_LIMIT,MAX_SEG_LEN

                           ;------- CPLO, DATA ACCESS RIGHTS

0282  26: C6 06 004D 93            MOV    BYTE PTR ES:(ES_TEMP.DATA_ACC_RIGHTS),CPLO_DATA_ACCESS

                           ;------- START WITH SEGMENT 010000 (SECOND 64K)

0288  26: C6 06 004C 00            MOV    BYTE PTR ES:(ES_TEMP.BASE_HI_BYTE),0
028E  26: C7 06 004A 0000          MOV    ES:ES_TEMP.BASE_LO_WORD,0
                                   ;---------------------------------------
                           ;------- SET TEMP DS DESCRIPTOR 64K SEGMENT LIMIT
                                   ;---------------------------------------

0295  26: C7 06 0060 FFFF          MOV    ES:DS_TEMP.SEG_LIMIT,MAX_SEG_LEN

                           ;------- CPLO, DATA ACCESS RIGHTS

029C  26: C6 06 0065 93            MOV    BYTE PTR ES:(DS_TEMP.DATA_ACC_RIGHTS),CPLO_DATA_ACCESS

                           ;------- START WITH SEGMENT 010000

02A2  26: C6 06 0064 00            MOV    BYTE PTR ES:(DS_TEMP.BASE_HI_BYTE),0
02A8  26: C7 06 0062 0000          MOV    ES:DS_TEMP.BASE_LO_WORD,0

                           ;------- TEMPORARY SEGMENT SAVE IN DMA PAGE REGISTER

02AF  2A C0                        SUB    AL,AL
02B1  E6 85                        OUT    DMA_PAGE+4,AL            ; HIGH BYTE OF LOW WORD OF SEGMENT
02B3  E6 86                        OUT    DMA_PAGE+5,AL            ; LOW BYTE OF LOW WORD OF SEGMENT
02B5  FE C0                        INC    AL                       ; SET HIGH BYTE OF SEGMENT WORD
02B7  E6 84                        OUT    DMA_PAGE+3,AL            ; HIGH BYTE OF SEGMENT

                           ;------- POINT TO NEXT BLOCK OF 32K WORDS

02B9  B8 0008              E21:    MOV    AX,GDT_PTR               ; POINT TO START OF DESCR TABLE
02BC  8E D8                        MOV    DS,AX                    ;
02BE  FE 06 0064                   INC    BYTE PTR DS:(DS_TEMP.BASE_HI_BYTE)
02C2  FE 06 004C                   INC    BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE)

                           ;-------- CHECK FOR END OF 256K PLANAR RAM
02C6  80 3E 0064 04                CMP    BYTE PTR DS:(DS_TEMP.BASE_HI_BYTE),04H
02CB  72 12                        JB     E21_0                    ; GO IF STILL BASE RAM
02CD  1E                           PUSH   DS                       ; SAVE THE CURRENT DATA SEGMENT
02CE  B8 0018                      MOV    AX,RSDA_PTR              ; POINT TO POST DATA SEGMENT
02D1  8E D8                        MOV    DS,AX                    ;
02D3  A0 0012 R                    MOV    AL,MFG_TST               ; GET THE JUMPER INFO
02D6  1F                           POP    DS                       ; RESTORE DS
02D7  A8 10                        TEST   AL,BASE_RAM              ; CHECK IF SECOND 256K ON BASE PLANAR
02D9  75 04                        JNZ    E21_0                    ; GO IF YES
02DB  B0 40                        MOV    AL,IO_CHK                ; SET IO CHANNEL CHECK TEST
02DD  E6 87                        OUT    DMA_PAGE+6,AL            ;

                           ;------- CHECK END OF FIRST 516K OR 640K (END OF BASE RAM)

02DF  B0 B3                E21_0:  MOV    AL,INFO_STATUS           ; SET 640K BASE RAM BIT
02E1  E6 70                        OUT    CMOS_PORT,AL             ;
02E3  EB 00                        JMP    SHORT $+2                ; IO DELAY
02E5  E4 71                        IN     AL,CMOS_PORT+1           ; GET THE CURRENT STATUS

                           ;-------- CHECK FOR END OF 512K PLANAR RAM

02E7  80 3E 0064 08                CMP    BYTE PTR DS:(DS_TEMP.BASE_HI_BYTE),08H
02EC  72 08                        JB     E12_A                    ; GO IF STILL BASE RAM

                           ;------- SET USE TEST IO CHECK

02EE  86 C4                        XCHG   AL,AH                    ; SAVE AL
02F0  B0 40                        MOV    AL,IO_CHK                ;
02F2  E6 87                        OUT    DMA_PAGE+6,AL            ;
02F4  86 C4                        XCHG   AL,AH                    ; RESTORE AL

                           ;------- CHECK FOR 640K BASE RAM (128K IO CARD)

02F6  A8 80                E12_A:  TEST   AL,M640K                 ; IS 640K BASE INSTALLED?
02F8  74 0A                        JZ     E12_B                    ; GO IF NO

02FA  80 3E 0064 0A                CMP    BYTE PTR DS:(DS_TEMP.BASE_HI_BYTE),0AH
02FF  75 14                        JNZ    NEXT1
0301  EB 08 90                     JMP    E12_C                    ; CONTINUE

0304  80 3E 0064 08        E12_B:  CMP    BYTE PTR DS:(DS_TEMP.BASE_HI_BYTE),08H
0309  75 0A                        JNZ    NEXT1

                           ;------- DO ADDITIONAL STORAGE ABOVE 1 MEG

030B  C6 06 0064 10        E12_C:  MOV    BYTE PTR DS:(DS_TEMP.BASE_HI_BYTE),10H
0310  C6 06 004C 10                MOV    BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE),10H

                           ;------- SAVE BASE_HI_BYTE IN DMA PAGE REGISTERS 3

0315  A0 0064              NEXT1:  MOV    AL,BYTE PTR DS:(DS_TEMP.BASE_HI_BYTE)
0318  E6 84                        OUT    DMA_PAGE+3,AL            ; SAVE THE HIGH BYTE OF SEGMENT
                                                                   ;  FOR POSIBLE ERROR

                           ;------- CHECK FOR TOP OF RAM (FE0000) 16MEG

031A  80 3E 004C FE                CMP    BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE),0FEH ; TOP OF RAM?
031F  75 03                        JNZ    NEXT                     ; GO IF NOT
0321  EB 66 90                     JMP    KB_LOOP3                 ; GO NEXT TEST

                           ;------- SET ES AND DS REGISTERS
```

SECTION 5

```
0324  B8 0060                   NEXT:   MOV     AX,DS_TEMP
0327  8E D8                             MOV     DS,AX
0329  B8 0048                           MOV     AX,ES_TEMP
032C  8E C0                             MOV     ES,AX

032E  B0 31                             MOV     AL,31H                  ;<><><><><><><><><><><><><><><>
0330  E6 80                             OUT     MFG_PORT,AL             ;<><><>CHECKPOINT 31 <><><><>

0332  B9 8000                           MOV     CX,2000H*4              ; SET COUNT FOR 32K WORDS
0335  E8 0000 E                         CALL    STGTST_CNT
0338  74 03                             JZ      N1                      ; CONTINUE IF OK
033A  E9 047D R                         JMP     E21A                    ; GO PRINT ERROR
033D  59                        N1:     POP     CX                      ; POP CX TO GET AX
033E  58                                POP     AX                      ; RECOVER TESTED MEMORY

                                ;------- WRITE THE CURRENT SIZE FOR (ADDRESS LINE 23-17 TEST) USED LATER

033F  2B FF                             SUB     DI,DI                   ; POINT TO BEGINING OR A BLOCK
0341  AB                                STOSW                           ; WRITE THE CURRENT SIZE
                                                                        ;  AT THE STARTING ADDRESS
0342  05 0040                           ADD     AX,16*4
0345  50                                PUSH    AX                      ; SAVE TESTED MEMORY
0346  51                                PUSH    CX                      ; SAVE LOOP COUNT

0347                            PRT_SIZ:
0347  50                                PUSH    AX                      ;
0348  BB 000A                           MOV     BX,10                   ; SET DECIMAL CONVERT
                                ;------- CONVERT AND SAVE
034B  B9 0005                           MOV     CX,5                    ; OF 5 NIBBLES XX,XXX KB
034E  2B FF                             SUB     DI,DI                   ; CRT BUFFER POSITION
0350                            DECIMAL_LOOP:
0350  33 D2                             XOR     DX,DX
0352  F7 F3                             DIV     BX                      ; DIVIDE BY 10
0354  80 CA 30                          OR      DL,30H                  ; MAKE INTO ASCII
0357  52                                PUSH    DX                      ; SAVE
0358  E2 F6                             LOOP    DECIMAL_LOOP            ;
                                ;------- DISPLAY LAST OK MEMORY         ;
035A  B9 0005                           MOV     CX,5                    ;
035D                            PRT_DEC_LOOP:                           ;
035D  58                                POP     AX                      ; RECOVER A NUMBER
035E  E8 0000 E                         CALL    PROT_PRT_HEX
0361  47                                INC     DI                      ; POINT TO CRT BUFF
0362  E2 F9                             LOOP    PRT_DEC_LOOP
0364  B9 0006                           MOV     CX,6
0367  BE 0000 E                         MOV     SI,OFFSET F3B           ; PRINT ' KB OK'
036A                            KB_LOOP:
036A  2E: 8A 04                         MOV     AL,CS:[SI]
036D  46                                INC     SI
036E  E8 0000 E                         CALL    PROT_PRT_HEX
0371  47                                INC     DI                      ; INCREMENT BUFF PTR
0372  E2 F6                             LOOP    KB_LOOP
0374  58                                POP     AX                      ; RECOVER WORK REGS
0375  3D 0040                           CMP     AX,16*4                 ; FIRST PASS?
0378  75 03                             JNZ     KB_LOOP1                ; GO IF NOT
037A  E9 01A0 R                         JMP     E20B
037D                            KB_LOOP1:
037D  59                                POP     CX                      ; RECOVER 64K BLOCK COUNT
037E  58                                POP     AX                      ;
037F  E2 03                             LOOP    KB_LOOP2                ; LOOP TILL ALL MEM. CHECKED
0381  EB 06 90                          JMP     KB_LOOP3                ; CONTINUE
0384                            KB_LOOP2:
0384  50                                PUSH    AX                      ;
0385  51                                PUSH    CX                      ; SAVE LOOP COUNT
0386  E9 02B9 R                         JMP     E21                     ; LOOP TILL ALL MEM CHECKED

0389                            KB_LOOP3:
                                        ;=======================
                                ;------- ADDRESS LINE 16-23 TEST
                                        ;=======================

                                ;------- CALCULATE NUMBER OF  64K BLOCKS

0389  B8 0040                           MOV     AX,64                           ; START AT SECOND 64K
038C  50                                PUSH    AX                              ; SAVE STARTING ADDR

038D  B8 0018                           MOV     AX,RSDA_PTR                     ; GET THE MEMORY SIZE
0390  8E D8                             MOV     DS,AX                           ;

0392  8B 1E 0017 R                      MOV     BX,WORD PTR KB_FLAG             ; GET THE TOTAL MEMORY SIZE
                                                                                ; KB_FLAG USED AS TEMP STORAGE
0396  83 EB 40                          SUB     BX,64                           ; START AT SECOND 64K BOUNDRY
0399  B1 06                             MOV     CL,06H                          ; DIVIDE BY 64K
039B  D3 EB                             SHR     BX,CL                           ;
039D  53                                PUSH    BX                              ; SAVE LOOP COUNT

                                ;------- INITIALIZE DS DESCRIPTOR

039E  B8 0008                           MOV     AX,GDT_PTR
03A1  8E C0                             MOV     ES,AX
03A3  26: C6 06 0064 00                 MOV     BYTE PTR ES:(DS_TEMP.BASE_HI_BYTE),0
03A9  26: C7 06 0062 0000               MOV     ES:DS_TEMP.BASE_LO_WORD,0

                                ;------- TEMPORARY SEGMENT SAVE IN DMA PAGE REGISTER

03B0  2A C0                             SUB     AL,AL
03B2  E6 85                             OUT     DMA_PAGE+4,AL           ; HIGH BYTE OF LOW WORD OF SEGMENT
03B4  E6 86                             OUT     DMA_PAGE+5,AL           ; LOW BYTE OF LOW WORD OF SEGMENT
03B6  B0 01                             MOV     AL,01H                  ; SET HIGH BYTE OF SEGMENT WORD
03B8  E6 84                             OUT     DMA_PAGE+3,AL           ; HIGH BYTE OF SEGMENT

                                ;------- POINT TO NEXT BLOCK OF 64K

03BA                            E21_A:

03BA  B0 33                             MOV     AL,33H                  ;<><><><><><><><><><><><><><><>
03BC  E6 80                             OUT     MFG_PORT,AL             ;<><><>CHECKPOINT 33 <><><><>
03BE  26: 80 06 0064 01                 ADD     BYTE PTR ES:(DS_TEMP.BASE_HI_BYTE),01

                                ;------- CHECK END OF FIRST 516K OR 640K (END OF BASE RAM)

03C4  B0 B3                             MOV     AL,INFO_STATUS          ; SET 640K BASE RAM BIT
03C6  E6 70                             OUT     CMOS_PORT,AL            ;
03C8  EB 00                             JMP     SHORT $+2               ; IO DELAY
03CA  E4 71                             IN      AL,CMOS_PORT+1          ; GET THE CURRENT STATUS
03CC  A8 80                             TEST    AL,M640K                ; CHECK FOR 640K BASE RAM
03CE  74 0B                             JZ      NEXT_A1                 ; GO IF ONLY 512K

                                ;-------- CHECK FOR END OF 512K PLANAR RAM

03D0  26: 80 3E 0064 0A                 CMP     BYTE PTR ES:(DS_TEMP.BASE_HI_BYTE),0AH
03D6  75 15                             JNZ     NEXT_A                  ; GO IF STILL BASE RAM
03D8  EB 09 90                          JMP     NEXT_A2                 ;
03DB  26: 80 3E 0064 08         NEXT_A1:CMP     BYTE PTR ES:(DS_TEMP.BASE_HI_BYTE),08H
03E1  75 0A                             JNZ     NEXT_A
```

```
                              ;------- DO ADDITIONAL STORAGE ABOVE 1 MEG
03E3  26: C6 06 0064 10       NEXT_A2:MOV   BYTE PTR ES:(DS_TEMP.BASE_HI_BYTE),10H

                              ;------- SET USE TEST IO CHECK

03E9  B0 40                           MOV   AL,IO_CHK             ;
03EB  E6 87                           OUT   DMA_PAGE+6,AL         ;

03ED  26: A0 0064             NEXT_A: MOV   AL,BYTE PTR ES:(DS_TEMP.BASE_HI_BYTE)

                              ;------- DMA PAGE REGISTERS 3

03F1  E6 84                           OUT   DMA_PAGE+3,AL         ; SAVE THE HIGH BYTE OF SEGMENT
                                                                  ;  FOR POSIBLE ERROR

                              ;------- CHECK FOR TOP OF RAM (FE0000) 16MEG

03F3  26: 80 3E 0064 FE               CMP   BYTE PTR ES:(DS_TEMP.BASE_HI_BYTE),0FEH ; TOP OF RAM?
03F9  75 03                           JNZ   NEXT_B                ; GO IF NOT
03FB  EB 79 90                        JMP   KB_LOOP_3             ; GO NEXT TEST

                              ;------- SET DS REGISTER

03FE  B8 0060                 NEXT_B: MOV   AX,DS_TEMP
0401  8E D8                           MOV   DS,AX
0403  2B FF                           SUB   DI,DI                 ; POINT TO START OF BLOCK
0405  8B 05                           MOV   AX,DS:[DI]            ; GET THE VALUE OF THIS BLOCK
0407  8B D0                           MOV   DX,AX                 ; SAVE
0409  8B F7                           MOV   SI,DI                 ; SET SI FOR POSSIBLE ERROR
040B  2B C0                           SUB   AX,AX                 ; CLEAR RAM LOCATION
040D  89 05                           MOV   DS:[DI],AX            ;
                              ;------- ALLOW CRT TIME TO DISPLAY MSG
040F  2B C9                           SUB   CX,CX                 ;
0411  E2 FE                   Z2:     LOOP  Z2
0413  59                              POP   CX                    ; GET THE LOOP COUNT
0414  58                              POP   AX                    ; RECOVER TESTED MEMORY
0415  50                              PUSH  AX                    ; SAVE TESTED MEMORY
0416  51                              PUSH  CX                    ; SAVE LOOP COUNT
0417  3B C2                           CMP   AX,DX                 ; DOES THE BLOCK ID MATCH
0419  8B C2                           MOV   AX,DX                 ; GET THE BLOCK ID FOR POSSIBLE ERROR
041B  75 60                           JNZ   E21A                  ; GO PRINT ERROR
041D  59                              POP   CX                    ; POP CX TO GET AX
041E  58                              POP   AX                    ; RECOVER TESTED MEMORY
041F  05 0040                         ADD   AX,64                 ; 64K INCREMENTS
0422  50                              PUSH  AX                    ; SAVE TESTED MEMORY
0423  51                              PUSH  CX                    ; SAVE LOOP COUNT

0424  50                              PUSH  AX                    ;
0425  BB 000A                         MOV   BX,10                 ; SET DECIMAL CONVERT

                              ;------- CONVERT AND SAVE

0428  B9 0005                         MOV   CX,5                  ; OF 5 NIBBLES XX,XXX KB
042B  2B FF                           SUB   DI,DI                 ; CRT BUFFER PCSITION
042D                          DEC_LOOP:
042D  33 D2                           XOR   DX,DX
042F  F7 F3                           DIV   BX                    ; DIVIDE BY 10
0431  80 CA 30                        OR    DL,30H                ; MAKE INTO ASCII
0434  52                              PUSH  DX                    ; SAVE
0435  E2 F6                           LOOP  DEC_LOOP              ;
                              ;------- DISPLAY LAST OK MEMORY     ;
0437  B9 0005                         MOV   CX,5                  ;
043A                          PRT_DEC:                            ;
043A  58                              POP   AX                    ; RECOVER A NUMBER
043B  E8 0000 E                       CALL  PROT_PRT_HEX
043E  47                              INC   DI                    ; POINT TO CRT BUFF
043F  E2 F9                           LOOP  PRT_DEC
0441  B9 0006                         MOV   CX,6
0444  BE 0000 E                       MOV   SI,OFFSET F3B         ; PRINT ' KB OK'
0447                          KB_LOOP_1:
0447  2E: 8A 04                       MOV   AL,CS:[SI]
044A  46                              INC   SI
044B  E8 0000 E                       CALL  PROT_PRT_HEX
044E  47                              INC   DI                    ; INCREMENT BUFF PTR
044F  E2 F6                           LOOP  KB_LOOP_1
0451  58                              POP   AX                    ; RECOVER WORK REGS
0452  59                              POP   CX                    ; RECOVER 64K BLOCK COUNT
0453  58                              POP   AX                    ;
0454  E2 1B                           LOOP  KB_LOOP_2             ; LOOP TILL ALL MEM. CHECKED

                              ;-------- CHECK PARITY

0456  E6 89                           OUT   DMA_PAGE+8,AL         ; SAVE AX
0458  86 C4                           XCHG  AL,AH                 ;
045A  E6 8A                           OUT   DMA_PAGE+9,AL         ;
045C  E4 61                           IN    AL,PORT_B             ; CHECK FOR IO OR PAR CHECK
045E  24 C0                           AND   AL,PARITY_ERR         ; STRIP UNWANTED BITS
0460  86 C4                           XCHG  AL,AH                 ; SAVE ERROR
0462  E4 87                           IN    AL,DMA_PAGE+6         ; CHECK FOR R/W OR IO ERR
0464  22 E0                           AND   AH,AL                 ;
0466  E4 8A                           IN    AL,DMA_PAGE+9         ; RESTORE AX
0468  86 C4                           XCHG  AL,AH                 ;
046A  E4 89                           IN    AL,DMA_PAGE+8         ;
046C  75 0F                           JNZ   E21A                  ; GO IF PARITY ERROR

046E  EB 06 90                        JMP   KB_LOOP_3             ; CONTINUE
0471                          KB_LOOP_2:
0471  50                              PUSH  AX                    ;
0472  51                              PUSH  CX                    ; SAVE LOOP COUNT

0473  E9 03BA R                       JMP   E21_A                 ; CONTINUE TILL DONE

                              ;------- BACK TO REAL MODE
0476                          KB_LOOP_3:

0476  B0 34                           MOV   AL,34H                ;<><><><><><><><><><><><><><>
0478  E6 80                           OUT   MFG_PORT,AL           ;<><><>CHECKPOINT 34 <><><><>

047A  E9 0000 E                       JMP   PROC_SHUTDOWN         ; BACK TO REAL MODE
                                                                  ; NEXT TEST VIA JUMP TABLE (SHUT2)

                              ;----- PRINT FAILING ADDRESS AND XOR'ED PATTERN IF DATA COMPARE ERROR

                              ;----- USE DMA PAGE REGISTERS AS TEMPORY SAVE AREA FOR ERROR
                              ;      SET SHUTDOWN 3

047D  E6 82                   E21A:   OUT   DMA_PAGE+1,AL         ; SAVE FAILING BIT PATTERN (LOW BYTE)
047F  8A C4                           MOV   AL,AH                 ; SAVE HIGH BYTE
0481  EB 00                           JMP   SHORT $+2             ; IO DELAY
0483  E6 83                           OUT   DMA_PAGE+2,AL         ;
0485  8B C6                           MOV   AX,SI                 ; GET THE FAILING OFFSET
0487  E6 86                           OUT   DMA_PAGE+5,AL         ;
0489  86 E0                           XCHG  AH,AL                 ;
048B  EB 00                           JMP   SHORT $+2             ; IO DELAY
048D  E6 85                           OUT   DMA_PAGE+4,AL         ;
```

SECTION 5

```
                                      ;------- CLEAR IO CH CHK OR R/W PAR CHK

048F  2B F6                                  SUB     SI,SI                    ; WRITE TO FAILING BLOCK
0491  AB                                     STOSW                            ;
0492  E4 61                                  IN      AL,PORT_B                ;
0494  0C 0C                                  OR      AL,RAM_PAR_OFF           ; TOGGLE IO/PAR CHECK ENABLE
0496  EB 00                                  JMP     SHORT $+2                ; IO DELAY
0498  E6 61                                  OUT     PORT_B,AL                ;
049A  24 F3                                  AND     AL,RAM_PAR_ON            ;
049C  EB 00                                  JMP     SHORT $+2                ; IO DELAY
049E  E6 61                                  OUT     PORT_B,AL                ;
                                             ;===============
                                      ;------- SET MEMORY SIZE
                                             ;===============
04A0  B8 0018                                MOV     AX,RSDA_PTR              ; SET THE DATA SEGMENT
04A3  8E D8                                  MOV     DS,AX                    ; IN PROTECTED MODE

                                      ;------- GET THE DIAG_STATUS FROM CMOS

04A5  B0 8E                                  MOV     AL,DIAG_STATUS           ;
04A7  E6 70                                  OUT     CMOS_PORT,AL             ;
04A9  EB 00                                  JMP     SHORT $+2                ; IO DELAY
04AB  E4 71                                  IN      AL,CMOS_PORT+1           ;
04AD  8A D8                                  MOV     BL,AL                    ; SAVE THE STATUS BYTE

04AF  B0 B3                                  MOV     AL,INFO_STATUS           ;
04B1  E6 70                                  OUT     CMOS_PORT,AL             ;
04B3  EB 00                                  JMP     SHORT $+2                ; IO DELAY
04B5  E4 71                                  IN      AL,CMOS_PORT+1           ;
04B7  8A F8                                  MOV     BH,AL                    ; SAVE THE STATUS BYTE

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

04B9  59                                     POP     CX                       ;
04BA  58                                     POP     AX                       ; GET THE LAST OF GOOD MEMORY
04BB  8B C8                                  MOV     CX,AX                    ; SAVE IT

                                      ;------- BELOW 512K?

04BD  3D 0200                                CMP     AX,512                   ; LAST GOOD MEMORY BELOW 512K?
04C0  72 39                                  JB      M3                       ; GO IF YES

                                      ;------- BELOW 640K?

04C2  3D 0280                                CMP     AX,640                   ; LAST GOOD MEMORY BELOW 640K?
04C5  72 11                                  JB      M1                       ; GO IF YES

                                      ;------- 640K UP ERROR

04C7  F6 C7 80                               TEST    BH,M640K                 ; IS BASE RAM 640K
04CA  75 06                                  JNZ     M0                       ;
04CC  2D 0200                                SUB     AX,512                   ; 512K BASE RAM
04CF  EB 0F 90                               JMP     M2                       ;
04D2  2D 0280                         M0:    SUB     AX,640                   ; 640K BASE RAM
04D5  EB 09 90                               JMP     M2                       ;

                                      ;------- 512K TO 640K ERROR

04D8  F6 C7 80                        M1:    TEST    BH,M640K                 ; IS BASE RAM 640K?
04DB  75 1E                                  JNZ     M3                       ; GO IF YES
04DD  2D 0200                                SUB     AX,512                   ; STRIP BASE RAM FROM IO RAM
                                      ;------- WRITE SIZE TO CMOS

04E0  8B C8                           M2:    MOV     CX,AX                    ; SAVE ADJUSTED MEMORY SIZE
04E2  B0 B1                                  MOV     AL,M_SIZE_HI             ;
04E4  E6 70                                  OUT     CMOS_PORT,AL             ;
04E6  8A C5                                  MOV     AL,CH                    ; GET THE HIGH BYTE MEMORY SIZE
04E8  EB 00                                  JMP     SHORT $+2                ; IO DELAY
04EA  E6 71                                  OUT     CMOS_PORT+1,AL           ; WRITE IT
04EC  B0 B0                                  MOV     AL,M_SIZE_LO             ; DO THE LOW BYTE
04EE  EB 00                                  JMP     SHORT $+2                ;
04F0  E6 70                                  OUT     CMOS_PORT,AL             ;
04F2  8A C1                                  MOV     AL,CL                    ; GET THE LOW BYTE
04F4  EB 00                                  JMP     SHORT $+2                ; IO DELAY
04F6  E6 71                                  OUT     CMOS_PORT+1,AL           ; WRITE IT
04F8  EB 04 90                               JMP     M4                       ; CONTINUE

                                      ;------- SET BASE MEMORY SIZE

04FB  A3 0013 R                       M3:    MOV     MEMORY_SIZE,AX           ; TO INDICATE HOW MUCH MEM WORKING

                                      ;------- SET SHUTDOWN 3

04FE  B0 8F                           M4:    MOV     AL,SHUT_DOWN             ; ADDR FOR SHUTDOWN RETURN
0500  E6 70                                  OUT     CMOS_PORT,AL             ;
0502  B0 03                                  MOV     AL,3                     ; SET RETURN 3
0504  EB 00                                  JMP     SHORT $+2                ; IO DELAY
0506  E6 71                                  OUT     CMOS_PORT+1,AL           ;

                                      ;------- SHUTDOWN

0508  E9 0000 E                              JMP     PROC_SHUTDOWN            ;
                                      PAGE
                                      ;------- ENTRY 3 FROM PROCESSOR SHUTDOWN

                                      ;-----------------------------------------------------------------
                                      ; MEMORY ERROR REPORTING                                          :
                                      ;                                                                 :
                                      ; DESCRIPTION FOR ERRORS 201(CMP ERROR or PARITY)                 :
                                      ;                  or 202(ADDRESS LINE 0-15 ERROR)                :
                                      ;       R/W MEMORY ERRORS WILL BE REPORTED AS FOLLOWS             :
                                      ;                                                                 :
                                      ;       AABBCC DDEE 201(or 202)                                   :
                                      ;       AA=HIGH BYTE OF 24 BIT ADDRESS                            :
                                      ;       BB=MIDDLE BYTE OF 24 BIT ADDRESS                          :
                                      ;       CC=LOW BYTE OF 24 BIT ADDRESS                             :
                                      ;       DD=HIGH BYTE OF XOR FAILING BIT PATTERN                   :
                                      ;       EE=LOW BYTE OF XOR FAILING BIT PATTERN                    :
                                      ;                                                                 :
                                      ; DESCRIPTION FOR ERROR 202 (ADDRESS LINE 00-15)                  :
                                      ;       A WORD OF FFFF IS WRITTEN AT THE FIRST WORD AND LAST WORD :
                                      ;       OF EACH 64K BLOCK WITH ZEROS AT ALL OTHER LOCATIONS OF THE :
                                      ;       BLOCK.  A SCAN OF THE BLOCK IS MADE TO INSURE ADDRESS LINE :
                                      ;       0-15 ARE FUNCTIONING.                                     :
                                      ;                                                                 :
                                      ; DESCRIPTION FOR ERROR 203 (ADDRESS LINE 16-23)                  :
                                      ;       AT THE LAST PASS OF THE STORAGE TEST, FOR EACH BLOCK OF   :
                                      ;       64K, THE CURRENT STORAGE SIZE (ID) IS WRITTEN AT THE FIRST :
                                      ;       WORD OF EACH BLOCK.  IT IS USED TO DETERMINE ADDRESSING   :
                                      ;       FAILURES.                                                 :
                                      ;                                                                 :
                                      ;       AABBCC DDEE 203                                           :
                                      ;       SAME AS ABOVE EXCEPT FOR DDEE                             :
                                      ;
```

```
                          ; GENERAL DESCRIPTION FOR BLOCK ID (DDEE WILL NOW CONTAINT THE ID)  :
                          ;       DD=HIGH BYTE OF BLOCK ID                                    :
                          ;       EE=LOW BYTE OF BLOCK ID                                     :
                          ;                                                                   :
                          ;       BLOCK ID         ADDRESS RANGE                              :
                          ;       0000             000000 --> 00FFFF                          :
                          ;       0040             010000 --> 01FFFF                          :
                          ;       //                                                          :
                          ;       0200             090000 --> 09FFFF (512->576K) IF 640K BASE :
                          ;                        100000 --> 10FFFF (1024->1088K) IF 512K BASE:
                          ;                                                                   :
                          ; EXAMPLE (640K BASE RAM + 512K IO RAM = 1152K TOTAL)               :
                          ;       NOTE: THE CORRECT BLOCK ID FOR THIS FAILURE IS 0280 HEX.    :
                          ;             DUE TO AN ADDRESS FAILUE THE BLOCK ID+128K OVER-      :
                          ;             LAYED THE CORRECT BLOCK ID.                           :
                          ;       00640K OK        <-- LAST OK MEMORY                         :
                          ;       10000 0300 202   <-- ERROR DUE TO ADDRESS FAILURE           :
                          ; :                                                                 :
                          ;                                                                   :
                          ; DMA PAGE REGISTERS ARE USED AS TEMPORARY SAVE AREAS FOR SEGMENT   :
                          ; DESCRIPTER VALUES.                                                :
                          ;--------------------------------------------------------------------
                          ;
050B  B8 ---- R           SHUT3:  MOV     AX,DATA                 ; SET REAL MODE DATA SEGMENT
050E  8E D8                       MOV     DS,AX                   ;

                          ;------- INIT AND SET MFG ERROR

0510  C6 06 0016 R 00             MOV     MFG_ERR_FLAG+1,0        ; CLEAR FLAG

0515  80 0E 0016 R 01             OR      MFG_ERR_FLAG+1,MEM_FAIL ;<><><><><><><><><><><><><><><>
                                                                  ;<><> MEMORY FAILED<><><><><><>

051A  B0 0D                       MOV     AL,13                   ; CARRAGE RETURN
051C  E8 0000 E                   CALL    PRT_HEX
051F  B0 0A                       MOV     AL,10
0521  E8 0000 E                   CALL    PRT_HEX
0524  E4 84                       IN      AL,DMA_PAGE+3           ; GET THE HIGH BYTE OF 24 BIT ADDRESS
0526  E8 0000 E                   CALL    XPC_BYTE                ; CONVERT AND PRINT CODE
                                                                  ; CHECKPOINT 00->FE
0529  E4 85                       IN      AL,DMA_PAGE+4           ; GET THE MIDDLE BYTE OF 24 BIT ADDRESS
052B  E8 0000 E                   CALL    XPC_BYTE                ;
052E  E4 86                       IN      AL,DMA_PAGE+5           ; GET THE LOW BYTE OF 24 BIT ADDRESS
0530  E8 0000 E                   CALL    XPC_BYTE                ;
0533  B0 20                       MOV     AL,' '                  ; SPACE TO MESSAGE
0535  E8 0000 E                   CALL    PRT_HEX                 ;
0538  E4 83                       IN      AL,DMA_PAGE+2           ; GET HIGH BYTE FAILING BIT PATTERN
053A  E8 0000 E                   CALL    XPC_BYTE                ; CONVERT AND PRINT CODE
053D  E4 82                       IN      AL,DMA_PAGE+1           ; GET LOW BYTE FAILING BIT PATTERN
053F  E8 0000 E                   CALL    XPC_BYTE                ; CONVERT AND PRINT CODE

                          ;-------- CHECK FOR ADDRESS ERROR

0542  E4 80                       IN      AL,MFG_PORT             ; GET THE CHECKPOINT
0544  3C 33                       CMP     AL,33H                  ; IS IT AN ADDRESS FAILURE?
0546  BE 0000 E                   MOV     SI,OFFSET ADERR         ; PRELOAD ADDRESS ERROR 16->23
0549  74 0A                       JZ      ERR2                    ; GO IF YES
054B  BE 0000 E                   MOV     SI,OFFSET ADERR1        ; PRELOAD ADDRESS ERROR 00->15
054E  3C 32                       CMP     AL,32H                  ; GO IF YES
0550  74 03                       JZ      ERR2
0552  BE 0000 E                   MOV     SI,OFFSET E1            ; SETUP ADDRESS OF ERROR MSG
0555  E8 0000 E           ERR2:   CALL    E_MSG                   ; PRINT ERROR MSG
                          ;------- ENTRY FROM SHUTDOWN

0558                      SHUT2:

                          ;--------------------------------------------------------
                          ; TEST.20                                               :
                          ;       ADDITIONAL PROTECTED (VIRTUAL MODE) TEST        :
                          ; DESCRIPTION                                           :
                          ;       THE PROCESSOR IS PUT IN PROTECTED MODE AND      :
                          ;       THE FOLLOWING FUNCTIONS ARE VERIFIED            :
                          ;                                                       :
                          ;       1. VERIFY PROTECTED MODE                        :
                          ;          THE MACHINE STATUS IS CHECK FOR VIRTUAL MODE :
                          ;       2. PROCRAMMED INTERRUPT TEST                    :
                          ;          AN PROGRAMMED INTERRUPT 32 IS ISSUED AND     :
                          ;          AND VERIFIED                                 :
                          ;       3. EXCEPTION INT 13 TEST                        :
                          ;          A DESCRIPTOR SEGMENT LIMIT IS SET TO ZERO    :
                          ;          AND A WRITE TO THAT SEGMENT IS ATTEMPTED     :
                          ;          AN EXCEPTION 13 IS EXPECTED AND VERIFIED     :
                          ;       4. LDT/SDT LTR/STR TEST                         :
                          ;          LOAD LDT REGISTER AND VERIFY CORRECT         :
                          ;          LOAD TASK REGISTER AND VERIFY CORRECT        :
                          ;          THEY ARE VERIFIED VIA THE STORE INSTRUCTION  :
                          ;       5. THE CONTROL FLACS OF THE 286 FOR DIRECTION   :
                          ;          ARE VERIFIED VIA THE STD AND CLD COMMANDS    :
                          ;          IN PROTECTED MODE                            :
                          ;       6. BOUND INSTRUCTION TEST (EXC INT 5)           :
                          ;          CREATE A SIGNED ARRAY INDEX WITHIN AND       :
                          ;          OUTSIDE THE LIMITS.  CHECK THAT NO EXC INT   :
                          ;          IF WITHIN LIMIT AND THAT AN EXC INT 5        :
                          ;          OCCURS IF OUTSIDE THE LIMITS.                :
                          ;       7. PUSH ALL POP ALL TEST                        :
                          ;          SET ALL GENERAL PURPOSE REGS TO DIFFERENT    :
                          ;          VALUES ISSUE A PUSH ALL, CLEAR THE REGS      :
                          ;          ISSUE A POP ALL AND VERIFY CORRECT.          :
                          ;       8. CHECK THE VERR/VERW INSTRUCTIONS             :
                          ;          THE ACCESS BYTE IS SET TO READ ONLY THEN TO  :
                          ;          A WRITE ONLY AND THE VERR/VERW INST ARE      :
                          ;          VERIFIED.                                    :
                          ;       9. CAUSE AN INTERRUPT 13 VIA A WRITE TO A       :
                          ;          READ ONLY SEGMENT                            :
                          ;      10. VERIFY THE ARPL INSTRUCTION FUNCTIONS        :
                          ;          SET THE RPL FIELD OF A SELECTOR AND          :
                          ;          VERIFY THAT CURRENT SELECTOR RPL IS SET      :
                          ;          CORRECTLY.                                   :
                          ;      11. VERIFY THE LAR INSTRUCTION FUNCTIONS         :
                          ;      12. VERIFY THE LSL INSTRUCTION FUNCTIONS         :
                          ;      13. LOW MEG CHIP SELECT TEST                     :
                          ;--------------------------------------------------------

0558  E9 0000 E                   JMP     POST7                   ; GO TEST THE 286 PROTECTED MODE

                          ;-------- FAILURE ENTRY FROM A SHUTDOWN

055B  E8 0000 E           SHUT7:  CALL    DDS                     ; ESTABLISH THE DATA SEGMENT
055E  E4 80                       IN      AL,MFG_PORT             ; CHECK FOR CHIP SELECT ERROR
0560  3C 35                       CMP     AL,35H                  ;
0562  BE 0000 E                   MOV     SI,OFFSET CM4_D         ; PRINT ERROR 109
0565  74 0E                       JZ      SHUT7B                  ; GO IF NOT
0567  BE 0000 E           SHUT7A: MOV     SI,OFFSET VIR_ERR       ; PROTECTED MODE FAILED

056A  80 0E 0016 R 02             OR      MFG_ERR_FLAG+1,PRO_FAIL ;<><><><><><><><><><><><><><><>
```

SECTION 5

```
                                                                    ;<><> VIRTUAL MODE FAILED<><><>

056F  E8 0000 E                     CALL     E_MSG                    ; PRINT MSG
0572  EB 09 90                      JMP      SHUT6                    ;
0575  E8 0000 E            SHUT7B:  CALL     E_MSG                    ; PRINT MSG

0578  80 0E 0016 R 04               OR       MFG_ERR_FLAG+1,LMCS_FAIL;<><><><><><><><><><><><><><>
                                                                    ;<><> LOW MEG CHIP SELECT  <><>

                           ;-------- PROTECTED MODE TEST PASSED ENTRY FROM A SHUTDOWN

057D  E8 0000 E            SHUT6:   CALL     DDS                      ; PROTECTED MODE TEST PASSED
0580  2B C0                         SUB      AX,AX                    ; CLEAR KEYBOARD STATE FLAGS
0582  A3 0017 R                     MOV      WORD PTR KB_FLAG,AX      ;
0585  B9 000E                       MOV      CX,0EH                   ; CLEAR PAGE REGS
0588  BA 0082                       MOV      DX,DMA_PAGE+1            ;
058B                       CLR_LOOP:
058B  2A C0                         SUB      AL,AL                    ;
058D  EE                            OUT      DX,AL                    ;
058E  42                            INC      DX                       ;
058F  E2 FA                         LOOP     CLR_LOOP                 ;

                           ;----------------------------------------------
                           ; TEST.21                                      :
                           ;        KEYBOARD TEST                         :
                           ; DESCRIPTION                                  :
                           ;        RESET THE KEYBOARD AND CHECK THAT SCAN :
                           ;        CODE `AA' IS RETURNED TO THE CPU.     :
                           ;        CHECK FOR STUCK KEYS.                 :
                           ;----------------------------------------------

0591  B0 35                         MOV      AL,35H                   ;<><><><><><><><><><><><><>
0593  E6 80                         OUT      MFG_PORT,AL              ;<><><>CHECKPOINT 35 <><><>

0595  F6 06 0012 R 20               TEST     MFG_TST,LOOP_POST        ; MANUFACTURING BURN IN TEST MODE?
059A  75 03                         JNZ      F7_A
059C  E9 0651 R                     JMP      F7                       ; YES - SKIP KEYBOARD TEST
059F  80 3E 0072 R 64      F7_A:    CMP      BYTE PTR RESET_FLAG,064H ; MANUFACUTRING RUN IN MODE?
05A4  75 03                         JNZ      F7_B
05A6  E9 0651 R                     JMP      F7                       ; YES - SKIP KEYBOARD TEST
05A9  B0 36                F7_B:    MOV      AL,36H                   ;<><><><><><><><><><><><><>
05AB  E6 80                         OUT      MFG_PORT,AL              ;<><><>CHECKPOINT 36 <><><>
05AD  FA                            CLI
05AE  81 3E 0072 R 1234             CMP      RESET_FLAG,1234H         ; SOFT RESET?
05B4  74 17                         JZ       G10
05B6  80 3E 0072 R AA               CMP      BYTE PTR RESET_FLAG,0AAH ; CHECK FOR AA ALREADY RECIEVED
05BB  74 10                         JZ       G10                      ; GO IF YES
05BD  B0 AE                         MOV      AL,ENA_KBD
05BF  E8 0000 E                     CALL     C8042                    ; ENABLE KEYBOARD
05C2  B7 04                         MOV      BH,4                     ; TRY 4 TIMES
05C4  E8 0000 E            LOOP1:   CALL     OBF_42                   ; CHECK FOR OUTPUT BUFFER FULL
05C7  75 04                         JNZ      G10                      ; GO IF BUFFER FULL
05C9  FE CF                         DEC      BH
05CB  75 F7                         JNZ      LOOP1                    ;
05CD  B0 AD                G10:     MOV      AL,DIS_KBD               ; DISABLE KEYBOARD
05CF  E8 0000 E                     CALL     C8042                    ;
05D2  E4 60                         IN       AL,PORT_A                ; FLUSH
05D4  B0 E0                         MOV      AL,KYBD_CLK_DATA         ; GET THE CLOCK AND DATA LINES
05D6  E8 0000 E                     CALL     C8042                    ;
05D9  E8 0000 E                     CALL     OBF_42                   ; WAIT FOR OUTPUT BUFFER FULL
05DC  E4 60                         IN       AL,PORT_A                ; GET THE RESULTS
05DE  A8 01                         TEST     AL,KYBD_CLK              ; KEYBOARD CLOCK MUST BE LOW
05E0  74 0B                         JZ       G11
05E2  80 0E 0016 R 08               OR       MFG_ERR_FLAG+1,KYCLK_FAIL; <><><><><><><><><><><><><>
                                                                    ;<><> KEYBOARD CLOCK HIGH<><><>
05E7  BE 0000 E                     MOV      SI,OFFSET F1_B           ; DISPLAY 304 ERROR
05EA  EB 62 90                      JMP      F6D                      ; REPORT ERROR
05ED  E8 0000 E            G11:     CALL     KBD_RESET                ; ISSUE RESET TO KEYBRD
05F0  E3 28                         JCXZ     F6                       ; PRINT ERR MSG IF NO INTERRUPT
05F2  B0 37                         MOV      AL,37H                   ;<><><><><><><><><><><><><>
05F4  E6 80                         OUT      MFG_PORT,AL              ;<><><>CHECKPOINT 37 <><><>
05F6  80 FB AA                      CMP      BL,0AAH                  ; SCAN CODE AS EXPECTED?
05F9  75 1F                         JNE      F6                       ; NO - DISPLAY ERROR MSG

                           ;----- CHECK FOR STUCK KEYS

05FB  B0 38                         MOV      AL,38H                   ;<><><><><><><><><><><><><>
05FD  E6 80                         OUT      MFG_PORT,AL              ;<><><>CHECKPOINT 38 <><><>

05FF  B0 AE                         MOV      AL,ENA_KBD               ; ASSURE KYBOARD ENABLED
0601  E8 0000 E                     CALL     C8042                    ; ISSUE THE COMMAND
0604  2B C9                         SUB      CX,CX
0606  E2 FE                F5:      LOOP     F5                       ; DELAY FOR A WHILE
0608  E4 64                         IN       AL,STATUS_PORT           ; CHECK FOR STUCK KEYS
060A  A8 01                         TEST     AL,OUT_BUF_FULL          ; OUT BUFFER FULL?
060C  74 43                         JE       F7                       ; YES - CONTINUE TESTING

060E  B0 39                         MOV      AL,39H                   ;<><><><><><><><><><><><><>
0610  E6 80                         OUT      MFG_PORT,AL              ;<><><>CHECKPOINT 39 <><><>

0612  E4 60                         IN       AL,PORT_A                ; GET THE SCAN CODE
0614  E8 0000 E                     CALL     XPC_BYTE                 ; CONVERT AND PRINT
0617  EB 2D 90                      JMP      F6C                      ; CONTINUE

                           ;------- KEYBOARD ERROR TRY TO DETERMINE IF 8042 INTERFACE IS WORKING

061A  FA                   F6:      CLI                               ;
061B  B0 AB                         MOV      AL,INTR_FACE_CK          ; COMMAND TO 8042
061D  E6 64                         OUT      STATUS_PORT,AL           ;
061F  2B C9                         SUB      CX,CX                    ;
0621  B7 05                         MOV      BH,05                    ; WAIT FOR OUTPUT BUFFER FULL
0623  E4 64                F6A:     IN       AL,STATUS_PORT           ;
0625  A8 01                         TEST     AL,OUT_BUF_FULL          ; 8042 FINISHED TEST?
0627  E1 FA                         LOOPZ    F6A                      ;
0629  75 0A                         JNZ      F6B                      ; GO CHECK RESULTS
062B  FE CF                         DEC      BH                       ;
062D  75 F4                         JNZ      F6A                      ; TRY AGAIN
062F  BE 0000 E                     MOV      SI,OFFSET F1_A           ; INDICATE PLANAR FAILURE
0632  EB 1A 90                      JMP      F6D                      ;    (REMOVE KEYBOARD TRY AGAIN)
0635  E4 60                F6B:     IN       AL,PORT_A                ; GET THE RESULTS OF INTERFACE TEST
0637  3C 00                         CMP      AL,0                     ; IS THE INTERFACE OK?
0639  74 0B                         JZ       F6C                      ;
063B  80 0E 0016 R 10               OR       MFG_ERR_FLAG+1,KY_SYS_FAIL;<><><><><><><><><><><><><>
                                                                    ;<><> KEYBOARD/SYSTEM<><><><>
0640  BE 0000 E                     MOV      SI,OFFSET F1_A           ; PLANAR FAILURE
0643  EB 09 90                      JMP      F6D                      ; GO IF YES
0646  BE 0000 E            F6C:     MOV      SI,OFFSET F1             ; GET MSG ADDR

0649  80 0E 0016 R 20               OR       MFG_ERR_FLAG+1,KYBD_FAIL;<><><><><><><><><><><><><><>
                                                                    ;<><> KEYBOARD FAILED<><><><>

064E  E8 0000 E            F6D:     CALL     E_MSG                    ; PRINT MSG ON SCREEN

                           ;------- INITIALIZE 8042 TO HONOR KEY LOCK
```

```
0651  B0 3A                 F7:     MOV     AL,3AH                  ;<><><><><><><><><><><><><><>
0653  E6 80                         OUT     MFG_PORT,AL             ;<><><>CHECKPOINT 3A <><><><>

0655  B0 FF                         MOV     AL,0FFH                 ; DISABLE INTERRUPTS
0657  E6 21                         OUT     INTA01,AL               ;
0659  FA                            CLI
065A  B0 60                         MOV     AL,60H                  ; WRITE 8042 RAM COMMAND
065C  E8 0000 E                     CALL    C8042                   ; ISSUE THE COMMAND
065F  B0 45                         MOV     AL,45H                  ; SET SYSTEM FLAG - OUTBUF INT -
0661  E6 60                         OUT     PORT_A,AL               ; SYSTEM FLAG - PC 1 COMPATABILITY
                                                                    ; RESET INHIBIT OVER RIDE
                            ;------- DEGATE ADDRESS LINE 20

0663  B4 DD                         MOV     AH,DISABLE_BIT20        ; SET COMMAND IN AH
0665  E8 0000 E                     CALL    GATE_A20                ; ISSUE THE COMMAND

                            ;------- SETUP HARDWARE INT VECTOR TABLE LVL 0-7

0668  2B C0                         SUB     AX,AX                   ;
066A  8E C0                         MOV     ES,AX
066C  B9 0008                       MOV     CX,08                   ; GET VECTOR CNT
066F  0E                            PUSH    CS                      ; SETUP DS SEG REG
0670  1F                            POP     DS
0671  BE 0000 E                     MOV     SI,OFFSET VECTOR_TABLE
0674  BF 0020 R                     MOV     DI,OFFSET INT_PTR
0677  A5                    F7A:    MOVSW
0678  47                            INC     DI                      ; SKIP OVER SEGMENT
0679  47                            INC     DI
067A  E2 FB                         LOOP    F7A

                            ;------- SETUP HARDWARE INT VECTOR TABLE LVL 8-15 (VECTORS START AT INT 50H)

067C  2B C0                         SUB     AX,AX                   ;
067E  8E C0                         MOV     ES,AX
0680  B9 0008                       MOV     CX,08                   ; GET VECTOR CNT
0683  0E                            PUSH    CS                      ; SETUP DS SEG REG
0684  1F                            POP     DS
0685  BE 0000 E                     MOV     SI,OFFSET SLAVE_VECTOR_TABLE
0688  BF 01C0 R                     MOV     DI,OFFSET SLAVE_INT_PTR
068B  A5                    F7A1:   MOVSW
068C  47                            INC     DI                      ; SKIP OVER SEGMENT
068D  47                            INC     DI
068E  E2 FB                         LOOP    F7A1

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

                                    ASSUME  DS:ABS0
0690  2B C0                         SUB     AX,AX                           ; DS=0
0692  8E D8                         MOV     DS,AX
0694  C7 06 0008 R 0000 E           MOV     NMI_PTR,OFFSET NMI_INT          ; NMI INTERRUPT
069A  C7 06 0014 R 0000 E           MOV     INT5_PTR,OFFSET PRINT_SCREEN    ; PRINT SCREEN
06A0  C7 06 0062 R F600             MOV     BASIC_PTR+2,0F600H              ; SEGMENT FOR CASSETTE BASIC

                            ;-------- ZERO RESERVED VECTORS

06A6  BF 0180                       MOV     DI,60H*4                        ; INT 60 THRU 67 FILL WITH ZERO
06A9  B9 000E                       MOV     CX,14                           ; CLEAR 14 WORDS
06AC  C7 05 0000            F7A2:   MOV     WORD PTR DS:[DI],0
06B0  83 C7 02                      ADD     DI,2                            ; POINT TO NEXT LOCATION
06B3  E2 F7                         LOOP    F7A2                            ;

                            ;----- SETUP TIMER 0 TO BLINK LED IF MANUFACTURING TEST MODE
06B5  F6 06 0412 R 20               TEST    DATA_AREA[MFG_TST-DATA_base],LOOP_POST ; MFG. TEST MODE?
06BA  75 0A                         JNZ     F9
06BC  C7 06 0020 R 0000 E           MOV     INT_ADDR,OFFSET BLINK_INT       ; SETUP TIMER INTR TO BLINK LED
06C2  B0 FE                         MOV     AL,0FEH                         ; ENABLE TIMER INTERRUPT
06C4  E6 21                         OUT     INTA01,AL
06C6  FB                    F9:     STI                                     ; ALLOW INTERRUPTS

                                    ASSUME  DS:DATA
06C7  E8 0000 E                     CALL    DDS                     ; ESTABLISH DATA SEGMENT
                                                                    ;   THE OPERATING SYSTEM

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

06CA  81 3E 0072 R 1234             CMP     RESET_FLAG,1234H        ; SOFT RESET?
06D0  75 0E                         JNZ     F9A                     ; CONTINUE IF NOT
06D2  B9 00FF                       MOV     CX,0FFH                 ;
06D5  BA 03F6                       MOV     DX,03F6H                ;
06D8  B0 04                         MOV     AL,04H                  ; RESET
06DA  EE                            OUT     DX,AL                   ;
06DB  E2 FE                 F9_A:   LOOP    F9_A                    ; HOLD RESET
06DD  2A C0                         SUB     AL,AL                   ;
06DF  EE                            OUT     DX,AL                   ; REMOVE RESET

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

06E0  B0 3C                 F9A:    MOV     AL,3CH                  ;<><><><><><><><><><><><><><>
06E2  E6 80                         OUT     MFG_PORT,AL             ;<><><>CHECKPOINT 3C <><><><>

06E4  B0 02                         MOV     AL,02H                  ; SET DATA RATE TO 250 K BITS / SEC
06E6  BA 03F7                       MOV     DX,3F7H                 ;
06E9  EE                            OUT     DX,AL                   ;
06EA  F6 06 0010 R 01               TEST    BYTE PTR EQUIP_FLAG,01H ; DISKETTE PRESENT?
06EF  74 4F                         JZ      F15
06F1  F6 06 0012 R 20               TEST    MFG_TST,LOOP_POST       ; MFG JUMPER INSTALLED?
06F6  74 48                         JZ      F15                     ; GO IF YES
06F8                        F10:                                    ; DISK_TEST:
06F8  E4 21                         IN      AL,INTA01
06FA  EB 00                         JMP     SHORT $+2               ; IO DELAY
06FC  24 BF                         AND     AL,0BFH                 ; ENABLE DISKETTE INTERRUPTS
06FE  E6 21                         OUT     INTA01,AL
0700  B4 00                         MOV     AH,0                    ; RESET NEC FDC
0702  8A D4                         MOV     DL,AH                   ; SET FOR DRIVE 0
0704  CD 13                         INT     13H                     ; VERIFY STATUS AFTER RESET
0706  F6 C4 FF                      TEST    AH,0FFH                 ; STATUS OK?
0709  75 24                         JNZ     F13                     ; NO - FDC FAILED

                            ;----- TURN DRIVE 0 MOTOR ON

070B  BA 03F2                       MOV     DX,03F2H                ; GET ADDR OF FDC CARD
070E  B0 1C                         MOV     AL,1CH                  ; TURN MOTOR ON, EN DMA/INT
0710  EE                            OUT     DX,AL                   ; WRITE FDC CONTROL REG
0711  2B C9                         SUB     CX,CX
0713  B2 0C                         MOV     DL,12                   ; WAIT 1 SECOND
0715                        F11:                                    ; MOTOR_WAIT:
```

SECTION 5

```
0715  E2 FE                        LOOP    F11                    ; WAIT FOR 1 SECOND
0717  FE CA                        DEC     DL                     ; DECREMENT OUTTER LOOP
0719  75 FA                        JNZ     F11                    ;

071B  33 D2                        XOR     DX,DX                  ; SELECT DRIVE 0
071D  B5 01                        MOV     CH,1                   ; SELECT TRACK 1
071F  88 16 003E R                 MOV     SEEK_STATUS,DL
0723  E8 0000 E                    CALL    SEEK                   ; RECALIBRATE DISKETTE
0726  72 07                        JC      F13                    ; GO TO ERR SUBROUTINE IF ERR
0728  B5 22                        MOV     CH,34                  ; SELECT TRACK 34
072A  E8 0000 E                    CALL    SEEK                   ; SEEK TO TRACK 34
072D  73 0B                        JNC     F14                    ; OK, TURN MOTOR OFF
072F                       F13:                                   ; DSK_ERR:

072F  80 0E 0016 R 40              OR      MFG_ERR_FLAG+1,DSK_FAIL ;<><><><><><><><><><><><><><>
                                                                  ;<><> DISKETTE FAILED<><><><><>
0734  BE 0000 E                    MOV     SI,OFFSET F3           ; GET ADDR OF MSG
0737  E8 0000 E                    CALL    E_MSG                  ; GO PRINT ERROR MSG

                           ;----- TURN DRIVE 0 MOTOR OFF

073A                       F14:                                   ; DR0_OFF:
073A  B0 0C                        MOV     AL,0CH                 ; TURN DRIVE 0 MOTOR OFF
073C  BA 03F2                      MOV     DX,03F2H               ; FDC CTL ADDRESS
073F  EE                           OUT     DX,AL

                           ;------ SETUP KEYBOARD PARAMETERS

0740  C6 06 006B R 00      F15:    MOV     INTR_FLAG,00H          ; SET STRAY INTERRUPT FLAG = 00
0745  BE 001E R                    MOV     SI,OFFSET KB_BUFFER    ; SETUP KEYBOARD PARAMETERS
0748  89 36 001A R                 MOV     BUFFER_HEAD,SI
074C  89 36 001C R                 MOV     BUFFER_TAIL,SI
0750  89 36 0080 R                 MOV     BUFFER_START,SI
0754  83 C6 20                     ADD     SI,32                  ;DEFAULT BUFFER OF 32 BYTES
0757  89 36 0082 R                 MOV     BUFFER_END,SI

                           ;-------- SET PRINTER TIMEOUT DEFAULT

075B  BF 0078 R                    MOV     DI,OFFSET PRINT_TIM_OUT ;SET DEFAULT PRINTER TIMEOUT
075E  1E                           PUSH    DS
075F  07                           POP     ES
0760  B8 1414                      MOV     AX,1414H               ; DEFAULT=20
0763  AB                           STOSW
0764  AB                           STOSW

                           ;-------- SET 4S232 DEFAULT

0765  B8 0101                      MOV     AX,0101H               ;RS232 DEFAULT=01
0768  AB                           STOSW
0769  AB                           STOSW

                           ;-------- ENABLE TIMER INTERRUPTS

076A  E4 21                        IN      AL,INTA01
076C  24 FE                        AND     AL,0FEH                ; ENABLE TIMER AND KB INTS
076E  EB 00                        JMP     SHORT $+2              ; IO DELAY
0770  E6 21                        OUT     INTA01,AL

                           ;------- CHECK CMOS BATTERY/CHECKSUM

0772  F6 06 0012 R 20              TEST    MFG_TST,LOOP_POST      ; MFG JUMPER?
0777  75 03                        JNZ     B1_OK                  ; GO IF NOT
0779  E9 0858 R                    JMP     F15C                   ; BYPASS IF YES
077C  B0 8E                B1_OK:  MOV     AL,DIAG_STATUS         ;
077E  E6 70                        OUT     CMOS_PORT,AL           ;
0780  EB 00                        JMP     SHORT $+2              ; IO DELAY
0782  E4 71                        IN      AL,CMOS_PORT+1         ;
0784  24 E0                        AND     AL,0E0H                ; BAD BATTERY, CHK SUM, OR MIN CONFIG?
0786  74 16                        JZ      C_OK                   ; GO IF NOT
0788  A8 80                        TEST    AL,80H                 ; BATTERY BAD?
078A  BE 0000 E                    MOV     SI,OFFSET CM1          ; PRELOAD BATTERY MSG
078D  74 06                        JZ      B2_OK                  ; GO IF BATTERY OK
078F  E8 0000 E                    CALL    E_MSG                  ; PRINT BATTERY MSG
0792  EB 62 90                     JMP     H_OK1A                 ; CONTINUE(BYPASS CLOCK ETC)
0795                       B2_OK:
0795  BE 0000 E                    MOV     SI,OFFSET CM2          ; PRE LOAD CKSUM BAD
0798  E8 0000 E            B_OK:   CALL    E_MSG                  ; PRINT MSG
079B  EB 59 90                     JMP     H_OK1A                 ; BYPASS CLOCK TEST-MEM SIZE

                           ;------- TEST CLOCK UPDATING

079E  B3 03                C_OK:   MOV     BL,03H                 ; OUTER LOOP COUNT
07A0  2B C9                D_OK:   SUB     CX,CX                  ; INNER LOOP COUNT
07A2  B0 8A                E_OK:   MOV     AL,CLK_UP              ; GET THE CLOCK UPDATE BYTE
07A4  E6 70                        OUT     CMOS_PORT,AL           ;
07A6  EB 00                        JMP     SHORT $+2              ; IO DELAY
07A8  E4 71                        IN      AL,CMOS_PORT+1         ;
07AA  A8 80                        TEST    AL,80H                 ; CHECK FOR UPDATE IN PROGRESS
07AC  75 25                        JNZ     G_OK                   ; GO IF YES
07AE  E2 F2                        LOOP    E_OK                   ; TRY AGAIN
07B0  FE CB                        DEC     BL                     ; DEC OUTER LOOP
07B2  75 EC                        JNZ     D_OK                   ; TRY AGAIN
07B4  BE 0000 E            F_OK:   MOV     SI,OFFSET CM3          ; PRINT MSG
07B7  E8 0000 E                    CALL    E_MSG                  ;

                           ;------- SET CMOS DIAG_STATUS 04 (CLOCK ERROR)

07BA  B0 8E                        MOV     AL,DIAG_STATUS         ; SET CLOCK ERROR
07BC  E6 70                        OUT     CMOS_PORT,AL           ;
07BE  86 C4                        XCHG    AL,AH                  ; SAVE STATUS ADDRESS
07C0  EB 00                        JMP     SHORT $+2              ; IO DELAY
07C2  E4 71                        IN      AL,CMOS_PORT+1         ; GET THE CURRENT STATUS
07C4  0C 04                        OR      AL,CMOS_CLK_FAIL       ; SET NEW STATUS
07C6  86 C4                        XCHG    AL,AH                  ; GET STATUS ADDR AND SAVE NEW STATUS
07C8  E6 70                        OUT     CMOS_PORT,AL           ;
07CA  86 C4                        XCHG    AL,AH                  ;
07CC  EB 00                        JMP     SHORT $+2              ; IO DELAY
07CE  E6 71                        OUT     CMOS_PORT+1,AL         ;
07D0  EB 12 90                     JMP     H_OK                   ; CONTINUE

                           ;-------- CHECK CLOCK UDATE

07D3  B9 0258              G_OK:   MOV     CX,600                 ; LOOP COUNT
07D6  B0 8A                I_OK:   MOV     AL,CLK_UP              ; CHECK FOR OPPOSITE STATE
07D8  E6 70                        OUT     CMOS_PORT,AL           ;
07DA  EB 00                        JMP     SHORT $+2              ; IO DELAY
07DC  E4 71                        IN      AL,CMOS_PORT+1         ;
07DE  A8 80                        TEST    AL,80H                 ;
07E0  E0 F4                        LOOPNZ  I_OK                   ; TRY AGAIN
07E2  E3 D0                        JCXZ    F_OK                   ; PRINT ERROR IF TIMEOUT

                           ;-------- CHECK MEMORY SIZE DETERMINED = CONFIG

07E4                       H_OK:
07E4  B0 8E                        MOV     AL,DIAG_STATUS         ; GET THE STATUS BYTE
07E6  E6 70                        OUT     CMOS_PORT,AL           ;
```

```
07E8  EB 00                        JMP    SHORT $+2               ; IO DELAY
07EA  E4 71                        IN     AL,CMOS_PORT+1          ;
07EC  A8 10                        TEST   AL,W_MEM_SIZE           ; WAS THE CONFIG=MEM_SIZE_DETERMINED?
07EE  74 06                        JZ     H_OK1A                  ; GO IF YES

                           ;-------- MEMORY SIZE ERROR

07F0  BE 0000 E                    MOV    SI,OFFSET E1_A          ; PRINT SIZE ERROR
07F3  E8 0000 E                    CALL   E_MSG                   ; DISPLAY ERROR

                           ;-------- CHECK FOR CRT ERROR

07F6  80 3E 0015 R 0C      H_OK1A: CMP    MFG_ERR_FLAG,0CH        ; CHECK FOR MONO CRT ERROR
07FB  BE 0000 E                    MOV    SI,OFFSET E1_B          ; PRELOAD MONO CRT ERROR
07FE  74 0A                        JZ     H_OK1B                  ; GO IF YES

0800  80 3E 0015 R 0D              CMP    MFG_ERR_FLAG,0DH        ; CHECK FOR COLOR CRT ERROR
0805  75 06                        JNZ    J_OK                    ; CONTINUE IF NOT
0807  BE 0000 E                    MOV    SI,OFFSET E1_C          ; CRT ERROR MSG
080A  E8 0000 E            H_OK1B: CALL   E_MSG                   ;


                                  ;======================================
                           ;------- CHECK FOR COMBO HARD FILE/DISKETTE CARD
                                  ;======================================
080D                       J_OK:
080D  B3 0F                        MOV    BL,0FH                  ; OUTTER LOOP COUNT WAIT FOR BUSY OFF
080F  2B C9                        SUB    CX,CX                   ;
0811  BA 01F7                      MOV    DX,01F7H                ; HARD FILE STATUS PORT
0814  EC                   J_OK1:  IN     AL,DX                   ; GET THE STATUS
0815  A8 80                        TEST   AL,080H                 ; IS THE CONTROLLER BUSY?
0817  74 0D                        JZ     J_OK2                   ; CONTINUE IF NOT
0819  E2 F9                        LOOP   J_OK1                   ; TRY AGAIN
081B  FE CB                        DEC    BL                      ; DECREMENT OUTTER LOOP
081D  75 F5                        JNZ    J_OK1                   ; TRY AGAIN IF NOT ZERO
081F  24 0C                        AND    AL,0CH                  ; BITS 2 & 3 = 0 IF COMBO CARD
0821  74 1A                        JZ     J_OK3                   ; GO IF YES
0823  EB 33 90                     JMP    F15C                    ; NO COMBO CARD

0826  BA 01F4              J_OK2:  MOV    DX,1F4H                 ; VERIFY COMBO CARD
0829  B0 55                        MOV    AL,055H                 ;  WRITE TO THE CYL BYTE
082B  EE                           OUT    DX,AL                   ;
082C  EB 00                        JMP    SHORT $+2               ; IO DELAY
082E  EC                           IN     AL,DX                   ; CHECK DATA WRITTEN = DATA READ
082F  3C 55                        CMP    AL,055H                 ;
0831  75 25                        JNZ    F15C                    ; GO IF NOT
0833  B0 AA                        MOV    AL,0AAH                 ; WRITE ANOTHER PATTERN
0835  EE                           OUT    DX,AL                   ;
0836  EB 00                        JMP    SHORT $+2               ; IO DELAY
0838  EC                           IN     AL,DX                   ;
0839  3C AA                        CMP    AL,0AAH                 ; IS DATA PATTERN THE SAME?
083B  75 1B                        JNZ    F15C                    ; GO IF NOT

083D  C6 06 008F R 01      J_OK3:  MOV    HF_CNTRL,DUAL           ; SET THE HF/FLOPPY SWITCH ON

                           ;------- INITIALIZE FLOPPY FOR DRIVE TYPE

0842  B0 3D                        MOV    AL,3DH                  ;<><><><><><><><><><><><><><>
0844  E6 80                        OUT    MFG_PORT,AL             ;<><><>CHECKPOINT 3D <><><><>
0846  E8 0000 E                    CALL   DSKETTE_SETUP           ; INITIALIZE FLOPPY

                           ;-------- CHECK FOR 2ND DISKETTE DRIVE
0849  E8 0000 E                    CALL   DDS                     ; INSURE DATA SEGMENT
084C  80 3E 0091 R 00              CMP    DSK_STATE+1,0           ; IS THERE A DRIVE 2 ATTACHED?
0851  74 05                        JZ     F15C                    ; GO IF NOT
0853  80 0E 0010 R 40              OR     BYTE PTR EQUIP_FLAG,40H ; SET SECOND DRIVE INSTALLED

                           ;------- INITIALIZE HARD FILE

0858  B0 3E                F15C:   MOV    AL,3EH                  ;<><><><><><><><><><><><><><>
085A  E6 80                        OUT    MFG_PORT,AL             ;<><><>CHECKPOINT 3E <><><><>

085C  B0 8E                        MOV    AL,DIAG_STATUS          ; GET THE CMOS STATUS
085E  E6 70                        OUT    CMOS_PORT,AL            ;
0860  EB 00                        JMP    SHORT $+2               ;
0862  E4 71                        IN     AL,CMOS_PORT+1          ;
0864  A8 C0                        TEST   AL,0C0H                 ; BATTERY/CHECKSUM OK
0866  75 0F                        JNZ    ROM_SCAN1               ; BYPASS DISK SETUP IF NOT

0868  B0 92                        MOV    AL,HD_FILE_TYPE         ; INSURE CMOS DEFINES THE TYPE OF HARD FILE
086A  E6 70                        OUT    CMOS_PORT,AL            ;
086C  EB 00                        JMP    SHORT $+2               ;
086E  E4 71                        IN     AL,CMOS_PORT+1          ;
0870  3C 00                        CMP    AL,0H                   ; INSURE TYPE IS DEFINED
0872  74 03                        JZ     ROM_SCAN1               ; BYPASS DISK SETUP IF NOT

0874  E8 0000 E                    CALL   DISK_SETUP              ; INITIALIZE HARD FILE

                           ;-------------------------------------------------------------
                           ; TEST.22                                                    :
                           ; CHECK FOR OPTIONAL ROM FROM C800->E000 IN 2K BLOCKS        :
                           ;       (A VALID MODULE HAS '55AA' IN THE FIRST 2 LOCATIONS:
                           ;       LENGTH INDICATOR (LENGTH/512) IN THE 3RD LOCATION    :
                           ;       AND TEST/INIT. CODE STARTING IN THE 4TH LOCATION)    :
                           ;-------------------------------------------------------------
0877                       ROM_SCAN1:
0877  FB                           STI                                    ; ALLOW INTERRUPTS
0878  B0 3B                        MOV    AL,3BH                          ;<><><><><><><><><><><><><>
087A  E6 80                        OUT    MFG_PORT,AL                     ;<><><>CHECKPOINT 3B <><><><>
087C  E8 0000 E                    CALL   DDS                             ; SET REAL MODE DATA SEGMENT
087F  B0 0A                        MOV    AL,10                           ; LINE FEED ON CRT
0881  E8 0000 E                    CALL   PRT_HEX                         ;
0884                       ROM_SCAN:

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

0884  2A C0                        SUB    AL,AL                   ;
0886  E6 D2                        OUT    DMA18+2,AL              ; SEND ZERO TO MASK REG
0888  EB 00                        JMP    SHORT $+2               ;
088A  E6 D4                        OUT    DMA18+4,AL              ; SEND ZERO TO REQ REG
088C  BA C800                      MOV    DX,0C800H               ; SET BEGINNING ADDRESS
088F                       ROM_SCAN2:
088F  8E DA                        MOV    DS,DX                   ;
0891  2B DB                        SUB    BX,BX                   ; SET BX=0000
0893  8B 07                        MOV    AX,[BX]                 ; GET 1ST WORD FROM MODULE
0895  53                           PUSH   BX                      ;
0896  5B                           POP    BX                      ; BUS SETTLING
0897  3D AA55                      CMP    AX,0AA55H               ; = TO ID WORD?
089A  75 06                        JNZ    NEXT_ROM                ; PROCEED TO NEXT ROM IF NOT
089C  E8 0000 E                    CALL   ROM_CHECK               ; GO CHECK OUT MODULE
089F  EB 05 90                     JMP    ARE_WE_DONE             ; CHECK FOR END OF ROM SPACE
08A2                       NEXT_ROM:
08A2  81 C2 0080                   ADD    DX,0080H                ; POINT TO NEXT 2K ADDRESS
08A6                       ARE_WE_DONE:
08A6  81 FA E000                   CMP    DX,0E000H               ; AT E0000 YET?
```

SECTION 5

```
08AA  7C E3                     JL      ROM_SCAN2               ; GO CHECK ANOTHER ADD. IF NOT

                         ENDIF
                         ;-------- TEST FOR KEYBOARD LOCKED

08AC  E8 0000 E                 CALL    DDS                     ; SET DATA SEGMENT
08AF  E4 64                     IN      AL,STATUS_PORT          ; IS KEYBOARD UNLOCKED?
08B1  24 10                     AND     AL,KYBD_INH             ;
08B3  74 03                     JZ      KEY1
08B5  EB 0C 90                  JMP     KEY10                   ; GO IF OFF
08B8                     KEY1:
08B8  80 0E 0016 R 80           OR      MFG_ERR_FLAG+1,KEY_FAIL ;<><><><><><><><><><><><><><><><><><>
                                                                ;<><> KEYBOARD IS LOCKED <><><>

                         ELSE

08BD                     KEY9:
                                ASSUME  DS:DATA

08BD  BE 0000 E                 MOV     SI,OFFSET LOCK          ; PRINT LOCKED MESSAGE  (302)
08C0  E8 0000 E                 CALL    E_MSG                   ;
                         ENDIF

08C3                     KEY10:
                                ;===================
                         ;-------- SETUP PRINTER_BASE
                                ;===================

08C3  BF 0000 E                 MOV     DI,OFFSET F4            ; PRT_SRC_TBL
08C6  BE 0000                   MOV     SI,0
08C9                     F16:                                   ; PRT_BASE:
08C9  2E: 8B 15                 MOV     DX,CS:[DI]              ; GET PRINTER BASE ADDR
08CC  B0 AA                     MOV     AL,0AAH                 ; WRITE DATA TO PORT A
08CE  EE                        OUT     DX,AL
08CF  EB 00                     JMP     SHORT $+2               ; IO DELAY
08D1  1E                        PUSH    DS                      ; BUS SETTLING
08D2  EC                        IN      AL,DX                   ; READ PORT A
08D3  1F                        POP     DS
08D4  3C AA                     CMP     AL,0AAH                 ; DATA PATTERN SAME
08D6  75 06                     JNE     F17                     ; NO - CHECK NEXT PRT CD
08D8  89 94 0008 R              MOV     PRINTER_BASE[SI],DX     ; YES - STORE PRT BASE ADDR
08DC  46                        INC     SI                      ; INCREMENT TO NEXT WORD
08DD  46                        INC     SI
08DE                     F17:
08DE  47                        INC     DI                      ; POINT TO NEXT BASE ADDR
08DF  47                        INC     DI
08E0  81 FF 0000 E              CMP     DI,OFFSET F4E           ; ALL POSSIBLE ADDRS CHECKED?
08E4  75 E3                     JNE     F16                     ; PRT_BASE
                                ;=============
                         ;------- SETUP RS232
                                ;=============
08E6  BB 0000                   MOV     BX,0                    ; POINTER TO RS232 TABLE
08E9  BA 03FA                   MOV     DX,3FAH                 ; CHECK IF RS232 CD 1 ATTCH?
08EC  EC                        IN      AL,DX                   ; READ INTR ID REG
08ED  A8 F8                     TEST    AL,0F8H
08EF  75 08                     JNZ     F18
08F1  C7 87 0000 R 03F8         MOV     RS232_BASE[BX],3F8H     ; SETUP RS232 CD #1 ADDR
08F7  43                        INC     BX
08F8  43                        INC     BX
08F9  BA 02FA            F18:   MOV     DX,2FAH                 ; CHECK IF RS232 CD 2 ATTCH
08FC  EC                        IN      AL,DX                   ; READ INTERRUPT ID REG
08FD  A8 F8                     TEST    AL,0F8H
08FF  75 08                     JNZ     F19                     ; BASE_END
0901  C7 87 0000 R 02F8         MOV     RS232_BASE[BX],2F8H     ; SETUP RS232 CD #2
0907  43                        INC     BX
0908  43                        INC     BX
                              ;==============================================================
                         ;----- SET UP EQUIP_FLAG TO INDICATE NUMBER OF PRINTERS AND RS232 CARDS
                              ;==============================================================
0909                     F19:                                   ; BASE_END:
0909  8B C6                     MOV     AX,SI                   ; SI HAS 2* NUMBER OF RS232
090B  B1 03                     MOV     CL,3                    ; SHIFT COUNT
090D  D2 C8                     ROR     AL,CL                   ; ROTATE RIGHT 3 POSITIONS
090F  0A C3                     OR      AL,BL                   ; OR IN THE PRINTER COUNT
0911  A2 0011 R                 MOV     BYTE PTR EQUIP_FLAG+1,AL        ; STORE AS SECOND BYTE

                                ;================================
                         ;------- TEST FOR ANY ERRORS (BP NOT ZERO)
                                ;================================

0914                     K_OK:
                         ;-------- CLEAR KEYBOARD STATE FLAGS

0914  2B C0                     SUB     AX,AX                   ; RESET ALL KEYBOARD STATE FLAGS
0916  A3 0017 R                 MOV     WORD PTR KB_FLAG,AX     ;

                         ;-------- ENABLE KEYBOARD INTERRUPTS

0919  E4 21                     IN      AL,INTA01
091B  24 FD                     AND     AL,0FDH                 ; ENABLE TIMER AND KB INTS
091D  EB 00                     JMP     SHORT $+2               ; IO DELAY
091F  E6 21                     OUT     INTA01,AL

0921  C6 06 0015 R 00           MOV     BYTE PTR MFG_ERR_FLAG,0 ; CLEAR MFG ERROR FLAG
0926  83 FD 00                  CMP     BP,0000H                ; CHECK FOR BP= NON-ZERO
                                                                ; (ERROR HAPPENED)
0929  74 3D                     JE      F15A_0                  ; CONTINUE IF NO ERROR

092B  80 3E 0072 R 64           CMP     BYTE PTR RESET_FLAG,64H ; MFG RUN IN MODE?
0930  75 08                     JNZ     ERR_WAIT                ; GO IF NOT

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

0932  C6 06 0015 R AA           MOV     BYTE PTR MFG_ERR_FLAG,0AAH      ; INDICATE ERROR
0937  EB 2F 90                  JMP     F15A_0                          ; CONTINUE
093A                     ERR_WAIT:
093A  BA 0002                   MOV     DX,2                    ; 2 SHORT BEEPS (ERROR)
093D  E8 0000 E                 CALL    ERR_BEEP

0940  E4 64                     IN      AL,STATUS_PORT          ; CHECK IF RESUME MSG TO BE DISPLAYED
0942  24 10                     AND     AL,KYBD_INH             ;
0944  BE 0000 E                 MOV     SI,OFFSET F3D           ; RESUME ERROR MSG
0947  75 09                     JNZ     ERR_WAIT2
0949  BE 0000 E                 MOV     SI,OFFSET F3D1          ; ERROR MSG FOR KEYBOARD LOCKED
094C  E8 0000 E                 CALL    P_MSG                   ;
094F  BE 0000 E                 MOV     SI,OFFSET F3D           ; RESUME MSG
0952                     ERR_WAIT2:
0952  E8 0000 E                 CALL    P_MSG

                         ;-------- INIT PRINTER  (ALT DISPLAY DEVICE)

0955  B4 01                     MOV     AH,1                    ;
0957  2B D2                     SUB     DX,DX                   ; FIRST PRINTER
0959  CD 17                     INT     17H                     ;
095B                     ERR_WAIT1:
095B  B0 3F                     MOV     AL,3FH                  ;<><><><><><><><><><><><><><><><><><>
095D  E6 80                     OUT     MFG_PORT,AL             ;<><><>CHECKPOINT 3F <><><><><>
```

```
095F  B4 00                          MOV   AH,00
0961  CD 16                          INT   16H                       ; WAIT FOR 'F1' KEY
0963  80 FC 3B                       CMP   AH,3BH
0966  75 F3                          JNE   ERR_WAIT1
0968                       F15A_0:
0968  F6 06 0012 R 20                TEST  MFG_TST,LOOP_POST         ; MFG BURN IN MODE
096D  75 03                          JNZ   F15A                      ; GO IF NOT
096F  E9 0000 E                      JMP   START_1                   ; GO LOOP POST
0972  80 3E 0072 R 64      F15A:     CMP   BYTE PTR RESET_FLAG,64H   ; MFG RUN IN?
0977  74 06                          JZ    F15B                      ; BYPASS BEEP IF YES

0979  BA 0001                        MOV   DX,1                      ; 1 SHORT BEEP (NO ERRORS)
097C  E8 0000 E                      CALL  ERR_BEEP

097F  2A E4                F15B:     SUB   AH,AH                     ; CLEAR FLAGS
0981  A0 0049 R                      MOV   AL,CRT_MODE
0984  CD 10                          INT   10H                       ; CLEAR SCREEN

                           ;-------- CLEAR DESCRIPTOR TABLES

0986  B9 01F4              F20:      MOV   CX,0500                   ; CLEAR 1K
0989  BF D0A0                        MOV   DI,SYS_IDT_LOC            ; POINT ES TO START OF DESCRIPTORS
098C  2B C0                          SUB   AX,AX
098E  8E C0                          MOV   ES,AX                     ;
0990  26: 89 05            F20_A:    MOV   ES:[DI],AX                ; CLEAR
0993  83 C7 02                       ADD   DI,2                      ; POINT TO NEXT LOCATION
0996  E2 F8                          LOOP  F20_A                     ; CONTINUE TILL DONE

                                     ;===============
                           ;-------- SET TIME OF DAY
                                     ;===============

0998  E8 0000 E                      CALL  SET_TOD                   ;

                           ;-------- SET SYSTEM STACK

099B  B8 ---- R                      MOV   AX,STACK                  ; GET THE STACK SEGMENT
099E  8E D0                          MOV   SS,AX                     ;
09A0  BC 0100 R                      MOV   SP,OFFSET TOS             ;

                           ;------- ENABLE HARDWARE INTERRUPT IF MATH PROCESSOR (X287)

09A3  B0 40                          MOV   AL,40H                    ;<><><><><><><><><><><><><>
09A5  E6 80                          OUT   MFG_PORT,AL               ;<><><>CHECKPOINT 40 <><><><>
09A7  A1 0067 R                      MOV   AX,IO_ROM_INIT            ; TEMP STORAGE
09AA  50                             PUSH  AX
09AB  2B C0                          SUB   AX,AX                     ; CLEAR IO_ROM_INIT
09AD  A3 0067 R                      MOV   IO_ROM_INIT,AX            ;
09B0  DB E3                          ESC   28,BX                     ;
09B2  33 C0                          XOR   AX,AX                     ;
09B4  D9 3E 0067 R                   ESC   15,IO_ROM_INIT            ;
                                     PUSHA                           ; TIME FOR 287 TO RESPOND
09B8  60                   +         DB    060H
                                     POPA                            ;
09B9  61                   +         DB    061H
09BA  81 26 0067 R 1F3F              AND   IO_ROM_INIT,01F3FH        ; CLEAR UNUSED 287 BITS
09C0  81 3E 0067 R 033F              CMP   IO_ROM_INIT,0033FH        ; IS THE 287 INSTALLED?
09C6  75 24                          JNZ   NO_287                    ; GO IF MATH PROCESSOR IS NOT INSTALLED

09C8  9B                             WAIT
09C9  DD 3E 0067 R                   ESC   02FH,IO_ROM_INIT          ; STORE THE STATUS WORD
                                     PUSHA                           ; TIME FOR 287 TO RESPOND
09CD  60                   +         DB    060H
                                     POPA                            ;
09CE  61                   +         DB    061H
09CF  F7 06 0067 R B8BF              TEST  IO_ROM_INIT,0B8BFH        ; ALL BITS SHOULD BE OFF
09D5  75 15                          JNZ   NO_287                    ; GO IF NOT INSTALLED

09D7  E4 A1                          IN    AL,INTB01                 ; GET THE SLAVE INT MASK
09D9  24 DF                          AND   AL,0DFH                   ; ENABLE 287 INTERRUPTS
09DB  EB 00                          JMP   SHORT $+2                 ; IO DELAY
09DD  E6 A1                          OUT   INTB01,AL                 ;

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

09DF  E4 21                          IN    AL,INTA01                 ; GET THE CURRENT MASK
09E1  24 FB                          AND   AL,0FBH                   ;
09E3  EB 00                          JMP   SHORT $+2                 ; IO DELAY
09E5  E6 21                          OUT   INTA01,AL                 ;
09E7  80 0E 0010 R 02                OR    BYTE PTR EQUIP_FLAG,02H   ; SET 287 BIT ON
09EC                       NO_287:

09EC  58                             POP   AX                        ; RESTORE IO_ROM_INIT
09ED  A3 0067 R                      MOV   IO_ROM_INIT,AX            ;

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

09F0  80 3E 0072 R 64                CMP   BYTE PTR RESET_FLAG,64H   ; IS THE THE MFG RUN-IN TEST?
09F5  75 03                          JNZ   END_287                   ; GO IF NOT
09F7  EB 63 90                       JMP   SHUT4                     ; BOOT LOAD IF YES

                           ;-------- UNMASK SLAVE HARDWARE INT 9 (LEVEL 71)
09FA                       END_287:
09FA  E4 A1                          IN    AL,INTB01                 ; GET THE CURRENT MASK
09FC  24 FD                          AND   AL,0FDH                   ;
09FE  EB 00                          JMP   SHORT $+2                 ; IO DELAY
0A00  E6 A1                          OUT   INTB01,AL                 ; SET NEW MASK

                           ;------------------------------------------------------------
                           ; TEST FOR SYSTEM CODE AT SEGMENT E000:0
                           ;    FIRST WORD = AA55H
                           ;    LAST BYTE = CHECKSUM
                           ;    ENTRY POINT = FIRST BYTE + 3
                           ; IF TEST IS SUCCESSFUL A CALL FAR TO THE ENTRY POINT IS EXCUTED
                           ;------------------------------------------------------------
0A02  B0 41                          MOV   AL,41H                    ;<><><><><><><><><><><><><>
0A04  E6 80                          OUT   MFG_PORT,AL               ;<><><>CHECKPOINT 41 <><><><>

0A06  B0 AD                          MOV   AL,CMOS_END               ; INSURE NMI OFF
0A08  E6 70                          OUT   CMOS_PORT,AL              ;

                           ENDIF

0A0A  C6 06 0072 R 00                MOV   BYTE PTR RESET_FLAG,0     ; CLEAR FLAG
0A0F  B8 E000                        MOV   AX,0E000H                 ; SEGMENT OF SYSTEM CODE
0A12  8E C0                          MOV   ES,AX                     ;
0A14  2B FF                          SUB   DI,DI
0A16  26: 8B 05                      MOV   AX,ES:[DI]                ; CHECK FOR AA55
0A19  53                             PUSH  BX                        ; BUS SETTLE
0A1A  5B                             POP   BX
0A1B  3D AA55                        CMP   AX,0AA55H                 ;
0A1E  9C                             PUSHF                           ; SAVE FLAGS
0A1F  26: 89 05                      MOV   ES:[DI],AX                ; CLEAR POSSIBLE PARITY CHECK
0A22  E4 61                          IN    AL,PORT_B                 ;
0A24  0C 0C                          OR    AL,RAM_PAR_OFF            ; TOGGLE IO/PAR CHECK ENABLE
```

SECTION 5

```
0A26  EB 00                     JMP    SHORT $+2              ; IO DELAY
0A28  E6 61                     OUT    PORT_B,AL              ;
0A2A  24 F3                     AND    AL,RAM_PAR_ON          ;
0A2C  EB 00                     JMP    SHORT $+2              ; IO DELAY
0A2E  E6 61                     OUT    PORT_B,AL              ;
0A30  9D                        POPF                          ; RESTORE FLAGS
0A31  75 29                     JNZ    SHUT4                  ; CONTINUE

                        ;------- CHECKSUM SYSTEM CODE

0A33  1E                        PUSH   DS
0A34  06                        PUSH   ES                     ; SET SEGMENT TO TEST
0A35  1F                        POP    DS                     ;
0A36  2B DB                     SUB    BX,BX                  ; STARTING OFFSET
0A38  E8 0000 E                 CALL   ROS_CHECKSUM           ;
0A3B  1F                        POP    DS                     ; RESTORE DATA SEGMENT
0A3C  75 1E                     JNZ    SHUT4                  ; GO IF CHECKSUM NOT OK

                        ;-------- ENABLE NMI AND IO/PAR CHECKS

0A3E  B0 2D                     MOV    AL,2DH                 ; ENABLE NMI
0A40  E6 70                     OUT    CMOS_PORT,AL           ;

0A42  E4 61                     IN     AL,PORT_B              ; ENABLE PARITY
0A44  EB 00                     JMP    SHORT $+2              ; IO DELAY
0A46  24 F3                     AND    AL,RAM_PAR_ON          ; ENABLE RAM PCK AND IO CH
0A48  E6 61                     OUT    PORT_B,AL

0A4A  C7 06 0067 R 0003         MOV    DS:IO_ROM_INIT,0003H   ; SET THE OFFSET
0A50  8C 06 0069 R              MOV    DS:IO_ROM_SEG,ES       ; SET THE SEGMENT

0A54  B0 42                     I V    AL,42H                 ;<><><><><><><><><><><><><>
0A56  E6 80                     OUT    MFG_PORT,AL            ;<><><>CHECKPOINT 42 <><><>

                        ;-------- EXIT TO SYSTEM CODE

0A58  FF 1E 0067 R              CALL   DWORD PTR DS:IO_ROM_INIT       ; GO TO SYSTEM CODE
                                                                      ; VIA CALL

                        ;----- ENABLE NMI INTERRUPTS + ENTRY FROM SHUTDOWN WITH BOOT REQUEST

0A5C  B0 2D             SHUT4:  MOV    AL,2DH                 ; ENABLE NMI
0A5E  E6 70                     OUT    CMOS_PORT,AL           ;

0A60  E4 61                     IN     AL,PORT_B              ; ENABLE PARITY
0A62  EB 00                     JMP    SHORT $+2              ; IO DELAY
0A64  24 F3                     AND    AL,RAM_PAR_ON          ; ENABLE RAM PCK AND IO CH
0A66  E6 61                     OUT    PORT_B,AL

0A68  B0 43                     MOV    AL,43H                 ;<><><><><><><><><><><><><>
0A6A  E6 80                     OUT    MFG_PORT,AL            ;<><><>CHECKPOINT 43 <><><>

                        ENDIF

0A6C  CD 19                     INT    19H                    ; GO TO BOOT LOADER
                        ENDIF
0A6E                    POST2      ENDP
0A6E                    CODE       ENDS
                                   END
```

```
                                      TITLE 09-26-83 TEST3     POST UTILITIES
                                      .LIST
                                      PUBLIC  POST3
                                      PUBLIC  ROS_CHECKSUM
                                      PUBLIC  BLINK_INT
                                      PUBLIC  ROM_CHECK
                                      PUBLIC  XPC_BYTE
                                      PUBLIC  PRT_HEX
                                      PUBLIC  XLAT_PR
                                      PUBLIC  PROT_PRT_HEX
                                      PUBLIC  PROC_SHUTDOWN

                                    C INCLUDE SEGMENT.SRC
 0000                               C CODE SEGMENT BYTE PUBLIC
                                    C

                                      EXTRN   ROM_ERR:NEAR
                                      ;----------------------------------------
                                      ;       ROS CHECKSUM SUBROUTINE          :
                                      ;----------------------------------------

                                              ASSUME  CS:CODE, DS:ABSO
 0000                                 POST3:
 0000                                 ROS_CHECKSUM    PROC    NEAR           ; NEXT_ROS_MODULE
 0000  2B C9                                  SUB     CX,CX                  ; NUMBER OF BYTES TO ADD IS 64K
 0002                                 ROS_CHECKSUM_CNT:                      ; ENTRY FOR OPTIONAL ROS TEST
 0002  32 C0                                  XOR     AL,AL
 0004                                 C26:
 0004  02 07                                  ADD     AL,DS:[BX]
 0006  43                                     INC     BX                     ; POINT TO NEXT BYTE
 0007  E2 FB                                  LOOP    C26                    ; ADD ALL BYTES IN ROS MODULE
 0009  0A C0                                  OR      AL,AL                  ; SUM = 0?
 000B  C3                                     RET
 000C                                 ROS_CHECKSUM    ENDP
                                      ;--------------------------------------------------------------------
                                      ;       BLINK LED PROCEDURE FOR MFG RUN-IN TESTS
                                      ;        IF LED IS ON, TURN IT OFF. IF OFF, TURN ON.
                                      ;--------------------------------------------------------------------
                                              ASSUME  DS:DATA
 000C                                 BLINK_INT       PROC    NEAR
 000C  FB                                     STI
 000D  50                                     PUSH    AX                             ; SAVE AX REG CONTENTS
 000E  E4 80                                  IN      AL,MFG_PORT                    ; READ CURRENT VAL OF MFG_PORT
 0010  8A E0                                  MOV     AH,AL                          ;
 0012  F6 D0                                  NOT     AL                             ; FLIP ALL BITS
 0014  24 40                                  AND     AL,01000000B                   ; ISOLATE CONTROL BIT
 0016  80 E4 BF                               AND     AH,10111111B                   ; MASK OUT OF ORIGINAL VAL
 0019  0A C4                                  OR      AL,AH                          ; OR NEW CONTROL BIT IN
 001B  E6 80                                  OUT     MFG_PORT,AL
 001D  B0 20                                  MOV     AL,EOI
 001F  E6 20                                  OUT     INTA00,AL
 0021  58                                     POP     AX                             ; RESTORE AX REG
 0022  CF                                     IRET
 0023                                 BLINK_INT       ENDP
                                      ;----------------------------------------------------
                                      ; THIS ROUTINE CHECKSUMS OPTIONAL ROM MODULES AND
                                      ; IF CHECKSUM IS OK, CALLS INIT/TEST CODE IN MODULE
                                      ;----------------------------------------------------
 0023                                 ROM_CHECK       PROC    NEAR
 0023  B8 ---- R                              MOV     AX,DATA                ; POINT ES TO DATA AREA
 0026  8E C0                                  MOV     ES,AX                  ;
 0028  2A E4                                  SUB     AH,AH                  ; ZERO OUT AH
 002A  8A 47 02                               MOV     AL,[BX+2]              ; GET LENGTH INDICATOR
 002D  B1 09                                  MOV     CL,09H                 ; MULTIPLY BY 512
 002F  D3 E0                                  SHL     AX,CL                  ;
 0031  8B C8                                  MOV     CX,AX                  ; SET COUNT
 0033  51                                     PUSH    CX                     ; SAVE COUNT
 0034  B9 0004                                MOV     CX,4                   ; ADJUST
 0037  D3 E8                                  SHR     AX,CL                  ;
 0039  03 D0                                  ADD     DX,AX                  ; SET POINTER TO NEXT MODULE
 003B  59                                     POP     CX                     ; RETRIVE COUNT
 003C  E8 0002 R                              CALL    ROS_CHECKSUM_CNT       ; DO CHECKSUM
 003F  74 06                                  JZ      ROM_CHECK_1            ;
 0041  E8 0000 E                              CALL    ROM_ERR                ; POST CHECKSUM ERROR
 0044  EB 14 90                               JMP     ROM_CHECK_END          ; AND EXIT
 0047                                 ROM_CHECK_1:
 0047  52                                     PUSH    DX                     ; SAVE POINTER
 0048  26: C7 06 0067 R 0003                  MOV     ES:IO_ROM_INIT,0003H           ; LOAD OFFSET
 004F  26: 8C 1E 0069 R                       MOV     ES:IO_ROM_SEG,DS               ; LOAD SEGMENT
 0054  26: FF 1E 0067 R                       CALL    DWORD PTR ES:IO_ROM_INIT       ; CALL INIT./TEST ROUTINE
 0059  5A                                     POP     DX                             ;
 005A                                 ROM_CHECK_END:
 005A  C3                                     RET                                    ; RETURN TO CALLER
 005B                                 ROM_CHECK       ENDP
                                      ;--------------------------------------------------------------------
                                      ;       CONVERT AND PRINT ASCII CODE
                                      ;
                                      ;       AL MUST CONTAIN NUMBER TO BE CONVERTED.
                                      ;       AX AND BX DESTROYED.
                                      ;--------------------------------------------------------------------
 005B                                 XPC_BYTE        PROC    NEAR
 005B  50                                     PUSH    AX                     ; SAVE FOR LOW NIBBLE DISPLAY
 005C  B1 04                                  MOV     CL,4                   ; SHIFT COUNT
 005E  D2 E8                                  SHR     AL,CL                  ; NIBBLE SWAP
 0060  E8 0066 R                              CALL    XLAT_PR                ; DO THE HIGH NIBBLE DISPLAY
 0063  58                                     POP     AX                     ; RECOVER THE NIBBLE
 0064  24 0F                                  AND     AL,0FH                 ; ISOLATE TO LOW NIBBLE
                                                                             ; FALL INTO LOW NIBBLE CONVERSION
 0066                                 XLAT_PR PROC    NEAR                   ;       CONVERT 00-0F TO ASCII CHARACTER
 0066  04 90                                  ADD     AL,090H                ; ADD FIRST CONVERSION FACTOR
 0068  27                                     DAA                            ; ADJUST FOR NUMERIC AND ALPHA RANGE
 0069  14 40                                  ADC     AL,040H                ; ADD CONVERSION AND ADJUST LOW NIBBLE
 006B  27                                     DAA                            ; ADJUST HIGH NIBBLE TO ASCHI RANGE
 006C                                 PRT_HEX PROC    NEAR
 006C  B4 0E                                  MOV     AH,14                  ; DISPLAY CHARACTER IN AL
 006E  B7 00                                  MOV     BH,0
 0070  CD 10                                  INT     10H                    ; CALL VIDEO_IO
 0072  C3                                     RET
 0073                                 PRT_HEX ENDP
 0073                                 XLAT_PR ENDP
 0073                                 XPC_BYTE        ENDP
                                      ;----------------------------------------------------
                                      ; PUT CHARACTER TO THE CRT FOR TEST.11 IN
                                      ;  PROTECTED MODE
                                      ;
                                      ; AL=ASCII CHARTER DI=CRT BUFFER POSITION
                                      ;----------------------------------------------------
 0073                                 PROT_PRT_HEX    PROC    NEAR
 0073  1E                                     PUSH    DS                     ; SAVE CURRENT SEGMENT REGS
 0074  53                                     PUSH    BX                     ;

                                      ;------- B/W VIDEO CARD

 0075  BB 0020                                MOV     BX,C_BWCRT_PTR         ;
 0078  8E DB                                  MOV     DS,BX                  ; SET DS TO BW CRT BUFFER
 007A  E8 0098 R                              CALL    PROT_PRT               ; GO PRINT CHARACTER
```

SECTION 5

```
                                ;-------- COMPATIBLE COLOR

007D  BB 0028                         MOV    BX,C_CCRT_PTR         ; SET DS TO COMPATIBLE COLOR RAM
0080  8E DB                           MOV    DS,BX                 ;
0082  E8 0098 R                       CALL   PROT_PRT              ;

                                ;-------- ENHANCED COLOR

0085  BB 0030                         MOV    BX,E_CCRT_PTR         ; ENHANCED COLOR
0088  8E DB                           MOV    DS,BX                 ;
008A  E8 0098 R                       CALL   PROT_PRT              ;
008D  BB 0038                         MOV    BX,E_CCRT_PTR2        ; ENHANCED COLOR PTR HI 64K
0090  8E DB                           MOV    DS,BX                 ;
0092  E8 0098 R                       CALL   PROT_PRT              ;
0095  5B                              POP    BX                    ;
0096  1F                              POP    DS                    ;
0097  C3                              RET
0098                            PROT_PRT:
0098  57                              PUSH   DI                    ; SAVE DISPLACEMENT
0099  D1 C7                           ROL    DI,1                  ; MULT *2
009B  88 05                           MOV    DS:[DI],AL            ; WRITE TO CRT BUFFER
009D  5F                              POP    DI                    ; RESTORE DISPLACEMENT
009E  C3                              RET                          ;
009F                            PROT_PRT_HEX       ENDP

009F                            PROC_SHUTDOWN      PROC
009F  B0 FE                           MOV    AL,SHUT_CMD           ; SHUTDOWN COMMAND
00A1  E6 64                           OUT    STATUS_PORT,AL        ;
00A3  F4                        PROC_S: HLT                        ;
00A4  EB FD                           JMP    PROC_S                ; INSURE HALT
00A6                            PROC_SHUTDOWN      ENDP
00A6                            CODE   ENDS
                                      END
```

```
                                   TITLE 10/05/83 TEST4    POST UTILITIES
                                   .LIST
                                   PUBLIC  POST4
                                   PUBLIC  E_MSG
                                   PUBLIC  KBD_RESET
                                   PUBLIC  BEEP
                                   PUBLIC  ERR_BEEP
                                   PUBLIC  E_MSG
                                   PUBLIC  DDS
                                   PUBLIC  P_MSG
                                   PUBLIC  PRT_SEG
                                   PUBLIC  DUMMY_RETURN_1
                                   PUBLIC  D11
                                   PUBLIC  INT_287
                                   PUBLIC  RE_DIRECT

                                 C INCLUDE SEGMENT.SRC
0000                             C CODE SEGMENT BYTE PUBLIC
                                 C

                                   EXTRN   PRT_HEX:NEAR
                                   EXTRN   XPC_BYTE:NEAR
                                   EXTRN   XMIT_8042:NEAR
                                   EXTRN   OBF_42:NEAR
                                   ASSUME  CS:CODE,DS:ABS0
0000                               POST4:
                                   ;-----------------------------------------------------------------
                                   ;       THIS SUBROUTINE WILL PRINT A MESSAGE ON THE DISPLAY
                                   ;
                                   ; ENTRY REQUIREMENTS:
                                   ;       SI = OFFSET(ADDRESS) OF MESSAGE BUFFER
                                   ;       CX = MESSAGE BYTE COUNT
                                   ;       MAXIMUM MESSAGE LENGTH IS 36 CHARACTERS
                                   ;-----------------------------------------------------------------
0000                               E_MSG   PROC    NEAR
0000  8B EE                                MOV     BP,SI                  ; SET BP NON-ZERO TO FLAG ERR
0002  E8 0019 R                            CALL    P_MSG                  ; PRINT MESSAGE
0005  1E                                   PUSH    DS
                                   ASSUME  DS:DATA
0006  E8 00AA R                            CALL    DDS
0009  A0 0010 R                            MOV     AL,BYTE PTR EQUIP_FLAG ; LOOP/HALT ON ERROR
000C  24 01                                AND     AL,01H                 ; SWITCH ON?
000E  75 07                                JNZ     NOT_ON                 ; NO - RETURN
0010                               MFG_HALT:
0010  FA                                   CLI                            ; YES -  HALT SYSTEM
0011  A0 0015 R                            MOV     AL,MFG_ERR_FLAG        ; RECOVER ERROR INDICATOR
0014  E6 80                                OUT     MFG_PORT,AL            ; SET INTO MFG PORT
0016  F4                                   HLT                            ; HALT SYS
0017                               NOT_ON:
0017  1F                                   POP     DS                     ; WRITE_MSG:
0018  C3                                   RET
0019                               E_MSG   ENDP
0019                               P_MSG   PROC    NEAR
0019  2E: 8A 04                    G12A:   MOV     AL,CS:[SI]             ; PUT CHAR IN AL
001C  46                                   INC     SI                     ; POINT TO NEXT CHAR
001D  50                                   PUSH    AX                     ; SAVE PRINT CHAR
001E  E8 0000 E                            CALL    PRT_HEX                ; CALL VIDEO_IO
0021  58                                   POP     AX                     ; RECOVER PRINT CHAR
0022  3C 0A                                CMP     AL,10                  ; WAS IT LINE FEED?
0024  75 F3                                JNE     G12A                   ; NO,KEEP PRINTING STRING
0026  C3                                   RET
0027                               P_MSG   ENDP
                                   ;-----------------------------------------
                                   ;  INITIAL RELIABILITY TEST -- SUBROUTINES
                                   ;-----------------------------------------
                                           ASSUME   CS:CODE,DS:DATA
                                   ;-------------------------------------------------------------------
                                   ;       SUBROUTINES FOR POWER ON DIAGNOSTICS
                                   ;-------------------------------------------------------------------
                                   ;       THIS PROCEDURE WILL ISSUE ONE LONG TONE (3 SECS) AND ONE OR
                                   ;       MORE SHORT TONES (1 SEC) TO INDICATE A FAILURE ON THE PLANAR
                                   ;       BOARD, A BAD RAM MODULE, OR A PROBLEM WITH THE CRT.
                                   ; ENTRY PARAMETERS:
                                   ;       DH = NUMBER OF LONG TONES TO BEEP
                                   ;       DL = NUMBER OF SHORT TONES TO BEEP.
                                   ;-------------------------------------------------------------------
0027                               ERR_BEEP  PROC  NEAR
0027  9C                                   PUSHF                          ; SAVE FLAGS
0028  FA                                   CLI                            ; DISABLE SYSTEM INTERRUPTS
0029  1E                                   PUSH    DS                     ; SAVE DS REG CONTENTS
002A  E8 00AA R                            CALL    DDS
002D  0A F6                                OR      DH,DH          ; ANY LONG ONES TO BEEP
002F  74 14                                JZ      G3             ; NO, DO THE SHORT ONES
0031                               G1:                            ; LONG_BEEP:
0031  B3 06                                MOV     BL,6           ; COUNTER FOR BEEPS
0033  E8 0057 R                            CALL    BEEP           ; DO THE BEEP
0036  E2 FE                        G2:     LOOP    G2             ; DELAY BETWEEN BEEPS
0038  FE CE                                DEC     DH             ; ANY MORE TO DO
003A  75 F5                                JNZ     G1             ; DO IT
003C  80 3E 0012 R 01                      CMP     MFG_TST,1      ; MFG TEST MODE?
0041  75 02                                JNE     G3             ; YES - CONTINUE BEEPING SPEAKER
0043  EB CB                                JMP     MFG_HALT       ; STOP BLINKING LED
0045                               G3:                            ; SHORT_BEEP:
0045  B3 01                                MOV     BL,1           ; COUNTER FOR A SHORT BEEP
0047  E8 0057 R                            CALL    BEEP           ; DO THE SOUND
004A  E2 FE                        G4:     LOOP    G4             ; DELAY BETWEEN BEEPS
004C  FE CA                                DEC     DL             ; DONE WITH SHORTS
004E  75 F5                                JNZ     G3             ; DO SOME MORE
0050  E2 FE                        G5:     LOOP    G5             ; LONG DELAY BEFORE RETURN
0052  E2 FE                        G6:     LOOP    G6
0054  1F                                   POP     DS             ; RESTORE ORIG CONTENTS OF DS
0055  9D                                   POPF                   ; RESTORE FLAGS TO ORIG SETTINGS
0056  C3                                   RET                    ; RETURN TO CALLER
0057                               ERR_BEEP        ENDP

                                   ;       ROUTINE TO SOUND BEEPER

0057                               BEEP    PROC    NEAR
0057  B0 B6                                MOV     AL,10110110B           ; SEL TIM 2,LSB,MSB,BINARY
0059  E6 43                                OUT     TIMER+3,AL             ; WRITE THE TIMER MODE REG
005B  EB 00                                JMP     SHORT $+2              ; IO DELAY
005D  B8 0533                              MOV     AX,533H                ; DIVISOR FOR 896 HZ
0060  E6 42                                OUT     TIMER+2,AL             ; WRITE TIMER 2 CNT - LSB
0062  EB 00                                JMP     SHORT $+2              ; IO DELAY
0064  8A C4                                MOV     AL,AH
0066  E6 42                                OUT     TIMER+2,AL             ; WRITE TIMER 2 CNT - MSB
0068  E4 61                                IN      AL,PORT_B              ; GET CURRENT SETTING OF PORT
006A  8A E0                                MOV     AH,AL                  ; SAVE THAT SETTING
006C  EB 00                                JMP     SHORT $+2              ; IO DELAY
006E  0C 03                                OR      AL,03                  ; TURN SPEAKER ON
0070  E6 61                                OUT     PORT_B,AL
0072  2B C9                                SUB     CX,CX                  ; SET CNT TO WAIT 500 MS
0074  E2 FE                        G7:     LOOP    G7                     ; DELAY BEFORE TURNING OFF
0076  FE CB                                DEC     BL                     ; DELAY CNT EXPIRED?
0078  75 FA                                JNZ     G7                     ; NO - CONTINUE BEEPING SPK
```

```
007A  8A C4                 MOV     AL,AH                   ; RECOVER VALUE OF PORT
007C  E6 61                 OUT     PORT_B,AL
007E  C3                    RET                             ; RETURN TO CALLER
007F                 BEEP   ENDP

                     ;--------------------------------------------------------------
                     ;      THIS PROCEDURE WILL SEND A SOFTWARE RESET TO THE KEYBOARD.
                     ;      SCAN CODE `AA' SHOULD BE RETURNED TO THE CPU.
                     ;      SCAN CODE `65' IS DEFINED FOR MANUFACTURING TEST
                     ;--------------------------------------------------------------
007F                 KBD_RESET      PROC    NEAR
007F  B0 FF                 MOV     AL,0FFH                 ; SET KEYBOARD RESET COMMAND
0081  E8 0000 E             CALL    XMIT_8042               ; GO ISSUE THE COMMAND
0084  E3 23                 JCXZ    G13                     ; GO IF ERROR

0086  3C FA                 CMP     AL,KB_ACK               ;
0088  75 1F                 JNZ     G13

008A  B0 FD                 MOV     AL,0FDH                 ; ENABLE KEYBOARD INTERRUPTS
008C  E6 21                 OUT     INTA01,AL               ; WRITE 8259 IMR
008E  C6 06 006B R 00       MOV     INTR_FLAG,0             ; RESET INTERRUPT INDICATOR
0093  FB                    STI                             ; ENABLE INTERRUPTS

0094  B3 0A                 MOV     BL,10                   ; TRY FOR 400 MSEC
0096  2B C9                 SUB     CX,CX                   ; SETUP INTERRUPT TIMEOUT CNT
0098  F6 06 006B R 02 G11:  TEST    INTR_FLAG,02H           ; DID A KEYBOARD INTR OCCUR?
009D  75 06                 JNZ     G12                     ; YES - READ SCAN CODE RETURNED
009F  E2 F7                 LOOP    G11                     ; NO - LOOP TILL TIMEOUT
00A1  FE CB                 DEC     BL                      ;
00A3  75 F3                 JNZ     G11                     ; TRY AGAIN

00A5  E4 60          G12:   IN      AL,PORT_A               ; READ KEYBOARD SCAN CODE
00A7  8A D8                 MOV     BL,AL                   ; SAVE SCAN CODE JUST READ
00A9  C3             G13:   RET                             ; RETURN TO CALLER
00AA                 KBD_RESET      ENDP

00AA                 DDS    PROC    NEAR
00AA  50                    PUSH    AX
00AB  B8 ---- R             MOV     AX,DATA
00AE  8E D8                 MOV     DS,AX
00B0  58                    POP     AX
00B1  C3                    RET
00B2                 DDS    ENDP

                     ;--------------------------------------------------
                     ; TEMPORARY INTERRUPT SERVICE ROUTINE             :
                     ;      1. THIS ROUTINE IS ALSO LEFT IN PLACE AFTER THE :
                     ;      POWER ON DIAGNOSTICS TO SERVICE UNUSED     :
                     ;      INTERRUPT VECTORS. LOCATION 'INTR_FLAG' WILL :
                     ;      CONTAIN EITHER: 1. LEVEL OF HARDWARE INT. THAT :
                     ;      CAUSED CODE TO BE EXEC.                    :
                     ;      2. 'FF' FOR NON-HARDWARE INTERUPTS THAT WAS :
                     ;      EXECUTED ACCIDENTLY.                       :
                     ;--------------------------------------------------
00B2                 D11    PROC    NEAR
                            ASSUME  DS:DATA
00B2  1E                    PUSH    DS
00B3  52                    PUSH    DX
00B4  50                    PUSH    AX                      ; SAVE REG AX CONTENTS
00B5  53                    PUSH    BX                      ;
00B6  E8 00AA R             CALL    DDS                     ; SET DATA SEGMENT
00B9  B0 0B                 MOV     AL,0BH                  ; READ IN-SERVICE REG
00BB  E6 20                 OUT     INTA00,AL               ; (FIND OUT WHAT LEVEL BEING
00BD  EB 00                 JMP     SHORT $+2               ; IO DELAY
00BF  90                    NOP                             ; SERVICED)
00C0  E4 20                 IN      AL,INTA00               ; GET LEVEL
00C2  8A E0                 MOV     AH,AL                   ; SAVE IT
00C4  0A C4                 OR      AL,AH                   ; 00? (NO HARDWARE ISR ACTIVE)
00C6  75 04                 JNZ     HW_INT
00C8  B4 FF                 MOV     AH,0FFH
00CA  EB 2A                 JMP     SHORT SET_INTR_FLAG     ; SET FLAG TO FF IF NON-HDWARE
00CC                 HW_INT:
00CC  B0 0B                 MOV     AL,0BH                  ;
00CE  E6 A0                 OUT     INTB00,AL               ; READ IN-SERVICE REG INT CHIP 2
00D0  EB 00                 JMP     SHORT $+2               ; IO DELAY
00D2  E4 A0                 IN      AL,INTB00               ; CHECK THE SECOND INT CHIP
00D4  8A F8                 MOV     BH,AL                   ; SAVE IT
00D6  0A FF                 OR      BH,BH                   ;
00D8  74 0E                 JZ      NOT_SEC                 ; CONTINUE IF NOT
00DA  E4 A1                 IN      AL,INTB01               ; GET SECOND INT MASK
00DC  0A C7                 OR      AL,BH                   ; MASK OFF LVL BEING SERVICED
00DE  EB 00                 JMP     SHORT $+2               ; IO DELAY
00E0  E6 A1                 OUT     INTB01,AL               ;
00E2  B0 20                 MOV     AL,EOI                  ; SEND EOI TO SECOND CHIP
00E4  EB 00                 JMP     SHORT $+2               ; IO DELAY
00E6  E6 A0                 OUT     INTB00,AL               ;
00E8  E4 21          NOT_SEC:IN     AL,INTA01               ; GET MASK VALUE
00EA  EB 00                 JMP     SHORT $+2               ; IO DELAY
00EC  0A C4                 OR      AL,AH                   ; MASK OFF LVL BEING SERVICED
00EE  E6 21                 OUT     INTA01,AL
00F0  EB 00                 JMP     SHORT $+2               ; IO DELAY
00F2  B0 20                 MOV     AL,EOI
00F4  E6 20                 OUT     INTA00,AL
00F6                 SET_INTR_FLAG:
00F6  88 26 006B R          MOV     INTR_FLAG,AH            ; SET FLAG
00FA  5B                    POP     BX
00FB  58                    POP     AX                      ; RESTORE REG AX CONTENTS
00FC  5A                    POP     DX
00FD  1F                    POP     DS
00FE                 DUMMY_RETURN_1:                          ; NEED IRET FOR VECTOR TABLE
00FE  CF                    IRET
00FF                 D11    ENDP

                     ;--HARDWARE INT 13 (LEVEL 75H) --------------------------
                     ; SERVICE X287 INTERRUPTS                          :
                     ;      THIS ROUTINE FIELDS X287 INTERRUPTS AND CONTROL :
                     ;      IS PASSED TO THE NMI INTERRUPT HANDLER FOR  :
                     ;      COMPATABILITY.                              :
                     ;                                                  :
                     ;--------------------------------------------------
00FF                 INT_287 PROC   NEAR
00FF  50                    PUSH    AX                      ; SAVE AX
0100  32 C0                 XOR     AL,AL                   ;
0102  E6 F0                 OUT     X287,AL                 ; REMOVE THE INT REQUEST

0104  B0 20                 MOV     AL,EOI                  ; ENABLE THE INTERRUPT
0106  E6 A0                 OUT     INTB00,AL               ;  THE SLAVE
0108  E6 20                 OUT     INTA00,AL               ;  THE MASTER

010A  58                    POP     AX                      ; RESTORE AX
010B  CD 02                 INT     2                       ; GIVE CONTROL TO NMI
```

```
010D  CF                        IRET                        ; RETURN
010E                    INT_287 ENDP

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

010E                    RE_DIRECT PROC  NEAR
010E  50                        PUSH    AX                  ; SAVE AX
010F  B0 20                     MOV     AL,EOI
0111  E6 A0                     OUT     INTB00,AL           ; EOI TO SLAVE INT CONTROLLER
0113  58                        POP     AX                  ; RESTORE AX
0114  CD 0A                     INT     0AH                 ; GIVE CONTROL TO HARDWARE LEVEL 2

0116  CF                        IRET                        ; RETURN
0117                    RE_DIRECT ENDP
                        ;-----------------------------------------------------------
                        ;       PRINT A SEGMET VALUE TO LOOK LIKE A 21 BIT ADDRESS    :
                        ;       DX MUST CONTAIN SEGMENT VALUE TO BE PRINTED           :
                        ;-----------------------------------------------------------
0117                    PRT_SEG PROC    NEAR
0117  8A C6                     MOV     AL,DH               ;GET MSB
0119  E8 0000 E                 CALL    XPC_BYTE            ;
011C  8A C2                     MOV     AL,DL               ;LSB
011E  E8 0000 E                 CALL    XPC_BYTE            ;
0121  B0 30                     MOV     AL,'0'              ; PRINT A '0 '
0123  E8 0000 E                 CALL    PRT_HEX             ;
0126  B0 20                     MOV     AL,' '              ;SPACE
0128  E8 0000 E                 CALL    PRT_HEX             ;
012B  C3                        RET                         ;
012C                    PRT_SEG ENDP
012C                    CODE    ENDS
                                END
```

SECTION 5

```
                                TITLE 12/16/83 TEST5  EXCEPTION INTERRUPT HANDLER
                                .LIST
                                PUBLIC  POST5
                                PUBLIC  EXC_00
                                PUBLIC  EXC_01
                                PUBLIC  EXC_02
                                PUBLIC  EXC_03
                                PUBLIC  EXC_04
                                PUBLIC  EXC_05
                                PUBLIC  EXC_06
                                PUBLIC  EXC_07
                                PUBLIC  EXC_08
                                PUBLIC  EXC_09
                                PUBLIC  EXC_10
                                PUBLIC  EXC_11
                                PUBLIC  EXC_12
                                PUBLIC  EXC_13
                                PUBLIC  EXC_14
                                PUBLIC  EXC_15
                                PUBLIC  EXC_16
                                PUBLIC  EXC_17
                                PUBLIC  EXC_18
                                PUBLIC  EXC_19
                                PUBLIC  EXC_20
                                PUBLIC  EXC_21
                                PUBLIC  EXC_22
                                PUBLIC  EXC_23
                                PUBLIC  EXC_24
                                PUBLIC  EXC_25
                                PUBLIC  EXC_26
                                PUBLIC  EXC_27
                                PUBLIC  EXC_28
                                PUBLIC  EXC_29
                                PUBLIC  EXC_30
                                PUBLIC  EXC_31

                                PUBLIC  SYS_32
                                PUBLIC  SYS_33
                                PUBLIC  SYS_34
                                PUBLIC  SYS_35
                                PUBLIC  SYS_36
                                PUBLIC  SYS_37
                                PUBLIC  SYS_38

                              C  INCLUDE SEGMENT.SRC
0000                          C  CODE SEGMENT BYTE PUBLIC
                              C

                                ;----------------------------------------
                                ;       EXCEPTION INTERRUPT ROUTINE     :
                                ;----------------------------------------

                                        ASSUME  CS:CODE, DS:ABS0

0000                            POST5:
0000                            EXC_00:
0000  B0 90                             MOV     AL,90H              ;<><><>SET CHECKPOINT<><><><>
0002  E9 00D7 R                         JMP     TEST_EXC            ; GO TEST IF EXCEPTION WAS EXPECTED
0005                            EXC_01:
0005  B0 91                             MOV     AL,91H              ;<><><>SET CHECKPOINT<><><><>
0007  E9 00D7 R                         JMP     TEST_EXC            ; GO TEST IF EXCEPTION WAS EXPECTED
000A                            EXC_02:
000A  B0 92                             MOV     AL,92H              ;<><><>SET CHECKPOINT<><><><>
000C  E9 00D7 R                         JMP     TEST_EXC            ; GO TEST IF EXCEPTION WAS EXPECTED
000F                            EXC_03:
000F  B0 93                             MOV     AL,93H              ;<><><>SET CHECKPOINT<><><><>
0011  E9 00D7 R                         JMP     TEST_EXC            ; GO TEST IF EXCEPTION WAS EXPECTED
0014                            EXC_04:
0014  B0 94                             MOV     AL,94H              ;<><><>SET CHECKPOINT<><><><>
0016  E9 00D7 R                         JMP     TEST_EXC            ; GO TEST IF EXCEPTION WAS EXPECTED
0019                            EXC_05:

0019  06                                PUSH    ES

001A  B8 0048                           MOV     AX,ES_TEMP          ; LOAD ES REGISTER
001D  8E C0                             MOV     ES,AX               ;

                                ;------- FIX BOUND PARAMETERS

001F  2B FF                             SUB     DI,DI               ; POINT BEGINING OF THE BLOCK
0021  26: C7 05 0000                    MOV     WORD PTR ES:[DI],0  ; SET FIRST WORD TO ZERO

0026  26: C7 45 02 7FFF                 MOV     WORD PTR ES:[DI+2],07FFFH ; SET SECOND TO 07FFFH
002C  07                                POP     ES

002D  B0 95                             MOV     AL,95H              ;<><><>SET CHECKPOINT<><><><>
002F  E9 00D7 R                         JMP     TEST_EXC            ; GO TEST IF EXCEPTION WAS EXPECTED
0032                            EXC_06:
0032  B0 96                             MOV     AL,96H              ;<><><>SET CHECKPOINT<><><><>
0034  E9 00D7 R                         JMP     TEST_EXC            ; GO TEST IF EXCEPTION WAS EXPECTED
0037                            EXC_07:
0037  B0 97                             MOV     AL,97H              ;<><><>SET CHECKPOINT<><><><>
0039  E9 00D7 R                         JMP     TEST_EXC            ; GO TEST IF EXCEPTION WAS EXPECTED
003C                            EXC_08:
003C  B0 98                             MOV     AL,98H              ;<><><>SET CHECKPOINT<><><><>
003E  E9 00D7 R                         JMP     TEST_EXC            ; GO TEST IF EXCEPTION WAS EXPECTED
0041                            EXC_09:
0041  B0 99                             MOV     AL,99H              ;<><><>SET CHECKPOINT<><><><>
0043  E9 00D7 R                         JMP     TEST_EXC            ; GO TEST IF EXCEPTION WAS EXPECTED
0046                            EXC_10:
0046  B0 9A                             MOV     AL,9AH              ;<><><>SET CHECKPOINT<><><><>
0048  E9 00D7 R                         JMP     TEST_EXC            ; GO TEST IF EXCEPTION WAS EXPECTED
004B                            EXC_11:
004B  B0 9B                             MOV     AL,9BH              ;<><><>SET CHECKPOINT<><><><>
004D  E9 00D7 R                         JMP     TEST_EXC            ; GO TEST IF EXCEPTION WAS EXPECTED
0050                            EXC_12:
0050  B0 9C                             MOV     AL,9CH              ;<><><>SET CHECKPOINT<><><><>
0052  E9 00D7 R                         JMP     TEST_EXC            ; GO TEST IF EXCEPTION WAS EXPECTED
0055                            EXC_13:
0055  B0 9D                             MOV     AL,9DH              ;<><><>SET CHECKPOINT<><><><>
0057  EB 7E 90                          JMP     TEST_EXC            ; GO TEST IF EXCEPTION WAS EXPECTED
005A                            EXC_14:
005A  B0 9E                             MOV     AL,9EH              ;<><><>SET CHECKPOINT<><><><>
005C  EB 79 90                          JMP     TEST_EXC            ; GO TEST IF EXCEPTION WAS EXPECTED
005F                            EXC_15:
005F  B0 9F                             MOV     AL,9FH              ;<><><>SET CHECKPOINT<><><><>
0061  EB 74 90                          JMP     TEST_EXC            ; GO TEST IF EXCEPTION WAS EXPECTED
0064                            EXC_16:
0064  B0 A0                             MOV     AL,0A0H             ;<><><>SET CHECKPOINT<><><><>
0066  EB 6F 90                          JMP     TEST_EXC            ; GO TEST IF EXCEPTION WAS EXPECTED
0069                            EXC_17:
0069  B0 A1                             MOV     AL,0A1H             ;<><><>SET CHECKPOINT<><><><>
006B  EB 6A 90                          JMP     TEST_EXC            ; GO TEST IF EXCEPTION WAS EXPECTED
006E                            EXC_18:
006E  B0 A2                             MOV     AL,0A2H             ;<><><>SET CHECKPOINT<><><><>
```

SECTION 5

```
0070  EB 65 90                      JMP    TEST_EXC             ; GO TEST IF EXCEPTION WAS EXPECTED
0073                          EXC_19:
0073  B0 A2                         MOV    AL,0A2H              ;<><><>SET CHECKPOINT<><><><>
0075  EB 60 90                      JMP    TEST_EXC             ; GO TEST IF EXCEPTION WAS EXPECTED
0078                          EXC_20:
0078  B0 A3                         MOV    AL,0A3H              ;<><><>SET CHECKPOINT<><><><>
007A  EB 5B 90                      JMP    TEST_EXC             ; GO TEST IF EXCEPTION WAS EXPECTED
007D                          EXC_21:
007D  B0 A4                         MOV    AL,0A4H              ;<><><>SET CHECKPOINT<><><><>
007F  EB 56 90                      JMP    TEST_EXC             ; GO TEST IF EXCEPTION WAS EXPECTED
0082                          EXC_22:
0082  B0 A5                         MOV    AL,0A5H              ;<><><>SET CHECKPOINT<><><><>
0084  EB 51 90                      JMP    TEST_EXC             ; GO TEST IF EXCEPTION WAS EXPECTED
0087                          EXC_23:
0087  B0 A6                         MOV    AL,0A6H              ;<><><>SET CHECKPOINT<><><><>
0089  EB 4C 90                      JMP    TEST_EXC             ; GO TEST IF EXCEPTION WAS EXPECTED
008C                          EXC_24:
008C  B0 A7                         MOV    AL,0A7H              ;<><><>SET CHECKPOINT<><><><>
008E  EB 47 90                      JMP    TEST_EXC             ; GO TEST IF EXCEPTION WAS EXPECTED
0091                          EXC_25:
0091  B0 A8                         MOV    AL,0A8H              ;<><><>SET CHECKPOINT<><><><>
0093  EB 42 90                      JMP    TEST_EXC             ; GO TEST IF EXCEPTION WAS EXPECTED
0096                          EXC_26:
0096  B0 A9                         MOV    AL,0A9H              ;<><><>SET CHECKPOINT<><><><>
0098  EB 3D 90                      JMP    TEST_EXC             ; GO TEST IF EXCEPTION WAS EXPECTED
009B                          EXC_27:
009B  B0 AA                         MOV    AL,0AAH              ;<><><>SET CHECKPOINT<><><><>
009D  EB 38 90                      JMP    TEST_EXC             ; GO TEST IF EXCEPTION WAS EXPECTED
00A0                          EXC_28:
00A0  B0 AB                         MOV    AL,0ABH              ;<><><>SET CHECKPOINT<><><><>
00A2  EB 33 90                      JMP    TEST_EXC             ; GO TEST IF EXCEPTION WAS EXPECTED
00A5                          EXC_29:
00A5  B0 AC                         MOV    AL,0ACH              ;<><><>SET CHECKPOINT<><><><>
00A7  EB 2E 90                      JMP    TEST_EXC             ; GO TEST IF EXCEPTION WAS EXPECTED
00AA                          EXC_30:
00AA  B0 AD                         MOV    AL,0ADH              ;<><><>SET CHECKPOINT<><><><>
00AC  EB 29 90                      JMP    TEST_EXC             ; GO TEST IF EXCEPTION WAS EXPECTED
00AF                          EXC_31:
00AF  B0 AE                         MOV    AL,0AEH              ;<><><>SET CHECKPOINT<><><><>
00B1  EB 24 90                      JMP    TEST_EXC             ; GO TEST IF EXCEPTION WAS EXPECTED

00B4                          SYS_32:
00B4  B0 AF                         MOV    AL,0AFH              ;<><><>SET CHECKPOINT<><><><>
00B6  EB 1F 90                      JMP    TEST_EXC             ; GO TEST IF INTERRUPT WAS EXPECTED
00B9                          SYS_33:
00B9  B0 B0                         MOV    AL,0B0H              ;<><><>SET CHECKPOINT<><><><>
00BB  EB 1A 90                      JMP    TEST_EXC             ; GO TEST IF INTERRUPT WAS EXPECTED
00BE                          SYS_34:
00BE  B0 B1                         MOV    AL,0B1H              ;<><><>SET CHECKPOINT<><><><>
00C0  EB 15 90                      JMP    TEST_EXC             ; GO TEST IF INTERRUPT WAS EXPECTED
00C3                          SYS_35:
00C3  B0 B2                         MOV    AL,0B2H              ;<><><>SET CHECKPOINT<><><><>
00C5  EB 10 90                      JMP    TEST_EXC             ; GO TEST IF INTERRUPT WAS EXPECTED
00C8                          SYS_36:
00C8  B0 B3                         MOV    AL,0B3H              ;<><><>SET CHECKPOINT<><><><>
00CA  EB 0B 90                      JMP    TEST_EXC             ; GO TEST IF INTERRUPT WAS EXPECTED
00CD                          SYS_37:
00CD  B0 B4                         MOV    AL,0B4H              ;<><><>SET CHECKPOINT<><><><>
00CF  EB 06 90                      JMP    TEST_EXC             ; GO TEST IF INTERRUPT WAS EXPECTED
00D2                          SYS_38:
00D2  B0 B5                         MOV    AL,0B5H              ;<><><>SET CHECKPOINT<><><><>
00D4  EB 01 90                      JMP    TEST_EXC             ; GO TEST IF INTERRUPT WAS EXPECTED
00D7                          TEST_EXC:
00D7  E6 80                         OUT    MFG_PORT,AL          ; OUTPUT THE CHECKPOINT
00D9  3C AE                         CMP    AL,0AEH              ; CHECK FOR EXCEPTION
00DB  77 22                         JA     TEST_EXC0            ; GO IF A SYSTEM INT

00DD  1E                            PUSH   DS                   ; SAVE THE CURRENT DATA SEGMENT
00DE  50                            PUSH   AX                   ;
00DF  B8 0008                       MOV    AX,GDT_PTR           ;
00E2  8E D8                         MOV    DS,AX                ;
00E4  C7 06 0048 FFFF               MOV    DS:ES_TEMP.SEG_LIMIT,MAX_SEG_LEN
00EA  C6 06 004D 93                 MOV    BYTE PTR DS:(ES_TEMP.DATA_ACC_RIGHTS),CPLO_DATA_ACCESS
00EF  B8 0048                       MOV    AX,ES_TEMP
00F2  8E C0                         MOV    ES,AX
00F4  58                            POP    AX                   ; RESTORE REGS
00F5  1F                            POP    DS
00F6  5A                            POP    DX                   ; CHECK IF CODE SEG SECOND ON STACK
00F7  59                            POP    CX
00F8  51                            PUSH   CX                   ;
00F9  83 F9 40                      CMP    CX,SYS_ROM_CS        ;
00FC  75 01                         JNZ    TEST_EXC0            ; CONTINUE IF ERROR CODE
00FE  52                            PUSH   DX                   ; PUT SEGMENT BACK ON STACK

00FF                          TEST_EXC0:
00FF  86 E0                         XCHG   AH,AL                ; SAVE THE CHECKPOINT
0101  E4 8B                         IN     AL,DMA_PAGE+0AH      ;
0103  3A C4                         CMP    AL,AH                ; WAS THE EXCEPTION EXPECTED?
0105  74 0E                         JZ     TEST_EXC3            ; GO IF YES
0107                          TEST_EXC1:
0107  E4 80                         IN     AL,MFG_PORT          ; CHECK THE CURRENT CHKPT
0109  3C 3B                         CMP    AL,03BH              ;  HALT IF CHKPT BELOW 3BH
010B  72 01                         JB     TEST_EXC2            ;
010D  CF                            IRET                        ;
010E                          TEST_EXC2:
010E  86 E0                         XCHG   AH,AL                ; OUTPUT THE CURRENT CHECKPOINT
0110  E6 80                         OUT    MFG_PORT,AL          ; <><><> CKPT 90 THRU B5 <><><>

0112  F4                            HLT
0113  EB F9                         JMP    TEST_EXC2            ; INSURE SYSTEM HALT
0115                          TEST_EXC3:
0115  2A C0                         SUB    AL,AL                ; CLEAR DMA PAGE
0117  E6 8B                         OUT    DMA_PAGE+0AH,AL      ;
0119  B8 0100                       MOV    AX,0100H             ; USED FOR BOUND INSTR EXPECTED INT5
011C  CF                            IRET                        ; RETURN
011D                          CODE  ENDS
                                    END
```

```
                              TITLE 01/03/84 TEST6 POWER ON SELF TEST
                              .LIST
                              PUBLIC  STGTST_CNT
                              PUBLIC  ROM_ERR
                              PUBLIC  BOOT_STRAP_1
                              PUBLIC  XMIT_8042
                              PUBLIC  POST6
                              PUBLIC  H5

                            C INCLUDE SEGMENT.SRC
0000                        C CODE SEGMENT BYTE PUBLIC
                            C
                              ;
                              EXTRN   E0:NEAR
                              EXTRN   E_MSG:NEAR
                              EXTRN   KBD_RESET:NEAR
                              EXTRN   XPC_BYTE:NEAR
                              EXTRN   F1:NEAR
                              EXTRN   VECTOR_TABLE:NEAR
                              EXTRN   NMI_INT:NEAR
                              EXTRN   PRINT_SCREEN_1:NEAR
                              EXTRN   BLINK_INT:NEAR
                              EXTRN   PRT_HEX:NEAR
                              EXTRN   F3B:NEAR
                              EXTRN   PRT_SEG:NEAR
                              EXTRN   XPC_BYTE:NEAR
                              EXTRN   E1:NEAR
                              EXTRN   ROM_CHECK:NEAR
                              EXTRN   ROS_CHECKSUM:NEAR
                              EXTRN   SEEK:NEAR
                              EXTRN   F3:NEAR
                              EXTRN   ERR_BEEP:NEAR
                              EXTRN   P_MSG:NEAR
                              EXTRN   START_1:NEAR
                              EXTRN   F4:NEAR
                              EXTRN   F4E:NEAR
                              EXTRN   DDS:NEAR
                              EXTRN   F3A:NEAR
                              EXTRN   DISK_BASE:NEAR
                              EXTRN   F3D:NEAR
                              EXTRN   PROC_SHUTDOWN:NEAR
                              EXTRN   SYSINIT1:NEAR
                              EXTRN   PROT_PRT_HEX:NEAR
                              EXTRN   DISK_IO:NEAR
                              EXTRN   HD_INT:NEAR
                              EXTRN   C8042:NEAR
                              EXTRN   BOOT_INVA:NEAR
                              PAGE
                                      ASSUME  CS:CODE
                                      ASSUME  DS:DATA

0000                          POST6   PROC    NEAR

                              ;--------------------------------------------------------------
                              ; THIS SUBROUTINE PERFORMS A READ/WRITE STORAGE TEST ON A BLOCK :
                              ;       OF STORAGE.                                             :
                              ; ENTRY REQUIREMENTS:                                           :
                              ;       ES = ADDRESS OF STORAGE SEGMENT BEING TESTED            :
                              ;       DS = ADDRESS OF STORAGE SEGMENT BEING TESTED            :
                              ;       CX = WORD COUNT OF STORAGE BLOCK TO BE TESTED           :
                              ; EXIT PARAMETERS:                                              :
                              ;       ZERO FLAG = 0 IF STORAGE ERROR (DATA COMPARE OR PARITY  :
                              ;       CHECK). AL=0 DENOTES A PARITY CHECK. ELSE AL=XOR'ED     :
                              ;       BIT PATTERN OF THE EXPECTED DATA PATTERN VS THE ACTUAL  :
                              ;       DATA READ.                                              :
                              ; AX,BX,CX,DX,DI, AND SI ARE ALL DESTROYED.                     :
                              ;--------------------------------------------------------------

0000                          STGTST_CNT      PROC    NEAR
0000  8B D9                           MOV     BX,CX                   ; SAVE WORD COUNT OF BLOCK TO TEST
0002  E4 61                           IN      AL,PORT_B
0004  EB 00                           JMP     SHORT $+2               ; IO DELAY
0006  0C 0C                           OR      AL,RAM_PAR_OFF          ; TOGGLE PARITY CHECK LATCHES
0008  E6 61                           OUT     PORT_B,AL
000A  EB 00                           JMP     SHORT $+2               ; IO DELAY
000C  24 F3                           AND     AL,RAM_PAR_ON           ;
000E  E6 61                           OUT     PORT_B,AL

                              ;-------- ROLL A BIT THROUGH THE FIRST WORD

0010  BA 0001                         MOV     DX,0001H                ; WRITE THE INIT DATA PATTERN
0013  B9 0010                         MOV     CX,16                   ; ROLL 16 BIT POSITIONS
0016  2B FF                   C1:     SUB     DI,DI                   ; START AT BEGINING OF BLOCK
0018  2B F6                           SUB     SI,SI                   ; INITIALIZE DESTINATION POINTER
001A  8B C2                           MOV     AX,DX                   ; GET THE PATTERN
001C  AB                              STOSW                           ; STORE DATA PATTERN
001D  2B F6                           SUB     SI,SI                   ; START AT BEGINNING
001F  AD                              LODSW                           ; GET THE FIRST WRITTEN
0020  33 C2                           XOR     AX,DX                   ; INSURE DATA AS EXPECTED
0022  74 03                           JZ      C1_A                    ;
0024  E9 00C5 R                       JMP     C13                     ; EXIT IF NOT
0027  D1 E2                   C1_A:   SHL     DX,1                    ; SHIFT BIT TO NEXT BIT POSITION
0029  E2 EB                           LOOP    C1                      ; LOOP TILL DONE

                              ;-------- CHECK CAS LINES FOR HIGH BYTE LOW BYTE

002B  2B FF                           SUB     DI,DI                   ; START AT BEGINING OF BLOCK
002D  2B F6                           SUB     SI,SI                   ; INITIALIZE DESTINATION POINTER
002F  2B C0                           SUB     AX,AX                   ; WRITE 0
0031  BA FF00                         MOV     DX,0FF00H               ;
0034  AB                              STOSW                           ; STORE DATA PATTERN
0035  BF 0001                         MOV     DI,1                    ; AT THE FIRST ODD LOCATION
0038  C6 05 FF                        MOV     BYTE PTR [DI],0FFH      ; WRITE A BYTE OF FF
003B  2B FF                           SUB     DI,DI
003D  8B 05                           MOV     AX,WORD PTR [DI]        ; GET THE DATA
003F  33 C2                           XOR     AX,DX                   ; CHECK THE FIRST WRITTEN
0041  74 03                           JZ      C1_B
0043  E9 00C5 R                       JMP     C13                     ; EXIT IF NOT

0046  2B FF                   C1_B:   SUB     DI,DI                   ; START AT BEGINING OF BLOCK
0048  2B C0                           SUB     AX,AX                   ; WRITE 0
004A  BA 00FF                         MOV     DX,000FFH               ;
004D  AB                              STOSW                           ; STORE DATA PATTERN
004E  2B FF                           SUB     DI,DI                   ; AT THE FIRST EVEN LOCATION
0050  C6 05 FF                        MOV     BYTE PTR [DI],0FFH      ; WRITE A BYTE OF FF
0053  2B FF                           SUB     DI,DI                   ; BUS SETTLE
0055  8B 05                           MOV     AX,WORD PTR [DI]        ; GET THE DATA
0057  33 C2                           XOR     AX,DX                   ; CHECK THE FIRST WRITTEN
0059  75 6A                           JNZ     C13                     ; EXIT IF NOT

                              ;------- TEMP SAVE FOR AX (PUSH NOT ALLOWED)

005B  E6 89                           OUT     DMA_PAGE+8,AL           ; SAVE AX
005D  86 C4                           XCHG    AL,AH                   ;
005F  EB 00                           JMP     SHORT $+2               ;
0061  E6 8A                           OUT     DMA_PAGE+9,AL           ;
```

SECTION 5

```
                                ;-------- CHECK IO OR BASE RAM

0063  E4 61                            IN      AL,PORT_B               ; CHECK FOR IO/PAR CHECK
0065  24 C0                            AND     AL,PARITY_ERR           ; STRIP UNWANTED BITS
0067  86 C4                            XCHG    AL,AH                   ; SAVE ERROR
0069  E4 87                            IN      AL,DMA_PAGE+6           ; CHECK FOR R/W OR IO ERR
006B  22 E0                            AND     AH,AL                   ;

                                ;------- RESTORE AX

006D  E4 8A                            IN      AL,DMA_PAGE+9           ; GET AH
006F  86 C4                            XCHG    AL,AH                   ;
0071  E4 89                            IN      AL,DMA_PAGE+8           ; GET AL

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

0073  75 50                            JNZ     C13                     ; GO IF YES
0075  BA AA55                          MOV     DX,0AA55H               ; WRITE THE INIT DATA PATTERN
0078  2B FF                     C3:    SUB     DI,DI                   ; START AT BEGINING OF BLOCK
007A  2B F6                     C4:    SUB     SI,SI                   ; INITIALIZE DESTINATION POINTER
007C  8B CB                            MOV     CX,BX                   ; SETUP BYTE COUNT FOR LOOP
007E  8B C2                            MOV     AX,DX                   ; GET THE PATTERN
0080  F3/ AB                    C5:    REP     STOSW                   ; STORE 64K BYTES (32K WORDS)
0082  8B CB                            MOV     CX,BX                   ; SET COUNT
0084  2B F6                            SUB     SI,SI                   ; START AT BEGINNING
0086  AD                        C6:    LODSW                           ; GET THE FIRST WRITTEN
0087  33 C2                            XOR     AX,DX                   ; INSURE DATA AS EXPECTED
0089  75 3A                            JNZ     C13                     ; EXIT IF NOT
008B  E2 F9                            LOOP    C6                      ; LOOP TILL DONE

                                ;------- TEMP SAVE FOR AX (PUSH NOT ALLOWED)

008D  E6 89                            OUT     DMA_PAGE+8,AL           ; SAVE AX
008F  86 C4                            XCHG    AL,AH                   ;
0091  EB 00                            JMP     SHORT $+2               ;
0093  E6 8A                            OUT     DMA_PAGE+9,AL           ;

                                ;-------- CHECK IO OR BASE RAM

0095  E4 61                            IN      AL,PORT_B               ; CHECK FOR IO/PAR CHECK
0097  24 C0                            AND     AL,PARITY_ERR           ; STRIP UNWANTED BITS
0099  86 C4                            XCHG    AL,AH                   ; SAVE ERROR
009B  E4 87                            IN      AL,DMA_PAGE+6           ; CHECK FOR R/W OR IO ERR
009D  22 E0                            AND     AH,AL                   ;

                                ;------- RESTORE AX

009F  E4 8A                            IN      AL,DMA_PAGE+9           ; GET AH
00A1  86 C4                            XCHG    AL,AH                   ;
00A3  E4 89                            IN      AL,DMA_PAGE+8           ; GET AL

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

00A5  75 1E                            JNZ     C13                     ; GO IF YES

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

00A7  23 D2                            AND     DX,DX                   ; ENDING ZERO PATTERN WRITTEN TO STG ?
00A9  74 1A                            JZ      C14                     ; YES - RETURN TO CALLER WITH AL=0

                                ;------- SETUP NEXT PATTERN

00AB  81 FA 55AA                       CMP     DX,055AAH               ; CHECK IF LAST PATTERN =55AA
00AF  74 0F                            JZ      C9                      ; GO IF NOT
00B1  81 FA 0101                       CMP     DX,0101H                ; LAST PATTERN 0101?
00B5  74 0F                            JZ      C10                     ; GO IF YES
00B7  BA 55AA                          MOV     DX,055AAH               ; WRITE 55AA TO STORAGE
00BA  EB BC                            JMP     C3                      ;

                                ;-------- LAST PATTERN = 0000

00BC  2B D2                     C8:    SUB     DX,DX                   ; WRITE 0000 TO STORAGE
00BE  EB B8                            JMP     C3                      ;

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

00C0  BA 0101                   C9:    MOV     DX,0101H                ; WRITE 0101 TO STORAGE
00C3  EB B3                            JMP     C3

                                ;-------- EXIT

00C5                            C13:
00C5  C3                        C14:   RET

                                ;------- CHECKER BOARD TEST

00C6  2B FF                     C10:   SUB     DI,DI                   ; POINT TO START OF BLOCK
00C8  8B CB                            MOV     CX,BX                   ; GET THE BLOCK COUNT
00CA  D1 E9                            SHR     CX,1                    ; DIVIDE BY 2
00CC  B8 5555                   C11:   MOV     AX,0101010101010101B    ; FIRST CHECKER PATTERN
00CF  AB                               STOSW                           ; WRITE IT
00D0  B8 AAAA                          MOV     AX,1010101010101010B    ; SECOND CHECKER PATTERN
00D3  AB                               STOSW                           ; WRITE IT
00D4  E2 F6                            LOOP    C11                     ; DO IT FOR CX COUNT
00D6  2B F6                            SUB     SI,SI                   ; POINT TO START OF BLOCK
00D8  8B CB                            MOV     CX,BX                   ; GET THE BLOCK COUNT
00DA  D1 E9                            SHR     CX,1                    ; DIVIDE BY 2
00DC  AD                        C12:   LODSW                           ; GET THE DATA
00DD  35 5555                          XOR     AX,0101010101010101B    ; CHECK CORRECT
00E0  75 E3                            JNZ     C13                     ; EXIT IF NOT
00E2  AD                               LODSW                           ; GET NEXT DATA
00E3  35 AAAA                          XOR     AX,1010101010101010B    ;
00E6  75 DD                            JNZ     C13                     ; GO IF NOT CORRECT
00E8  E2 F2                            LOOP    C12                     ; CONTINUE TILL DONE

                                ;------- TEMP SAVE FOR AX (PUSH NOT ALLOWED)

00EA  E6 89                            OUT     DMA_PAGE+8,AL           ; SAVE AX
00EC  86 C4                            XCHG    AL,AH                   ;
```

```
                          TITLE 12/28/83 TEST7 EXCEPTION INTERRUPT TEST
                          ;-----------------------------------------------------------
                          ; TEST.20                                                   :
                          ;       ADDITIONAL PROTECTED (VIRTUAL MODE) TEST            :
                          ; DESCRIPTION                                               :
                          ;       THE PROCESSOR IS PUT IN PROTECTED MODE AND          :
                          ;       THE FOLLOWING FUNCTIONS ARE VERIFIED                :
                          ;                                                           :
                          ;       1. VERIFY PROTECTED MODE                            :
                          ;          THE MACHINE STATUS IS CHECK FOR VIRTUAL MODE     :
                          ;       2. PROGRAMMED INTERRUPT TEST                        :
                          ;          AN PROGRAMMED INTERRUPT 32 IS ISSUED AND         :
                          ;          AND VERIFIED                                     :
                          ;       3. EXCEPTION INT 13 TEST                            :
                          ;          A DESCRIPTOR SEGMENT LIMIT IS SET TO ZERO        :
                          ;          AND A WRITE TO THAT SEGMENT IS ATTEMPTED         :
                          ;          AN EXCEPTION 13 IS EXPECTED AND VERIFIED         :
                          ;       4. LDT/SDT LTR/STR TEST                             :
                          ;          LOAD LDT REGISTER AND VERIFY CORRECT             :
                          ;          LOAD TASK REGISTER AND VERIFY CORRECT            :
                          ;          THEY ARE VERIFIED VIA THE STORE INSTRUCTION      :
                          ;       5. THE CONTROL FLAGS OF THE 286 FOR DIRECTION       :
                          ;          ARE VERIFIED VIA THE STD AND CLD COMMANDS        :
                          ;          IN PROTECTED MODE                                :
                          ;       6. BOUND INSTRUCTION TEST (EXC INT 5)               :
                          ;          CREATE A SIGNED ARRAY INDEX WITHIN AND           :
                          ;          OUTSIDE THE LIMITS.  CHECK THAT NO EXC INT       :
                          ;          IF WITHIN LIMIT AND THAT AN EXC INT 5            :
                          ;          OCCURS IF OUTSIDE THE LIMITS.                    :
                          ;       7. PUSH ALL POP ALL TEST                            :
                          ;          SET ALL GENERAL PURPOSE REGS TO DIFFERENT        :
                          ;          VALUES ISSUE A PUSH ALL, CLEAR THE REGS          :
                          ;          ISSUE A POP ALL AND VERIFY CORRECT.              :
                          ;       8. CHECK THE VERR/VERW INSTRUCTIONS                 :
                          ;          THE ACCESS BYTE IS SET TO READ ONLY THEN TO      :
                          ;          A WRITE ONLY AND THE VERR/VERW INST ARE          :
                          ;          VERIFIED.                                        :
                          ;       9. CAUSE AN INTERRUPT 13 VIA A WRITE TO A           :
                          ;          READ ONLY SEGMENT                                :
                          ;      10. VERIFY THE ARPL INSTRUCTION FUNCTIONS            :
                          ;          SET THE RPL FIELD OF A SELECTOR AND              :
                          ;          VERIFY THAT CURRENT SELECTOR RPL IS SET          :
                          ;          CORRECTLY.                                       :
                          ;      11. VERIFY THE LAR INSTRUCTION FUNCTIONS             :
                          ;      12. VERIFY THE LSL INSTRUCTION FUNCTIONS             :
                          ;      13. LOW MEG CHIP SELECT TEST                         :
                          ;-----------------------------------------------------------
                          .LIST
                          PUBLIC  POST7

                        C INCLUDE SEGMENT.SRC
0000                    C CODE SEGMENT BYTE PUBLIC
                        C

                          EXTRN   E_MSG:NEAR
                          EXTRN   XPC_BYTE:NEAR
                          EXTRN   F1:NEAR
                          EXTRN   VECTOR_TABLE:NEAR
                          EXTRN   PRINT_SCREEN:NEAR
                          EXTRN   BLINK_INT:NEAR
                          EXTRN   PRT_HEX:NEAR
                          EXTRN   F3B:NEAR
                          EXTRN   PRT_SEG:NEAR
                          EXTRN   XPC_BYTE:NEAR
                          EXTRN   E1:NEAR

                          EXTRN   F3:NEAR
                          EXTRN   ERR_BEEP:NEAR
                          EXTRN   P_MSG:NEAR
                          EXTRN   START_1:NEAR
                          EXTRN   F4:NEAR
                          EXTRN   F4E:NEAR
                          EXTRN   F3A:NEAR
                          EXTRN   DISK_BASE:NEAR
                          EXTRN   F3D:NEAR
                          EXTRN   F3D1:NEAR
                          EXTRN   PROC_SHUTDOWN:NEAR
                          EXTRN   SYSINIT1:NEAR
                          EXTRN   PROT_PRT_HEX:NEAR
                          EXTRN   DISK_IO:NEAR
                          EXTRN   HD_INT:NEAR
                          EXTRN   C8042:NEAR
                          EXTRN   OBF_42:NEAR
                          EXTRN   STGTST_CNT:NEAR
                          EXTRN   BOOT_STRAP_1:NEAR
                          EXTRN   XMIT_8042:NEAR
                          EXTRN   ROM_ERR:NEAR
                          EXTRN   DDS:NEAR
                          EXTRN   CM1:NEAR
                          EXTRN   CM2:NEAR
                          EXTRN   CM3:NEAR
                          EXTRN   LOCK:NEAR
                          EXTRN   DISK_SETUP:NEAR
                          EXTRN   ADERR:NEAR
                          EXTRN   ADERR1:NEAR
                                  ASSUME  CS:CODE, DS:DATA
0000                      POST7   PROC
0000  E8 0000 E                   CALL    DDS                     ; SET DATA SEGMENT
0003  B0 F0                       MOV     AL,0F0H                 ;<><><><><><><><><><><><><>
0005  E6 80                       OUT     MFG_PORT,AL             ;<><><>CHECKPOINT F0 <><><>

                          ;------- SET SHUTDOWN RETURN 7

0007  B0 8F                       MOV     AL,SHUT_DOWN            ; ADDR FOR SHUTDOWN BYTE
0009  E6 70                       OUT     CMOS_PORT,AL            ;
000B  B0 07                       MOV     AL,7                    ; SET ERROR EXIT (DOUBLE EXECPTION?)
000D  EB 00                       JMP     SHORT $+2               ; IO DELAY
000F  E6 71                       OUT     CMOS_PORT+1,AL          ;
                                  ;---------------------
                          ;------- ENABLE PROTECTED MODE
                                  ;---------------------
0011  BC 0000                     MOV     SP,POST_SS              ; SET STACK FOR SYSINIT1
0014  8E D4                       MOV     SS,SP                   ;
0016  BC 8000                     MOV     SP,POST_SP              ;

0019  E8 0000 E                   CALL    SYSINIT1                ; GO ENABLE PROTECTED MODE

                          ;------- SET TEMPORY STACK

001C  B8 0008                     MOV     AX,GDT_PTR              ;
001F  8E C0                       MOV     ES,AX                   ;
0021  8E D8                       MOV     DS,AX
0023  26: C7 06 005A 0000         MOV     ES:SS_TEMP.BASE_LO_WORD,0
002A  26: C6 06 005C 00           MOV     BYTE PTR ES:(SS_TEMP.BASE_HI_BYTE),0
0030  BE 0058                     MOV     SI,SS_TEMP
0033  8E D6                       MOV     SS,SI
```

SECTION 5

```
0035  BC FFFD                          MOV     SP,MAX_SEG_LEN-2
                                       ;---------------------
                               ;-------VERIFY PROTECTED MODE
                                       ;---------------------

                                       SMSW    AX                          ; GET THE MACHINE STATUS WORD
0038  0F                       +       DB      00FH
0039                           + ??0000 LABEL  BYTE
0039  D1 E0                    +       SHL     AX,1
003B                           + ??0001 LABEL  BYTE
0039                           +       ORG     OFFSET CS:??0000
0039  01                       +       DB      001H
003B                           +       ORG     OFFSET CS:??0001
003B  A9 0001                          TEST    AX,VIRTUAL_ENABLE           ; ARE WE IN PROTECTED MODE
003E  75 03                            JNZ     T7_1
0040  E9 02EA R                        JMP     ERROR_EXIT                  ; ERROR IF NOT

0043  B0 F1                    T7_1:   MOV     AL,0F1H             ;<><><><><><><><><><><><><>
0045  E6 80                            OUT     MFG_PORT,AL         ;<><><>CHECKPOINT F1 <><><><>

                                       ;----------------------------------------
                               ;-------- INTERRUPT TEST (PROGRAMMED INTERRUPT 32)
                                       ;----------------------------------------
0047  B0 AF                            MOV     AL,0AFH                     ; SET EXCEPTION FLAG
0049  E6 8B                            OUT     DMA_PAGE+0AH,AL             ;   FOR INT 10
004B  CD 20                            INT     32                          ; INTERRUPT
004D  2B C9                            SUB     CX,CX                       ; WAIT FOR INT
004F  E4 8B                    LOOP1:  IN      AL,DMA_PAGE+0AH             ;
0051  22 C0                            AND     AL,AL                       ; DID THE INTERRUPT OCCUR?
0053  E0 FA                            LOOPNZ  LOOP1                       ;
0055  74 03                            JZ      T7_2
0057  E9 02EA R                        JMP     ERROR_EXIT                  ; MISSING INTERRUPT

                               ;-------- CAUSE AN EXCEPTION INTERRUPT (GENERAL PROTECTION INT 13D)

005A  B0 F2                    T7_2:   MOV     AL,0F2H             ;<><><><><><><><><><><><><>
005C  E6 80                            OUT     MFG_PORT,AL         ;<><><>CHECKPOINT F2 <><><><>

005E  B0 9D                            MOV     AL,9DH                      ; SET INT 13 FLAG
0060  E6 8B                            OUT     DMA_PAGE+0AH,AL             ;  FOR THE INT HANDLER

                               ;------- MODIFY DESCRIPTOR TABLES
                                                                           ;
                                       ;----------------------------------------
                               ;-------SET TEMP ES DESCRIPTOR TO SEGMENT LIMIT
                                       ;----------------------------------------

0062  C7 06 0048 0000                  MOV     DS:ES_TEMP.SEG_LIMIT,0      ; SET SEGMENT TO 0

                               ;------- CPLO, DATA ACCESS RIGHTS

0068  C6 06 004D 93                    MOV     BYTE PTR DS:(ES_TEMP.DATA_ACC_RIGHTS),CPLO_DATA_ACCESS
006D  C6 06 004C 01                    MOV     BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE),01  ; DO ALL TESTS ON 2ND 64K
0072  C7 06 004A 0000                  MOV     WORD PTR DS:(ES_TEMP.BASE_LO_WORD),0

                               ;-------- SET ES REGISTER

0078  B8 0048                          MOV     AX,ES_TEMP                  ; LOAD ES
007B  8E C0                            MOV     ES,AX                       ;

                               ;------- CAUSE AN EXCEPTION 13 INTERRUPT
007D  2B FF                            SUB     DI,DI                       ;
007F  26: 8B 05                        MOV     AX,ES:[DI]                  ; THIS SHOULD CAUSE AND EXCEPTION

0082  2B C9                            SUB     CX,CX                       ; WAIT FOR INT
0084  E4 8B                    LOOP2:  IN      AL,DMA_PAGE+0AH             ;
0086  22 C0                            AND     AL,AL                       ; DID THE INTERRUPT OCCUR?
0088  E0 FA                            LOOPNZ  LOOP2                       ;
008A  74 03                            JZ      T7_3                        ; CONTINUE IF INTERRUPT
008C  E9 02EA R                        JMP     ERROR_EXIT                  ; MISSING INTERRUPT

008F                           T7_3:
                               ;------------------------------------------------
                               ;                                               :
                               ;       VERIFY 286 LDT/SDT LTR/STR              :
                               ;       INSTRUCTIONS                            :
                               ; DESCRIPTION                                   :
                               ;       LOAD LDT  REGISTERS WITH A DESCRIPTOR   :
                               ;       VERIFY CORRECT                          :
                               ;------------------------------------------------

                               ;------- WRITE TO 286 LDT REGISTER

008F  B0 F3                            MOV     AL,0F3H                     ;<><><><><><><><><><><><><>
0091  E6 80                            OUT     MFG_PORT,AL                 ;<><><>CHECKPOINT F3 <><><><>
0093  BF 0078                          MOV     DI,POST_LDTR
                                       LLDT    DI                  ;  REGISTER FROM THIS AREA
0096  0F                       +       DB      00FH
0097                           + ??0002 LABEL  BYTE
0097  8B D7                    +       MOV     DX,DI
0099                           + ??0003 LABEL  BYTE
0097                           +       ORG     OFFSET CS:??0002
0097  00                       +       DB      000H
0099                           +       ORG     OFFSET CS:??0003

                               ;------- READ AND VERIFY 286 LDT SELECTOR

0099  2B C0                            SUB     AX,AX               ; CLEAR AX
                                       SLDT    AX                  ; GET THE LDT SELECTOR
009B  0F                       +       DB      00FH
009C                           + ??0004 LABEL  BYTE
009C  03 C0                    +       ADD     AX,AX
009E                           + ??0005 LABEL  BYTE
009C                           +       ORG     OFFSET CS:??0004
009C  00                       +       DB      000H
009E                           +       ORG     OFFSET CS:??0005
009E  25 00F8                          AND     AX,0F8H             ; STRIP TI/RPL
00A1  3D 0078                          CMP     AX,POST_LDTR        ; CORRECT SELECTOR?
00A4  75 1B                            JNZ     ERROR               ; GO IF NOT

                               ;-------- WRITE TO 286 TR

00A6  BF 0068                          MOV     DI,POST_TR
                                       LTR     DI                 ;  REGISTER FROM THIS AREA
00A9  0F                       +       DB      00FH
00AA                           + ??0006 LABEL  BYTE
00AA  8B DF                    +       MOV     BX,DI
00AC                           + ??0007 LABEL  BYTE
00AA                           +       ORG     OFFSET CS:??0006
00AA  00                       +       DB      000H
00AC                           +       ORG     OFFSET CS:??0007

                               ;------- VERIFY 286 TR REGISTERS
```

```
00AC  2B C0                                     SUB     AX,AX                 ;

                                                STR     AX                    ; GET THE TR  REG
00AE  0F                             +          DB      00FH
00AF                                 + ??0008   LABEL   BYTE
00AF  8B C8                          +          MOV     CX,AX
00B1                                 + ??0009   LABEL   BYTE
00AF                                 +          ORG     OFFSET CS:??0008
00AF  00                             +          DB      000H
00B1                                 +          ORG     OFFSET CS:??0009
00B1  25 00F8                                   AND     AX,0F8H               ;
00B4  3D 0068                                   CMP     AX,POST_TR            ; CORRECT SELECTOR?
00B7  75 08                                     JNZ     ERROR

                                     ;-------- TEST 286 CONTROL FLAGS

00B9  FD                                        STD                           ; SET DIRECTION FLAG FOR DECREMENT
00BA  9C                                        PUSHF                         ; GET THE FLAGS
00BB  58                                        POP     AX                    ;
00BC  A9 0200                                   TEST    AX,0200H              ; INTERRUPT FLAG SHOULD BE OFF
00BF  74 03                                     JZ      T7_4                  ; CONTINUE IF OFF
00C1  E9 02EA R                      ERROR:     JMP     ERROR_EXIT            ; GO IF NOT
00C4                                 T7_4:
00C4  A9 0400                                   TEST    AX,0400H              ; CHECK DIRECTION FLAG
00C7  75 03                                     JNZ     T7_5                  ;
00C9  E9 02EA R                                 JMP     ERROR_EXIT            ; GO IF NOT SET
00CC  FC                             T7_5:      CLD                           ; CLEAR DIRECTION FLAG

00CD  9C                                        PUSHF                         ; INSURE DIRECTION FLAG IS RESET
00CE  58                                        POP     AX                    ;
00CF  A9 0400                                   TEST    AX,0400H              ;
00D2  74 03                                     JZ      T7_6                  ;
00D4  E9 02EA R                                 JMP     ERROR_EXIT            ; GO IF NOT
00D7                                 T7_6:
                                     ;----------------------------------------------
                                     ;          VERIFY 286 BOUND INSTRUCTION       :
                                     ; DESCRIPTION                                 :
                                     ;          CREATE A SIGNED ARRAY INDEX WITHIN AND  :
                                     ;          OUTSIDE THE LIMITS (EXPECT INT 5)  :
                                     ;----------------------------------------------

00D7  B0 F4                                     MOV     AL,0F4H                         ;<><><><><><><><><><><><><><>
00D9  E6 80                                     OUT     MFG_PORT,AL                     ;<><><>CHECKPOINT F4 <><><>
00DB  B8 0048                                   MOV     AX,ES_TEMP            ; LOAD ES REGISTER
00DE  8E C0                                     MOV     ES,AX                 ;

                                     ;------- CHECK BOUND FUNCTIONS CORRECTLY

00E0  2B FF                                     SUB     DI,DI                 ; POINT BEGINING OF THE BLOCK
00E2  26: C7 05 0000                            MOV     WORD PTR ES:[DI],0    ; SET FIRST WORD TO ZERO

00E7  26: C7 45 02 7FFF                         MOV     WORD PTR ES:[DI+2],07FFFH ; SET SECOND TO 07FFFH

00ED  B0 95                                     MOV     AL,095H               ; SET INTERRUPT 5 FLAG
00EF  E6 8B                                     OUT     DMA_PAGE+0AH,AL       ;

00F1  B8 1000                                   MOV     AX,1000H              ; SET AX WITHIN BOUNDS
                                                SEGOV   ES                    ; USE THE ES REG
00F4  26                             +          DB      026H
                                                BOUND   AX,[DI]               ;
00F5                                 + ??000B   LABEL   BYTE
00F5  8B 05                          +          MOV     AX,[DI]
00F7                                 + ??000C   LABEL   BYTE
00F5                                 +          ORG     OFFSET CS:??000B
00F5  62                             +          DB      062H
00F7                                 +          ORG     OFFSET CS:??000C
00F7  2B C9                                     SUB     CX,CX                 ; WAIT FOR POSSIBLE INTERRUPT
00F9  E2 FE                          LOOPA:     LOOP    LOOPA                 ;
00FB  E4 8B                                     IN      AL,DMA_PAGE+0AH       ; GET THE RESULTS
00FD  3C 00                                     CMP     AL,0                  ; DID AN INTERRUPT OCCUR?
00FF  75 03                                     JNZ     T7_7                  ; CONTINUE IF NOT
0101  E9 02EA R                                 JMP     ERROR_EXIT            ; GO IF YES
0104                                 T7_7:
                                     ;-------- CHECK LOW BOUND WORD CAUSES INT 5

0104  2B FF                                     SUB     DI,DI                 ; POINT BEGINING OF THE BLOCK
0106  26: C7 05 3FF0                            MOV     WORD PTR ES:[DI],03FF0H ; SET FIRST WORD TO 03FF0H

010B  B8 1000                                   MOV     AX,1000H              ; SET AX OUT OF BOUNDS
                                                SEGOV   ES                    ; USE THE ES REG
010E  26                             +          DB      026H
                                                BOUND   AX,[DI]               ;
010F                                 + ??000E   LABEL   BYTE
010F  8B 05                          +          MOV     AX,[DI]
0111                                 + ??000F   LABEL   BYTE
010F                                 +          ORG     OFFSET CS:??000E
010F  62                             +          DB      062H
0111                                 +          ORG     OFFSET CS:??000F
0111  2B C9                                     SUB     CX,CX                 ; WAIT FOR POSSIBLE INTERRUPT
0113                                 LOOPB:                                   ;
0113  E4 8B                                     IN      AL,DMA_PAGE+0AH       ; GET THE RESULTS
0115  3C 00                                     CMP     AL,0H                 ; DID AN INTERRUPT OCCUR?
0117  E0 FA                                     LOOPNZ  LOOPB                 ; TRY AGAIN
0119  74 03                                     JZ      T7_8                  ; CONTINUE IF INTERRUPT
011B  E9 02EA R                                 JMP     ERROR_EXIT            ; GO IF NO INTERRUPT

                                     ;-------- CHECK HIGH BOUND WORD CAUSES INT 5

011E  B0 95                          T7_8:      MOV     AL,95H                ; SET FLAG FOR INTERRUPT
0120  E6 8B                                     OUT     DMA_PAGE+0AH,AL       ;

0122  2B FF                                     SUB     DI,DI                 ; POINT BEGINING OF THE BLOCK
0124  26: C7 05 0000                            MOV     WORD PTR ES:[DI],0    ; SET FIRST WORD TO 0
0129  26: C7 45 02 0FFF                         MOV     WORD PTR ES:[DI+2],0FFFH ; SET SECOND TO 0FFFH
012F  B8 1000                                   MOV     AX,1000H              ; SET AX OUT OF BOUNDS
                                                SEGOV   ES                    ; USE THE ES REG
0132  26                             +          DB      026H
                                                BOUND   AX,[DI]               ;
0133                                 + ??0011   LABEL   BYTE
0133  8B 05                          +          MOV     AX,[DI]
0135                                 + ??0012   LABEL   BYTE
0133                                 +          ORG     OFFSET CS:??0011
0133  62                             +          DB      062H
0135                                 +          ORG     OFFSET CS:??0012
0135  2B C9                                     SUB     CX,CX                 ; WAIT FOR POSSIBLE INTERRUPT
0137                                 LOOPC:                                   ;
0137  E4 8B                                     IN      AL,DMA_PAGE+0AH       ; GET THE RESULTS
0139  3C 00                                     CMP     AL,0H                 ; DID AN INTERRUPT OCCUR?
013B  E0 FA                                     LOOPNZ  LOOPC                 ; TRY AGAIN
013D  74 03                                     JZ      T7_9                  ;
013F  E9 02EA R                                 JMP     ERROR_EXIT            ; GO IF NO INTERRUPT

                                     ;----------------------------------------------
                                     ;          VERIFY PUSH ALL AND POP ALL INSTRUCTIONS:
                                     ; DESCRIPTION                                 :
```

SECTION 5

```
                                  ;         SET REGISTERS TO A KNOWN VALUE AND        :
                                  ;         PUSH ALL.  RESET THE REGISTERS POPALL     :
                                  ;         AND VERIFY                                :
                                  ;-------------------------------------------------

0142                              T7_9:
0142  B0 F5                                MOV     AL,0F5H                    ;<><><><><><><><><><><><><><>
0144  E6 80                                OUT     MFG_PORT,AL                ;<><><>CHECKPOINT F5 <><><><>
0146  B8 0001                              MOV     AX,01                  ; SET AX=1
0149  8B D8                                MOV     BX,AX                  ; SET BX=2
014B  43                                   INC     BX                     ;
014C  8B CB                                MOV     CX,BX                  ; SET CX=3
014E  41                                   INC     CX                     ;
014F  8B D1                                MOV     DX,CX                  ;
0151  42                                   INC     DX                     ; SET DX=4
0152  8B FA                                MOV     DI,DX                  ;
0154  47                                   INC     DI                     ; SET DI=5
0155  8B F7                                MOV     SI,DI                  ;
0157  46                                   INC     SI                     ; SET SI=6
0158  55                                   PUSH    BP                     ; SAVE THE BP REGISTER
0159  8B EE                                MOV     BP,SI                  ; SET BP=7
015B  45                                   INC     BP                     ;
                                           PUSHA                          ; ISSUE THE PUSH ALL COMMAND
015C  60                          +        DB      060H
015D  2B C0                                SUB     AX,AX                  ; CLEAR ALL REGS
015F  8B D8                                MOV     BX,AX                  ;
0161  8B C8                                MOV     CX,AX                  ;
0163  8B D0                                MOV     DX,AX                  ;
0165  8B F8                                MOV     DI,AX                  ;
0167  8B F0                                MOV     SI,AX                  ;
0169  8B E8                                MOV     BP,AX                  ;
                                           POPA                           ; GET THE REGISTERS BACK
016B  61                          +        DB      061H
016C  83 FD 07                             CMP     BP,07                  ; BP SHOULD BE 7
016F  5D                                   POP     BP                     ; RESTORE BP
0170  75 21                                JNZ     ERROR_EXIT1            ; GO IF NOT
0172  3D 0001                              CMP     AX,01                  ; AX SHOULD BE 1
0175  75 1C                                JNZ     ERROR_EXIT1            ; GO IF NOT
0177  83 FB 02                             CMP     BX,02                  ; BX SHOULD BE 2
017A  75 17                                JNZ     ERROR_EXIT1            ; GO IF NOT
017C  83 F9 03                             CMP     CX,03                  ; CX SHOULD BE 3
017F  75 12                                JNZ     ERROR_EXIT1            ; GO IF NOT
0181  83 FA 04                             CMP     DX,04                  ; DX SHOULD BE 4
0184  75 0D                                JNZ     ERROR_EXIT1            ; GO IF NOT
0186  83 FF 05                             CMP     DI,05                  ; DI SHOULD BE 5
0189  75 08                                JNZ     ERROR_EXIT1            ; GO IF NOT
018B  83 FE 06                             CMP     SI,06                  ; SI SHOULD BE 6
018E  75 03                                JNZ     ERROR_EXIT1            ; GO IF NOT
0190  EB 04 90                             JMP     T7_10

                                  ;-----------ERROR EXIT
0193                              ERROR_EXIT1:
0193  E9 02EA R                            JMP     ERROR_EXIT
                                  ;-------------------------------------------------
                                  ;         VERIFY ACCESS RIGHTS FUNCTION CORRECTLY :
                                  ; DESCRIPTION                                     :
                                  ;         SET ACCESS RIGHTS OF DESCRIPTER TO        :
                                  ;         READ ONLY.  VERIFY THE VERW/VERR INSTR    :
                                  ;         ACCESS A READ ONLY WITH A WRITE AND       :
                                  ;         VERIFY AN EXCEPTION INT 13                :
                                  ;-------------------------------------------------

0196  B0 F6                       T7_10:   MOV     AL,0F6H                    ;<><><><><><><><><><><><><><>
0198  E6 80                                OUT     MFG_PORT,AL                ;<><><>CHECKPOINT F6 <><><><>
019A  C7 06 0048 FFFF                      MOV     DS:ES_TEMP.SEG_LIMIT,MAX_SEG_LEN  ; SET SEGMENT TO 0FFFFH
01A0  C6 06 004C 00                        MOV     BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE),0 ;SET THE ADDRESS
01A5  C7 06 004A F000                      MOV     DS:ES_TEMP.BASE_LO_WORD,0F000H

01AB  B8 0048                              MOV     AX,ES_TEMP             ; LOAD ES REGISTER
01AE  8E C0                                MOV     ES,AX                  ; THIS SEGMENT SHOULD BE WRITEABLE

                                  ;-------- INSURE ACCESS RIGHTS MAY BE WRITTEN

                                           SEGOV   DS                     ; SET SEGMENT OVERIDE TO START OF TABLE
01B0  3E                          +        DB      03EH
                                           VERW    AX                     ; CHECK THE ACCESS RIGHTS OF ES_TEMP
01B1  0F                          +        DB      00FH
01B2                              + ??0014 LABEL   BYTE
01B2  8B E8                       +        MOV     BP,AX
01B4                              + ??0015 LABEL   BYTE
01B2                              +        ORG     OFFSET CS:??0014
01B2  00                          +        DB      000H
01B4                              +        ORG     OFFSET CS:??0015

01B4  75 DD                                JNZ     ERROR_EXIT1            ; ERROR IF SEGMENT CAN NOT WRITE

                                  ;------- SET ACCESS RIGHTS TO READ ONLY

01B6  C6 06 004D 91                        MOV     BYTE PTR DS:(ES_TEMP.DATA_ACC_RIGHTS),91H

01BB  B8 0048                              MOV     AX,ES_TEMP             ; LOAD ES REGISTER
01BE  8E C0                                MOV     ES,AX                  ;

                                           SEGOV   DS                     ; SET SEGMENT OVERIDE TO START OF TABLE
01C0  3E                          +        DB      03EH
                                           VERW    AX                     ; CHECK THE ACCESS RIGHTS OF ES_TEMP
01C1  0F                          +        DB      00FH
01C2                              + ??0017 LABEL   BYTE
01C2  8B E8                       +        MOV     BP,AX
01C4                              + ??0018 LABEL   BYTE
01C2                              +        ORG     OFFSET CS:??0017
01C2  00                          +        DB      000H
01C4                              +        ORG     OFFSET CS:??0018

01C4  74 CD                                JZ      ERROR_EXIT1            ; ERROR IF SEGMENT IS WRITEABLE

01C6  B8 0048                              MOV     AX,ES_TEMP             ; INSURE THAT SEGMENT IS READABLE
                                           SEGOV   DS                     ;
01C9  3E                          +        DB      03EH
                                           VERR    AX                     ;
01CA  0F                          +        DB      00FH
01CB                              + ??001A LABEL   BYTE
01CB  8B E0                       +        MOV     SP,AX
01CD                              + ??001B LABEL   BYTE
01CB                              +        ORG     OFFSET CS:??001A
01CB  00                          +        DB      000H
01CD                              +        ORG     OFFSET CS:??001B

01CD  75 C4                                JNZ     ERROR_EXIT1            ; GO IF SEGMENT NOT READABLE

                                  ;------- CAUSE AN EXCEPTION 13 INTERRUPT

01CF  B0 9D                                MOV     AL,09DH                        ; SET EXCEPTION FLAG
01D1  E6 8B                                OUT     DMA_PAGE+0AH,AL                ;   FOR INT 13

01D3  2B F6                                SUB     SI,SI                          ;
```

```
01D5  26: C6 04 00                MOV    BYTE PTR ES:[SI],00            ; WRITE A BYTE THAT SHOULD
                                                                        ;  CAUSE AN EXCEPTION
01D9  2B C9                       SUB    CX,CX                          ; WAIT FOR INT
01DB  E4 8B               LOOPD:  IN     AL,DMA_PAGE+0AH                ;
01DD  22 C0                       AND    AL,AL                          ; DID THE INTERRUPT OCCUR?
01DF  E0 FA                       LOOPNZ LOOPD                          ;
01E1  75 B0                       JNZ    ERROR_EXIT1                    ; MISSING INTERRUPT

                          ;-------- RESTORE THE ACCESS RIGHTS BYTE

01E3  C6 06 004D 93               MOV    BYTE PTR DS:(ES_TEMP.DATA_ACC_RIGHTS),CPLO_DATA_ACCESS

                          ;-----------------------------------------------
                          ;          VERIFY ADJUST RPL FIELD OF SELECTOR    :
                          ;          INSTRUCTION (ARPL) FUNCTIONS           :
                          ; DESCRIPTION                                     :
                          ;          SET THE RPL FIELD OF A SELECTOR        :
                          ;          AND VERIFY THAT THE ZERO FLAG IS SET   :
                          ;          CORRECTLY AND THAT THE SELECTOR RPL    :
                          ;          FIELD IS SET CORRECTLY                 :
                          ;-----------------------------------------------

01E8  B0 F7                       MOV    AL,0F7H                        ;<><><><><><><><><><><><><><>
01EA  E6 80                       OUT    MFG_PORT,AL                    ;<><><>CHECKPOINT F7 <><><>
01EC  B8 0048                     MOV    AX,ES_TEMP                ; PUT A SELECTOR IN AX
01EF  BB 0060                     MOV    BX,DS_TEMP                ; PUT A SELECTOR IN BX

01F2  0D 0003                     OR     AX,03H                    ; MAKE ACCESS OF AX < BX

                          ;-------- NOTE BX = FIRST OPERAND  AX = SECOND OPERAND

                                  ARPL   AX,BX                     ; ISSUE THE RPL COMMAND
01F5                      + ??001C LABEL BYTE                      ; NOTE: SOURCE / TARGET REGS ARE REVERSED
01F5  8B C3               +       MOV    AX,BX             ;       DUE TO OPCODE BIT 1
01F7                      + ??001D LABEL BYTE
01F5                      +       ORG    OFFSET CS:??001C
01F5  63                  +       DB     063H
01F7                      +       ORG    OFFSET CS:??001D
01F7  75 9A                       JNZ    ERROR_EXIT1               ; GO IF RPL WAS NOT CHANGED
01F9  80 E3 03                    AND    BL,03H                    ; STRIP UNWANTED BITS
01FC  80 FB 03                    CMP    BL,03H                    ; AS EXPECTED?
01FF  75 92                       JNZ    ERROR_EXIT1               ; GO IF NOT

                          ;-------- CHECK THAT ACCESS RIGHTS DO NOT CHANGE

0201  BB 0060                     MOV    BX,DS_TEMP                ; PUT A SELECTOR IN BX
0204  B8 0048                     MOV    AX,ES_TEMP                ; PUT A SELECTOR IN AX
0207  80 CB 03                    OR     BL,03H                    ; MAKE ACCESS OF BX < AX

                          ;-------- NOTE BX = FIRST OPERAND  AX = SECOND OPERAND

                                  ARPL   AX,BX                     ; ISSUE THE RPL COMMAND
020A                      + ??001E LABEL BYTE                      ; NOTE: SOURCE / TARGET REGS ARE REVERSED
020A  8B C3               +       MOV    AX,BX             ;       DUE TO OPCODE BIT 1
020C                      + ??001F LABEL BYTE
020A                      +       ORG    OFFSET CS:??001E
020A  63                  +       DB     063H
020C                      +       ORG    OFFSET CS:??001F
020C  74 85                       JZ     ERROR_EXIT1               ; GO IF RPL WAS NOT CHANGED
020E  80 E3 03                    AND    BL,03H                    ; STRIP UNWANTED BITS
0211  80 FB 03                    CMP    BL,03H                    ; AS EXPECTED?
0214  75 2F                       JNZ    ERROR_EXIT2               ; GO IF NOT
                          ;-----------------------------------------------
                          ;          VERIFY LOAD SEGMENT LIMIT (LSL)        :
                          ;          AND LOAD ACCESS RIGHTS (LAR) INSTR     :
                          ;-----------------------------------------------
                                   ;=======================
                          ;-------- CHECK THE LAR INSTRUCTION
                                   ;=======================

0216  B0 F8                       MOV    AL,0F8H                        ;<><><><><><><><><><><><><><>
0218  E6 80                       OUT    MFG_PORT,AL                    ;<><><>CHECKPOINT F8 <><><>

                          ;------- SET THE DESCRIPTOR TO LEVEL 3

021A  C6 06 004D F3               MOV    BYTE PTR DS:(ES_TEMP.DATA_ACC_RIGHTS),CPL3_DATA_ACCESS

021F  BB 0048                     MOV    BX,ES_TEMP
0222  2B C0                       SUB    AX,AX                     ; CLEAR AX

                          ;-------- GET THE CURRENT DESCRIPTOR"S ACCESS RIGHTS

                                  LAR    AX,BX                     ; ISSUE THE LAR COMMAND
0224  0F                  +       DB     00FH
0225                      + ??0020 LABEL BYTE
0225  8B C3               +       MOV    AX,BX
0227                      + ??0021 LABEL BYTE
0225                      +       ORG    OFFSET CS:??0020
0225  02                  +       DB     002H
0227                      +       ORG    OFFSET CS:??0021

                          ;-------- INSURE THE DESCRIPTOR WAS VISABLE

0227  75 1C                       JNZ    ERROR_EXIT2               ; GO IF LAR WAS NOT CHANGED

                          ;-------- THE DISCRIPTOR"S ACCESS RIGHTS MUST BE 3


0229  80 FC F3                    CMP    AH,CPL3_DATA_ACCESS       ; AS EXPECTED?
022C  75 17                       JNZ    ERROR_EXIT2               ; GO IF NOT
                                  ;=================================
                          ;-------CHECK THE LSL (LOAD SEGMENT LIMITS)
                                  ;=================================

022E  B0 F9                       MOV    AL,0F9H                        ;<><><><><><><><><><><><><><>
0230  E6 80                       OUT    MFG_PORT,AL                    ;<><><>CHECKPOINT F9 <><><>
0232  C7 06 0048 AAAA             MOV    DS:ES_TEMP.SEG_LIMIT,0AAAAH      ; SET SEGMENT LIMIT TO 0AAAAH

0238  C6 06 004D 93               MOV    BYTE PTR DS:(ES_TEMP.DATA_ACC_RIGHTS),CPLO_DATA_ACCESS

023D  B8 0048                     MOV    AX,ES_TEMP                ; LOAD ES REGISTER

                                  LSL    BX,AX                     ; GET THE DESCRIPTOR SEGMENT LIMIT
0240  0F                  +       DB     00FH
0241                      + ??0022 LABEL BYTE
0241  8B D8               +       MOV    BX,AX
0243                      + ??0023 LABEL BYTE
0241                      +       ORG    OFFSET CS:??0022
0241  03                  +       DB     003H
0243                      +       ORG    OFFSET CS:??0023
0243  74 03                       JZ     R07                       ; GO IF OK

0245                      ERROR_EXIT2:
0245  E9 02EA R                   JMP    ERROR_EXIT                ; GO IF NOT SUCCESSFUL
```

SECTION 5

```
0248  81 FB AAAA               R07:    CMP     BX,0AAAAH                 ; INSURE CORRECT SEGMENT LIMIT

024C  C7 06 0048 5555                  MOV     DS:ES_TEMP.SEG_LIMIT,05555H     ;SET THE SETMENT LIMIT TO 05555H

0252  B8 0048                          MOV     AX,ES_TEMP
                                       LSL     BX,AX                     ; GET THE DESCRIPTOR SEGMENT LIMIT
0255  0F                       +       DB      00FH
0256                           + ??0024 LABEL  BYTE
0256  8B D8                    +       MOV     BX,AX
0258                           + ??0025 LABEL  BYTE
0256                           +       ORG     OFFSET CS:??0024
0256  03                       +       DB      003H
0258                           +       ORG     OFFSET CS:??0025
0258  75 EB                            JNZ     ERROR_EXIT2               ; GO IF NOT SUCCESSFUL

025A  81 FB 5555                       CMP     BX,05555H                 ; INSURE CORRECT SEGMENT LIMIT
025E  75 E5                            JNZ     ERROR_EXIT2               ; GO IF NOT

                               ;-----------------------------------------------------------
                               ; LOW MEG CHIP SELECT TEST
                               ;  TEST THAT A WRITE TO ADDRESS 1B0000 DOES NOT WRITE TO
                               ;  B000:0, OR 1B8000 DOES NOT WRITE TO B800:0
                               ;-----------------------------------------------------------

0260  B0 FA                            MOV     AL,0FAH                          ;<><><><><><><><><><><><><>
0262  E6 80                            OUT     MFG_PORT,AL                      ;<><><>CHECKPOINT FA <><><><>
0264  B8 0008                          MOV     AX,GDT_PTR                       ; MODIFY THE DESCRIPTER TABLE
0267  8E D8                            MOV     DS,AX                            ;
                                       ;---------------------------------------
                               ;------- SET TEMP ES DESCRIPTOR 64K SEGMENT LIMIT/CPL0 DATA ACCESS
                                       ;---------------------------------------

0269  C7 06 0048 FFFF                  MOV     DS:ES_TEMP.SEG_LIMIT,MAX_SEG_LEN
026F  C6 06 004D 93                    MOV     BYTE PTR DS:(ES_TEMP.DATA_ACC_RIGHTS),CPL0_DATA_ACCESS

                               ;------- START WITH SEGMENT 1B0000

0274  C6 06 004C 1B                    MOV     BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE),1BH
0279  C7 06 004A 0000                  MOV     DS:ES_TEMP.BASE_LO_WORD,0

027F  B8 0048                          MOV     AX,ES_TEMP                       ; LOAD ES REG
0282  8E C0                            MOV     ES,AX                            ;

0284  2B FF                            SUB     DI,DI                            ; POINT TO FIRST LOCATION
0286  26: C7 05 AA55                   MOV     WORD PTR ES:[DI],0AA55H          ; WRITE A ZERO

                               ;------- DO FOR SEGMENT 1B8000

028B  C7 06 004A 8000                  MOV     DS:ES_TEMP.BASE_LO_WORD,8000H

0291  B8 0048                          MOV     AX,ES_TEMP                       ; LOAD ES REG
0294  8E C0                            MOV     ES,AX                            ;

0296  26: C7 05 AA55                   MOV     WORD PTR ES:[DI],0AA55H          ; WRITE A ZERO

                               ;-------- DO FOR SEGMENT 1A0000

029B  C6 06 004C 1A                    MOV     BYTE PTR DS:(ES_TEMP.BASE_HI_BYTE),1AH
02A0  C7 06 004A 0000                  MOV     DS:ES_TEMP.BASE_LO_WORD,0

02A6  B8 0048                          MOV     AX,ES_TEMP                       ; LOAD ES REG
02A9  8E C0                            MOV     ES,AX                            ;
02AB  26: C7 05 AA55                   MOV     WORD PTR ES:[DI],0AA55H          ; WRITE A ZERO

                               ;------- B/W VIDEO CARD

02B0  BB 0020                          MOV     BX,C_BWCRT_PTR            ;
02B3  8E DB                            MOV     DS,BX                     ; SET DS TO BW CRT BUFFER
02B5  8B 05                            MOV     AX,DS:[DI]                ; GET THE WORD FROM B/W VIDEO

                               ;-------- COMPATIBLE COLOR

02B7  BB 0028                          MOV     BX,C_CCRT_PTR             ; SET DS TO COMPATIBLE COLOR RAM
02BA  8E DB                            MOV     DS,BX                     ;
02BC  8B 1D                            MOV     BX,DS:[DI]                ; GET THE WORD FROM COLOR RAM

                               ;-------- AGC COLOR

02BE  B9 0030                          MOV     CX,E_CCRT_PTR             ; AGC COLOR CRT PTR LOW 64K
02C1  8E D9                            MOV     DS,CX                     ;
02C3  8B 0D                            MOV     CX,DS:[DI]                ;

                               ;-------- TEST FOR ERROR

02C5  50                               PUSH    AX                        ; SAVE RESULTS
02C6  B0 35                            MOV     AL,35H                    ; <><><><><><><><><><><><>
02C8  E6 80                            OUT     MFG_PORT,AL               ; <><>CHECKPOINT 35<><><>
02CA  58                               POP     AX                        ;
02CB  3D AA55                          CMP     AX,0AA55H                 ;
02CE  74 1A                            JZ      ERROR_EXIT                ;
02D0  81 FB AA55                       CMP     BX,0AA55H                 ;
02D4  74 14                            JZ      ERROR_EXIT                ;
02D6  81 F9 AA55                       CMP     CX,0AA55H                 ;
02DA  74 0E                            JZ      ERROR_EXIT                ;
02DC  B0 34                            MOV     AL,34H                    ; RESTORE CHECKPOINT
02DE  E6 80                            OUT     MFG_PORT,AL               ; <><>CHECKPOINT 34 <><><><>

                               ;------- SHUTDOWN

02E0                           NORMAL_EXIT:

02E0  B0 8F                            MOV     AL,SHUT_DOWN                     ; ADDR FOR SHUTDOWN BYTE
02E2  E6 70                            OUT     CMOS_PORT,AL                     ;
02E4  B0 06                            MOV     AL,6                             ; SET GOOD ENDING
02E6  EB 00                            JMP     SHORT $+2                        ; IO DELAY
02E8  E6 71                            OUT     CMOS_PORT+1,AL                   ;
02EA                           ERROR_EXIT:
02EA  E9 0000 E                        JMP     PROC_SHUTDOWN             ;

02ED                           POST7      ENDP
02ED                           CODE       ENDS
                                          END
```

```
                                 TITLE   SYSINIT1 - 09/26/83 INITIALIZE FOR PROTECTED MODE (POST TEST)
                                 ;
                                 ;       SYSINIT1 Include files
                                 ;
                                 ;               INCLUDE SYSDATA.INC
                                 ;               INCLUDE ACCESS.INC
                                 ;               INCLUDE SYSDATA.MAC
                                 ;               INCLUDE IAPX286.MAC
                                 ;               INCLUDE POSTEQU.SRC
                                         .LIST
                                                 PUBLIC  SYSINIT1

                                                 EXTRN   SIDT_BLD:NEAR
                                                 EXTRN   GDT_BLD:NEAR
                               C INCLUDE SEGMENT.SRC
 0000                          C CODE SEGMENT BYTE PUBLIC
                               C
                                                 ASSUME  CS:CODE
                                                 ASSUME  SS:NOTHING
                                                 ASSUME  DS:NOTHING
                                                 ASSUME  ES:NOTHING
                                 PAGE
 0000                            SYSINIT1        PROC    NEAR
                                 ;--------------------------------------------------------------
                                 ;       THIS BUILDS THE DESCRIPTOR TABLES REQUIRED FOR PROTECTED MODE
                                 ;               PROCESSOR MUST BE IN REAL MODE
                                 ;--------------------------------------------------------------

 0000  FA                                CLI                             ; NO INTERRUPTS ALLOWED
 0001  55                                PUSH    BP                      ; SAVE BP
 0002  B0 81                             MOV     AL,81H                  ;<><><><><><><><><><><><>
 0004  E6 80                             OUT     MFG_PORT,AL             ;<><><>CHECKPOINT 81 <><><>
 0006  E8 0000 E                         CALL    SIDT_BLD
 0009  8B EF                             MOV     BP,DI                   ; SAVE THE POINTER TO JUST PAST THE IDT
                                                                         ; SINCE WE HAVE NO SDA, USE THE SIX BYTES
                                                                         ;   HERE TO LOAD THE IDTR.  WE WILL SIDT
                                                                         ;   WHEN WE GET TO SDA INITIALIZATION.
 000B  B8 0800                           MOV     AX,SYS_IDT_LEN          ; SEGMENT LIMIT = LENGTH OF IDT
 000E  AB                                STOSW                           ; STORE THAT AS IDT LIMIT
 000F  B8 D0A0                           MOV     AX,SYS_IDT_LOC          ; IDT ADDRESS
 0012  AB                                STOSW                           ;   AND ACCESS RIGHTS BYTE (UNDEFINED)
 0013  B8 0000                           MOV     AX,0                    ;
 0016  AB                                STOSW                           ;
                                         SEGOV   ES                      ; LOAD THE IDT
 0017  26                      +         DB      026H
                                         LIDT    [BP]                    ;   REGISTER FROM THIS AREA
 0018  0F                      +         DB      00FH
 0019                          + ??0001  LABEL   BYTE
 0019  8B 5E 00                +         MOV     BX,WORD PTR [BP]
 001C                          + ??0002  LABEL   BYTE
 0019                          +         ORG     OFFSET CS:??0001
 0019  01                      +         DB      001H
 001C                          +         ORG     OFFSET CS:??0002
 001C  8B FD                             MOV     DI,BP                   ; ES:DI NOW --> END OF IDT AGAIN

                                 ;
                                 ;       BUILD THE GDT.
                                 ;

 001E  BF D8A0                           MOV     DI,GDT_LOC              ;
 0021  E8 0000 E                         CALL    GDT_BLD
 0024  8B EF                             MOV     BP,DI                   ; SAVE THE ES:DI POINTER
 0026  B8 0088                           MOV     AX,GDT_LEN              ; AX = LENGTH OF THE GDT
 0029  AB                                STOSW                           ; PUT THAT IN THE LIMIT FIELD
 002A  B8 D8A0                           MOV     AX,GDT_LOC              ; AX = LOW WORD OF GDT ADDRESS
 002D  AB                                STOSW                           ; PUT THAT IN BASE FIELD - LOW
 002E  B8 0000                           MOV     AX,0                    ; AX = HIGH BYTE OF ADDRESS, AND
 0031  AB                                STOSW                           ;   ACCESS RIGHTS BYTE IS UNDEFINED
                                         SEGOV   ES                      ; LOAD THE GDTR
 0032  26                      +         DB      026H
                                         LGDT    [BP]                    ;   FROM THIS AREA
 0033  0F                      +         DB      00FH
 0034                          + ??0004  LABEL   BYTE
 0034  8B 56 00                +         MOV     DX,WORD PTR [BP]
 0037                          + ??0005  LABEL   BYTE
 0034                          +         ORG     OFFSET CS:??0004
 0034  01                      +         DB      001H
 0037                          +         ORG     OFFSET CS:??0005
 0037  8B FD                             MOV     DI,BP                   ; RESTORE THE ES:DI POINTER
 0039  AB                                STOSW
 003A  AB                                STOSW
 003B  8B FD                             MOV     DI,BP

                                 PAGE
                                 ;
                                 ;       SWITCH TO VIRTUAL MODE
                                 ;

 003D  5D                                POP     BP                              ; RESTORE BP
 003E  B8 0001                           MOV     AX,VIRTUAL_ENABLE               ; MACHINE STATUS WORD NEEDED TO
                                         LMSW    AX                              ;   SWITCH TO VIRTUAL MODE
 0041  0F                      +         DB      00FH
 0042                          + ??0006  LABEL   BYTE
 0042  8B F0                   +         MOV     SI,AX
 0044                          + ??0007  LABEL   BYTE
 0042                          +         ORG     OFFSET CS:??0006
 0042  01                      +         DB      001H
 0044                          +         ORG     OFFSET CS:??0007
                                         JUMPFAR DONE,SYS_ROM_CS                 ; MUST PURGE PRE-FETCH QUEUE
 0044  EA                      +         DB      0EAH                    ; Jump far direct
 0045  0049 R                  +         DW      (OFFSET DONE)           ;   to this offset
 0047  0040                    +         DW      SYS_ROM_CS              ;     in this segment
 0049                            DONE:
 0049  B0 85                             MOV     AL,85H                          ;<><><><><><><><><><><><>
 004B  E6 80                             OUT     MFG_PORT,AL                     ;<><><>CHECKPOINT 82 <><><>
 004D  C3                                RET     0                               ; SYSTEM INITIALIZATION
 004E                            SYSINIT1        ENDP
 004E                            CODE            ENDS                            ;
                                                 END
```

SECTION 5

```
                              TITLE   GDT_BLD - 09/26/83 BUILD THE GDT
                              .LIST
                      C  INCLUDE SEGMENT.SRC
0000                  C  CODE SEGMENT BYTE PUBLIC
                      C
                                    ASSUME  CS:CODE
                                    ASSUME  SS:NOTHING
                                    ASSUME  DS:CODE
                                    ASSUME  ES:NOTHING

                                    PUBLIC  GDT_BLD

                         PAGE
                         ;
                         ;
                         ;
                         ;        THE FOLLOWING DATA DEFINES THE PRE-INITIALIZED GDT.
                         ;        THESE MUST BE INITIALIZED IN THE ORDER IN WHICH THEY APPEAR
                         ;        IN THE GDT_DEF STRUCTURE DEFINITION AS IT IS IN SYSDATA.INC.
                         ;
                         ;
                         ;
0000                     GDT_DATA_START  LABEL   WORD
                         ;
                         ;
                         ;                FIRST ENTRY UNUSABLE
                         ;
                                  DESCR_DEF       SEG, 0, 0, 0, 0
0000  0000            +           DW      0               ; Segment limit
0002  0000            +           DW      0               ; Segment base address - low word
0004  00              +           DB      0               ; Segment base address - high byte
0005  00              +           DB      0               ; Access rights byte
0006  0000            +           DW      0               ; Reserved
                         ;
                         ;
                         ;                THE GDT ITSELF
                         ;
                                  DESCR_DEF       SEG, GDT_LEN, GDT_LOC, 0, CPLO_DATA_ACCESS
0008  0088            +           DW      GDT_LEN         ; Segment limit
000A  D8A0            +           DW      GDT_LOC         ; Segment base address - low word
000C  00              +           DB      0               ; Segment base address - high byte
000D  93              +           DB      CPLO_DATA_ACCESS          ; Access rights byte
000E  0000            +           DW      0               ; Reserved
                         PAGE
                         ;
                         ;                THE SYSTEM IDT DESCRIPTOR
                         ;
                                  DESCR_DEF       SEG, SYS_IDT_LEN, SYS_IDT_LOC, 0, CPLO_DATA_ACCESS
0010  0800            +           DW      SYS_IDT_LEN             ; Segment limit
0012  D0A0            +           DW      SYS_IDT_LOC             ; Segment base address - low word
0014  00              +           DB      0               ; Segment base address - high byte
0015  93              +           DB      CPLO_DATA_ACCESS          ; Access rights byte
0016  0000            +           DW      0               ; Reserved
                         ;
                         ;                THE SYSTEM DATA AREA DESCRIPTOR
                         ;
                                  DESCR_DEF       SEG, SDA_LEN, SDA_LOC, 0, CPLO_DATA_ACCESS
0018  0300            +           DW      SDA_LEN         ; Segment limit
001A  0400            +           DW      SDA_LOC         ; Segment base address - low word
001C  00              +           DB      0               ; Segment base address - high byte
001D  93              +           DB      CPLO_DATA_ACCESS          ; Access rights byte
001E  0000            +           DW      0               ; Reserved
                         PAGE
                         ;
                         ;                COMPATIBLE MONOCHROME CRT
                         ;
                                  DESCR_DEF       SEG, MCRT_SIZE, MCRT@_LO, MCRT@_HI, CPLO_DATA_ACCESS
0020  1000            +           DW      MCRT_SIZE               ; Segment limit
0022  0000            +           DW      MCRT@_LO                ; Segment base address - low word
0024  0B              +           DB      MCRT@_HI                ; Segment base address - high byte
0025  93              +           DB      CPLO_DATA_ACCESS          ; Access rights byte
0026  0000            +           DW      0               ; Reserved
                         ;
                         ;                COMPATIBLE COLOR CRT
                         ;
                                  DESCR_DEF       SEG, CCRT_SIZE, CCRT@_LO, CCRT@_HI, CPLO_DATA_ACCESS
0028  4000            +           DW      CCRT_SIZE               ; Segment limit
002A  8000            +           DW      CCRT@_LO                ; Segment base address - low word
002C  0B              +           DB      CCRT@_HI                ; Segment base address - high byte
002D  93              +           DB      CPLO_DATA_ACCESS          ; Access rights byte
002E  0000            +           DW      0               ; Reserved
                         ;
                         ;                ENHANCED COLOR CRT - ONE ENTRY FOR EACH 64K
                         ;
                                  DESCR_DEF       SEG, ECCRT_SIZE, ECCRT@_LO_LO, ECCRT@_LO_HI, CPLO_DATA_ACCESS
0030  FFFF            +           DW      ECCRT_SIZE              ; Segment limit
0032  0000            +           DW      ECCRT@_LO_LO            ; Segment base address - low word
0034  0A              +           DB      ECCRT@_LO_HI            ; Segment base address - high byte
0035  93              +           DB      CPLO_DATA_ACCESS          ; Access rights byte
0036  0000            +           DW      0               ; Reserved
                         ;
                         ;                SECOND PART OF CRT
                         ;
                                  DESCR_DEF       SEG, ECCRT_SIZE, ECCRT@_HI_LO, ECCRT@_HI_HI, CPLO_DATA_ACCESS
0038  FFFF            +           DW      ECCRT_SIZE              ; Segment limit
003A  0000            +           DW      ECCRT@_HI_LO            ; Segment base address - low word
003C  0C              +           DB      ECCRT@_HI_HI            ; Segment base address - high byte
003D  93              +           DB      CPLO_DATA_ACCESS          ; Access rights byte
003E  0000            +           DW      0               ; Reserved
                         PAGE
                         ;
                         ;                CODE SEGMENT FOR POST CODE, SYSTEM IDT
                         ;
                                  DESCR_DEF       SEG, MAX_SEG_LEN, CSEG@_LO, CSEG@_HI, CPLO_CODE_ACCESS
0040  FFFF            +           DW      MAX_SEG_LEN             ; Segment limit
0042  0000            +           DW      CSEG@_LO                ; Segment base address - low word
0044  0F              +           DB      CSEG@_HI                ; Segment base address - high byte
0045  9B              +           DB      CPLO_CODE_ACCESS          ; Access rights byte
0046  0000            +           DW      0               ; Reserved
                         ;
                         ;
                         ;                TEMPORARY DESCRIPTORS FOR ES, CS, SS, AND DS
                         ;
                                  DESCR_DEF       SEG, MAX_SEG_LEN, NSEG@_LO, NSEG@_HI, CPLO_DATA_ACCESS
0048  FFFF            +           DW      MAX_SEG_LEN             ; Segment limit
004A  0000            +           DW      NSEG@_LO                ; Segment base address - low word
004C  00              +           DB      NSEG@_HI                ; Segment base address - high byte
004D  93              +           DB      CPLO_DATA_ACCESS          ; Access rights byte
004E  0000            +           DW      0               ; Reserved
                         ;
                                  DESCR_DEF       SEG, MAX_SEG_LEN, NSEG@_LO, NSEG@_HI, CPLO_DATA_ACCESS
0050  FFFF            +           DW      MAX_SEG_LEN             ; Segment limit
0052  0000            +           DW      NSEG@_LO                ; Segment base address - low word
0054  00              +           DB      NSEG@_HI                ; Segment base address - high byte
0055  93              +           DB      CPLO_DATA_ACCESS          ; Access rights byte
0056  0000            +           DW      0               ; Reserved
                         ;
```

SECTION 5

```
                                                 DESCR_DEF        SEG, MAX_SEG_LEN, NSEG@_LO, NSEG@_HI, CPLO_DATA_ACCESS
0058  FFFF                              +        DW       MAX_SEG_LEN           ; Segment limit
005A  0000                              +        DW       NSEG@_LO              ; Segment base address - low word
005C  00                                +        DB       NSEG@_HI              ; Segment base address - high byte
005D  93                                +        DB       CPLO_DATA_ACCESS             ; Access rights byte
005E  0000                              +        DW       0               ; Reserved
                                         ;
                                                 DESCR_DEF        SEG, MAX_SEG_LEN, NSEG@_LO, NSEG@_HI, CPLO_DATA_ACCESS
0060  FFFF                              +        DW       MAX_SEG_LEN           ; Segment limit
0062  0000                              +        DW       NSEG@_LO              ; Segment base address - low word
0064  00                                +        DB       NSEG@_HI              ; Segment base address - high byte
0065  93                                +        DB       CPLO_DATA_ACCESS             ; Access rights byte
0066  0000                              +        DW       0               ; Reserved

                                         ; POST_TR
0068                                     TR_LOC:
                                                 DESCR_DEF        SEG, 800H, 0C000H, 0, FREE_TSS
0068  0800                              +        DW       800H            ; Segment limit
006A  C000                              +        DW       0C000H          ; Segment base address - low word
006C  00                                +        DB       0               ; Segment base address - high byte
006D  81                                +        DB       FREE_TSS              ; Access rights byte
006E  0000                              +        DW       0               ; Reserved
                                         ; POST_TSS_PTR
                                                 DESCR_DEF        SEG, 800H, TR_LOC, 0, CPLO_DATA_ACCESS
0070  0800                              +        DW       800H            ; Segment limit
0072  0068 R                            +        DW       TR_LOC          ; Segment base address - low word
0074  00                                +        DB       0               ; Segment base address - high byte
0075  93                                +        DB       CPLO_DATA_ACCESS             ; Access rights byte
0076  0000                              +        DW       0               ; Reserved
0078                                     LDT_LOC:
                                         ; POST_LDTR
                                                 DESCR_DEF        SEG, GDT_LEN, 0D000H, 0, LDT_DESC
0078  0088                              +        DW       GDT_LEN         ; Segment limit
007A  D000                              +        DW       0D000H          ; Segment base address - low word
007C  00                                +        DB       0               ; Segment base address - high byte
007D  E2                                +        DB       LDT_DESC              ; Access rights byte
007E  0000                              +        DW       0               ; Reserved
                                         ; POST_LDT_PTR
                                                 DESCR_DEF        SEG, GDT_LEN, LDT_LOC, 0, CPLO_DATA_ACCESS
0080  0088                              +        DW       GDT_LEN         ; Segment limit
0082  0078 R                            +        DW       LDT_LOC         ; Segment base address - low word
0084  00                                +        DB       0               ; Segment base address - high byte
0085  93                                +        DB       CPLO_DATA_ACCESS             ; Access rights byte
0086  0000                              +        DW       0               ; Reserved
                                         PAGE
                                         ;
0088                                     GDT_DATA_END     LABEL    WORD
                                         ;
                                         ;       END OF PRE-ALLOCATED GDT
                                         ;
0088                                     GDT_BLD          PROC     NEAR

0088  BE 0000 R                                  MOV      SI,OFFSET GDT_DATA_START    ; DS:SI --> GDT
008B  B9 0044                                    MOV      CX,(GDT_DATA_END-GDT_DATA_START)/2  ; NUMBER OF WORDS TO COPY
008E  F3/ A5                                     REP      MOVSW                 ; COPY GDT INTO RAM

0090  C3                                         RET      0


0091                                     GDT_BLD          ENDP

0091                                     CODE             ENDS          ;    MPC              ENDS
                                                          END
```

```
                                TITLE   SIDT_BLD 6/10/83 PROTECTED MODE INTERRUPT TABLE
                            ;
                            ;   SIDT_BLD Include files
                            ;
                            ;           INCLUDE SYSDATA.INC
                            ;           INCLUDE ACCESS.INC
                            ;
                            ;           INCLUDE SYSDATA.MAC
                            ;           INCLUDE IAPX286.MAC
                            ;
                                .LIST
                          C  INCLUDE SEGMENT.SRC
0000                      C  CODE SEGMENT BYTE PUBLIC
                          C
                                        ASSUME  CS:CODE
                                        ASSUME  SS:NOTHING
                                        ASSUME  DS:NOTHING
                                        ASSUME  ES:NOTHING

                                        PUBLIC  SIDT_BLD
0000                        SIDT_BLD    PROC    NEAR

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

0000  BE 0066 R                 MOV     SI,OFFSET SYS_IDT_OFFSETS  ; MAKE DS:SI POINT TO
0003  8C C8                     MOV     AX,CS                 ;       INTERRUPT ENTRY POINTS
0005  8E D8                     MOV     DS,AX                 ;
0007  BF D0A0                   MOV     DI,SYS_IDT_LOC        ; POINT TO SYS_IDT_LOC
000A  2B C0                     SUB     AX,AX                 ;
000C  8E C0                     MOV     ES,AX                 ;    WHERE THE IDT WILL BE.

000E  BB 0040                   MOV     BX,SYS_ROM_CS         ; CS IS THE SAME FOR ALL INTERRUPTS
0011  B6 87                     MOV     DH,TRAP_GATE          ; ACCESS RIGHTS BYTE FOR THE GATE
0013  B2 00                     MOV     DL,0                  ; THE WORD COUNT FIELD IS UNUSED

0015  B9 0020                   MOV     CX,32                 ; THERE ARE 32 RESERVED INTERRUPTS

0018                        LOW_IDT:                          ; THIS LOOP BUILDS 32 DESCRIPTORS IN THE
                                                              ;   IDT FOR THE RESERVED INTERRUPTS
0018  A5                        MOVSW                         ; GET A ROUTINE ENTRY POINT
                                                              ;   AND PUT IT IN THE OFFSET FIELD
0019  8B C3                     MOV     AX,BX                 ; GET THE SYSTEM CODE SEGMENT SELECTOR
001B  AB                        STOSW                         ;   AND PUT IT IN THE SELECTOR FIELD
001C  8B C2                     MOV     AX,DX                 ; GET THE INTERRUPT GATE BYTE
001E  AB                        STOSW                         ;   AND PUT IT IN THE ACCESS RIGHTS FIELD
001F  B8 0000                   MOV     AX,0                  ; ZERO OUT
0022  AB                        STOSW                         ;   THE RESERVED POSTITIONS
0023  E2 F3                     LOOP    LOW_IDT               ;     AND REPEAT AS DIRECTED

0025  B9 00E0                   MOV     CX,256-32             ; 256 TOTAL - 32 DONE = WHATEVER IS LEFT
0028  BD 00A6 R                 MOV     BP,OFFSET FREE_INTS   ; THERE IS A COPY OF AN UNINITIALIZED
                                                              ;   INTERRUPT DESCRIPTOR AT FREE_INTS
                            PAGE

002B                        HIGH_IDT:

002B  8B F5                     MOV     SI,BP                 ; DS:SI --> FREE DESCRIPTOR
                                                              ; (ES:DI LEFT OFF AT INT 32)
002D  A5                        MOVSW                         ; MOVE THE OFFSET OF THE IRET INSTRUCTION
002E  A5                        MOVSW                         ; MOVE THE CS SELECTOR
002F  A5                        MOVSW                         ; MOVE THE ACCESS RIGHTS BYTE
0030  AB                        STOSW                         ; ZERO OUT THE RESERVED WORD
0031  E2 F8                     LOOP    HIGH_IDT              ; FILL THE REMAINDER OF THE TABLE
                            ;
                            ;
                            ;   INITIALIZE THE ENTRY POINTS FOR POST TEST
                            ;
0033  26: C7 06 D1A0 0000 E     MOV   WORD PTR ES:(SYS_IDT_LOC+(032*DESC_LEN).ENTRY_POINT),OFFSET SYS_32

003A  26: C7 06 D1A8 0000 E     MOV   WORD PTR ES:(SYS_IDT_LOC+(033*DESC_LEN).ENTRY_POINT),OFFSET SYS_33

0041  26: C7 06 D1B0 0000 E     MOV   WORD PTR ES:(SYS_IDT_LOC+(034*DESC_LEN).ENTRY_POINT),OFFSET SYS_34

0048  26: C7 06 D1B8 0000 E     MOV   WORD PTR ES:(SYS_IDT_LOC+(035*DESC_LEN).ENTRY_POINT),OFFSET SYS_35

004F  26: C7 06 D1C0 0000 E     MOV   WORD PTR ES:(SYS_IDT_LOC+(036*DESC_LEN).ENTRY_POINT),OFFSET SYS_36

0056  26: C7 06 D1C8 0000 E     MOV   WORD PTR ES:(SYS_IDT_LOC+(037*DESC_LEN).ENTRY_POINT),OFFSET SYS_37

005D  26: C7 06 D1D0 0000 E     MOV   WORD PTR ES:(SYS_IDT_LOC+(038*DESC_LEN).ENTRY_POINT),OFFSET SYS_38


0064  C3                        RET     0

                            PAGE

0065                        IRET_ADDR       LABEL   WORD      ; FOR UNINITIALIZED INTERRUPTS

0065  CF                                    IRET

                            ;
                            ;   EXTRNS FOR THE FIRST 32 SYSTEM INTERRUPTS
                            ;
                                EXTRN   EXC_00:NEAR
                                EXTRN   EXC_01:NEAR
                                EXTRN   EXC_02:NEAR
                                EXTRN   EXC_03:NEAR
                                EXTRN   EXC_04:NEAR
                                EXTRN   EXC_05:NEAR
                                EXTRN   EXC_06:NEAR
                                EXTRN   EXC_07:NEAR
                                EXTRN   EXC_08:NEAR
                                EXTRN   EXC_09:NEAR
                                EXTRN   EXC_10:NEAR
                                EXTRN   EXC_11:NEAR
                                EXTRN   EXC_12:NEAR
                                EXTRN   EXC_13:NEAR
                                EXTRN   EXC_14:NEAR
                                EXTRN   EXC_15:NEAR
                                EXTRN   EXC_16:NEAR
                                EXTRN   EXC_17:NEAR
                                EXTRN   EXC_18:NEAR
                                EXTRN   EXC_19:NEAR
                                EXTRN   EXC_20:NEAR
                                EXTRN   EXC_21:NEAR
                                EXTRN   EXC_22:NEAR
                                EXTRN   EXC_23:NEAR
                                EXTRN   EXC_24:NEAR
                                EXTRN   EXC_25:NEAR
                                EXTRN   EXC_26:NEAR
                                EXTRN   EXC_27:NEAR
```

SECTION 5

```
                                      EXTRN   EXC_28:NEAR
                                      EXTRN   EXC_29:NEAR
                                      EXTRN   EXC_30:NEAR
                                      EXTRN   EXC_31:NEAR

                                      EXTRN   SYS_32:NEAR
                                      EXTRN   SYS_33:NEAR
                                      EXTRN   SYS_34:NEAR
                                      EXTRN   SYS_35:NEAR
                                      EXTRN   SYS_36:NEAR
                                      EXTRN   SYS_37:NEAR
                                      EXTRN   SYS_38:NEAR
                              PAGE
                              ;
                              ;       Entry points for the first 32 system interrupts
                              ;
0066                          SYS_IDT_OFFSETS         LABEL   WORD
                                                                      ; INTERRUPTS AS DEFINED

0066  0000 E                          DW      OFFSET EXC_00           ; EXCPT 00 - DIVIDE ERROR
0068  0000 E                          DW      OFFSET EXC_01           ; EXCPT 01 - SINGLE STEP
006A  0000 E                          DW      OFFSET EXC_02           ; EXCPT 02 - NMI, SYS REQ FOR D1
006C  0000 E                          DW      OFFSET EXC_03           ; EXCPT 03 - BREAKPOINT
006E  0000 E                          DW      OFFSET EXC_04           ; EXCPT 04 - INTO DETECT
0070  0000 E                          DW      OFFSET EXC_05           ; EXCPT 05 - BOUND
0072  0000 E                          DW      OFFSET EXC_06           ; EXCPT 06 - INVALID OPCODE
0074  0000 E                          DW      OFFSET EXC_07           ; EXCPT 07 - PROCESSOR EXT NOT AVAIL
0076  0000 E                          DW      OFFSET EXC_08           ; EXCPT 08 - DOUBLE EXCEPTION
0078  0000 E                          DW      OFFSET EXC_09           ; EXCPT 09 - PROCESSOR EXT SEGMENT ERR
007A  0000 E                          DW      OFFSET EXC_10           ; EXCPT 10 - STK PL BAD IN GATE TRANSFER
007C  0000 E                          DW      OFFSET EXC_11           ; EXCPT 11 - SEGMENT NOT PRESENT
007E  0000 E                          DW      OFFSET EXC_12           ; EXCPT 12 - STACK SEGMENT NOT PRESENT
0080  0000 E                          DW      OFFSET EXC_13           ; EXCPT 13 - GENERAL PROTECTION
0082  0000 E                          DW      OFFSET EXC_14
0084  0000 E                          DW      OFFSET EXC_15
0086  0000 E                          DW      OFFSET EXC_16           ; EXCPT 16 - PROCESSOR EXTENSION ERROR
0088  0000 E                          DW      OFFSET EXC_17
008A  0000 E                          DW      OFFSET EXC_18
008C  0000 E                          DW      OFFSET EXC_19
008E  0000 E                          DW      OFFSET EXC_20
0090  0000 E                          DW      OFFSET EXC_21
0092  0000 E                          DW      OFFSET EXC_22
0094  0000 E                          DW      OFFSET EXC_23
0096  0000 E                          DW      OFFSET EXC_24
0098  0000 E                          DW      OFFSET EXC_25
009A  0000 E                          DW      OFFSET EXC_26
009C  0000 E                          DW      OFFSET EXC_27
009E  0000 E                          DW      OFFSET EXC_28
00A0  0000 E                          DW      OFFSET EXC_29
00A2  0000 E                          DW      OFFSET EXC_30
00A4  0000 E                          DW      OFFSET EXC_31
                              PAGE
                              ;
                              ;       FORMAT INTERRUPT DESCRIPTORS (GATES) 32 - 255
                              ;
00A6  0065 R                  FREE_INTS       DW      OFFSET IRET_ADDR        ; DESTINATION OFFSET
00A8  0040                                    DW      SYS_ROM_CS              ; DESTINATION SEGMENT
00AA  00 86                                   DB      0,INT_GATE              ; UNUSED BYTE, ACCESS RIGHTS BYTE
00AC                          SIDT_BLD        ENDP
00AC                          CODE            ENDS

                                      END
```

```
                              TITLE DSKETTE DATE 01-12-84 DISKETTE BIOS
                              .LIST
                            C INCLUDE SEGMENT.SRC
0000                        C CODE SEGMENT BYTE PUBLIC
                            C

                              PUBLIC  DISK_INT_1
                              PUBLIC  SEEK
                              PUBLIC  DSKETTE_SETUP
                              EXTRN   DDS:NEAR

                              ;-- INT 13 ------------------------------------------------------------
                              ; DISKETTE I/O
                              ;       THIS INTERFACE PROVIDES ACCESS TO THE 5 1/4" DISKETTE DRIVES
                              ;       320/360K DISKETTE DRIVES AND 1.2M DISKETTE DRIVES SUPPORTED
                              ; INPUT
                              ;       (AH)=0  RESET DISKETTE SYSTEM
                              ;               HARD RESET TO NEC, PREPARE COMMAND, RECAL REQD ON ALL DRIVES
                              ;       (AH)=1  READ THE STATUS OF THE SYSTEM INTO (AH)
                              ;               DISKETTE_STATUS FROM LAST OP'N IS USED
                              ;       REGISTERS FOR READ/WRITE/VERIFY/FORMAT
                              ;       (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)
                              ;       (DH) - HEAD NUMBER (0-1 ALLOWED, NOT VALUE CHECKED)
                              ;       (CH) - TRACK NUMBER (NOT VALUE CHECKED)
                              ;               MEDIA       DRIVE           TRACK NUMBER
                              ;              320/360     320/360             0-39
                              ;              320/360      1.2M               0-39
                              ;               1.2M        1.2M               0-79
                              ;       (CL) - SECTOR NUMBER (NOT VALUE CHECKED, NOT USED FOR FORMAT)
                              ;               MEDIA       DRIVE           SECTOR NUMBER
                              ;              320/360     320/360             1-8/9
                              ;              320/360      1.2M               1-8/9
                              ;               1.2M        1.2M               1-15
                              ;       (AL) - NUMBER OF SECTORS (NOT VALUE CHECKED)
                              ;               MEDIA       DRIVE           MAX NUMBER OF SECTORS
                              ;              320/360     320/360               8/9
                              ;              320/360      1.2M                 8/9
                              ;               1.2M        1.2M                 15
                              ;
                              ;       (ES:BX) - ADDRESS OF BUFFER ( REQUIRED FOR VERIFY)
                              ;
                              ;       (AH)=2  READ THE DESIRED SECTORS INTO MEMORY
                              ;       (AH)=3  WRITE THE DESIRED SECTORS FROM MEMORY
                              ;       (AH)=4  VERIFY THE DESIRED SECTORS
                              ;       (AH)=5  FORMAT THE DESIRED TRACK
                              ;               FOR THE FORMAT OPERATION, THE BUFFER POINTER (ES,BX) MUST
                              ;               POINT TO THE COLLECTION OF DESIRED ADDRESS FIELDS FOR THE
                              ;               TRACK.  EACH FIELD IS COMPOSED OF 4 BYTES, (C,H,R,N), WHERE
                              ;               C = TRACK NUMBER, H=HEAD NUMBER, R = SECTOR NUMBER, N= NUMBER
                              ;               OF BYTES PER SECTOR (00=128, 01=256, 02=512, 03=1024,)
                              ;               THERE MUST BE ONE ENTRY FOR EVERY SECTOR ON THE TRACK.
                              ;               THIS INFORMATION IS USED TO FIND THE REQUESTED SECTOR DURING
                              ;               READ/WRITE ACCESS.
                              ;               PRIOR TO FORMATTING A DISKETTE, FUNCTION CALL 17 OF THIS
                              ;               ROUTINE MUST BE INVOKED TO SET THE DISKETTE TYPE THAT IS TO
                              ;               BE FORMATTED.
                              ;               IN ORDER TO FORMAT 320/360K MEDIA IN EITHER A 320/360K OR
                              ;               1.2M DISKETTE DRIVE THE GAP LENGTH FOR FORMAT PARAMETER
                              ;               OF DISK_BASE MUST BE CHANGE TO 050H.  ALSO THE EOT
                              ;               PARAMETER (LAST SECTOR ON TRACK) MUST BE SET TO THE
                              ;               DESIRED NUMBER OF SECTORS/TRACK - 8 FOR 320K, 9 FOR 360K.
                              ;               DISK_BASE IS POINTED TO BY DISK POINTER LOCATED AT
                              ;               ABSOLUTE ADDRESS 0:78.
                              ;               WHEN 320/360K FORMAT OPERATIONS ARE COMPLETE, THE PARAMETERS
                              ;               SHOULD BE RESTORED TO THEIR RESPECTIVE INITIAL VALUES.
                              ;       (AH)=15 READ DASD TYPE
                              ;       REGISTERS
                              ;       (AH) - ON RETURN IF CARRY FLAG NOT SET, OTHERWISE ERROR
                              ;               00 - DRIVE NOT PRESENT
                              ;               01 - DISKETTE, NO CHANGE LINE AVAILABLE
                              ;               02 - DISKETTE, CHANGE LINE AVAILABLE
                              ;               03 - FIXED DISK
                              ;       (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)
                              ;
                              ;       (AH)=16 DISK CHANGE LINE STATUS
                              ;       REGISTERS
                              ;       (AH)=00 - DISK CHANGE LINE NOT ACTIVE
                              ;            06 - DISK CHANGE LINE ACTIVE & CARRY BIT ON
                              ;       (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)
                              ;
                              ;       (AH)=17 SET DASD TYPE FOR FORMAT
                              ;       REGISTERS
                              ;       (AL) -    00 - NOT USED
                              ;                 01 - DISKETTE 320/360K IN 320/360K DRIVE
                              ;                 02 - DISKETTE 320/360K IN 1.2M DRIVE
                              ;                 03 - DISKETTE 1.2M IN 1.2M DRIVE
                              ;       (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED;
                              ;               DO NOT USE WHEN DISKETTE ATTACH CARD USED)
                              ;       DISK CHANGE STATUS IS ONLY CHECKED WHEN A 1.2M BYTE DISKETTE
                              ;       DRIVE IS SPECIFIED.  IF THE DISK CHANGE LINE IS FOUND TO BE
                              ;       ACTIVE THE FOLLOWING ACTIONS TAKE PLACE:
                              ;               ATTEMPT TO RESET DISK CHANGE LINE TO INACTIVE STATE.
                              ;               IF ATTEMPT SUCCEEDS SET DASD TYPE FOR FORMAT AND RETURN DISK
                              ;               CHANGE ERROR CODE
                              ;               IF ATTEMPT FAILS RETURN TIMEOUT ERROR CODE AND SET DASD TYPE
                              ;               TO A PREDETERMINED STATE INDICATING MEDIA TYPE UNKNOWN.
                              ;       IF THE DISK CHANGE LINE IN INACTIVE PERFORM SET DASD TYPE FOR FORMAT.
                              ;
                              ; DATA VARIABLE -- DISK_POINTER
                              ;       DOUBLE WORD POINTER TO THE CURRENT SET OF DISKETTE PARAMETERS
                              ; OUTPUT
                              ;       AH = STATUS OF OPERATION
                              ;               STATUS BITS ARE DEFINED IN THE EQUATES FOR DISKETTE_STATUS
                              ;               VARIABLE IN THE DATA SEGMENT OF THIS MODULE
                              ;       CY = 0  SUCCESSFUL OPERATION (AH=0 ON RETURN, EXCEPT FOR READ DASD
                              ;               TYPE AH=(15)).
                              ;       CY = 1  FAILED OPERATION (AH HAS ERROR REASON)
                              ;       FOR READ/WRITE/VERIFY
                              ;               DS,BX,DX,CH,CL PRESERVED
                              ;       NOTE: IF AN ERROR IS REPORTED BY THE DISKETTE CODE, THE APPROPRIATE
                              ;               ACTION IS TO RESET THE DISKETTE, THEN RETRY THE OPERATION.
                              ;               ON READ ACCESSES, NO MOTOR START DELAY IS TAKEN, SO THAT
                              ;               THREE RETRIES ARE REQUIRED ON READS TO ENSURE THAT THE
                              ;               PROBLEM IS NOT DUE TO MOTOR START-UP.
                              ;-----------------------------------------------------------
                              ;
                              ;       DISKETTE STATE MACHINE - ABSOLUTE ADDRESS 40:90 & 91
                              ;        (DRIVE 0 - 90, DRIVE 1 - 91)
                              ;       BITS
                              ;
                              ;   -----------------------------------------------------------------
                              ;   |       |       |       |       |       |       |       |       |
                              ;   |   7   |   6   |   5   |   4   |   3   |   2   |   1   |   0   |
                              ;   |       |       |       |       |       |       |       |       |
                              ;   -----------------------------------------------------------------
                              ;       |       |       |       |       |       |       |       |
                              ;       |       |       |       |       |       -----------------
```

SECTION 5

```
                                    ;       |       |       |       |    |              |
                                    ;       |       |       |       |  RESERVED         --PRESENT STATE
                                    ;       |       |       |       |
                                    ;       |       |       |       |      000: 360K IN 360K DRIVE UNESTABLISHED
                                    ;       |       |       |       |      001: 360K IN 1.2M DRIVE UNESTABLISHED
                                    ;       |       |       |       |      002: 1.2M IN 1.2M DRIVE UNESTABLISHED
                                    ;       |       |       |       |      003: 360K IN 360K DRIVE ESTABLISHED
                                    ;       |       |       |       |      004: 360K IN 1.2M DRIVE ESTABLISHED
                                    ;       |       |       |       |      005: 1.2M IN 1.2M DRIVE ESTABLISHED
                                    ;       |       |       |       |
                                    ;       |       |       |       ------> MEDIA/DRIVE ESTABLISHED
                                    ;       |       |       |
                                    ;       |       |       --------------> DOUBLE STEPPING REQUIRED (360K IN 1.2M
                                    ;       |       |                         DRIVE)
                                    ;       |       |
                                    ;       ------------------------------> DATA TRANSFER RATE FOR THIS DRIVE:
                                    ;
                                    ;                                             00: 500 KBS
                                    ;                                             01: 300 KBS
                                    ;                                             10: 250 KBS
                                    ;                                             11: RESERVED
                                    ;
                                    ;       STATE OPERATION STARTED - ABSOLUTE ADDRESS 40:92 & 93
                                    ;        (DRIVE 0 - 92, DRIVE 1 - 93)
                                    ;       PRESENT CYLINDER NUMBER - ABSOLUTE ADDRESS 40:94 & 95
                                    ;        (DRIVE 0 - 94, DRIVE 1 - 95)
                                    ;-------------------------------------------------------------
                                            ASSUME  CS:CODE,DS:DATA,ES:DATA
                                    PUBLIC  DISKETTE_IO_1
 0000                               DISKETTE_IO_1   PROC    FAR     ;>>> ENTRY POINT FOR ORG 0EC59H
 0000  FB                                   STI                     ; INTERRUPTS BACK ON
 0001  53                                   PUSH    BX              ; SAVE ADDRESS
 0002  51                                   PUSH    CX
 0003  1E                                   PUSH    DS              ; SAVE SEGMENT REGISTER VALUE
 0004  56                                   PUSH    SI              ; SAVE ALL REGISTERS DURING OPERATION
 0005  57                                   PUSH    DI
 0006  55                                   PUSH    BP
 0007  52                                   PUSH    DX
 0008  8B EC                                MOV     BP,SP           ; SET UP POINTER TO HEAD PARM
 000A  BE ---- R                            MOV     SI,DATA
 000D  8E DE                                MOV     DS,SI           ; SET DATA REGION
 000F  80 FC 01                             CMP     AH,1            ; CHECK FOR RESET AND STATUS OPERATIONS
 0012  76 0F                                JBE     R4              ; BYPASS DRIVE CHECK IF YES
                                    ;
 0014  80 FA 01                             CMP     DL,1            ; CHECK DRIVE NUMBER FOR VALIDITY
 0017  76 0A                                JBE     R4              ; IF VALID CONTINUE
                                    ;
 0019  C6 06 0041 R 01              R5:     MOV     DISKETTE_STATUS,BAD_CMD ; INVALID DRIVE ADDRESS, TERMINATE
 001E  BE 0000                              MOV     SI,0            ; INSURE THAT RETURN STATUS GETS SETUP
 0021  EB 49                                JMP     SHORT OK        ; GO TERMINATE COMMAND
                                    ;
 0023  50                           R4:     PUSH    AX              ; SAVE ORIGINAL OPERATION FOR RETRY LATER ON
 0024  E8 010C R                            CALL    J1              ; CALL THE REST TO ENSURE DS RESTORED
 0027  5E                                   POP     SI              ; RESTORE ORIGINAL OPERATION FOR RETRY
 0028  8B D6                                MOV     DX,SI           ; GET ORIGINAL OPERATION FOR TESTING
 002A  80 FE 01                             CMP     DH,1            ; SEE IF IT IS A RESET OR STATUS OPERATION
 002D  76 3D                                JBE     OK              ; BYPASS STATE UPDATE
                                    ;
 002F  F6 06 008F R 01                      TEST    HF_CNTRL,DUAL   ; GO DETERMINE TYPE OF CONTROLLER CARD
 0034  74 36                                JZ      OK              ; DISKETTE ATTACH CARD
                                    ;
 0036  80 FE 15                             CMP     DH,15H          ; READ DISK CHANGE STATUS OR DISK TYPE COMMAND
 0039  73 31                                JAE     OK              ; IF YES, BYPASS STATE PROCESSING
                                    ;
 003B  8B 56 00                             MOV     DX,[BP]         ; RESTORE DRIVE PARAMETER
 003E  32 FF                                XOR     BH,BH           ; SETUP ADDRESS TO MEDIA STATE FOR THIS DRIVE
 0040  8A DA                                MOV     BL,DL           ; *
 0042  8A 26 0041 R                         MOV     AH,DISKETTE_STATUS      ; GET STATUS OF OPERATION
 0046  0A E4                                OR      AH,AH           ; SEE IF ANY ERRORS
 0048  75 4C                                JNZ     RETRY           ; JUMP TO CHECK FOR MEDIA CHANGE
                                    ;
 004A  8A A7 0090 R                         MOV     AH,DSK_STATE[BX] ; GET MEDIA STATE OF DRIVE
 004E  F6 C4 10                             TEST    AH,DETERMINED   ; SEE IF MEDIA STATE SET ALREADY
 0051  75 14                                JNZ     OK2             ; IF SET, DONT CHANGE STATE
                                    ;
 0053  8A CC                                MOV     CL,AH           ; GET PRESENT STATE
 0055  80 E1 07                             AND     CL,STATE_MSK    ; ISOLATE STATE NUMBER
 0058  80 C1 03                             ADD     CL,3            ; ELEVATE STATE TO SET ALREADY
 005B  80 E4 F8                             AND     AH,REV_STATE    ; CLEAR OUT STATE NUMBER
 005E  0A E1                                OR      AH,CL           ; SET NEW STATE NUMBER
 0060  80 CC 10                             OR      AH,DETERMINED   ; MAKE MEDIA STATE SET
 0063  88 A7 0090 R                         MOV     DSK_STATE[BX],AH ; SAVE IN DRIVE STATE INDICATOR
 0067  C6 87 0092 R 00              OK2:    MOV     DSK_STATE[BX+2],0 ; CLEAR ORIGINAL STATE OPERATION STARTED IN
 006C  BB 0004                      OK:     MOV     BX,4            ; GET THE MOTOR WAIT PARAMETER
 006F  8B D6                                MOV     DX,SI           ; GET ORIGINAL OP AGAIN
 0071  50                                   PUSH    AX              ; SAVE RETURN VALUE
 0072  E8 0382 R                            CALL    GET_PARM
 0075  88 26 0040 R                         MOV     MOTOR_COUNT,AH  ; SET THE TIMER COUNT FOR THE MOTOR
 0079  58                                   POP     AX              ; RESTORE RETURN VALUE
 007A  80 FE 15                             CMP     DH,015H         ; SEE IF READ DASD OPERATION
 007D  75 05                                JNE     R20             ; IF NOT BYPASS
                                    ;
 007F  86 E0                                XCHG    AH,AL           ; PUT RESULT IN AH
 0081  F8                                   CLC                     ; SET SUCCESSFUL OPERATION
 0082  EB 08                                JMP     SHORT R19       ; GO LEAVE
                                    ;
 0084  8A 26 0041 R                 R20:    MOV     AH,DISKETTE_STATUS      ; GET STATUS OF OPERATION
 0088  80 FC 01                             CMP     AH,1            ; SET THE CARRY FLAG TO INDICATE
 008B  F5                                   CMC                     ;  SUCCESS OR FAILURE
 008C  5A                           R19:    POP     DX              ; RESTORE ALL REGISTERS
 008D  5D                                   POP     BP
 008E  5F                                   POP     DI
 008F  5E                                   POP     SI
 0090   F                                   POP     DS
 0091  59                                   POP     CX
 0092  5B                                   POP     BX              ; RECOVER ADDRESS
 0093  CA 0002                              RET     2               ; THROW AWAY SAVED FLAGS
                                    ;
 0096  80 3E 0041 R 06              RETRY:  CMP     DISKETTE_STATUS,MEDIA_CHANGE ; CHECK FOR DISK CHANGE ERROR
 009B  74 54                                JE      OK1             ; TRUE ERROR DONT RETRY
                                    ;
 009D  8A A7 0090 R                         MOV     AH,DSK_STATE[BX] ; GET MEDIA STATE OF DRIVE
 00A1  80 E4 07                             AND     AH,STATE_MSK    ; ISOLATE STATE
 00A4  80 FC 03                             CMP     AH,3            ; SEE IF IN STATE 3
 00A7  73 BE                                JAE     OK2             ; IF ESTABLISHED STATE THEN TRUE ERROR
                                    ;
                                    ;------ HANDLE STATES 0, 1 & 2
                                    ;
 00A9  FE C4                                INC     AH              ; TRY NEXT STATE
 00AB  80 FC 03                             CMP     AH,3            ; SEE IF OVERFLOW IN NON-ESTABLISHED STATES
 00AE  75 02                                JNE     R2              ; SKIP RESET TO BEGINNING IF YES
                                    ;
 00B0  B4 00                                MOV     AH,0            ; NEXT STATE TO TRY AFTER OVERFLOW
 00B2  8A AF 0092 R                 R2:     MOV     CH,DSK_STATE[BX+2] ; GET START RETRY STATE
 00B6  80 E5 07                             AND     CH,STATE_MSK    ; ISOLATE STATE BITS
 00B9  3A EC                                CMP     CH,AH           ; ALL STATES TRIED
```

```
00BB  74 47                   JE     OK3             ; IF YES, THEN TRUE ERROR
                        ;
                        ;------ SETUP STATE INDICATOR FOR RETRY ATTEMPT
                        ;
00BD  8A AF 0090 R            MOV    CH,DSK_STATE[BX] ; GET STATE INDICATOR
00C1  D0 C5                   ROL    CH,1            ; MOVE TRANSFER RATE TO LOW ORDER BITS
00C3  D0 C5                   ROL    CH,1            ; *
00C5  80 E5 03                AND    CH,TRAN_MSK     ; ISOLATE TRANSFER RATE BITS
00C8  FE CD                   DEC    CH              ; CONVERT TO NEXT RATE
00CA  80 FD FF                CMP    CH,0FFH         ; SEE IF OVERFLOW OCCURRED
00CD  75 02                   JNE    R3              ; JUMP IF NO OVERFLOW
                        ;
00CF  B5 02                   MOV    CH,XRATE        ; SET TO NEXT RATE
00D1  D0 CD             R3:   ROR    CH,1            ; PUT TRANSFER BITS BACK WHERE THEY BELONG
00D3  D0 CD                   ROR    CH,1            ; *
00D5  80 FC 01                CMP    AH,1            ; SEE IF THIS STATE REQUIRES DOUBLE STEP
00D8  75 03                   JNE    R9              ; IF NOT, BYPASS SETTING DOUBLE STEP
                        ;
00DA  80 CD 20                OR     CH,DOUBLE_STEP  ; TURN ON DOUBLE STEP REQUIRED
00DD  0A E5             R9:   OR     AH,CH           ; COMBINE WITH STATE TO MAKE NEW INDICATOR
00DF  88 A7 0090 R            MOV    DSK_STATE[BX],AH ; SAVE AS NEW INDICATOR
                        ;
                        ;------ SETUP FOR ACTUAL RETRY OPERATION
                        ;
00E3  8B 56 00                MOV    DX,[BP]         ; RESTORE PARAMETERS FROM STACK
00E6  8B 4E 0A                MOV    CX,[BP+10]      ; *
00E9  8B 5E 0C                MOV    BX,[BP+12]      ; *
00EC  8B C6                   MOV    AX,SI           ; *
00EE  E9 0023 R               JMP    R4              ; GO RETRY OPERATION
                        ;
00F1  8B 56 00          OK1:  MOV    DX,[BP]         ; RESTORE DRIVE PARMETER
00F4  E8 0604 R               CALL   READ_DSKCHNG    ; GO READ DISK CHANGE LINE STATUS
00F7  75 03                   JNZ    OK4             ; IF ACTIVE, NO DISKETTE IN DRIVE, TIMEOUT
                        ;
00F9  E9 0067 R               JMP    OK2             ; IF NOT ACTIVE, DISKETTE IN DRIVE, DISK CHANGE
                        ;
00FC  C6 06 0041 R 80   OK4:  MOV    DISKETTE_STATUS,TIME_OUT ; INDICATE TIMEOUT IF DRIVE EMPTY
0101  E9 0067 R               JMP    OK2
                        ;
0104  C6 87 0090 R 80   OK3:  MOV    DSK_STATE[BX],POA_START ; ERROR PUT STATE AT POWER ON ASSUMPTION
0109  E9 0067 R               JMP    OK2
                        ;
010C                    DISKETTE_IO_1  ENDP
                        ;
                        ;------ DETERMINE NEW MEDIA TYPE, NEED TO RESET DISK CHANGE LINE HERE
                        ;
010C                    J1    PROC   NEAR
010C  80 FC 01                CMP    AH,1            ; TEST FOR RESET AND STATUS OPERATION
010F  76 76                   JBE    J1E             ; BYPASS STATE CHECK AND UPDATE
                        ;
0111  F6 06 008F R 01         TEST   HF_CNTRL,DUAL   ; GO DETERMINE TYPE OF CONTROLLER CARD
0116  74 11                   JZ     J1A             ; DISKETTE ATTACH CARD
                        ;
0118  80 FC 15                CMP    AH,15H          ; TEST FOR DISK CHANGE STATUS OR DISK TYPE
011B  73 6A                   JAE    J1E             ; BYPASS STATE CHECK AND UPDATE
                        ;
011D  50                      PUSH   AX              ; SAVE ORIGINAL PARAMETERS
011E  53                      PUSH   BX              ; SAVE PARAMETERS
011F  51                      PUSH   CX              ; *
0120  52                      PUSH   DX              ; *
0121  E8 0604 R               CALL   READ_DSKCHNG    ; GO READ DISK CHANGE LINE STATE
0124  74 0C                   JZ     J1I             ; BYPASS HANDLING DISK CHANGE LINE
                        ;
0126  E9 05E2 R               JMP    J1F             ; HANDLE DISK CHANGE LINE ACTIVE
                        ;
0129  50                J1A:  PUSH   AX              ; SAVE ORIGINAL PARAMETERS
012A  53                      PUSH   BX              ; SAVE PARAMETERS
012B  51                      PUSH   CX              ; *
012C  52                      PUSH   DX              ; *
012D  E8 0604 R               CALL   READ_DSKCHNG    ; SELECT DRIVE FOR DISKETTE ATTACH CARD
0130  EB 51                   JMP    SHORT J1H       ; IGNORE DISK CHANGE STATUS
                        ;
0132  8A 87 0090 R      J1I:  MOV    AL,DSK_STATE[BX] ; GET MEDIA STATE INFORMATION FOR DRIVE
0136  0A C0                   OR     AL,AL           ; CHECK FOR NO STATE INFORMATION AT ALL
0138  75 06                   JNZ    J1D             ; IF INFORMATION DONT DEFAULT
                        ;
013A  B0 80                   MOV    AL,POA_START    ; GET DEFAULT TO STATE 0
013C  88 87 0090 R            MOV    DSK_STATE[BX],AL ; SET UP DEFAULT TO STATE 1
                        ;
0140  3C 61             J1D:  CMP    AL,POA_DUAL     ; SEE IF DOUBLE STEP RATE
0142  75 1E                   JNE    J1G             ; BYPASS TRACK CHECK
                        ;
0144  8B 4E 0A                MOV    CX,[BP+10]      ; GET ORIGINAL TRACK PARAMETER
0147  80 FD 28                CMP    CH,40           ; SEE IF TRACK IS PAST END OF DISKETTE(320)
014A  72 16                   JB     J1G             ; GO TRY OPERATION AT THIS STATE IF NOT
                        ;
014C  C6 87 0090 R 02         MOV    DSK_STATE[BX],02H ; SET NEXT STATE TO TRY IN ALGORITHM
0151  B0 02                   MOV    AL,02H          ; PUT NEW STATE IN WORKING REGISTER
0153  8A B7 0092 R            MOV    DH,DSK_STATE[BX+2] ; GET OPERATION START STATE
0157  0A F6                   OR     DH,DH           ; CHECK FOR OPERATION START
0159  75 13                   JNZ    J1C             ; IF STARTED PREVIOUSLY, BYPASS SETTING IT UP
                        ;
015B  C6 87 0092 R 61         MOV    DSK_STATE[BX+2],POA_DUAL ; SETUP STARTING STATE
0160  EB 0C                   JMP    SHORT J1C       ; BYPASS NEXT STEP ALREADY DONE
                        ;
0162  8A 97 0092 R      J1G:  MOV    DL,DSK_STATE[BX+2] ; GET START MEDIA STATE
0166  0A D2                   OR     DL,DL           ; SEE IF THIS IS ORIGINAL OPERATION OR A RETRY
0168  75 04                   JNZ    J1C             ; IF RETRY IGNORE
                        ;
016A  88 87 0092 R            MOV    DSK_STATE[BX+2],AL ; SAVE AS STARTING DATA RATE
016E  8A 0E 008B R      J1C:  MOV    CL,LASTRATE     ; GET LAST DATA RATE SELECTED
0172  3A C1                   CMP    AL,CL           ; COMPARE TO LAST OPERATION
0174  74 0D                   JE     J1H             ; IF SAME DONT SELECT NEW TRANSFER RATE
                        ;
0176  A2 008B R               MOV    LASTRATE,AL     ; SAVE NEW TRANSFER RATE FOR NEXT CHECK
0179  D0 C0                   ROL    AL,1            ; MOVE TRANSFER RATE DATA TO LOW BITS
017B  D0 C0                   ROL    AL,1            ; *
017D  24 03                   AND    AL,TRAN_MSK     ; CLEAR ALL BITS BUT DATA TRANSFER RATE BITS
017F  BA 03F7                 MOV    DX,03F7H        ; ADDRESS FLOPPY CONTROL REGISTER
0182  EE                      OUT    DX,AL           ; SET DATA TRANSFER RATE
0183  5A                J1H:  POP    DX              ; RESTORE PARAMETERS
0184  59                      POP    CX              ; *
0185  5B                      POP    BX              ; *
0186  58                      POP    AX              ; *
0187  8A F0             J1E:  MOV    DH,AL           ; SAVE # SECTORS IN DH
0189  80 26 003F R 7F         AND    MOTOR_STATUS,07FH      ; INDICATE A READ OPERATION
018E  0A E4                   OR     AH,AH           ; AH=0
0190  74 38                   JZ     DISK_RESET
0192  FE CC                   DEC    AH              ; AH=1
0194  74 76                   JZ     DISK_STATUS
0196  C6 06 0041 R 00         MOV    DISKETTE_STATUS,0      ; RESET THE STATUS INDICATOR
019B  FE CC                   DEC    AH              ; AH=2
019D  74 6E                   JZ     DISK_READ
019F  FE CC                   DEC    AH              ; AH=3
01A1  75 03                   JNZ    J2              ; TEST_DISK_VERF
01A3  E9 0240 R               JMP    DISK_WRITE
```

SECTION 5

```
01A6                        J2:                              ; TEST_DISK_VERF
01A6  FE CC                         DEC     AH               ; AH=4
01A8  74 6C                         JZ      DISK_VERF
01AA  FE CC                         DEC     AH               ; AH=5
01AC  74 6C                         JZ      DISK_FORMAT
01AE  80 EC 10                      SUB     AH,10H           ; AH=15H
01B1  75 03                         JNZ     J3               ; BYPASS DISK TYPE OPERATION
                            ;
01B3  E9 0698 R                     JMP     DISK_TYPE        ; GO PERFORM DISK TYPE OPERATION

01B6  FE CC                 J3:     DEC     AH               ; AH = 16H
01B8  75 03                         JNZ     J4               ; BYPASS DISK CHANGE STATUS
                            ;
01BA  E9 0646 R                     JMP     DISK_CHANGE      ; GO CHECK DISK CHANGE LINE STATUS
                            ;
01BD  FE CC                 J4:     DEC     AH               ; AH = 17H
01BF  75 03                         JNZ     J5               ; BAD COMMAND
                            ;
01C1  E9 070D R                     JMP     FORMAT_SET       ; GO SET MEDIA/DRIVE TYPE FOR FORMAT
                            ;
01C4  C6 06 0041 R 01       J5:     MOV     DISKETTE_STATUS,BAD_CMD ; ERROR CODE, NO SECTORS TRANSFERRED
01C9  C3                            RET                      ; UNDEFINED OPERATION
01CA                        J1      ENDP

                            ;------ RESET THE DISKETTE SYSTEM

01CA                        DISK_RESET      PROC    NEAR
01CA  BA 03F2                       MOV     DX,03F2H         ; ADAPTER CONTROL PORT
01CD  FA                            CLI                      ; NO INTERRUPTS
01CE  A0 003F R                     MOV     AL,MOTOR_STATUS  ; WHICH MOTOR IS ON
01D1  24 3F                         AND     AL,03FH          ; STRIP OFF UNWANTED BITS
01D3  B1 04                         MOV     CL,4             ; SHIFT COUNT
01D5  D2 C0                         ROL     AL,CL            ; MOVE MOTOR VALUE TO HIGH NIBBLE, DRIVE SELECT
                                                             ;  TO LOW NIBBLE
01D7  0C 08                         OR      AL,8             ; TURN ON INTERRUPT ENABLE
01D9  EE                            OUT     DX,AL            ; RESET THE ADAPTER
01DA  C6 06 003E R 00               MOV     SEEK_STATUS,0    ; SET RECAL REQUIRED ON ALL DRIVES
01DF  C6 06 0041 R 00               MOV     DISKETTE_STATUS,0 ; SET OK STATUS FOR DISKETTE
01E4  EB 00                         JMP     $+2              ; I/O WAIT STATE
01E6  0C 04                         OR      AL,4             ; TURN OFF RESET
01E8  EE                            OUT     DX,AL            ; TURN OFF THE RESET
01E9  FB                            STI                      ; REENABLE THE INTERRUPTS
01EA  E8 051A R                     CALL    CHK_STAT_2       ; DO SENSE INTERRUPT STATUS FOLLOWING RESET
01ED  A0 0042 R                     MOV     AL,NEC_STATUS    ; IGNORE ERROR RETURN AND DO OWN TEST
01F0  3C C0                         CMP     AL,0C0H          ; TEST FOR DRIVE READY TRANSITION
01F2  74 06                         JZ      J7               ; EVERYTHING OK
01F4  80 0E 0041 R 20               OR      DISKETTE_STATUS,BAD_NEC ; SET ERROR CODE
01F9  C3                            RET

                            ;------ SEND SPECIFY COMMAND TO NEC

01FA                        J7:                              ; DRIVE_READY
01FA  B4 03                         MOV     AH,03H           ; SPECIFY COMMAND
01FC  E8 03E2 R                     CALL    NEC_OUTPUT       ; OUTPUT THE COMMAND
01FF  BB 0001                       MOV     BX,1             ; FIRST BYTE PARM IN BLOCK
0202  E8 0382 R                     CALL    GET_PARM         ;  TO THE NEC CONTROLLER
0205  BB 0003                       MOV     BX,3             ; SECOND BYTE PARM IN BLOCK
0208  E8 0382 R                     CALL    GET_PARM         ;  TO THE NEC CONTROLLER
                                                             ; RESET_RE
020B  C3                            RET                      ; RETURN TO CALLER
020C                        DISK_RESET      ENDP

                            ;------ DISKETTE STATUS ROUTINE

020C                        DISK_STATUS     PROC    NEAR
020C  C3                            RET
020D                        DISK_STATUS     ENDP

                            ;------ DISKETTE READ

020D                        DISK_READ       PROC    NEAR
020D  B0 46                         MOV     AL,046H          ; READ COMMAND FOR DMA
020F                        J9:                              ; DISK_READ_CONT
020F  E8 04CA R                     CALL    DMA_SETUP        ; SET UP THE DMA
0212  B4 E6                         MOV     AH,0E6H          ; SET UP READ COMMAND FOR NEC CONTROLLER
0214  EB 36                         JMP     SHORT RW_OPN     ; GO DO THE OPERATION
0216                        DISK_READ       ENDP

                            ;------ DISKETTE VERIFY

0216                        DISK_VERF       PROC    NEAR
0216  B0 42                         MOV     AL,042H                  ;  VERIFY COMMAND FOR DMA
0218  EB F5                         JMP     J9                       ;  DO AS IF DISK READ
021A                        DISK_VERF       ENDP

                            ;------ DISKETTE FORMAT

021A                        DISK_FORMAT     PROC    NEAR
021A  80 0E 003F R 80               OR      MOTOR_STATUS,WRITE_OP    ;  INDICATE WRITE OPERATION
021F  B0 4A                         MOV     AL,04AH                  ;  WILL WRITE TO THE DISKETTE
0221  E8 04CA R                     CALL    DMA_SETUP                ; SET UP THE DMA
0224  B4 4D                         MOV     AH,04DH                  ; ESTABLISH THE FORMAT COMMAND
0226  EB 24                         JMP     SHORT RW_OPN             ; DO THE OPERATION
0228                        J10:                                     ; CONTINUATION OF RW_OPN FOR FMT
0228  BB 0007                       MOV     BX,7                     ; GET THE
022B  E8 0382 R                     CALL    GET_PARM                 ;  BYTES/SECTOR VALUE TO NEC
022E  BB 0009                       MOV     BX,9                     ; GET THE
0231  E8 0382 R                     CALL    GET_PARM                 ;  SECTORS/TRACK VALUE TO NEC
0234  BB 000F                       MOV     BX,15                    ; GET THE
0237  E8 0382 R                     CALL    GET_PARM                 ;  GAP LENGTH VALUE TO NEC
023A  BB 0011                       MOV     BX,17                    ; GET THE FILLER BYTE
023D  E9 032A R                     JMP     J16                      ;  TO THE CONTROLLER
0240                        DISK_FORMAT     ENDP

                            ;------ DISKETTE WRITE ROUTINE

0240                        DISK_WRITE      PROC    NEAR
0240  80 0E 003F R 80               OR      MOTOR_STATUS,WRITE_OP    ; INDICATE WRITE OPERATION
0245  B0 4A                         MOV     AL,04AH                  ; DMA WRITE COMMAND
0247  E8 04CA R                     CALL    DMA_SETUP
024A  B4 C5                         MOV     AH,0C5H                  ; NEC COMMAND TO WRITE TO DISKETTE
024C                        DISK_WRITE      ENDP
                            ;----- ALLOW WRITE ROUTINE TO FALL INTO RW_OPN
                            ;----------------------------------------
                            ; RW_OPN
                            ;       THIS ROUTINE PERFORMS THE READ/WRITE/VERIFY OPERATION
                            ;----------------------------------------
024C                        RW_OPN  PROC    NEAR
024C  73 08                         JNC     J11                      ; TEST FOR DMA ERROR
024E  C6 06 0041 R 09               MOV     DISKETTE_STATUS,DMA_BOUNDARY    ; SET ERROR
0253  B0 00                         MOV     AL,0                     ; NO SECTORS TRANSFERRED
0255  C3                            RET                              ; RETURN TO MAIN ROUTINE
0256                        J11:                                     ; DO_RW_OPN
0256  50                            PUSH    AX                       ; SAVE THE COMMAND

                            ;------ TURN ON THE MOTOR AND SELECT THE DRIVE
```

```
0257  51                    PUSH   CX                   ; SAVE THE T/S PARMS
0258  8A CA                 MOV    CL,DL                ; GET DRIVE NUMBER AS SHIFT COUNT
025A  B0 01                 MOV    AL,1                 ; MASK FOR DETERMINING MOTOR BIT
025C  D2 E0                 SAL    AL,CL                ; SHIFT THE MASK BIT
025E  FA                    CLI                         ; NO INTERRUPTS WHILE DETERMINING MOTOR STATUS
025F  84 06 003F R          TEST   AL,MOTOR_STATUS      ; IS THIS MOTOR ON
0263  74 0C                 JZ     R13                  ; IF NOT GO TEST FOR WAIT NECESSARY

0265  80 3E 0040 R EC       CMP    MOTOR_COUNT,0ECH     ; SEE IF THE MOTOR HAS BEEN ON LONG ENOUGH
026A  C6 06 0040 R FF       MOV    MOTOR_COUNT,0FFH     ; ENSURE MOTOR DOESNT TURN OFF DURING OPERATION
026F  72 42                 JB     J14                  ; IS LESS THAN EC, THEN TURN ON NOT DUE TO
                                                        ;   READING OF DISK CHANGE LINE, OTHERWISE
                                                        ;   GO TEST FOR WAIT NECESSARY
                     ;
0271  08 06 003F R   R13:   OR     MOTOR_STATUS,AL      ; TURN ON THE CURRENT MOTOR
0275  B1 04                 MOV    CL,4                 ; SHIFT COUNT TO MOVE DRIVE TO HIGH NIBBLE
0277  80 26 003F R CF       AND    MOTOR_STATUS,0CFH    ; CLEAR ENCODED DRIVE SELECT BITS(4 & 5)
027C  D2 C2                 ROL    DL,CL                ; MOVE DRIVE ENCODED BITS TO HIGH NIBBLE
027E  08 16 003F R          OR     MOTOR_STATUS,DL      ; SAVE AS SELECTED DRIVE
0282  D2 CA                 ROR    DL,CL                ; RESTORE
0284  FB                    STI                         ; INTERRUPTS BACK ON
0285  A0 003F R             MOV    AL,MOTOR_STATUS      ; GET MOTORS ON AND DRIVE SELECTED
0288  24 3F                 AND    AL,03FH              ; STRIP OFF UNWANTED BITS
028A  D2 C0                 ROL    AL,CL                ; SHIFT BITS AROUND TO DESIRED POSITIONS
028C  0C 0C                 OR     AL,0CH               ; NO RESET, ENABLE DMA/INT
028E  52                    PUSH   DX                   ; SAVE REG
028F  BA 03F2               MOV    DX,03F2H             ; CONTROL PORT ADDRESS
0292  EE                    OUT    DX,AL
0293  5A                    POP    DX                   ; RECOVER REGISTERS

                     ;------ WAIT FOR MOTOR

0294  F8                    CLC                         ; CLEAR TIMEOUT INDICATOR
0295  B8 90FD               MOV    AX,090FDH            ; LOAD WAIT CODE & TYPE
0298  CD 15                 INT    15H                  ; PERFORM OTHER FUNCTION
029A  72 17                 JC     J14                  ; BYPASS TIMING LOOP IF TIMEOUT OCCURRED
                     ;
029C  BB 0014               MOV    BX,20                ; GET THE MOTOR WAIT
029F  E8 0382 R             CALL   GET_PARM             ;  PARAMETER
02A2  0A E4                 OR     AH,AH                ; TEST FOR NO WAIT
02A4                 J12:                               ; TEST_WAIT_TIME
02A4  74 0D                 JZ     J14                  ; EXIT WITH TIME EXPIRED
02A6  2B C9                 SUB    CX,CX                ; SET UP 1/8 SECOND LOOP TIME
02A8  E2 FE          J13:   LOOP   J13                  ; WAIT FOR THE REQUIRED TIME
                     ;
02AA  B9 6D06               MOV    CX,06D06H            ; *
02AD  E2 FE          R18:   LOOP   R18                  ; *
                     ;
02AF  FE CC                 DEC    AH                   ; DECREMENT TIME VALUE
02B1  75 F1                 JNZ   J12                   ; ARE WE DONE YET

02B3                 J14:                               ; MOTOR_RUNNING
02B3  FB                    STI                         ; INTERRUPTS BACK ON FOR BYPASS WAIT
02B4  59                    POP    CX

                     ;------ DO THE SEEK OPERATION

02B5  E8 041C R             CALL   SEEK             ; MOVE TO CORRECT TRACK
02B8  58                    POP    AX               ; RECOVER COMMAND
02B9  8A FC                 MOV    BH,AH            ; SAVE COMMAND IN BH
02BB  B6 00                 MOV    DH,0             ; SET NO SECTORS READ IN CASE OF ERROR
02BD  72 72                 JC     J17              ; IF ERROR, THEN EXIT AFTER MOTOR OFF
02BF  BE 0331 R             MOV    SI,OFFSET J17    ; DUMMY RETURN ON STACK FOR NEC_OUTPUT
02C2  56                    PUSH   SI               ;  SO THAT IT WILL RETURN TO MOTOR OFF LOCATION

                     ;------ SEND OUT THE PARAMETERS TO THE CONTROLLER

02C3  E8 03E2 R             CALL   NEC_OUTPUT       ; OUTPUT THE OPERATION COMMAND
02C6  8A 66 01              MOV    AH,[BP+1]        ; GET THE CURRENT HEAD NUMBER
02C9  D0 E4                 SAL    AH,1             ; MOVE IT TO BIT 2
02CB  D0 E4                 SAL    AH,1
02CD  80 E4 04              AND    AH,4             ; ISOLATE THAT BIT
02D0  0A E2                 OR     AH,DL            ; OR IN THE DRIVE NUMBER
02D2  E8 03E2 R             CALL   NEC_OUTPUT

                     ;------ TEST FOR FORMAT COMMAND

02D5  80 FF 4D              CMP    BH,04DH          ; IS THIS A FORMAT OPERATION
02D8  75 03                 JNE    J15              ; NO. CONTINUE WITH R/W/V
02DA  E9 0228 R             JMP    J10              ; IF SO, HANDLE SPECIAL

02DD  8A E5          J15:   MOV    AH,CH            ; CYLINDER NUMBER
02DF  E8 03E2 R             CALL   NEC_OUTPUT
02E2  8A 66 01              MOV    AH,[BP+1]        ; HEAD NUMBER FROM STACK
02E5  E8 03E2 R             CALL   NEC_OUTPUT
02E8  8A E1                 MOV    AH,CL            ; SECTOR NUMBER
02EA  E8 03E2 R             CALL   NEC_OUTPUT
02ED  BB 0007               MOV    BX,7             ; BYTES/SECTOR PARM FROM BLOCK
02F0  E8 0382 R             CALL   GET_PARM         ;  TO THE NEC
02F3  BB 0009               MOV    BX,9             ; EOT PARM FROM BLOCK
02F6  E8 0382 R             CALL   GET_PARM         ;  TO THE NEC
02F9  8B 5E 00              MOV    BX,[BP]          ; RESTORE DRIVE NUMBER FROM PARMS
02FC  32 FF                 XOR    BH,BH            ; CLEAR HIGH ORDER INDEX REGISTER
02FE  8A A7 0090 R          MOV    AH,DSK_STATE[BX] ; GET DRIVE STATE VALUE
0302  F6 C4 10              TEST   AH,DETERMINED    ; SEE IF STATE ALREADY ESTABLISHED
0305  74 06                 JZ     D0               ; BYPASS STATE REDUCTION FOR GAP LENGTH
                     ;
0307  80 E4 07              AND    AH,07H           ; STRIP OFF HIGH BITS
030A  80 EC 03              SUB    AH,03H           ; REDUCE STATES
                     ;
030D  80 E4 07       D0:    AND    AH,07H           ; STRIP OFF HIGH BITS
0310  80 FC 00              CMP    AH,0             ; CHECK FOR DISKETTE ATTACH CARD OR 320 DRIVE
0313  75 04                 JNE    R16              ; IF NOT CHECK FOR NEXT STATE
                     ;
0315  B4 2A                 MOV    AH,02AH          ; LOAD 320/360 DRIVE GAP LENGTH
0317  EB 0B                 JMP    SHORT R15        ; GO OUTPUT
                     ;
0319  80 FC 01       R16:   CMP    AH,1             ; CHECK FOR 320 MEDIA IN 1.2 DRIVE
031C  75 04                 JNE    R17              ; IF NOT, THEN HANDLE 1.2 MEDIA IN 1.2 DRIVE
                     ;
031E  B4 23                 MOV    AH,023H          ; LOAD 320/360 MEDIA IN 1.2 DRIVE GAP LENGTH
0320  EB 02                 JMP    SHORT R15        ;
                     ;
0322  B4 1B          R17:   MOV    AH,01BH          ; LOAD 1.2 MEDIA IN 1.2 DRIVE GAP LENGTH
0324  E8 03E2 R      R15:   CALL   NEC_OUTPUT
0327  BB 000D               MOV    BX,13            ; DTL PARM FROM BLOCK
032A                 J16:                           ; RW_OPN_FINISH
032A  E8 0382 R             CALL   GET_PARM         ;  TO THE NEC
032D  5E                    POP    SI               ; CAN NOW DISCARD THAT DUMMY RETURN ADDRESS

                     ;------ LET THE OPERATION HAPPEN

032E  E8 053B R             CALL   WAIT_INT         ; WAIT FOR THE INTERRUPT
0331                 J17:                           ; MOTOR_OFF
0331  72 45                 JC     J21              ; LOOK FOR ERROR
```

SECTION 5

```
0333  E8 0580 R                     CALL    RESULTS           ; GET THE NEC STATUS
0336  72 3F                         JC      J20               ; LOOK FOR ERROR

                            ;------ CHECK THE RESULTS RETURNED BY THE CONTROLLER

0338  FC                            CLD                         ; SET THE CORRECT DIRECTION
0339  BE 0042 R                     MOV     SI,OFFSET NEC_STATUS    ; POINT TO STATUS FIELD
033C  AC                            LODS    NEC_STATUS              ; GET STO
033D  24 C0                         AND     AL,0C0H                 ; TEST FOR NORMAL TERMINATION
033F  74 3B                         JZ      J22                     ; OPN_OK
0341  3C 40                         CMP     AL,040H                 ; TEST FOR ABNORMAL TERMINATION
0343  75 29                         JNZ     J18                     ; NOT ABNORMAL, BAD NEC

                            ;------ ABNORMAL TERMINATION, FIND OUT WHY

0345  AC                            LODS    NEC_STATUS              ; GET ST1
0346  D0 E0                         SAL     AL,1                    ; TEST FOR EOT FOUND
0348  B4 04                         MOV     AH,RECORD_NOT_FND
034A  72 24                         JC      J19                     ; RW_FAIL
034C  D0 E0                         SAL     AL,1
034E  D0 E0                         SAL     AL,1                    ; TEST FOR CRC ERROR
0350  B4 10                         MOV     AH,BAD_CRC
0352  72 1C                         JC      J19                     ; RW_FAIL
0354  D0 E0                         SAL     AL,1                    ; TEST FOR DMA OVERRUN
0356  B4 08                         MOV     AH,BAD_DMA
0358  72 16                         JC      J19                     ; RW_FAIL
035A  D0 E0                         SAL     AL,1
035C  D0 E0                         SAL     AL,1                    ; TEST FOR RECORD NOT FOUND
035E  B4 04                         MOV     AH,RECORD_NOT_FN
0360  72 0E                         JC      J19                     ; RW_FAIL
0362  D0 E0                         SAL     AL,1
0364  B4 03                         MOV     AH,WRITE_PROTECT        ; TEST FOR WRITE_PROTECT
0366  72 08                         JC      J19                     ; RW_FAIL
0368  D0 E0                         SAL     AL,1                    ; TEST MISSING ADDRESS MARK
036A  B4 02                         MOV     AH,BAD_ADDR_MARK
036C  72 02                         JC      J19                     ; RW_FAIL

                            ;------ NEC MUST HAVE FAILED

036E                        J18:                                    ; RW-NEC-FAIL
036E  B4 20                         MOV     AH,BAD_NEC
0370                        J19:                                    ; RW-FAIL
0370  08 26 0041 R                  OR      DISKETTE_STATUS,AH
0374  E8 05CB R                     CALL    NUM_TRANS               ; HOW MANY WERE REALLY TRANSFERRED
0377                        J20:                                    ; RW_ERR
0377  C3                            RET                     ; RETURN TO CALLER

0378                        J21:                            ; RW_ERR_RES
0378  E8 0580 R                     CALL    RESULTS         ; FLUSH THE RESULTS BUFFER
037B  C3                            RET

                            ;------ OPERATION WAS SUCCESSFUL

037C                        J22:                                    ; OPN_OK
037C  E8 05CB R                     CALL    NUM_TRANS               ; HOW MANY GOT MOVED
037F  32 E4                         XOR     AH,AH                   ; NO ERRORS
0381  C3                            RET
0382                        RW_OPN  ENDP
                            ;----------------------------------------
                            ; GET_PARM
                            ;  THIS ROUTINE FETCHES THE INDEXED POINTER FROM
                            ;  THE DISK_BASE BLOCK POINTED AT BY THE DATA
                            ;  VARIABLE DISK_POINTER
                            ; A BYTE FROM THAT TABLE IS THEN MOVED INTO AH,
                            ;  THE INDEX OF THAT BYTE BEING THE PARM IN BX
                            ; ENTRY --
                            ;   BX = INDEX OF BYTE TO BE FETCHED * 2
                            ;        IF THE LOW BIT OF BX IS ON, THE BYTE IS IMMEDIATELY
                            ;        OUTPUT TO THE NEC CONTROLLER
                            ; EXIT --
                            ;   AH = THAT BYTE FROM BLOCK
                            ;------------------------------------------
0382                        GET_PARM        PROC    NEAR
0382  1E                            PUSH    DS                      ; SAVE SEGMENT
0383  56                            PUSH    SI                      ; SAVE
0384  2B C0                         SUB     AX,AX                   ; ZERO TO AX
0386  8E D8                         MOV     DS,AX
                                    ASSUME  DS:ABS0
0388  C5 36 0078 R                  LDS     SI,DISK_POINTER         ; POINT TO BLOCK
038C  D1 EB                         SHR     BX,1                    ; DIVIDE BX BY 2, AND SET FLAG FOR EXIT
038E  8A 20                         MOV     AH,[SI+BX]              ; GET THE WORD
0390  5E                            POP     SI                      ; RESTORE
0391  1F                            POP     DS                      ; RESTORE SEGMENT
0392  9C                            PUSHF                           ; SAVE RESULTS FOR EXIT
                                    ASSUME  DS:DATA
0393  83 FB 0A                      CMP     BX,10                   ; LOOK FOR MOTOR STARTUP DELAY PARM
0396  75 19                         JNE     GP0                     ; BYPASS IF NOT PARM LOOKING FOR
                            ;
0398  F6 06 003F R 80               TEST    MOTOR_STATUS,WRITE_OP   ; IS THIS A WRITE
039D  74 09                         JZ      GP1                     ; NO, ENFORCE MINIMUM READ WAIT
                            ;
039F  80 FC 08                      CMP     AH,8                    ; SEE IF AT LEAST A SECOND IS SPECIFIED
03A2  73 3A                         JAE     GP2                     ; IF YES, CONTINUE
                            ;
03A4  B4 08                         MOV     AH,8                    ; FORCE A SECOND WAIT FOR MOTOR START
03A6  EB 36                         JMP     SHORT GP2               ; CONTINUE
                            ;
03A8  80 FC 05              GP1:    CMP     AH,5                    ; SEE IF A 625 MS WAIT ON READ
03AB  73 31                         JAE     GP2                     ; IF THERE GO CONTINUE
                            ;
03AD  B4 05                         MOV     AH,5                    ; ENFORCE A 625 MS WAIT
03AF  EB 2D                         JMP     SHORT GP2               ; CONTINUE
                            ;
03B1  83 FB 09              GP0:    CMP     BX,9                    ; IS THIS HEAD SETTLE PARM
03B4  75 28                         JNE     GP2                     ; BYPASS IF NOT HEAD SETTLE
                            ;
03B6  F6 06 003F R 80               TEST    MOTOR_STATUS,WRITE_OP   ; SEE IF A WRITE OPERATION
03BB  74 21                         JZ      GP2                     ; IF NOT, DONT ENFORCE ANY VALUES
                            ;
03BD  0A E4                         OR      AH,AH                   ; CHECK FOR ANY WAIT?
03BF  75 1D                         JNZ     GP2                     ; IF THERE DONT ENFORCE
                            ;
03C1  52                            PUSH    DX                      ; SAVE REGISTER
03C2  53                            PUSH    BX                      ; SAVE REGISTER
03C3  8B 56 00                      MOV     DX,[BP]                 ; GET ORIGINAL DRIVE REQUESTED
03C6  32 FF                         XOR     BH,BH                   ; SET UP ADDRESSING TO STATE INDICATOR
03C8  8A DA                         MOV     BL,DL                   ; *
03CA  B4 0F                         MOV     AH,HD12_SETTLE          ; SPEC'ED HEAD SETTLE TIME FOR 1.2 DRIVE
03CC  8A 87 0090 R                  MOV     AL,DSK_STATE[BX]        ; GET MEDIA/DRIVE STATE
03D0  5B                            POP     BX                      ; RESTORE
03D1  5A                            POP     DX                      ; RESTORE
03D2  24 07                         AND     AL,STATE_MSK            ; ISOLATE STATE NUMBER
03D4  75 04                         JNZ     GP4                     ; BRANCH IF STATES 1 THRU 5
                            ;
03D6  B4 14                 GP3:    MOV     AH,HD320_SETTLE         ; SPEC'ED HEAD SETTLE TIME FOR 320 DRIVE
03D8  EB 04                         JMP     SHORT GP2               ; GO TO WAIT LOOP
```

```
03DA  3C 03                     ;
                                GP4:    CMP     AL,3                    ; SEE IF STATE 3(320 DRIVE/320 MEDIA)
03DC  74 F8                             JE      GP3                     ; GO REESTABLISH WAIT TIME
                                ;
03DE  9D                        GP2:    POPF                            ; RESTORE EXIT RESULTS
03DF  72 01                             JC      NEC_OUTPUT              ; IF FLAG SET, OUTPUT TO CONTROLLER
03E1  C3                                RET                             ; RETURN TO CALLER
03E2                            GET_PARM        ENDP
                                ;-----------------------------------------------
                                ; NEC_OUTPUT
                                ;       THIS ROUTINE SENDS A BYTE TO THE NEC CONTROLLER
                                ;       AFTER TESTING FOR CORRECT DIRECTION AND CONTROLLER READY
                                ;       THIS ROUTINE WILL TIME OUT IF THE BYTE IS NOT ACCEPTED
                                ;       WITHIN A REASONABLE AMOUNT OF TIME, SETTING THE DISKETTE STATUS
                                ;       ON COMPLETION
                                ; INPUT
                                ;       (AH)    BYTE TO BE OUTPUT
                                ; OUTPUT
                                ;       CY = 0  SUCCESS
                                ;       CY = 1  FAILURE -- DISKETTE STATUS UPDATED
                                ;               IF A FAILURE HAS OCCURRED, THE RETURN IS MADE ONE LEVEL
                                ;               HIGHER THAN THE CALLER OF NEC_OUTPUT
                                ;               THIS REMOVES THE REQUIREMENT OF TESTING AFTER EVERY CALL
                                ;               OF NEC_OUTPUT
                                ;       (AL) DESTROYED
                                ;-----------------------------------------------
03E2                            NEC_OUTPUT      PROC    NEAR
03E2  52                                PUSH    DX              ; SAVE REGISTERS
03E3  51                                PUSH    CX
03E4  53                                PUSH    BX
03E5  BA 03F4                           MOV     DX,03F4H        ; STATUS PORT
03E8  B3 02                             MOV     BL,2            ; HIGH ORDER COUNTER
03EA  33 C9                     R11:    XOR     CX,CX           ; COUNT FOR TIME OUT
03EC                            J23:
03EC  EC                                IN      AL,DX           ; GET STATUS
03ED  A8 40                             TEST    AL,040H         ; TEST DIRECTION BIT
03EF  74 11                             JZ      R12             ; DIRECTION OK
03F1  E2 F9                             LOOP    J23
                                ;
03F3  FE CB                             DEC     BL              ; DECREMENT COUNTER
03F5  75 F3                             JNZ     R11             ; REPEAT TIL DELAY FINISHED
                                ;
03F7                            J24:                            ; TIME_ERROR
03F7  80 0E 0041 R 80                   OR      DISKETTE_STATUS,TIME_OUT
03FC  5B                                POP     BX              ; RESTORE REGISTERS
03FD  59                                POP     CX
03FE  5A                                POP     DX              ; SET ERROR CODE AND RESTORE REGS
03FF  58                                POP     AX              ; DISCARD THE RETURN ADDRESS
0400  F9                                STC                     ; INDICATE ERROR TO CALLER
0401  C3                                RET
0402  B3 02                     R12:    MOV     BL,2            ; HIGH ORDER COUNT
0404                            J25:
0404  33 C9                             XOR     CX,CX           ; RESET THE COUNT
0406                            J26:
0406  EC                                IN      AL,DX           ; GET THE STATUS
0407  A8 80                             TEST    AL,080H         ; IS IT READY
0409  75 08                             JNZ     J27             ; YES, GO OUTPUT
                                ;
040B  E2 F9                             LOOP    J26             ; COUNT DOWN AND TRY AGAIN
                                ;
040D  FE CB                             DEC     BL              ; DECREMENT COUNTER
040F  75 F3                             JNZ     J25             ; REPEAT TIL DELAY FINISHED
                                ;
0411  EB E4                             JMP     J24             ; ERROR CONDITION
0413                            J27:                            ; OUTPUT
0413  8A C4                             MOV     AL,AH           ; GET BYTE TO OUTPUT
                                ;       MOV     DX,03F5H        ; DATA PORT
0415  B2 F5                             MOV     DL,0F5H
0417  EE                                OUT     DX,AL           ; OUTPUT THE BYTE
0418  5B                                POP     BX              ; RECOVER REGISTERS
0419  59                                POP     CX              ; RECOVER REGISTERS
041A  5A                                POP     DX
041B  C3                                RET                     ; CY = 0 FROM TEST INSTRUCTION
041C                            NEC_OUTPUT      ENDP
                                ;-----------------------------------------------
                                ; SEEK
                                ;       THIS ROUTINE WILL MOVE THE HEAD ON THE NAMED DRIVE
                                ;       TO THE NAMED TRACK.  IF THE DRIVE HAS NOT BEEN ACCESSED
                                ;       SINCE THE DRIVE RESET COMMAND WAS ISSUED, THE DRIVE WILL BE
                                ;       RECALIBRATED.
                                ; INPUT
                                ;       (DL) = DRIVE TO SEEK ON
                                ;       (CH) = TRACK TO SEEK TO
                                ; OUTPUT
                                ;       CY = 0 SUCCESS
                                ;       CY = 1 FAILURE -- DISKETTE_STATUS SET ACCORDINGLY
                                ;       (AX) DESTROYED
                                ;-----------------------------------------------
041C                            SEEK    PROC    NEAR
041C  B0 01                             MOV     AL,1            ; ESTABLISH MASK FOR RECAL TEST
041E  51                                PUSH    CX              ; SAVE INPUT VALUES
041F  8A CA                             MOV     CL,DL           ; GET DRIVE VALUE INTO CL
0421  D2 C0                             ROL     AL,CL           ; SHIFT IT BY THE DRIVE VALUE
0423  59                                POP     CX              ; RECOVER TRACK VALUE
0424  84 06 003E R                      TEST    AL,SEEK_STATUS  ; TEST FOR RECAL REQUIRED
0428  75 37                             JNZ     J28             ; NO_RECAL
                                ;
042A  08 06 003E R                      OR      SEEK_STATUS,AL  ; TURN ON THE NO RECAL BIT IN FLAG
042E  B4 07                             MOV     AH,07H          ; RECALIBRATE COMMAND
0430  E8 03E2 R                         CALL    NEC_OUTPUT
0433  8A E2                             MOV     AH,DL
0435  E8 03E2 R                         CALL    NEC_OUTPUT      ; OUTPUT THE DRIVE NUMBER
0438  E8 051A R                         CALL    CHK_STAT_2      ; GET THE INTERUPT AND SENSE INT STATUS
043B  73 14                             JNC     J28A            ; SEEK_COMPLETE
                                ;
                                ;----- ISSUE RECALIBRATE FOR 80 TRACK DISKETTES
                                ;
043D  C6 06 0041 R 00                   MOV     DISKETTE_STATUS,0 ; CLEAR OUT INVALID STATUS
0442  B4 07                             MOV     AH,07H          ; RECALIBRATE COMMAND
0444  E8 03E2 R                         CALL    NEC_OUTPUT
0447  8A E2                             MOV     AH,DL
0449  E8 03E2 R                         CALL    NEC_OUTPUT      ; OUTPUT THE DRIVE NUMBER
044C  E8 051A R                         CALL    CHK_STAT_2      ; GET THE INTERUPT AND SENSE INT STATUS
044F  72 78                             JC      RB              ; SEEK_ERROR
                                ;
0451                            J28A:
0451  F6 06 008F R 01                   TEST    HF_CNTRL,DUAL   ; GO DETERMINE TYPE OF CONTROLLER CARD
0456  74 09                             JZ      J28             ; DISKETTE ATTACH CARD
                                ;
0458  32 FF                             XOR     BH,BH           ; SET UP ADDRESSING TO STATE INDICATOR
045A  8A DA                             MOV     BL,DL           ; *
045C  C6 87 0094 R 00                   MOV     DSK_TRK[BX],0   ; SAVE NEW CYLINDER AS PRESENT POSITION
                                ;
                                ;----- DRIVE IS IN SYNCH WITH CONTROLLER, SEEK TO TRACK

0461                            J28:
```

```
0461  32 FF                        XOR    BH,BH             ; SET UP ADDRESSING TO STATE INDICATOR
0463  8A DA                        MOV    BL,DL             ; *
0465  F6 06 008F R 01              TEST   HF_CNTRL,DUAL     ; GO DETERMINE TYPE OF CONTROLLER CARD
046A  74 09                        JZ     R7                ; DISKETTE ATTACH CARD
                            ;
046C  F6 87 0090 R 20              TEST   DSK_STATE[BX],DOUBLE_STEP ; CHECK FOR DOUBLE STEP REQUIRED
0471  74 02                        JZ     R7                ; SINGLE STEP REQUIRED BYPASS DOUBLE
                            ;
0473  D0 E5                        SHL    CH,1              ; DOUBLE NUMBER OF STEP TO TAKE
0475                        R7:
0475  3A AF 0094 R                 CMP    CH,DSK_TRK[BX]    ; SEEK IF ALREADY AT THE DESIRED TRACK
0479  74 3E                        JE     J32               ; IF YES, DONT NEED TO SEEK
                            ;
047B  88 AF 0094 R                 MOV    DSK_TRK[BX],CH    ; SAVE NEW CYLINDER AS PRESENT POSITION
047F  B4 0F                        MOV    AH,0FH            ; SEEK COMMAND TO NEC
0481  E8 03E2 R                    CALL   NEC_OUTPUT
0484  8A E2                        MOV    AH,DL             ; DRIVE NUMBER
0486  E8 03E2 R                    CALL   NEC_OUTPUT
0489  8A E5                        MOV    AH,CH             ; GET CYLINDER NUMBER
048B  E8 03E2 R                    CALL   NEC_OUTPUT
048E  E8 051A R                    CALL   CHK_STAT_2        ; GET ENDING INTERRUPT AND SENSE STATUS
0491  F6 06 008F R 01              TEST   HF_CNTRL,DUAL     ; GO DETERMINE TYPE OF CONTROLLER CARD
0496  74 09                        JZ     RA                ; DISKETTE ATTACH CARD
                            ;
0498  F6 87 0090 R 20              TEST   DSK_STATE[BX],DOUBLE_STEP ; CHECK FOR DOUBLE STEP REQUIRED
049D  74 02                        JZ     RA                ; SINGLE STEP REQUIRED BYPASS DOUBLE
                            ;
049F  D0 ED                        SHR    CH,1              ; SET BACK TO LOGICAL SECTOR
04A1                        RA:

                            ;----- WAIT FOR HEAD SETTLE

04A1  9C                           PUSHF                    ; SAVE STATUS FLAGS
04A2  BB 0012                      MOV    BX,18             ; GET HEAD SETTLE PARAMETER
04A5  E8 0382 R                    CALL   GET_PARM          ; *
04A8  51                           PUSH   CX                ; SAVE REGISTER
04A9                        J29:                            ; HEAD_SETTLE
04A9  B9 0320                      MOV    CX,800            ; 1 MS LOOP
04AC  0A E4                        OR     AH,AH             ; TEST FOR TIME EXPIRED
04AE  74 06                        JZ     J31
04B0  E2 FE                 J30:   LOOP   J30               ; DELAY FOR 1 MS
04B2  FE CC                        DEC    AH                ; DECREMENT THE COUNT
04B4  EB F3                        JMP    J29               ; DO IT SOME MORE
04B6                        J31:
04B6  59                           POP    CX                ; RECOVER STATE
04B7  9D                           POPF
04B8  C3                           RET                      ; RETURN TO CALLER
                            ;
04B9                        J32:                            ; SEEK_ERROR
04B9  F6 06 008F R 01              TEST   HF_CNTRL,DUAL     ; GO DETERMINE TYPE OF CONTROLLER CARD
04BE  74 09                        JZ     RB                ; DISKETTE ATTACH CARD
                            ;
04C0  F6 87 0090 R 20              TEST   DSK_STATE[BX],DOUBLE_STEP ; CHECK FOR DOUBLE STEP REQUIRED
04C5  74 02                        JZ     RB                ; SINGLE STEP REQUIRED BYPASS DOUBLE
                            ;
04C7  D0 ED                        SHR    CH,1              ; SET BACK TO LOGICAL SECTOR
04C9                        RB:
04C9  C3                           RET                      ; RETURN TO CALLER
04CA                        SEEK   ENDP
                            ;--------------------------------------------------
                            ; DMA_SETUP
                            ;       THIS ROUTINE SETS UP THE DMA FOR READ/WRITE/VERIFY
                            ;       OPERATIONS.
                            ; INPUT
                            ;       (AL) = MODE BYTE FOR THE DMA
                            ;       (ES:BX) - ADDRESS TO READ/WRITE THE DATA
                            ; OUTPUT
                            ;       (AX) DESTROYED
                            ;--------------------------------------------------
04CA                        DMA_SETUP      PROC   NEAR
04CA  51                           PUSH   CX                ; SAVE THE REGISTER
04CB  FA                           CLI                      ; DISABLE INTERRUPTS DURING DMA SET-UP
04CC  E6 0C                        OUT    DMA+12,AL         ; SET THE FIRST/LAST F/F
04CE  EB 00                        JMP    $+2               ; WAIT FOR IO
04D0  E6 0B                        OUT    DMA+11,AL         ; OUTPUT THE MODE BYTE
04D2  8C C0                        MOV    AX,ES             ; GET THE ES VALUE
04D4  B1 04                        MOV    CL,4              ; SHIFT COUNT
04D6  D3 C0                        ROL    AX,CL             ; ROTATE LEFT
04D8  8A E8                        MOV    CH,AL             ; GET HIGHEST NYBBLE OF ES TO CH
04DA  24 F0                        AND    AL,0F0H           ; ZERO THE LOW NYBBLE FROM SEGMENT
04DC  03 C3                        ADD    AX,BX             ; TEST FOR CARRY FROM ADDITION
04DE  73 02                        JNC    J33
04E0  FE C5                        INC    CH                ; CARRY MEANS HIGH 4 BITS MUST BE INC
04E2                        J33:
04E2  50                           PUSH   AX                ; SAVE START ADDRESS
04E3  E6 04                        OUT    DMA+4,AL          ; OUTPUT LOW ADDRESS
04E5  EB 00                        JMP    $+2               ; WAIT FOR IO
04E7  8A C4                        MOV    AL,AH
04E9  E6 04                        OUT    DMA+4,AL          ; OUTPUT HIGH ADDRESS
04EB  8A C5                        MOV    AL,CH             ; GET HIGH 4 BITS
04ED  EB 00                        JMP    $+2               ; I/O WAIT STATE
04EF  24 0F                        AND    AL,0FH
04F1  E6 81                        OUT    081H,AL           ; OUTPUT THE HIGH 4 BITS TO PAGE REGISTER

                            ;------ DETERMINE COUNT

04F3  8A E6                        MOV    AH,DH             ; NUMBER OF SECTORS
04F5  2A C0                        SUB    AL,AL             ;  TIMES 256 INTO AX
04F7  D1 E8                        SHR    AX,1              ; SECTORS * 128 INTO AX
04F9  50                           PUSH   AX
04FA  BB 0006                      MOV    BX,6              ; GET THE BYTES/SECTOR PARM
04FD  E8 0382 R                    CALL   GET_PARM
0500  8A CC                        MOV    CL,AH             ; USE AS SHIFT COUNT (0=128, 1=256 ETC)
0502  58                           POP    AX
0503  D3 E0                        SHL    AX,CL             ; MULTIPLY BY CORRECT AMOUNT
0505  48                           DEC    AX                ;-1 FOR DMA VALUE
0506  50                           PUSH   AX                ; SAVE COUNT VALUE
0507  E6 05                        OUT    DMA+5,AL          ; LOW BYTE OF COUNT
0509  EB 00                        JMP    $+2               ; WAIT FOR IO
050B  8A C4                        MOV    AL,AH
050D  E6 05                        OUT    DMA+5,AL          ; HIGH BYTE OF COUNT
050F  FB                           STI                      ; RE-ENABLE INTERRUPTS
0510  59                           POP    CX                ; RECOVER COUNT VALUE
0511  58                           POP    AX                ; RECOVER ADDRESS VALUE
0512  03 C1                        ADD    AX,CX             ; ADD, TEST FOR 64K OVERFLOW
0514  59                           POP    CX                ; RECOVER REGISTER
0515  B0 02                        MOV    AL,2              ; MODE FOR 8237
0517  E6 0A                        OUT    DMA+10,AL         ; INITIALIZE THE DISKETTE CHANNEL
0519  C3                           RET                      ; RETURN TO CALLER, CFL SET BY ABOVE IF ERROR
051A                        DMA_SETUP      ENDP
                            ;--------------------------------------------------
                            ; CHK_STAT_2
                            ;       THIS ROUTINE HANDLES THE INTERRUPT RECEIVED AFTER
                            ;       A RECALIBRATE, SEEK, OR RESET TO THE ADAPTER.
                            ;       THE INTERRUPT IS WAITED FOR, THE INTERRUPT STATUS SENSED,
                            ;       AND THE RESULT RETURNED TO THE CALLER.
```

```
                        ; INPUT
                        ;       NONE
                        ; OUTPUT
                        ;       CY = 0 SUCCESS
                        ;       CY = 1 FAILURE -- ERROR IS IN DISKETTE_STATUS
                        ;       (AX) DESTROYED
                        ;------------------------------------------------
051A                    CHK_STAT_2      PROC    NEAR
051A  E8 053B R                 CALL    WAIT_INT                ; WAIT FOR THE INTERRUPT
051D  72 14                     JC      J34                     ; IF ERROR, RETURN IT
051F  B4 08                     MOV     AH,08H                  ; SENSE INTERRUPT STATUS COMMAND
0521  E8 03E2 R                 CALL    NEC_OUTPUT
0524  E8 0580 R                 CALL    RESULTS                 ; READ IN THE RESULTS
0527  72 0A                     JC      J34                     ; CHK2_RETURN
0529  A0 0042 R                 MOV     AL,NEC_STATUS           ; GET THE FIRST STATUS BYTE
052C  24 60                     AND     AL,060H                 ; ISOLATE THE BITS
052E  3C 60                     CMP     AL,060H                 ; TEST FOR CORRECT VALUE
0530  74 02                     JZ      J35                     ; IF ERROR, GO MARK IT
0532  F8                        CLC                             ; GOOD RETURN
0533                    J34:
0533  C3                        RET                             ; RETURN TO CALLER
0534                    J35:                                    ; CHK2_ERROR
0534  80 0E 0041 R 40           OR      DISKETTE_STATUS,BAD_SEEK
0539  F9                        STC                             ; ERROR RETURN CODE
053A  C3                        RET
053B                    CHK_STAT_2      ENDP

                        ;------------------------------------------
                        ; WAIT_INT
                        ;       THIS ROUTINE WAITS FOR AN INTERRUPT TO OCCUR
                        ;       A TIME OUT ROUTINE TAKES PLACE DURING THE WAIT, SO
                        ;       THAT AN ERROR MAY BE RETURNED IF THE DRIVE IS NOT READY
                        ; INPUT
                        ;       NONE
                        ; OUTPUT
                        ;       CY = 0 SUCCESS
                        ;       CY = 1 FAILURE -- DISKETTE_STATUS IS SET ACCORDINGLY
                        ;       (AX) DESTROYED
                        ;------------------------------------------
053B                    WAIT_INT        PROC    NEAR
053B  FB                        STI                             ; TURN ON INTERRUPTS, JUST IN CASE
053C  50                        PUSH    AX                      ; SAVE REGISTERS
053D  53                        PUSH    BX                      ; *
053E  51                        PUSH    CX                      ; *
053F  F8                        CLC                             ; CLEAR TIMEOUT INDICATOR
0540  B8 9001                   MOV     AX,09001H               ; LOAD WAIT CODE AND TYPE
0543  CD 15                     INT     15H                     ; PERFORM OTHER FUNCTION
0545  72 11                     JC      J36A                    ; BYPASS TIMING LOOP IF TIMEOUT OCCURRED
                        ;
0547  B3 04                     MOV     BL,4                    ; CLEAR THE COUNTERS
0549  33 C9                     XOR     CX,CX                   ; FOR 2 SECOND WAIT
054B                    J36:
054B  F6 06 003E R 80           TEST    SEEK_STATUS,INT_FLAG    ; TEST FOR INTERRUPT OCCURRING
0550  75 0C                     JNZ     J37
0552  E2 F7                     LOOP    J36                     ; COUNT DOWN WHILE WAITING
0554  FE CB                     DEC     BL                      ; SECOND LEVEL COUNTER
0556  75 F3                     JNZ     J36
                        ;
0558  80 0E 0041 R 80   J36A:   OR      DISKETTE_STATUS,TIME_OUT ; NOTHING HAPPENED
055D  F9                        STC                             ; ERROR RETURN
055E                    J37:
055E  9C                        PUSHF                           ; SAVE CURRENT CARRY
055F  80 26 003E R 7F           AND     SEEK_STATUS,NOT INT_FLAG ; TURN OFF INTERRUPT FLAG
0564  9D                        POPF                            ; RECOVER CARRY
0565  59                        POP     CX                      ; RECOVER REGISTERS
0566  5B                        POP     BX                      ; *
0567  58                        POP     AX                      ; *
0568  C3                        RET                             ; GOOD RETURN CODE COMES FROM TEST INST
0569                    WAIT_INT        ENDP

                        ;--------------------------------------------
                        ; DISK_INT
                        ;       THIS ROUTINE HANDLES THE DISKETTE INTERRUPT
                        ; INPUT
                        ;       NONE
                        ; OUTPUT
                        ;       THE INTERRUPT FLAG IS SET IS SEEK_STATUS
                        ;--------------------------------------------

0569                    DISK_INT_1      PROC    FAR             ;>>> ENTRY POINT FOR ORG 0EF57H
0569  FB                        STI                             ; RE ENABLE INTERRUPTS
056A  1E                        PUSH    DS                      ; SAVE REGISTERS
056B  50                        PUSH    AX                      ; *
056C  E8 0000 E                 CALL    DDS                     ; SETUP DATA ADDRESSING
056F  80 0E 003E R 80           OR      SEEK_STATUS,INT_FLAG    ; TURN ON INTERRUPT OCCURRED
0574  B0 20                     MOV     AL,20H                  ; END OF INTERRUPT MARKER
0576  E6 20                     OUT     20H,AL                  ; INTERRUPT CONTROL PORT
0578  B8 9101                   MOV     AX,09101H               ; INTERRUPT POST CODE & TYPE
057B  CD 15                     INT     15H                     ; GO PERFORM OTHER TASK
057D  58                        POP     AX                      ; RECOVER REG
057E  1F                        POP     DS                      ; *
057F  CF                        IRET                            ; RETURN FROM INTERRUPT
0580                    DISK_INT_1      ENDP
                        ;--------------------------------------------
                        ; RESULTS
                        ;       THIS ROUTINE WILL READ ANYTHING THAT THE NEC CONTROLLER
                        ;       HAS TO SAY FOLLOWING AN INTERRUPT.
                        ; INPUT
                        ;       NONE
                        ; OUTPUT
                        ;       CY = 0  SUCCESSFUL TRANSFER
                        ;       CY = 1  FAILURE -- TIME OUT IN WAITING FOR STATUS
                        ;       NEC_STATUS AREA HAS STATUS BYTE LOADED INTO IT
                        ;       (AH) DESTROYED
                        ;--------------------------------------------
0580                    RESULTS PROC    NEAR
0580  FC                        CLD
0581  BF 0042 R                 MOV     DI,OFFSET NEC_STATUS    ; POINTER TO DATA AREA
0584  51                        PUSH    CX                      ; SAVE COUNTER
0585  52                        PUSH    DX
0586  53                        PUSH    BX
0587  B3 07                     MOV     BL,7                    ; MAX STATUS BYTES

                        ;------ WAIT FOR REQUEST FOR MASTER
0589  B7 02             R10:    MOV     BH,2                    ; HIGH ORDER COUNTER
058B                    J38:                                    ; INPUT_LOOP
058B  33 C9                     XOR     CX,CX                   ; COUNTER
058D  BA 03F4                   MOV     DX,03F4H                ; STATUS PORT
0590                    J39:                                    ; WAIT FOR MASTER
0590  EC                        IN      AL,DX                   ; GET STATUS
0591  A8 80                     TEST    AL,080H                 ; MASTER READY
0593  75 10                     JNZ     J40A                    ; TEST_DIR
0595  E2 F9                     LOOP    J39                     ; WAIT_MASTER
                        ;
0597  FE CF                     DEC     BH                      ; DECREMENT HIGH ORDER COUNTER
```

```
0599  75 F0                   JNZ     J38                    ; REPEAT TIL DELAY DONE
                        ;
059B  80 0E 0041 R 80         OR      DISKETTE_STATUS,TIME_OUT
05A0                    J40:                                 ; RESULTS_ERROR
05A0  F9                      STC                            ; SET ERROR RETURN
05A1  5B                      POP     BX
05A2  5A                      POP     DX
05A3  59                      POP     CX
05A4  C3                      RET

                        ;------ TEST THE DIRECTION BIT

05A5  EC                J40A: IN      AL,DX              ; GET STATUS REG AGAIN
05A6  A8 40                   TEST    AL,040H            ; TEST DIRECTION BIT
05A8  75 07                   JNZ     J42                ; OK TO READ STATUS
05AA                    J41:                             ; NEC_FAIL
05AA  80 0E 0041 R 20         OR      DISKETTE_STATUS,BAD_NEC
05AF  EB EF                   JMP     J40                ; RESULTS_ERROR

                        ;------ READ IN THE STATUS

05B1                    J42:                             ; INPUT_STAT
05B1  42                      INC     DX                 ; POINT AT DATA PORT
05B2  EC                      IN      AL,DX                  ; GET THE DATA
05B3  88 05                   MOV     [DI],AL                ; STORE THE BYTE
05B5  47                      INC     DI                     ; INCREMENT THE POINTER
05B6  B9 0014                 MOV     CX,20                  ; LOOP TO KILL TIME FOR NEC
05B9  E2 FE             J43:  LOOP    J43
05BB  4A                      DEC     DX                     ; POINT AT STATUS PORT
05BC  EC                      IN      AL,DX                  ; GET STATUS
05BD  A8 10                   TEST    AL,010H                ; TEST FOR NEC STILL BUSY
05BF  74 06                   JZ      J44                    ; RESULTS DONE
05C1  FE CB                   DEC     BL                     ; DECREMENT THE STATUS COUNTER
05C3  75 C4                   JNZ     R10                   ; GO BACK FOR MORE
05C5  EB E3                   JMP     J41                    ; CHIP HAS FAILED

                        ;------ RESULT OPERATION IS DONE

05C7                    J44:
05C7  5B                      POP     BX
05C8  5A                      POP     DX
05C9  59                      POP     CX                     ; RECOVER REGISTERS
05CA  C3                      RET                            ; GOOD RETURN CODE FROM TEST INST
                        ;-----------------------------------------------
                        ; NUM_TRANS
                        ;       THIS ROUTINE CALCULATES THE NUMBER OF SECTORS THAT
                        ;       WERE ACTUALLY TRANSFERRED TO/FROM THE DISKETTE
                        ; INPUT
                        ;       (CH) = CYLINDER OF OPERATION
                        ;       (CL) = START SECTOR OF OPERATION
                        ; OUTPUT
                        ;       (AL) = NUMBER ACTUALLY TRANSFERRED
                        ;       NO OTHER REGISTERS MODIFIED
                        ;-----------------------------------------------
05CB                    NUM_TRANS       PROC    NEAR
05CB  A0 0045 R               MOV     AL,NEC_STATUS+3        ; GET CYLINDER ENDED UP ON
05CE  3A C5                   CMP     AL,CH                  ; SAME AS WE STARTED
05D0  A0 0047 R               MOV     AL,NEC_STATUS+5        ; GET ENDING SECTOR
05D3  74 0A                   JZ      J45                    ; IF ON SAME CYL, THEN NO ADJUST
05D5  BB 0008                 MOV     BX,8
05D8  E8 0382 R               CALL    GET_PARM               ; GET EOT VALUE
05DB  8A C4                   MOV     AL,AH                  ;  INTO AL
05DD  FE C0                   INC     AL                     ; USE EOT+1 FOR CALCULATION
05DF  2A C1             J45:  SUB     AL,CL                  ; SUBTRACT START FROM END
05E1  C3                      RET
05E2                    NUM_TRANS       ENDP
05E2                    RESULTS ENDP

                        ;-----------------------------------------------
                        ;
                        ;       HANDLE DISK CHANGE IF FOUND TO BE
                        ;       ACTIVE
                        ;
                        ;-----------------------------------------------
05E2  C6 87 0090 R 61   J1F:  MOV     DSK_STATE[BX],POA_DUAL  ; CLEAR STATE FOR THIS DRIVE
                        ;
                        ;       THIS SEQUENCE OF SEEKS IS USED TO RESET DISKETTE CHANGE SIGNAL
                        ;
05E7  E8 01CA R               CALL    DISK_RESET         ; RESET NEC
05EA  8B 56 00                MOV     DX,[BP]            ; RESTORE DRIVE PARMETER
05ED  B5 01                   MOV     CH,01H             ; MOVE TO CYLINDER 1
05EF  E8 041C R               CALL    SEEK               ; ISSUE SEEK
05F2  8B 56 00                MOV     DX,[BP]            ; RESTORE DRIVE PARMETER
05F5  B5 00                   MOV     CH,00H             ; MOVE TO CYLINDER 0
05F7  E8 041C R               CALL    SEEK               ; ISSUE SEEK
05FA  C6 06 0041 R 06         MOV     DISKETTE_STATUS,MEDIA_CHANGE ; INDICATE MEDIA REMOVED FROM DRIVE
05FF  5A                      POP     DX                 ; RESTORE PARAMETERS
0600  59                      POP     CX                 ; *
0601  5B                      POP     BX                 ; *
0602  58                      POP     AX                 ; *
0603  C3                      RET                        ; MEDIA CHANGE, GO DETERMINE NEW TYPE
                        ;-----------------------------------------------
                        ; READ_DSKCHNG
                        ;  THIS ROUTINE READS THE STATE OF THE
                        ;  DISK CHANGE LINE
                        ;    ZERO FLAG:
                        ;       0 - DISK CHANGE LINE INACTIVE
                        ;       1 - DISK CHANGE LINE ACTIVE
                        ;-----------------------------------------------
0604                    READ_DSKCHNG    PROC    NEAR
0604  32 FF                   XOR     BH,BH              ; CLEAR HIGH ORDER OFFSET
0606  8A DA                   MOV     BL,DL              ; LOAD DRIVE NUMBER AS OFFSET
0608  B0 01                   MOV     AL,01              ; MASK FOR DETERMINING MOTOR BIT
060A  80 26 003F R CF         AND     MOTOR_STATUS,0CFH ; CLEAR ENCODED DRIVE SELECT BITS(4 & 5)
060F  B1 04                   MOV     CL,4               ; SHIFT DRIVE NUMBER INTO HIGH NIBBLE COUNT
0611  D2 C3                   ROL     BL,CL              ; SHIFT DRIVE NUMBER INTO HIGH NIBBLE
0613  08 1E 003F R            OR      MOTOR_STATUS,BL    ; ADD IN DRIVE NUMBER SELECTED FOR LATER USE
0617  D2 CB                   ROR     BL,CL              ; RESTORE DRIVE NUMBER
0619  8A CB                   MOV     CL,BL              ; RESTORE DRIVE NUMBER
061B  D2 E0                   SHL     AL,CL              ; FORM MOTOR ON BIT MASK
061D  FA                      CLI                        ; NO INTERRUPTS WHILE DETERMING MOTOR STATUS
061E  84 06 003F R            TEST    AL,MOTOR_STATUS    ; TEST
0622  75 09                   JNZ     R8                 ; DONT NEED TO SELECT DEVICE IF MOTOR ON
                        ;
0624  08 06 003F R            OR      MOTOR_STATUS,AL ; TURN ON CURRENT MOTOR
0628  C6 06 0040 R FF         MOV     MOTOR_COUNT,0FFH ; SET LARGE COUNT DURING OPERATION
062D  FB                R8:   STI                        ; ENABLE INTERRUPTS AGAIN
062E  BA 03F2                 MOV     DX,03F2H           ; ADDRESS DIGITAL OUTPUT REGISTER
0631  A0 003F R               MOV     AL,MOTOR_STATUS    ; GET DIGITAL OUTPUT REGISTER REFLECTION
0634  24 3F                   AND     AL,03FH            ; STRIP AWAY UNWANTED BITS
0636  B1 04                   MOV     CL,4               ; SHIFT COUNT
0638  D2 C0                   ROL     AL,CL              ; PUT BITS IN DESIRED POSITIONS
063A  0C 0C                   OR      AL,0CH             ; NO RESET, ENABLE DMA/INT
063C  EE                      OUT     DX,AL              ; SELECT DRIVE
063D  BA 03F7                 MOV     DX,03F7H           ; ADDRESS DIGITIAL INPUT REGISTER
0640  EB 00                   JMP     $+2                ; DELAY FOR SUPPORT CHIP
```

```
0642  EC                       IN     AL,DX              ; INPUT DIR
0643  A8 80                    TEST   AL,DSK_CHG         ; CHECK FOR DISK CHANGE LINE ACTIVE
0645  C3                       RET                       ; RETURN TO CALLER WITH ZERO FLAG SET
0646                   READ_DSKCHNG    ENDP
                       ;-------------------------------------------------
                       ; DISK_CHANGE
                       ;   THIS ROUTINE RETURNS THE STATE OF THE
                       ;   DISK CHANGE LINE
                       ;     DISKETTE_STATUS:
                       ;        00 - DISK CHANGE LINE INACTIVE
                       ;        06 - DISK CHANGE LINE ACTIVE
                       ;-------------------------------------------------
0646                   DISK_CHANGE     PROC    NEAR
0646  F6 06 008F R 01          TEST   HF_CNTRL,DUAL      ; GO DETERMINE TYPE OF CONTROLLER CARD
064B  74 29                    JZ     DC2                ; DISKETTE ATTACH CARD, SET CHANGE LINE ACTIVE
                       ;
064D  32 FF                    XOR    BH,BH              ; CLEAR HIGH ORDER OFFSET
064F  8A DA                    MOV    BL,DL              ; LOAD DRIVE NUMBER AS OFFSET
0651  8A 87 0090 R             MOV    AL,DSK_STATE[BX] ; GET MEDIA STATE INFORMATION FOR DRIVE
0655  24 07                    AND    AL,STATE_MSK       ; ISOLATE STATE
0657  3C 03                    CMP    AL,3               ; CHECK FOR 48TPI DRIVE & NOT ESTABLISHED STATES
0659  74 07                    JE     SETIT              ; IF FOUND SET DISK CHANGE ACTIVE
                       ;
065B  72 0B                    JB     DCO                ; IF NOT ESTABLISHED, GO CHECK FOR NO DRIVE
                       ;
065D  E8 0604 R                CALL   READ_DSKCHNG       ; GO CHECK STATE OF DISK CHANGE LINE
0660  74 05                    JZ     FINIS              ; CHANGE LINE NOT ACTIVE, RETURN
                       ;
0662  C6 06 0041 R 06  SETIT:  MOV    DISKETTE_STATUS,MEDIA_CHANGE ; INDICATE MEDIA REMOVED FROM DRIVE
0667  C3               FINIS:  RET                       ; RETURN TO CALLER
                       ;
0668  8A 87 0090 R     DCO:    MOV    AL,DSK_STATE[BX] ; GET MEDIA STATE INFORMATION FOR DRIVE
066C  0A C0                    OR     AL,AL              ; CHECK FOR NO DRIVE INSTALLED
066E  75 F2                    JNZ    SETIT              ; IF DRIVE PRESENT, SET CHANGE LINE ACTIVE
                       ;
0670  80 0E 0041 R 80  DC1:    OR     DISKETTE_STATUS,TIME_OUT ; SET TIMEOUT BECAUSE NO DRIVE PRESENT
0675  C3                       RET                       ; RETURN TO CALLER
                       ;
0676  B0 0E            DC2:    MOV    AL,CMOSDSB_ADDR ; GET CMOS DIAGNOSTIC STATUS BYTE ADDRESS
0678  E6 70                    OUT    CADR_PRT,AL        ; WRITE ADDRESS TO READ OUT TO CMOS
067A  EB 00                    JMP    $+2                ; DELAY
067C  E4 71                    IN     AL,CDATA_PRT       ; GET CMOS STATUS
067E  A8 C0                    TEST   AL,CMOS_GOOD       ; SEE IF BATTERY GOOD AND CHECKSUM VALID
0680  75 EE                    JNZ    DC1                ; ERROR IF EITHER BIT ON
                       ;
0682  B0 10                    MOV    AL,CMOSDSK_BYTE ; ADDRESS OF DSKETTE BYTE IN CMOS
0684  E6 70                    OUT    CADR_PRT,AL        ; WRITE ADDRESS TO READ OUT TO CMOS
0686  EB 00                    JMP    $+2                ; DELAY
0688  E4 71                    IN     AL,CDATA_PRT       ; GET DSKETTE BYTE
068A  0A D2                    OR     DL,DL              ; SEE WHICH DRIVE IN QUESTION
068C  75 04                    JNZ    DC3                ; IF DRIVE 1, DATA ALREADY IN LOW NIBBLE
                       ;
068E  B1 04                    MOV    CL,4               ; GET ROTATE COUNT TO SHIFT HIGH TO LOW NIBBLE
0690  D2 C8                    ROR    AL,CL              ; EXCHANGE NIBBLES
0692  24 0F            DC3:    AND    AL,LOWNIB          ; CLEAR AWAY UNDESIRED DRIVE DATA
0694  74 DA                    JZ     DC1                ; NO DRIVE THEN SET TIMEOUT ERROR
                       ;
0696  EB CA                    JMP    SHORT SETIT        ; DRIVE, ON 320/360K DRIVES SET DISK CHANGE
0698                   DISK_CHANGE     ENDP
                       ;-------------------------------------------------
                       ; DISK_TYPE
                       ;   THIS ROUTINE IS USED TO EITHER ESTABLISH THE
                       ;   TYPE OF MEDIA/DRIVE TO BE USED IN THE NEXT
                       ;   OPERATION(FOR FORMAT ONLY) OR RETURN THE
                       ;   TYPE OF MEDIA/DRIVE INSTALLED AT THE DRIVE
                       ;   SPECIFIED
                       ;-------------------------------------------------
0698                   DISK_TYPE       PROC    NEAR
0698  F6 06 008F R 01          TEST   HF_CNTRL,DUAL      ; GO DETERMINE TYPE OF CONTROLLER CARD
069D  74 49                    JZ     T2                 ; DISKETTE ATTACH CARD, GO DO TYPE OPERATION
                       ;
069F  32 FF                    XOR    BH,BH              ; CLEAR HIGH ORDER OFFSET
06A1  8A DA                    MOV    BL,DL              ; LOAD DRIVE NUMBER AS OFFSET
06A3  8A A7 0090 R             MOV    AH,DSK_STATE[BX] ; GET PRESENT STATE INFORMATION
                       ;
06A7  F6 C4 10                 TEST   AH,DETERMINED      ; SEE IF MEDIA/DRIVE TYPE ALREADY ESTABLISHED
06AA  74 0B                    JZ     T5                 ; IF NOT, GO RETURN ZERO VALUE
                       ;
06AC  80 E4 07                 AND    AH,STATE_MSK       ; STRIP OFF HIGH ORDER BITS
06AF  80 EC 03                 SUB    AH,03H             ; CONVERT TO TYPE FOR OUTPUT
06B2  75 0C                    JNZ    T7                 ; SKIP IF NOT 320/360 DRIVE AND MEDIA
                       ;
06B4  B0 01                    MOV    AL,NOCHGLN         ; INDICATE NO CHANGE LINE AVAILABLE
06B6  C3                       RET                       ; RETURN TO CALLER
                       ;
06B7  0A E4            T5:     OR     AH,AH              ; CHECK FOR NO DRIVE
06B9  74 2A                    JZ     T1                 ; IF NONE GO INDICATE SUCH TO CALLER
                       ;
06BB  80 E4 07                 AND    AH,STATE_MSK       ; STRIP OFF HIGH ORDER BITS
06BE  74 03                    JZ     TA                 ; IF STATE 0 CHECK CMOS
                       ;
06C0  B0 02            T7:     MOV    AL,CHGLN           ; 1.2 DRIVE
06C2  C3                       RET                       ; RETURN TO CALLER
                       ;
06C3  B0 0E            TA:     MOV    AL,CMOSDSB_ADDR ; GET CMOS DIAGNOSTIC STATUS BYTE ADDRESS
06C5  E6 70                    OUT    CADR_PRT,AL        ; WRITE ADDRESS TO READ OUT TO CMOS
06C7  EB 00                    JMP    $+2                ; DELAY
06C9  E4 71                    IN     AL,CDATA_PRT       ; GET CMOS STATUS
06CB  A8 C0                    TEST   AL,CMOS_GOOD       ; SEE IF BATTERY GOOD AND CHECKSUM VALID
06CD  75 16                    JNZ    T1                 ; ERROR IF EITHER BIT ON
                       ;
06CF  B0 10                    MOV    AL,CMOSDSK_BYTE ; ADDRESS OF DSKETTE BYTE IN CMOS
06D1  E6 70                    OUT    CADR_PRT,AL        ; WRITE ADDRESS TO READ OUT TO CMOS
06D3  EB 00                    JMP    $+2                ; DELAY
06D5  E4 71                    IN     AL,CDATA_PRT       ; GET DSKETTE BYTE
06D7  0A D2                    OR     DL,DL              ; SEE WHICH DRIVE IN QUESTION
06D9  75 04                    JNZ    TB                 ; IF DRIVE 1, DATA ALREADY IN LOW NIBBLE
                       ;
06DB  B1 04                    MOV    CL,4               ; GET ROTATE COUNT TO SHIFT HIGH TO LOW NIBBLE
06DD  D2 C8                    ROR    AL,CL              ; EXCHANGE NIBBLES
06DF  24 0F            TB:     AND    AL,LOWNIB          ; CLEAR AWAY UNDESIRED DRIVE DATA
06E1  3C 03                    CMP    AL,3               ; SEE IF UNDEFINED DISKETTE TYPE
06E3  72 02                    JB     TC                 ; RETURN IF NOT, RESULTS IN AL
                       ;
06E5  32 C0            T1:     XOR    AL,AL              ; STATE NO DRIVE PRESENT OR UNKNOWN
06E7  C3               TC:     RET                       ; RETURN TO CALLER
                       ;
06E8  B0 0E            T2:     MOV    AL,CMOSDSB_ADDR ; GET CMOS DIAGNOSTIC STATUS BYTE ADDRESS
06EA  E6 70                    OUT    CADR_PRT,AL        ; WRITE ADDRESS TO READ OUT TO CMOS
06EC  EB 00                    JMP    $+2                ; DELAY
06EE  E4 71                    IN     AL,CDATA_PRT       ; GET CMOS STATUS
06F0  A8 C0                    TEST   AL,CMOS_GOOD       ; SEE IF BATTERY GOOD AND CHECKSUM VALID
06F2  75 F1                    JNZ    T1                 ; ERROR IF EITHER BIT ON
                       ;
06F4  B0 10                    MOV    AL,CMOSDSK_BYTE ; ADDRESS OF DSKETTE BYTE IN CMOS
06F6  E6 70                    OUT    CADR_PRT,AL        ; WRITE ADDRESS TO READ OUT TO CMOS
```

SECTION 5

```
06F8  EB 00                      JMP    $+2              ; DELAY
06FA  E4 71                      IN     AL,CDATA_PRT     ; GET DSKETTE BYTE
06FC  0A D2                      OR     DL,DL            ; SEE WHICH DRIVE IN QUESTION
06FE  75 04                      JNZ    T3               ; IF DRIVE 1, DATA ALREADY IN LOW NIBBLE
                         ;
0700  B1 04                      MOV    CL,4             ; GET ROTATE COUNT TO SHIFT HIGH TO LOW NIBBLE
0702  D2 C8                      ROR    AL,CL            ; EXCHANGE NIBBLES
0704  24 0F              T3:     AND    AL,LOWNIB        ; CLEAR AWAY UNDESIRED DRIVE DATA
0706  3C 02                      CMP    AL,INVALID_DRV   ; SEE IF UNDEFINED DISKETTE TYPE
0708  72 02                      JB     T6               ; RETURN IF NOT, RESULTS IN AL
                         ;
070A  32 C0                      XOR    AL,AL            ; STATE NO DRIVE PRESENT OR UNKNOWN
070C  C3                 T6:     RET                     ; RETURN TO CALLER
070D                     DISK_TYPE ENDP
                         ;-------------------------------------------------
                         ; FORMAT_SET
                         ;  THIS ROUTINE IS USED TO ESTABLISH THE
                         ;  TYPE OF MEDIA/DRIVE TO BE USED FOR THE FOLLOWING
                         ;  FORMAT OPERATION
                         ;-------------------------------------------------
070D                     FORMAT_SET      PROC    NEAR
070D  F6 06 008F R 01            TEST   HF_CNTRL,DUAL    ; GO DETERMINE TYPE OF CONTROLLER CARD
0712  74 5C                      JZ     S0               ; DISKETTE ATTACH CARD, GO DO TYPE OPERATION
                         ;
0714  32 FF                      XOR    BH,BH            ; CLEAR HIGH ORDER OFFSET
0716  8A DA                      MOV    BL,DL            ; LOAD DRIVE NUMBER AS OFFSET
0718  FE C8                      DEC    AL               ; CHECK FOR 320/360K MEDIA & DRIVE
071A  75 06                      JNZ    S1               ; BYPASS IF NOT
                         ;
071C  C6 87 0090 R 93            MOV    DSK_STATE[BX],M326D326 ; SET STATE VARIABLE
0721  C3                         RET                     ; RETURN TO CALLER
                         ;
0722  50                 S1:     PUSH   AX               ; SAVE TYPE VALUE
0723  E8 0604 R                  CALL   READ_DSKCHNG     ; GO CHECK DISK CHANGE LINE
0726  74 2E                      JZ     S3               ; NOT ACTIVE GO ON PROCESSING
                         ;
0728  C6 06 0041 R 06            MOV    DISKETTE_STATUS,MEDIA_CHANGE ; INDICATE DISK CHANGE ACTIVE
072D  8B 56 00                   MOV    DX,[BP]          ; RESTORE DRIVE PARMETER
0730  B5 01                      MOV    CH,01H           ; MOVE TO CYLINDER 1
0732  E8 041C R                  CALL   SEEK             ; ISSUE SEEK
0735  8B 56 00                   MOV    DX,[BP]          ; RESTORE DRIVE PARMETER
0738  B5 00                      MOV    CH,00H           ; MOVE TO CYLINDER 0
073A  E8 041C R                  CALL   SEEK             ; ISSUE SEEK
073D  8B 56 00                   MOV    DX,[BP]          ; RESTORE DRIVE PARMETER
0740  E8 0604 R                  CALL   READ_DSKCHNG     ; GO CHECK DISK CHANGE LINE
0743  74 11                      JZ     S3               ; CHANGE LINE INACTIVE, GO SET TYPE
                         ;
0745  58                         POP    AX               ; RESTORE TYPE VALUE
0746  C6 06 0041 R 80            MOV    DISKETTE_STATUS,TIME_OUT ; INDICATE NO MEDIA IN DRIVE
074B  8B 5E 00                   MOV    BX,[BP]          ; RESTORE DRIVE PARMETER FOR USE AS INDEX
074E  32 FF                      XOR    BH,BH            ; CLEAR HIGH ORDER OFFSET
0750  C6 87 0090 R 61            MOV    DSK_STATE[BX],POA_DUAL ; SET STATE TO POWER ON ASSUMPTION
0755  C3                         RET                     ; RETURN TO CALLER
                         ;
0756  58                 S3:     POP    AX               ; RESTORE TYPE VALUE
0757  FE C8                      DEC    AL               ; CHECK FOR 320/360K MEDIA IN 1.2M DRIVE
0759  75 06                      JNZ    S2               ; BYPASS IF NOT
                         ;
075B  C6 87 0090 R 74            MOV    DSK_STATE[BX],M326D12 ; SET STATE VARIABLE
0760  C3                         RET                     ; RETURN TO CALLER
                         ;
0761  FE C8              S2:     DEC    AL               ; CHECK FOR 1.2M MEDIA IN 1.2M DRIVE
0763  75 06                      JNZ    SE               ; BYPASS IF NOT, ERROR CONDITION NOW EXISTS
                         ;
0765  C6 87 0090 R 15            MOV    DSK_STATE[BX],M12D12 ; SET STATE VARIABLE
076A  C3                         RET                     ; RETURN TO CALLER
                         ;
076B  C6 06 0041 R 01    SE:     MOV    DISKETTE_STATUS,BAD_CMD ; UNKNOWN STATE,BAD COMMAND
0770  C3                 S0:     RET                     ; RETURN TO CALLER
0771                     FORMAT_SET      ENDP

                         ;-------------------------------------------------
                         ; DSKETTE_SETUP
                         ;  THIS ROUTINE DOES A PRELIMINARY CHECK TO SEE
                         ;  WHAT TYPE OF DISKETTE DRIVES ARE ATTACH TO THE
                         ;  SYSTEM.  TEST IS ONLY PERFORMED WHEN A DUAL
                         ;  ATTACHMENT CARD EXISTS.
                         ;-------------------------------------------------
0771                     DSKETTE_SETUP PROC      NEAR
0771  50                         PUSH   AX               ; SAVE REGISTERS
0772  53                         PUSH   BX               ; *
0773  51                         PUSH   CX               ; *
0774  52                         PUSH   DX               ; *
0775  56                         PUSH   SI               ; *
0776  57                         PUSH   DI               ; *
0777  06                         PUSH   ES               ; *
0778  1E                         PUSH   DS               ; *
0779  55                         PUSH   BP               ; *
077A  E8 0000 E                  CALL   DDS              ; LOAD DATA SEGMENT REGISTER TO ROM BIOS AREA
077D  BB 0000                    MOV    BX,0             ; INITIALIZE DRIVE POINTER
0780  C7 87 0090 R 0000          MOV    WORD PTR DSK_STATE[BX],0 ; INITIALIZE STATES
0786  C7 87 0092 R 0000          MOV    WORD PTR DSK_STATE[BX+2],0 ; INITIALIZE START STATES
078C  C6 06 008B R 00            MOV    LASTRATE,0       ; INITIALIZE LAST DATA TRANSFER RATE
0791  C6 06 003E R 00            MOV    SEEK_STATUS,0    ; INDICATE RECALIBRATES NEEDED
0796  C6 06 0040 R 00            MOV    MOTOR_COUNT,0    ; INITIALIZE MOTOR COUNT
079B  C6 06 003F R 00            MOV    MOTOR_STATUS,0   ; INITIALIZE DRIVES TO OFF STATE
07A0  53                 SUP0:   PUSH   BX               ; SAVE POINTER
07A1  B0 01                      MOV    AL,01            ; MASK FOR DETERMINING MOTOR BIT
07A3  80 26 003F R CF            AND    MOTOR_STATUS,0CFH ; CLEAR ENCODED DRIVE SELECT BITS(4 & 5)
07A8  B1 04                      MOV    CL,4             ; SHIFT DRIVE NUMBER INTO HIGH NIBBLE COUNT
07AA  D2 C3                      ROL    BL,CL            ; SHIFT DRIVE NUMBER INTO HIGH NIBBLE
07AC  08 1E 003F R               OR     MOTOR_STATUS,BL  ; ADD IN DRIVE NUMBER SELECTED FOR LATER USE
07B0  D2 CB                      ROR    BL,CL            ; RESTORE DRIVE NUMBER
07B2  8A CB                      MOV    CL,BL            ; RESTORE DRIVE NUMBER
07B4  D2 E0                      SHL    AL,CL            ; FORM MOTOR ON BIT MASK
07B6  FA                         CLI                     ; NO INTERRUPTS WHILE DETERMING MOTOR STATUS
07B7  84 06 003F R               TEST   AL,MOTOR_STATUS  ; TEST
07BB  75 09                      JNZ    SUP2             ; DONT NEED TO SELECT DEVICE IF MOTOR ON
                         ;
07BD  08 06 003F R               OR     MOTOR_STATUS,AL  ; TURN ON CURRENT MOTOR
07C1  C6 06 0040 R FF            MOV    MOTOR_COUNT,0FFH ; SET LARGE COUNT DURING OPERATION
07C6  FB                 SUP2:   STI                     ; ENABLE INTERRUPTS AGAIN
07C7  BA 03F2                    MOV    DX,03F2H         ; ADDRESS DIGITAL OUTPUT REGISTER
07CA  A0 003F R                  MOV    AL,MOTOR_STATUS  ; GET DIGITAL OUTPUT REGISTER REFLECTION
07CD  24 3F                      AND    AL,03FH          ; STRIP AWAY UNWANTED BITS
07CF  B1 04                      MOV    CL,4             ; SHIFT COUNT
07D1  D2 C0                      ROL    AL,CL            ; PUT BITS IN DESIRED POSITIONS
07D3  0C 0C                      OR     AL,0CH           ; NO RESET, ENABLE DMA/INT
07D5  EE                         OUT    DX,AL            ; SELECT DRIVE
07D6  8B D3                      MOV    DX,BX            ; ESTABLISH DRIVE PARM FOR SEEK ROUTINE
07D8  B5 30                      MOV    CH,TRK_SLAP      ; GET TRACK TO SEEK TO(>40)
07DA  E8 041C R                  CALL   SEEK             ; SEEK TO TRACK
07DD  5A                         POP    DX               ; RESTORE POINTER
07DE  52                         PUSH   DX               ; SAVE POINTER
07DF  B5 0A                      MOV    CH,QUIET_SEEK    ; SEEK SO FAR IN, BEFORE ISSUING SINGLE STEPS
07E1  E8 041C R                  CALL   SEEK             ; SEEK TO TRACK 10
```

```
07E4  B5 0A                  MOV    CH,QUIET_SEEK     ; GET TRACK AT PRESENTLY
07E6  33 F6                  XOR    SI,SI             ; CLEAR SEEK COUNTER
07E8  FE CD          SUP3:   DEC    CH                ; SEEK TO NEXT TRACK, TOWARDS TRACK 0
07EA  5A                     POP    DX                ; RESTORE POINTER
07EB  52                     PUSH   DX                ; SAVE POINTER
07EC  56                     PUSH   SI                ; SAVE COUNTER
07ED  E8 041C R              CALL   SEEK              ; SEEK TO TRACK
                     ;
07F0  B4 04                  MOV    AH,SENSE_DRV_ST ; SENSE DRIVE STATUS COMMAND BYTE
07F2  E8 082C R              CALL   SUP5              ; ISSUE THE COMMAND
07F5  E8 0580 R              CALL   RESULTS           ; GO GET STATUS
07F8  5E                     POP    SI                ; RESTORE COUNTER
07F9  46                     INC    SI                ; COUNT NUMBER OF SEEKS TIL AT HOME(TRACK 0)
                     ;
07FA  F6 06 0042 R 10        TEST   NEC_STATUS,HOME ; LOOK TO SEE IF HEAD IS AT TRACK 0
07FF  75 08                  JNZ    SUP4              ; GO DETERMINE DRIVE TYPE
                     ;
0801  83 FE 0B               CMP    SI,QUIET_SEEK+1 ; SEE IF THE NUMBER OF SEEKS = NUMBER ISSUED
0804  72 E2                  JB     SUP3              ; IF LESS THAN, NOT DONE YET
                     ;
0806  5B                     POP    BX                ; RESTORE POINTER
0807  EB 10                  JMP    SHORT NXT_DRV     ; DRIVE NOT INSTALLED, BYPASS
                     ;
0809  5B             SUP4:   POP    BX                ; RESTORE POINTER
080A  83 FE 0A               CMP    SI,QUIET_SEEK     ; SEE IF SEEKS STEPPED EQUAL THE ORIGINAL
080D  C6 87 0090 R 61        MOV    DSK_STATE[BX],POA_DUAL ; SETUP POWER ON ASSUMPTION
0812  73 05                  JAE    NXT_DRV           ; IF YES 1.2 DRIVE
                     ;
0814  C6 87 0090 R 93        MOV    DSK_STATE[BX],M326D326 ; ESTABLISH 320/360K STATE
0819                 NXT_DRV:
0819  43                     INC    BX                ; POINT TO NEXT DRIVE
081A  83 FB 02               CMP    BX,MAX_DRV        ; SEE IF DONE
081D  74 03                  JE     SUP1              ; IF FINISHED LEAVE TEST
                     ;
081F  E9 07A0 R              JMP    SUP0              ; REPEAT TIL DONE FOR EACH DRIVE
                     ;
0822  5D             SUP1:   POP    BP                ; RESTORE ALL REGISTERS
0823  1F                     POP    DS                ; *
0824  07                     POP    ES                ; *
0825  5F                     POP    DI                ; *
0826  5E                     POP    SI                ; *
0827  5A                     POP    DX                ; *
0828  59                     POP    CX                ; *
0829  5B                     POP    BX                ; *
082A  58                     POP    AX                ; *
082B  C3                     RET                      ; OTHERWISE RETURN

                     ;------- KEEP STACK CORRECT FOR CALL TO NEC_OUTPUT IF ERROR

082C  E8 03E2 R      SUP5:   CALL   NEC_OUTPUT        ; OUTPUT TO NEC
082F  8A E2                  MOV    AH,DL             ; GET DRIVE NUMBER SELECTED
0831  E8 03E2 R              CALL   NEC_OUTPUT        ; OUTPUT TO NEC
0834  C3                     RET                      ;

0835                 DSKETTE_SETUP ENDP

0835                 CODE    ENDS
                             END
```

SECTION 5

```
TITLE FIXED DISK BIOS FOR IBM DISK CONTROLLER 1-11-84

PUBLIC  DISK_IO
PUBLIC  HD_INT
PUBLIC  DISK_SETUP

EXTRN   F1780:NEAR
EXTRN   F1781:NEAR
EXTRN   F1782:NEAR
EXTRN   F1790:NEAR
EXTRN   F1791:NEAR
EXTRN   FD_TBL:NEAR
;-- INT 13 ---------------------------------------------------------
;                                                                   :
; 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         :
;       THE LISTINGS  ARE INCLUDED  ONLY FOR  COMPLETENESS,         :
;       NOT FOR  REFERENCE.  APPLICATIONS WHICH  REFERENCE          :
;       ABSOLUTE   ADDRESSES   WITHIN   THE   CODE  SEGMENT         :
;       VIOLATE THE STRUCTURE AND DESIGN OF BIOS.                   :
;-------------------------------------------------------------------

; INPUT    (AH = HEX VALUE)
;
;       (AH)=00 RESET DISK (DL = 80H,81H) / DISKETTE
;       (AH)=01 READ THE STATUS OF THE LAST DISK OPERATION INTO (AL)
;               NOTE: DL < 80H - DISKETTE
;                     DL > 80H - DISK
;       (AH)=02 READ THE DESIRED SECTORS INTO MEMORY
;       (AH)=03 WRITE THE DESIRED SECTORS FROM MEMORY
;       (AH)=04 VERIFY THE DESIRED SECTORS
;       (AH)=05 FORMAT THE DESIRED TRACK
;       (AH)=06 UNUSED
;       (AH)=07 UNUSED
;       (AH)=08 RETURN THE CURRENT DRIVE PARAMETERS
;       (AH)=09 INITIALIZE DRIVE PAIR CHARACTERISTICS
;               INTERRUPT 41 POINTS TO DATA BLOCK FOR DRIVE 0
;               INTERRUPT 46 POINTS TO DATA BLOCK FOR DRIVE 1
;       (AH)=0A READ LONG
;       (AH)=0B WRITE LONG
;       NOTE: READ AND WRITE LONG ENCOMPASS 512 + 4 BYTES ECC
;       (AH)=0C SEEK
;       (AH)=0D ALTERNATE DISK RESET (SEE DL)
;       (AH)=0E UNUSED
;       (AH)=0F UNUSED
;       (AH)=10 TEST DRIVE READY
;       (AH)=11 RECALIBRATE
;       (AH)=12 UNUSED
;       (AH)=13 UNUSED
;       (AH)=14 CONTROLLER INTERNAL DIAGNOSTIC
;       (AH)=15 READ DASD TYPE
PAGE
;               REGISTERS USED FOR FIXED DISK OPERATIONS
;
;               (DL)    -  DRIVE NUMBER     (80H-81H FOR DISK, VALUE CHECKED)
;               (DH)    -  HEAD NUMBER      (0-15 ALLOWED, NOT VALUE CHECKED)
;               (CH)    -  CYLINDER NUMBER  (0-1023, NOT VALUE CHECKED)(SEE CL)
;               (CL)    -  SECTOR NUMBER    (1-17, NOT VALUE CHECKED)
;
;                          NOTE: HIGH 2 BITS OF CYLINDER NUMBER ARE PLACED
;                                IN THE HIGH 2 BITS OF THE CL REGISTER
;                                (10 BITS TOTAL)
;               (AL)    -  NUMBER OF SECTORS (MAXIMUM POSSIBLE RANGE 1-80H,
;                                             FOR READ/WRITE LONG 1-79H)
;               (ES:BX) -  ADDRESS OF BUFFER FOR READS AND WRITES,
;                          (NOT REQUIRED FOR VERIFY)
;
;               FORMAT (AH=5) ES:BX POINTS TO A 512 BYTE BUFFER.  THE FIRST
;                          2*(SECTORS/TRACK) BYTES CONTAIN F,N FOR EACH SECTOR.
;                          F = 00H FOR A GOOD SECTOR
;                              80H FOR A BAD SECTOR
;                          N = SECTOR NUMBER
;                          FOR AN INTERLEAVE OF 2 AND 17 SECTORS/TRACK
;                          THE TABLE SHOULD BE:
;               DB      00H,01H,00H,0AH,00H,02H,00H,0BH,00H,03H,00H,0CH
;               DB      00H,04H,00H,0DH,00H,05H,00H,0EH,00H,06H,00H,0FH
;               DB      00H,07H,00H,10H,00H,08H,00H,11H,00H,09H
;
; 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 = NUMBER OF CONSECUTIVE ACKNOWLEDGING DRIVES ATTACHED (0-2)
;            (CONTROLLER CARD ZERO TALLY ONLY)
;       DH = MAXIMUM USEABLE VALUE FOR HEAD NUMBER
;       CH = MAXIMUM USEABLE VALUE FOR CYLINDER NUMBER
;       CL = MAXIMUM USEABLE VALUE FOR SECTOR NUMBER
;            AND CYLINDER NUMBER HIGH BITS
;
;       IF READ DASD TYPE WAS REQUESTED,
;
;       AH = 0 - NOT PRESENT
;            1 - DISKETTE - NO CHANGE LINE AVAILABLE
;            2 - DISKETTE - CHANGE LINE AVAILABLE
;            3 - FIXED DISK
;       CX,DX = NUMBER OF 512 BYTE BLOCKS WHEN AH = 3
;
;       REGISTERS WILL BE PRESERVED EXCEPT WHEN THEY ARE USED TO RETURN
;       INFORMATION.
;
;       NOTE: IF AN ERROR IS REPORTED BY THE DISK CODE, THE APPROPRIATE
;             ACTION IS TO RESET THE DISK, THEN RETRY THE OPERATION.
;
;-------------------------------------------------------------------------
= 00FF          SENSE_FAIL      EQU     0FFH            ; NOT IMPLEMENTED
= 00E0          NO_ERR          EQU     0E0H            ; STATUS ERROR/ERROR REG=0
= 00CC          WRITE_FAULT     EQU     0CCH            ; WRITE FAULT ON SELECTED DRIVE
= 00BB          UNDEF_ERR       EQU     0BBH            ; UNDEFINED ERROR OCCURRED
= 00AA          NOT_RDY         EQU     0AAH            ; DRIVE NOT READY
= 0080          TIME_OUT        EQU     80H             ; ATTACHMENT FAILED TO RESPOND
= 0040          BAD_SEEK        EQU     40H             ; SEEK OPERATION FAILED
```

SECTION 5

```
= 0020                        BAD_CNTLR       EQU     20H             ; CONTROLLER HAS FAILED
= 0011                        DATA_CORRECTED  EQU     11H             ; ECC CORRECTED DATA ERROR
= 0010                        BAD_ECC         EQU     10H             ; BAD ECC ON DISK READ
= 000B                        BAD_TRACK       EQU     0BH             ; NOT IMPLEMENTED
= 000A                        BAD_SECTOR      EQU     0AH             ; BAD SECTOR FLAG DETECTED
= 0009                        DMA_BOUNDARY    EQU     09H             ; DATA EXTENDS TOO FAR
= 0007                        INIT_FAIL       EQU     07H             ; DRIVE PARAMETER ACTIVITY FAILED
= 0005                        BAD_RESET       EQU     05H             ; RESET FAILED
= 0004                        RECORD_NOT_FND  EQU     04H             ; REQUESTED SECTOR NOT FOUND
= 0002                        BAD_ADDR_MARK   EQU     02H             ; ADDRESS MARK NOT FOUND
= 0001                        BAD_CMD         EQU     01H             ; BAD COMMAND PASSED TO DISK I/O
                              PAGE
                              ;-------------------------------------------------------
                              ; FIXED DISK PARAMETER TABLE                            :
                              ;                                                       :
                              ;   -  THE TABLE IS COMPOSED OF A BLOCK DEFINED AS:     :
                              ;                                                       :
                              ;   +0    (1 WORD) - MAXIMUM NUMBER OF CYLINDERS        :
                              ;   +2    (1 BYTE) - MAXIMUM NUMBER OF HEADS            :
                              ;   +3    (1 WORD) - NOT USED/SEE PC-XT                 :
                              ;   +5    (1 WORD) - STARTING WRITE PRECOMPENSATION CYL :
                              ;   +7    (1 BYTE) - MAXIMUM ECC DATA BURST LENGTH      :
                              ;   +8    (1 BYTE) - CONTROL BYTE                       :
                              ;                    BIT    7 DISABLE RETRIES -OR-      :
                              ;                    BIT    6 DISABLE RETRIES           :
                              ;                    BIT    3 MORE THAN 8 HEADS         :
                              ;   +9    (3 BYTES)- NOT USED/SEE PC-XT                 :
                              ;  +12    (1 WORD) - LANDING ZONE                       :
                              ;  +14    (1 BYTE) - NUMBER OF SECTORS/TRACK            :
                              ;  +15    (1 BYTE) - RESERVED FOR FUTURE USE            :
                              ;                                                       :
                              ;        - TO DYNAMICALLY DEFINE A SET OF PARAMETERS    :
                              ;          BUILD A TABLE FOR UP TO 15 TYPES AND PLACE   :
                              ;          THE CORRESPONDING VECTOR INTO INTERRUPT 41   :
                              ;          FOR DRIVE 0 AND INTERRUPT 46 FOR DRIVE 1.    :
                              ;                                                       :
                              ;-------------------------------------------------------

                              .LIST
                              PAGE
                           C  INCLUDE SEGMENT.SRC
0000                       C  CODE SEGMENT BYTE PUBLIC
                           C
                              ;-------------------------------------------------------
                              ; HARDWARE SPECIFIC VALUES                              :
                              ;                                                       :
                              ;   -  CONTROLLER I/O PORT                              :
                              ;      > WHEN READ FROM:                                :
                              ;        HF_PORT+0 - READ DATA (FROM CONTROLLER TO CPU) :
                              ;        HF_PORT+1 - GET ERROR REGISTER                 :
                              ;        HF_PORT+2 - GET SECTOR COUNT                   :
                              ;        HF_PORT+3 - GET SECTOR NUMBER                  :
                              ;        HF_PORT+4 - GET CYLINDER LOW                   :
                              ;        HF_PORT+5 - GET CYLINDER HIGH (2 BITS)         :
                              ;        HF_PORT+6 - GET SIZE/DRIVE/HEAD                :
                              ;        HF_PORT+7 - GET STATUS REGISTER                :
                              ;      > WHEN WRITTEN TO:                               :
                              ;        HF_PORT+0 - WRITE DATA (FROM CPU TO CONTROLLER) :
                              ;        HF_PORT+1 - SET PRECOMPENSATION CYLINDER       :
                              ;        HF_PORT+2 - SET SECTOR COUNT                   :
                              ;        HF_PORT+3 - SET SECTOR NUMBER                  :
                              ;        HF_PORT+4 - SET CYLINDER LOW                   :
                              ;        HF_PORT+5 - SET CYLINDER HIGH (2 BITS)         :
                              ;        HF_PORT+6 - SET SIZE/DRIVE/HEAD                :
                              ;        HF_PORT+7 - SET COMMAND REGISTER               :
                              ;                                                       :
                              ;-------------------------------------------------------

= 01F0                        HF_PORT         EQU     01F0H           ; DISK PORT
= 03F6                        HF_REG_PORT     EQU     3F6H

                              ;       STATUS REGISTER
= 0001                        ST_ERROR        EQU     00000001B       ;
= 0002                        ST_INDEX        EQU     00000010B       ;
= 0004                        ST_CORRCTD      EQU     00000100B       ; ECC CORRECTION SUCCESSFUL
= 0008                        ST_DRQ          EQU     00001000B       ;
= 0010                        ST_SEEK_COMPL   EQU     00010000B       ; SEEK COMPLETE
= 0020                        ST_WRT_FLT      EQU     00100000B       ; WRITE FAULT
= 0040                        ST_READY        EQU     01000000B       ;
= 0080                        ST_BUSY         EQU     10000000B       ;

                              ;       ERROR REGISTER
= 0001                        ERR_DAM         EQU     00000001B       ; DATA ADDRESS MARK NOT FOUND
= 0002                        ERR_TRK_0       EQU     00000010B       ; TRACK 0 NOT FOUND ON RECAL
= 0004                        ERR_ABORT       EQU     00000100B       ; ABORTED COMMAND
                              ;               EQU     00001000B       ; NOT USED
= 0010                        ERR_ID          EQU     00010000B       ; ID NOT FOUND
                              ;               EQU     00100000B       ; NOT USED
= 0040                        ERR_DATA_ECC    EQU     01000000B       ;
= 0080                        ERR_BAD_BLOCK   EQU     10000000B       ;

= 0010                        RECAL_CMD       EQU     00010000B       ; DRIVE RECAL   (10H)
= 0020                        READ_CMD        EQU     00100000B       ;       READ    (20H)
= 0030                        WRITE_CMD       EQU     00110000B       ;       WRITE   (30H)
= 0040                        VERIFY_CMD      EQU     01000000B       ;       VERIFY  (40H)
= 0050                        FMTTRK_CMD      EQU     01010000B       ; FORMT TRACK   (50H)
= 0060                        INIT_CMD        EQU     01100000B       ;   INITIALIZE  (60H)
= 0070                        SEEK_CMD        EQU     01110000B       ;       SEEK    (70H)
= 0090                        DIAG_CMD        EQU     10010000B       ; DIAGNOSTIC    (90H)
= 0091                        SET_PARM_CMD    EQU     10010001B       ; DRIVE PARMS   (91H)
= 0001                        NO_RETRIES      EQU     00000001B       ; CMD MODIFIER (01H)
= 0002                        ECC_MODE        EQU     00000010B       ; CMD MODIFIER (02H)
= 0008                        BUFFER_MODE     EQU     00001000B       ; CMD MODIFIER (08H)

= 00A0                        INT_CTL_PORT    EQU     0A0H            ; 8259 CONTROL PORT #2
= 0020                        INTT_CTL_PORT   EQU     020H            ; 8259 CONTROL PORT #1
= 0020                        EOI             EQU     20H             ; END OF INTERRUPT COMMAND

= 0002                        MAX_FILE        EQU     2
= 0002                        S_MAX_FILE      EQU     2

= 0020                        DELAY_1         EQU     20H             ; DELAY FOR OP COMPLETE
= 0600                        DELAY_2         EQU     0600H           ; DELAY FOR READY
= 0100                        DELAY_3         EQU     0100H           ; DELAY FOR DATA REQUEST

= 0008                        HF_FAIL         EQU     08H             ; CMOS FLAG IN BYTE 0EH
                                                                      ; TO INHIBIT DISK IPL
                                              EXTRN   P_MSG:NEAR

                                      ASSUME  CS:CODE

                              PAGE
                              ;-------------------------------------------------------------------
                              ; FIXED DISK I/O SETUP                                              :
                              ;                                                                   :
```

```
                                 ;  -  ESTABLISH TRANSFER VECTORS FOR THE FIXED DISK           :
                                 ;  -  PERFORM POWER ON DIAGNOSTICS                            :
                                 ;     SHOULD AN ERROR OCCUR A "1701" MESSAGE IS DISPLAYED     :
                                 ;                                                             :
                                 ;--------------------------------------------------------------

0000                             DISK_SETUP      PROC    NEAR
                                         ASSUME  ES:ABS0
0000  2B C0                              SUB     AX,AX                           ; ZERO
0002  8E C0                              MOV     ES,AX
0004  FA                                 CLI
0005  26: A1 004C R                      MOV     AX,WORD PTR ORG_VECTOR          ; GET DISKETTE VECTOR
0009  26: A3 0100 R                      MOV     WORD PTR DISK_VECTOR,AX         ;  INTO INT 40H
000D  26: A1 004E R                      MOV     AX,WORD PTR ORG_VECTOR+2
0011  26: A3 0102 R                      MOV     WORD PTR DISK_VECTOR+2,AX
0015  26: C7 06 004C R 0197 R            MOV     WORD PTR ORG_VECTOR, OFFSET DISK_IO     ; HDISK HANDLER
001C  26: 8C 0E 004E R                   MOV     WORD PTR ORG_VECTOR+2,CS
0021  B8 06CA R                          MOV     AX, OFFSET HD_INT               ; HDISK INTERRUPT
0024  26: A3 01D8 R                      MOV     WORD PTR HDISK_INT,AX
0028  26: 8C 0E 01DA R                   MOV     WORD PTR HDISK_INT+2,CS
002D  26: C7 06 0104 R 0000 E            MOV     WORD PTR HF_TBL_VEC,OFFSET FD_TBL       ; PARM TBL DRV 80
0034  26: 8C 0E 0106 R                   MOV     WORD PTR HF_TBL_VEC+2,CS
0039  26: C7 06 0118 R 0000 E            MOV     WORD PTR HF1_TBL_VEC,OFFSET FD_TBL      ; PARM TBL DRV 81
0040  26: 8C 0E 011A R                   MOV     WORD PTR HF1_TBL_VEC+2,CS
0045  FB                                 STI                             ; ** IO DELAY NOT REQUIRED **
0046  E4 A1                              IN      AL,INT_CTL_PORT+1       ; TURN ON SECOND INTERRUPT CHIP
0048  24 BF                              AND     AL,0BFH
004A  E6 A1                              OUT     INT_CTL_PORT+1,AL
004C  E4 21                              IN      AL,INT1_CTL_PORT+1      ; LET INTERRUPTS PASS THRU TO
004E  24 FB                              AND     AL,0FBH                 ;  SECOND CHIP
0050  E6 21                              OUT     INT1_CTL_PORT+1,AL

                                         ASSUME  DS:DATA
0052  B8 ---- R                          MOV     AX,DATA                 ; ESTABLISH SEGMENT
0055  8E D8                              MOV     DS,AX
0057  C6 06 0074 R 00                    MOV     DISK_STATUS1,0          ; RESET THE STATUS INDICATOR
005C  C6 06 0075 R 00                    MOV     HF_NUM,0                ; ZERO NUMBER OF HARD FILES
0061  C6 06 0076 R 00                    MOV     CONTROL_BYTE,0
0066  B0 8E                              MOV     AL,8EH
0068  E6 70                              OUT     70H,AL                  ; CHECK CMOS VALIDITY
006A  EB 00                              JMP     SHORT $+2
006C  E4 71                              IN      AL,71H
006E  8A E0                              MOV     AH,AL                   ; SAVE CMOS FLAG
0070  24 C0                              AND     AL,0C0H
0072  75 64                              JNZ     POD_DONE                ; CMOS NOT VALID -- NO HARD FILES
0074  80 E4 F7                           AND     AH,NOT HF_FAIL          ; ALLOW HARD FILE IPL
0077  B0 8E                              MOV     AL,8EH                  ; WRITE IT BACK
0079  E6 70                              OUT     70H,AL
007B  8A C4                              MOV     AL,AH
007D  EB 00                              JMP     SHORT $+2
007F  E6 71                              OUT     71H,AL
0081  B0 92                              MOV     AL,92H
0083  E6 70                              OUT     70H,AL                  ; ACCESS HARD FILE BYTE IN CMOS
0085  EB 00                              JMP     SHORT $+2
0087  E4 71                              IN      AL,71H
0089  C6 06 0077 R 00                    MOV     PORT_OFF,0              ; ZERO CARD OFFSET
008E  8A D8                              MOV     BL,AL                   ; SAVE HARD FILE BYTE
0090  B4 00                              MOV     AH,0
0092  24 F0                              AND     AL,0F0H                 ; GET FIRST DRIVE TYPE
0094  74 42                              JZ      POD_DONE                ; NO HARD FILES
0096  05 FFF0 E                          ADD     AX,OFFSET FD_TBL-16D    ; COMPUTE OFFSET
0099  26: A3 0104 R                      MOV     WORD PTR HF_TBL_VEC,AX
009D  C6 06 0075 R 01                    MOV     HF_NUM,1                ; AT LEAST ONE DRIVE
00A2  8A C3                              MOV     AL,BL
                                         ISHL    AL,4                    ; GET SECOND DRIVE TYPE
00A4                          + ??0000   LABEL   BYTE
00A4  D0 E0                   +          SHL     AL,1
00A6                          + ??0001   LABEL   BYTE
00A4                          +          ORG     OFFSET CS:??0000
00A4  C0                      +           DB      0C0H
00A6                          +          ORG     OFFSET CS:??0001
00A6  04                      +          DB      4
00A7  74 0E                              JZ      SHORT L4                ; ONLY ONE DRIVE
00A9  B4 00                              MOV     AH,0
00AB  05 FFF0 E                          ADD     AX,OFFSET FD_TBL-16D    ; COMPUTE OFFSET FOR DRIVE 1
00AE  26: A3 0118 R                      MOV     WORD PTR HF1_TBL_VEC,AX
00B2  C6 06 0075 R 02                    MOV     HF_NUM,2                ; TWO DRIVES
00B7  B2 80                      L4:     MOV     DL,80H                  ; CHECK THE CONTROLLER
00B9  B4 14                              MOV     AH,14H
00BB  CD 13                              INT     13H
00BD  72 22                              JC      CTL_ERRX
00BF  A1 006C R                          MOV     AX,TIMER_LOW            ; GET START TIMER COUNTS
00C2  8B D8                              MOV     BX,AX
00C4  05 0444                            ADD     AX,6*182                ; 60 SECONDS * 18.2
00C7  8B C8                              MOV     CX,AX
00C9  E8 00EF R                          CALL    HD_RESET_1              ; SET UP DRIVE 0
00CC  80 3E 0075 R 01                    CMP     HF_NUM,1                ; WERE THERE TWO DRIVES?
00D1  76 05                              JBE     POD_DONE                ; NO-ALL DONE
00D3  B2 81                              MOV     DL,81H                  ; SET UP DRIVE 1
00D5  E8 00EF R                          CALL    HD_RESET_1
00D8                             POD_DONE:
00D8  FA                                 CLI                             ; ** IO DELAY NOT REQUIRED **
00D9  E4 21                              IN      AL,021H                 ; BE SURE TIMER IS ENABLED
00DB  24 FE                              AND     AL,0FEH
00DD  E6 21                              OUT     021H,AL
00DF  FB                                 STI
00E0  C3                                 RET

                                 ;----- POD ERROR

00E1                             CTL_ERRX:
00E1  BE 0000 E                          MOV     SI,OFFSET F1782         ; CONTROLLER ERROR
00E4  E8 0161 R                          CALL    SET_FAIL                ; DONT IPL FROM DISK
00E7  E8 0000 E                          CALL    P_MSG                   ; DISPLAY ERROR
00EA  BD 000F                            MOV     BP,0FH                  ; POD ERROR FLAG
00ED  EB E9                              JMP     SHORT POD_DONE

00EF                             HD_RESET_1      PROC    NEAR
00EF  53                                 PUSH    BX                      ; SAVE TIMER LIMITS
00F0  51                                 PUSH    CX
00F1  B4 09                      RES_1:  MOV     AH,09H                  ; SET DRIVE PARMS
00F3  CD 13                              INT     13H
00F5  72 06                              JC      RES_2
00F7  B4 11                              MOV     AH,11H                  ; RECALIBRATE DRIVE
00F9  CD 13                              INT     13H
00FB  73 15                              JNC     RES_CK                  ; DRIVE OK
00FD  E8 0178 R                  RES_2:  CALL    POD_TCHK                ; CHECK TIME OUT
0100  73 EF                              JNC     RES_1
0102  BE 0000 E                  RES_FL: MOV     SI,OFFSET F1781         ; INDICATE DISK 1 FAILURE
0105  F6 C2 01                           TEST    DL,1
0108  75 4E                              JNZ     RES_E1
010A  BE 0000 E                          MOV     SI,OFFSET F1780         ; INDICATE DISK 0 FAILURE
010D  E8 0161 R                          CALL    SET_FAIL                ; DONT TRY TO IPL DISK 0
0110  EB 46                              JMP     SHORT RES_E1
0112  B4 08                      RES_CK: MOV     AH,08H                  ; GET MAX CYL,HEAD,SECTOR
```

SECTION 5

```
0114 8A DA                       MOV    BL,DL                 ; SAVE DRIVE CODE
0116 CD 13                       INT    13H
0118 72 33                       JC     RES_ER
011A 8A D3                       MOV    DL,BL                 ; RESTORE DRIVE CODE
011C B8 0401            RES_3:   MOV    AX,0401H              ; VERIFY THE LAST SECTOR
011F CD 13                       INT    13H
0121 73 3B                       JNC    RES_OK                ; VERIFY OK
0123 80 FC 0A                    CMP    AH,BAD_SECTOR         ; OK ALSO IF JUST ID READ
0126 74 36                       JE     RES_OK
0128 80 FC 11                    CMP    AH,DATA_CORRECTED
012B 74 31                       JE     RES_OK
012D 80 FC 10                    CMP    AH,BAD_ECC
0130 74 2C                       JE     RES_OK
0132 E8 0178 R                   CALL   POD_TCHK              ; CHECK FOR TIME OUT
0135 72 16                       JC     RES_ER                ; FAILED
0137 A0 0044 R                   MOV    AL,CMD_BLOCK+2        ; GET SECTOR ADDRESS
013A FE C8                       DEC    AL                    ; TRY PREVIOUS ONE
013C 74 D4                       JZ     RES_CK                ; WE'VE TRIED ALL SECTORS ON TRACK
013E 8A 2E 0045 R                MOV    CH,CMD_BLOCK+3        ; GET CYLINDER
0142 8A 0E 0046 R                MOV    CL,CMD_BLOCK+4        ;  NUMBER
                                 ISHL   CL,6                  ; MOVE THE BITS UP
0146                   + ??0003  LABEL  BYTE
0146 D0 E1             +         SHL    CL,1
0148                   + ??0004  LABEL  BYTE
0146                   +         ORG    OFFSET CS:??0003
0146 C0                +          DB     0C0H
0148                   +         ORG    OFFSET CS:??0004
0148 06                +         DB     6
0149 0A C8                       OR     CL,AL                 ; PUT SECTOR NUMBER IN PLACE
014B EB CF                       JMP    RES_3                 ; TRY AGAIN
014D BE 0000 E          RES_ER:  MOV    SI,OFFSET F1791       ; INDICATE DISK 1 ERROR
0150 F6 C2 01                    TEST   DL,1
0153 75 03                       JNZ    RES_E1
0155 BE 0000 E                   MOV    SI,OFFSET F1790       ; INDICATE DISK 0 ERROR
0158 E8 0000 E          RES_E1:  CALL   P_MSG
015B BD 000F                     MOV    BP,0FH
015E 59                 RES_OK:  POP    CX                    ; RESTORE TIMER LIMITS
015F 5B                          POP    BX
0160 C3                          RET
0161                    HD_RESET_1      ENDP

0161                    SET_FAIL        PROC    NEAR
0161 B0 8E                       MOV    AL,8EH                ; GET CMOS ERROR BYTE
0163 E6 70                       OUT    70H,AL
0165 EB 00                       JMP    SHORT $+2
0167 E4 71                       IN     AL,71H
0169 0C 08                       OR     AL,HF_FAIL            ; SET DONT IPL FROM DISK FLAG
016B 8A E0                       MOV    AH,AL                 ; SAVE IT
016D B0 8E                       MOV    AL,8EH                ; CMOS BYTE ADDRESS
016F E6 70                       OUT    70H,AL
0171 8A C4                       MOV    AL,AH
0173 EB 00                       JMP    SHORT $+2
0175 E6 71                       OUT    71H,AL                ; PUT IT OUT
0177 C3                          RET
0178                    SET_FAIL        ENDP

0178                    POD_TCHK        PROC NEAR             ; CHECK FOR 30 SECOND TIME OUT
0178 58                          POP    AX                    ; SAVE RETURN
0179 59                          POP    CX                    ; GET TIME OUT LIMITS
017A 5B                          POP    BX
017B 53                          PUSH   BX                    ; AND SAVE THEM AGAIN
017C 51                          PUSH   CX
017D 50                          PUSH   AX                    ; RESTORE RETURN
017E A1 006C R                   MOV    AX,TIMER_LOW          ; AX = CURRENT TIME
                                                              ; BX = START TIME
                                                              ; CX = END TIME
0181 3B D9                       CMP    BX,CX
0183 72 06                       JB     TCHK1                 ; START < END
0185 3B D8                       CMP    BX,AX
0187 72 0C                       JB     TCHKG                 ; END < START < CURRENT
0189 EB 04                       JMP    SHORT TCHK2           ; END, CURRENT < START
018B 3B C3              TCHK1:   CMP    AX,BX
018D 72 04                       JB     TCHKNG                ; CURRENT < START < END
018F 3B C1              TCHK2:   CMP    AX,CX
0191 72 02                       JB     TCHKG                 ; START < CURRENT < END
                                                              ; OR CURRENT < END < START
0193 F9                 TCHKNG:  STC                          ; CARRY SET INDICATES TIME OUT
0194 C3                          RET
0195 F8                 TCHKG:   CLC                          ; INDICATE STILL TIME
0196 C3                          RET
0197                    POD_TCHK        ENDP

0197                    DISK_SETUP      ENDP
                        PAGE
                        ;----------------------------------------
                        ;          FIXED DISK BIOS ENTRY POINT   :
                        ;----------------------------------------

0197                    DISK_IO PROC    FAR
                                 ASSUME DS:NOTHING,ES:NOTHING
0197 80 FA 80                    CMP    DL,80H                ; TEST FOR FIXED DISK DRIVE
019A 73 05                       JAE    HARD_DISK             ; YES, HANDLE HERE
019C CD 40                       INT    40H                   ; DISKETTE HANDLER
019E                    RET_2:
019E CA 0002                     RET    2                     ; BACK TO CALLER
01A1                    HARD_DISK:
                                 ASSUME DS:DATA
01A1 FB                          STI                          ; ENABLE INTERRUPTS
01A2 0A E4                       OR     AH,AH
01A4 75 09                       JNZ    A2
01A6 CD 40                       INT    40H                   ; RESET NEC WHEN AH=0
01A8 2A E4                       SUB    AH,AH
01AA 80 FA 81                    CMP    DL,(80H + S_MAX_FILE - 1)
01AD 77 EF                       JA     RET_2
01AF                    A2:
01AF 80 FC 08                    CMP    AH,08H                ; GET PARAMETERS IS A SPECIAL CASE
01B2 75 03                       JNZ    A3
01B4 E9 038B R                   JMP    GET_PARM_N
01B7 80 FC 15           A3:      CMP    AH,15H                ; READ DASD TYPE IS ALSO
01BA 75 03                       JNZ    A4
01BC E9 0349 R                   JMP    READ_DASD_TYPE
01BF                    A4:
01BF 53                          PUSH   BX                    ; SAVE REGISTERS DURING OPERATION
01C0 51                          PUSH   CX
01C1 52                          PUSH   DX
01C2 1E                          PUSH   DS
01C3 06                          PUSH   ES
01C4 56                          PUSH   SI
01C5 57                          PUSH   DI
01C6 0A E4                       OR     AH,AH                 ; CHECK FOR RESET
01C8 75 02                       JNZ    A5
01CA B2 80                       MOV    DL,80H                ; FORCE DRIVE 80 FOR RESET
01CC E8 0212 R          A5:      CALL   DISK_IO_CONT          ; PERFORM THE OPERATION
01CF 50                          PUSH   AX
01D0 B8 ---- R                   MOV    AX,DATA
01D3 8E D8                       MOV    DS,AX                 ; ESTABLISH SEGMENT
```

```
01D5  58                                 POP    AX
01D6  8A 26 0074 R                       MOV    AH,DISK_STATUS1          ; GET STATUS FROM OPERATION
01DA  80 FC 01                           CMP    AH,1                     ; SET THE CARRY FLAG TO INDICATE
01DD  F5                                 CMC                             ;  SUCCESS OR FAILURE
01DE  5F                                 POP    DI                       ; RESTORE REGISTERS
01DF  5E                                 POP    SI
01E0  07                                 POP    ES
01E1  1F                                 POP    DS
01E2  5A                                 POP    DX
01E3  59                                 POP    CX
01E4  5B                                 POP    BX
01E5  CA 0002                            RET    2                        ; THROW AWAY SAVED FLAGS
01E8                         DISK_IO     ENDP

01E8                         M1          LABEL  WORD                     ; FUNCTION TRANSFER TABLE
01E8  02B3 R                             DW     DISK_RESET               ; 000H
01EA  0307 R                             DW     RETURN_STATUS            ; 001H
01EC  0310 R                             DW     DISK_READ                ; 002H
01EE  0318 R                             DW     DISK_WRITE               ; 003H
01F0  0320 R                             DW     DISK_VERF                ; 004H
01F2  0333 R                             DW     FMT_TRK                  ; 005H
01F4  02AB R                             DW     BAD_COMMAND              ; 006H  FORMAT BAD SECTORS
01F6  02AB R                             DW     BAD_COMMAND              ; 007H  FORMAT DRIVE
01F8  02AB R                             DW     BAD_COMMAND              ; 008H  RETURN PARMS
01FA  03EA R                             DW     INIT_DRV                 ; 009H
01FC  041F R                             DW     RD_LONG                  ; 00AH
01FE  0427 R                             DW     WR_LONG                  ; 00BH
0200  042F R                             DW     DISK_SEEK                ; 00CH
0202  02B3 R                             DW     DISK_RESET               ; 00DH
0204  02AB R                             DW     BAD_COMMAND              ; 00EH  READ BUFFER
0206  02AB R                             DW     BAD_COMMAND              ; 00FH  WRITE BUFFER
0208  044E R                             DW     TST_RDY                  ; 010H
020A  0465 R                             DW     HDISK_RECAL              ; 011H
020C  02AB R                             DW     BAD_COMMAND              ; 012H  RAM DIAGNOSTIC
020E  02AB R                             DW     BAD_COMMAND              ; 013H  DRIVE DIAGNOSTIC
0210  0489 R                             DW     CTLR_DIAGNOSTIC          ; 014H  CONTROLLER DIAGNOSTIC
= 002A                       M1L         EQU    $-M1

0212                         DISK_IO_CONT       PROC    NEAR
0212  50                                 PUSH   AX
0213  B8 ---- R                          MOV    AX,DATA
0216  8E D8                              MOV    DS,AX                    ; ESTABLISH SEGMENT
0218  58                                 POP    AX
0219  80 FC 01                           CMP    AH,01H                   ; RETURN STATUS
021C  75 03                              JNZ    SU0
021E  E9 0307 R                          JMP    RETURN_STATUS
0221                         SU0:
0221  C6 06 0074 R 00                    MOV    DISK_STATUS1,0           ; RESET THE STATUS INDICATOR
0226  53                                 PUSH   BX                       ; SAVE DATA ADDRESS
0227  8A 1E 0075 R                       MOV    BL,HF_NUM                ; GET NUMBER OF DRIVES
022B  50                                 PUSH   AX
022C  80 E2 7F                           AND    DL,7FH                   ; GET DRIVE AS 0 OR 1
022F  3A DA                              CMP    BL,DL
0231  76 76                              JBE    BAD_COMMAND_POP          ; INVALID DRIVE
0233  06                                 PUSH   ES
0234  E8 06B4 R                          CALL   GET_VEC                  ; GET DISK PARMS
0237  26: 8B 47 05                       MOV    AX,WORD PTR ES:[BX][5]   ; GET WRITE PRE-COMP CYL
                                         ISHR   AX,2
023B                       + ??0006      LABEL  BYTE
023B  D1 E8                +             SHR    AX,1
023D                       + ??0007      LABEL  BYTE
023B                       +             ORG    OFFSET CS:??0006
023B                       + ??0008        LABEL   NEAR
023B  C1                   +               DB     0C1H
023D                       +             ORG    OFFSET CS:??0007
023D  02                   +             DB     2
023E  A2 0042 R                          MOV    CMD_BLOCK,AL
0241  26: 8A 47 08                       MOV    AL,BYTE PTR ES:[BX][8]   ; GET CONTROL BYTE MODIFIER
0245  52                                 PUSH   DX
0246  BA 03F6                            MOV    DX,HF_REG_PORT
0249  EE                                 OUT    DX,AL                    ; SET EXTRA HEAD OPTION
024A  5A                                 POP    DX
024B  07                                 POP    ES
024C  8A 26 0076 R                       MOV    AH,CONTROL_BYTE          ; SET EXTRA HEAD OPTION IN
0250  80 E4 C0                           AND    AH,0C0H                  ; CONTROL BYTE
0253  0A E0                              OR     AH,AL
0255  88 26 0076 R                       MOV    CONTROL_BYTE,AH
0259  58                                 POP    AX
025A  A2 0043 R                          MOV    CMD_BLOCK+1,AL           ; SECTOR COUNT
025D  50                                 PUSH   AX
025E  8A C1                              MOV    AL,CL                    ; GET SECTOR NUMBER
0260  24 3F                              AND    AL,3FH
0262  A2 0044 R                          MOV    CMD_BLOCK+2,AL
0265  88 2E 0045 R                       MOV    CMD_BLOCK+3,CH           ; GET CYLINDER NUMBER
0269  8A C1                              MOV    AL,CL
                                         ISHR   AL,6
026B                       + ??0009      LABEL  BYTE
026B  D0 E8                +             SHR    AL,1
026D                       + ??000A      LABEL  BYTE
026B                       +             ORG    OFFSET CS:??0009
026B  C0                   +               DB     0C0H
026D                       +             ORG    OFFSET CS:??000A
026D  06                   +             DB     6
026E  A2 0046 R                          MOV    CMD_BLOCK+4,AL           ; CYLINDER HIGH ORDER 2 BITS
0271  8A C2                              MOV    AL,DL                    ; DRIVE NUMBER
                                         ISHL   AL,4
0273                       + ??000C      LABEL  BYTE
0273  D0 E0                +             SHL    AL,1
0275                       + ??000D      LABEL  BYTE
0273                       +             ORG    OFFSET CS:??000C
0273  C0                   +               DB     0C0H
0275                       +             ORG    OFFSET CS:??000D
0275  04                   +             DB     4
0276  80 E6 0F                           AND    DH,0FH                   ; HEAD NUMBER
0279  0A C6                              OR     AL,DH
027B  0C A0                              OR     AL,80H OR 20H            ; ECC AND 512 BYTE SECTORS
027D  A2 0047 R                          MOV    CMD_BLOCK+5,AL           ; ECC/SIZE/DRIVE/HEAD
0280  58                                 POP    AX
0281  50                                 PUSH   AX
0282  8A C4                              MOV    AL,AH                    ; GET INTO LOW BYTE
0284  32 E4                              XOR    AH,AH                    ; ZERO HIGH BYTE
0286  D1 E0                              SAL    AX,1                     ; *2 FOR TABLE LOOKUP
0288  8B F0                              MOV    SI,AX                    ; PUT INTO SI FOR BRANCH
028A  3D 002A                            CMP    AX,M1L                   ; TEST WITHIN RANGE
028D  73 1A                              JNB    BAD_COMMAND_POP
028F  58                                 POP    AX                       ; RESTORE AX
0290  5B                                 POP    BX                       ; AND DATA ADDRESS
0291  51                                 PUSH   CX
0292  50                                 PUSH   AX                       ; ADJUST ES:BX
0293  8B CB                              MOV    CX,BX                    ; GET 3 HIGH ORDER NYBBLES OF BX
                                         ISHR   CX,4
0295                       + ??000F      LABEL  BYTE
0295  D1 E9                +             SHR    CX,1
0297                       + ??0010      LABEL  BYTE
0295                       +             ORG    OFFSET CS:??000F
0295                       + ??0011        LABEL   NEAR
```

SECTION 5

```
0295 C1                    +          DB     0C1H
0297                       +          ORG    OFFSET CS:??0010
0297 04                    +          DB     4
0298 8C C0                            MOV    AX,ES
029A 03 C1                            ADD    AX,CX
029C 8E C0                            MOV    ES,AX
029E 81 E3 000F                       AND    BX,000FH             ; ES:BX CHANGED TO ES:000X
02A2 58                               POP    AX
02A3 59                               POP    CX
02A4 2E: FF A4 01E8 R                 JMP    WORD PTR CS:[SI + OFFSET M1]
02A9                         BAD_COMMAND_POP:
02A9 58                               POP    AX
02AA 5B                               POP    BX
02AB                         BAD_COMMAND:
02AB C6 06 0074 R 01                  MOV    DISK_STATUS1,BAD_CMD     ; COMMAND ERROR
02B0 B0 00                            MOV    AL,0
02B2 C3                               RET
02B3                         DISK_IO_CONT    ENDP

                             ;-----------------------------------------------
                             ;         RESET THE DISK SYSTEM  (AH = 000H)    :
                             ;-----------------------------------------------

02B3                         DISK_RESET      PROC    NEAR
02B3 FA                               CLI                         ; ** IO DELAY NOT REQUIRED **
02B4 E4 A1                            IN     AL,INT_CTL_PORT+1    ; GET THE MASK REG
02B6 24 BF                            AND    AL,0BFH              ; ENABLE HARD FILE INT.
02B8 E6 A1                            OUT    INT_CTL_PORT+1,AL
02BA FB                               STI                         ; START INTERRUPTS
02BB B0 04                            MOV    AL,04H
02BD BA 03F6                          MOV    DX,HF_REG_PORT
02C0 EE                               OUT    DX,AL                ; RESET
02C1 B9 000A                          MOV    CX,10                ; DELAY COUNT
02C4 49                      DRD:     DEC    CX
02C5 75 FD                            JNZ    DRD                  ; WAIT 4.8 MICRO-SEC
02C7 A0 0076 R                        MOV    AL,CONTROL_BYTE
02CA 24 0F                            AND    AL,0FH               ; SET HEAD OPTION
02CC EE                               OUT    DX,AL                ; TURN RESET OFF
02CD E8 05DF R                        CALL   NOT_BUSY
02D0 75 2F                            JNZ    DRERR                ; TIME OUT ON RESET
02D2 BA 01F1                          MOV    DX,HF_PORT+1
02D5 EC                               IN     AL,DX                ; GET RESET STATUS
02D6 3C 01                            CMP    AL,1
02D8 75 27                            JNZ    DRERR                ; BAD RESET STATUS
02DA 80 26 0047 R EF                  AND    CMD_BLOCK+5,0EFH     ; SET TO DRIVE 0
02DF 2A D2                            SUB    DL,DL
02E1 E8 03EA R                        CALL   INIT_DRV             ; SET MAX HEADS
02E4 E8 0465 R                        CALL   HDISK_RECAL          ; RECAL TO RESET SEEK SPEED
02E7 80 3E 0075 R 01                  CMP    HF_NUM,1             ; CHECK FOR DRIVE 1
02EC 76 0D                            JBE    DRE
02EE 80 0E 0047 R 10                  OR     CMD_BLOCK+5,010H     ; SET TO DRIVE 1
02F3 B2 01                            MOV    DL,1
02F5 E8 03EA R                        CALL   INIT_DRV             ; SET MAX HEADS
02F8 E8 0465 R                        CALL   HDISK_RECAL          ; RECAL TO RESET SEEK SPEED
02FB C6 06 0074 R 00         DRE:     MOV    DISK_STATUS1,0       ; IGNORE ANY SET UP ERRORS
0300 C3                               RET
0301 C6 06 0074 R 05         DRERR:   MOV    DISK_STATUS1,BAD_RESET  ; CARD FAILED
0306 C3                               RET
0307                         DISK_RESET      ENDP

                             ;-----------------------------------------------
                             ;         DISK STATUS ROUTINE   (AH = 001H)     :
                             ;-----------------------------------------------
0307                         RETURN_STATUS   PROC    NEAR
0307 A0 0074 R                        MOV    AL,DISK_STATUS1         ; OBTAIN PREVIOUS STATUS
030A C6 06 0074 R 00                  MOV    DISK_STATUS1,0          ; RESET STATUS
030F C3                               RET
0310                         RETURN_STATUS   ENDP

                             ;-----------------------------------------------
                             ;         DISK READ ROUTINE    (AH = 002H)      :
                             ;-----------------------------------------------

0310                         DISK_READ       PROC    NEAR
0310 C6 06 0048 R 20                  MOV    CMD_BLOCK+6,READ_CMD
0315 E9 04BB R                        JMP    COMMANDI
0318                         DISK_READ       ENDP

                             ;-----------------------------------------------
                             ;         DISK WRITE ROUTINE    (AH = 003H)     :
                             ;-----------------------------------------------

0318                         DISK_WRITE      PROC    NEAR
0318 C6 06 0048 R 30                  MOV    CMD_BLOCK+6,WRITE_CMD
031D E9 04FB R                        JMP    COMMANDO
0320                         DISK_WRITE      ENDP

                             ;-----------------------------------------------
                             ;         DISK VERIFY    (AH = 004H)            :
                             ;-----------------------------------------------

0320                         DISK_VERF       PROC    NEAR
0320 C6 06 0048 R 40                  MOV    CMD_BLOCK+6,VERIFY_CMD
0325 E8 0544 R                        CALL   COMMAND
0328 75 08                            JNZ    VERF_EXIT       ; CONTROLLER STILL BUSY
032A E8 05A5 R                        CALL   WAIT
032D 75 03                            JNZ    VERF_EXIT       ; TIME OUT
032F E8 061E R                        CALL   CHECK_STATUS
0332                         VERF_EXIT:
0332 C3                               RET
0333                         DISK_VERF       ENDP

                             ;-----------------------------------------------
                             ;         FORMATTING   (AH = 005H )            :
                             ;-----------------------------------------------

0333                         FMT_TRK PROC    NEAR                 ; FORMAT TRACK  (AH = 005H)
0333 C6 06 0048 R 50                  MOV    CMD_BLOCK+6,FMTTRK_CMD
0338 06                               PUSH   ES
0339 53                               PUSH   BX
033A E8 06B4 R                        CALL   GET_VEC              ; GET DISK PARMS ADDRESS
033D 26: 8A 47 0E                     MOV    AL,ES:[BX][14]       ; GET SECTORS/TRACK
0341 A2 0043 R                        MOV    CMD_BLOCK+1,AL       ; SET SECTOR COUNT IN COMMAND
0344 5B                               POP    BX
0345 07                               POP    ES
0346 E9 0500 R                        JMP    CMD_OF               ; GO EXECUTE THE COMMAND
0349                         FMT_TRK ENDP
                             PAGE
                             ;-----------------------------------------------
                             ;         READ DASD TYPE    (AH = 15H)          :
                             ;-----------------------------------------------

0349                         READ_DASD_TYPE  LABEL   NEAR
0349                         READ_D_T        PROC    FAR          ; GET DRIVE PARAMETERS
0349 1E                               PUSH   DS                   ; SAVE REGISTERS
```

```
034A  06                              PUSH    ES
034B  53                              PUSH    BX
034C  B8  ---- R                      MOV     AX,DATA               ; ESTABLISH ADDRESSING
034F  8E D8                           MOV     DS,AX
                                      ASSUME  DS:DATA
0351  C6 06 0074 R 00                 MOV     DISK_STATUS1,0
0356  8A 1E 0075 R                    MOV     BL,HF_NUM             ; GET NUMBER OF DRIVES
035A  80 E2 7F                        AND     DL,7FH                ; GET DRIVE NUMBER
035D  3A DA                           CMP     BL,DL
035F  76 22                           JBE     RDT_NOT_PRESENT       ; RETURN DRIVE NOT PRESENT
0361  E8 06B4 R                       CALL    GET_VEC               ; GET DISK PARM ADDRESS
0364  26: 8A 47 02                    MOV     AL,ES:[BX][2]         ; HEADS
0368  26: 8A 4F 0E                    MOV     CL,ES:[BX][14]
036C  F6 E9                           IMUL    CL                    ; * NUMBER OF SECTORS
036E  26: 8B 0F                       MOV     CX,ES:[BX]            ; MAX NUMBER OF CYLINDERS
0371  49                              DEC     CX                    ; LEAVE ONE FOR DIAGNOSTICS
0372  F7 E9                           IMUL    CX                    ; NUMBER OF SECTORS
0374  8B CA                           MOV     CX,DX                 ; HIGH ORDER HALF
0376  8B D0                           MOV     DX,AX                 ; LOW ORDER HALF
0378  2B C0                           SUB     AX,AX
037A  B4 03                           MOV     AH,03H                ; INDICATE FIXED DISK
037C  5B                    RDT2:     POP     BX                    ; RESTORE REGS
037D  07                              POP     ES
037E  1F                              POP     DS
037F  F8                              CLC                           ; CLEAR CARRY
0380  CA 0002                         RET     2
0383                        RDT_NOT_PRESENT:
0383  2B C0                           SUB     AX,AX                 ; DRIVE NOT PRESENT RETURN
0385  8B C8                           MOV     CX,AX                 ; ZERO BLOCK COUNT
0387  8B D0                           MOV     DX,AX
0389  EB F1                           JMP     RDT2
038B                        READ_D_T          ENDP
                            PAGE
                            ;--------------------------------------------------
                            ;         GET PARAMETERS    (AH = 8)              :
                            ;--------------------------------------------------

038B                        GET_PARM_N        LABEL   NEAR
038B                        GET_PARM          PROC    FAR           ; GET DRIVE PARAMETERS
038B  1E                              PUSH    DS                    ; SAVE REGISTERS
038C  06                              PUSH    ES
038D  53                              PUSH    BX
                                      ASSUME  DS:ABS0
038E  2B C0                           SUB     AX,AX                 ; ESTABLISH ADDRESSING
0390  8E D8                           MOV     DS,AX
0392  F6 C2 01                        TEST    DL,1                  ; CHECK FOR DRIVE 1
0395  74 06                           JZ      G0
0397  C4 1E 0118 R                    LES     BX,HF1_TBL_VEC
039B  EB 04                           JMP     SHORT G1
039D  C4 1E 0104 R          G0:       LES     BX,HF_TBL_VEC
                                      ASSUME  DS:DATA
03A1  B8  ---- R            G1:       MOV     AX,DATA
03A4  8E D8                           MOV     DS,AX                 ; ESTABLISH SEGMENT
03A6  80 EA 80                        SUB     DL,80H
03A9  80 FA 02                        CMP     DL,MAX_FILE           ; TEST WITHIN RANGE
03AC  73 2C                           JAE     G4
03AE  C6 06 0074 R 00                 MOV     DISK_STATUS1,0
03B3  26: 8B 07                       MOV     AX,ES:[BX]            ; MAX NUMBER OF CYLINDERS
03B6  2D 0002                         SUB     AX,2                  ; ADJUST FOR 0-N
03B9  8A E8                           MOV     CH,AL
03BB  25 0300                         AND     AX,0300H              ; HIGH TWO BITS OF CYL
03BE  D1 E8                           SHR     AX,1
03C0  D1 E8                           SHR     AX,1
03C2  26: 0A 47 0E                    OR      AL,ES:[BX][14]        ; SECTORS
03C6  8A C8                           MOV     CL,AL
03C8  26: 8A 77 02                    MOV     DH,ES:[BX][2]         ; HEADS
03CC  FE CE                           DEC     DH                    ; 0-N RANGE
03CE  8A 16 0075 R                    MOV     DL,HF_NUM             ; DRIVE COUNT
03D2  2B C0                           SUB     AX,AX
03D4                        G5:
03D4  5B                              POP     BX                    ; RESTORE REGISTERS
03D5  07                              POP     ES
03D6  1F                              POP     DS
03D7  CA 0002                         RET     2
03DA                        G4:
03DA  C6 06 0074 R 07                 MOV     DISK_STATUS1,INIT_FAIL  ; OPERATION FAILED
03DF  B4 07                           MOV     AH,INIT_FAIL
03E1  2A C0                           SUB     AL,AL
03E3  2B D2                           SUB     DX,DX
03E5  2B C9                           SUB     CX,CX
03E7  F9                              STC                           ; SET ERROR FLAG
03E8  EB EA                           JMP     G5
03EA                        GET_PARM          ENDP
                                      PAGE
                            ;----------------------------------------
                            ;  INITIALIZE DRIVE
                            ;----------------------------------------
03EA                        INIT_DRV          PROC    NEAR
03EA  C6 06 0048 R 91                 MOV     CMD_BLOCK+6,SET_PARM_CMD
03EF  E8 06B4 R                       CALL    GET_VEC               ; ES:BX -> PARM BLOCK
03F2  26: 8A 47 02                    MOV     AL,ES:[BX][2]         ; GET NUMBER OF HEADS
03F6  FE C8                           DEC     AL                    ; CONVERT TO 0-INDEX
03F8  8A 26 0047 R                    MOV     AH,CMD_BLOCK+5        ; GET SDH REGISTER
03FC  80 E4 F0                        AND     AH,0F0H               ; CHANGE HEAD NUMBER
03FF  0A E0                           OR      AH,AL                 ;  TO MAX HEAD
0401  88 26 0047 R                    MOV     CMD_BLOCK+5,AH
0405  26: 8A 47 0E                    MOV     AL,ES:[BX][14]        ; MAX SECTOR NUMBER
0409  A2 0043 R                       MOV     CMD_BLOCK+1,AL
040C  2B C0                           SUB     AX,AX
040E  A2 0045 R                       MOV     CMD_BLOCK+3,AL        ; ZERO FLAGS
0411  E8 0544 R                       CALL    COMMAND               ; TELL CONTROLLER
0414  75 08                           JNZ     INIT_EXIT             ; CONTROLLER BUSY ERROR
0416  E8 05DF R                       CALL    NOT_BUSY              ; WAIT FOR IT TO BE DONE
0419  75 03                           JNZ     INIT_EXIT             ; TIME OUT
041B  E8 061E R                       CALL    CHECK_STATUS
041E                        INIT_EXIT:
041E  C3                              RET
041F                        INIT_DRV          ENDP

                            ;----------------------------------------
                            ;         READ LONG   (AH = 0AH)        :
                            ;----------------------------------------

041F                        RD_LONG           PROC    NEAR
041F  C6 06 0048 R 22                 MOV     CMD_BLOCK+6,READ_CMD OR ECC_MODE
0424  E9 04BB R                       JMP     COMMANDI
0427                        RD_LONG           ENDP

                            ;----------------------------------------
                            ;         WRITE LONG   (AH = 0BH)       :
                            ;----------------------------------------

0427                        WR_LONG           PROC    NEAR
0427  C6 06 0048 R 32                 MOV     CMD_BLOCK+6,WRITE_CMD OR ECC_MODE
042C  E9 04FB R                       JMP     COMMANDO
042F                        WR_LONG           ENDP
```

SECTION 5

```
                                ;---------------------------------------
                                ;       SEEK    (AH = 0CH)             :
                                ;---------------------------------------

042F                            DISK_SEEK       PROC    NEAR
042F  C6 06 0048 R 70                   MOV     CMD_BLOCK+6,SEEK_CMD
0434  E8 0544 R                         CALL    COMMAND
0437  75 14                             JNZ     DS_EXIT                 ; CONTROLLER BUSY ERROR
0439  E8 05A5 R                         CALL    WAIT
043C  75 0F                             JNZ     DS_EXIT                 ; TIME OUT ON SEEK
043E  E8 061E R                         CALL    CHECK_STATUS
0441  80 3E 0074 R 40                   CMP     DISK_STATUS1,BAD_SEEK
0446  75 05                             JNE     DS_EXIT
0448  C6 06 0074 R 00                   MOV     DISK_STATUS1,0
044D                            DS_EXIT:
044D  C3                                RET
044E                            DISK_SEEK       ENDP

                                ;-----------------------------------------------
                                ;       TEST DISK READY   (AH = 010H)          :
                                ;-----------------------------------------------

044E                            TST_RDY PROC    NEAR
044E  E8 05DF R                         CALL    NOT_BUSY                ; WAIT FOR CONTROLLER
0451  75 11                             JNZ     TR_EX
0453  A0 0047 R                         MOV     AL,CMD_BLOCK+5          ; SELECT DRIVE
0456  BA 01F6                           MOV     DX,HF_PORT+6
0459  EE                                OUT     DX,AL
045A  E8 0630 R                         CALL    CHECK_ST                ; CHECK STATUS ONLY
045D  75 05                             JNZ     TR_EX
045F  C6 06 0074 R 00                   MOV     DISK_STATUS1,0          ; WIPE OUT DATA CORRECTED ERROR
0464  C3                        TR_EX:  RET
0465                            TST_RDY ENDP

                                ;-----------------------------------------------
                                ;       RECALIBRATE    (AH = 011H)             :
                                ;-----------------------------------------------

0465                            HDISK_RECAL     PROC    NEAR
0465  C6 06 0048 R 10                   MOV     CMD_BLOCK+6,RECAL_CMD
046A  E8 0544 R                         CALL    COMMAND                 ; START THE OPERATION
046D  75 14                             JNZ     RECAL_EXIT              ; ERROR
046F  E8 05A5 R                         CALL    WAIT                    ; WAIT FOR COMPLETION
0472  75 0F                             JNZ     RECAL_EXIT              ; TIME OUT
0474  E8 061E R                         CALL    CHECK_STATUS
0477  80 3E 0074 R 40                   CMP     DISK_STATUS1,BAD_SEEK   ; SEEK NOT COMPLETE
047C  75 05                             JNE     RECAL_EXIT              ; IS OK
047E  C6 06 0074 R 00                   MOV     DISK_STATUS1,0
0483                            RECAL_EXIT:
0483  80 3E 0074 R 00                   CMP     DISK_STATUS1,0
0488  C3                                RET
0489                            HDISK_RECAL     ENDP

                                ;---------------------------------------
                                ; CONTROLLER DIAGNOSTIC (AH = 14H)     :
                                ;---------------------------------------

0489                            CTLR_DIAGNOSTIC PROC    NEAR            ; ** IO DELAY NOT REQUIRED **
0489  E4 A1                             IN      AL,INT_CTL_PORT+1       ; TURN ON SECOND INTERRUPT CHIP
048B  24 BF                             AND     AL,0BFH
048D  E6 A1                             OUT     INT_CTL_PORT+1,AL
048F  E4 21                             IN      AL,INT1_CTL_PORT+1      ; LET INTERRUPTS PASS THRU TO
0491  24 FB                             AND     AL,0FBH                 ;  SECOND CHIP
0493  E6 21                             OUT     INT1_CTL_PORT+1,AL
0495  E8 05DF R                         CALL    NOT_BUSY                ; WAIT FOR CARD
0498  75 1A                             JNZ     CD_ERR                  ; BAD CARD
049A  BA 01F7                           MOV     DX,HF_PORT+7
049D  B0 90                             MOV     AL,DIAG_CMD             ; START DIAGNOSE
049F  EE                                OUT     DX,AL
04A0  E8 05DF R                         CALL    NOT_BUSY                ; WAIT FOR IT TO COMPLETE
04A3  B4 80                             MOV     AH,TIME_OUT
04A5  75 0F                             JNZ     CD_EXIT                 ; TIME OUT ON DIAGNOSTIC
04A7  BA 01F1                           MOV     DX,HF_PORT+1            ; GET ERROR REGISTER
04AA  EC                                IN      AL,DX
04AB  A2 008D R                         MOV     HF_ERROR,AL             ; SAVE IT
04AE  B4 00                             MOV     AH,0
04B0  3C 01                             CMP     AL,1                    ; CHECK FOR ALL OK
04B2  74 02                             JE      SHORT CD_EXIT
04B4  B4 20                     CD_ERR: MOV     AH,BAD_CNTLR
04B6                            CD_EXIT:
04B6  88 26 0074 R                      MOV     DISK_STATUS1,AH
04BA  C3                                RET
04BB                            CTLR_DIAGNOSTIC ENDP

                                ;-----------------------------------------------
                                ; COMMANDI                                     :
                                ;       REPEATEDLY INPUTS DATA TIL NSECTOR     :
                                ;       RETURNS ZERO                           :
                                ;-----------------------------------------------
04BB                            COMMANDI:
04BB  E8 068F R                         CALL    CHECK_DMA               ; CHECK 64K BOUNDARY ERROR
04BE  72 3A                             JC      CMD_ABORT
04C0  8B FB                             MOV     DI,BX
04C2  E8 0544 R                         CALL    COMMAND                 ; OUTPUT COMMAND
04C5  75 33                             JNZ     CMD_ABORT
04C7                            CMD_I1:
04C7  E8 05A5 R                         CALL    WAIT                    ; WAIT FOR DATA REQUEST INT
04CA  75 2E                             JNZ     TM_OUT                  ; TIME OUT
04CC  B9 0100                           MOV     CX,256D                 ; SECTOR SIZE IN WORDS
04CF  BA 01F0                           MOV     DX,HF_PORT
04D2  FC                                CLD
                                        REP_INSW                        ; GET THE SECTOR
04D3  F3 6D                   +                 DB      0F3H,06DH
04D5  F6 06 0048 R 02                   TEST    CMD_BLOCK+6,ECC_MODE        ; CHECK FOR NORMAL INPUT
04DA  74 12                             JZ      CMD_I3
04DC  E8 0608 R                         CALL    WAIT_DRQ                ; WAIT FOR DATA REQUEST
04DF  72 19                             JC      TM_OUT
04E1  BA 01F0                           MOV     DX,HF_PORT
04E4  B9 0004                           MOV     CX,4                    ; GET ECC BYTES
04E7  EC                        CMD_I2: IN      AL,DX
04E8  26: 88 05                         MOV     ES:BYTE PTR [DI],AL     ; GO SLOW FOR BOARD
04EB  47                                INC     DI
04EC  E2 F9                             LOOP    CMD_I2
04EE  E8 061E R                 CMD_I3: CALL    CHECK_STATUS
04F1  75 07                             JNZ     CMD_ABORT               ; ERROR RETURNED
04F3  F6 06 008C R 80                   TEST    HF_STATUS,ST_BUSY       ; CHECK FOR MORE
04F8  75 CD                             JNZ     SHORT CMD_I1
04FA                            CMD_ABORT:
04FA                            TM_OUT:
04FA  C3                                RET

                                ;-----------------------------------------------
                                ; COMMANDO                                     :
                                ;       REPEATEDLY OUTPUTS DATA TIL NSECTOR    :
                                ;       RETURNS ZERO                           :
```

```
                                   ;--------------------------------------------
04FB                               COMMANDO:
04FB  E8 068F R                            CALL    CHECK_DMA             ; CHECK 64K BOUNDARY ERROR
04FE  72 FA                                JC      CMD_ABORT
0500  8B F3                        CMD_OF: MOV     SI,BX
0502  E8 0544 R                            CALL    COMMAND               ; OUTPUT COMMAND
0505  75 F3                                JNZ     CMD_ABORT
0507  E8 0608 R                            CALL    WAIT_DRQ              ; WAIT FOR DATA REQUEST
050A  72 EE                                JC      TM_OUT                ; TOO LONG
050C  1E                           CMD_01: PUSH    DS
050D  06                                   PUSH    ES                    ; MOVE ES TO DS
050E  1F                                   POP     DS
050F  B9 0100                              MOV     CX,256D               ; PUT THE DATA OUT TO THE CARD
0512  BA 01F0                              MOV     DX,HF_PORT
0515  FC                                   CLD
                                           REP_OUTSW
0516  F3 6F                    +                   DB      0F3H,06FH
0518  1F                                   POP     DS                    ; RESTORE DS
0519  F6 06 0048 R 02                      TEST    CMD_BLOCK+6,ECC_MODE  ; CHECK FOR NORMAL OUTPUT
051E  74 12                                JZ      CMD_03
0520  E8 0608 R                            CALL    WAIT_DRQ              ; WAIT FOR DATA REQUEST
0523  72 D5                                JC      TM_OUT
0525  BA 01F0                              MOV     DX,HF_PORT
0528  B9 0004                              MOV     CX,4                  ; OUTPUT THE ECC BYTES
052B  26: 8A 04                    CMD_02: MOV     AL,ES:BYTE PTR [SI]
052E  EE                                   OUT     DX,AL
052F  46                                   INC     SI
0530  E2 F9                                LOOP    CMD_02
0532                               CMD_03:
0532  E8 05A5 R                            CALL    WAIT                  ; WAIT FOR SECTOR COMPLETE INT
0535  75 C3                                JNZ     TM_OUT                ; ERROR RETURNED
0537  E8 061E R                            CALL    CHECK_STATUS
053A  75 BE                                JNZ     CMD_ABORT
053C  F6 06 008C R 08                      TEST    HF_STATUS,ST_DRQ      ; CHECK FOR MORE
0541  75 C9                                JNZ     SHORT CMD_01
0543  C3                                   RET

                                   ;------------------------------------------------------
                                   ; COMMAND                                              :
                                   ;       THIS ROUTINE OUTPUTS THE COMMAND BLOCK         :
                                   ; OUTPUT                                               :
                                   ;       BL = STATUS                                    :
                                   ;       BH = ERROR REGISTER                            :
                                   ;                                                      :
                                   ;------------------------------------------------------

0544                               COMMAND PROC    NEAR
0544  53                                   PUSH    BX                    ; WAIT FOR SEEK COMPLETE AND READY
0545  B9 0600                              MOV     CX,DELAY_2            ; SET INITIAL DELAY BEFORE TEST
0548                               COMMAND1:
0548  51                                   PUSH    CX                    ; SAVE LOOP COUNT
0549  E8 044E R                            CALL    TST_RDY               ; CHECK DRIVE READY
054C  59                                   POP     CX
054D  74 0B                                JZ      COMMAND2              ; DRIVE IS READY
054F  80 3E 0074 R 80                      CMP     DISK_STATUS1,TIME_OUT ; TST_RDY TIMED OUT--GIVE UP
0554  74 43                                JZ      CMD_TIMEOUT
0556  E2 F0                                LOOP    COMMAND1              ; KEEP TRYING FOR A WHILE
0558  EB 44                                JMP     SHORT COMMAND4        ; ITS NOT GOING TO GET READY
055A                               COMMAND2:
055A  5B                                   POP     BX
055B  57                                   PUSH    DI                    ; ** IO DELAY NOT REQUIRED **
055C  C6 06 008E R 00                      MOV     HF_INT_FLAG,0         ; RESET INTERRUPT FLAG
0561  E4 A1                                IN      AL,INT_CTL_PORT+1     ; TURN ON SECOND INTERRUPT CHIP
0563  24 BF                                AND     AL,0BFH
0565  E6 A1                                OUT     INT_CTL_PORT+1,AL
0567  E4 21                                IN      AL,INT1_CTL_PORT+1    ; LET INTERRUPTS PASS THRU TO
0569  24 FB                                AND     AL,0FBH               ;  SECOND CHIP
056B  E6 21                                OUT     INT1_CTL_PORT+1,AL
056D  BF 0042 R                            MOV     DI,OFFSET CMD_BLOCK   ; INDEX THE COMMAND TABLE
0570  BA 01F1                              MOV     DX,HF_PORT+1          ; DISK ADDRESS
0573  F6 06 0076 R C0                      TEST    CONTROL_BYTE,0C0H     ; CHECK FOR RETRY SUPPRESSION
0578  74 12                                JZ      COMMAND3
057A  A0 0048 R                            MOV     AL,CMD_BLOCK+6        ; YES-GET OP CODE
057D  24 F0                                AND     AL,0F0H               ; GET RID OF MODIFIERS
057F  3C 20                                CMP     AL,20H                ; 20H-40H IS READ, WRITE, VERIFY
0581  72 09                                JB      COMMAND3
0583  3C 40                                CMP     AL,40H
0585  77 05                                JA      COMMAND3
0587  80 0E 0048 R 01                      OR      CMD_BLOCK+6,NO_RETRIES ; VALID OP FOR RETRY SUPPRESS
058C                               COMMAND3:
058C  8A 05                                MOV     AL,[DI]               ; GET THE COMMAND STRING
058E  EE                                   OUT     DX,AL                 ; GIVE IT TO CONTROLLER
058F  47                                   INC     DI                    ; NEXT BYTE
0590  42                                   INC     DX                    ; NEXT DISK REGISTER
0591  81 FA 01F8                           CMP     DX,HF_PORT+8          ; ALL DONE?
0595  75 F5                                JNZ     COMMAND3              ; NO--GO DO NEXT ONE
0597  5F                                   POP     DI
0598  C3                                   RET                           ; ZERO FLAG IS SET
0599                               CMD_TIMEOUT:
0599  C6 06 0074 R 20                      MOV     DISK_STATUS1,BAD_CNTLR
059E                               COMMAND4:
059E  5B                                   POP     BX
059F  80 3E 0074 R 00                      CMP     DISK_STATUS1,0        ; SET CONDITION CODE FOR CALLER
05A4  C3                                   RET
05A5                               COMMAND ENDP

                                   ;------------------------------------------------------
                                   ; WAIT FOR INTERRUPT                                   :
                                   ;------------------------------------------------------

05A5                               WAIT    PROC    NEAR
05A5  FB                                   STI                           ; MAKE SURE INTERRUPTS ARE ON
05A6  2B C9                                SUB     CX,CX                 ; SET INITIAL DELAY BEFORE TEST
05A8  F8                                   CLC
05A9  B8 9000                              MOV     AX,9000H              ; DEVICE WAIT INTERRUPT
05AC  CD 15                                INT     15H
05AE  72 28                                JC      WT3                   ; DEVICE TIMED OUT
05B0  F6 06 008E R 80                      TEST    HF_INT_FLAG,80H       ; TEST FOR INTERRUPT ALREADY
05B5  75 11                                JNZ     WT2
05B7  B3 20                                MOV     BL,DELAY_1            ; SET DELAY COUNT
                                   ;
                                   ;       WAIT LOOP
                                   ;
05B9  F6 06 008E R 80              WT1:    TEST    HF_INT_FLAG,80H       ; TEST FOR INTERRUPT
05BE  E1 F9                                LOOPZ   WT1
05C0  75 06                                JNZ     WT2                   ; INTERRUPT--LETS GO
05C2  FE CB                                DEC     BL
05C4  75 F3                                JNZ     WT1                   ; KEEP TRYING FOR A WHILE
05C6  EB 10                                JMP     SHORT WT3
05C8  C6 06 0074 R 00              WT2:    MOV     DISK_STATUS1,0
05CD  C6 06 008E R 00                      MOV     HF_INT_FLAG,0
05D2  80 3E 0074 R 00              WTX:    CMP     DISK_STATUS1,0        ; SET CONDITION CODE FOR CALLER
05D7  C3                                   RET
05D8  C6 06 0074 R 80              WT3:    MOV     DISK_STATUS1,TIME_OUT ; REPORT TIME OUT ERROR
05DD  EB F3                                JMP     WTX
05DF                               WAIT    ENDP
```

SECTION 5

```
;-----------------------------------------------------
; WAIT FOR CONTROLLER NOT BUSY                        :
;-----------------------------------------------------
05DF                    NOT_BUSY        PROC    NEAR
05DF  FB                        STI                             ; MAKE SURE INTERRUPTS ARE ON
05E0  53                        PUSH    BX
05E1  B3 20                     MOV     BL,DELAY_1
05E3  2B C9                     SUB     CX,CX                   ; SET INITIAL DELAY BEFORE TEST
05E5  BA 01F7                   MOV     DX,HF_PORT+7
05E8  EC                NB1:    IN      AL,DX                   ; CHECK STATUS
05E9  A8 80                     TEST    AL,ST_BUSY
05EB  E0 FB                     LOOPNZ  NB1
05ED  74 06                     JZ      NB2                     ; NOT BUSY--LETS GO
05EF  FE CB                     DEC     BL
05F1  75 F5                     JNZ     NB1                     ; KEEP TRYING FOR A WHILE
05F3  EB 0C                     JMP     SHORT NB3
05F5  C6 06 0074 R 00   NB2:    MOV     DISK_STATUS1,0
05FA  5B                NBX:    POP     BX
05FB  80 3E 0074 R 00           CMP     DISK_STATUS1,0          ; SET CONDITION CODE FOR CALLER
0600  C3                        RET
0601  C6 06 0074 R 80   NB3:    MOV     DISK_STATUS1,TIME_OUT   ; REPORT TIME OUT ERROR
0606  EB F2                     JMP     NBX
0608                    NOT_BUSY        ENDP

;------------------------------------------
; WAIT FOR DATA REQUEST                   :
;------------------------------------------
0608                    WAIT_DRQ        PROC    NEAR
0608  B9 0100                   MOV     CX,DELAY_3
060B  BA 01F7                   MOV     DX,HF_PORT+7
060E  EC                WQ_1:   IN      AL,DX                   ; GET STATUS
060F  A8 08                     TEST    AL,ST_DRQ               ; WAIT FOR DRQ
0611  75 09                     JNZ     WQ_OK
0613  E2 F9                     LOOP    WQ_1                    ; KEEP TRYING FOR A SHORT WHILE
0615  C6 06 0074 R 80           MOV     DISK_STATUS1,TIME_OUT   ; ERROR
061A  F9                        STC
061B  C3                        RET
061C  F8                WQ_OK:  CLC
061D  C3                        RET
061E                    WAIT_DRQ        ENDP

;------------------------------------------
; CHECK HARD FILE STATUS                  :
;------------------------------------------
061E                    CHECK_STATUS    PROC    NEAR
061E  E8 0630 R                 CALL    CHECK_ST                ; CHECK THE STATUS BYTE
0621  75 07                     JNZ     CHECK_S1                ; AN ERROR WAS FOUND
0623  A8 01                     TEST    AL,ST_ERROR             ; WERE THERE ANY OTHER ERRORS
0625  74 03                     JZ      CHECK_S1                ; NO ERROR REPORTED
0627  E8 0664 R                 CALL    CHECK_ER                ; ERROR REPORTED
062A                    CHECK_S1:
062A  80 3E 0074 R 00           CMP     DISK_STATUS1,0          ; SET STATUS FOR CALLER
062F  C3                        RET
0630                    CHECK_STATUS    ENDP
;------------------------------------------
; CHECK HARD FILE STATUS BYTE             :
;------------------------------------------
0630                    CHECK_ST        PROC    NEAR
0630  BA 01F7                   MOV     DX,HF_PORT+7            ; GET THE STATUS
0633  EC                        IN      AL,DX
0634  A2 008C R                 MOV     HF_STATUS,AL
0637  B4 00                     MOV     AH,0
0639  A8 80                     TEST    AL,ST_BUSY              ; IF STILL BUSY
063B  75 1A                     JNZ     CKST_EXIT               ;  REPORT OK
063D  B4 CC                     MOV     AH,WRITE_FAULT
063F  A8 20                     TEST    AL,ST_WRT_FLT           ; CHECK FOR WRITE FAULT
0641  75 14                     JNZ     CKST_EXIT
0643  B4 AA                     MOV     AH,NOT_RDY
0645  A8 40                     TEST    AL,ST_READY             ; CHECK FOR NOT READY
0647  74 0E                     JZ      CKST_EXIT
0649  B4 40                     MOV     AH,BAD_SEEK
064B  A8 10                     TEST    AL,ST_SEEK_COMPL        ; CHECK FOR SEEK NOT COMPLETE
064D  74 08                     JZ      CKST_EXIT
064F  B4 11                     MOV     AH,DATA_CORRECTED
0651  A8 04                     TEST    AL,ST_CORRCTD           ; CHECK FOR CORRECTED ECC
0653  75 02                     JNZ     CKST_EXIT
0655  B4 00                     MOV     AH,0
0657                    CKST_EXIT:
0657  88 26 0074 R              MOV     DISK_STATUS1,AH         ; SET ERROR FLAG
065B  80 FC 11                  CMP     AH,DATA_CORRECTED       ; KEEP GOING WITH DATA CORRECTED
065E  74 03                     JZ      CKST_EXT
0660  80 FC 00                  CMP     AH,0
0663                    CKST_EX1:
0663  C3                        RET
0664                    CHECK_ST        ENDP
;------------------------------------------
; CHECK HARD FILE ERROR REGISTER          :
;------------------------------------------
0664                    CHECK_ER        PROC    NEAR
0664  BA 01F1                   MOV     DX,HF_PORT+1            ; GET THE ERROR REG
0667  EC                        IN      AL,DX
0668  A2 008D R                 MOV     HF_ERROR,AL
066B  53                        PUSH    BX
066C  B9 0008                   MOV     CX,8                    ; TEST ALL 8 BITS
066F  D0 E0             CK1:    SHL     AL,1                    ; MOVE NEXT ERROR BIT TO CARRY
0671  72 02                     JC      CK2                     ; FOUND THE ERROR
0673  E2 FA                     LOOP    CK1                     ; KEEP TRYING
0675  BB 0686 R         CK2:    MOV     BX,OFFSET ERR_TBL       ; COMPUTE ADDRESS OF
0678  03 D9                     ADD     BX,CX                   ; ERROR CODE
067A  2E: 8A 27                 MOV     AH,BYTE PTR CS:[BX]     ; GET ERROR CODE
067D  88 26 0074 R      CKEX:   MOV     DISK_STATUS1,AH         ; SAVE ERROR CODE
0681  5B                        POP     BX
0682  80 FC 00                  CMP     AH,0
0685  C3                        RET
0686  E0                ERR_TBL DB      NO_ERR
0687  02 40 01 BB               DB      BAD_ADDR_MARK,BAD_SEEK,BAD_CMD,UNDEF_ERR
068B  04 BB 10 0A               DB      RECORD_NOT_FND,UNDEF_ERR,BAD_ECC,BAD_SECTOR

068F                    CHECK_ER        ENDP

;-----------------------------------------------------
; CHECK_DMA                                           :
;  -CHECK ES:BX AND # SECTORS TO MAKE SURE THAT IT WILL :
;   FIT WITHOUT SEGMENT OVERFLOW.                     :
;  -ES:BX HAS BEEN REVISED TO THE FORMAT SSSS:000X    :
;  -OK IF # SECTORS < 80H (7FH IF LONG READ OR WRITE) :
;  -OK IF # SECTORS = 80H (7FH) AND BX <= 00H (04H)   :
;  -ERROR OTHERWISE                                   :
;-----------------------------------------------------
068F                    CHECK_DMA       PROC    NEAR
068F  50                        PUSH    AX                      ; SAVE REGS
0690  B8 8000                   MOV     AX,8000H                ; AH = MAX # SECTORS
                                                                ; AL = MAX OFFSET
0693  F6 06 0048 R 02           TEST    CMD_BLOCK+6,ECC_MODE
```

```
0698  74 03                         JZ      CKD1
069A  B8 7F04                       MOV     AX,7F04H                ; ECC IS 4 MORE BYTES
069D  3A 26 0043 R          CKD1:   CMP     AH,CMD_BLOCK+1          ; NUMBER OF SECTORS
06A1  77 06                         JA      CKDOK                   ; IT WILL FIT
06A3  72 07                         JB      CKDERR                  ; TOO MANY
06A5  3A C3                         CMP     AL,BL                   ; CHECK OFFSET ON MAX SECTORS
06A7  72 03                         JB      CKDERR                  ; ERROR
06A9  F8                    CKDOK:  CLC                             ; CLEAR CARRY
06AA  58                            POP     AX
06AB  C3                            RET                             ; NORMAL RETURN
06AC  F9                    CKDERR: STC                             ; INDICATE ERROR
06AD  C6 06 0074 R 09               MOV     DISK_STATUS1,DMA_BOUNDARY
06B2  58                            POP     AX
06B3  C3                            RET
06B4                        CHECK_DMA       ENDP

                            ;----------------------------------
                            ; SET UP ES:BX->DISK PARMS        :
                            ;----------------------------------
06B4                        GET_VEC PROC    NEAR
06B4  2B C0                         SUB     AX,AX                   ; GET DISK PARAMETER ADDRESS
06B6  8E C0                         MOV     ES,AX
                                    ASSUME  ES:ABS0
06B8  F6 C2 01                      TEST    DL,1
06BB  74 07                         JZ      GV_0
06BD  26: C4 1E 0118 R              LES     BX,HF1_TBL_VEC          ; ES:BX -> DRIVE PARAMETERS
06C2  EB 05                         JMP     SHORT GV_EXIT
06C4                        GV_0:
06C4  26: C4 1E 0104 R              LES     BX,HF_TBL_VEC           ; ES:BX -> DRIVE PARAMETERS
06C9                        GV_EXIT:
06C9  C3                            RET
06CA                        GET_VEC ENDP
                            ;----------------------------------
                            ; HARD DISK INTERRUPT ROUTINE     :
                            ;----------------------------------
06CA                        HD_INT  PROC    NEAR
06CA  50                            PUSH    AX
06CB  1E                            PUSH    DS
06CC  B8 ---- R                     MOV     AX,DATA
06CF  8E D8                         MOV     DS,AX
06D1  C6 06 008E R FF               MOV     HF_INT_FLAG,0FFH        ; ALL DONE
06D6  B0 20                         MOV     AL,EOI                  ; NON-SPECIFIC END OF INTERRUPT
06D8  E6 A0                         OUT     INT_CTL_PORT,AL         ; FOR CONTROLLER #2
06DA  EB 00                         JMP     $+2                     ; WAIT
06DC  E6 20                         OUT     INT1_CTL_PORT,AL        ; FOR CONTROLLER #1
06DE  1F                            POP     DS
06DF  FB                            STI                             ; RE-ENABLE INTERRUPTS
06E0  B8 9100                       MOV     AX,9100H                ; DEVICE POST
06E3  CD 15                         INT     15H                     ;  INTERRUPT
06E5  58                            POP     AX
06E6  CF                            IRET                            ; RETURN FROM INTERRUPT
06E7                        HD_INT  ENDP

06E7  31 2F 31 31 2F 38             DB      '1/11/84'               ; RELEASE MARKER
      34
06EE                        END_ADDRESS     LABEL   BYTE
06EE                        CODE    ENDS
                                    END
```

SECTION 5

```
                            TITLE 01/04/84 KEYBOARD BIOS
                            .LIST

                            PUBLIC  KEYBOARD_IO_1
                            PUBLIC  KB_INT_1
                            PUBLIC  K16

0000                        CODE    SEGMENT BYTE PUBLIC
                            EXTRN   DDS:NEAR
                            EXTRN   START_1:NEAR
                            EXTRN   K6:BYTE
                            EXTRN   K6L:ABS
                            EXTRN   K7:BYTE
                            EXTRN   K8:BYTE
                            EXTRN   K9:BYTE
                            EXTRN   K10:BYTE
                            EXTRN   K11:BYTE
                            EXTRN   K12:BYTE
                            EXTRN   K13:BYTE
                            EXTRN   K14:BYTE
                            EXTRN   K15:BYTE

                            ;---- INT 16 ------------------------------------------------------
                            ; KEYBOARD I/O
                            ;       THESE ROUTINES PROVIDE KEYBOARD SUPPORT
                            ; INPUT
                            ;       (AH)=0  READ THE NEXT ASCII CHARACTER STRUCK FROM THE KEYBOARD
                            ;               RETURN THE RESULT IN (AL), SCAN CODE IN (AH)
                            ;       (AH)=1  SET THE Z FLAG TO INDICATE IF AN ASCII CHARACTER IS AVAILABLE
                            ;               TO BE READ.
                            ;               (ZF)=1 -- NO CODE AVAILABLE
                            ;               (ZF)=0 -- CODE IS AVAILABLE
                            ;               IF ZF = 0, THE NEXT CHARACTER IN THE BUFFER TO BE READ IS
                            ;               IN AX, AND THE ENTRY REMAINS IN THE BUFFER
                            ;       (AH)=2  RETURN THE CURRENT SHIFT STATUS IN AL REGISTER
                            ;               THE BIT SETTINGS FOR THIS CODE ARE INDICATED IN THE
                            ;               THE EQUATES FOR KB_FLAG
                            ; OUTPUT
                            ;       AS NOTED ABOVE, ONLY AX AND FLAGS CHANGED
                            ;       ALL REGISTERS RETAINED
                            ;----------------------------------------
                                    ASSUME  CS:CODE,DS:DATA

0000                        KEYBOARD_IO_1   PROC    FAR     ;>>> ENTRY POINT FOR ORG 0E82EH
0000  FB                            STI                     ; INTERRUPTS BACK ON
0001  1E                            PUSH    DS              ; SAVE CURRENT DS
0002  53                            PUSH    BX              ; SAVE BX TEMPORARILY
0003  E8 0000 E                     CALL    DDS             ; ESTABLISH POINTER TO DATA REGION
0006  0A E4                         OR      AH,AH           ; AH=0
0008  74 0B                         JZ      K1B             ; ASCII_READ
000A  FE CC                         DEC     AH              ; AH=1
000C  74 45                         JZ      K2              ; ASCII_STATUS
000E  FE CC                         DEC     AH              ; AH=2
0010  74 67                         JZ      K3              ; SHIFT_STATUS
0012  5B                            POP     BX              ; RECOVER REGISTER
0013  1F                            POP     DS
0014  CF                            IRET                    ; INVALID COMMAND

                            ;------ READ THE KEY TO FIGURE OUT WHAT TO DO

0015  8B 1E 001A R          K1B:    MOV     BX,BUFFER_HEAD  ; GET POINTER TO HEAD OF BUFFER
0019  3B 1E 001C R                  CMP     BX,BUFFER_TAIL  ; TEST END OF BUFFER
001D  75 07                         JNE     K1C             ; IF ANYTHING IN BUFFER DONT DO INTERRUPT
                            ;
001F  B8 9002                       MOV     AX,09002H       ; MOVE IN WAIT CODE & TYPE
0022  CD 15                         INT     15H             ; PERFORM OTHER FUNCTION
0024                        K1:                             ; ASCII READ
0024  FB                            STI                     ; INTERRUPTS BACK ON DURING LOOP
0025  90                            NOP                     ; ALLOW AN INTERRUPT TO OCCUR
0026  FA                    K1C:    CLI                     ; INTERRUPTS BACK OFF
0027  8B 1E 001A R                  MOV     BX,BUFFER_HEAD  ; GET POINTER TO HEAD OF BUFFER
002B  3B 1E 001C R                  CMP     BX,BUFFER_TAIL  ; TEST END OF BUFFER
002F  53                            PUSH    BX              ; SAVE ADDRESS
0030  9C                            PUSHF                   ; SAVE FLAG
0031  E8 048A R                     CALL    MAKE_LED        ; GO GET MODE INDICATOR DATA BYTE
0034  8A 1E 0097 R                  MOV     BL,KB_FLAG_2    ; GET PREVIOUS BITS
0038  32 D8                         XOR     BL,AL           ; SEE IF ANY DIFFERENT
003A  80 E3 07                      AND     BL,07H          ; ISOLATE INDICATOR BITS
003D  74 04                         JZ      K1A             ; IF NO CHANGE BYPASS UPDATE
                            ;
003F  E8 044C R                     CALL    SND_LED1        ; GO TURN ON MODE INDICATORS
0042  FA                            CLI                     ; DISABLE INTERRUPTS
0043  9D                    K1A:    POPF                    ; RESTORE FLAGS
0044  5B                            POP     BX              ; RESTORE ADDRESS
0045  74 DD                         JZ      K1              ; LOOP UNTIL SOMETHING IN BUFFER
                            ;
0047  8B 07                         MOV     AX,[BX]         ; GET SCAN CODE AND ASCII CODE
0049  E8 007F R                     CALL    K4              ; MOVE POINTER TO NEXT POSITION
004C  89 1E 001A R                  MOV     BUFFER_HEAD,BX  ; STORE VALUE IN VARIABLE
                            ;
0050  5B                            POP     BX              ; RECOVER REGISTER
0051  1F                            POP     DS              ; RECOVER SEGMENT
0052  CF                            IRET                    ; RETURN TO CALLER

                            ;------ ASCII STATUS

0053                        K2:
0053  FA                            CLI                     ; INTERRUPTS OFF
0054  8B 1E 001A R                  MOV     BX,BUFFER_HEAD  ; GET HEAD POINTER
0058  3B 1E 001C R                  CMP     BX,BUFFER_TAIL  ; IF EQUAL (Z=1) THEN NOTHING THERE
005C  8B 07                         MOV     AX,[BX]
005E  9C                            PUSHF                   ; SAVE FLAGS
                            ;
005F  50                            PUSH    AX              ; SAVE CODE
0060  E8 048A R                     CALL    MAKE_LED        ; GO GET MODE INDICATOR DATA BYTE
0063  8A 1E 0097 R                  MOV     BL,KB_FLAG_2    ; GET PREVIOUS BITS
0067  32 D8                         XOR     BL,AL           ; SEE IF ANY DIFFERENT
0069  80 E3 07                      AND     BL,07H          ; ISOLATE INDICATOR BITS
006C  74 03                         JZ      SK2             ; IF NO CHANGE BYPASS UPDATE
                            ;
006E  E8 044C R                     CALL    SND_LED1        ; GO TURN ON MODE INDICATORS
0071  58                    SK2:    POP     AX              ; RESTORE CODE
0072  9D                            POPF                    ; RESTORE FLAGS
0073  FB                            STI                     ; INTERRUPTS BACK ON
0074  5B                            POP     BX              ; RECOVER REGISTER
0075  1F                            POP     DS              ; RECOVER SEGMENT
0076  CA 0002                       RET     2               ; THROW AWAY FLAGS

                            ;------ SHIFT STATUS

0079                        K3:
0079  A0 0017 R                     MOV     AL,KB_FLAG      ; GET THE SHIFT STATUS FLAGS
007C  5B                            POP     BX              ; RECOVER REGISTER
007D  1F                            POP     DS              ; RECOVER REGISTERS
007E  CF                            IRET                    ; RETURN TO CALLER
007F                        KEYBOARD_IO_1   ENDP
```

```
                        ;------ INCREMENT A BUFFER POINTER

007F                    K4      PROC    NEAR
007F  43                        INC     BX              ; MOVE TO NEXT WORD IN LIST
0080  43                        INC     BX
0081  3B 1E 0082 R              CMP     BX,BUFFER_END   ; AT END OF BUFFER?
0085  75 04                     JNE     K5              ; NO, CONTINUE
0087  8B 1E 0080 R              MOV     BX,BUFFER_START ; YES, RESET TO BUFFER BEGINNING
008B                    K5:
008B  C3                        RET
008C                    K4      ENDP

                        ;------ KEYBOARD INTERRUPT ROUTINE

008C                    KB_INT_1 PROC   FAR
008C  FB                        STI                     ; ENABLE INTERRUPTS
008D  55                        PUSH    BP
008E  50                        PUSH    AX
008F  53                        PUSH    BX
0090  51                        PUSH    CX
0091  52                        PUSH    DX
0092  56                        PUSH    SI
0093  57                        PUSH    DI
0094  1E                        PUSH    DS
0095  06                        PUSH    ES
0096  FC                        CLD                     ; FORWARD DIRECTION
0097  E8 0000 E                 CALL    DDS             ; SET UP ADDRESSING
009A  B0 AD                     MOV     AL,DIS_KBD      ; DISABLE THE KEYBOARD
009C  E8 0498 R                 CALL    SHIP_IT         ; EXECUTE DISABLE

                        ;------- WAIT FOR COMMAND TO ACCEPTED
009F  FA                        CLI                     ; DISABLE INTERRUPTS
00A0  2B C9                     SUB     CX,CX           ;
00A2                    KB_INT_01:
00A2  E4 64                     IN      AL,STATUS_PORT  ;
00A4  A8 02                     TEST    AL,INPT_BUF_FULL
00A6  E0 FA                     LOOPNZ  KB_INT_01       ; WAIT FOR COMMAND TO BE ACCEPTED

00A8  E4 60                     IN      AL,KB_DATA      ; READ IN THE CHARACTER
00AA  FB                        STI                     ; ENABLE INTERRUPTS AGAIN

                        ;--------CHECK FOR A RESEND COMMAND TO KEYBOARD

00AB  3C FE                     CMP     AL,KB_RESEND    ; IS THE INPUT A RESEND
00AD  74 0D                     JE      KB_INT_4        ; GO IF RESEND

                        ;------- CHECK FOR RESPONSE TO A COMMAND TO KEYBOARD

00AF  3C FA                     CMP     AL,KB_ACK       ; IS THE INPUT AN ACKNOWLEDGE
00B1  75 12                     JNZ     KB_INT_2        ; GO IF NOT

                        ;------- A COMMAND TO THE KEYBOARD WAS ISSUED

00B3  FA                        CLI                     ; DISABLE INTERRUPTS
00B4  80 0E 0097 R 10           OR      KB_FLAG_2,KB_FA ; INDICATE ACK RECEIVED
00B9  E9 01E2 R                 JMP     K26             ; RETURN IF NOT (THIS ACK RETURNED FOR DATA)

                        ;-------- RESEND THE LAST BYTE

00BC                    KB_INT_4:
00BC  FA                        CLI                     ; DISABLE INTERRUPTS
00BD  80 0E 0097 R 20           OR      KB_FLAG_2,KB_FE ; INDICATE RESEND RECEIVED
00C2  E9 01E2 R                 JMP     K26             ; RETURN IF NOT (THIS ACK RETURNED FOR DATA)

00C5                    KB_INT_2:

                        ;--------UPDATE MODE INDICATORS IF CHANGE IN STATE

00C5  50                        PUSH    AX              ; SAVE DATA IN
00C6  E8 048A R                 CALL    MAKE_LED        ; GO GET MODE INDICATOR DATA BYTE
00C9  8A 1E 0097 R              MOV     BL,KB_FLAG_2    ; GET PREVIOUS BITS
00CD  32 D8                     XOR     BL,AL           ; SEE IF ANY DIFFERENT
00CF  80 E3 07                  AND     BL,07H          ; ISOLATE INDICATOR BITS
00D2  74 03                     JZ      UP0             ; IF NO CHANGE BYPASS UPDATE
                        ;
00D4  E8 0439 R                 CALL    SND_LED         ; GO TURN ON MODE INDICATORS
00D7  58                UP0:    POP     AX              ; RESTORE DATA IN
00D8  8A E0                     MOV     AH,AL           ; SAVE SCAN CODE IN AH ALSO

                        ;------ TEST FOR OVERRUN SCAN CODE FROM KEYBOARD

00DA  3C FF                     CMP     AL,0FFH         ; IS THIS AN OVERRUN CHAR
00DC  75 03                     JNZ     K16             ; NO, TEST FOR SHIFT KEY
00DE  E9 03D6 R                 JMP     K62             ; BUFFER_FULL_BEEP

                        ;------ TEST FOR SHIFT KEYS

00E1                    K16:                            ; TEST_SHIFT
00E1  24 7F                     AND     AL,07FH         ; TURN OFF THE BREAK BIT
00E3  0E                        PUSH    CS
00E4  07                        POP     ES              ; ESTABLISH ADDRESS OF SHIFT TABLE

                        ;------- TEST FOR SYSTEM KEY

00E5  3C 54                     CMP     AL,SYS_KEY      ; IS IT THE SYSTEM KEY?
00E7  75 3D                     JNZ     K16A            ; CONTINUE IF NOT
                        ;
00E9  F6 C4 80                  TEST    AH,080H         ; CHECK IF THIS A BREAK CODE
00EC  75 21                     JNZ     K16C            ; DONT TOUCH SYSTEM INDICATOR IF TRUE
                        ;
00EE  F6 06 0018 R 04           TEST    KB_FLAG_1,SYS_SHIFT ; SEE IF IN SYSTEM KEY HELD DOWN
00F3  75 17                     JNZ     K16B            ; IF YES, DONT PROCESS SYSTEM INDICATOR
                        ;
00F5  80 0E 0018 R 04           OR      KB_FLAG_1,SYS_SHIFT ; INDICATE SYSTEM KEY DEPRESSED
00FA  B0 20                     MOV     AL,EOI                  ; END OF INTERRUPT COMMAND
00FC  E6 20                     OUT     020H,AL                 ; SEND COMMAND TO INTERRUPT CONTROL PORT
                                                                ; INTERRUPT-RETURN-NO-EOI
00FE  B0 AE                     MOV     AL,ENA_KBD              ; INSURE KEYBOARD IS ENABLED
0100  E8 0498 R                 CALL    SHIP_IT                 ; EXECUTE ENABLE
0103  B8 8500                   MOV     AX,08500H       ; FUNCTION VALUE FOR MAKE OF SYSTEM KEY
0106  FB                        STI                     ; MAKE SURE INTERRUPTS ENABLED
0107  CD 15                     INT     15H             ; USER INTERRUPT
0109  E9 01EC R                 JMP     K27A            ; END PROCESSING
010C  E9 01E2 R         K16B:   JMP     K26             ; IGNORE SYSTEM KEY
                        ;
010F  80 26 0018 R FB   K16C:   AND     KB_FLAG_1,NOT SYS_SHIFT ; TURN OFF SHIFT KEY HELD DOWN
0114  B0 20                     MOV     AL,EOI                  ; END OF INTERRUPT COMMAND
0116  E6 20                     OUT     020H,AL                 ; SEND COMMAND TO INTERRUPT CONTROL PORT
                                                                ; INTERRUPT-RETURN-NO-EOI
0118  B0 AE                     MOV     AL,ENA_KBD              ; INSURE KEYBOARD IS ENABLED
011A  E8 0498 R                 CALL    SHIP_IT                 ; EXECUTE ENABLE
011D  B8 8501                   MOV     AX,08501H       ; FUNCTION VALUE FOR BREAK OF SYSTEM KEY
0120  FB                        STI                     ; MAKE SURE INTERRUPTS ENABLED
0121  CD 15                     INT     15H             ; USER INTERRUPT
```

```
0123  E9 01EC R                       JMP     K27A              ; IGNORE SYSTEM KEY
                                ;
0126  BF 0000 E               K16A:   MOV     DI,OFFSET K6      ; SHIFT KEY TABLE
0129  B9 0000 E                       MOV     CX,OFFSET K6L     ; LENGTH
012C  F2/ AE                          REPNE   SCASB             ; LOOK THROUGH THE TABLE FOR A MATCH
012E  8A C4                           MOV     AL,AH             ; RECOVER SCAN CODE
0130  74 03                           JE      K17               ; JUMP IF MATCH FOUND
0132  E9 01CE R                       JMP     K25               ; IF NO MATCH, THEN SHIFT NOT FOUND

                              ;------ SHIFT KEY FOUND

0135  81 EF 0001 E            K17:    SUB     DI,OFFSET K6+1    ; ADJUST PTR TO SCAN CODE MTCH
0139  2E: 8A A5 0000 E                MOV     AH,CS:K7[DI]      ; GET MASK INTO AH
013E  A8 80                           TEST    AL,80H            ; TEST FOR BREAK KEY
0140  74 02                           JZ      K17C              ; BREAK_SHIFT_FOUND
0142  EB 63                           JMP     SHORT K23         ; CONTINUE

                              ;-------- DETERMINE SET OR TOGGLE

0144  80 FC 10                K17C:   CMP     AH,SCROLL_SHIFT
0147  73 07                           JAE     K18                         ; IF SCROLL SHIFT OR ABOVE, TOGGLE KEY

                              ;------ PLAIN SHIFT KEY, SET SHIFT ON

0149  08 26 0017 R                    OR      KB_FLAG,AH                  ; TURN ON SHIFT BIT
014D  E9 01E2 R                       JMP     K26                         ; INTERRUPT_RETURN

                              ;------ TOGGLED SHIFT KEY, TEST FOR 1ST MAKE OR NOT

0150                          K18:                                        ; SHIFT-TOGGLE
0150  F6 06 0017 R 04                 TEST    KB_FLAG, CTL_SHIFT          ; CHECK CTL SHIFT STATE
0155  74 03                           JZ      K18A                        ; JUMP IF NOT CTL STATE

0157  EB 75 90                        JMP     K25                         ; JUMP IF CTL STATE
015A  3C 52                   K18A:   CMP     AL, INS_KEY                 ; CHECK FOR INSERT KEY
015C  75 25                           JNZ     K22                         ; JUMP IF NOT INSERT KEY
015E  F6 06 0017 R 08                 TEST    KB_FLAG, ALT_SHIFT          ; CHECK FOR ALTERNATE SHIFT
0163  74 03                           JZ      K19                         ; JUMP IF NOT ALTERNATE SHIFT
0165  EB 67 90                        JMP     K25               ; JUMP IF ALTERNATE SHIFT
0168  F6 06 0017 R 20         K19:    TEST    KB_FLAG, NUM_STATE          ; CHECK FOR BASE STATE
016D  75 0D                           JNZ     K21                         ; JUMP IF NUM LOCK IS ON
016F  F6 06 0017 R 03                 TEST    KB_FLAG, LEFT_SHIFT+ RIGHT_SHIFT        ;
0174  74 0D                           JZ      K22                         ; JUMP IF BASE STATE

0176                          K20:                                        ; NUMERIC ZERO, NOT INSERT KEY
0176  B8 5230                         MOV     AX, 5230H                   ; PUT OUT AN ASCII ZERO
0179  E9 0375 R                       JMP     K57                         ; BUFFER_FILL
017C                          K21:                                        ; MIGHT BE NUMERIC
017C  F6 06 0017 R 03                 TEST    KB_FLAG, LEFT_SHIFT+ RIGHT_SHIFT        ;
0181  74 F3                           JZ      K20                         ; JUMP NUMERIC, NOT INSERT

0183                          K22:                                        ; SHIFT TOGGLE KEY HIT; PROCESS IT
0183  84 26 0018 R                    TEST    AH,KB_FLAG_1                ; IS KEY ALREADY DEPRESSED
0187  74 02                           JZ      K22A0                       ; GO IF NOT
0189  EB 57                           JMP     SHORT K26                   ; JUMP IF KEY ALREADY DEPRESSED
018B  08 26 0018 R            K22A0:  OR      KB_FLAG_1,AH                ; INDICATE THAT THE KEY IS DEPRESSED
018F  30 26 0017 R                    XOR     KB_FLAG,AH                  ; TOGGLE THE SHIFT STATE

                              ;------- TOGGLE LED IF CAPS OR NUM KEY DEPRESSED

0193  F6 C4 70                        TEST    AH,CAPS_SHIFT+NUM_SHIFT+SCROLL_SHIFT ; SHIFT TOGGLE?
0196  74 05                           JZ      K22B                        ; GO IF NOT
0198  50                              PUSH    AX                          ; SAVE SCAN CODE AND SHIFT MASK
0199  E8 0439 R                       CALL    SND_LED                     ; GO TURN MODE INDICATORS ON
019C  58                              POP     AX                          ; RESTORE SCAN CODE

019D  3C 52                   K22B:   CMP     AL,INS_KEY                  ; TEST FOR 1ST MAKE OF INSERT KEY
019F  75 41                           JNE     K26                         ; JUMP IF NOT INSERT KEY
01A1  B8 5200                         MOV     AX,INS_KEY*256              ; SET SCAN CODE INTO AH, 0 INTO AL
01A4  E9 0375 R                       JMP     K57                         ; PUT INTO OUTPUT BUFFER

                              ;------ BREAK SHIFT FOUND

01A7                          K23:                                        ; BREAK-SHIFT-FOUND
01A7  80 FC 10                        CMP     AH,SCROLL_SHIFT             ; IS THIS A TOGGLE KEY
01AA  73 1A                           JAE     K24                         ; YES, HANDLE BREAK TOGGLE
01AC  F6 D4                           NOT     AH                          ; INVERT MASK
01AE  20 26 0017 R                    AND     KB_FLAG,AH                  ; TURN OFF SHIFT BIT
01B2  3C B8                           CMP     AL,ALT_KEY+80H              ; IS THIS ALTERNATE SHIFT RELEASE
01B4  75 2C                           JNE     K26                         ; INTERRUPT_RETURN

                              ;------ ALTERNATE SHIFT KEY RELEASED, GET THE VALUE INTO BUFFER

01B6  A0 0019 R                       MOV     AL,ALT_INPUT
01B9  B4 00                           MOV     AH,0                        ; SCAN CODE OF 0
01BB  88 26 0019 R                    MOV     ALT_INPUT,AH                ; ZERO OUT THE FIELD
01BF  3C 00                           CMP     AL,0                        ; WAS THE INPUT=0
01C1  74 1F                           JE      K26                         ; INTERRUPT_RETURN
01C3  E9 037E R                       JMP     K58                         ; IT WASN'T, SO PUT IN BUFFER

01C6                          K24:                                        ; BREAK-TOGGLE
01C6  F6 D4                           NOT     AH                          ; INVERT MASK
01C8  20 26 0018 R                    AND     KB_FLAG_1,AH                ; INDICATE NO LONGER DEPRESSED
01CC  EB 14                           JMP     SHORT K26                   ; INTERRUPT_RETURN

                              ;------ TEST FOR HOLD STATE

01CE                          K25:                                        ; NO-SHIFT-FOUND
01CE  3C 80                           CMP     AL,80H                      ; TEST FOR BREAK KEY
01D0  73 10                           JAE     K26                         ; NOTHING FOR BREAK CHARS FROM HERE ON
01D2  F6 06 0018 R 08                 TEST    KB_FLAG_1,HOLD_STATE        ; ARE WE IN HOLD STATE
01D7  74 1E                           JZ      K28                         ; BRANCH AROUND TEST IF NOT
01D9  3C 45                           CMP     AL,NUM_KEY
01DB  74 05                           JE      K26                         ; CAN'T END HOLD ON NUM_LOCK
01DD  80 26 0018 R F7                 AND     KB_FLAG_1,NOT HOLD_STATE          ; TURN OFF THE HOLD STATE BIT

01E2                          K26:                              ; INTERRUPT-RETURN
01E2  FA                              CLI                       ; TURN OFF INTERRUPTS
01E3  B0 20                           MOV     AL,EOI            ; END OF INTERRUPT COMMAND
01E5  E6 20                           OUT     020H,AL           ; SEND COMMAND TO INTERRUPT CONTROL PORT
01E7                          K27:                              ; INTERRUPT-RETURN-NO-EOI
01E7  B0 AE                           MOV     AL,ENA_KBD        ; INSURE KEYBOARD IS ENABLED
01E9  E8 0498 R                       CALL    SHIP_IT           ; EXECUTE ENABLE

01EC  FA                      K27A:   CLI                       ; DISABLE INTERRUPTS
01ED  07                              POP     ES                ; RESTORE REGISTERS
01EE  1F                              POP     DS                ; *
01EF  5F                              POP     DI                ; *
01F0  5E                              POP     SI                ; *
01F1  5A                              POP     DX                ; *
01F2  59                              POP     CX                ; *
01F3  5B                              POP     BX                ; *
01F4  58                              POP     AX                ; *
01F5  5D                              POP     BP                ; *
01F6  CF                              IRET                      ; RETURN, INTERRUPTS BACK ON WITH FLAG CHANGE
```

SECTION 5

```
                                ;------ NOT IN  HOLD STATE

01F7                            K28:                              ; NO-HOLD-STATE
01F7  F6 06 0017 R 08                   TEST    KB_FLAG,ALT_SHIFT ; ARE WE IN ALTERNATE SHIFT
01FC  75 03                             JNZ     K29               ; JUMP IF ALTERNATE SHIFT
01FE  E9 0290 R                         JMP     K38               ; JUMP IF NOT ALTERNATE

                                ;------ TEST FOR RESET KEY SEQUENCE (CTL ALT DEL)

0201                            K29:                              ; TEST-RESET
0201  F6 06 0017 R 04                   TEST    KB_FLAG,CTL_SHIFT ; ARE WE IN CONTROL SHIFT ALSO
0206  74 31                             JZ      K31               ; NO_RESET
0208  3C 53                             CMP     AL,DEL_KEY        ; SHIFT STATE IS THERE, TEST KEY
020A  75 2D                             JNE     K31               ; NO_RESET

                                ;------ CTL-ALT-DEL HAS BEEN FOUND, DO I/O CLEANUP

020C  C7 06 0072 R 1234                 MOV     RESET_FLAG, 1234H ; SET FLAG FOR RESET FUNCTION
0212  E9 0000 E                         JMP     START_1           ; JUMP TO POWER ON DIAGNOSTICS

                                ;------ ALT-INPUT-TABLE
0215                            K30     LABEL   BYTE
0215  52 4F 50 51 4B 4C                 DB      82,79,80,81,75,76,77
      4D
021C  47 48 49                          DB      71,72,73          ; 10 NUMBERS ON KEYPAD
                                ;------ SUPER-SHIFT-TABLE
021F  10 11 12 13 14 15                 DB      16,17,18,19,20,21,22,23 ; A-Z TYPEWRITER CHARS
      16 17
0227  18 19 1E 1F 20 21                 DB      24,25,30,31,32,33,34,35
      22 23
022F  24 25 26 2C 2D 2E                 DB      36,37,38,44,45,46,47,48
      2F 30
0237  31 32                             DB      49,50

                                ;------ IN ALTERNATE SHIFT, RESET NOT FOUND

0239                            K31:                                      ; NO-RESET
0239  3C 39                             CMP     AL,57                     ; TEST FOR SPACE KEY
023B  75 05                             JNE     K32                       ; NOT THERE
023D  B0 20                             MOV     AL,' '                    ; SET SPACE CHAR
023F  E9 0375 R                         JMP     K57                       ; BUFFER_FILL

                                ;------ LOOK FOR KEY PAD ENTRY

0242                            K32:                                      ; ALT-KEY-PAD
0242  BF 0215 R                         MOV     DI,OFFSET K30             ; ALT-INPUT-TABLE
0245  B9 000A                           MOV     CX,10                     ; LOOK FOR ENTRY USING KEYPAD
0248  F2/ AE                            REPNE   SCASB                     ; LOOK FOR MATCH
024A  75 12                             JNE     K33                       ; NO_ALT_KEYPAD
024C  81 EF 0216 R                      SUB     DI,OFFSET K30+1           ; DI NOW HAS ENTRY VALUE
0250  A0 0019 R                         MOV     AL,ALT_INPUT              ; GET THE CURRENT BYTE
0253  B4 0A                             MOV     AH,10                     ; MULTIPLY BY 10
0255  F6 E4                             MUL     AH
0257  03 C7                             ADD     AX,DI                     ; ADD IN THE LATEST ENTRY
0259  A2 0019 R                         MOV     ALT_INPUT,AL              ; STORE IT AWAY
025C  EB 84                             JMP     K26                       ; THROW AWAY THAT KEYSTROKE

                                ;------ LOOK FOR SUPERSHIFT ENTRY

025E                            K33:                                      ; NO-ALT-KEYPAD
025E  C6 06 0019 R 00                   MOV     ALT_INPUT,0               ; ZERO ANY PREVIOUS ENTRY INTO INPUT
0263  B9 001A                           MOV     CX,26                     ; DI,ES ALREADY POINTING
0266  F2/ AE                            REPNE   SCASB                     ; LOOK FOR MATCH IN ALPHABET
0268  75 05                             JNE     K34                       ; NOT FOUND, FUNCTION KEY OR OTHER
026A  B0 00                             MOV     AL,0                      ; ASCII CODE OF ZERO
026C  E9 0375 R                         JMP     K57                       ; PUT IT IN THE BUFFER

                                ;------ LOOK FOR TOP ROW OF ALTERNATE SHIFT

026F                            K34:                                      ; ALT-TOP-ROW
026F  3C 02                             CMP     AL,2                      ; KEY WITH '1' ON IT
0271  72 0C                             JB      K35                       ;  NOT ONE OF INTERESTING KEYS
0273  3C 0E                             CMP     AL,14                     ;  IS IT IN THE REGION
0275  73 08                             JAE     K35                       ;  ALT-FUNCTION
0277  80 C4 76                          ADD     AH,118                    ;  CONVERT PSUEDO SCAN CODE TO RANGE
027A  B0 00                             MOV     AL,0                      ;  INDICATE AS SUCH
027C  E9 0375 R                         JMP     K57                       ;  BUFFER_FILL

                                ;------ TRANSLATE ALTERNATE SHIFT PSEUDO SCAN CODES

027F                            K35:                                      ; ALT-FUNCTION
027F  3C 3B                             CMP     AL,59                     ; TEST FOR IN TABLE
0281  73 03                             JAE     K37                       ; ALT-CONTINUE
0283                            K36:                                      ; CLOSE-RETURN
0283  E9 01E2 R                         JMP     K26                       ; IGNORE THE KEY
0286                            K37:                                      ; ALT-CONTINUE
0286  3C 47                             CMP     AL,71                     ; IN KEYPAD REGION
0288  73 F9                             JAE     K36                       ; IF SO, IGNORE
028A  BB 0000 E                         MOV     BX,OFFSET K13             ; ALT SHIFT PSEUDO SCAN TABLE
028D  E9 03CC R                         JMP     K63                       ; TRANSLATE THAT

                                ;------ NOT IN ALTERNATE SHIFT

0290                            K38:                                      ; NOT-ALT-SHIFT
0290  F6 06 0017 R 04                   TEST    KB_FLAG,CTL_SHIFT         ; ARE WE IN CONTROL SHIFT
0295  74 62                             JZ      K44                       ; NOT-CTL-SHIFT

                                ;------ CONTROL SHIFT, TEST SPECIAL CHARACTERS
                                ;------ TEST FOR BREAK AND PAUSE KEYS
0297  3C 46                             CMP     AL,SCROLL_KEY             ; TEST FOR BREAK
0299  75 1D                             JNE     K39                       ; NO-BREAK
029B  8B 1E 0080 R                      MOV     BX,BUFFER_START           ; RESET BUFFER TO EMPTY
029F  89 1E 001A R                      MOV     BUFFER_HEAD,BX            ;
02A3  89 1E 001C R                      MOV     BUFFER_TAIL,BX            ;
02A7  C6 06 0071 R 80                   MOV     BIOS_BREAK,80H            ; TURN ON BIOS_BREAK BIT

                                ;-------- ENABLE KEYBOARD

02AC  B0 AE                             MOV     AL,ENA_KBD                ; ENABLE KEYBOARD
02AE  E8 0498 R                         CALL    SHIP_IT                   ; EXECUTE ENABLE
02B1  CD 1B                             INT     1BH                       ; BREAK INTERRUPT VECTOR
02B3  2B C0                             SUB     AX,AX                     ; PUT OUT DUMMY CHARACTER
02B5  E9 0375 R                         JMP     K57                       ; BUFFER_FILL

02B8                            K39:                                      ; NO-BREAK
02B8  3C 45                             CMP     AL,NUM_KEY                ; LOOK FOR PAUSE KEY
02BA  75 26                             JNE     K41                       ; NO-PAUSE
02BC  80 0E 0018 R 08                   OR      KB_FLAG_1,HOLD_STATE      ; TURN ON THE HOLD FLAG

                                ;-------- ENABLE KEYBOARD

02C1  B0 AE                             MOV     AL,ENA_KBD                ; ENABLE KEYBOARD
02C3  E8 0498 R                         CALL    SHIP_IT                   ; EXECUTE ENABLE
02C6  B0 20                             MOV     AL,EOI                    ; END OF INTERRUPT TO CONTROL PORT
02C8  E6 20                             OUT     020H,AL                   ; ALLOW FURTHER KEYSTROKE INTS
```

```
                               ;------ DURING PAUSE INTERVAL, TURN CRT BACK ON

02CA  80 3E 0049 R 07                  CMP     CRT_MODE,7              ; IS THIS BLACK AND WHITE CARD
02CF  74 07                            JE      K40                     ; YES, NOTHING TO DO
02D1  BA 03D8                          MOV     DX,03D8H                ; PORT FOR COLOR CARD
02D4  A0 0065 R                        MOV     AL,CRT_MODE_SET         ; GET THE VALUE OF THE CURRENT MODE
02D7  EE                               OUT     DX,AL                   ; SET THE CRT MODE, SO THAT CRT IS ON
02D8                           K40:                                    ; PAUSE-LOOP

                               ENDIF

02D8                           K40A:
02D8  F6 06 0018 R 08                  TEST    KB_FLAG_1,HOLD_STATE
02DD  75 F9                            JNZ     K40A                    ; LOOP UNTIL FLAG TURNED OFF
02DF  E9 01EC R                        JMP     K27A                    ; INTERRUPT_RETURN_NO_EOI
02E2                           K41:                                    ; NO-PAUSE

                               ;------ TEST SPECIAL CASE KEY 55

02E2  3C 37                            CMP     AL,55
02E4  75 06                            JNE     K42                     ; NOT-KEY-55
02E6  B8 7200                          MOV     AX,114*256              ; START/STOP PRINTING SWITCH
02E9  E9 0375 R                        JMP     K57                     ; BUFFER_FILL

                               ;------ SET UP TO TRANSLATE CONTROL SHIFT

02EC                           K42:                                    ; NOT-KEY-55
02EC  BB 0000 E                        MOV     BX,OFFSET K8            ; SET UP TO TRANSLATE CTL
02EF  3C 3B                            CMP     AL,59                   ; IS IT IN TABLE
02F1  72 7E                            JB      K56                     ; YES, GO TRANSLATE CHAR
                                                                       ; CTL-TABLE-TRANSLATE
02F3  BB 0000 E                        MOV     BX,OFFSET K9            ; CTL TABLE SCAN
02F6  E9 03CC R                        JMP     K63                     ; TRANSLATE_SCAN

                               ;------ NOT IN CONTROL SHIFT

02F9                           K44:                                    ; NOT-CTL-SHIFT

02F9  3C 47                            CMP     AL,71                   ; TEST FOR KEYPAD REGION
02FB  73 33                            JAE     K48                     ; HANDLE KEYPAD REGION
02FD  F6 06 0017 R 03                  TEST    KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT
0302  74 62                            JZ      K54                     ; TEST FOR SHIFT STATE

                               ;------ UPPER CASE, HANDLE SPECIAL CASES

0304  3C 0F                            CMP     AL,15                   ; BACK TAB KEY
0306  75 05                            JNE     K45                     ; NOT-BACK-TAB
0308  B8 0F00                          MOV     AX,15*256               ; SET PSEUDO SCAN CODE
030B  EB 68                            JMP     SHORT K57               ; BUFFER_FILL

030D                           K45:                                    ; NOT-BACK-TAB
030D  3C 37                            CMP     AL,55                   ; PRINT SCREEN KEY
030F  75 10                            JNE     K46                     ; NOT-PRINT-SCREEN

                               ;------ ISSUE INTERRUPT TO INDICATE PRINT SCREEN FUNCTION

0311  B0 AE                            MOV     AL,ENA_KBD              ; INSURE KEYBOARD IS ENABLED
0313  E8 0498 R                        CALL    SHIP_IT                 ; EXECUTE ENABLE
0316  B0 20                            MOV     AL,EOI                  ; END OF CURRENT INTERRUPT
0318  E6 20                            OUT     020H,AL                 ;  SO FURTHER THINGS CAN HAPPEN
031A  55                               PUSH    BP                      ; SAVE POINTER
031B  CD 05                            INT     5H                      ; ISSUE PRINT SCREEN INTERRUPT
031D  5D                               POP     BP                      ; RESTORE POINTER
031E  E9 01E7 R                        JMP     K27                     ; GO BACK WITHOUT EOI OCCURRING

0321                           K46:                                    ; NOT-PRINT-SCREEN
0321  3C 3B                            CMP     AL,59                   ; FUNCTION KEYS
0323  72 06                            JB      K47                     ; NOT-UPPER-FUNCTION
0325  BB 0000 E                        MOV     BX,OFFSET K12           ; UPPER CASE PSEUDO SCAN CODES
0328  E9 03CC R                        JMP     K63                     ; TRANSLATE_SCAN

032B                           K47:                                    ; NOT-UPPER-FUNCTION
032B  BB 0000 E                        MOV     BX,OFFSET K11           ; POINT TO UPPER CASE TABLE
032E  EB 41                            JMP     SHORT K56               ; OK, TRANSLATE THE CHAR

                               ;------ KEYPAD KEYS, MUST TEST NUM LOCK FOR DETERMINATION

0330                           K48:                                    ; KEYPAD-REGION
0330  F6 06 0017 R 20                  TEST    KB_FLAG,NUM_STATE       ; ARE WE IN NUM_LOCK
0335  75 21                            JNZ     K52                     ; TEST FOR SURE
0337  F6 06 0017 R 03                  TEST    KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT  ; ARE WE IN SHIFT STATE
033C  75 21                            JNZ     K53                     ; IF SHIFTED, REALLY NUM STATE

                               ;------ BASE CASE FOR KEYPAD

033E                           K49:                                    ; BASE-CASE
                                                                       ;
033E  3C 4A                            CMP     AL,74                   ; SPECIAL CASE FOR A COUPLE OF KEYS
0340  74 0C                            JE      K50                     ; MINUS
0342  3C 4E                            CMP     AL,78
0344  74 0D                            JE      K51
0346  2C 47                            SUB     AL,71                   ; CONVERT ORIGIN
0348  BB 0000 E                        MOV     BX,OFFSET K15           ; BASE CASE TABLE
034B  E9 03CE R                        JMP     K64                     ; CONVERT TO PSEUDO SCAN

034E  B8 4A2D                  K50:    MOV     AX,74*256+'-'           ; MINUS
0351  EB 22                            JMP     SHORT K57               ; BUFFER_FILL

0353  B8 4E2B                  K51:    MOV     AX,78*256+'+'           ; PLUS
0356  EB 1D                            JMP     SHORT K57               ; BUFFER_FILL

                               ;------ MIGHT BE NUM LOCK, TEST SHIFT STATUS

0358                           K52:                                    ; ALMOST-NUM-STATE
0358  F6 06 0017 R 03                  TEST    KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT
035D  75 DF                            JNZ     K49                     ; SHIFTED TEMP OUT OF NUM STATE

035F                           K53:                                    ; REALLY_NUM_STATE
035F  2C 46                            SUB     AL,70                   ; CONVERT ORIGIN
0361  BB 0000 E                        MOV     BX,OFFSET K14           ; NUM STATE TABLE
0364  EB 0B                            JMP     SHORT K56               ; TRANSLATE_CHAR

                               ;------ PLAIN OLD LOWER CASE

0366                           K54:                                    ; NOT-SHIFT
0366  3C 3B                            CMP     AL,59                   ; TEST FOR FUNCTION KEYS
0368  72 04                            JB      K55                     ; NOT-LOWER-FUNCTION
036A  B0 00                            MOV     AL,0                    ; SCAN CODE IN AH ALREADY
036C  EB 07                            JMP     SHORT K57               ; BUFFER_FILL

036E                           K55:                                    ; NOT-LOWER-FUNCTION
036E  BB 0000 E                        MOV     BX,OFFSET K10           ; LC TABLE

                               ;------ TRANSLATE THE CHARACTER
```

SECTION 5

```
0371                    K56:                                    ; TRANSLATE-CHAR
0371  FE C8                     DEC     AL                      ; CONVERT ORIGIN
0373  2E: D7                    XLAT    CS:K11                  ; CONVERT THE SCAN CODE TO ASCII

                        ;------ PUT CHARACTER INTO BUFFER

0375                    K57:                                    ; BUFFER-FILL
0375  3C FF                     CMP     AL,-1                   ; IS THIS AN IGNORE CHAR
0377  74 1F                     JE      K59                     ; YES, DO NOTHING WITH IT
0379  80 FC FF                  CMP     AH,-1                   ; LOOK FOR -1 PSEUDO SCAN
037C  74 1A                     JE      K59                     ; NEAR_INTERRUPT_RETURN

                        ;------ HANDLE THE CAPS LOCK PROBLEM

037E                    K58:                                    ; BUFFER-FILL-NOTEST
037E  F6 06 0017 R 40           TEST    KB_FLAG,CAPS_STATE      ; ARE WE IN CAPS LOCK STATE
0383  74 20                     JZ      K61                     ; SKIP IF NOT

                        ;------ IN CAPS LOCK STATE

0385  F6 06 0017 R 03           TEST    KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT  ; TEST FOR SHIFT STATE
038A  74 0F                     JZ      K60                     ; IF NOT SHIFT, CONVERT LOWER TO UPPER

                        ;------ CONVERT ANY UPPER CASE TO LOWER CASE

038C  3C 41                     CMP     AL,'A'                  ; FIND OUT IF ALPHABETIC
038E  72 15                     JB      K61                     ; NOT_CAPS_STATE
0390  3C 5A                     CMP     AL,'Z'
0392  77 11                     JA      K61                     ; NOT_CAPS_STATE
0394  04 20                     ADD     AL,'a'-'A'              ; CONVERT TO LOWER CASE
0396  EB 0D                     JMP     SHORT K61               ; NOT_CAPS_STATE

0398                    K59:                                    ; NEAR-INTERRUPT-RETURN
0398  E9 01E2 R                 JMP     K26                     ; INTERRUPT_RETURN

                        ;------ CONVERT ANY LOWER CASE TO UPPER CASE

039B                    K60:                                    ; LOWER-TO-UPPER
039B  3C 61                     CMP     AL,'a'                  ; FIND OUT IF ALPHABETIC
039D  72 06                     JB      K61                     ; NOT_CAPS_STATE
039F  3C 7A                     CMP     AL,'z'
03A1  77 02                     JA      K61                     ; NOT_CAPS_STATE
03A3  2C 20                     SUB     AL,'a'-'A'              ; CONVERT TO UPPER CASE

03A5                    K61:                                    ; NOT-CAPS-STATE
03A5  8B 1E 001C R              MOV     BX,BUFFER_TAIL          ; GET THE END POINTER TO THE BUFFER
03A9  8B F3                     MOV     SI,BX                   ; SAVE THE VALUE
03AB  E8 007F R                 CALL    K4                      ; ADVANCE THE TAIL
03AE  3B 1E 001A R              CMP     BX,BUFFER_HEAD          ; HAS THE BUFFER WRAPPED AROUND
03B2  74 22                     JE      K62                     ; BUFFER_FULL_BEEP
03B4  89 04                     MOV     [SI],AX                 ; STORE THE VALUE
03B6  89 1E 001C R              MOV     BUFFER_TAIL,BX          ; MOVE THE POINTER UP
03BA  FA                        CLI                             ; TURN OFF INTERRUPTS
03BB  B0 20                     MOV     AL,EOI                  ; END OF INTERRUPT COMMAND
03BD  E6 20                     OUT     020H,AL                 ; SEND COMMAND TO INTERRUPT CONTROL PORT
03BF  B0 AE                     MOV     AL,ENA_KBD              ; INSURE KEYBOARD IS ENABLED
03C1  E8 0498 R                 CALL    SHIP_IT                 ; EXECUTE ENABLE
03C4  B8 9102                   MOV     AX,09102H               ; MOVE IN POST CODE & TYPE
03C7  CD 15                     INT     15H                     ; PERFORM OTHER FUNCTION
03C9  E9 01EC R                 JMP     K27A                    ; INTERRUPT_RETURN

                        ;------ TRANSLATE SCAN FOR PSEUDO SCAN CODES

03CC                    K63:                                    ; TRANSLATE-SCAN
03CC  2C 3B                     SUB     AL,59                   ; CONVERT ORIGIN TO FUNCTION KEYS
03CE                    K64:                                    ; TRANSLATE-SCAN-ORGD
03CE  2E: D7                    XLAT    CS:K9                   ; CTL TABLE SCAN
03D0  8A E0                     MOV     AH,AL                   ; PUT VALUE INTO AH
03D2  B0 00                     MOV     AL,0                    ; ZERO ASCII CODE
03D4  EB 9F                     JMP     K57                     ; PUT IT INTO THE BUFFER

03D6                    KB_INT_1        ENDP

03D6  B0 20             K62:    MOV     AL,EOI          ; ENABLE INTR. CTL. CHIP
03D8  E6 20                     OUT     INTA00,AL       ;
03DA  BB 0082                   MOV     BX,82H          ; NUMBER OF CYCLES FOR 1/8 SECOND TONE
03DD  E4 61                     IN      AL,KB_CTL       ; GET CONTROL INFORMATION
03DF  50                        PUSH    AX              ; SAVE
03E0                    K65:                            ; BEEP-CYCLE
03E0  24 FC                     AND     AL,0FCH         ; TURN OFF TIMER GATE AND SPEAKER DATA
03E2  EB 00                     JMP     SHORT $+2       ; IO DELAY
03E4  E6 61                     OUT     KB_CTL,AL       ; OUTPUT TO CONTROL
03E6  B9 00CE                   MOV     CX,0CEH         ; HALF CYCLE TIME FOR TONE
03E9  E2 FE             K66:    LOOP    K66             ; SPEAKER OFF
03EB  0C 02                     OR      AL,2            ; TURN ON SPEAKER BIT
03ED  E6 61                     OUT     KB_CTL,AL       ; OUTPUT TO CONTROL
03EF  B9 00E5                   MOV     CX,0E5H         ; SET UP COUNT
03F2  E2 FE             K67:    LOOP    K67             ; ANOTHER HALF CYCLE
03F4  4B                        DEC     BX              ; TOTAL TIME COUNT
03F5  75 E9                     JNZ     K65             ; DO ANOTHER CYCLE
03F7  58                        POP     AX              ; RECOVER CONTROL
03F8  E6 61                     OUT     KB_CTL,AL       ; OUTPUT THE CONTROL
03FA  E9 01E7 R                 JMP     K27             ; EXIT

                        ;--------------------------------------------------------------------------
                        ;
                        ;       SND_DATA
                        ;
                        ;               THIS ROUTINES HANDLES TRANSMISSION OF COMMAND AND DATA BYTES
                        ;               TO THE KEYBOARD AND RECEIPT OF ACKNOWLEDGEMENTS.  IT ALSO
                        ;               HANDLES ANY RETRIES IF REQUIRED
                        ;
                        ;--------------------------------------------------------------------------

03FD                    SND_DATA PROC   NEAR
03FD  50                        PUSH    AX              ; SAVE REGISTERS
03FE  53                        PUSH    BX              ; *
03FF  51                        PUSH    CX
0400  8A F8                     MOV     BH,AL           ; SAVE TRANSMITTED BY FOR RETRIES
0402  B3 03                     MOV     BL,3            ; LOAD RETRY COUNT
0404  FA                SD0:    CLI                     ; DISABLE INTERRUPTS
0405  80 26 0097 R CF           AND     KB_FLAG_2,NOT (KB_FE+KB_FA) ; CLEAR ACK AND RESEND FLAGS

                        ;------- WAIT FOR COMMAND TO ACCEPTED

040A  2B C9                     SUB     CX,CX           ;
040C                    SD5:
040C  E4 64                     IN      AL,STATUS_PORT  ;
040E  A8 02                     TEST    AL,INPT_BUF_FULL
0410  E0 FA                     LOOPNZ  SD5             ; WAIT FOR COMMAND TO BE ACCEPTED
                        ;
0412  8A C7                     MOV     AL,BH           ; REESTABLISH BYTE TO TRANSMIT
0414  E6 60                     OUT     PORT_A,AL       ; SEND BYTE
0416  FB                        STI                     ; ENABLE INTERRUPTS
0417  B9 1A00                   MOV     CX,01A00H       ; LOAD COUNT FOR 10mS+
```

```
041A F6 06 0097 R 30        SD1:    TEST    KB_FLAG_2,KB_FE+KB_FA ; SEE IF EITHER BIT SET
041F 75 0D                          JNZ     SD3             ; IF SET, SOMETHING RECEIVED GO PROCESS
                            ;
0421 E2 F7                          LOOP    SD1             ; OTHERWISE WAIT
                            ;
0423 FE CB                  SD2:    DEC     BL              ; DECREMENT RETRY COUNT
0425 75 DD                          JNZ     SD0             ; RETRY TRANSMISSION
                            ;
0427 80 0E 0097 R 80                OR      KB_FLAG_2,KB_ERR ; TURN ON TRANSMIT ERROR FLAG
042C EB 07                          JMP     SHORT SD4       ; RETRIES EXHAUSTED FORGET TRANSMISSION
                            ;
042E F6 06 0097 R 10        SD3:    TEST    KB_FLAG_2,KB_FA ; SEE IF THIS IS AN ACKNOWLEDGE
0433 74 EE                          JZ      SD2             ; IF NOT, GO RESEND
                            ;
0435 59                     SD4:    POP     CX              ; RESTORE REGISTERS
0436 5B                             POP     BX              ;
0437 58                             POP     AX              ; *
0438 C3                             RET                     ; RETURN, GOOD TRANSMISSION
0439                        SND_DATA ENDP

                            ;-------------------------------------------------------------------
                            ;
                            ;       SND_LED
                            ;
                            ;               THIS ROUTINES TURNS ON THE MODE INDICATORS.
                            ;
                            ;-------------------------------------------------------------------
0439                        SND_LED PROC    NEAR
0439 FA                             CLI                     ; TURN OFF INTERRUPTS
043A F6 06 0097 R 40                TEST    KB_FLAG_2,KB_PR_LED ; CHECK FOR MODE INDICATOR UPDATE
043F 75 47                          JNZ     SL1             ; DONT UPDATE AGAIN IF UPDATE UNDERWAY
                            ;
0441 80 0E 0097 R 40                OR      KB_FLAG_2,KB_PR_LED ; TURN ON UPDATE IN PROCESS
0446 B0 20                          MOV     AL,EOI          ; END OF INTERRUPT COMMAND
0448 E6 20                          OUT     020H,AL         ; SEND COMMAND TO INTERRUPT CONTROL PORT
044A EB 0D                          JMP     SHORT SL0       ; GO SEND MODE INDICATOR COMMAND
                            ;
044C                        SND_LED1:
044C FA                             CLI                     ; TURN OFF INTERRUPTS
044D F6 06 0097 R 40                TEST    KB_FLAG_2,KB_PR_LED ; CHECK FOR MODE INDICATOR UPDATE
0452 75 34                          JNZ     SL1             ; DONT UPDATE AGAIN IF UPDATE UNDERWAY
                            ;
0454 80 0E 0097 R 40                OR      KB_FLAG_2,KB_PR_LED ; TURN ON UPDATE IN PROCESS
0459 B0 ED                  SL0:    MOV     AL,LED_CMD      ; LED CMD BYTE
045B E8 03FD R                      CALL    SND_DATA        ; SEND DATA TO KEYBOARD
045E FA                             CLI                     ;
045F E8 048A R                      CALL    MAKE_LED        ; GO FORM INDICATOR DATA BYTE
0462 80 26 0097 R F8                AND     KB_FLAG_2,0F8H  ; CLEAR MODE INDICATOR BITS
0467 08 06 0097 R                   OR      KB_FLAG_2,AL    ; SAVE PRESENT INDICATORS STATES FOR NEXT TIME
046B F6 06 0097 R 80                TEST    KB_FLAG_2,KB_ERR ; TRANSMIT ERROR DETECTED
0470 75 0B                          JNZ     SL2             ; IF YES, BYPASS SECOND BYTE TRANSMISSION
                            ;
0472 E8 03FD R                      CALL    SND_DATA        ; SEND DATA TO KEYBOARD
0475 FA                             CLI                     ; TURN OFF INTERRUPTS
0476 F6 06 0097 R 80                TEST    KB_FLAG_2,KB_ERR ; TRANSMIT ERROR DETECTED
047B 74 06                          JZ      SL3             ; IF NOT, DONT SEND AN ENABLE COMMAND
                            ;
047D B0 F4                  SL2:    MOV     AL,KB_ENABLE    ; GET KEYBOARD CSA ENABLE COMMAND
047F E8 03FD R                      CALL    SND_DATA        ; SEND DATA TO KEYBOARD
0482 FA                             CLI                     ; TURN OFF INTERRUPTS
0483 80 26 0097 R 3F        SL3:    AND     KB_FLAG_2,NOT(KB_PR_LED+KB_ERR) ; TURN OFF MODE INDICATOR
                                                            ; UPDATE AND TRANSMIT ERROR FLAG
0488 FB                     SL1:    STI                     ; ENABLE INTERRUPTS
0489 C3                             RET                     ; RETURN TO CALLER
048A                        SND_LED ENDP

                            ;-------------------------------------------------------------------
                            ;
                            ;       MAKE_LED
                            ;
                            ;               THIS ROUTINES FORMS THE DATA BYTE NECESSARY TO TURN ON/OFF
                            ;               THE MODE INDICATORS
                            ;
                            ;-------------------------------------------------------------------
048A                        MAKE_LED PROC   NEAR
048A 51                             PUSH    CX              ; SAVE CX
048B A0 0017 R                      MOV     AL,KB_FLAG      ; GET CAPS & NUM LOCK INDICATORS
048E 24 70                          AND     AL,CAPS_STATE+NUM_STATE+SCROLL_STATE ; ISOLATE INDICATORS
0490 B1 04                          MOV     CL,4            ; SHIFT COUNT
0492 D2 C0                          ROL     AL,CL           ; SHIFT BITS OVER TO TURN ON INDICATORS
0494 24 07                          AND     AL,07H          ; MAKE SURE ONLY MODE BITS ON
0496 59                             POP     CX              ;
0497 C3                             RET                     ; RETURN TO CALLER
0498                        MAKE_LED ENDP
                            ;-------------------------------------------------------------------
                            ;
                            ;       SHIP_IT
                            ;
                            ;               THIS ROUTINES HANDLES TRANSMISSION OF COMMAND AND DATA BYTES
                            ;               TO THE KEYBOARD CONTROLLER.
                            ;
                            ;-------------------------------------------------------------------
0498                        SHIP_IT PROC    NEAR
0498 50                             PUSH    AX              ; SAVE DATA TO SEND
                            ;------- WAIT FOR COMMAND TO ACCEPTED

0499 FA                             CLI                     ; DISABLE INTERRUPTS
049A 2B C9                          SUB     CX,CX           ; CLEAR COUNTER
049C                        SI0:
049C E4 64                          IN      AL,STATUS_PORT  ;
049E A8 02                          TEST    AL,INPT_BUF_FULL
04A0 E0 FA                          LOOPNZ  SI0             ; WAIT FOR COMMAND TO BE ACCEPTED
                            ;
04A2 58                             POP     AX              ; GET DATA TO SEND
04A3 E6 64                          OUT     STATUS_PORT,AL  ; SEND TO KEYBOARD CONTROLLER
04A5 FB                             STI                     ; ENABLE INTERRUPTS AGAIN
04A6 C3                             RET                     ; RETURN TO CALLER
04A7                        SHIP_IT ENDP
04A7                        CODE    ENDS
                                    END
```

SECTION 5

```
                                 TITLE 09/09/83 PRINT BIOS
                                 .LIST
                               C INCLUDE SEGMENT.SRC
0000                           C CODE SEGMENT BYTE PUBLIC
                               C
                                 EXTRN   DDS:NEAR
                                 PUBLIC  PRINTER_IO_1
                                 ;--- INT 17 -------------------------------------------------
                                 ; PRINTER_IO
                                 ;       THIS ROUTINE PROVIDES COMMUNICATION WITH THE PRINTER
                                 ; INPUT
                                 ;       (AH)=0  PRINT THE CHARACTER IN (AL)
                                 ;               ON RETURN, AH=1 IF CHARACTER COULD NOT BE PRINTED (TIME OUT)
                                 ;               OTHER BITS SET AS ON NORMAL STATUS CALL
                                 ;       (AH)=1  INITIALIZE THE PRINTER PORT
                                 ;               RETURNS WITH (AH) SET WITH PRINTER STATUS
                                 ;       (AH)=2  READ THE PRINTER STATUS INTO (AH)
                                 ;               7       6       5       4       3       2-1     0
                                 ;               |       |       |       |       |       |       |_ TIME OUT
                                 ;               |       |       |       |       |       |_ UNUSED
                                 ;               |       |       |       |       |_ 1 = I/O ERROR
                                 ;               |       |       |       |_ 1 = SELECTED
                                 ;               |       |       |_ 1 = OUT OF PAPER
                                 ;               |       |_ 1 = ACKNOWLEDGE
                                 ;               |_ 1 = NOT BUSY
                                 ;
                                 ;       (DX) = PRINTER TO BE USED (0,1,2) CORRESPONDING TO ACTUAL VALUES
                                 ;               IN PRINTER_BASE AREA
                                 ; DATA AREA PRINTER_BASE CONTAINS THE BASE ADDRESS OF THE PRINTER CARD(S)
                                 ; AVAILABLE (LOCATED AT BEGINNING OF DATA SEGMENT, 408H ABSOLUTE, 3 WORDS)
                                 ;
                                 ; DATA AREA PRINT_TIM_OUT (BYTE) MAY BE CHANGE TO CAUSE DIFFERENT
                                 ; TIME OUT WAITS. DEFAULT=20 * 4
                                 ;
                                 ; REGISTERS     AH IS MODIFIED
                                 ;               ALL OTHERS UNCHANGED
                                 ;-------------------------------------------------------------
                                         ASSUME  CS:CODE,DS:DATA

0000                             PRINTER_IO_1    PROC    FAR             ; ENTRY POINT FOR ORG 0EFD2H
0000  FB                                 STI                             ; INTERRUPTS BACK ON
0001  1E                                 PUSH    DS                      ; SAVE SEGMENT
0002  52                                 PUSH    DX
0003  56                                 PUSH    SI
0004  51                                 PUSH    CX
0005  53                                 PUSH    BX
0006  E8 0000 E                          CALL    DDS
0009  8B F2                              MOV     SI,DX                   ; GET PRINTER PARM
000B  8A 9C 0078 R                       MOV     BL,PRINT_TIM_OUT[SI]    ; LOAD TIMEOUT VALUE
000F  D1 E6                              SHL     SI,1                    ; WORD OFFSET INTO TABLE
0011  8B 94 0008 R                       MOV     DX,PRINTER_BASE[SI]     ; GET BASE ADDRESS FOR PRINTER CARD
0015  0B D2                              OR      DX,DX                   ; TEST DX FOR ZERO, INDICATING NO PRINTE
0017  74 0C                              JZ      B1                      ; RETURN
0019  0A E4                              OR      AH,AH                   ; TEST FOR (AH)=0
001B  74 0E                              JZ      B2                      ; PRINT_AL
001D  FE CC                              DEC     AH                      ; TEST FOR (AH)=1
001F  74 54                              JZ      B8                      ; INIT_PRT
0021  FE CC                              DEC     AH                      ; TEST FOR (AH)=2
0023  74 3C                              JZ      B5                      ; PRINTER STATUS
0025                             B1:                                     ; RETURN
0025  5B                                 POP     BX
0026  59                                 POP     CX
0027  5E                                 POP     SI                      ; RECOVER REGISTERS
0028  5A                                 POP     DX                      ; RECOVER REGISTERS
0029  1F                                 POP     DS
002A  CF                                 IRET

                                 ;------ PRINT THE CHARACTER IN (AL)

002B                             B2:
002B  50                                 PUSH    AX              ; SAVE VALUE TO PRINT
002C  EE                                 OUT     DX,AL           ; OUTPUT CHAR TO PORT
002D  42                                 INC     DX              ; POINT TO STATUS PORT

                                 ;------- CHECK FOR PRINTER BUSY

002E  53                                 PUSH    BX              ;
002F  EC                                 IN      AL,DX           ; GET STATUS
0030  A8 80                              TEST    AL,80H          ; IS THE PRINTER CURRENTLY BUSY
0032  75 05                              JNZ     B2_A            ; OUT_STROBE

                                 ;-------- INT 15 DEVICE BUSY

0034  B8 90FE                            MOV     AX,90FEH        ; FUNCTION 90 PRINTER ID
0037  CD 15                              INT     15H             ;

                                 ;------ADJUST OUTTER LOOP COUNT

0039                             B2_A:
0039  2A FF                              SUB     BH,BH           ; CLEAR BH
003B  D1 D3                              RCL     BX,1            ; MULT BY 4
003D  D1 D3                              RCL     BX,1            ;
                                 ;
                                 ;------WAIT BUSY
                                 ;

003F  2B C9                      B3:     SUB     CX,CX           ; INNER LOOP (64K)
0041  EC                         B3_1:   IN      AL,DX           ; GET STATUS
0042  8A E0                              MOV     AH,AL           ; STATUS TO AH ALSO
0044  A8 80                              TEST    AL,80H          ; IS THE PRINTER CURRENTLY BUSY
0046  75 0E                              JNZ     B4              ; OUT_STROBE
0048  E2 F7                              LOOP    B3_1            ; LOOP IF NOT
004A  4B                                 DEC     BX              ; DROP OUTER LOOP COUNT ------
004B  75 F2                              JNZ     B3              ; MAKE ANOTHER PASS IF NOT ZERO

004D  5B                                 POP     BX              ; RESTORE BX -------

004E  80 CC 01                           OR      AH,1            ; SET ERROR FLAG
0051  80 E4 F9                           AND     AH,0F9H         ; TURN OFF THE UNUSED BITS
0054  EB 17                              JMP     SHORT B7        ; RETURN WITH ERROR FLAG SET
0056  5B                         B4:     POP     BX              ; RESTORE BX -------
                                                                 ; OUT_STROBE
0057  B0 0D                              MOV     AL,0DH          ; SET THE STROBE HIGH
0059  42                                 INC     DX
005A  EE                                 OUT     DX,AL
005B  B0 0C                              MOV     AL,0CH          ; SET THE STROBE LOW
005D  EB 00                              JMP     SHORT $+2       ; IO DELAY
005F  EE                                 OUT     DX,AL
0060  58                                 POP     AX              ; RECOVER THE OUTPUT CHAR

                                 ;------ PRINTER STATUS

0061                             B5:
0061  50                                 PUSH    AX              ; SAVE AL REG
0062                             B6:
0062  8B 94 0008 R                       MOV     DX,PRINTER_BASE[SI]
```

SECTION 5

```
0066  42                         INC     DX
0067  EC                         IN      AL,DX           ; GET PRINTER STATUS
0068  8A E0                      MOV     AH,AL
006A  80 E4 F8                   AND     AH,0F8H         ; TURN OFF UNUSED BITS
006D                     B7:                             ; STATUS_SET
006D  5A                         POP     DX              ; RECOVER AL REG
006E  8A C2                      MOV     AL,DL           ; GET CHARACTER INTO AL
0070  80 F4 48                   XOR     AH,48H          ; FLIP A COUPLE OF BITS
0073  EB B0                      JMP     B1              ; RETURN FROM ROUTINE

                         ;------ INITIALIZE THE PRINTER PORT

0075                     B8:
0075  50                         PUSH    AX              ; SAVE AL
0076  42                         INC     DX              ; POINT TO OUTPUT PORT
0077  42                         INC     DX
0078  B0 08                      MOV     AL,8            ; SET INIT LINE LOW
007A  EE                         OUT     DX,AL
007B  B8 0FA0                    MOV     AX,1000*4       ; -------
007E                     B9:                             ; INIT_LOOP
007E  48                         DEC     AX              ; LOOP FOR RESET TO TAKE
007F  75 FD                      JNZ     B9              ; INIT_LOOP
0081  B0 0C                      MOV     AL,0CH          ; NO INTERRUPTS, NON AUTO LF, INIT HIGH
0083  EE                         OUT     DX,AL
0084  EB DC                      JMP     B6              ; PRT_STATUS_1
0086                     PRINTER_IO_1    ENDP
0086                             CODE    ENDS
                                         END
```

```
                              TITLE DATE 07/06/83 RS232
                              .LIST
                            C INCLUDE SEGMENT.SRC
0000                        C CODE SEGMENT BYTE PUBLIC
                            C
                              EXTRN    DDS:NEAR
                              EXTRN    A1:NEAR
                              PUBLIC   RS232_IO_1

                              ;-----INT 14-------------------------------------------
                              ;RS232_IO
                              ;       THIS ROUTINE PROVIDES BYTE STREAM I/O TO THE COMMUNICATIONS
                              ;       PORT ACCORDING TO THE PARAMETERS:
                              ;       (AH)=0  INITIALIZE THE COMMUNICATIONS PORT
                              ;               (AL) HAS PARMS FOR INITIALIZATION
                              ;
                              ;               7       6       5       4       3       2         1       0
                              ;               ----- BAUD RATE --      -PARITY--       STOPBIT   --WORD LENGTH--
                              ;
                              ;               000 - 110               X0 - NONE       0 - 1     10 - 7 BITS
                              ;               001 - 150               01 - ODD        1 - 2     11 - 8 BITS
                              ;               010 - 300               11 - EVEN
                              ;               011 - 600
                              ;               100 - 1200
                              ;               101 - 2400
                              ;               110 - 4800
                              ;               111 - 9600
                              ;               ON RETURN, CONDITIONS SET AS IN CALL TO COMMO STATUS (AH=3)
                              ;       (AH)=1  SEND THE CHARACTER IN (AL) OVER THE COMMO LINE
                              ;               (AL) REGISTER IS PRESERVED
                              ;               ON EXIT, BIT 7 OF AH IS SET IF THE ROUTINE WAS UNABLE TO
                              ;                       TO TRANSMIT THE BYTE OF DATA OVER THE LINE.
                              ;                       IF BIT 7 OF AH IS NOT SET, THE
                              ;                       REMAINDER OF AH IS SET AS IN A STATUS REQUEST,
                              ;                       REFELECTING THE CURRENT STATUS OF THE LINE.
                              ;       (AH)=2  RECEIVE A CHARACTER IN (AL) FROM COMMO LINE BEFORE
                              ;                       RETURNING TO CALLER
                              ;               ON EXIT, AH HAS THE CURRENT LINE STATUS, AS SET BY THE
                              ;                       THE STATUS ROUTINE, EXCEPT THAT THE ONLY BITS
                              ;                       LEFT ON ARE THE ERROR BITS (7,4,3,2,1)
                              ;                       IF AH HAS BIT 7 ON (TIME OUT) THE REMAINING
                              ;                       BITS ARE NOT PREDICTABLE.
                              ;                       THUS, AH IS NON ZERO ONLY WHEN AN ERROR OCCURRED.
                              ;       (AH)=3  RETURN THE COMMO PORT STATUS IN (AX)
                              ;               AH CONTAINS THE LINE CONTROL STATUS
                              ;               BIT 7 = TIME OUT
                              ;               BIT 6 = TRANS SHIFT REGISTER EMPTY
                              ;               BIT 5 = TRAN HOLDING REGISTER EMPTY
                              ;               BIT 4 = BREAK DETECT
                              ;               BIT 3 = FRAMING ERROR
                              ;               BIT 2 = PARITY ERROR
                              ;               BIT 1 = OVERRUN ERROR
                              ;               BIT 0 = DATA READY
                              ;               AL CONTAINS THE MODEM STATUS
                              ;               BIT 7 = RECEVED LINE SIGNAL DETECT
                              ;               BIT 6 = RING INDICATOR
                              ;               BIT 5 = DATA SET READY
                              ;               BIT 4 = CLEAR TO SEND
                              ;               BIT 3 = DELTA RECEIVE LINE SIGNAL DETECT
                              ;               BIT 2 = TRAILING EDGE RING DETECTOR
                              ;               BIT 1 = DELTA DATA SET READY
                              ;               BIT 0 = DELTA CLEAR TO SEND
                              ;
                              ;       (DX) = PARAMETER INDICATING WHICH RS232 CARD (0,1 ALLOWED)
                              ;
                              ; DATA AREA RS232_BASE CONTAINS THE BASE ADDRESS OF THE 8250 ON THE CARD
                              ;       LOCATION 400H CONTAINS UP TO 4 RS232 ADDRESSES POSSIBLE
                              ;       DATA AREA LABLE RS232_TIM_OUT (BYTE) CONTAINS OUTER LOOP COUNT
                              ;       VALUE FOR TIMEOUT (DEFAULT=1)
                              ;OUTPUT
                              ;               AX MODIFIED ACCORDING TO PARMS OF CALL
                              ;               ALL OTHERS UNCHANGED
                              ;-----------------------------------------------
                                      ASSUME  CS:CODE,DS:DATA

0000                          RS232_IO_1      PROC    FAR

                              ;------ VECTOR TO APPROPRIATE ROUTINE

0000  FB                              STI                             ; INTERRUPTS BACK ON
0001  1E                              PUSH    DS                      ; SAVE SEGMENT
0002  52                              PUSH    DX
0003  56                              PUSH    SI
0004  57                              PUSH    DI
0005  51                              PUSH    CX
0006  53                              PUSH    BX
0007  8B F2                           MOV     SI,DX                   ; RS232 VALUE TO SI
0009  8B FA                           MOV     DI,DX                   ; AND TO DI (FOR TIMEOUTS)
000B  D1 E6                           SHL     SI,1                    ; WORD OFFSET
000D  E8 0000 E                       CALL    DDS
0010  8B 94 0000 R                    MOV     DX,RS232_BASE[SI]       ; GET BASE ADDRESS
0014  0B D2                           OR      DX,DX                   ; TEST FOR 0 BASE ADDRESS
0016  74 13                           JZ      A3                      ; RETURN
0018  0A E4                           OR      AH,AH                   ; TEST FOR (AH)=0
001A  74 16                           JZ      A4                      ; COMMUN INIT
001C  FE CC                           DEC     AH                      ; TEST FOR (AH)=1
001E  74 4B                           JZ      A5                      ; SEND AL
0020  FE CC                           DEC     AH                      ; TEST FOR (AH)=2
0022  74 70                           JZ      A12                     ; RECEIVE INTO AL
0024                          A2:
0024  FE CC                           DEC     AH                      ; TEST FOR (AH)=3
0026  75 03                           JNZ     A3
0028  E9 00B6 R                       JMP     A18                     ; COMMUNICATION STATUS
002B                          A3:                                     ; RETURN FROM RS232
002B  5B                              POP     BX
002C  59                              POP     CX
002D  5F                              POP     DI
002E  5E                              POP     SI
002F  5A                              POP     DX
0030  1F                              POP     DS
0031  CF                              IRET                            ; RETURN TO CALLER, NO ACTION

                              ;------ INITIALIZE THE COMMUNICATIONS PORT

0032                          A4:
0032  8A E0                           MOV     AH,AL                   ; SAVE INIT PARMS IN AH
0034  83 C2 03                        ADD     DX,3                    ; POINT TO 8250 CONTROL REGISTER
0037  B0 80                           MOV     AL,80H
0039  EE                              OUT     DX,AL                   ; SET DLAB=1

                              ;------ DETERMINE BAUD RATE DIVISOR

003A  8A D4                           MOV     DL,AH                   ; GET PARMS TO DL
003C  B1 04                           MOV     CL,4
003E  D2 C2                           ROL     DL,CL
0040  81 E2 000E                      AND     DX,0EH                  ; ISOLATE THEM
```

SECTION 5

```
0044  BF 0000 E                    MOV    DI,OFFSET A1           ; BASE OF TABLE
0047  03 FA                        ADD    DI,DX                  ; PUT INTO INDEX REGISTER
0049  8B 94 0000 R                 MOV    DX,RS232_BASE[SI]      ; POINT TO HIGH ORDER OF DIVISOR
004D  42                           INC    DX
004E  2E: 8A 45 01                 MOV    AL,CS:[DI]+1           ; GET HIGH ORDER OF DIVISOR
0052  EE                           OUT    DX,AL                  ; SET MS OF DIV TO 0
0053  4A                           DEC    DX
0054  EB 00                        JMP    SHORT $+2              ; IO DELAY
0056  2E: 8A 05                    MOV    AL,CS:[DI]             ; GET LOW ORDER OF DIVISOR
0059  EE                           OUT    DX,AL                  ; SET LOW OF DIVISOR
005A  83 C2 03                     ADD    DX,3
005D  8A C4                        MOV    AL,AH                  ; GET PARMS BACK
005F  24 1F                        AND    AL,01FH                ; STRIP OFF THE BAUD BITS
0061  EE                           OUT    DX,AL                  ; LINE CONTROL TO 8 BITS
0062  4A                           DEC    DX
0063  4A                           DEC    DX
0064  EB 00                        JMP    SHORT $+2              ; IO DELAY
0066  B0 00                        MOV    AL,0
0068  EE                           OUT    DX,AL                  ; INTERRUPT ENABLES ALL OFF
0069  EB 4B                        JMP    SHORT A18              ; COM_STATUS

                             ;------ SEND CHARACTER IN (AL) OVER COMMO LINE

006B                         A5:
006B  50                           PUSH   AX                     ; SAVE CHAR TO SEND
006C  83 C2 04                     ADD    DX,4                   ; MODEM CONTROL REGISTER
006F  B0 03                        MOV    AL,3                   ; DTR AND RTS
0071  EE                           OUT    DX,AL                  ; DATA TERMINAL READY, REQUEST TO SEND
0072  42                           INC    DX                     ; MODEM STATUS REGISTER
0073  42                           INC    DX
0074  B7 30                        MOV    BH,30H                 ; DATA SET READY & CLEAR TO SEND
0076  E8 00C5 R                    CALL   WAIT_FOR_STATUS        ; ARE BOTH TRUE
0079  74 08                        JE     A9                     ; YES, READY TO TRANSMIT CHAR
007B                         A7:
007B  59                           POP    CX
007C  8A C1                        MOV    AL,CL                  ; RELOAD DATA BYTE
007E                         A8:
007E  80 CC 80                     OR     AH,80H                 ; INDICATE TIME OUT
0081  EB A8                        JMP    A3                     ; RETURN

0083                         A9:                                 ; CLEAR_TO_SEND
0083  4A                           DEC    DX                     ; LINE STATUS REGISTER
0084                         A10:                                ; WAIT_SEND
0084  B7 20                        MOV    BH,20H                 ; IS TRANSMITTER READY
0086  E8 00C5 R                    CALL   WAIT_FOR_STATUS        ; TEST FOR TRANSMITTER READU
0089  75 F0                        JNZ    A7                     ; RETURN WITH TIME OUT SET
008B                         A11:                                ; OUT_CHAR
008B  83 EA 05                     SUB    DX,5                   ; DATA PORT
008E  59                           POP    CX                     ; RECOVER IN CX TEMPORARILY
008F  8A C1                        MOV    AL,CL                  ; MOVE CHAR TO AL FOR OUT, STATUS IN AH
0091  EE                           OUT    DX,AL                  ; OUTPUT CHARACTER
0092  EB 97                        JMP    A3                     ; RETURN

                             ;------ RECEIVE CHARACTER FROM COMMO LINE

0094                         A12:
0094  83 C2 04                     ADD    DX,4                   ; MODEM CONTROL REGISTER
0097  B0 01                        MOV    AL,1                   ; DATA TERMINAL READY
0099  EE                           OUT    DX,AL
009A  42                           INC    DX                     ; MODEM STATUS REGISTER
009B  42                           INC    DX
009C                         A13:                                ; WAIT_DSR
009C  B7 20                        MOV    BH,20H                 ; DATA SET READY
009E  E8 00C5 R                    CALL   WAIT_FOR_STATUS        ; TEST FOR DSR
00A1  75 DB                        JNZ    A8                     ; RETURN WITH ERROR
00A3                         A15:                                ; WAIT_DSR_END
00A3  4A                           DEC    DX                     ; LINE STATUS REGISTER
00A4                         A16:                                ; WAIT_RECV
00A4  B7 01                        MOV    BH,1                   ; RECEIVE BUFFER FULL
00A6  E8 00C5 R                    CALL   WAIT_FOR_STATUS        ; TEST FOR REC. BUFF. FULL
00A9  75 D3                        JNZ    A8                     ; SET TIME OUT ERROR
00AB                         A17:                                ; GET_CHAR
00AB  80 E4 1E                     AND    AH,00011110B           ; TEST FOR ERROR CONDITIONS ON RECV CHAR
00AE  8B 94 0000 R                 MOV    DX,RS232_BASE[SI]      ; DATA PORT
00B2  EC                           IN     AL,DX                  ; GET CHARACTER FROM LINE
00B3  E9 002B R                    JMP    A3                     ; RETURN

                             ;------ COMMO PORT STATUS ROUTINE

00B6                         A18:
00B6  8B 94 0000 R                 MOV    DX,RS232_BASE[SI]
00BA  83 C2 05                     ADD    DX,5                   ; CONTROL PORT
00BD  EC                           IN     AL,DX                  ; GET LINE CONTROL STATUS
00BE  8A E0                        MOV    AH,AL                  ; PUT IN AH FOR RETURN
00C0  42                           INC    DX                     ; POINT TO MODEM STATUS REGISTER
00C1  EC                           IN     AL,DX                  ; GET MODEM CONTROL STATUS
00C2  E9 002B R                    JMP    A3                     ; RETURN
                             ;----------------------------------------
                             ;      WAIT FOR STATUS ROUTINE
                             ;ENTRY: BH=STATUS BIT(S) TO LOOK FOR,
                             ;      DX=ADDR. OF STATUS REG
                             ;EXIT:  ZERO FLAG ON = STATUS FOUND
                             ;      ZERO FLAG OFF = TIMEOUT.
                             ;      AH=LAST STATUS READ
                             ;----------------------------------------
00C5                         WAIT_FOR_STATUS PROC    NEAR
00C5  8A 9D 007C R                 MOV    BL,RS232_TIM_OUT[DI]   ;LOAD OUTER LOOP COUNT

                             ;
                             ;------ADJUST OUTTER LOOP COUNT
                             ;
00C9  55                           PUSH   BP              ; SAVE BP ------
00CA  53                           PUSH   BX              ; SAVE BX ------
00CB  5D                           POP    BP              ; USE BP FOR OUTTER LOOP COUNT
00CC  81 E5 00FF                   AND    BP,00FFH        ; STRIP HIGH BITS
00D0  D1 D5                        RCL    BP,1            ; MULT OUTTER BY 4
00D2  D1 D5                        RCL    BP,1            ;

00D4  2B C9                  WFS0: SUB    CX,CX
00D6  EC                     WFS1: IN     AL,DX                  ;GET STATUS
00D7  8A E0                        MOV    AH,AL                  ; MOVE TO AH
00D9  22 C7                        AND    AL,BH                  ;ISOLATE BITS TO TEST
00DB  3A C7                        CMP    AL,BH                  ;EXACTLY = TO MASK
00DD  74 07                        JE     WFS_END                ;RETURN WITH ZERO FLAG ON
00DF  E2 F5                        LOOP   WFS1                   ;TRY AGAIN
00E1  4D                           DEC    BP                     ; -----
00E2  75 F0                        JNZ    WFS0                   ;
00E4  0A FF                        OR     BH,BH                  ;SET ZERO FLAG OFF
00E6                         WFS_END:
00E6  5D                           POP    BP                     ; RESTORE BP -----
00E7  C3                           RET                           ;
00E8                         WAIT_FOR_STATUS ENDP
00E8                         RS232_IO_1      ENDP

00E8                         CODE   ENDS
```

```
                        TITLE 08/18/83 VIDEO1
                        .LIST
                        ;
                        ;
                        ;  includes are  postequ.src,  dseg.src
                        ;
                      C  INCLUDE SEGMENT.SRC
0000                  C  CODE SEGMENT BYTE PUBLIC
                      C
                        EXTRN   DDS:NEAR
                        EXTRN   M5:WORD
                        EXTRN   M6:BYTE
                        EXTRN   M7:BYTE
                        EXTRN   CRT_CHAR_GEN:NEAR
                        EXTRN   BEEP:NEAR

                        PUBLIC  VIDEO_IO_1
= 0010                  M4      EQU     0010H           ;

                        ;--- INT 10 ------------------------------------------------------------
                        ; VIDEO_IO
                        ;       THESE ROUTINES PROVIDE THE CRT INTERFACE
                        ;       THE FOLLOWING FUNCTIONS ARE PROVIDED:
                        ;       (AH)=0  SET MODE (AL) CONTAINS MODE VALUE
                        ;               (AL)=0 40X25 BW (POWER ON DEFAULT)
                        ;               (AL)=1 40X25 COLOR
                        ;               (AL)=2  80X25 BW
                        ;               (AL)=3  80X25 COLOR
                        ;               GRAPHICS MODES
                        ;               (AL)=4  320X200 COLOR
                        ;               (AL)=5  320X200 BW
                        ;               (AL)=6  640X200 BW
                        ;               CRT MODE = 7 80X25 B&W CARD (USED INTERNAL TO VIDEO ONLY)
                        ;               *** NOTES -BW MODES OPERATE SAME AS COLOR MODES, BUT COLOR
                        ;                          BURST IS NOT ENABLED
                        ;                         -CURSOR IS NOT DISPLAYED IN GRAPHICS MODE
                        ;       (AH)=1  SET CURSOR TYPE
                        ;               (CH) =  BITS 4-0 = START LINE FOR CURSOR
                        ;                       ** HARDWARE WILL ALWAYS CAUSE BLINK
                        ;                       ** SETTING BIT 5 OR 6 WILL CAUSE ERRATIC BLINKING
                        ;                          OR NO CURSOR AT ALL
                        ;               (CL) =  BITS 4-0 = END LINE FOR CURSOR
                        ;       (AH)=2  SET CURSOR POSITION
                        ;               (DH,DL) = ROW,COLUMN  (0,0) IS UPPER LEFT
                        ;               (BH) = PAGE NUMBER (MUST BE 0 FOR GRAPHICS MODES)
                        ;       (AH)=3  READ CURSOR POSITION
                        ;               (BH) = PAGE NUMBER (MUST BE 0 FOR GRAPHICS MODES)
                        ;               ON EXIT (DH,DL) = ROW,COLUMN OF CURRENT CURSOR
                        ;                       (CH,CL) = CURSOR MODE CURRENTLY SET
                        ;       (AH)=4  READ LIGHT PEN POSITION
                        ;               ON EXIT:
                        ;               (AH) = 0 -- LIGHT PEN SWITCH NOT DOWN/NOT TRIGGERED
                        ;               (AH) = 1 -- VALID LIGHT PEN VALUE IN REGISTERS
                        ;                       (DH,DL) = ROW,COLUMN OF CHARACTER LP POSN
                        ;                       (CH) = RASTER LINE (0-199)
                        ;                       (BX) = PIXEL COLUMN (0-319,639)
                        ;       (AH)=5  SELECT ACTIVE DISPLAY PAGE (VALID ONLY FOR ALPHA MODES)
                        ;               (AL)=NEW PAGE VALUE (0-7 FOR MODES 0&1, 0-3 FOR MODES 2&3)
                        ;       (AH)=6  SCROLL ACTIVE PAGE UP
                        ;               (AL) = NUMBER OF LINES, INPUT LINES BLANKED AT BOTTOM OF WINDOW
                        ;                       AL = 0 MEANS BLANK ENTIRE WINDOW
                        ;               (CH,CL) = ROW,COLUMN OF UPPER LEFT CORNER OF SCROLL
                        ;               (DH,DL) = ROW,COLUMN OF LOWER RIGHT CORNER OF SCROLL
                        ;               (BH) = ATTRIBUTE TO BE USED ON BLANK LINE
                        ;       (AH)=7  SCROLL ACTIVE PAGE DOWN
                        ;               (AL) = NUMBER OF LINES, INPUT LINES BLANKED AT TOP OF WINDOW
                        ;                       AL = 0 MEANS BLANK ENTIRE WINDOW
                        ;               (CH,CL) = ROW,COLUMN OF UPPER LEFT CORNER OF SCROLL
                        ;               (DH,DL) = ROW,COLUMN OF LOWER RIGHT CORNER OF SCROLL
                        ;               (BH) = ATTRIBUTE TO BE USED ON BLANK LINE
                        ;
                        ;       CHARACTER HANDLING ROUTINES
                        ;
                        ;       (AH) = 8 READ ATTRIBUTE/CHARACTER AT CURRENT CURSOR POSITION
                        ;               (BH) = DISPLAY PAGE (VALID FOR ALPHA MODES ONLY)
                        ;               ON EXIT:
                        ;               (AL) = CHAR READ
                        ;               (AH) = ATTRIBUTE OF CHARACTER READ (ALPHA MODES ONLY)
                        ;       (AH) = 9 WRITE ATTRIBUTE/CHARACTER AT CURRENT CURSOR POSITION
                        ;               (BH) = DISPLAY PAGE (VALID FOR ALPHA MODES ONLY)
                        ;               (CX) = COUNT OF CHARACTERS TO WRITE
                        ;               (AL) = CHAR TO WRITE
                        ;               (BL) = ATTRIBUTE OF CHARACTER (ALPHA)/COLOR OF CHAR (GRAPHICS)
                        ;                       SEE NOTE ON WRITE DOT FOR BIT 7 OF BL = 1.
                        ;       (AH) = 10 WRITE CHARACTER ONLY AT CURRENT CURSOR POSITION
                        ;               (BH) = DISPLAY PAGE (VALID FOR ALPHA MODES ONLY)
                        ;               (CX) = COUNT OF CHARACTERS TO WRITE
                        ;               (AL) = CHAR TO WRITE
                        ;       FOR READ/WRITE CHARACTER INTERFACE WHILE IN GRAPHICS MODE, THE
                        ;               CHARACTERS ARE FORMED FROM A CHARACTER GENERATOR IMAGE
                        ;               MAINTAINED IN THE SYSTEM ROM.  ONLY THE 1ST 128 CHARS
                        ;               ARE CONTAINED THERE.  TO READ/WRITE THE SECOND 128 CHARS,
                        ;               THE USER MUST INITIALIZE THE POINTER AT INTERRUPT 1FH
                        ;               (LOCATION 0007CH) TO POINT TO THE 1K BYTE TABLE CONTAINING
                        ;               THE CODE POINTS FOR THE SECOND 128 CHARS (128-255).
                        ;       FOR WRITE CHARACTER INTERFACE IN GRAPHICS MODE, THE REPLICATION FACTOR
                        ;               CONTAINED IN (CX) ON ENTRY WILL PRODUCE VALID RESULTS ONLY
                        ;               FOR CHARACTERS CONTAINED ON THE SAME ROW.  CONTINUATION TO
                        ;               SUCCEEDING LINES WILL NOT PRODUCE CORRECTLY.
                        ;
                        ;       GRAPHICS INTERFACE
                        ;       (AH) = 11 SET COLOR PALETTE
                        ;               (BH) = PALLETTE COLOR ID BEING SET (0-127)
                        ;               (BL) = COLOR VALUE TO BE USED WITH THAT COLOR ID
                        ;                  NOTE: FOR THE CURRENT COLOR CARD, THIS ENTRY POINT HAS
                        ;                          MEANING ONLY FOR 320X200 GRAPHICS.
                        ;                       COLOR ID = 0 SELECTS THE BACKGROUND COLOR (0-15)
                        ;                       COLOR ID = 1 SELECTS THE PALLETTE TO BE USED:
                        ;                               0 = GREEN(1)/RED(2)/YELLOW(3)
                        ;                               1 = CYAN(1)/MAGENTA(2)/WHITE(3)
                        ;                       IN 40X25 OR 80X25 ALPHA MODES, THE VALUE SET FOR
                        ;                               PALLETTE COLOR 0 INDICATES THE BORDER COLOR
                        ;                               TO BE USED (VALUES 0-31, WHERE 16-31 SELECT THE
                        ;                               HIGH INTENSITY BACKGROUND SET.
                        ;       (AH) = 12 WRITE DOT
                        ;               (DX) = ROW NUMBER
                        ;               (CX) = COLUMN NUMBER
                        ;               (AL) = COLOR VALUE
                        ;                       IF BIT 7 OF AL = 1, THEN THE COLOR VALUE IS EXCLUSIVE
                        ;                       OR'D WITH THE CURRENT CONTENTS OF THE DOT
                        ;       (AH) = 13 READ DOT
                        ;               (DX) = ROW NUMBER
                        ;               (CX) = COLUMN NUMBER
                        ;               (AL) RETURNS THE DOT READ
```

SECTION 5

```
                                ;
                                ; ASCII TELETYPE ROUTINE FOR OUTPUT
                                ;
                                ;       (AH) = 14 WRITE TELETYPE TO ACTIVE PAGE
                                ;               (AL) = CHAR TO WRITE
                                ;               (BL) = FOREGROUND COLOR IN GRAPHICS MODE
                                ;               NOTE -- SCREEN WIDTH IS CONTROLLED BY PREVIOUS MODE SET
                                ;
                                ;       (AH) = 15 CURRENT VIDEO STATE
                                ;               RETURNS THE CURRENT VIDEO STATE
                                ;               (AL) = MODE CURRENTLY SET ( SEE AH=0 FOR EXPLANATION)
                                ;               (AH) = NUMBER OF CHARACTER COLUMNS ON SCREEN
                                ;               (BH) = CURRENT ACTIVE DISPLAY PAGE
                                ;
                                ;
                                ;       (AH) = 16 RESERVED
                                ;       (AH) = 17 RESERVED
                                ;       (AH) = 18 RESERVED
                                ;
                                ;       (AH) = 19 WRITE STRING
                                ;
                                ;                       ES:BP   -  POINTER TO STRING TO BE WRITTEN
                                ;                       CX      -  LENGTH OF CHARACER STRING TO WRITTEN
                                ;                       DX      -  CURSOR POSITION FOR STRING TO BE WRITTEN
                                ;                       BH      -  PAGE NUMBER
                                ;
                                ;               (AL) = 0
                                ;                       BL      -  ATTRIBUTE
                                ;                       STRING IS  {CHAR,CHAR, ... ,CHAR}
                                ;                       CURSOR NOT MOVED
                                ;               (AL) = 1
                                ;                       BL      -  ATTRIBUTE
                                ;                       STRING IS  {CHAR,CHAR,  ... ,CHAR}
                                ;                       CURSOR IS MOVED
                                ;               (AL) = 2
                                ;                       STRING IS  {CHAR,ATTR,CHAR,ATTR ..  ,CHAR,ATTR}
                                ;                       CURSOR IS NOT MOVED
                                ;               (AL) = 3
                                ;                       STRING IS  {CHAR,ATTR,CHAR,ATTR ..  ,CHAR,ATTR}
                                ;                       CURSOR IS MOVED
                                ;
                                ;               NOTE:  CARRIAGE RETURN, LINE FEED, BACKSPACE, AND BELL ARE
                                ;                      TREATED AS COMMANDS RATHER THAN PRINTABLE CHARACTERS.
                                ;
                                ;       SS,SP,ES,DS,DX,CX,BX,SI,DI,BP PRESERVED DURING CALL
                                ;       ALL OTHERS DESTROYED.
                                ;-------------------------------------------------------
                                        ASSUME  CS:CODE,DS:DATA,ES:VIDEO_RAM

                                PUBLIC SET_MODE
                                PUBLIC SET_CTYPE
                                PUBLIC SET_CPOS
                                PUBLIC READ_CURSOR
                                PUBLIC READ_LPEN
                                PUBLIC ACT_DISP_PAGE
                                PUBLIC SCROLL_UP
                                PUBLIC SCROLL_DOWN
                                PUBLIC READ_AC_CURRENT
                                PUBLIC WRITE_AC_CURRENT
                                PUBLIC WRITE_C_CURRENT
                                PUBLIC SET_COLOR
                                PUBLIC WRITE_DOT
                                PUBLIC READ_DOT
                                PUBLIC WRITE_TTY
                                PUBLIC VIDEO_STATE
0000                            M1      LABEL   WORD    ;  TABLE OF ROUTINES WITHIN VIDEO I/O
0000  0071 R                            DW      OFFSET  SET_MODE
0002  014D R                            DW      OFFSET  SET_CTYPE
0004  0174 R                            DW      OFFSET  SET_CPOS
0006  019E R                            DW      OFFSET  READ_CURSOR
0008  07DF R                            DW      OFFSET  READ_LPEN
000A  01B5 R                            DW      OFFSET  ACT_DISP_PAGE
000C  0222 R                            DW      OFFSET  SCROLL_UP
000E  02C6 R                            DW      OFFSET  SCROLL_DOWN
0010  0318 R                            DW      OFFSET  READ_AC_CURRENT
0012  035E R                            DW      OFFSET  WRITE_AC_CURRENT
0014  0391 R                            DW      OFFSET  WRITE_C_CURRENT
0016  01D9 R                            DW      OFFSET  SET_COLOR
0018  046F R                            DW      OFFSET  WRITE_DOT
001A  045E R                            DW      OFFSET  READ_DOT
001C  075B R                            DW      OFFSET  WRITE_TTY
001E  01FF R                            DW      OFFSET  VIDEO_STATE
0020  0144 R                            DW      OFFSET  VIDEO_RETURN            ; Reserved
0022  0144 R                            DW      OFFSET  VIDEO_RETURN            ; Reserved
0024  0144 R                            DW      OFFSET  VIDEO_RETURN            ; Reserved
0026  03C3 R                            DW      OFFSET  WRITE_STRING            ; CASE 19h, Write string
= 0028                          M1L     EQU     $-M1


0028                            VIDEO_IO_1      PROC    NEAR    ; ENTRY POINT FOR ORG 0F065H
0028  FB                                STI                     ; INTERRUPTS BACK ON
0029  FC                                CLD                     ; SET DIRECTION FORWARD
002A  06                                PUSH    ES
002B  1E                                PUSH    DS              ; SAVE SEGMENT REGISTERS
002C  52                                PUSH    DX
002D  51                                PUSH    CX
002E  53                                PUSH    BX
002F  56                                PUSH    SI
0030  57                                PUSH    DI
0031  55                                PUSH    BP
0032  50                                PUSH    AX              ; SAVE AX VALUE
0033  8A C4                             MOV     AL,AH           ; GET INTO LOW BYTE
0035  32 E4                             XOR     AH,AH           ; ZERO TO HIGH BYTE
0037  D1 E0                             SAL     AX,1            ; *2 FOR TABLE LOOKUP
0039  8B F0                             MOV     SI,AX           ; PUT INTO SI FOR BRANCH
003B  3D 0028                           CMP     AX,M1L          ; TEST FOR WITHIN RANGE
003E  72 04                             JB      M2              ; BRANCH AROUND BRANCH
0040  58                                POP     AX              ; THROW AWAY THE PARAMETER
0041  E9 0144 R                         JMP     VIDEO_RETURN            ; DO NOTHING IF NOT IN RANGE
0044                            M2:
0044  E8 0000 E                         CALL    DDS
0047  B8 B800                           MOV     AX,0B800H       ; SEGMENT FOR COLOR CARD
004A  8B 3E 0010 R                      MOV     DI,EQUIP_FLAG   ; GET EQUIPMENT SETTING
004E  81 E7 0030                        AND     DI,30H          ; ISOLATE CRT SWITCHES
0052  83 FF 30                          CMP     DI,30H          ; IS SETTING FOR BW CARD?
0055  75 02                             JNE     M3
0057  B4 B0                             MOV     AH,0B0H         ; SEGMENT FOR BW CARD
0059  8E C0                     M3:     MOV     ES,AX           ; SET UP TO POINT AT VIDEO RAM AREAS
005B  58                                POP     AX              ; RECOVER VALUE

005C  80 FC 13                          CMP     AH,13H          ; TEST FOR WRITE STRING OP
005F  75 07                             JNE     MM3
0061  55                                PUSH    BP              ; IF IT'S WRITE STRING THEN GET THE
0062  8B EC                             MOV     BP,SP           ; STRINGS SEGMENT, SINCE IT GET CLOBBERED
0064  8E 46 10                          MOV     ES,[BP].ES_POS  ;
```

```
0067  5D                            POP     BP              ;
0068                        MM3:
0068  8A 26 0049 R                  MOV     AH,CRT_MODE     ; GET CURRENT MODE INTO AH
006C  2E: FF A4 0000 R              JMP     WORD PTR CS:[SI+OFFSET M1]
0071                        VIDEO_IO_1      ENDP
                            ;----------------------------------------
                            ; SET_MODE
                            ;       THIS ROUTINE INITIALIZES THE ATTACHMENT TO
                            ;       THE SELECTED MODE.  THE SCREEN IS BLANKED.
                            ; INPUT
                            ;       (AL) = MODE SELECTED (RANGE 0-9)
                            ; OUTPUT
                            ;       NONE
                            ;----------------------------------------

0071                        SET_MODE        PROC    NEAR
0071  BA 03D4                       MOV     DX,03D4H                ; ADDRESS OF COLOR CARD
0074  B3 00                         MOV     BL,0                    ; MODE SET FOR COLOR CARD
0076  83 FF 30                      CMP     DI,30H                  ; IS BW CARD INSTALLED
0079  75 07                         JNE     M8                      ; OK WITH COLOR
007B  B0 07                         MOV     AL,7                    ; INDICATE BW CARD MODE
007D  BA 03B4                       MOV     DX,03B4H                ; ADDRESS OF BW CARD
0080  FE C3                         INC     BL                      ; MODE SET FOR BW CARD
0082  8A E0                 M8:     MOV     AH,AL           ; SAVE MODE IN AH
0084  A2 0049 R                     MOV     CRT_MODE,AL     ; SAVE IN GLOBAL VARIABLE
0087  89 16 0063 R                  MOV     ADDR_6845,DX    ; SAVE ADDRESS OF BASE
008B  1E                            PUSH    DS              ; SAVE POINTER TO DATA SEGMENT
008C  50                            PUSH    AX              ; SAVE MODE
008D  52                            PUSH    DX              ; SAVE OUTPUT PORT VALUE
008E  83 C2 04                      ADD     DX,4            ; POINT TO CONTROL REGISTER
0091  8A C3                         MOV     AL,BL           ; GET MODE SET FOR CARD
0093  EE                            OUT     DX,AL           ; RESET VIDEO
0094  5A                            POP     DX              ; BACK TO BASE REGISTER
0095  2B C0                         SUB     AX,AX           ; SET UP FOR ABSO SEGMENT
0097  8E D8                         MOV     DS,AX           ; ESTABLISH VECTOR TABLE ADDRESSING
                                    ASSUME  DS:ABSO
0099  C5 1E 0074 R                  LDS     BX,PARM_PTR     ; GET POINTER TO VIDEO PARMS
009D  58                            POP     AX              ; RECOVER PARMS
                                    ASSUME  DS:CODE
009E  B9 0010                       MOV     CX,M4           ; LENGTH OF EACH ROW OF TABLE
00A1  80 FC 02                      CMP     AH,2            ; DETERMINE WHICH ONE TO USE
00A4  72 10                         JC      M9                      ; MODE IS 0 OR 1
00A6  03 D9                         ADD     BX,CX                   ; MOVE TO NEXT ROW OF INIT TABLE
00A8  80 FC 04                      CMP     AH,4
00AB  72 09                         JC      M9                      ; MODE IS 2 OR 3
00AD  03 D9                         ADD     BX,CX                   ; MOVE TO GRAPHICS ROW OF INIT_TABLE
00AF  80 FC 07                      CMP     AH,7
00B2  72 02                         JC      M9                      ; MODE IS 4,5, OR 6
00B4  03 D9                         ADD     BX,CX                   ; MOVE TO BW CARD ROW OF INIT_TABLE

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

00B6                        M9:                                     ; OUT_INIT
00B6  50                            PUSH    AX                      ; SAVE MODE IN AH

00B7  06                            PUSH    ES                      ; SAVE SCREEN BUFFER'S SEGMENT
00B8  33 C0                         XOR     AX,AX                   ; ESTABLISH ADDRESSIBILITY TO ABSO
00BA  8E C0                         MOV     ES,AX                   ;
00BC  8B 47 0A                      MOV     AX,WORD PTR [BX+10]     ; GET THE CURSOR MODE FROM THE TABLE
00BF  86 E0                         XCHG    AH,AL                   ; PUT CURSOR MODE IN CORRECT POSTION
                                    ASSUME  ES:ABSO
00C1  26: A3 0460 R                 MOV     ES:WORD PTR DATA_AREA[CURSOR_MODE-DATA],AX
                                    ASSUME  ES:VIDEO_RAM
00C5  07                            POP     ES              ; RESTORE THE SCREEN BUFFER'S SEGMENT

00C6  32 E4                         XOR     AH,AH           ; AH WILL SERVE AS REGISTER NUMBER DURING LOOP

                            ;----- LOOP THROUGH TABLE, OUTPUTTTING REG ADDRESS, THEN VALUE FROM TABLE

00C8                        M10:                                    ; INIT LOOP
00C8  8A C4                         MOV     AL,AH                   ; GET 6845 REGISTER NUMBER
00CA  EE                            OUT     DX,AL
00CB  42                            INC     DX                      ; POINT TO DATA PORT
00CC  FE C4                         INC     AH                      ; NEXT REGISTER VALUE
00CE  8A 07                         MOV     AL,[BX]                 ; GET TABLE VALUE
00D0  EE                            OUT     DX,AL                   ; OUT TO CHIP
00D1  43                            INC     BX                      ; NEXT IN TABLE
00D2  4A                            DEC     DX                      ; BACK TO POINTER REGISTER
00D3  E2 F3                         LOOP    M10                     ; DO THE WHOLE TABLE
00D5  58                            POP     AX                      ; GET MODE BACK
00D6  1F                            POP     DS                      ; RECOVER SEGMENT VALUE
                                    ASSUME  DS:DATA

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

00D7  33 FF                         XOR     DI,DI                   ; SET UP POINTER FOR REGEN
00D9  89 3E 004E R                  MOV     CRT_START,DI            ; START ADDRESS SAVED IN GLOBAL
00DD  C6 06 0062 R 00               MOV     ACTIVE_PAGE,0           ; SET PAGE VALUE
00E2  B9 2000                       MOV     CX,8192                 ; NUMBER OF WORDS IN COLOR CARD
00E5  80 FC 04                      CMP     AH,4                    ; TEST FOR GRAPHICS
00E8  72 0B                         JC      M12                     ; NO_GRAPHICS_INIT
00EA  80 FC 07                      CMP     AH,7                    ; TEST FOR BW CARD
00ED  74 04                         JE      M11                     ; BW_CARD_INIT
00EF  33 C0                         XOR     AX,AX                   ; FILL FOR GRAPHICS MODE
00F1  EB 05                         JMP     SHORT M13               ; CLEAR_BUFFER
00F3                        M11:                                    ; BW_CARD_INIT
00F3  B5 08                         MOV     CH,08H                  ; BUFFER SIZE ON BW CARD (2048)
00F5                        M12:                                    ; NO_GRAPHICS_INIT
00F5  B8 0720                       MOV     AX,' '+7*256            ; FILL CHAR FOR ALPHA
00F8                        M13:                                    ; CLEAR_BUFFER
00F8  F3/ AB                        REP     STOSW                   ; FILL THE REGEN BUFFER WITH BLANKS

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

00FA  A0 0049 R                     MOV     AL,CRT_MODE             ; GET THE MODE
00FD  32 E4                         XOR     AH,AH                   ;  INTO AX REGISTER
00FF  8B F0                         MOV     SI,AX                   ; TABLE POINTER, INDEXED BY MODE
0101  8B 16 0063 R                  MOV     DX,ADDR_6845    ; PREPARE TO OUTPUT TO VIDEO ENABLE PORT
0105  83 C2 04                      ADD     DX,4

0108  2E: 8A 84 0000 E              MOV     AL,CS:[SI + OFFSET BYTE PTR M7]

010D  EE                            OUT     DX,AL                   ; SET VIDEO ENABLE PORT
010E  A2 0065 R                     MOV     CRT_MODE_SET,AL         ; SAVE THAT VALUE

                            ;----- DETERMINE NUMBER OF COLUMNS, BOTH FOR ENTIRE DISPLAY
                            ;----- AND THE NUMBER TO BE USED FOR TTY INTERFACE

0111  2E: 8A 84 0000 E              MOV     AL,CS:[SI + OFFSET BYTE PTR M6]
0116  32 E4                         XOR     AH,AH
0118  A3 004A R                     MOV     CRT_COLS,AX             ; NUMBER OF COLUMNS IN THIS SCREEN

                            ;----- SET CURSOR POSITIONS

011B  81 E6 000E                    AND     SI,0EH                  ; WORD OFFSET INTO CLEAR LENGTH TABLE
```

```
011F  2E: 8B 8C 0000 E          MOV     CX,CS:[SI + OFFSET M5]  ; LENGTH TO CLEAR

0124  89 0E 004C R              MOV     CRT_LEN,CX              ; SAVE LENGTH OF CRT -- NOT USED FOR BW
0128  B9 0008                   MOV     CX,8                    ; CLEAR ALL CURSOR POSITIONS
012B  BF 0050 R                 MOV     DI,OFFSET CURSOR_POSN
012E  1E                        PUSH    DS                      ; ESTABLISH SEGMENT
012F  07                        POP     ES                      ;   ADDRESSING
0130  33 C0                     XOR     AX,AX
0132  F3/ AB                    REP     STOSW                   ; FILL WITH ZEROES

                        ;----- SET UP OVERSCAN REGISTER

0134  42                        INC     DX                      ; SET OVERSCAN PORT TO A DEFAULT
0135  B0 30                     MOV     AL,30H                  ; VALUE OF 30H FOR ALL MODES EXCEPT 640X200
0137  80 3E 0049 R 06           CMP     CRT_MODE,6              ; SEE IF THE MODE IS 640X200 BW
013C  75 02                     JNZ     M14                     ; IF IT ISNT 640X200, THEN GOTO REGULAR
013E  B0 3F                     MOV     AL,3FH                  ; IF IT IS 640X200, THEN PUT IN 3FH
0140  EE                M14:    OUT     DX,AL                   ; OUTPUT THE CORRECT VALUE TO 3D9 PORT
0141  A2 0066 R                 MOV     CRT_PALLETTE,AL         ; SAVE THE VALUE FOR FUTURE USE

                        ;----- NORMAL RETURN FROM ALL VIDEO RETURNS

0144                    VIDEO_RETURN:
0144  5D                        POP     BP
0145  5F                        POP     DI
0146  5E                        POP     SI
0147  5B                        POP     BX
0148                    M15:                                    ; VIDEO_RETURN_C
0148  59                        POP     CX
0149  5A                        POP     DX
014A  1F                        POP     DS
014B  07                        POP     ES                      ; RECOVER SEGMENTS
014C  CF                        IRET                            ; ALL DONE
014D                    SET_MODE        ENDP
                        ;--------------------------------------------------
                        ; SET_CTYPE
                        ;       THIS ROUTINE SETS THE CURSOR VALUE
                        ; INPUT
                        ;       (CX) HAS CURSOR VALUE CH-START LINE, CL-STOP LINE
                        ; OUTPUT
                        ;       NONE
                        ;--------------------------------------------------
014D                    SET_CTYPE       PROC    NEAR
014D  B4 0A                     MOV     AH,10           ; 6845 REGISTER FOR CURSOR SET
014F  89 0E 0060 R              MOV     CURSOR_MODE,CX  ; SAVE IN DATA AREA
0153  E8 0158 R                 CALL    M16             ; OUTPUT CX REG
0156  EB EC                     JMP     VIDEO_RETURN

                        ;------ THIS ROUTINE OUTPUTS THE CX REGISTER TO THE 6845 REGS NAMED IN AH

0158                    M16:
0158  8B 16 0063 R              MOV     DX,ADDR_6845    ; ADDRESS REGISTER
015C  8A C4                     MOV     AL,AH           ; GET VALUE
015E  EE                        OUT     DX,AL           ; REGISTER SET
015F  42                        INC     DX              ; DATA REGISTER
0160  EB 00                     JMP     SHORT $+2       ; IO DELAY
0162  8A C5                     MOV     AL,CH           ; DATA
0164  EE                        OUT     DX,AL
0165  4A                        DEC     DX
0166  EB 00                     JMP     SHORT $+2       ; IO DELAY
0168  8A C4                     MOV     AL,AH
016A  FE C0                     INC     AL              ; POINT TO OTHER DATA REGISTER
016C  EE                        OUT     DX,AL           ; SET FOR SECOND REGISTER
016D  42                        INC     DX
016E  EB 00                     JMP     SHORT $+2       ; IO DELAY
0170  8A C1                     MOV     AL,CL           ; SECOND DATA VALUE
0172  EE                        OUT     DX,AL
0173  C3                        RET                     ; ALL DONE
0174                    SET_CTYPE       ENDP
                        ;----------------------------------------
                        ; SET_CPOS
                        ;       THIS ROUTINE SETS THE CURRENT CURSOR POSITION TO THE
                        ;       NEW X-Y VALUES PASSED
                        ; INPUT
                        ;       DX - ROW,COLUMN OF NEW CURSOR
                        ;       BH - DISPLAY PAGE OF CURSOR
                        ; OUTPUT
                        ;       CURSOR IS SET AT 6845 IF DISPLAY PAGE IS CURRENT DISPLAY
                        ;----------------------------------------
0174                    SET_CPOS        PROC    NEAR
0174  8A CF                     MOV     CL,BH
0176  32 ED                     XOR     CH,CH           ; ESTABLISH LOOP COUNT
0178  D1 E1                     SAL     CX,1            ; WORD OFFSET
017A  8B F1                     MOV     SI,CX           ; USE INDEX REGISTER
017C  89 94 0050 R              MOV     [SI+OFFSET CURSOR_POSN],DX      ; SAVE THE POINTER
0180  38 3E 0062 R              CMP     ACTIVE_PAGE,BH
0184  75 05                     JNZ     M17             ; SET_CPOS_RETURN
0186  8B C2                     MOV     AX,DX           ; GET ROW/COLUMN TO AX
0188  E8 018D R                 CALL    M18             ; CURSOR_SET
018B                    M17:                            ; SET_CPOS_RETURN
018B  EB B7                     JMP     VIDEO_RETURN
018D                    SET_CPOS        ENDP

                        ;------ SET CURSOR POSITION, AX HAS ROW/COLUMN FOR CURSOR

018D                    M18     PROC    NEAR
018D  E8 0211 R                 CALL    POSITION        ; DETERMINE LOCATION IN REGEN BUFFER
0190  8B C8                     MOV     CX,AX
0192  03 0E 004E R              ADD     CX,CRT_START    ; ADD IN THE START ADDRESS FOR THIS PAGE
0196  D1 F9                     SAR     CX,1            ; DIVIDE BY 2 FOR CHAR ONLY COUNT
0198  B4 0E                     MOV     AH,14           ; REGISTER NUMBER FOR CURSOR
019A  E8 0158 R                 CALL    M16             ; OUTPUT THE VALUE TO THE 6845
019D  C3                        RET
019E                    M18     ENDP
                        ;--------------------------------------------------
                        ; READ_CURSOR
                        ;       THIS ROUTINE READS THE CURRENT CURSOR VALUE FROM THE
                        ;       6845, FORMATS IT, AND SENDS IT BACK TO THE CALLER
                        ; INPUT
                        ;       BH - PAGE OF CURSOR
                        ; OUTPUT
                        ;       DX - ROW, COLUMN OF THE CURRENT CURSOR POSITION
                        ;       CX - CURRENT CURSOR MODE
                        ;----------------------------------------
019E                    READ_CURSOR     PROC    NEAR
019E  8A DF                     MOV     BL,BH
01A0  32 FF                     XOR     BH,BH
01A2  D1 E3                     SAL     BX,1            ; WORD OFFSET
01A4  8B 97 0050 R              MOV     DX,[BX+OFFSET CURSOR_POSN]
01A8  8B 0E 0060 R              MOV     CX,CURSOR_MODE
01AC  5D                        POP     BP
01AD  5F                        POP     DI
01AE  5E                        POP     SI
01AF  5B                        POP     BX
01B0  58                        POP     AX      ; DISCARD SAVED CX AND DX
```

```
01B1  58                     POP     AX
01B2  1F                     POP     DS
01B3  07                     POP     ES
01B4  CF                     IRET
01B5                 READ_CURSOR     ENDP
                     ;----------------------------------------------
                     ; ACT_DISP_PAGE
                     ;       THIS ROUTINE SETS THE ACTIVE DISPLAY PAGE, ALLOWING
                     ;       THE FULL USE OF THE RAM SET ASIDE FOR THE VIDEO ATTACHMENT
                     ; INPUT
                     ;       AL HAS THE NEW ACTIVE DISPLAY PAGE
                     ; OUTPUT
                     ;       THE 6845 IS RESET TO DISPLAY THAT PAGE
                     ;----------------------------------------------
01B5                 ACT_DISP_PAGE   PROC    NEAR
01B5  A2 0062 R              MOV     ACTIVE_PAGE,AL   ; SAVE ACTIVE PAGE VALUE
01B8  8B 0E 004C R           MOV     CX,CRT_LEN       ; GET SAVED LENGTH OF REGEN BUFFER
01BC  98                     CBW                      ; CONVERT AL TO WORD
01BD  50                     PUSH    AX               ; SAVE PAGE VALUE
01BE  F7 E1                  MUL     CX               ; DISPLAY PAGE TIMES REGEN LENGTH
01C0  A3 004E R              MOV     CRT_START,AX     ; SAVE START ADDRESS FOR LATER REQUIREMENTS
01C3  8B C8                  MOV     CX,AX            ; START ADDRESS TO CX
01C5  D1 F9                  SAR     CX,1             ; DIVIDE BY 2 FOR 6845 HANDLING
01C7  B4 0C                  MOV     AH,12            ; 6845 REGISTER FOR START ADDRESS
01C9  E8 0158 R              CALL    M16
01CC  5B                     POP     BX               ; RECOVER PAGE VALUE
01CD  D1 E3                  SAL     BX,1             ; *2 FOR WORD OFFSET
01CF  8B 87 0050 R           MOV     AX,[BX + OFFSET CURSOR_POSN]     ; GET CURSOR FOR THIS PAGE
01D3  E8 018D R              CALL    M18              ; SET THE CURSOR POSITION
01D6  E9 0144 R              JMP     VIDEO_RETURN
01D9                 ACT_DISP_PAGE   ENDP
                     ;----------------------------------------------
                     ; SET COLOR
                     ;       THIS ROUTINE WILL ESTABLISH THE BACKGROUND COLOR, THE OVERSCAN COLOR,
                     ;       AND THE FOREGROUND COLOR SET FOR MEDIUM RESOLUTION GRAPHICS
                     ; INPUT
                     ;       (BH) HAS COLOR ID
                     ;               IF BH=0, THE BACKGROUND COLOR VALUE IS SET
                     ;                       FROM THE LOW BITS OF BL (0-31)
                     ;               IF BH=1, THE PALLETTE SELECTION IS MADE
                     ;                       BASED ON THE LOW BIT OF BL:
                     ;                               0 = GREEN, RED, YELLOW FOR COLORS 1,2,3
                     ;                               1 = BLUE, CYAN, MAGENTA FOR COLORS 1,2,3
                     ;       (BL) HAS THE COLOR VALUE TO BE USED
                     ; OUTPUT
                     ;       THE COLOR SELECTION IS UPDATED
                     ;----------------------------------------------
01D9                 SET_COLOR       PROC    NEAR
01D9  8B 16 0063 R           MOV     DX,ADDR_6845     ; I/O PORT FOR PALETTE
01DD  83 C2 05               ADD     DX,5             ; OVERSCAN PORT
01E0  A0 0066 R              MOV     AL,CRT_PALLETTE  ; GET THE CURRENT PALLETTE VALUE
01E3  0A FF                  OR      BH,BH            ; IS THIS COLOR 0?
01E5  75 0E                  JNZ     M20              ; OUTPUT COLOR 1

                     ;------ HANDLE COLOR 0 BY SETTING THE BACKGROUND COLOR

01E7  24 E0                  AND     AL,0E0H          ; TURN OFF LOW 5 BITS OF CURRENT
01E9  80 E3 1F               AND     BL,01FH          ; TURN OFF HIGH 3 BITS OF INPUT VALUE
01EC  0A C3                  OR      AL,BL            ; PUT VALUE INTO REGISTER
01EE                 M19:                             ; OUTPUT THE PALLETTE
01EE  EE                     OUT     DX,AL            ; OUTPUT COLOR SELECTION TO 3D9 PORT
01EF  A2 0066 R              MOV     CRT_PALLETTE,AL  ; SAVE THE COLOR VALUE
01F2  E9 0144 R              JMP     VIDEO_RETURN

                     ;------ HANDLE COLOR 1 BY SELECTING THE PALLETTE TO BE USED

01F5                 M20:
01F5  24 DF                  AND     AL,0DFH          ; TURN OFF PALLETTE SELECT BIT
01F7  D0 EB                  SHR     BL,1             ; TEST THE LOW ORDER BIT OF BL
01F9  73 F3                  JNC     M19              ; ALREADY DONE
01FB  0C 20                  OR      AL,20H           ; TURN ON PALLETTE SELECT BIT
01FD  EB EF                  JMP     M19              ; GO DO IT
01FF                 SET_COLOR       ENDP
                     ;----------------------------------------------
                     ; VIDEO STATE
                     ;  RETURNS THE CURRENT VIDEO STATE IN AX
                     ;  AH = NUMBER OF COLUMNS ON THE SCREEN
                     ;  AL = CURRENT VIDEO MODE
                     ;  BH = CURRENT ACTIVE PAGE
                     ;----------------------------------------------
01FF                 VIDEO_STATE     PROC    NEAR
01FF  8A 26 004A R           MOV     AH,BYTE PTR CRT_COLS     ; GET NUMBER OF COLUMNS
0203  A0 0049 R              MOV     AL,CRT_MODE              ; CURRENT MODE
0206  8A 3E 0062 R           MOV     BH,ACTIVE_PAGE           ; GET CURRENT ACTIVE PAGE
020A  5D                     POP     BP                       ; RECOVER REGISTERS
020B  5F                     POP     DI                       ;
020C  5E                     POP     SI                       ;
020D  59                     POP     CX                       ; DISCARD SAVED BX
020E  E9 0148 R              JMP     M15                      ; RETURN TO CALLER
0211                 VIDEO_STATE     ENDP
                     ;-----------------------------------
                     ; POSITION
                     ;       THIS SERVICE ROUTINE CALCULATES THE REGEN BUFFER ADDRESS
                     ;       OF A CHARACTER IN THE ALPHA MODE
                     ; INPUT
                     ;       AX = ROW, COLUMN POSITION
                     ; OUTPUT
                     ;       AX = OFFSET OF CHAR POSITION IN REGEN BUFFER
                     ;-----------------------------------
0211                 POSITION        PROC    NEAR
0211  53                     PUSH    BX               ; SAVE REGISTER
0212  8B D8                  MOV     BX,AX
0214  8A C4                  MOV     AL,AH            ; ROWS TO AL
0216  F6 26 004A R           MUL     BYTE PTR CRT_COLS        ; DETERMINE BYTES TO ROW
021A  32 FF                  XOR     BH,BH
021C  03 C3                  ADD     AX,BX            ; ADD IN COLUMN VALUE
021E  D1 E0                  SAL     AX,1             ; * 2 FOR ATTRIBUTE BYTES
0220  5B                     POP     BX
0221  C3                     RET
0222                 POSITION        ENDP
                     ;---------------------------------------
                     ; SCROLL UP
                     ;       THIS ROUTINE MOVES A BLOCK OF CHARACTERS UP
                     ;       ON THE SCREEN
                     ; INPUT
                     ;       (AH) = CURRENT CRT MODE
                     ;       (AL) = NUMBER OF ROWS TO SCROLL
                     ;       (CX) = ROW/COLUMN OF UPPER LEFT CORNER
                     ;       (DX) = ROW/COLUMN OF LOWER RIGHT CORNER
                     ;       (BH) = ATTRIBUTE TO BE USED ON BLANKED LINE
                     ;       (DS) = DATA SEGMENT
                     ;       (ES) = REGEN BUFFER SEGMENT
                     ; OUTPUT
                     ;       NONE -- THE REGEN BUFFER IS MODIFIED
                     ;---------------------------------------
                             ASSUME  CS:CODE,DS:DATA,ES:DATA
```

SECTION 5

```
0222                         SCROLL_UP       PROC    NEAR

0222  E8 0303 R                      CALL    TEST_LINE_COUNT ;

0225  80 FC 04                       CMP     AH,4            ; TEST FOR GRAPHICS MODE
0228  72 08                          JC      N1              ; HANDLE SEPARATELY
022A  80 FC 07                       CMP     AH,7            ; TEST FOR BW CARD
022D  74 03                          JE      N1
022F  E9 04D5 R                      JMP     GRAPHICS_UP
0232                         N1:                             ; UP_CONTINUE
0232  53                             PUSH    BX              ; SAVE FILL ATTRIBUTE IN BH
0233  8B C1                          MOV     AX,CX           ; UPPER LEFT POSITION
0235  E8 026F R                      CALL    SCROLL_POSITION ; DO SETUP FOR SCROLL
0238  74 31                          JZ      N7              ; BLANK_FIELD
023A  03 F0                          ADD     SI,AX           ; FROM ADDRESS
023C  8A E6                          MOV     AH,DH           ; # ROWS IN BLOCK
023E  2A E3                          SUB     AH,BL           ; # ROWS TO BE MOVED
0240                         N2:                             ; ROW_LOOP
0240  E8 02B6 R                      CALL    N10             ; MOVE ONE ROW
0243  03 F5                          ADD     SI,BP
0245  03 FD                          ADD     DI,BP           ; POINT TO NEXT LINE IN BLOCK
0247  FE CC                          DEC     AH              ; COUNT OF LINES TO MOVE
0249  75 F5                          JNZ     N2              ; ROW_LOOP
024B                         N3:                             ; CLEAR_ENTRY
024B  58                             POP     AX              ; RECOVER ATTRIBUTE IN AH
024C  B0 20                          MOV     AL,' '          ; FILL WITH BLANKS
024E                         N4:                             ; CLEAR_LOOP
024E  E8 02BF R                      CALL    N11             ; CLEAR THE ROW
0251  03 FD                          ADD     DI,BP           ; POINT TO NEXT LINE
0253  FE CB                          DEC     BL              ; COUNTER OF LINES TO SCROLL
0255  75 F7                          JNZ     N4              ; CLEAR_LOOP
0257                         N5:                             ; SCROLL_END
0257  E8 0000 E                      CALL    DDS
025A  80 3E 0049 R 07                CMP     CRT_MODE,7      ; IS THIS THE BLACK AND WHITE CARD
025F  74 07                          JE      N6              ; IF SO, SKIP THE MODE RESET
0261  A0 0065 R                      MOV     AL,CRT_MODE_SET ; GET THE VALUE OF THE MODE SET
0264  BA 03D8                        MOV     DX,03D8H        ; ALWAYS SET COLOR CARD PORT
0267  EE                             OUT     DX,AL
0268                         N6:                             ; VIDEO_RET_HERE
0268  E9 0144 R                      JMP     VIDEO_RETURN
026B                         N7:                             ; BLANK_FIELD
026B  8A DE                          MOV     BL,DH           ; GET ROW COUNT
026D  EB DC                          JMP     N3              ; GO CLEAR THAT AREA
026F                         SCROLL_UP       ENDP

                             ;----- HANDLE COMMON SCROLL SET UP HERE

026F                         SCROLL_POSITION PROC    NEAR
026F  80 3E 0049 R 02                CMP     CRT_MODE,2      ; TEST FOR SPECIAL CASE HERE
0274  72 19                          JB      N9              ; HAVE TO HANDLE 80X25 SEPARATELY
0276  80 3E 0049 R 03                CMP     CRT_MODE,3
027B  77 12                          JA      N9

                             ;------ 80X25 COLOR CARD SCROLL

027D  52                             PUSH    DX
027E  BA 03DA                        MOV     DX,3DAH         ; GUARANTEED TO BE COLOR CARD HERE
0281  50                             PUSH    AX
0282                         N8:                             ; WAIT_DISP_ENABLE
0282  EC                             IN      AL,DX           ; GET PORT
0283  A8 08                          TEST    AL,8            ; WAIT FOR VERTICAL RETRACE
0285  74 FB                          JZ      N8              ; WAIT_DISP_ENABLE
0287  B0 25                          MOV     AL,25H
0289  BA 03D8                        MOV     DX,03D8H
028C  EE                             OUT     DX,AL           ; TURN OFF VIDEO
028D  58                             POP     AX              ; DURING VERTICAL RETRACE
028E  5A                             POP     DX
028F  E8 0211 R              N9:     CALL    POSITION        ; CONVERT TO REGEN POINTER
0292  03 06 004E R                   ADD     AX,CRT_START    ; OFFSET OF ACTIVE PAGE
0296  8B F8                          MOV     DI,AX           ; TO ADDRESS FOR SCROLL
0298  8B F0                          MOV     SI,AX           ; FROM ADDRESS FOR SCROLL
029A  2B D1                          SUB     DX,CX           ; DX = #ROWS, #COLS IN BLOCK
029C  FE C6                          INC     DH
029E  FE C2                          INC     DL              ; INCREMENT FOR 0 ORIGIN
02A0  32 ED                          XOR     CH,CH           ; SET HIGH BYTE OF COUNT TO ZERO
02A2  8B 2E 004A R                   MOV     BP,CRT_COLS     ; GET NUMBER OF COLUMNS IN DISPLAY
02A6  03 ED                          ADD     BP,BP           ; TIMES 2 FOR ATTRIBUTE BYTE
02A8  8A C3                          MOV     AL,BL           ; GET LINE COUNT
02AA  F6 26 004A R                   MUL     BYTE PTR CRT_COLS       ; DETERMINE OFFSET TO FROM ADDRESS
02AE  03 C0                          ADD     AX,AX           ; *2 FOR ATTRIBUTE BYTE
02B0  06                             PUSH    ES              ; ESTABLISH ADDRESSING TO REGEN BUFFER
02B1  1F                             POP     DS              ;  FOR BOTH POINTERS
02B2  80 FB 00                       CMP     BL,0            ; 0 SCROLL MEANS BLANK FIELD
02B5  C3                             RET                     ; RETURN WITH FLAGS SET
02B6                         SCROLL_POSITION ENDP

                             ;------ MOVE_ROW
02B6                         N10     PROC    NEAR
02B6  8A CA                          MOV     CL,DL           ; GET # OF COLS TO MOVE
02B8  56                             PUSH    SI
02B9  57                             PUSH    DI              ; SAVE START ADDRESS
02BA  F3/ A5                         REP     MOVSW           ; MOVE THAT LINE ON SCREEN
02BC  5F                             POP     DI
02BD  5E                             POP     SI              ; RECOVER ADDRESSES
02BE  C3                             RET
02BF                         N10     ENDP

                             ;------ CLEAR_ROW
02BF                         N11     PROC    NEAR
02BF  8A CA                          MOV     CL,DL           ; GET # COLUMNS TO CLEAR
02C1  57                             PUSH    DI
02C2  F3/ AB                         REP     STOSW           ; STORE THE FILL CHARACTER
02C4  5F                             POP     DI
02C5  C3                             RET
02C6                         N11     ENDP
                             ;----------------------------------
                             ; SCROLL_DOWN
                             ;       THIS ROUTINE MOVES THE CHARACTERS WITHIN A DEFINED
                             ;       BLOCK DOWN ON THE SCREEN, FILLING THE TOP LINES
                             ;       WITH A DEFINED CHARACTER
                             ; INPUT
                             ;       (AH) = CURRENT CRT MODE
                             ;       (AL) = NUMBER OF LINES TO SCROLL
                             ;       (CX) = UPPER LEFT CORNER OF REGION
                             ;       (DX) = LOWER RIGHT CORNER OF REGION
                             ;       (BH) = FILL CHARACTER
                             ;       (DS) = DATA SEGMENT
                             ;       (ES) = REGEN SEGMENT
                             ; OUPUT
                             ;       NONE -- SCREEN IS SCROLLED
                             ;-----------------------------------
02C6                         SCROLL_DOWN     PROC    NEAR
02C6  FD                             STD                     ; DIRECTION FOR SCROLL DOWN
02C7  E8 0303 R                      CALL    TEST_LINE_COUNT ;
02CA  80 FC 04                       CMP     AH,4            ; TEST FOR GRAPHICS
02CD  72 08                          JC      N12
```

```
02CF  80 FC 07              CMP     AH,7            ; TEST FOR BW CARD
02D2  74 03                 JE      N12
02D4  E9 052E R             JMP     GRAPHICS_DOWN
02D7                N12:                            ; CONTINUE_DOWN
02D7  53                    PUSH    BX              ; SAVE ATTRIBUTE IN BH
02D8  8B C2                 MOV     AX,DX           ; LOWER RIGHT CORNER
02DA  E8 026F R             CALL    SCROLL_POSITION ; GET REGEN LOCATION
02DD  74 20                 JZ      N16
02DF  2B F0                 SUB     SI,AX           ; SI IS FROM ADDRESS
02E1  8A E6                 MOV     AH,DH           ; GET TOTAL # ROWS
02E3  2A E3                 SUB     AH,BL           ; COUNT TO MOVE IN SCROLL
02E5                N13:
02E5  E8 02B6 R             CALL    N10             ; MOVE ONE ROW
02E8  2B F5                 SUB     SI,BP
02EA  2B FD                 SUB     DI,BP
02EC  FE CC                 DEC     AH
02EE  75 F5                 JNZ     N13
02F0                N14:
02F0  58                    POP     AX              ; RECOVER ATTRIBUTE IN AH
02F1  B0 20                 MOV     AL,' '
02F3                N15:
02F3  E8 02BF R             CALL    N11             ; CLEAR ONE ROW
02F6  2B FD                 SUB     DI,BP           ; GO TO NEXT ROW
02F8  FE CB                 DEC     BL
02FA  75 F7                 JNZ     N15
02FC  E9 0257 R             JMP     N5                      ; SCROLL_END
02FF                N16:
02FF  8A DE                 MOV     BL,DH
0301  EB ED                 JMP     N14
0303                SCROLL_DOWN     ENDP

                    ;
                    ;---------  TEST IF AMOUNT OF LINES TO BE SCROLLED = AMOUNT OF LINES IN WINDOW
                    ;---------  IF TRUE THEN WE ADJUST AL, IF FALSE WE RETURN...

0303                TEST_LINE_COUNT PROC    NEAR

0303  8A D8                 MOV     BL,AL           ; SAVE LINE COUNT IN BL
0305  0A C0                 OR      AL,AL           ; TEST IF AL IS ALREADY ZERO
0307  74 0E                 JZ      BL_SET          ; IF IT IS THEN RETURN...
0309  50                    PUSH    AX              ; SAVE AX
030A  8A C6                 MOV     AL,DH           ; SUBTRACT LOWER ROW FROM UPPER ROW
030C  2A C5                 SUB     AL,CH           ;
030E  FE C0                 INC     AL              ; ADJUST DIFERENCE BY 1
0310  3A C3                 CMP     AL,BL           ; TEST IF LINE COUNT = AMOUNT OF ROWS IN WINDOW
0312  58                    POP     AX              ; RESTORE AX
0313  75 02                 JNE     BL_SET          ; IF NOT THEN WE'RE ALL SET
0315  2A DB                 SUB     BL,BL           ; OTHERWISE SET BL TO ZERO
0317                BL_SET:
0317  C3                    RET                     ; RETURN
0318                TEST_LINE_COUNT ENDP
                    ;------------------------------------------
                    ; READ_AC_CURRENT
                    ;       THIS ROUTINE READS THE ATTRIBUTE AND CHARACTER AT THE CURRENT
                    ;       CURSOR POSITION AND RETURNS THEM TO THE CALLER
                    ; INPUT
                    ;       (AH) = CURRENT CRT MODE
                    ;       (BH) = DISPLAY PAGE ( ALPHA MODES ONLY )
                    ;       (DS) = DATA SEGMENT
                    ;       (ES) = REGEN SEGMENT
                    ; OUTPUT
                    ;       (AL) = CHAR READ
                    ;       (AH) = ATTRIBUTE READ
                    ;------------------------------------------
                            ASSUME  CS:CODE,DS:DATA,ES:DATA
0318                READ_AC_CURRENT PROC    NEAR
0318  80 FC 04              CMP     AH,4                    ; IS THIS GRAPHICS
031B  72 08                 JC      P1
031D  80 FC 07              CMP     AH,7                    ; IS THIS BW CARD
0320  74 03                 JE      P1
0322  E9 0669 R             JMP     GRAPHICS_READ
0325                P1:                                     ; READ_AC_CONTINUE
0325  E8 0342 R             CALL    FIND_POSITION
0328  8B F3                 MOV     SI,BX                   ; ESTABLISH ADDRESSING IN SI

                    ;------ WAIT FOR HORIZONTAL RETRACE

032A  8B 16 0063 R          MOV     DX,ADDR_6845            ; GET BASE ADDRESS
032E  83 C2 06              ADD     DX,6                    ; POINT AT STATUS PORT
0331  06                    PUSH    ES                      ;
0332  1F                    POP     DS                      ; GET SEGMENT FOR QUICK ACCESS
0333                P2:                                     ; WAIT FOR RETRACE LOW
0333  EC                    IN      AL,DX                   ; GET STATUS
0334  A8 01                 TEST    AL,1                    ; IS HORZ RETRACE LOW
0336  75 FB                 JNZ     P2                      ; WAIT UNTIL IT IS
0338  FA                    CLI                             ; NO MORE INTERRUPTS
0339                P3:                                     ; WAIT FOR RETRACE HIGH
0339  EC                    IN      AL,DX                   ; GET STATUS
033A  A8 01                 TEST    AL,1                    ; IS IT HIGH
033C  74 FB                 JZ      P3                      ; WAIT UNTIL IT IS
033E  AD                    LODSW                           ; GET THE CHAR/ATTR
033F  E9 0144 R             JMP     VIDEO_RETURN
0342                READ_AC_CURRENT ENDP

0342                FIND_POSITION   PROC    NEAR
0342  8A CF                 MOV     CL,BH                   ; DISPLAY PAGE TO CX
0344  32 ED                 XOR     CH,CH
0346  8B F1                 MOV     SI,CX                   ; MOVE TO SI FOR INDEX
0348  D1 E6                 SAL     SI,1                    ; * 2 FOR WORD OFFSET
034A  8B 84 0050 R          MOV     AX,[SI+ OFFSET CURSOR_POSN]     ; GET ROW/COLUMN OF THAT PAGE
034E  33 DB                 XOR     BX,BX                   ; SET START ADDRESS TO ZERO
0350  E3 06                 JCXZ    P5                      ; NO_PAGE
0352                P4:                                     ; PAGE_LOOP
0352  03 1E 004C R          ADD     BX,CRT_LEN              ; LENGTH OF BUFFER
0356  E2 FA                 LOOP    P4
0358                P5:                                     ; NO_PAGE
0358  E8 0211 R             CALL    POSITION                ; DETERMINE LOCATION IN REGEN
035B  03 D8                 ADD     BX,AX                   ; ADD TO START OF REGEN
035D  C3                    RET
035E                FIND_POSITION   ENDP

                    ;------------------------------------------
                    ; WRITE_AC_CURRENT
                    ;       THIS ROUTINE WRITES THE ATTRIBUTE AND CHARACTER AT
                    ;       THE CURRENT CURSOR POSITION
                    ; INPUT
                    ;       (AH) = CURRENT CRT MODE
                    ;       (BH) = DISPLAY PAGE
                    ;       (CX) = COUNT OF CHARACTERS TO WRITE
                    ;       (AL) = CHAR TO WRITE
                    ;       (BL) = ATTRIBUTE OF CHAR TO WRITE
                    ;       (DS) = DATA SEGMENT
                    ;       (ES) = REGEN SEGMENT
                    ; OUTPUT
                    ;       NONE
                    ;------------------------------------------
```

SECTION 5

```
035E                            WRITE_AC_CURRENT        PROC    NEAR
035E  80 FC 04                          CMP     AH,4                    ; IS THIS GRAPHICS
0361  72 08                             JC      P6
0363  80 FC 07                          CMP     AH,7                    ; IS THIS BW CARD
0366  74 03                             JE      P6
0368  E9 05B8 R                         JMP     GRAPHICS_WRITE
036B                            P6:                                     ; WRITE_AC_CONTINUE
036B  8A E3                             MOV     AH,BL                   ; GET ATTRIBUTE TO AH
036D  50                                PUSH    AX                      ; SAVE ON STACK
036E  51                                PUSH    CX                      ; SAVE WRITE COUNT
036F  E8 0342 R                         CALL    FIND_POSITION
0372  8B FB                             MOV     DI,BX                   ; ADDRESS TO DI REGISTER
0374  59                                POP     CX                      ; WRITE COUNT
0375  5B                                POP     BX                      ; CHARACTER IN BX REG
0376                            P7:                                     ; WRITE_LOOP

                                ;------ WAIT FOR HORIZONTAL RETRACE

0376  8B 16 0063 R                      MOV     DX,ADDR_6845            ; GET BASE ADDRESS
037A  83 C2 06                          ADD     DX,6                    ; POINT AT STATUS PORT
037D                            P8:
037D  EC                                IN      AL,DX                   ; GET STATUS
037E  A8 01                             TEST    AL,1                    ; IS IT LOW
0380  75 FB                             JNZ     P8                      ; WAIT UNTIL IT IS
0382  FA                                CLI                             ; NO MORE INTERRUPTS
0383                            P9:
0383  EC                                IN      AL,DX                   ; GET STATUS
0384  A8 01                             TEST    AL,1                    ; IS IT HIGH
0386  74 FB                             JZ      P9                      ; WAIT UNTIL IT IS
0388  8B C3                             MOV     AX,BX                   ; RECOVER THE CHAR/ATTR
038A  AB                                STOSW                           ; PUT THE CHAR/ATTR
038B  FB                                STI                             ; INTERRUPTS BACK ON
038C  E2 E8                             LOOP    P7                      ;  AS MANY TIMES AS REQUESTED
038E  E9 0144 R                         JMP     VIDEO_RETURN
0391                            WRITE_AC_CURRENT        ENDP
                                ;------------------------------------------
                                ; WRITE_C_CURRENT
                                ;       THIS ROUTINE WRITES THE CHARACTER AT
                                ;       THE CURRENT CURSOR POSITION, ATTRIBUTE UNCHANGED
                                ; INPUT
                                ;       (AH) = CURRENT CRT MODE
                                ;       (BH) = DISPLAY PAGE
                                ;       (CX) = COUNT OF CHARACTERS TO WRITE
                                ;       (AL) = CHAR TO WRITE
                                ;       (DS) = DATA SEGMENT
                                ;       (ES) = REGEN SEGMENT
                                ; OUTPUT
                                ;       NONE
                                ;---------------------------------------------
0391                            WRITE_C_CURRENT PROC    NEAR
0391  80 FC 04                          CMP     AH,4                    ; IS THIS GRAPHICS
0394  72 08                             JC      P10
0396  80 FC 07                          CMP     AH,7                    ; IS THIS BW CARD
0399  74 03                             JE      P10
039B  E9 05B8 R                         JMP     GRAPHICS_WRITE
039E                            P10:
039E  50                                PUSH    AX                      ; SAVE ON STACK
039F  51                                PUSH    CX                      ; SAVE WRITE COUNT
03A0  E8 0342 R                         CALL    FIND_POSITION
03A3  8B FB                             MOV     DI,BX                   ; ADDRESS TO DI
03A5  59                                POP     CX                      ; WRITE COUNT
03A6  5B                                POP     BX                      ; BL HAS CHAR TO WRITE
03A7                            P11:                                    ; WRITE_LOOP

                                ;------ WAIT FOR HORIZONTAL RETRACE

03A7  8B 16 0063 R                      MOV     DX,ADDR_6845            ; GET BASE ADDRESS
03AB  83 C2 06                          ADD     DX,6                    ; POINT AT STATUS PORT
03AE                            P12:
03AE  EC                                IN      AL,DX                   ; GET STATUS
03AF  A8 01                             TEST    AL,1                    ; IS IT LOW
03B1  75 FB                             JNZ     P12                     ; WAIT UNTIL IT IS
03B3  FA                                CLI                             ; NO MORE INTERRUPTS
03B4                            P13:
03B4  EC                                IN      AL,DX                   ; GET STATUS
03B5  A8 01                             TEST    AL,1                    ; IS IT HIGH
03B7  74 FB                             JZ      P13                     ; WAIT UNTIL IT IS
03B9  8A C3                             MOV     AL,BL                   ; RECOVER CHAR
03BB  FB                                STI                             ; ENABLE INTS.
03BC  AA                                STOSB                           ; PUT THE CHAR/ATTR
03BD  47                                INC     DI                      ; BUMP POINTER PAST ATTRIBUTE
03BE  E2 E7                             LOOP    P11                     ;  AS MANY TIMES AS REQUESTED
03C0  E9 0144 R                         JMP     VIDEO_RETURN
03C3                            WRITE_C_CURRENT ENDP
                                page
                                ;----------------------------------------------------------------
                                ; WRITE_STRING
                                ;       This routine writes a string of characters to the crt.
                                ;
                                ;
                                ; INPUT
                                ;       (AL) = WRITE STRING COMMAND  0 - 3
                                ;       (BH) = DISPLAY PAGE
                                ;       (CX) = COUNT OF CHARACTERS TO WRITE, IF CX == 0 THEN RETURN
                                ;       (BL) = ATTRIBUTE OF CHAR TO WRITE IF AL == 0 || AL == 1
                                ;       (ES) = STRING SEGMENT
                                ;       (BP) = STRING OFFSET
                                ;
                                ; OUTPUT
                                ;       N/A
                                ;---------------------------------------------

03C3                            WRITE_STRING    PROC    NEAR

03C3  3C 04                             CMP     AL,04                   ; TEST FOR INVALID WRITE STRING OPTION
03C5  72 03                             JB      W0                      ; IF OPTION INVALID THEN RETURN
03C7  E9 045B R                         JMP     DONE
03CA  0B C9                     W0:     OR      CX,CX                   ; TEST FOR ZERO LENGTH STRING
03CC  75 03                             JNZ     W1
03CE  E9 045B R                         JMP     DONE                    ; IF ZERO LENGTH STRING THEN RETURN
03D1  53                        W1:     PUSH    BX                      ; SAVE PAGE AND POSSIBLE ATTRIBUTE
03D2  8A DF                             MOV     BL,BH                   ; GET CURRENT CURSOR POSITION
03D4  32 FF                             XOR     BH,BH                   ;
03D6  D1 E3                             SAL     BX,1                    ;
03D8  8B B7 0050 R                      MOV     SI,[BX+OFFSET CURSOR_POSN]
03DC  5B                                POP     BX                      ; RESTORE BX
03DD  56                                PUSH    SI                      ; SAVE CURRENT CURSOR POSITION

03DE  50                                PUSH    AX                      ; SAVE WRITE STRING OPTION
03DF  B8 0200                           MOV     AX,0200H                ; SET NEW CURSOR POSITION
03E2  CD 10                             INT     10H
03E4  58                                POP     AX                      ; RESTORE WRITE STRING OPTION

03E5                            WRITE_CHAR:
03E5  51                                PUSH    CX
03E6  53                                PUSH    BX
```

```
03E7  50                    PUSH    AX
03E8  06                    PUSH    ES

03E9  86 E0                 XCHG    AH,AL               ; PUT THE WRITE STRING OPTION INTO AH
03EB  26: 8A 46 00          MOV     AL,ES:[BP]          ; GET CHARACTER FROM INPUT STRING
03EF  45                    INC     BP                  ; BUMP POINTER TO CHARACTER

                    ;----- TEST FOR SPECIAL CHARACTER'S

03F0  3C 08                 CMP     AL,8                ; IS IT A BACKSPACE
03F2  74 0C                 JE      DO_TTY              ; BACK_SPACE
03F4  3C 0D                 CMP     AL,0DH              ; IS IT CARRIAGE RETURN
03F6  74 08                 JE      DO_TTY              ; CAR_RET
03F8  3C 0A                 CMP     AL,0AH              ; IS IT A LINE FEED
03FA  74 04                 JE      DO_TTY              ; LINE_FEED
03FC  3C 07                 CMP     AL,07H              ; IS IT A BELL
03FE  75 13                 JNE     GET_ATTRIBUTE       ; IF NOT THEN DO WRITE CHARACTER
0400                DO_TTY:
0400  B4 0E                 MOV     AH,14               ; WRITE TTY CHARACTER TO THE CRT
0402  CD 10                 INT     10H                 ;
0404  8A DF                 MOV     BL,BH               ; GET CURRENT CURSOR POSITION
0406  D0 E7                 SAL     BH,1                ; INTO THE DX REGISTER
0408  8B 97 0050 R          MOV     DX,[BX+OFFSET CURSOR_POSN]
040C  07                    POP     ES
040D  58                    POP     AX                  ; RESTORE REGISTERS
040E  5B                    POP     BX
040F  59                    POP     CX
0410  EB 32 90              JMP     ROWS_SET

0413                GET_ATTRIBUTE:
0413  B9 0001               MOV     CX,1                ; SET CHARACTER WRITE AMOUNT TO ONE
0416  80 FC 02              CMP     AH,2                ; IS THE ATTRIBUTE IN THE STRING
0419  72 05                 JB      GOT_IT              ; IF NOT THEN JUMP
041B  26: 8A 5E 00          MOV     BL,ES:[BP]          ; ELSE GET IT
041F  45                    INC     BP                  ; BUMP STRING POINTER

0420                GOT_IT:
0420  B4 09                 MOV     AH,09               ; WRITE CHARACTER TO THE CRT
0422  CD 10                 INT     10H
0424  07                    POP     ES
0425  58                    POP     AX                  ; RESTORE REGISTERS
0426  5B                    POP     BX
0427  59                    POP     CX

0428  FE C2                 INC     DL                  ; INCREMENT COLUMN COUNTER
042A  3A 16 004A R          CMP     DL,BYTE PTR CRT_COLS ; IF COLS ARE WITHIN RANGE FOR
                                                        ;   THIS MODE THEN
042E  72 14                 JB      COLUMNS_SET         ;   GOTO COLS SET
0430  FE C6                 INC     DH                  ; BUMP ROW COUNTER BY ONE
0432  2A D2                 SUB     DL,DL               ; SET COLUMN COUNTER TO ZERO
0434  80 FE 19              CMP     DH,25               ; IF ROWS ARE < 25 THEN
0437  72 0B                 JB      ROWS_SET            ;   GOTO ROWS_SET
                                                        ; SAVE WRITE STRING PARAMETER REGS
0439  06                    PUSH    ES                  ; SAVE REG'S THAT GET CLOBBERED
043A  50                    PUSH    AX
043B  B8 0E0A               MOV     AX,0E0AH            ; DO SCROLL ONE LINE
043E  CD 10                 INT     10H                 ; RESET ROW COUNTER TO 24
0440  FE CE                 DEC     DH                  ;
0442  58                    POP     AX                  ; RESTORE REG'S
0443  07                    POP     ES
0444                ROWS_SET:
0444                COLUMNS_SET:
0444  50                    PUSH    AX                  ; SAVE WRITE STRING OPTION
0445  B8 0200               MOV     AX,0200H            ; SET NEW CURSOR POSITION
0448  CD 10                 INT     10H                 ;
044A  58                    POP     AX
044B  E2 98                 LOOP    WRITE_CHAR          ; DO IT ONCE MORE UNTIL CX = ZERO

044D  5A                    POP     DX                  ; RESTORE OLD CURSOR COORDINATES
044E  3C 01                 CMP     AL,1                ; IF CURSOR WAS TO BE MOVED THEN
0450  74 09                 JE      DONE                ;   WE'RE DONE
0452  3C 03                 CMP     AL,3                ;
0454  74 05                 JE      DONE                ;
0456  B8 0200               MOV     AX,0200H            ; ELSE RESTORE OLD CURSOR POSITION
0459  CD 10                 INT     10H
045B                DONE:
045B  E9 0144 R             JMP     VIDEO_RETURN        ; RETURN TO CALLER

045E                WRITE_STRING    ENDP
                    page
                    ;------------------------------------------
                    ; READ DOT -- WRITE DOT
                    ; THESE ROUTINES WILL WRITE A DOT, OR READ THE
                    ;  DOT AT THE INDICATED LOCATION
                    ; ENTRY --
                    ;   DX = ROW (0-199)    (THE ACTUAL VALUE DEPENDS ON THE MODE)
                    ;   CX = COLUMN ( 0-639) ( THE VALUES ARE NOT RANGE CHECKED )
                    ;   AL = DOT VALUE TO WRITE (1,2 OR 4 BITS DEPENDING ON MODE,
                    ;        REQ'D 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  CS:CODE,DS:DATA,ES:DATA
045E                READ_DOT        PROC    NEAR
045E  E8 0492 R             CALL    R3              ; DETERMINE BYTE POSITION OF DOT
0461  26: 8A 04             MOV     AL,ES:[SI]      ;       ; GET THE BYTE
0464  22 C4                 AND     AL,AH           ; MASK OFF THE OTHER BITS IN THE BYTE
0466  D2 E0                 SHL     AL,CL           ; LEFT JUSTIFY THE VALUE
0468  8A CE                 MOV     CL,DH           ; GET NUMBER OF BITS IN RESULT
046A  D2 C0                 ROL     AL,CL           ; RIGHT JUSTIFY THE RESULT
046C  E9 0144 R             JMP     VIDEO_RETURN    ; RETURN FROM VIDEO IO
046F                READ_DOT        ENDP

046F                WRITE_DOT       PROC    NEAR
046F  50                    PUSH    AX              ; SAVE DOT VALUE
0470  50                    PUSH    AX              ;  TWICE
0471  E8 0492 R             CALL    R3              ; DETERMINE BYTE POSITION OF THE DOT
0474  D2 E8                 SHR     AL,CL           ; SHIFT TO SET UP THE BITS FOR OUTPUT
0476  22 C4                 AND     AL,AH           ; STRIP OFF THE OTHER BITS
0478  26: 8A 0C             MOV     CL,ES:[SI]      ; GET THE CURRENT BYTE
047B  5B                    POP     BX              ; RECOVER XOR FLAG
047C  F6 C3 80              TEST    BL,80H          ; IS IT ON
047F  75 0D                 JNZ     R2              ; YES, XOR THE DOT
0481  F6 D4                 NOT     AH              ; SET THE MASK TO REMOVE THE INDICATED BITS
0483  22 CC                 AND     CL,AH
0485  0A C1                 OR      AL,CL           ; OR IN THE NEW VALUE OF THOSE BITS
0487                R1:                             ; FINISH_DOT
0487  26: 88 04             MOV     ES:[SI],AL      ; RESTORE THE BYTE IN MEMORY
048A  58                    POP     AX
048B  E9 0144 R             JMP     VIDEO_RETURN    ; RETURN FROM VIDEO IO
048E                R2:                             ; XOR_DOT
048E  32 C1                 XOR     AL,CL           ; EXCLUSIVE OR THE DOTS
```

SECTION 5

```
0490  EB F5                        JMP     R1              ; FINISH UP THE WRITING
0492                     WRITE_DOT         ENDP
                         ;---------------------------------------------
                         ; THIS SUBROUTINE DETERMINES THE REGEN BYTE LOCATION OF THE
                         ; INDICATED ROW COLUMN VALUE IN GRAPHICS MODE.
                         ; ENTRY --
                         ;  DX = ROW VALUE (0-199)
                         ;  CX = COLUMN VALUE (0-639)
                         ; EXIT --
                         ;  SI = OFFSET INTO REGEN BUFFER FOR BYTE OF INTEREST
                         ;  AH = MASK TO STRIP OFF THE BITS OF INTEREST
                         ;  CL = BITS TO SHIFT TO RIGHT JUSTIFY THE MASK IN AH
                         ;  DH = # BITS IN RESULT
                         ;---------------------------------------------
0492                     R3      PROC    NEAR
0492  53                         PUSH    BX              ; SAVE BX DURING OPERATION
0493  50                         PUSH    AX              ; WILL SAVE AL DURING OPERATION

                         ;------ DETERMINE 1ST BYTE IN IDICATED ROW BY MULTIPLYING ROW VALUE BY 40
                         ;------ ( LOW BIT OF ROW DETERMINES EVEN/ODD, 80 BYTES/ROW

0494  B0 28                      MOV     AL,40
0496  52                         PUSH    DX              ; SAVE ROW VALUE
0497  80 E2 FE                   AND     DL,0FEH         ; STRIP OFF ODD/EVEN BIT
049A  F6 E2                      MUL     DL              ; AX HAS ADDRESS OF 1ST BYTE OF INDICATED ROW
049C  5A                         POP     DX              ; RECOVER IT
049D  F6 C2 01                   TEST    DL,1            ; TEST FOR EVEN/ODD
04A0  74 03                      JZ      R4              ; JUMP IF EVEN ROW
04A2  05 2000                    ADD     AX,2000H        ; OFFSET TO LOCATION OF ODD ROWS
04A5                     R4:                             ; EVEN_ROW
04A5  8B F0                      MOV     SI,AX           ; MOVE POINTER TO SI
04A7  58                         POP     AX              ; RECOVER AL VALUE
04A8  8B D1                      MOV     DX,CX           ; COLUMN VALUE TO DX

                         ;------ DETERMINE GRAPHICS MODE CURRENTLY IN EFFECT

                         ; SET UP THE REGISTERS ACCORDING TO THE MODE
                         ; CH = MASK FOR LOW OF COLUMN ADDRESS ( 7/3 FOR HIGH/MED RES)
                         ; CL = # OF ADDRESS BITS IN COLUMN VALUE ( 3/2 FOR H/M)
                         ; BL = MASK TO SELECT BITS FROM POINTED BYTE (80H/C0H FOR H/M)
                         ; BH = NUMBER OF VALID BITS IN POINTED BYTE ( 1/2 FOR H/M)

04AA  BB 02C0                    MOV     BX,2C0H
04AD  B9 0302                    MOV     CX,302H         ; SET PARMS FOR MED RES
04B0  80 3E 0049 R 06            CMP     CRT_MODE,6
04B5  72 06                      JC      R5              ; HANDLE IF MED ARES
04B7  BB 0180                    MOV     BX,180H
04BA  B9 0703                    MOV     CX,703H         ; SET PARMS FOR HIGH RES

                         ;------ DETERMINE BIT OFFSET IN BYTE FROM COLUMN MASK
04BD                     R5:
04BD  22 EA                      AND     CH,DL           ; ADDRESS OF PEL WITHIN BYTE TO CH

                         ;------ DETERMINE BYTE OFFSET FOR THIS LOCATION IN COLUMN

04BF  D3 EA                      SHR     DX,CL           ; SHIFT BY CORRECT AMOUNT
04C1  03 F2                      ADD     SI,DX           ; INCREMENT THE POINTER
04C3  8A F7                      MOV     DH,BH           ; GET THE # OF BITS IN RESULT TO DH

                         ;------ MULTIPLY BH (VALID BITS IN BYTE) BY CH (BIT OFFSET)

04C5  2A C9                      SUB     CL,CL           ; ZERO INTO STORAGE LOCATION
04C7                     R6:
04C7  D0 C8                      ROR     AL,1            ; LEFT JUSTIFY THE VALUE IN AL (FOR WRITE)
04C9  02 CD                      ADD     CL,CH           ; ADD IN THE BIT OFFSET VALUE
04CB  FE CF                      DEC     BH              ; LOOP CONTROL
04CD  75 F8                      JNZ     R6              ; ON EXIT, CL HAS SHIFT COUNT TO RESTORE BITS
04CF  8A E3                      MOV     AH,BL           ;  GET MASK TO AH
04D1  D2 EC                      SHR     AH,CL           ;  MOVE THE MASK TO CORRECT LOCATION
04D3  5B                         POP     BX              ;  RECOVER REG
04D4  C3                         RET                     ;  RETURN WITH EVERYTHING SET UP
04D5                     R3      ENDP
                         ;-----------------------------------------------------
                         ;  SCROLL UP
                         ;   THIS ROUTINE SCROLLS UP THE INFORMATION ON THE CRT
                         ;  ENTRY --
                         ;  CH,CL = UPPER LEFT CORNER OF REGION TO SCROLL
                         ;  DH,DL = LOWER RIGHT CORNER OF REGION TO SCROLL
                         ;   BOTH OF THE ABOVE ARE IN CHARACTER POSITIONS
                         ;  BH = FILL VALUE FOR BLANKED LINES
                         ;  AL = # LINES TO SCROLL (AL=0 MEANS BLANK THE ENTIRE FIELD)
                         ;  DS = DATA SEGMENT
                         ;  ES = REGEN SEGMENT
                         ; EXIT --
                         ;  NOTHING, THE SCREEN IS SCROLLED
                         ;-----------------------------------------------------
04D5                     GRAPHICS_UP     PROC    NEAR
04D5  8A D8                      MOV     BL,AL   ; SAVE LINE COUNT IN BL
04D7  8B C1                      MOV     AX,CX   ; GET UPPER LEFT POSITION INTO AX REG

                         ;------ USE CHARACTER SUBROUTINE FOR POSITIONING
                         ;------ ADDRESS RETURNED IS MULTIPLIED BY 2 FROM CORRECT VALUE

04D9  E8 0748 R                  CALL    GRAPH_POSN
04DC  8B F8                      MOV     DI,AX           ; SAVE RESULT AS DESTINATION ADDRESS

                         ;------ DETERMINE SIZE OF WINDOW

04DE  2B D1                      SUB     DX,CX
04E0  81 C2 0101                 ADD     DX,101H         ; ADJUST VALUES
04E4  D0 E6                      SAL     DH,1            ; MULTIPLY # ROWS BY 4 SINCE 8 VERT DOTS/CHAR
04E6  D0 E6                      SAL     DH,1            ;  AND EVEN/ODD ROWS

                         ;------ DETERMINE CRT MODE

04E8  80 3E 0049 R 06            CMP     CRT_MODE,6      ; TEST FOR MEDIUM RES
04ED  73 04                      JNC     R7              ; FIND_SOURCE

                         ;------ MEDIUM RES UP
04EF  D0 E2                      SAL     DL,1            ; # COLUMNS * 2, SINCE 2 BYTES/CHAR
04F1  D1 E7                      SAL     DI,1            ; OFFSET *2 SINCE 2 BYTES/CHAR

                         ;------ DETERMINE THE SOURCE ADDRESS IN THE BUFFER
04F3                     R7:                             ; FIND_SOURCE
04F3  06                         PUSH    ES              ; GET SEGMENTS BOTH POINTING TO REGEN
04F4  1F                         POP     DS
04F5  2A ED                      SUB     CH,CH           ; ZERO TO HIGH OF COUNT REG
04F7  D0 E3                      SAL     BL,1            ; MULTIPLY NUMBER OF LINES BY 4
04F9  D0 E3                      SAL     BL,1
04FB  74 2D                      JZ      R11             ; IF ZERO, THEN BLANK ENTIRE FIELD
04FD  8A C3                      MOV     AL,BL           ; GET NUMBER OF LINES IN AL
04FF  B4 50                      MOV     AH,80           ; 80 BYTES/ROW
0501  F6 E4                      MUL     AH              ; DETERMINE OFFSET TO SOURCE
0503  8B F7                      MOV     SI,DI           ; SET UP SOURCE
0505  03 F0                      ADD     SI,AX           ;  ADD IN OFFSET TO IT
0507  8A E6                      MOV     AH,DH           ; NUMBER OF ROWS IN FIELD
```

```
0509  2A E3                     SUB     AH,BL           ; DETERMINE NUMBER TO MOVE

                        ;------ LOOP THROUGH, MOVING ONE ROW AT A TIME, BOTH EVEN AND ODD FIELDS
050B                    R8:                             ; ROW_LOOP
050B  E8 058E R                 CALL    R17             ; MOVE ONE ROW
050E  81 EE 1FB0                SUB     SI,2000H-80     ; MOVE TO NEXT ROW
0512  81 EF 1FB0                SUB     DI,2000H-80
0516  FE CC                     DEC     AH              ; NUMBER OF ROWS TO MOVE
0518  75 F1                     JNZ     R8              ; CONTINUE TILL ALL MOVED

                        ;------ FILL IN THE VACATED LINE(S)
051A                    R9:                             ; CLEAR_ENTRY
051A  8A C7                     MOV     AL,BH           ; ATTRIBUTE TO FILL WITH
051C                    R10:
051C  E8 05A7 R                 CALL    R18             ; CLEAR THAT ROW
051F  81 EF 1FB0                SUB     DI,2000H-80     ; POINT TO NEXT LINE
0523  FE CB                     DEC     BL              ; NUMBER OF LINES TO FILL
0525  75 F5                     JNZ     R10             ; CLEAR_LOOP
0527  E9 0144 R                 JMP     VIDEO_RETURN    ; EVERYTHING DONE

052A                    R11:                            ; BLANK_FIELD
052A  8A DE                     MOV     BL,DH           ; SET BLANK COUNT TO EVERYTHING IN FIELD
052C  EB EC                     JMP     R9              ; CLEAR THE FIELD
052E                    GRAPHICS_UP     ENDP
                        ;----------------------------------------------------------
                        ; SCROLL DOWN
                        ;  THIS ROUTINE SCROLLS DOWN THE INFORMATION ON THE CRT
                        ; ENTRY --
                        ;  CH,CL = UPPER LEFT CORNER OF REGION TO SCROLL
                        ;  DH,DL = LOWER RIGHT CORNER OF REGION TO SCROLL
                        ;   BOTH OF THE ABOVE ARE IN CHARACTER POSITIONS
                        ;  BH = FILL VALUE FOR BLANKED LINES
                        ;  AL = # LINES TO SCROLL (AL=0 MEANS BLANK THE ENTIRE FIELD)
                        ;  DS = DATA SEGMENT
                        ;  ES = REGEN SEGMENT
                        ; EXIT --
                        ;  NOTHING, THE SCREEN IS SCROLLED
                        ;----------------------------------------------------------

052E                    GRAPHICS_DOWN   PROC    NEAR
052E  FD                        STD             ; SET DIRECTION
052F  8A D8                     MOV     BL,AL   ; SAVE LINE COUNT IN BL
0531  8B C2                     MOV     AX,DX   ; GET LOWER RIGHT POSITION INTO AX REG

                        ;------ USE CHARACTER SUBROUTINE FOR POSITIONING
                        ;------ ADDRESS RETURNED IS MULTIPLIED BY 2 FROM CORRECT VALUE

0533  E8 0748 R                 CALL    GRAPH_POSN
0536  8B F8                     MOV     DI,AX           ; SAVE RESULT AS DESTINATION ADDRESS

                        ;------ DETERMINE SIZE OF WINDOW

0538  2B D1                     SUB     DX,CX
053A  81 C2 0101                ADD     DX,101H         ; ADJUST VALUES
053E  D0 E6                     SAL     DH,1            ; MULTIPLY # ROWS BY 4 SINCE 8 VERT DOTS/CHAR
0540  D0 E6                     SAL     DH,1            ;  AND EVEN/ODD ROWS

                        ;------ DETERMINE CRT MODE

0542  80 3E 0049 R 06           CMP     CRT_MODE,6      ; TEST FOR MEDIUM RES
0547  73 05                     JNC     R12             ; FIND_SOURCE_DOWN

                        ;------ MEDIUM RES DOWN

0549  D0 E2                     SAL     DL,1            ; # COLUMNS * 2, SINCE 2 BYTES/CHAR (OFFSET OK)
054B  D1 E7                     SAL     DI,1            ; OFFSET *2 SINCE 2 BYTES/CHAR
054D  47                        INC     DI              ; POINT TO LAST BYTE

                        ;------ DETERMINE THE SOURCE ADDRESS IN THE BUFFER
054E                    R12:                            ; FIND_SOURCE_DOWN
054E  06                        PUSH    ES              ; BOTH SEGMENTS TO REGEN
054F  1F                        POP     DS
0550  2A ED                     SUB     CH,CH           ; ZERO TO HIGH OF COUNT REG
0552  81 C7 00F0                ADD     DI,240          ; POINT TO LAST ROW OF PIXELS
0556  D0 E3                     SAL     BL,1            ; MULTIPLY NUMBER OF LINES BY 4
0558  D0 E3                     SAL     BL,1
055A  74 2E                     JZ      R16             ; IF ZERO, THEN BLANK ENTIRE FIELD
055C  8A C3                     MOV     AL,BL           ; GET NUMBER OF LINES IN AL
055E  B4 50                     MOV     AH,80           ; 80 BYTES/ROW
0560  F6 E4                     MUL     AH              ; DETERMINE OFFSET TO SOURCE
0562  8B F7                     MOV     SI,DI           ; SET UP SOURCE
0564  2B F0                     SUB     SI,AX           ;  SUBTRACT THE OFFSET
0566  8A E6                     MOV     AH,DH           ; NUMBER OF ROWS IN FIELD
0568  2A E3                     SUB     AH,BL           ; DETERMINE NUMBER TO MOVE

                        ;------ LOOP THROUGH, MOVING ONE ROW AT A TIME, BOTH EVEN AND ODD FIELDS
056A                    R13:                            ; ROW_LOOP_DOWN
056A  E8 058E R                 CALL    R17             ; MOVE ONE ROW
056D  81 EE 2050                SUB     SI,2000H+80     ; MOVE TO NEXT ROW
0571  81 EF 2050                SUB     DI,2000H+80
0575  FE CC                     DEC     AH              ; NUMBER OF ROWS TO MOVE
0577  75 F1                     JNZ     R13             ; CONTINUE TILL ALL MOVED

                        ;------ FILL IN THE VACATED LINE(S)
0579                    R14:                            ; CLEAR_ENTRY_DOWN
0579  8A C7                     MOV     AL,BH           ; ATTRIBUTE TO FILL WITH
057B                    R15:                            ; CLEAR_LOOP_DOWN
057B  E8 05A7 R                 CALL    R18             ; CLEAR A ROW
057E  81 EF 2050                SUB     DI,2000H+80     ; POINT TO NEXT LINE
0582  FE CB                     DEC     BL              ; NUMBER OF LINES TO FILL
0584  75 F5                     JNZ     R15             ; CLEAR_LOOP_DOWN
0586  FC                        CLD                     ; RESET THE DIRECTION FLAG
0587  E9 0144 R                 JMP     VIDEO_RETURN    ; EVERYTHING DONE

058A                    R16:                            ; BLANK_FIELD_DOWN
058A  8A DE                     MOV     BL,DH           ; SET BLANK COUNT TO EVERYTHING IN FIELD
058C  EB EB                     JMP     R14             ; CLEAR THE FIELD
058E                    GRAPHICS_DOWN   ENDP

                        ;------ ROUTINE TO MOVE ONE ROW OF INFORMATION

058E                    R17     PROC    NEAR
058E  8A CA                     MOV     CL,DL           ; NUMBER OF BYTES IN THE ROW
0590  56                        PUSH    SI
0591  57                        PUSH    DI              ; SAVE POINTERS
0592  F3/ A4                    REP     MOVSB           ; MOVE THE EVEN FIELD
0594  5F                        POP     DI
0595  5E                        POP     SI
0596  81 C6 2000                ADD     SI,2000H
059A  81 C7 2000                ADD     DI,2000H                ; POINT TO THE ODD FIELD
059E  56                        PUSH    SI
059F  57                        PUSH    DI              ; SAVE THE POINTERS
05A0  8A CA                     MOV     CL,DL           ; COUNT BACK
05A2  F3/ A4                    REP     MOVSB           ; MOVE THE ODD FIELD
05A4  5F                        POP     DI
05A5  5E                        POP     SI              ; POINTERS BACK
05A6  C3                        RET                     ; RETURN TO CALLER
```

SECTION 5

```
05A7                          R17     ENDP

                              ;------ CLEAR A SINGLE ROW

05A7                          R18     PROC    NEAR
05A7  8A CA                           MOV     CL,DL              ; NUMBER OF BYTES IN FIELD
05A9  57                              PUSH    DI                 ; SAVE POINTER
05AA  F3/ AA                          REP     STOSB              ; STORE THE NEW VALUE
05AC  5F                              POP     DI                 ; POINTER BACK
05AD  81 C7 2000                      ADD     DI,2000H                   ; POINT TO ODD FIELD
05B1  57                              PUSH    DI
05B2  8A CA                           MOV     CL,DL
05B4  F3/ AA                          REP     STOSB              ; FILL THE ODD FILELD
05B6  5F                              POP     DI
05B7  C3                              RET                        ; RETURN TO CALLER
05B8                          R18     ENDP
                              ;------------------------------------------------
                              ; GRAPHICS WRITE
                              ;  THIS ROUTINE WRITES THE ASCII CHARACTER TO THE CURRENT
                              ;  POSITION ON THE SCREEN.
                              ; ENTRY --
                              ;  AL = CHARACTER TO WRITE
                              ;  BL = COLOR ATTRIBUTE TO BE USED FOR FOREGROUND COLOR
                              ;       IF BIT 7 IS SET, THE CHAR IS XOR'D INTO THE REGEN BUFFER
                              ;       (0 IS USED FOR THE BACKGROUND COLOR)
                              ;  CX = NUMBER OF CHARS TO WRITE
                              ;  DS = DATA SEGMENT
                              ;  ES = REGEN SEGMENT
                              ; EXIT --
                              ;  NOTHING IS RETURNED
                              ;
                              ; GRAPHICS READ
                              ;   THIS ROUTINE READS THE ASCII CHARACTER AT THE CURRENT CURSOR
                              ;   POSITION ON THE SCREEN BY MATCHING THE DOTS ON THE SCREEN TO THE
                              ;   CHARACTER GENERATOR CODE POINTS
                              ; ENTRY --
                              ;  NONE  (0 IS ASSUMED AS THE BACKGROUND COLOR)
                              ; EXIT --
                              ;  AL = CHARACTER READ AT THAT POSITION (0 RETURNED IF NONE FOUND)
                              ;
                              ; FOR BOTH ROUTINES, THE IMAGES USED TO FORM CHARS ARE CONTAINED IN ROM
                              ;  FOR THE 1ST 128 CHARS.  TO ACCESS CHARS IN THE SECOND HALF, THE USER
                              ;  MUST INITIALIZE THE VECTOR AT INTERRUPT 1FH (LOCATION 0007CH) TO
                              ;  POINT TO THE USER SUPPLIED TABLE OF GRAPHIC IMAGES (8X8 BOXES).
                              ;  FAILURE TO DO SO WILL CAUSE IN STRANGE RESULTS
                              ;------------------------------------------------
                                      ASSUME  CS:CODE,DS:DATA,ES:DATA
05B8                          GRAPHICS_WRITE  PROC    NEAR
05B8  B4 00                           MOV     AH,0               ; ZERO TO HIGH OF CODE POINT
05BA  50                              PUSH    AX                 ; SAVE CODE POINT VALUE

                              ;------ DETERMINE POSITION IN REGEN BUFFER TO PUT CODE POINTS

05BB  E8 0745 R                       CALL    S26                ; FIND LOCATION IN REGEN BUFFER
05BE  8B F8                           MOV     DI,AX              ; REGEN POINTER IN DI

                              ;------ DETERMINE REGION TO GET CODE POINTS FROM

05C0  58                              POP     AX                 ; RECOVER CODE POINT
05C1  3C 80                           CMP     AL,80H             ; IS IT IN SECOND HALF
05C3  73 06                           JAE     S1                 ; YES

                              ;------ IMAGE IS IN FIRST HALF, CONTAINED IN ROM

05C5  BE 0000 E                       MOV     SI,OFFSET CRT_CHAR_GEN  ; OFFSET OF IMAGES
05C8  0E                              PUSH    CS                      ; SAVE SEGMENT ON STACK
05C9  EB 0F                           JMP     SHORT S2                ; DETERMINE_MODE

                              ;------ IMAGE IS IN SECOND HALF, IN USER RAM

05CB                          S1:                                     ; EXTEND_CHAR
05CB  2C 80                           SUB     AL,80H                  ; ZERO ORIGIN FOR SECOND HALF
05CD  1E                              PUSH    DS                      ; SAVE DATA POINTER
05CE  2B F6                           SUB     SI,SI
05D0  8E DE                           MOV     DS,SI                   ; ESTABLISH VECTOR ADDRESSING
                                      ASSUME  DS:ABS0
05D2  C5 36 007C R                    LDS     SI,EXT_PTR              ; GET THE OFFSET OF THE TABLE
05D6  8C DA                           MOV     DX,DS                   ; GET THE SEGMENT OF THE TABLE
                                      ASSUME  DS:DATA
05D8  1F                              POP     DS                      ; RECOVER DATA SEGMENT
05D9  52                              PUSH    DX                      ; SAVE TABLE SEGMENT ON STACK

                              ;------ DETERMINE GRAPHICS MODE IN OPERATION

05DA                          S2:                                     ; DETERMINE_MODE
05DA  D1 E0                           SAL     AX,1                    ; MULTIPLY CODE POINT
05DC  D1 E0                           SAL     AX,1                    ;  VALUE BY 8
05DE  D1 E0                           SAL     AX,1
05E0  03 F0                           ADD     SI,AX                   ; SI HAS OFFSET OF DESIRED CODES
05E2  80 3E 0049 R 06                 CMP     CRT_MODE,6
05E7  1F                              POP     DS                 ; RECOVER TABLE POINTER SEGMENT
05E8  72 2C                           JC      S7                 ; TEST FOR MEDIUM RESOLUTION MODE

                              ;------ HIGH RESOLUTION MODE
05EA                          S3:                                ; HIGH_CHAR
05EA  57                              PUSH    DI                 ; SAVE REGEN POINTER
05EB  56                              PUSH    SI                 ; SAVE CODE POINTER
05EC  B6 04                           MOV     DH,4               ; NUMBER OF TIMES THROUGH LOOP
05EE                          S4:
05EE  AC                              LODSB                      ; GET BYTE FROM CODE POINTS
05EF  F6 C3 80                        TEST    BL,80H             ; SHOULD WE USE THE FUNCTION
05F2  75 16                           JNZ     S6                 ;  TO PUT CHAR IN
05F4  AA                              STOSB                      ; STORE IN REGEN BUFFER
05F5  AC                              LODSB
05F6                          S5:                                ;
05F6  26: 88 85 1FFF                  MOV     ES:[DI+2000H-1],AL         ; STORE IN SECOND HALF
05FB  83 C7 4F                        ADD     DI,79              ; MOVE TO NEXT ROW IN REGEN
05FE  FE CE                           DEC     DH                 ; DONE WITH LOOP
0600  75 EC                           JNZ     S4
0602  5E                              POP     SI
0603  5F                              POP     DI                 ; RECOVER REGEN POINTER
0604  47                              INC     DI                 ; POINT TO NEXT CHAR POSITION
0605  E2 E3                           LOOP    S3                 ; MORE CHARS TO WRITE
0607  E9 0144 R                       JMP     VIDEO_RETURN

060A                          S6:
060A  26: 32 05                       XOR     AL,ES:[DI]         ; EXCLUSIVE OR WITH CURRENT
060D  AA                              STOSB                      ; STORE THE CODE POINT
060E  AC                              LODSB                      ; AGAIN FOR ODD FIELD
060F  26: 32 85 1FFF                  XOR     AL,ES:[DI+2000H-1]         ;
0614  EB E0                           JMP     S5                 ; BACK TO MAINSTREAM

                              ;------ MEDIUM RESOLUTION WRITE
0616                          S7:                                ; MED_RES_WRITE
0616  8A D3                           MOV     DL,BL              ; SAVE HIGH COLOR BIT
0618  D1 E7                           SAL     DI,1               ; OFFSET*2 SINCE 2 BYTES/CHAR
```

```
061A  E8 06F1 R                  CALL   S19                ; EXPAND BL TO FULL WORD OF COLOR
061D                      S8:                              ; MED_CHAR
061D  57                         PUSH   DI                 ; SAVE REGEN POINTER
061E  56                         PUSH   SI                     ; SAVE THE CODE POINTER
061F  B6 04                      MOV    DH,4               ; NUMBER OF LOOPS
0621                      S9:
0621  AC                         LODSB                     ; GET CODE POINT
0622  E8 0706 R                  CALL   S21                ; DOUBLE UP ALL THE BITS
0625  23 C3                      AND    AX,BX              ; CONVERT THEM TO FOREGROUND COLOR ( 0 BACK )
0627  F6 C2 80                   TEST   DL,80H             ; IS THIS XOR FUNCTION
062A  74 07                      JZ     S10                ; NO, STORE IT IN AS IT IS
062C  26: 32 25                  XOR    AH,ES:[DI]         ; DO FUNCTION WITH HALF
062F  26: 32 45 01               XOR    AL,ES:[DI+1]       ;  AND WITH OTHER HALF
0633                      S10:                             ;
0633  26: 88 25                  MOV    ES:[DI],AH         ; STORE FIRST BYTE
0636  26: 88 45 01               MOV    ES:[DI+1],AL       ; STORE SECOND BYTE
063A  AC                         LODSB                     ; GET CODE POINT
063B  E8 0706 R                  CALL   S21
063E  23 C3                      AND    AX,BX              ; CONVERT TO COLOR
0640  F6 C2 80                   TEST   DL,80H             ; AGAIN, IS THIS XOR FUNCTION
0643  74 0A                      JZ     S11                ; NO, JUST STORE THE VALUES
0645  26: 32 A5 2000             XOR    AH,ES:[DI+2000H]       ; FUNCTION WITH FIRST HALF
064A  26: 32 85 2001             XOR    AL,ES:[DI+2001H]       ; AND WITH SECOND HALF
064F                      S11:                             ;
064F  26: 88 A5 2000             MOV    ES:[DI+2000H],AH
0654  26: 88 85 2001             MOV    ES:[DI+2000H+1],AL     ; STORE IN SECOND PORTION OF BUFFER
0659  83 C7 50                   ADD    DI,80              ; POINT TO NEXT LOCATION
065C  FE CE                      DEC    DH
065E  75 C1                      JNZ    S9                 ; KEEP GOING
0660  5E                         POP    SI                 ; RECOVER CODE PONTER
0661  5F                         POP    DI                 ; RECOVER REGEN POINTER
0662  47                         INC    DI                 ; POINT TO NEXT CHAR POSITION
0663  47                         INC    DI
0664  E2 B7                      LOOP   S8                 ; MORE TO WRITE
0666  E9 0144 R                  JMP    VIDEO_RETURN
0669                      GRAPHICS_WRITE  ENDP
                          ;---------------------------------------
                          ; GRAPHICS READ
                          ;---------------------------------------
0669                      GRAPHICS_READ   PROC    NEAR
0669  E8 0745 R                  CALL   S26                ; CONVERTED TO OFFSET IN REGEN
066C  8B F0                      MOV    SI,AX              ; SAVE IN SI
066E  83 EC 08                   SUB    SP,8               ; ALLOCATE SPACE TO SAVE THE READ CODE POINT
0671  8B EC                      MOV    BP,SP              ; POINTER TO SAVE AREA

                          ;------ DETERMINE GRAPHICS MODES

0673  80 3E 0049 R 06            CMP    CRT_MODE,6
0678  06                         PUSH   ES
0679  1F                         POP    DS                 ; POINT TO REGEN SEGMENT
067A  72 1A                      JC     S13                ; MEDIUM RESOLUTION

                          ;------ HIGH RESOLUTION READ

                          ;------ GET VALUES FROM REGEN BUFFER AND CONVERT TO CODE POINT
067C  B6 04                      MOV    DH,4               ; NUMBER OF PASSES
067E                      S12:
067E  8A 04                      MOV    AL,[SI]            ; GET FIRST BYTE
0680  88 46 00                   MOV    [BP],AL            ; SAVE IN STORAGE AREA
0683  45                         INC    BP                 ; NEXT LOCATION
0684  8A 84 2000                 MOV    AL,[SI+2000H]      ; GET LOWER REGION BYTE
0688  88 46 00                   MOV    [BP],AL            ; ADJUST AND STORE
068B  45                         INC    BP
068C  83 C6 50                   ADD    SI,80              ; POINTER INTO REGEN
068F  FE CE                      DEC    DH                 ; LOOP CONTROL
0691  75 EB                      JNZ    S12                ; DO IT SOME MORE
0693  EB 17 90                   JMP    S15                ; GO MATCH THE SAVED CODE POINTS

                          ;------ MEDIUM RESOLUTION READ
0696                      S13:                             ; MED_RES_READ
0696  D1 E6                      SAL    SI,1               ; OFFSET*2 SINCE 2 BYTES/CHAR
0698  B6 04                      MOV    DH,4               ; NUMBER OF PASSES
069A                      S14:
069A  E8 0728 R                  CALL   S23                ; GET PAIR BYTES FROM REGEN INTO SINGLE SAVE
069D  81 C6 2000                 ADD    SI,2000H           ; GO TO LOWER REGION
06A1  E8 0728 R                  CALL   S23                ; GET THIS PAIR INTO SAVE
06A4  81 EE 1FB0                 SUB    SI,2000H-80        ; ADJUST POINTER BACK INTO UPPER
06A8  FE CE                      DEC    DH
06AA  75 EE                      JNZ    S14                ; KEEP GOING UNTIL ALL 8 DONE

                          ;-------- SAVE AREA HAS CHARACTER IN IT, MATCH IT
06AC                      S15:                             ; FIND_CHAR
06AC  BF 0000 E                  MOV    DI,OFFSET CRT_CHAR_GEN  ; ESTABLISH ADDRESSING
06AF  0E                         PUSH   CS
06B0  07                         POP    ES                     ; CODE POINTS IN CS
06B1  83 ED 08                   SUB    BP,8               ; ADJUST POINTER TO BEGINNING OF SAVE AREA
06B4  8B F5                      MOV    SI,BP
06B6  FC                         CLD                       ; ENSURE DIRECTION
06B7  B0 00                      MOV    AL,0               ; CURRENT CODE POINT BEING MATCHED
06B9                      S16:
06B9  16                         PUSH   SS                 ; ESTABLISH ADDRESSING TO STACK
06BA  1F                         POP    DS                 ; FOR THE STRING COMPARE
06BB  BA 0080                    MOV    DX,128             ; NUMBER TO TEST AGAINST
06BE                      S17:
06BE  56                         PUSH   SI                 ; SAVE SAVE AREA POINTER
06BF  57                         PUSH   DI                 ; SAVE CODE POINTER
06C0  B9 0008                    MOV    CX,8               ; NUMBER OF BYTES TO MATCH
06C3  F3/ A6                     REPE   CMPSB              ; COMPARE THE 8 BYTES
06C5  8A 1E 0017 R               MOV    BL,KB_FLAG         ; READ ANY BYTE OF STORAGE
06C9  5F                         POP    DI                 ; RECOVER THE POINTERS
06CA  5E                         POP    SI
06CB  74 1E                      JZ     S18                ; IF ZERO FLAG SET, THEN MATCH OCCURRED
06CD  FE C0                      INC    AL                 ; NO MATCH, MOVE ON TO NEXT
06CF  83 C7 08                   ADD    DI,8               ; NEXT CODE POINT
06D2  4A                         DEC    DX                 ; LOOP CONTROL
06D3  75 E9                      JNZ    S17                ; DO ALL OF THEM

                          ;------ CHAR NOT MATCHED, MIGHT BE IN USER SUPPLIED SECOND HALF

06D5  3C 00                      CMP    AL,0               ;  AL<> 0 IF ONLY 1ST HALF SCANNED
06D7  74 12                      JE     S18                ;  IF = 0, THEN ALL HAS BEEN SCANNED
06D9  2B C0                      SUB    AX,AX
06DB  8E D8                      MOV    DS,AX              ;  ESTABLISH ADDRESSING TO VECTOR
                                 ASSUME DS:ABS0
06DD  C4 3E 007C R               LES    DI,EXT_PTR         ;  GET POINTER
06E1  8C C0                      MOV    AX,ES              ;  SEE IF THE POINTER REALLY EXISTS
06E3  0B C7                      OR     AX,DI              ; IF ALL 0, THEN DOESN'T EXIST
06E5  74 04                      JZ     S18                ; NO SENSE LOOKING
06E7  B0 80                      MOV    AL,128             ; ORIGIN FOR SECOND HALF
06E9  EB CE                      JMP    S16                ; GO BACK AND TRY FOR IT
                                 ASSUME DS:DATA

                          ;------ CHARACTER IS FOUND ( AL=0 IF NOT FOUND )
06EB                      S18:
06EB  83 C4 08                   ADD    SP,8               ; READJUST THE STACK, THROW AWAY SAVE
06EE  E9 0144 R                  JMP    VIDEO_RETURN       ; ALL DONE
```

SECTION 5

```
06F1                          GRAPHICS_READ   ENDP
                              ;---------------------------------------------
                              ; EXPAND_MED_COLOR
                              ;  THIS ROUTINE EXPANDS THE LOW 2 BITS IN BL TO
                              ;  FILL THE ENTIRE BX REGISTER
                              ; ENTRY --
                              ;  BL = COLOR TO BE USED ( LOW 2 BITS )
                              ; EXIT --
                              ;  BX = COLOR TO BE USED ( 8 REPLICATIONS OF THE 2 COLOR BITS )
                              ;---------------------------------------------
06F1                          S19     PROC    NEAR
06F1  80 E3 03                        AND     BL,3            ; ISOLATE THE COLOR BITS
06F4  8A C3                           MOV     AL,BL           ; COPY TO AL
06F6  51                              PUSH    CX              ; SAVE REGISTER
06F7  B9 0003                         MOV     CX,3            ; NUMBER OF TIMES TO DO THIS
06FA                          S20:
06FA  D0 E0                           SAL     AL,1
06FC  D0 E0                           SAL     AL,1            ; LEFT SHIFT BY 2
06FE  0A D8                           OR      BL,AL           ; ANOTHER COLOR VERSION INTO BL
0700  E2 F8                           LOOP    S20             ; FILL ALL OF BL
0702  8A FB                           MOV     BH,BL           ; FILL UPPER PORTION
0704  59                              POP     CX              ; REGISTER BACK
0705  C3                              RET                     ; ALL DONE
0706                          S19     ENDP
                              ;---------------------------------------------
                              ; EXPAND_BYTE
                              ;  THIS ROUTINE TAKES THE BYTE IN AL AND DOUBLES ALL
                              ;  OF THE BITS, TURNING THE 8 BITS INTO 16 BITS.
                              ;  THE RESULT IS LEFT IN AX
                              ;---------------------------------------------
0706                          S21     PROC    NEAR
0706  52                              PUSH    DX              ; SAVE REGISTERS
0707  51                              PUSH    CX
0708  53                              PUSH    BX
0709  2B D2                           SUB     DX,DX           ; RESULT REGISTER
070B  B9 0001                         MOV     CX,1            ; MASK REGISTER
070E                          S22:
070E  8B D8                           MOV     BX,AX           ; BASE INTO TEMP
0710  23 D9                           AND     BX,CX           ; USE MASK TO EXTRACT A BIT
0712  0B D3                           OR      DX,BX           ; PUT INTO RESULT REGISTER
0714  D1 E0                           SHL     AX,1
0716  D1 E1                           SHL     CX,1            ; SHIFT BASE AND MASK BY 1
0718  8B D8                           MOV     BX,AX           ; BASE TO TEMP
071A  23 D9                           AND     BX,CX           ; EXTRACT THE SAME BIT
071C  0B D3                           OR      DX,BX           ; PUT INTO RESULT
071E  D1 E1                           SHL     CX,1            ; SHIFT ONLY MASK NOW, MOVING TO NEXT BASE
0720  73 EC                           JNC     S22             ; USE MASK BIT COMING OUT TO TERMINATE
0722  8B C2                           MOV     AX,DX           ; RESULT TO PARM REGISTER
0724  5B                              POP     BX
0725  59                              POP     CX              ; RECOVER REGISTERS
0726  5A                              POP     DX
0727  C3                              RET                     ; ALL DONE
0728                          S21     ENDP
                              ;---------------------------------------------
                              ; MED_READ_BYTE
                              ; THIS ROUTINE WILL TAKE 2 BYTES FROM THE REGEN BUFFER,
                              ;  COMPARE AGAINST THE CURRENT FOREGROUND COLOR, AND PLACE
                              ;  THE CORRESPONDING ON/OFF BIT PATTERN INTO THE CURRENT
                              ;  POSITION IN THE SAVE AREA
                              ; ENTRY --
                              ;  SI,DS = POINTER TO REGEN AREA OF INTEREST
                              ;  BX = EXPANDED FOREGROUND COLOR
                              ;  BP = POINTER TO SAVE AREA
                              ; EXIT --
                              ;  BP IS INCREMENT AFTER SAVE
                              ;---------------------------------------------
0728                          S23     PROC    NEAR
0728  8A 24                           MOV     AH,[SI]         ; GET FIRST BYTE
072A  8A 44 01                        MOV     AL,[SI+1]       ; GET SECOND BYTE
072D  B9 C000                         MOV     CX,0C000H       ; 2 BIT MASK TO TEST THE ENTRIES
0730  B2 00                           MOV     DL,0            ; RESULT REGISTER
0732                          S24:
0732  85 C1                           TEST    AX,CX           ; IS THIS SECTION BACKGROUND?
0734  F8                              CLC                     ; CLEAR CARRY IN HOPES THAT IT IS
0735  74 01                           JZ      S25             ; IF ZERO, IT IS BACKGROUND
0737  F9                              STC                     ; WASN'T, SO SET CARRY
0738  D0 D2                   S25:    RCL     DL,1            ; MOVE THAT BIT INTO THE RESULT
073A  D1 E9                           SHR     CX,1
073C  D1 E9                           SHR     CX,1            ; MOVE THE MASK TO THE RIGHT BY 2 BITS
073E  73 F2                           JNC     S24             ; DO IT AGAIN IF MASK DIDN'T FALL OUT
0740  88 56 00                        MOV     [BP],DL         ; STORE RESULT IN SAVE AREA
0743  45                              INC     BP              ; ADJUST POINTER
0744  C3                              RET                     ; ALL DONE
0745                          S23     ENDP
                              ;---------------------------------------------
                              ; V4_POSITION
                              ;  THIS ROUTINE TAKES THE CURSOR POSITION CONTAINED IN
                              ;  THE MEMORY LOCATION, AND CONVERTS IT INTO AN OFFSET
                              ;  INTO THE REGEN BUFFER, ASSUMING ONE BYTE/CHAR.
                              ;  FOR MEDIUM RESOLUTION GRAPHICS, THE NUMBER MUST
                              ;  BE DOUBLED.
                              ; ENTRY -- NO REGISTERS,MEMORY LOCATION CURSOR_POSN IS USED
                              ; EXIT--
                              ;  AX CONTAINS OFFSET INTO REGEN BUFFER
                              ;---------------------------------------------
0745                          S26     PROC    NEAR
0745  A1 0050 R                       MOV     AX,CURSOR_POSN          ; GET CURRENT CURSOR
0748                          GRAPH_POSN      LABEL   NEAR
0748  53                              PUSH    BX                      ; SAVE REGISTER
0749  8B D8                           MOV     BX,AX                   ; SAVE A COPY OF CURRENT CURSOR
074B  8A C4                           MOV     AL,AH                   ; GET ROWS TO AL
074D  F6 26 004A R                    MUL     BYTE PTR CRT_COLS       ; MULTIPLY BY BYTES/COLUMN
0751  D1 E0                           SHL     AX,1                    ; MULTIPLY * 4 SINCE 4 ROWS/BYTE
0753  D1 E0                           SHL     AX,1
0755  2A FF                           SUB     BH,BH                   ; ISOLATE COLUMN VALUE
0757  03 C3                           ADD     AX,BX                   ; DETERMINE OFFSET
0759  5B                              POP     BX                      ; RECOVER POINTER
075A  C3                              RET                             ; ALL DONE
075B                          S26     ENDP
                              ;---------------------------------------------
                              ; WRITE_TTY
                              ; THIS INTERFACE PROVIDES A TELETYPE LIKE INTERFACE TO THE
                              ;   VIDEO CARD.  THE INPUT CHARACTER IS WRITTEN TO THE CURRENT
                              ;   CURSOR POSITION, AND THE CURSOR IS MOVED TO THE NEXT POSITION.
                              ;   IF THE CURSOR LEAVES THE LAST COLUMN OF THE FIELD, THE COLUMN
                              ;   IS SET TO ZERO, AND THE ROW VALUE IS INCREMENTED.  IF THE ROW
                              ;   ROW VALUE LEAVES THE FIELD, THE CURSOR IS PLACED ON THE LAST ROW,
                              ;   FIRST COLUMN, AND THE ENTIRE SCREEN IS SCROLLED UP ONE LINE.
                              ;   WHEN THE SCREEN IS SCROLLED UP, THE ATTRIBUTE FOR FILLING THE
                              ;   NEWLY BLANKED LINE IS READ FROM THE CURSOR POSITION ON THE PREVIOUS
                              ;   LINE BEFORE THE SCROLL, IN CHARACTER MODE.  IN GRAPHICS MODE,
                              ;   THE 0 COLOR IS USED.
                              ; ENTRY --
                              ;   (AH) = CURRENT CRT MODE
                              ;   (AL) = CHARACTER TO BE WRITTEN
                              ;         NOTE THAT BACK SPACE, CAR RET, BELL AND LINE FEED ARE HANDLED
```

```
                                ;           AS COMMANDS RATHER THAN AS DISPLAYABLE GRAPHICS
                                ;    (BL) = FOREGROUND COLOR FOR CHAR WRITE IF CURRENTLY IN A GRAPHICS MODE
                                ;  EXIT --
                                ;     ALL REGISTERS SAVED
                                ;--------------------------------------------------
                                        ASSUME  CS:CODE,DS:DATA
075B                            WRITE_TTY       PROC    NEAR
075B  50                                PUSH    AX              ; SAVE REGISTERS
075C  50                                PUSH    AX              ; SAVE CHAR TO WRITE
075D  B4 03                             MOV     AH,3
075F  8A 3E 0062 R                      MOV     BH,ACTIVE_PAGE  ; GET CURRENT PAGE SETTING
0763  CD 10                             INT     10H             ; READ THE CURRENT CURSOR POSITION
0765  58                                POP     AX              ; RECOVER CHAR

                                ;----- DX NOW HAS THE CURRENT CURSOR POSITION

0766  3C 08                             CMP     AL,8                    ; IS IT A BACKSPACE
0768  74 52                             JE      U8                      ; BACK_SPACE
076A  3C 0D                             CMP     AL,0DH                  ; IS IT CARRIAGE RETURN
076C  74 57                             JE      U9                      ; CAR_RET
076E  3C 0A                             CMP     AL,0AH                  ; IS IT A LINE FEED
0770  74 57                             JE      U10                     ; LINE_FEED
0772  3C 07                             CMP     AL,07H                  ; IS IT A BELL
0774  74 5A                             JE      U11                     ; BELL

                                ;----- WRITE THE CHAR TO THE SCREEN

0776  B4 0A                             MOV     AH,10                   ; WRITE CHAR ONLY
0778  B9 0001                           MOV     CX,1                    ; ONLY ONE CHAR
077B  CD 10                             INT     10H                     ; WRITE THE CHAR

                                ;----- POSITION THE CURSOR FOR NEXT CHAR

077D  FE C2                             INC     DL
077F  3A 16 004A R                      CMP     DL,BYTE PTR CRT_COLS    ; TEST FOR COLUMN OVERFLOW
0783  75 33                             JNZ     U7                      ; SET_CURSOR
0785  B2 00                             MOV     DL,0                    ; COLUMN FOR CURSOR
0787  80 FE 18                          CMP     DH,24
078A  75 2A                             JNZ     U6                      ; SET_CURSOR_INC

                                ;----- SCROLL REQUIRED
078C                            U1:

078C  B4 02                             MOV     AH,2
078E  CD 10                             INT     10H                     ; SET THE CURSOR

                                ;----- DETERMINE VALUE TO FILL WITH DURING SCROLL

0790  A0 0049 R                         MOV     AL,CRT_MODE             ; GET THE CURRENT MODE
0793  3C 04                             CMP     AL,4
0795  72 06                             JC      U2                      ; READ-CURSOR
0797  3C 07                             CMP     AL,7
0799  B7 00                             MOV     BH,0                    ; FILL WITH BACKGROUND
079B  75 06                             JNE     U3                      ; SCROLL-UP

079D                            U2:                                     ; READ-CURSOR
079D  B4 08                             MOV     AH,8
079F  CD 10                             INT     10H                     ; READ CHAR/ATTR AT CURRENT CURSOR
07A1  8A FC                             MOV     BH,AH                   ; STORE IN BH

07A3                            U3:                                     ; SCROLL-UP
07A3  B8 0601                           MOV     AX,601H                 ; SCROLL ONE LINE
07A6  2B C9                             SUB     CX,CX                   ; UPPER LEFT CORNER
07A8  B6 18                             MOV     DH,24                   ; LOWER RIGHT ROW
07AA  8A 16 004A R                      MOV     DL,BYTE PTR CRT_COLS    ; LOWER RIGHT COLUMN
07AE  FE CA                             DEC     DL
07B0                            U4:                                     ; VIDEO-CALL-RETURN
07B0  CD 10                             INT     10H                     ; SCROLL UP THE SCREEN
07B2                            U5:                                     ; TTY-RETURN
07B2  58                                POP     AX                      ; RESTORE THE CHARACTER
07B3  E9 0144 R                         JMP     VIDEO_RETURN            ; RETURN TO CALLER

07B6                            U6:                                     ; SET-CURSOR-INC
07B6  FE C6                             INC     DH                      ; NEXT ROW
07B8                            U7:                                     ; SET-CURSOR
07B8  B4 02                             MOV     AH,2
07BA  EB F4                             JMP     U4                      ; ESTABLISH THE NEW CURSOR

                                ;----- BACK SPACE FOUND

07BC                            U8:
07BC  80 FA 00                          CMP     DL,0                    ; ALREADY AT END OF LINE
07BF  74 F7                             JE      U7                      ; SET_CURSOR
07C1  FE CA                             DEC     DL                      ; NO -- JUST MOVE IT BACK
07C3  EB F3                             JMP     U7                      ; SET_CURSOR

                                ;----- CARRIAGE RETURN FOUND

07C5                            U9:
07C5  B2 00                             MOV     DL,0                    ; MOVE TO FIRST COLUMN
07C7  EB EF                             JMP     U7                      ; SET_CURSOR

                                ;----- LINE FEED FOUND

07C9                            U10:
07C9  80 FE 18                          CMP     DH,24                   ; BOTTOM OF SCREEN
07CC  75 E8                             JNE     U6                      ; YES, SCROLL THE SCREEN
07CE  EB BC                             JMP     U1                      ; NO, JUST SET THE CURSOR

                                ;----- BELL FOUND

07D0                            U11:
07D0  B3 02                             MOV     BL,2                    ; SET UP COUNT FOR BEEP
07D2  E8 0000 E                         CALL    BEEP                    ; SOUND THE POD BELL
07D5  EB DB                             JMP     U5                      ; TTY_RETURN
07D7                            WRITE_TTY       ENDP
                                ;--------------------------------------------------------------------
                                ; LIGHT PEN                                                          :
                                ;       THIS ROUTINE TESTS THE LIGHT PEN SWITCH AND THE LIGHT        :
                                ;       PEN TRIGGER. IF BOTH ARE SET, THE LOCATION OF THE LIGHT      :
                                ;       PEN IS DETERMINED. OTHERWISE, A RETURN WITH NO INFORMATION   :
                                ;       IS MADE.                                                     :
                                ; ON EXIT:                                                           :
                                ;       (AH) = 0 IF NO LIGHT PEN INFORMATION IS AVAILABLE            :
                                ;               BX,CX,DX ARE DESTROYED                               :
                                ;       (AH) = 1 IF LIGHT PEN IS AVAILABLE                           :
                                ;               (DH,DL) = ROW,COLUMN OF CURRENT LIGHT PEN POSITION   :
                                ;               (CH) = RASTER POSITION                               :
                                ;               (BX) = BEST GUESS AT PIXEL HORIZONTAL POSITION       :
                                ;--------------------------------------------------------------------
                                        ASSUME  CS:CODE,DS:DATA
                                ;----- SUBTRACT_TABLE
07D7                            V1      LABEL   BYTE
07D7  03 03 05 05 03 03                 DB      3,3,5,5,3,3,3,4 ;
      03 04
07DF                            READ_LPEN       PROC    NEAR
```

SECTION 5

```
                                   ;----- WAIT FOR LIGHT PEN TO BE DEPRESSED

07DF  B4 00                               MOV     AH,0                    ; SET NO LIGHT PEN RETURN CODE
07E1  8B 16 0063 R                        MOV     DX,ADDR_6845            ; GET BASE ADDRESS OF 6845
07E5  83 C2 06                            ADD     DX,6                    ; POINT TO STATUS REGISTER
07E8  EC                                  IN      AL,DX                   ; GET STATUS REGISTER
07E9  A8 04                               TEST    AL,4                    ; TEST LIGHT PEN SWITCH
07EB  74 03                               JZ      V6_A                    ; GO IF YES
07ED  E9 0872 R                           JMP     V6                      ; NOT SET, RETURN

                                   ;----- NOW TEST FOR LIGHT PEN TRIGGER

07F0  A8 02                        V6_A:  TEST    AL,2                    ; TEST LIGHT PEN TRIGGER
07F2  75 03                               JNZ     V7A                     ; RETURN WITHOUT RESETTING TRIGGER
07F4  E9 087C R                           JMP     V7

                                   ;----- TRIGGER HAS BEEN SET, READ THE VALUE IN

07F7                               V7A:
07F7  B4 10                               MOV     AH,16                   ; LIGHT PEN REGISTERS ON 6845

                                   ;----- INPUT REGS POINTED TO BY AH, AND CONVERT TO ROW COLUMN IN DX

07F9  8B 16 0063 R                        MOV     DX,ADDR_6845            ; ADDRESS REGISTER FOR 6845
07FD  8A C4                               MOV     AL,AH                   ; REGISTER TO READ
07FF  EE                                  OUT     DX,AL                   ; SET IT UP
0800  EB 00                               JMP     SHORT $+2               ; IO DELAY
0802  42                                  INC     DX                      ; DATA REGISTER
0803  EC                                  IN      AL,DX                   ; GET THE VALUE
0804  8A E8                               MOV     CH,AL                   ; SAVE IN CX
0806  4A                                  DEC     DX                      ; ADDRESS REGISTER
0807  FE C4                               INC     AH
0809  8A C4                               MOV     AL,AH                   ; SECOND DATA REGISTER
080B  EE                                  OUT     DX,AL
080C  42                                  INC     DX                      ; POINT TO DATA REGISTER
080D  EB 00                               JMP     SHORT $+2               ; IO DELAY
080F  EC                                  IN      AL,DX                   ; GET SECOND DATA VALUE
0810  8A E5                               MOV     AH,CH                   ; AX HAS INPUT VALUE

                                   ;----- AX HAS THE VALUE READ IN FROM THE 6845

0812  8A 1E 0049 R                        MOV     BL,CRT_MODE
0816  2A FF                               SUB     BH,BH                   ; MODE VALUE TO BX
0818  2E: 8A 9F 07D7 R                    MOV     BL,CS:V1[BX]            ; DETERMINE AMOUNT TO SUBTRACT
081D  2B C3                               SUB     AX,BX                   ; TAKE IT AWAY
081F  8B 1E 004E R                        MOV     BX,CRT_START
0823  D1 EB                               SHR     BX,1
0825  2B C3                               SUB     AX,BX                   ; CONVERT TO CORRECT PAGE ORIGIN
0827  79 02                               JNS     V2                      ; IF POSITIVE, DETERMINE MODE
0829  2B C0                               SUB     AX,AX                   ; <0 PLAYS AS 0

                                   ;----- DETERMINE MODE OF OPERATION

082B                               V2:                                    ; DETERMINE_MODE
082B  B1 03                               MOV     CL,3                    ; SET *8 SHIFT COUNT
082D  80 3E 0049 R 04                     CMP     CRT_MODE,4              ; DETERMINE IF GRAPHICS OR ALPHA
0832  72 2A                               JB      V4                      ; ALPHA_PEN
0834  80 3E 0049 R 07                     CMP     CRT_MODE,7
0839  74 23                               JE      V4                      ; ALPHA_PEN

                                   ;----- GRAPHICS MODE

083B  B2 28                               MOV     DL,40                   ; DIVISOR FOR GRAPHICS
083D  F6 F2                               DIV     DL                      ; DETERMINE ROW(AL) AND COLUMN(AH)
                                                                          ;  AL RANGE 0-99, AH RANGE 0-39
                                   ;----- DETERMINE GRAPHIC ROW POSITION

083F  8A E8                               MOV     CH,AL                   ; SAVE ROW VALUE IN CH
0841  02 ED                               ADD     CH,CH                   ; *2 FOR EVEN/ODD FIELD
0843  8A DC                               MOV     BL,AH                   ; COLUMN VALUE TO BX
0845  2A FF                               SUB     BH,BH                   ; MULTIPLY BY 8 FOR MEDIUM RES
0847  80 3E 0049 R 06                     CMP     CRT_MODE,6              ; DETERMINE MEDIUM OR HIGH RES
084C  75 04                               JNE     V3                      ; NOT_HIGH_RES
084E  B1 04                               MOV     CL,4                    ; SHIFT VALUE FOR HIGH RES
0850  D0 E4                               SAL     AH,1                    ; COLUMN VALUE TIMES 2 FOR HIGH RES
0852                               V3:                                    ; NOT_HIGH_RES
0852  D3 E3                               SHL     BX,CL                   ; MULTIPLY *16 FOR HIGH RES

                                   ;----- DETERMINE ALPHA CHAR POSITION

0854  8A D4                               MOV     DL,AH                   ; COLUMN VALUE FOR RETURN
0856  8A F0                               MOV     DH,AL                   ; ROW VALUE
0858  D0 EE                               SHR     DH,1                    ; DIVIDE BY 4
085A  D0 EE                               SHR     DH,1                    ;  FOR VALUE IN 0-24 RANGE
085C  EB 12                               JMP     SHORT V5                ; LIGHT_PEN_RETURN_SET

                                   ;----- ALPHA MODE ON LIGHT PEN

085E                               V4:                                    ; ALPHA_PEN
085E  F6 36 004A R                        DIV     BYTE PTR CRT_COLS       ; DETERMINE ROW,COLUMN VALUE
0862  8A F0                               MOV     DH,AL                   ; ROWS TO DH
0864  8A D4                               MOV     DL,AH                   ; COLS TO DL
0866  D2 E0                               SAL     AL,CL                   ; MULTIPLY ROWS * 8
0868  8A E8                               MOV     CH,AL                   ; GET RASTER VALUE TO RETURN REG
086A  8A DC                               MOV     BL,AH                   ; COLUMN VALUE
086C  32 FF                               XOR     BH,BH                   ;  TO BX
086E  D3 E3                               SAL     BX,CL
0870                               V5:                                    ; LIGHT_PEN_RETURN_SET
0870  B4 01                               MOV     AH,1                    ; INDICATE EVERTHING SET
0872                               V6:                                    ; LIGHT_PEN_RETURN
0872  52                                  PUSH    DX                      ; SAVE RETURN VALUE (IN CASE)
0873  8B 16 0063 R                        MOV     DX,ADDR_6845            ; GET BASE ADDRESS
0877  83 C2 07                            ADD     DX,7                    ; POINT TO RESET PARM
087A  EE                                  OUT     DX,AL                   ; ADDRESS, NOT DATA, IS IMPORTANT
087B  5A                                  POP     DX                      ; RECOVER VALUE
087C                               V7:                                    ; RETURN_NO_RESET
087C  5D                                  POP     BP
087D  5F                                  POP     DI
087E  5E                                  POP     SI
087F  1F                                  POP     DS                      ; DISCARD SAVED BX,CX,DX
0880  1F                                  POP     DS
0881  1F                                  POP     DS
0882  1F                                  POP     DS
0883  07                                  POP     ES
0884  CF                                  IRET
0885                               READ_LPEN      ENDP
0885                               CODE   ENDS
                                          END
```

```
                                        TITLE 11/22/83 BIOS
                                        .LIST
                                      C INCLUDE SEGMENT.SRC
0000                                  C CODE SEGMENT BYTE PUBLIC
                                      C
                                        EXTRN   C8042:NEAR
                                        EXTRN   OBF_42:NEAR
                                        EXTRN   DDS:NEAR
                                        EXTRN   PRT_HEX:NEAR
                                        EXTRN   D1:NEAR
                                        EXTRN   D2:NEAR
                                        EXTRN   P_MSG:NEAR
                                        EXTRN   D2A:NEAR
                                        EXTRN   PRT_SEG:NEAR
                                        EXTRN   PROC_SHUTDOWN:NEAR
                                        EXTRN   CM3:NEAR
                                        EXTRN   E_MSG:NEAR

                                        PUBLIC  MEMORY_SIZE_DETERMINE_1
                                        PUBLIC  EQUIPMENT_1
                                        PUBLIC  NMI_INT_1
                                        PUBLIC  SET_TOD

                                        ;--- INT 12 ---------------------------------------------------------:
                                        ; MEMORY_SIZE_DETERMINE                                              :
                                        ;       THIS ROUTINE RETURNSS THE AMOUNT OF MEMORY IN THE            :
                                        ;       SYSTEM AS DETERMINED BY THE POST ROUTINES.                   :
                                        ;       NOTE THAT THE SYSTEM MAY NOT BE ABLE TO USE I/O MEMORY       :
                                        ;       UNLESS THERE IS A FULL COMPLEMENT OF 512K BYTES ON THE       :
                                        ;       PLANAR.                                                      :
                                        ; INPUT                                                              :
                                        ;       NO REGISTERS                                                 :
                                        ;       THE MEMORY_SIZE VARIABLE IS SET DURING POWER ON              :
                                        ;       DIAGNOSTICS ACCORDING TO THE FOLLOWING ASSUMPTIONS:          :
                                        ;                                                                    :
                                        ;       1. CONFIGURATION RECORD IN NON-VOLATILE MEMORY               :
                                        ;          EQUALS THE ACTUAL MEMORY SIZE INSTALLED.                  :
                                        ;                                                                    :
                                        ;       2. ALL INSTALLED MEMORY IS FUNCTIONAL.  IF THE               :
                                        ;          MEMORY TEST DURING POST INDICATES LESS, THEN THIS         :
                                        ;          VALUE BECOMES THE DEFAULT.  IF NON-VOLATILE MEMORY        :
                                        ;          IS NOT VALID (NOT INITIALIZED OR BATTERY FAILURE)         :
                                        ;          THEN ACTUAL MEMORY DETERMINED BECOMES THE DEFAULT.        :
                                        ;                                                                    :
                                        ;       3. ALL MEMORY FROM 0 TO 640K MUST BE CONTIGUOUS.             :
                                        ;                                                                    :
                                        ; OUTPUT                                                             :
                                        ;       (AX) = NUMBER OF CONTIGUOUS 1K BLOCKS OF MEMORY              :
                                        ;--------------------------------------------------------------------:
                                                ASSUME  CS:CODE,DS:DATA

0000                                    MEMORY_SIZE_DETERMINE_1  PROC FAR  ;
0000  FB                                        STI                     ; INTERRUPTS BACK ON
0001  1E                                        PUSH    DS              ; SAVE SEGMENT
0002  E8 0000 E                                 CALL    DDS             ; ESTABLISH ADDRESSING
0005  A1 0013 R                                 MOV     AX,MEMORY_SIZE  ; GET VALUE
0008  1F                                        POP     DS              ; RECOVER SEGMENT
0009  CF                                        IRET                    ; RETURN TO CALLER
000A                                    MEMORY_SIZE_DETERMINE_1 ENDP

                                        ;--- INT 11 ---------------------------------------------------------:
                                        ; EQUIPMENT DETERMINATION                                            :
                                        ;       THIS ROUTINE ATTEMPTS TO DETERMINE WHAT OPTIONAL             :
                                        ;       DEVICES ARE ATTACHED TO THE SYSTEM.                          :
                                        ; INPUT                                                              :
                                        ;       NO REGISTERS                                                 :
                                        ;       THE EQUIP_FLAG VARIABLE IS SET DURING THE POWER ON           :
                                        ;       DIAGNOSTICS USING THE FOLLOWING HARDWARE ASSUMPTIONS:        :
                                        ;       PORT 3FA = INTERRUPT ID REGISTER OF 8250 (PRIMARY)           :
                                        ;            2FA = INTERRUPT ID REGISTER OF 8250 (SECONDARY)         :
                                        ;                 BITS 7-3 ARE ALWAYS 0                              :
                                        ;       PORT 378 = OUTPUT PORT OF PRINTER (PRIMARY)                  :
                                        ;            278 = OUTPUT PORT OF PRINTER (SECONDARY)                :
                                        ;            3BC = OUTPUT PORT OF PRINTER (MONO-PRINTER)             :
                                        ;  OUTPUT                                                            :
                                        ;       (AX) IS SET, BIT SIGNIFICANT, TO INDICATE ATTACHED I/O       :
                                        ;       BIT 15,14 = NUMBER OF PRINTERS ATTACHED                      :
                                        ;       BIT 13,12 NOT USED                                           :
                                        ;       BIT 11,10,9 = NUMBER OF RS232 CARDS ATTACHED                 :
                                        ;       BIT 8 = NOT USED                                             :
                                        ;       BIT 7,6 = NUMBER OF DISKETTE DRIVES                          :
                                        ;             00=1, 01=2 ONLY IF BIT 0 = 1                           :
                                        ;       BIT 5,4 = INITIAL VIDEO MODE                                 :
                                        ;                      00 - UNUSED                                   :
                                        ;                      01 - 40X25 BW USING COLOR CARD                :
                                        ;                      10 - 80X25 BW USING COLOR CARD                :
                                        ;                      11 - 80X25 BW USING BW CARD                   :
                                        ;                                                                    :
                                        ;       BIT 3 = NOT USED                                             :
                                        ;       BIT 2 = NOT USED                                             :
                                        ;       BIT 1 = MATH COPROCESSOR                                     :
                                        ;       BIT 0 = 1 (IPL DISKETTE INSTALLED)                           :
                                        ;       NO OTHER REGISTERS AFFECTED                                  :
                                        ;--------------------------------------------------------------------:
                                                ASSUME  CS:CODE,DS:DATA

000A                                    EQUIPMENT_1  PROC  FAR              ; >>>  ENTRY POINT FOR ORG OF84DH
000A  FB                                        STI                         ; INTERRUPTS BACK ON
000B  1E                                        PUSH    DS                  ; SAVE SEGMENT REGISTER
000C  E8 0000 E                                 CALL    DDS                 ; ESTABLISH ADDRESSING
000F  A1 0010 R                                 MOV     AX,EQUIP_FLAG       ; GET THE CURRENT SETTINGS
0012  1F                                        POP     DS                  ; RECOVER SEGMENT
0013  CF                                        IRET                        ; RETURN TO CALLER
0014                                    EQUIPMENT_1     ENDP
                                        ;-- INT 2 -----------------------------------------------------------:
                                        ; NON-MASKABLE INTERRUPT ROUTINE (REAL MODE)                         :
                                        ;       THIS ROUTINE WILL PRINT A "PARITY CHECK 1 OR 2" MESSAGE      :
                                        ;       AND ATTEMPT TO FIND THE STORAGE LOCATION CONTAINING THE      :
                                        ;       BAD PARITY.  IF FOUND, THE SEGMENT ADDRESS WILL BE           :
                                        ;       PRINTED.  IF NO PARITY ERROR CAN BE FOUND (INTERMITTENT      :
                                        ;       READ PROBLEM) ?????<-WILL BE PRINTED WHERE THE ADDRESS       :
                                        ;       WOULD NORMALLY GO.                                           :
                                        ;                                                                    :
                                        ;       PARITY CHECK 1 = PLANAR BOARD MEMORY FAILURE.                :
                                        ;       PARITY CHECK 2 = OFF PLANAR BOARD MEMORY FAILURE.            :
                                        ;--------------------------------------------------------------------:
0014                                    NMI_INT_1 PROC    NEAR
                                                ASSUME  DS:DATA
0014  50                                        PUSH    AX                  ; SAVE ORIG CONTENTS OF AX
0015  E4 80                                     IN      AL,MFG_PORT         ; INCREMENT NMI COUNT
0017  FE C0                                     INC     AL                  ;
0019  EB 00                                     JMP     SHORT $+2           ; IN DELAY
001B  E6 80                                     OUT     MFG_PORT,AL         ; SET COUNT

001D  E4 61                                     IN      AL,PORT_B
001F  A8 C0                                     TEST    AL,PARITY_ERR       ; PARITY CHECK?
```

SECTION 5

```
0021  8A E0                     MOV    AH,AL                 ; SAVE PARITY STATUS
0023  75 03                     JNZ    NMI_1
0025  E9 00C1 R                 JMP    D14                   ; NO, EXIT FROM ROUTINE
0028                     NMI_1:
                         ;------- GET THE SWITCH SETTINGS

0028  B0 AD                     MOV    AL,DIS_KBD            ; DISABLE THE KEYBOARD
002A  E8 0000 E                 CALL   C8042                 ;
002D  E4 60                     IN     AL,PORT_A             ; FLUSH
002F  B0 C0                     MOV    AL,READ_8042_INPUT    ; GET THE SWITCH SETTINGS
0031  E8 0000 E                 CALL   C8042                 ; ISSUE THE COMMAND
0034  E8 0000 E                 CALL   OBF_42                ; WAIT FOR OUTPUT BUFF FULL
0037  E4 60                     IN     AL,PORT_A             ; GET THE SWITCH
0039  E6 80                     OUT    MFG_PORT,AL           ; SAVE SWITCH

003B  BA ---- R                 MOV    DX,DATA
003E  8E DA                     MOV    DS,DX
0040  BE 0000 E                 MOV    SI,OFFSET D1          ; ADDR OF ERROR MSG
0043  F6 C4 40                  TEST   AH,40H                ; I/O PARITY CHECK
0046  75 03                     JNZ    NMI_2                 ; DISPLAY ERROR MSG
0048  BE 0000 E                 MOV    SI,OFFSET D2          ; MUST BE PLANAR
004B                     NMI_2:
004B  B4 00                     MOV    AH,0                  ; INIT AND SET MODE FOR VIDEO
004D  A0 0049 R                 MOV    AL,CRT_MODE
0050  CD 10                     INT    10H                   ; CALL VIDEO_IO PROCEDURE
0052  E8 0000 E                 CALL   P_MSG                 ; PRINT ERROR MSG
                         ;----- SEE IF LOCATION THAT CAUSED PARITY CHECK CAN BE FOUND
0055  B0 FF                     MOV    AL,0FFH               ; MASK TRAP
0057  E6 70                     OUT    CMOS_PORT,AL
0059  E4 61                     IN     AL,PORT_B
005B  EB 00                     JMP    SHORT $+2             ; IO DELAY
005D  0C 0C                     OR     AL,RAM_PAR_OFF        ; TOGGLE PARITY CHECK ENABLES
005F  E6 61                     OUT    PORT_B,AL
0061  EB 00                     JMP    SHORT $+2             ;IO DELAY
0063  24 F3                     AND    AL,RAM_PAR_ON
0065  E6 61                     OUT    PORT_B,AL
0067  8B 1E 0013 R              MOV    BX,MEMORY_SIZE        ; GET MEMORY SIZE WORD
006B  FC                        CLD                          ; SET DIR FLAG TO INCRIMENT
006C  2B D2                     SUB    DX,DX                 ; POINT DX AT START OF MEM
006E                     NMI_LOOP:
006E  8E DA                     MOV    DS,DX
0070  8E C2                     MOV    ES,DX
0072  B9 8000                   MOV    CX,4000H*2            ; SET FOR 64KB SCAN
0075  2B F6                     SUB    SI,SI                 ; SET SI TO BE REALTIVE TO
                                                             ; START OF ES
0077  F3/ AD                    REP    LODSW                 ; READ 64KB OF MEMORY

0079  E4 61                     IN     AL,PORT_B             ; SEE IF PARITY CHECK HAPPENED
007B  86 C4                     XCHG   AL,AH                 ; SAVE PARITY CHECK
007D  81 FA 4000                CMP    DX,4000H              ; CHECK FOR END OF OF FIRST 256K
0081  72 0C                     JB     NMI_3
0083  81 FA 8000                CMP    DX,8000H              ; CHECK ABOVE 512K
0087  73 0C                     JAE    NMI_4                 ; CHECK FOR IO CHECK
0089  E4 80                     IN     AL,MFG_PORT           ; GET THE SWITCH SETTINGS
008B  A8 10                     TEST   AL,BASE_RAM           ; CHECK FOR 2ND 256K ON PLANAR
008D  74 06                     JZ     NMI_4                 ; GO IF NOT
008F  F6 C4 80           NMI_3: TEST   AH,PRTY_CHK           ; CHECK FOR PARITY ERR
0092  EB 04 90                  JMP    NMI_5                 ; CONTINUE
0095  F6 C4 40           NMI_4: TEST   AH,IO_CHK             ; TEST FOR IO ERROR
0098  75 11              NMI_5: JNZ    PRT_NMI               ; GO PRINT ADDRESS IF IT DID
009A  81 C2 1000                ADD    DX,1000H              ; POINT TO NEXT 64K BLOCK
009E  83 EB 40                  SUB    BX,16D*4
00A1  75 CB                     JNZ    NMI_LOOP
00A3  BE 0000 E                 MOV    SI,(OFFSET D2A)       ; PRINT ROW OF ????? IF PARITY
00A6  E8 0000 E                 CALL   P_MSG                 ; CHECK COULD NOT BE RE-CREATED
00A9  FA                        CLI
00AA  F4                        HLT                          ; HALT SYSTEM
00AB                     PRT_NMI:
00AB  8C DA                     MOV    DX,DS
00AD  E8 0000 E                 CALL   PRT_SEG               ; PRINT SEGMENT VALUE
00B0  B0 28                     MOV    AL,'('                ; PRINT (S)
00B2  E8 0000 E                 CALL   PRT_HEX
00B5  B0 53                     MOV    AL,'S'
00B7  E8 0000 E                 CALL   PRT_HEX
00BA  B0 29                     MOV    AL,')'
00BC  E8 0000 E                 CALL   PRT_HEX
00BF  FA                        CLI                          ; HALT SYSTEM
00C0  F4                        HLT
00C1                     D14:
00C1  B0 8F                     MOV    AL,8FH                ; TOGGLE NMI
00C3  E6 70                     OUT    CMOS_PORT,AL          ;
00C5  EB 00                     JMP    SHORT $+2             ; IO DELAY
00C7  B0 0F                     MOV    AL,0FH                ;
00C9  E6 70                     OUT    CMOS_PORT,AL          ;
00CB  58                        POP    AX                    ; RESTORE ORIG CONTENTS OF AX
00CC  CF                        IRET
00CD                     NMI_INT_1 ENDP
                         PAGE
                         ;--------------------------------------------------------
                         ;                                                       :
                         ;       THIS ROUTINE INITIALIZES THE TIMER DATA AREA IN THE  :
                         ;       ROM BIOS DATA AREA.  IT IS CALLED BY THE POWER ON    :
                         ;       ROUTINES.  IT CONVERTS HR:MIN:SEC FROM CMOS TO TIMER :
                         ;       TICS.  IF CMOS IS INVALID, TIMER DATA IS SET TO ZERO.:
                         ;--------------------------------------------------------
                         ;
                         ; INPUT    NONE PASSED TO ROUTINE BY CALLER
                         ;
                         ; CMOS BYTES USED FOR SETUP
                         ;
                         ;       00        SECONDS
                         ;       02        MINUTES
                         ;       04        HOURS
                         ;       0A        REGISTER A (UPDATE IN PROGRESS)
                         ;       0E        CMOS VALID IF ZERO
                         ;
                         ; OUTPUT
                         ;       TIMER_LOW
                         ;       TIMER_HIGH
                         ;       TIMER_OFL
                         ;       ALL REGISTERS UNCHANGED
                         ;
                         ;----------------------------------------------------------------
= 0012                   COUNTS_SEC     EQU     18
= 0444                   COUNTS_MIN     EQU     1092
= 0007                   COUNTS_HOUR    EQU     7            ; 65543 - 65536
= 0070                   CMOS_ADR       EQU     70H
= 0071                   CMOS_DATA      EQU     71H
= 000E                   CMOS_VALID     EQU     0EH
= 0000                   CMOS_SECONDS   EQU     00H
= 0002                   CMOS_MINUTES   EQU     02H
= 0004                   CMOS_HOURS     EQU     04H
= 000A                   CMOS_REGA      EQU     0AH
= 0080                   UPDATE_TIMER   EQU     80H
00CD                     SET_TOD PROC   NEAR
                                PUSHA
```

```
00CD  60                    +           DB      060H
00CE  1E                                PUSH    DS
                                        ASSUME  DS:DATA
00CF  B8  ---- R                        MOV     AX,DATA                 ; ESTABLISH SEGMENT
00D2  8E D8                             MOV     DS,AX
00D4  2B C0                             SUB     AX,AX
00D6  A2 0070 R                         MOV     TIMER_OFL,AL            ; RESET TIMER ROLL OVER INDICATOR
00D9  A3 006C R                         MOV     TIMER_LOW,AX            ; AND TIMER COUNT
00DC  A3 006E R                         MOV     TIMER_HIGH,AX
00DF  B0 0E                             MOV     AL,CMOS_VALID
00E1  E6 70                             OUT     CMOS_ADR,AL             ; CHECK CMOS VALIDITY
00E3  EB 00                             JMP     SHORT $+2
00E5  E4 71                             IN      AL,CMOS_DATA
00E7  24 C4                             AND     AL,0C4H                 ; BAD BATTERY, CHKSUM ERROR OR CLOCK ERROR
00E9  75 61                             JNZ     POD_DONE                ; CMOS NOT VALID -- TIMER SET TO ZERO
00EB  2B C9                             SUB     CX,CX
00ED  B0 0A                 UIP:        MOV     AL,CMOS_REGA
00EF  E6 70                             OUT     CMOS_ADR,AL             ; ACCESS REGISTER A
00F1  EB 00                             JMP     SHORT $+2
00F3  E4 71                             IN      AL,CMOS_DATA
00F5  A8 80                             TEST    AL,UPDATE_TIMER
00F7  74 05                             JZ      READ_SEC
00F9  E2 F2                             LOOP    UIP
00FB  EB 4F 90                          JMP     POD_DONE                ; CMOS CLOCK STUCK
00FE                        READ_SEC:
00FE  B0 00                             MOV     AL,CMOS_SECONDS
0100  E6 70                             OUT     CMOS_ADR,AL             ; ACCESS SECONDS VALUE IN CMOS
0102  EB 00                             JMP     SHORT $+2
0104  E4 71                             IN      AL,CMOS_DATA

0106  3C 59                             CMP     AL,59H                  ; ARE THE SECONDS WITHIN LIMITS?
0108  77 4D                             JA      TOD_ERROR               ; GO IF NOT

010A  E8 0176 R                         CALL    CVT_BINARY              ; CONVERT IT TO BINARY
010D  B3 12                             MOV     BL,COUNTS_SEC
010F  F6 E3                             MUL     BL                      ; COUNT FOR SECONDS
0111  8B C8                             MOV     CX,AX
0113  B0 02                             MOV     AL,CMOS_MINUTES
0115  E6 70                             OUT     CMOS_ADR,AL             ; ACCESS MINUTES VALUE IN CMOS
0117  EB 00                             JMP     SHORT $+2
0119  E4 71                             IN      AL,CMOS_DATA

011B  3C 59                             CMP     AL,59H                  ; ARE THE MINUTES WITHIN LIMITS?
011D  77 38                             JA      TOD_ERROR               ; GO IF NOT

011F  E8 0176 R                         CALL    CVT_BINARY              ; CONVERT IT TO BINARY
0122  BB 0444                           MOV     BX,COUNTS_MIN
0125  F7 E3                             MUL     BX                      ; COUNT FOR MINUTES
0127  03 C1                             ADD     AX,CX
0129  8B C8                             MOV     CX,AX
012B  B0 04                             MOV     AL,CMOS_HOURS
012D  E6 70                             OUT     CMOS_ADR,AL             ; ACCESS HOURS VALUE IN CMOS
012F  EB 00                             JMP     SHORT $+2
0131  E4 71                             IN      AL,CMOS_DATA
0133  3C 23                             CMP     AL,23H                  ; ARE THE HOURS WITHIN LIMITS?
0135  77 20                             JA      TOD_ERROR               ; GO IF NOT
0137  E8 0176 R                         CALL    CVT_BINARY              ; CONVERT IT TO BINARY
013A  8B D0                             MOV     DX,AX
013C  B3 07                             MOV     BL,COUNTS_HOUR
013E  F6 E3                             MUL     BL                      ; COUNT FOR HOURS
0140  03 C1                             ADD     AX,CX
0142  83 D2 00                          ADC     DX,0000H
0145  89 16 006E R                      MOV     TIMER_HIGH,DX
0149  A3 006C R                         MOV     TIMER_LOW,AX
014C                        POD_DONE:
014C  FA                                CLI                             ; ** IO DELAY NOT REQUIRED **
014D  E4 21                             IN      AL,021H                 ; BE SURE TIMER IS ENABLED
014F  24 FE                             AND     AL,0FEH
0151  E6 21                             OUT     021H,AL
0153  FB                                STI
0154  1F                                POP     DS
                                        POPA
0155  61                    +           DB      061H
0156  C3                                RET
0157                        TOD_ERROR:
0157  1F                                POP     DS                      ; RESTORE SEGMENT
                                        POPA                            ; RESTORE REGS
0158  61                    +           DB      061H
0159  BE 0000 E                         MOV     SI,OFFSET CM3           ; DISPLAY CLOCK ERROR
015C  E8 0000 E                         CALL    E_MSG                   ;
015F  B0 8E                             MOV     AL,DIAG_STATUS          ; SET CLOCK ERROR
0161  E6 70                             OUT     CMOS_PORT,AL            ;
0163  86 C4                             XCHG    AL,AH                   ; SAVE STATUS ADDRESS
0165  EB 00                             JMP     SHORT $+2               ; IO DELAY
0167  E4 71                             IN      AL,CMOS_PORT+1          ; GET THE CURRENT STATUS
0169  0C 04                             OR      AL,CMOS_CLK_FAIL        ; SET NEW STATUS
016B  86 C4                             XCHG    AL,AH                   ; GET STATUS ADDR AND SAVE NEW STATUS
016D  E6 70                             OUT     CMOS_PORT,AL            ;
016F  86 C4                             XCHG    AL,AH                   ;
0171  EB 00                             JMP     SHORT $+2               ; IO DELAY
0173  E6 71                             OUT     CMOS_PORT+1,AL          ;
0175  C3                                RET

0176                        SET_TOD ENDP

0176                        CVT_BINARY          PROC    NEAR
0176  8A E0                             MOV     AH,AL                   ; UNPACK 2 BCD DIGITS IN AL
                                        ISHR    AH,4
0178                        + ??0000    LABEL   BYTE
0178  D0 EC                 +           SHR     AH,1
017A                        + ??0001    LABEL   BYTE
0178                        +           ORG     OFFSET CS:??0000
0178  C0                    +            DB      0C0H
017A                        +           ORG     OFFSET CS:??0001
017A  04                    +           DB      4
017B  24 0F                             AND     AL,0FH                  ; RESULT IS IN AX
017D  D5 0A                             AAD                             ; CONVERT UNPACKED BCD TO BINARY
017F  C3                                RET
0180                        CVT_BINARY          ENDP
0180                        CODE        ENDS
                                        END
```

SECTION 5

```
                              TITLE 11/22/83 BIOS1
                              .LIST
                            C INCLUDE SEGMENT.SRC
0000                        C CODE SEGMENT BYTE PUBLIC
                            C
                              EXTRN   DDS:NEAR
                              EXTRN   PRT_HEX:NEAR
                              EXTRN   D1:NEAR
                              EXTRN   D2:NEAR
                              EXTRN   P_MSG:NEAR
                              EXTRN   D2A:NEAR
                              EXTRN   PRT_SEG:NEAR
                              EXTRN   PROC_SHUTDOWN:NEAR

                              PUBLIC  SHUT9
                              PUBLIC  GATE_A20
                              PUBLIC CASSETTE_IO_1

                              ;--- INT 15 ----------------------------------------------------------
                              ;     INPUT - CASSETTE I/O FUNCTIONS                                   :
                              ;            (AH) = 00                                                 :
                              ;            (AH) = 01                                                 :
                              ;            (AH) = 02                                                 :
                              ;            (AH) = 03                                                 :
                              ;         RETURNS FOR THESE FUNCTIONS ALWAYS (AH) = 86H, CF = 1)       :
                              ;         IF CASSETTE PORT NOT PRESENT                                 :
                              ;----------------------------------------------------------------------
                              ;     INPUT - UNUSED FUNCTIONS                                         :
                              ;             (AH) = 04 THROUGH 7F                                     :
                              ;         RETURNS FOR THESE FUNCTIONS ALWAYS (AH) = 86H, CF = 1)       :
                              ;----------------------------------------------------------------------
                              ; Extensions                                                           :
                              ;            (AH) = 80H     DEVICE OPEN                                :
                              ;                 (BX) = DEVICE ID                                     :
                              ;                 (CX) = PROCESS ID                                    :
                              ;                                                                      :
                              ;            (AH) = 81H     DEVICE CLOSE                               :
                              ;                 (BX) = DEVICE ID                                     :
                              ;                 (CX) = PROCESS ID                                    :
                              ;                                                                      :
                              ;            (AH) = 82H     PROGRAM TERMINATION                        :
                              ;                 (BX) = DEVICE ID                                     :
                              ;                                                                      :
                              ;            (AH) = 83H     EVENT WAIT                                 :
                              ;                 (AL) = 0 SET INTERVAL                                :
                              ;                (ES:BX) POINTER TO A BYTE IN CALLERS MEMORY           :
                              ;                        THAT WILL HAVE THE HIGH ORDER BIT SET         :
                              ;                        AS SOON AS POSSIBLE AFTER THE INTERVAL        :
                              ;                        EXPIRES.                                      :
                              ;                (CX,DX) NUMBER OF MICROSECONDS TO ELAPSE BEFORE       :
                              ;                        POSTING.                                      :
                              ;                 (AL) = 1 CANCEL                                      :
                              ;                                                                      :
                              ;            (AH) = 84H     JOYSTICK SUPPORT                           :
                              ;                 (DX) = 0 - READ THE CURRENT SWITCH SETTINGS          :
                              ;                        RETURNS AL = SWITCH SETTINGS (BITS 7-4)       :
                              ;                 (DX) = 1 - READ THE RESISTIVE INPUTS                 :
                              ;                        RETURNS AX = A(x) VALUE                       :
                              ;                                    BX = A(y) VALUE                   :
                              ;                                    CX = B(x) VALUE                   :
                              ;                                    DX = B(y) VALUE                   :
                              ;                                                                      :
                              ;            (AH) = 85H     SYSTEM REQUEST KEY PRESSED                 :
                              ;                  (AL) = 00 MAKE OF KEY                               :
                              ;                  (AL) = 01 BREAK OF KEY                              :
                              ;            (AH) = 86H     WAIT                                       :
                              ;                (CX,DX) NUMBER OF MICROSECONDS TO ELAPSE BEFORE       :
                              ;                        RETURN TO CALLER                              :
                              ;            (AH) = 87H     MOVE BLOCK                                 :
                              ;                  (CX)     NUMBER OF WORDS TO MOVE                    :
                              ;                (ES:SI)    POINTER TO DESCRIPTOR TABLE                :
                              ;            (AH) = 88H     EXTENDED MEMORY SIZE DETERMINE             :
                              ;            (AH) = 89H     PROCESSOR TO VIRTUAL MODE                  :
                              ;                                                                      :
                              ;            (AH) = 90H     DEVICE BUSY LOOP                           :
                              ;                  (AL)     SEE TYPE CODE                              :
                              ;                                                                      :
                              ;            (AH) = 91H     INTERRUPT COMPLETE FLAG SET                :
                              ;                  (AL)     TYPE CODE                                  :
                              ;                           00H -> 7FH                                 :
                              ;                                   SERIALLY REUSABELE DEVICES;        :
                              ;                                   OPERATING SYSTEM MUST SERIALIZE    :
                              ;                                   ACCESS                             :
                              ;                           80H -> BFH                                 :
                              ;                                   REENTRANT DEVICES; ES:BX IS        :
                              ;                                   USED TO DISTINGUISH DIFFERENT      :
                              ;                                   CALLS (MULTIPLE I/O CALLS ARE      :
                              ;                                   ALLOWED SIMULTANEUSLY)             :
                              ;                           C0H -> FFH                                 :
                              ;                                   WAIT ONLY CALLS;  THERE IS NO      :
                              ;                                   COMPLEMENTARY 'POST' FOR THESE     :
                              ;                                   WAITS - - THESE ARE TIMEOUT        :
                              ;                                   ONLY.  TIMES ARE FUNCTION NUMBER:
                              ;                                   DEPENDENT                          :
                              ;                                                                      :
                              ;                           TYPE  DESCRIPTION             TIMEOUT      :
                              ;                                                                      :
                              ;                           00H = DISK                    YES          :
                              ;                           01H = DISKETTE                YES          :
                              ;                           02H = KEYBOARD                NO           :
                              ;                           80H = NETWORK                 NO           :
                              ;                                 ES:BX --> NCB                        :
                              ;                           FDH = DISKETTE MOTOR START    YES          :
                              ;                           FEH = PRINTER                 YES          :
                              ;----------------------------------------------------------------------
                                      ASSUME  CS:CODE
0000                          CASSETTE_IO_1   PROC    FAR
0000  FB                              STI
0001  80 FC 80                        CMP     AH,80H          ; CHECK FOR RANGE
0004  72 46                           JB      C1              ; RETURN IF 00-7FH
0006  80 EC 80                        SUB     AH,80H          ; BASE ON 0
0009  0A E4                           OR      AH,AH
000B  74 45                           JZ      DEV_OPEN        ; DEVICE OPEN
000D  FE CC                           DEC     AH
000F  74 41                           JZ      DEV_CLOSE       ; DEVICE CLOSE
0011  FE CC                           DEC     AH
0013  74 3D                           JZ      PROG_TERM       ; PROGRAM TERMINATION
0015  FE CC                           DEC     AH
0017  74 3B                           JZ      EVENT_WAIT      ; EVEMT WAIT
0019  FE CC                           DEC     AH
001B  74 78                           JZ      JOY_STICK       ; JOYSTICK BIOS
001D  FE CC                           DEC     AH              ;
001F  74 31                           JZ      SYS_REQ         ; SYSTEM REQUEST KEY
0021  FE CC                           DEC     AH              ;
0023  74 07                           JZ      C1_A            ; WAIT
0025  FE CC                           DEC     AH              ;
```

SECTION 5

```
0027  75 06                          JNZ     C1_B             ;
0029  E9 0183 R                      JMP     BLOCKMOVE        ; MOVE BLOCK

002C  E9 0132 R              C1_A:   JMP     WAIT             ; WAIT

002F  FE CC                  C1_B:   DEC     AH               ;

0031  75 03                          JNZ     C1_C             ;
0033  E9 03D2 R                      JMP     EXT_MEMORY       ; GO GET THE EXTENDED MEMORY

0036  FE CC                  C1_C:   DEC     AH               ;
0038  75 03                          JNZ     C1_D             ; CHECK FOR FUNTION 89
003A  E9 03E6 R                      JMP     SET_VMODE        ; SWAP TO VIRTUAL MODE

003D  80 EC 07               C1_D:   SUB     AH,7             ; CHECK FOR FUNCTION 90
0040  75 03                          JNZ     C1_E             ; GO IF NOT
0042  E9 0475 R                      JMP     DEVICE_BUSY      ;

0045  FE CC                  C1_E:   DEC     AH               ; CHECK FOR FUNCTION 8B
0047  75 03                          JNZ     C1               ; GO IF NOT
0049  E9 0479 R                      JMP     INT_COMPLETE     ;

004C  B4 86                  C1:     MOV     AH,86H           ; SET BAD COMMAND
004E  F9                             STC                      ; SET CARRY FLAG ON
004F                         C1_F:
004F  CA 0002                        RET     2                ;


0052                         DEV_OPEN:

0052                         DEV_CLOSE:

0052                         PROG_TERM:

0052                         SYS_REQ:
0052  EB FB                          JMP     C1_F             ;RETURN
0054                         CASSETTE_IO_1   ENDP

0054                         EVENT_WAIT      PROC    NEAR
                                     ASSUME  CS:CODE,DS:DATA
0054  1E                             PUSH    DS                       ; SAVE
0055  E8 0000 E                      CALL    DDS                      ;
0058  F6 06 00A0 R 01                TEST    RTC_WAIT_FLAG,01         ; CHECK FOR FUNCTION ACTIVE
005D  74 04                          JZ      EVENT_WAIT_1             ;
005F  1F                             POP     DS
0060  F9                             STC                              ; SET ERROR
0061  EB EC                          JMP     C1_F                     ; RETURN
0063                         EVENT_WAIT_1:
0063  FA                             CLI                              ; NO INTERRUPTS ALLOWED
0064  E4 A1                          IN      AL,0A1H                  ;ENSURE INTERRUPT UNMASKED
0066  24 FE                          AND     AL,0FEH                  ;
0068  E6 A1                          OUT     0A1H,AL                  ;
006A  8C 06 009A R                   MOV     USER_FLAG_SEG,ES         ; SET UP TRANSFER TABLE
006E  89 1E 0098 R                   MOV     USER_FLAG,BX             ;
0072  89 0E 009E R                   MOV     RTC_HIGH,CX              ;
0076  89 16 009C R                   MOV     RTC_LOW,DX               ;
007A  C6 06 00A0 R 01                MOV     RTC_WAIT_FLAG,01         ; SET ON FUNCTION ACTIVE SWITCH
007F  B0 0B                          MOV     AL,0BH                   ; ENABLE PIE
0081  E6 70                          OUT     CMOS_PORT,AL             ;
0083  E4 71                          IN      AL,CMOS_PORT+1           ;
0085  24 7F                          AND     AL,07FH                  ;
0087  0C 40                          OR      AL,040H                  ;
0089  50                             PUSH    AX                       ;
008A  B0 0B                          MOV     AL,0BH                   ;
008C  E6 70                          OUT     CMOS_PORT,AL             ;
008E  58                             POP     AX                       ;
008F  E6 71                          OUT     CMOS_PORT+1,AL           ;
0091  FB                             STI                              ; ENABLE INTERRUPTS
0092  1F                             POP     DS
0093  EB BA                          JMP     C1_F
0095                         EVENT_WAIT      ENDP
                             ;--- JOY_STICK --------------------------------------------------:
                             ;       THIS ROUTINE WILL READ THE JOYSTICK PORT               :
                             ;                                                              :
                             ;       INPUT                                                  :
                             ;       (DX)=0 READ THE CURRENT SWITCHES                       :
                             ;              RETURNS (AL)= SWITCH SETTINGS IN BITS 7-4       :
                             ;                                                              :
                             ;       (DX)=1  READ THE RESISTIVE INPUTS                      :
                             ;               RETURNS (AX)=A(x) VALUE                        :
                             ;                       (BX)=A(y) VALUE                        :
                             ;                       (CX)=B(x) VALUE                        :
                             ;                       (DX)=B(y) VALUE                        :
                             ;                                                              :
                             ;       CY FLAG ON IF NO ADAPTER CARD OR INVALID CALL          :
                             ;--------------------------------------------------------------:

                                     ASSUME CS:CODE
0095                         JOY_STICK       PROC    NEAR
0095  FB                             STI                      ; INTERRUPTS BACK ON
0096  8B C2                          MOV     AX,DX            ; GET SUBFUNCTION CODE
0098  BA 0201                        MOV     DX,201H          ; ADDRESS OF PORT
009B  0A C0                          OR      AL,AL            ;
009D  74 09                          JZ      JOY_2            ; READ SWITCHES
009F  FE C8                          DEC     AL               ;
00A1  74 0A                          JZ      JOY_3            ; READ RESISTIVE INPUTS
00A3  EB A7                          JMP     C1               ; GO TO ERROR RETURN
00A5                         JOY_1:
00A5  FB                             STI                      ;
00A6  EB A7                          JMP     C1_F             ; GO TO COMMON RETURN
                             ;
00A8                         JOY_2:
00A8  EC                             IN      AL,DX            ;
00A9  24 F0                          AND     AL,0F0H          ; STRIP UNWANTED BITS OFF
00AB  EB F8                          JMP     JOY_1            ; FINISHED

00AD                         JOY_3:
00AD  B3 01                          MOV     BL,1             ;
00AF  E8 00CB R                      CALL    TEST_CORD        ;
00B2  51                             PUSH    CX               ; SAVE A(x) VALUE
00B3  B3 02                          MOV     BL,2             ;
00B5  E8 00CB R                      CALL    TEST_CORD        ;
00B8  51                             PUSH    CX               ; SAVE A(y) VALUE
00B9  B3 04                          MOV     BL,4             ;
00BB  E8 00CB R                      CALL    TEST_CORD        ;
00BE  51                             PUSH    CX               ; SAVE B(x) VALUE
00BF  B3 08                          MOV     BL,8             ;
00C1  E8 00CB R                      CALL    TEST_CORD        ;
00C4  8B D1                          MOV     DX,CX            ; SAVE B(y) VALUE
00C6  59                             POP     CX               ; GET B(x) VALUE
00C7  5B                             POP     BX               ; GET A(y) VALUE
00C8  58                             POP     AX               ; GET A(x) VALUE
00C9  EB DA                          JMP     JOY_1            ; FINISHED - RETURN

00CB                         TEST_CORD       PROC   NEAR
00CB  52                             PUSH    DX               ; SAVE
```

```
00CC  FA                         CLI                          ; BLOCK INTERRUPTS WHILE READING
00CD  B0 00                      MOV     AL,0                 ; SET UP TO LATCH TIMER 0
00CF  E6 43                      OUT     TIMER+3,AL           ;
00D1  EB 00                      JMP     SHORT $+2
00D3  E4 40                      IN      AL,TIMER             ; READ LOW BYTE OF TIMER 0
00D5  EB 00                      JMP     SHORT $+2
00D7  8A E0                      MOV     AH,AL                ;
00D9  E4 40                      IN      AL,TIMER             ; READ HIGH BYTE OF TIMER 0
00DB  86 E0                      XCHG    AH,AL                ; REARRANGE TO HIGH,LOW
00DD  50                         PUSH    AX                   ; SAVE
00DE  B9 04FF                    MOV     CX,4FFH              ; SET COUNT
00E1  EE                         OUT     DX,AL                ; FIRE TIMER
00E2  EB 00                      JMP     SHORT $+2
00E4                      TEST_CORD_1:
00E4  EC                         IN      AL,DX                ; READ VALUES
00E5  84 C3                      TEST    AL,BL                ; HAS PULSE ENDED?
00E7  E0 FB                      LOOPNZ  TEST_CORD_1          ;
00E9  83 F9 00                   CMP     CX,0                 ;
00EC  59                         POP     CX                   ; ORIGINAL COUNT
00ED  75 04                      JNZ     SHORT TEST_CORD_2    ;
00EF  2B C9                      SUB     CX,CX                ; SET 0 COUNT FOR RETURN
00F1  EB 2D                      JMP     SHORT TEST_CORD_3       ; EXIT WITH COUNT = 0
00F3                      TEST_CORD_2:
00F3  B0 00                      MOV     AL,0                 ; SET UP TO LATCH TIMER 0
00F5  E6 43                      OUT     TIMER+3,AL           ;
00F7  EB 00                      JMP     SHORT $+2
00F9  E4 40                      IN      AL,TIMER             ; READ LOW BYTE OF TIMER 0
00FB  8A E0                      MOV     AH,AL                ;
00FD  EB 00                      JMP     SHORT $+2
00FF  E4 40                      IN      AL,TIMER             ; READ HIGH BYTE OF TIMER 0
0101  86 E0                      XCHG    AH,AL                ; REARRANGE TO HIGH,LOW

0103  3B C8                      CMP     CX,AX                ; CHECK FOR COUNTER WRAP
0105  73 0B                      JAE     TEST_CORD_4          ; GO IF NO
0107  52                         PUSH    DX                   ;
0108  BA FFFF                    MOV     DX,-1                ;

010B  2B D0                      SUB     DX,AX                ; ADJUST FOR WRAP
010D  03 CA                      ADD     CX,DX                ;
010F  5A                         POP     DX                   ;
0110  EB 02                      JMP     SHORT TEST_CORD_5

0112                      TEST_CORD_4:
0112  2B C8                      SUB     CX,AX                ;
0114                      TEST_CORD_5:
0114  81 E1 1FF0                 AND     CX,1FF0H             ; ADJUST
0118  D1 E9                      SHR     CX,1                 ;
011A  D1 E9                      SHR     CX,1                 ;
011C  D1 E9                      SHR     CX,1
011E  D1 E9                      SHR     CX,1

0120                      TEST_CORD_3:
0120  FB                         STI                          ; INTERRUPTS BACK ON
0121  BA 0201                    MOV     DX,201H              ; FLUSH OTHER INPUTS
0124  51                         PUSH    CX                   ;
0125  50                         PUSH    AX
0126  B9 04FF                    MOV     CX,4FFH              ; COUNT
0129                      TEST_CORD_6:
0129  EC                         IN      AL,DX                ;
012A  A8 0F                      TEST    AL,0FH               ;
012C  E0 FB                      LOOPNZ  TEST_CORD_6          ;

012E  58                         POP     AX                   ;
012F  59                         POP     CX                   ;
0130  5A                         POP     DX                   ; SET COUNT

0131  C3                         RET                          ; RETURN
0132                      TEST_CORD      ENDP
0132                      JOY_STICK      ENDP

0132                      WAIT    PROC    NEAR
0132  1E                         PUSH    DS                           ; SAVE
0133  E8 0000 E                  CALL    DDS                          ;
0136  F6 06 00A0 R 01            TEST    RTC_WAIT_FLAG,01             ; TEST FOR FUNCTION ACTIVE
013B  74 05                      JZ      WAIT_1
013D  1F                         POP     DS
013E  F9                         STC                                  ; SET ERROR
013F  E9 004F R                  JMP     C1_F                         ; RETURN
0142                      WAIT_1:
0142  FA                         CLI                                  ; NO INTERRUPTS ALLOWED
0143  E4 A1                      IN      AL,0A1H                      ; ENSURE INTERRUPT UNMASKED
0145  24 FE                      AND     AL,0FEH                      ;
0147  E6 A1                      OUT     0A1H,AL                      ;
0149  8C 1E 009A R               MOV     USER_FLAG_SEG,DS             ; SET UP TRANSFER TABLE
014D  C7 06 0098 R 00A0 R        MOV     USER_FLAG,OFFSET RTC_WAIT_FLAG
0153  89 0E 009E R               MOV     RTC_HIGH,CX                  ;
0157  89 16 009C R               MOV     RTC_LOW,DX                   ;
015B  C6 06 00A0 R 01            MOV     RTC_WAIT_FLAG,01             ; SET ON FUNCTION ACTIVE SWITCH
0160  B0 0B                      MOV     AL,0BH                       ; ENABLE PIE
0162  E6 70                      OUT     CMOS_PORT,AL                 ;
0164  E4 71                      IN      AL,CMOS_PORT+1               ;
0166  24 7F                      AND     AL,07FH                      ;
0168  0C 40                      OR      AL,040H                      ;
016A  50                         PUSH    AX                           ;
016B  B0 0B                      MOV     AL,0BH                       ;
016D  E6 70                      OUT     CMOS_PORT,AL                 ;
016F  58                         POP     AX                           ;
0170  E6 71                      OUT     CMOS_PORT+1,AL               ;
0172  FB                         STI                                  ; ENABLE INTERRUPTS
0173                      WAIT_2:
0173  F6 06 00A0 R 80            TEST    RTC_WAIT_FLAG,080H           ; CHECK FOR END OF WAIT
0178  74 F9                      JZ      WAIT_2                       ;
017A  C6 06 00A0 R 00            MOV     RTC_WAIT_FLAG,0              ; SET FUNCTION INACTIVE
017F  1F                         POP     DS                           ;
0180  E9 004F R                  JMP     C1_F

0183                      WAIT    ENDP
                          PAGE
                          ;------ INT 15 (FUNCTION 87H - MOVE BLOCK) ---------------------
                          ; PURPOSE:                                                      :
                          ;        THIS BIOS FUNCTION PROVIDES A MEANS TO TRANSFER A BLOCK :
                          ;        OF STORAGE TO AND FROM STORAGE ABOVE THE 1 MEG ADDRESS  :
                          ;        RANGE IN VIRTUAL (PROTECTED) MODE.                      :
                          ;                                                                :
                          ; ENTRY REQUIREMENTS:                                            :
                          ;                                                                :
                          ;        ES:SI POINTS TO A DESCRIPTOR TABLE (GDT) BUILT BEFORE   :
                          ;        INTERRUPTING TO THIS FUNCTION.  THESE DESCRIPTORS ARE   :
                          ;        ARE USED BY THIS FUNCTION TO PERFORM THE BLOCK MOVE.    :
                          ;        THE SOURCE AND TARGET DESCRIPTORS BUILT BY THE USER     :
                          ;        MUST HAVE THE SEGMENT LENGTH = 2 * CX - 1 OR GREATER.   :
                          ;        THE DATA ACCESS RIGHTS BYTE WILL BE SET TO CPL0-R/W(93H):
                          ;        THE 24 BIT ADDRESS (BYTE HI, WORD LOW) WILL BE SET      :
                          ;        TO THE TARGET/SOURCE.                                   :
                          ;                                                                :
                          ; THE DESCRIPTORS ARE DEFINED AS FOLLOWS:                        :
```

```
                                  ;                                                                   :
                                  ;       1.  THE FIRST DESCRIPTOR IS THE REQUIRED DUMMY.             :
                                  ;           (USER INITIALIZED TO 0)                                 :
                                  ;       2.  THE SECOND DESCRIPTOR POINTS TO THE GDT TABLE AS        :
                                  ;           A DATA SEGMENT.                                         :
                                  ;           (USER INITIALIZED TO 0)                                 :
                                  ;       3.  THE THIRD  DESCRIPTOR IS THE DESCRIPTOR THAT POINTS     :
                                  ;           TO THE SOURCE TO BE MOVED. (FROM)                       :
                                  ;           (USER INITIALIZED)                                      :
                                  ;       4.  THE FOURTH DESCRIPTOR IS THE DESCRIPTOR THAT POINTS     :
                                  ;           TO THE DESTINATION. (TO)                                :
                                  ;           (USER INITIALIZED)                                      :
                                  ;       5.  THE FIFTH IS A DESCRIPTOR THAT THIS FUNCTION USES       :
                                  ;           TO CREATE A VIRTUAL CODE SEGMENT                        :
                                  ;           (USER INITIALIZED TO 0)                                 :
                                  ;       6.  THE SIXTH IS A DESCRIPTOR THAT THIS FUNCTION USES       :
                                  ;           TO CREATE A VIRTUAL STACK SEGMENT.  (POINTS TO USERS:
                                  ;           STACK)                                                  :
                                  ;           (USER INITIALIZED TO 0)                                 :
                                  ;-------------------------------------------------------------------:
                                  PAGE
                                  ;------ INT 15 (FUNCTION 87H CONTINUED) ----------------------------:
                                  ;                                                                   :
                                  ;       AH=87   (FUNCTION CALL)                                     :
                                  ;       ES:SI = LOCATION OF THE GDT TABLE BUILD BY ROUTINE          :
                                  ;       USING THIS FUNCTION.                                        :
                                  ;       CX = WORD COUNT OF STORAGE BLOCK TO BE MOVE.                :
                                  ;                                                                   :
                                  ;            NOTE: MAX COUNT = 8000H  32K WORDS                     :
                                  ;                                                                   :
                                  ; EXIT PARAMETERS:                                                  :
                                  ;                                                                   :
                                  ;       AH = 0  IF SUCCESSFUL                                       :
                                  ;       AH = 1  IF RAM PARITY (PARITY ERROR IS CLEARED)             :
                                  ;       AH = 2  IF EXCEPTION INTERRUPT ERROR                        :
                                  ;       AH = 3  IF GATE ADDRESS LINE 20 FAILED                      :
                                  ;       ALL REGISTER ARE RESTORED EXCEPT AX.                        :
                                  ;       CARRY FLAG = 1 IF ERROR                                     :
                                  ;       ZERO FLAG  = 1 IF SUCCESSFUL                                :
                                  ; CONSIDERATIONS:                                                   :
                                  ;                                                                   :
                                  ;       NO INTERRUPTS ARE ALLOWED.                                  :
                                  ;           TIME OF DAY (ADJUSTED BY USER???)                       :
                                  ;                                                                   :
                                  ; DESCRIPTION:                                                      :
                                  ;                                                                   :
                                  ;       1.  CLI (NO INTERRUPTS ALLOWED) WHILE THIS FUNCTION IS      :
                                  ;           EXECUTING.                                              :
                                  ;       2.  ADDRESS LINE 20 IS GATED ACTIVE.                        :
                                  ;       3.  THE IDT (INTERRUPT DESCRIPTOR TABLE) IS ROM RESIDENT:
                                  ;       4.  THE CURRENT USER STACK SEGMENT AND OFFSET IS SAVED.     :
                                  ;       5.  THE GDTR IS LOADED WITH THE OFFSET INTO ES:SI           :
                                  ;       6.  THE IDTR SELECTOR IS ROM RESIDENT AND IS LOADED.        :
                                  ;       7.  THE PROCESSOR IS PUT IN VIRTUAL MODE                    :
                                  ;       8.  DATA SEGMENT IS LOADED WITH THE SOURCE DESCRIPTOR       :
                                  ;       9.  EXTRA SEGMENT IS LOADED WITH THE TARGET DESCRIPTOR      :
                                  ;      10.  DS:SI (SOURCE) ES:DI (TARGET) REP MOVSW IS EXECUTED     :
                                  ;      11.  SHUTDOWN 09 IS EXECUTED.                                :
                                  ;      12.  STACK SEGMENT/OFFSET IS RESTORED.                       :
                                  ;      13.  ADDRESS LINE 20 IS DEGATED.                             :
                                  ;      14.  INTERRUPTS ARE ALLOWED                                  :
                                  ;-------------------------------------------------------------------
                                  page
                                  ;       THE FOLLOWING DIAGRAM DEPICTS THE ORGANIZATION
                                  ;       OF GDT.
                                  ;-------------------------------------------------------------------:
                                  ;                             G D T                                 :
                                  ;                                                                   :
                                  ;                            .-------------.                        :
                                  ;                            v             |                        :
                                  ;     (ES:SI)-->>  +00 .----------------.  |                        :
                                  ;                      |     DUMMY      |  |                        :
                                  ;                      |                |  |                        :
                                  ;                  +08 |----------------|  |                        :
                                  ;                      |    GDT LOC     | ---'                      :
                                  ;                      |                |                           :
                                  ;                  +10 |----------------|                           :
                                  ;                      |     SOURCE     |                           :
                                  ;                      |      GDT       |                           :
                                  ;                  +18 |----------------|                           :
                                  ;                      |     TARGET     |                           :
                                  ;                      |      GDT       |                           :
                                  ;                  +20 |----------------|                           :
                                  ;                      |      BIOS      |                           :
                                  ;                      |       CS       |                           :
                                  ;                  +28 |----------------|                           :
                                  ;                      |       SS       |                           :
                                  ;                      |                |                           :
                                  ;                      '----------------'                           :
                                  ;                                                                   :
                                  ;                                                                   :
                                  ;       SAMPLE OF SOURCE OR TARGET DESCRIPTOR                       :
                                  ;                                                                   :
                                  ;   SOURCE_TARGET_DEF          STRUC                                :
                                  ;                                                                   :
                                  ;     SEG_LIMIT          DW      ; SEGMENT LIMIT (1-65536 BYTES)    :
                                  ;     BASE_LO_WORD       DW      ; 24 BIT SEGMENT PHYSICAL          :
                                  ;     BASE_HI_BYTE       DB      ;    ADDRESS (0 TO (16M-1))        :
                                  ;     DATA_ACC_RIGHTS    DB      ; ACCESS RIGHTS BYTE               :
                                  ;     DATA_RESERVED      DW      ; RESERVED WORD                    :
                                  ;                                                                   :
                                  ;   SOURCE_TARGET              ENDS                                 :
                                  ;                                                                   :
                                  ;-------------------------------------------------------------------:
                                  ;-------------------------------------------------------------------------
                                  ;     THE GLOBAL DESCRIPTOR TABLE (ACTUAL LOCATION POINTED TO BY ES:SI)
                                  ;-------------------------------------------------------------------------

                                  BLOCKMOVE_GDT_DEF        STRUC
                                  ;
0000  00 00 00 00 00 00           DUMMY     DQ     0                       ; FIRST DESCRIPTOR NOT ACCESSIBLE
      00 00
0008  00 00 00 00 00 00           CGDT_LOC  DQ     0                       ; LOCATION OF CALLING ROUTINE GDT
      00 00
0010  00 00 00 00 00 00           SOURCE    DQ     0                       ; SOURCE DESCRIPTOR
      00 00
0018  00 00 00 00 00 00           TARGET    DQ     0                       ; TARGET DESCRIPTOR
      00 00
0020  00 00 00 00 00 00           BIOS_CS   DQ     0                       ; BIOS CODE DESCRIPTOR
      00 00
0028  00 00 00 00 00 00           TEMP_SS   DQ     0                       ; STACK DESCRIPTOR
      00 00
                                  ;
0030                              BLOCKMOVE_GDT_DEF        ENDS
                                  ;-------------------------------------------------------------------------
```

```
                                   ASSUME  CS:CODE
                                   ASSUME  DS:DATA

0183                        BLOCKMOVE    PROC     NEAR

                            ;------- INITIALIZE FOR VIRTUAL MODE

0183  FA                           CLI                            ; NO INTERRUPTS ALLOWED
0184  FC                           CLD                            ; SET DIRECTION
                                   PUSHA                          ; SAVE GENERAL PURPOSE REGS
0185  60                    +      DB      060H
0186  06                           PUSH    ES                     ; SAVE EXTRA SEGMENT
0187  1E                           PUSH    DS                     ;

                            ;------- CLEAR EXCEPTION ERROR FLAG

0188  2A C0                        SUB     AL,AL                  ;
018A  E6 80                        OUT     MFG_PORT,AL            ; SET TO 0

                            ;------- GATE ADDRESS BIT 20 ON

018C  B4 DF                        MOV     AH,ENABLE_BIT20        ;
018E  E8 03B0 R                    CALL    GATE_A20               ;
0191  3C 00                        CMP     AL,0                   ; WAS THE COMMAND ACCEPTED?
0193  74 07                        JZ      BL4                    ; GO IF YES
0195  B0 03                        MOV     AL,03H                 ; SET THE ERROR FLAG
0197  E6 80                        OUT     MFG_PORT,AL            ;
0199  E9 0270 R                    JMP     SHUT9                  ; EARLY EXIT

                            ;-------- SET SHUDOWN RETURN ADDR

019C  B0 8F                 BL4:   MOV     AL,SHUT_DOWN           ; SET THE SHUTDOWN BYTE
019E  E6 70                        OUT     CMOS_PORT,AL           ;  TO SHUT DOWN 9
01A0  EB 00                        JMP     SHORT $+2              ; IO DELAY
01A2  B0 09                        MOV     AL,9                   ;
01A4  E6 71                        OUT     CMOS_PORT+1,AL         ;

                                   ;=========================
                            ;-------- SET UP THE GDT DEFINITION
                                   ;=========================
                            ;-------- MAKE A 24 BIT ADDRESS OUT OF THE ES:SI

01A6  8C C0                        MOV     AX,ES                  ; GET THE CURRENT DATA SEGMENT
01A8  8B DE                        MOV     BX,SI                  ; GET THE CURRENT OFFSET
01AA  8A F4                        MOV     DH,AH                  ; DEVELOPE THE HIGH BYTE OF THE 24BIT ADDR
01AC  80 E6 F0                     AND     DH,0F0H                ; USE ONLY THE HIGH NIBBLE
                                   ISHR    DH,4                   ; SHIFT RIGHT 4
01AF                        + ??0000  LABEL   BYTE
01AF  D0 EE                 +      SHR     DH,1
01B1                        + ??0001  LABEL   BYTE
01AF                        +      ORG     OFFSET CS:??0000
01AF  C0                    +      DB      0C0H
01B1                        +      ORG     OFFSET CS:??0001
01B1  04                    +      DB      4
01B2  80 E4 0F                     AND     AH,00FH                ; STRIP HIGH NIBBLE FROM AH
                                   ISHL    AX,4                   ; SHIFT AX
01B5                        + ??0003  LABEL   BYTE
01B5  D1 E0                 +      SHL     AX,1
01B7                        + ??0004  LABEL   BYTE
01B5                        +      ORG     OFFSET CS:??0003
01B5                        + ??0005    LABEL    NEAR
01B5  C1                    +       DB       0C1H
01B7                        +      ORG     OFFSET CS:??0004
01B7  04                    +      DB      4
01B8  03 D8                        ADD     BX,AX                  ; DEVELOPE THE LOW WORD ADDRESS
01BA  73 02                        JNC     BL3A                   ; GO IF NO CARRY
01BC  FE C6                        INC     DH                     ; INCREMENT THE HIGH BYTE ADDRES
                                   ;=================
                            ;-------- SET THE GDT_LOC
                                   ;=================
01BE  26: 88 74 0C          BL3A:  MOV     ES:[SI].CGDT_LOC.BASE_HI_BYTE,DH   ; SET THE HIGH BYTE
01C2  26: 89 5C 0A                 MOV     ES:[SI].CGDT_LOC.BASE_LO_WORD,BX   ; SET THE LOW WORD

01C6  26: C7 44 08 FFFF            MOV     ES:[SI].CGDT_LOC.SEG_LIMIT,MAX_SEG_LEN
01CC  26: C7 44 0E 0000            MOV     ES:[SI].CGDT_LOC.DATA_RESERVED,0     ; RESERVED
                                   ;==============
                            ;-------- LOAD THE IDT
                                   ;==============
01D2  BD 02A1 R                    MOV     BP,OFFSET ROM_IDT_LOC
                                   SEGOV   CS                     ; LOAD THE IDT
01D5  2E                    +      DB      02EH
                                   LIDT    [BP]                   ;   REGISTER FROM THIS AREA
01D6  0F                    +      DB      00FH
01D7                        + ??0007  LABEL   BYTE
01D7  8B 5E 00              +      MOV     BX,WORD PTR [BP]
01DA                        + ??0008  LABEL   BYTE
01D7                        +      ORG     OFFSET CS:??0007
01D7  01                    +      DB      001H
01DA                        +      ORG     OFFSET CS:??0008
                                   ;=============
                            ;------- LOAD THE GDTR
                                   ;=============
                                   SEGOV   ES                     ; LOAD THE GLOBAL DESCRIPTOR TABLE REG
01DA  26                    +      DB      026H
                                   LGDT    [SI].CGDT_LOC
01DB  0F                    +      DB      00FH
01DC                        + ??000A  LABEL   BYTE
01DC  8B 54 08              +      MOV     DX,WORD PTR [SI].CGDT_LOC
01DF                        + ??000B  LABEL   BYTE
01DC                        +      ORG     OFFSET CS:??000A
01DC  01                    +      DB      001H
01DF                        +      ORG     OFFSET CS:??000B

                            ;------- SET THE DATA SEGMENT TO BIOS RAM

01DF  E8 0000 E                    CALL    DDS                    ; SET DS TO DATA AREA

                            ;-------- SAVE THE CALLING ROUTINE"S STACK

01E2  8C D0                        MOV     AX,SS                  ; GET THE STACK SEGMENT
01E4  A3 0069 R                    MOV     IO_ROM_SEG,AX          ; SAVE STACK SEGMENT
01E7  8B C4                        MOV     AX,SP                  ; SAVE STACK POINTER
01E9  A3 0067 R                    MOV     IO_ROM_INIT,AX         ;

                            PAGE
                            ;-------- MAKE A 24 BIT ADDRESS OUT OF THE SS (SP REMAINS USER SP)

01EC  8C D0                        MOV     AX,SS                  ; GET THE CURRENT STACK SEGMENT
01EE  8A F4                        MOV     DH,AH                  ; DEVELOPE THE HIGH BYTE OF THE 24BIT ADDR
01F0  80 E6 F0                     AND     DH,0F0H                ; USE ONLY THE HIGH NIBBLE
                                   ISHR    DH,4                   ; SHIFT RIGHT 4
01F3                        + ??000C  LABEL   BYTE
01F3  D0 EE                 +      SHR     DH,1
01F5                        + ??000D  LABEL   BYTE
01F3                        +      ORG     OFFSET CS:??000C
01F3  C0                    +       DB      0C0H
```

SECTION 5

```
01F5                       +          ORG    OFFSET CS:??000D
01F5  04                   +          DB     4
01F6  80 E4 0F                        AND    AH,0FH                    ; STRIP HIGH NIBBLE FROM AH
                                      ISHL   AX,4                      ; SHIFT AX
01F9                       + ??000F   LABEL  BYTE
01F9  D1 E0                +          SHL    AX,1
01FB                       + ??0010   LABEL  BYTE
01F9                       +          ORG    OFFSET CS:??000F
01F9                       + ??0011     LABEL  NEAR
01F9  C1                   +            DB     0C1H
01FB                       +          ORG    OFFSET CS:??0010
01FB  04                   +          DB     4

                             ;-------- SS IS NOW IN POSITION FOR A 24 BIT ADDRESS --> SETUP THE DESCRIPTOR

01FC  26: 88 74 2C           BL3:     MOV    ES:[SI].TEMP_SS.BASE_HI_BYTE,DH          ; SET THE HIGH BYTE
0200  26: 89 44 2A                    MOV    ES:[SI].TEMP_SS.BASE_LO_WORD,AX          ; SET THE LOW WORD
0204  26: C7 44 28 FFFF               MOV    ES:[SI].TEMP_SS.SEG_LIMIT,MAX_SEG_LEN    ; SET THE SS SEGMENT LIMIT
020A  26: C6 44 2D 93                 MOV    ES:[SI].TEMP_SS.DATA_ACC_RIGHTS,CPLO_DATA_ACCESS  ; SET CPL 0

                             ;-------- STACK IS NOW SET ---> SET UP THE CODE SEGMENT DESCRIPTOR

020F  26: C6 44 24 0F                 MOV    ES:[SI].BIOS_CS.BASE_HI_BYTE,CSEG@_HI    ; HIGH BYTE OF CS=0F
0214  26: C7 44 22 0000               MOV    ES:[SI].BIOS_CS.BASE_LO_WORD,CSEG@_LO    ; LOW WORD OF CS=0
021A  26: C7 44 20 FFFF               MOV    ES:[SI].BIOS_CS.SEG_LIMIT,MAX_SEG_LEN
0220  26: C6 44 25 9B                 MOV    ES:[SI].BIOS_CS.DATA_ACC_RIGHTS,CPLO_CODE_ACCESS
0225  26: C7 44 26 0000               MOV    ES:[SI].BIOS_CS.DATA_RESERVED,0          ; RESERVED


                             ;------  SWITCH TO VIRTUAL MODE

022B  B8 0001                         MOV    AX,VIRTUAL_ENABLE         ; MACHINE STATUS WORD NEEDED TO
                                      LMSW   AX                        ;  SWITCH TO VIRTUAL MODE
022E  0F                   +          DB     00FH
022F                       + ??0012   LABEL  BYTE
022F  8B F0                +          MOV    SI,AX
0231                       + ??0013   LABEL  BYTE
022F                       +          ORG    OFFSET CS:??0012
022F  01                   +          DB     001H
0231                       +          ORG    OFFSET CS:??0013
                                      JUMPFAR VIRT,BIOS_CS             ; MUST PURGE PRE-FETCH QUEUE
0231  EA                   +          DB     0EAH                      ; Jump far direct
0232  0236 R               +          DW     (OFFSET VIRT)             ;   to this offset
0234  0020                 +          DW     BIOS_CS          ;      in this segment
0236                         VIRT:

                             ;------- SET STACK SEGMENT  (NEEDED FOR POSSIBLE EXCEPTIONS)

0236  B8 0028                         MOV    AX,TEMP_SS                ; USER'S SS+SP IS NOT A DESCRIPTOR
0239  8E D0                           MOV    SS,AX                     ;

                             ;------- SETUP SOURCE/TARGET REGISTERS

023B  B8 0010                         MOV    AX,SOURCE                 ; GET THE SOURCE ENTRY
023E  8E D8                           MOV    DS,AX                     ;

0240  B8 0018                         MOV    AX,TARGET                 ; GET THE TARGET ENTRY
0243  8E C0                           MOV    ES,AX                     ;

0245  2B FF                           SUB    DI,DI                     ; SET INDEX REGS TO ZERO
0247  2B F6                           SUB    SI,SI                     ;

0249  F3/ A5                 REP      MOVSW                            ; MOVE THE BLOCK

                             ;------- CHECK FOR RAM PARITY BEFORE SHUTDOWN

024B  E4 61                           IN     AL,PORT_B                 ; GET THE PARITY LATCHES
024D  24 C0                           AND    AL,PARITY_ERR             ; STRIP UNWANTED BITS
024F  74 1C                           JZ     DONE1                     ; GO IF NO PARITY ERROR

                             ;-------- CLEAR PARITY BEFORE SHUTDOWN

0251  26: 8B 04                       MOV    AX,ES:[SI]                ; FETCH CURRENT TARGET DATA
0254  26: 89 04                       MOV    ES:[SI],AX                ; WRITE IT BACK
0257  8B 05                           MOV    AX,DS:[DI]                ; FETCH CURRENT SOURCE DATA
0259  89 05                           MOV    DS:[DI],AX                ; WRITE IT BACK
025B  B0 01                           MOV    AL,01                     ; SET PARITY CHECK ERROR
025D  E6 80                           OUT    MFG_PORT,AL               ;

025F  E4 61                           IN     AL,PORT_B
0261  EB 00                           JMP    SHORT $+2                 ; IO DELAY
0263  0C 0C                           OR     AL,RAM_PAR_OFF            ; TOGGLE PARITY CHECK LATCHES
0265  E6 61                           OUT    PORT_B,AL
0267  EB 00                           JMP    SHORT $+2                 ; IO DELAY
0269  24 F3                           AND    AL,RAM_PAR_ON             ;
026B  E6 61                           OUT    PORT_B,AL


                             ;-------- CAUSE A SHUTDOWN

026D  E9 0000 E              DONE1:   JMP    PROC_SHUTDOWN             ;

                                      ;====================
                             ;-------- RETURN FROM SHUTDOWN
                                      ;====================
0270                         SHUT9:
                             ;------- ENABLE NMI INTERRUPTS

0270  2A C0                           SUB    AL,AL                     ;
0272  E6 70                           OUT    CMOS_PORT,AL              ;

                             ;------- GATE ADDRESS BIT 20 OFF

0274  B4 DD                           MOV    AH,DISABLE_BIT20          ;
0276  E8 03B0 R                       CALL   GATE_A20                  ;
0279  3C 00                           CMP    AL,0                      ; COMMAND ACCEPTED?
027B  74 0A                           JZ     DONE3                     ; GO IF YES
027D  E4 80                           IN     AL,MFG_PORT               ; CHECK FOR ERROR
027F  3C 00                           CMP    AL,0                      ; WAS THERE AN ERROR?
0281  75 04                           JNZ    DONE3                     ; GO IF YES
0283  B0 03                           MOV    AL,03H                    ; SET ERROR FLAG
0285  E6 80                           OUT    MFG_PORT,AL               ;

                             ;-------- RESTORE USERS STACK

0287  E8 0000 E              DONE3:   CALL   DDS                       ; SET DS TO DATA AREA

028A  A1 0069 R                       MOV    AX,IO_ROM_SEG             ; SAVE STACK SEGMENT
028D  8E D0                           MOV    SS,AX                     ; RESTORE THE STACK POINTER

028F  A1 0067 R                       MOV    AX,IO_ROM_INIT            ;
0292  8B E0                           MOV    SP,AX                     ;

                             ;------- RESTORE THE USER DATA SEGMENT

0294  1F                              POP    DS                        ; RESTORE USER DATA SEGMENT
```

```
0295 07              POP     ES                      ; RESTORE USER EXTRA SEGMENT
                     POPA                            ; RESTORE THE GENERAL PURPOSE REGS
0296 61          +   DB      061H
0297 86 C4           XCHG    AL,AH                   ; SAVE AL

0299 E4 80           IN      AL,MFG_PORT             ; CHECK THE ENDING STATUS
029B 3C 00           CMP     AL,0                    ; SET THE ZERO FLAG
029D 86 E0           XCHG    AH,AL                   ; RESTORE AL
029F FB              STI                             ; TURN INTERRUPTS ON
02A0 CF              IRET                            ; RETURN TO USER

          ;-------- ROM IDT LOCATION

= 0100    ROM_IDT_LEN    EQU     32*8                ; SIZE OF THE EXCEPTION INTERRUPTS

02A1      ROM_IDT_LOC:

                     IDT_GDT_DEF   ROM_IDT_LEN,ROM_IDT,CSEG@_HI
02A1 0100        +   DW      ROM_IDT_LEN         ; Segment limit
02A3 02A7 R      +   DW      ROM_IDT         ; Segment base address - low word
02A5 0F          +   DB      CSEG@_HI            ; Segment base address - high byte
02A6 00          +   DB      0               ; Reserved

          ;-------- THE ROM EXCEPTION INTERRUPT VECTORS

02A7      ROM_IDT:
          ;EXCEPTION 00
                     DESCR_DEF  GATE,EX_INT,BIOS_CS,0,TRAP_GATE
02A7 03A7 R      +   DW      EX_INT          ; Destination offset
02A9 0020        +   DW      BIOS_CS         ; Destination segment selector
02AB 00          +   DB      0               ; Word count for stack-to-stack copy (only for call gates when PL cha
          nges)
02AC 87          +   DB      TRAP_GATE           ; Access rights byte
02AD 0000        +   DW      0               ; Reserved
          ;EXCEPTION 01
                     DESCR_DEF  GATE,EX_INT,BIOS_CS,0,TRAP_GATE
02AF 03A7 R      +   DW      EX_INT          ; Destination offset
02B1 0020        +   DW      BIOS_CS         ; Destination segment selector
02B3 00          +   DB      0               ; Word count for stack-to-stack copy (only for call gates when PL cha
          nges)
02B4 87          +   DB      TRAP_GATE           ; Access rights byte
02B5 0000        +   DW      0               ; Reserved
          ;EXCEPTION 02
                     DESCR_DEF  GATE,EX_INT,BIOS_CS,0,TRAP_GATE
02B7 03A7 R      +   DW      EX_INT          ; Destination offset
02B9 0020        +   DW      BIOS_CS         ; Destination segment selector
02BB 00          +   DB      0               ; Word count for stack-to-stack copy (only for call gates when PL cha
          nges)
02BC 87          +   DB      TRAP_GATE           ; Access rights byte
02BD 0000        +   DW      0               ; Reserved
          ;EXCEPTION 03
                     DESCR_DEF  GATE,EX_INT,BIOS_CS,0,TRAP_GATE
02BF 03A7 R      +   DW      EX_INT          ; Destination offset
02C1 0020        +   DW      BIOS_CS         ; Destination segment selector
02C3 00          +   DB      0               ; Word count for stack-to-stack copy (only for call gates when PL cha
          nges)
02C4 87          +   DB      TRAP_GATE           ; Access rights byte
02C5 0000        +   DW      0               ; Reserved
          ;EXCEPTION 04
                     DESCR_DEF  GATE,EX_INT,BIOS_CS,0,TRAP_GATE
02C7 03A7 R      +   DW      EX_INT          ; Destination offset
02C9 0020        +   DW      BIOS_CS         ; Destination segment selector
02CB 00          +   DB      0               ; Word count for stack-to-stack copy (only for call gates when PL cha
          nges)
02CC 87          +   DB      TRAP_GATE           ; Access rights byte
02CD 0000        +   DW      0               ; Reserved
          ;EXCEPTION 05
                     DESCR_DEF  GATE,EX_INT,BIOS_CS,0,TRAP_GATE
02CF 03A7 R      +   DW      EX_INT          ; Destination offset
02D1 0020        +   DW      BIOS_CS         ; Destination segment selector
02D3 00          +   DB      0               ; Word count for stack-to-stack copy (only for call gates when PL cha
          nges)
02D4 87          +   DB      TRAP_GATE           ; Access rights byte
02D5 0000        +   DW      0               ; Reserved
          ;EXCEPTION 06
                     DESCR_DEF  GATE,EX_INT,BIOS_CS,0,TRAP_GATE
02D7 03A7 R      +   DW      EX_INT          ; Destination offset
02D9 0020        +   DW      BIOS_CS         ; Destination segment selector
02DB 00          +   DB      0               ; Word count for stack-to-stack copy (only for call gates when PL cha
          nges)
02DC 87          +   DB      TRAP_GATE           ; Access rights byte
02DD 0000        +   DW      0               ; Reserved
          ;EXCEPTION 07
                     DESCR_DEF  GATE,EX_INT,BIOS_CS,0,TRAP_GATE
02DF 03A7 R      +   DW      EX_INT          ; Destination offset
02E1 0020        +   DW      BIOS_CS         ; Destination segment selector
02E3 00          +   DB      0               ; Word count for stack-to-stack copy (only for call gates when PL cha
          nges)
02E4 87          +   DB      TRAP_GATE           ; Access rights byte
02E5 0000        +   DW      0               ; Reserved
          ;EXCEPTION 08
                     DESCR_DEF  GATE,EX_INT,BIOS_CS,0,TRAP_GATE
02E7 03A7 R      +   DW      EX_INT          ; Destination offset
02E9 0020        +   DW      BIOS_CS         ; Destination segment selector
02EB 00          +   DB      0               ; Word count for stack-to-stack copy (only for call gates when PL cha
          nges)
02EC 87          +   DB      TRAP_GATE           ; Access rights byte
02ED 0000        +   DW      0               ; Reserved
          ;EXCEPTION 09
                     DESCR_DEF  GATE,EX_INT,BIOS_CS,0,TRAP_GATE
02EF 03A7 R      +   DW      EX_INT          ; Destination offset
02F1 0020        +   DW      BIOS_CS         ; Destination segment selector
02F3 00          +   DB      0               ; Word count for stack-to-stack copy (only for call gates when PL cha
          nges)
02F4 87          +   DB      TRAP_GATE           ; Access rights byte
02F5 0000        +   DW      0               ; Reserved
          ;EXCEPTION 10
                     DESCR_DEF  GATE,EX_INT,BIOS_CS,0,TRAP_GATE
02F7 03A7 R      +   DW      EX_INT          ; Destination offset
02F9 0020        +   DW      BIOS_CS         ; Destination segment selector
02FB 00          +   DB      0               ; Word count for stack-to-stack copy (only for call gates when PL cha
          nges)
02FC 87          +   DB      TRAP_GATE           ; Access rights byte
02FD 0000        +   DW      0               ; Reserved
          ;EXCEPTION 11
                     DESCR_DEF  GATE,EX_INT,BIOS_CS,0,TRAP_GATE
02FF 03A7 R      +   DW      EX_INT          ; Destination offset
0301 0020        +   DW      BIOS_CS         ; Destination segment selector
0303 00          +   DB      0               ; Word count for stack-to-stack copy (only for call gates when PL cha
          nges)
0304 87          +   DB      TRAP_GATE           ; Access rights byte
0305 0000        +   DW      0               ; Reserved
          ;EXCEPTION 12
                     DESCR_DEF  GATE,EX_INT,BIOS_CS,0,TRAP_GATE
0307 03A7 R      +   DW      EX_INT          ; Destination offset
0309 0020        +   DW      BIOS_CS         ; Destination segment selector
```

```
030B 00           +             DB      0                 ; Word count for stack-to-stack copy (only for call gates when PL cha
                  nges)
030C 87           +             DB      TRAP_GATE                 ; Access rights byte
030D 0000         +             DW      0                 ; Reserved
                  ;EXCEPTION 13
                                DESCR_DEF  GATE,EX_INT,BIOS_CS,0,TRAP_GATE
030F 03A7 R       +             DW      EX_INT            ; Destination offset
0311 0020         +             DW      BIOS_CS           ; Destination segment selector
0313 00           +             DB      0                 ; Word count for stack-to-stack copy (only for call gates when PL cha
                  nges)
0314 87           +             DB      TRAP_GATE                 ; Access rights byte
0315 0000         +             DW      0                 ; Reserved
                  ;EXCEPTION 14
                                DESCR_DEF  GATE,EX_INT,BIOS_CS,0,TRAP_GATE
0317 03A7 R       +             DW      EX_INT            ; Destination offset
0319 0020         +             DW      BIOS_CS           ; Destination segment selector
031B 00           +             DB      0                 ; Word count for stack-to-stack copy (only for call gates when PL cha
                  nges)
031C 87           +             DB      TRAP_GATE                 ; Access rights byte
031D 0000         +             DW      0                 ; Reserved
                  ;EXCEPTION 15
                                DESCR_DEF  GATE,EX_INT,BIOS_CS,0,TRAP_GATE
031F 03A7 R       +             DW      EX_INT            ; Destination offset
0321 0020         +             DW      BIOS_CS           ; Destination segment selector
0323 00           +             DB      0                 ; Word count for stack-to-stack copy (only for call gates when PL cha
                  nges)
0324 87           +             DB      TRAP_GATE                 ; Access rights byte
0325 0000         +             DW      0                 ; Reserved
                  ;EXCEPTION 16
                                DESCR_DEF  GATE,EX_INT,BIOS_CS,0,TRAP_GATE
0327 03A7 R       +             DW      EX_INT            ; Destination offset
0329 0020         +             DW      BIOS_CS           ; Destination segment selector
032B 00           +             DB      0                 ; Word count for stack-to-stack copy (only for call gates when PL cha
                  nges)
032C 87           +             DB      TRAP_GATE                 ; Access rights byte
032D 0000         +             DW      0                 ; Reserved
                  ;EXCEPTION 17
                                DESCR_DEF  GATE,EX_INT,BIOS_CS,0,TRAP_GATE
032F 03A7 R       +             DW      EX_INT            ; Destination offset
0331 0020         +             DW      BIOS_CS           ; Destination segment selector
0333 00           +             DB      0                 ; Word count for stack-to-stack copy (only for call gates when PL cha
                  nges)
0334 87           +             DB      TRAP_GATE                 ; Access rights byte
0335 0000         +             DW      0                 ; Reserved
                  ;EXCEPTION 18
                                DESCR_DEF  GATE,EX_INT,BIOS_CS,0,TRAP_GATE
0337 03A7 R       +             DW      EX_INT            ; Destination offset
0339 0020         +             DW      BIOS_CS           ; Destination segment selector
033B 00           +             DB      0                 ; Word count for stack-to-stack copy (only for call gates when PL cha
                  nges)
033C 87           +             DB      TRAP_GATE                 ; Access rights byte
033D 0000         +             DW      0                 ; Reserved
                  ;EXCEPTION 19
                                DESCR_DEF  GATE,EX_INT,BIOS_CS,0,TRAP_GATE
033F 03A7 R       +             DW      EX_INT            ; Destination offset
0341 0020         +             DW      BIOS_CS           ; Destination segment selector
0343 00           +             DB      0                 ; Word count for stack-to-stack copy (only for call gates when PL cha
                  nges)
0344 87           +             DB      TRAP_GATE                 ; Access rights byte
0345 0000         +             DW      0                 ; Reserved
                  ;EXCEPTION 20
                                DESCR_DEF  GATE,EX_INT,BIOS_CS,0,TRAP_GATE
0347 03A7 R       +             DW      EX_INT            ; Destination offset
0349 0020         +             DW      BIOS_CS           ; Destination segment selector
034B 00           +             DB      0                 ; Word count for stack-to-stack copy (only for call gates when PL cha
                  nges)
034C 87           +             DB      TRAP_GATE                 ; Access rights byte
034D 0000         +             DW      0                 ; Reserved
                  ;EXCEPTION 21
                                DESCR_DEF  GATE,EX_INT,BIOS_CS,0,TRAP_GATE
034F 03A7 R       +             DW      EX_INT            ; Destination offset
0351 0020         +             DW      BIOS_CS           ; Destination segment selector
0353 00           +             DB      0                 ; Word count for stack-to-stack copy (only for call gates when PL cha
                  nges)
0354 87           +             DB      TRAP_GATE                 ; Access rights byte
0355 0000         +             DW      0                 ; Reserved
                  ;EXCEPTION 22
                                DESCR_DEF  GATE,EX_INT,BIOS_CS,0,TRAP_GATE
0357 03A7 R       +             DW      EX_INT            ; Destination offset
0359 0020         +             DW      BIOS_CS           ; Destination segment selector
035B 00           +             DB      0                 ; Word count for stack-to-stack copy (only for call gates when PL cha
                  nges)
035C 87           +             DB      TRAP_GATE                 ; Access rights byte
035D 0000         +             DW      0                 ; Reserved
                  ;EXCEPTION 23
                                DESCR_DEF  GATE,EX_INT,BIOS_CS,0,TRAP_GATE
035F 03A7 R       +             DW      EX_INT            ; Destination offset
0361 0020         +             DW      BIOS_CS           ; Destination segment selector
0363 00           +             DB      0                 ; Word count for stack-to-stack copy (only for call gates when PL cha
                  nges)
0364 87           +             DB      TRAP_GATE                 ; Access rights byte
0365 0000         +             DW      0                 ; Reserved
                  ;EXCEPTION 24
                                DESCR_DEF  GATE,EX_INT,BIOS_CS,0,TRAP_GATE
0367 03A7 R       +             DW      EX_INT            ; Destination offset
0369 0020         +             DW      BIOS_CS           ; Destination segment selector
036B 00           +             DB      0                 ; Word count for stack-to-stack copy (only for call gates when PL cha
                  nges)
036C 87           +             DB      TRAP_GATE                 ; Access rights byte
036D 0000         +             DW      0                 ; Reserved
                  ;EXCEPTION 25
                                DESCR_DEF  GATE,EX_INT,BIOS_CS,0,TRAP_GATE
036F 03A7 R       +             DW      EX_INT            ; Destination offset
0371 0020         +             DW      BIOS_CS           ; Destination segment selector
0373 00           +             DB      0                 ; Word count for stack-to-stack copy (only for call gates when PL cha
                  nges)
0374 87           +             DB      TRAP_GATE                 ; Access rights byte
0375 0000         +             DW      0                 ; Reserved
                  ;EXCEPTION 26
                                DESCR_DEF  GATE,EX_INT,BIOS_CS,0,TRAP_GATE
0377 03A7 R       +             DW      EX_INT            ; Destination offset
0379 0020         +             DW      BIOS_CS           ; Destination segment selector
037B 00           +             DB      0                 ; Word count for stack-to-stack copy (only for call gates when PL cha
                  nges)
037C 87           +             DB      TRAP_GATE                 ; Access rights byte
037D 0000         +             DW      0                 ; Reserved
                  ;EXCEPTION 27
                                DESCR_DEF  GATE,EX_INT,BIOS_CS,0,TRAP_GATE
037F 03A7 R       +             DW      EX_INT            ; Destination offset
0381 0020         +             DW      BIOS_CS           ; Destination segment selector
0383 00           +             DB      0                 ; Word count for stack-to-stack copy (only for call gates when PL cha
                  nges)
0384 87           +             DB      TRAP_GATE                 ; Access rights byte
0385 0000         +             DW      0                 ; Reserved
                  ;EXCEPTION 28
                                DESCR_DEF  GATE,EX_INT,BIOS_CS,0,TRAP_GATE
```

```
0387  03A7 R     +          DW      EX_INT          ; Destination offset
0389  0020       +          DW      BIOS_CS         ; Destination segment selector
038B  00         +          DB      0               ; Word count for stack-to-stack copy (only for call gates when PL cha
                 nges)
038C  87         +          DB      TRAP_GATE               ; Access rights byte
038D  0000       +          DW      0               ; Reserved
                 ;EXCEPTION 29
                            DESCR_DEF  GATE,EX_INT,BIOS_CS,0,TRAP_GATE
038F  03A7 R     +          DW      EX_INT          ; Destination offset
0391  0020       +          DW      BIOS_CS         ; Destination segment selector
0393  00         +          DB      0               ; Word count for stack-to-stack copy (only for call gates when PL cha
                 nges)
0394  87         +          DB      TRAP_GATE               ; Access rights byte
0395  0000       +          DW      0               ; Reserved
                 ;EXCEPTION 30
                            DESCR_DEF  GATE,EX_INT,BIOS_CS,0,TRAP_GATE
0397  03A7 R     +          DW      EX_INT          ; Destination offset
0399  0020       +          DW      BIOS_CS         ; Destination segment selector
039B  00         +          DB      0               ; Word count for stack-to-stack copy (only for call gates when PL cha
                 nges)
039C  87         +          DB      TRAP_GATE               ; Access rights byte
039D  0000       +          DW      0               ; Reserved
                 ;EXCEPTION 31
                            DESCR_DEF  GATE,EX_INT,BIOS_CS,0,TRAP_GATE
039F  03A7 R     +          DW      EX_INT          ; Destination offset
03A1  0020       +          DW      BIOS_CS         ; Destination segment selector
03A3  00         +          DB      0               ; Word count for stack-to-stack copy (only for call gates when PL cha
                 nges)
03A4  87         +          DB      TRAP_GATE               ; Access rights byte
03A5  0000       +          DW      0               ; Reserved

                 ;------- EXCEPTION INTERRUPT HANDLER

03A7             EX_INT:
03A7  B0 02                 MOV     AL,02H                  ; SET EXCEPTION INT
03A9  E6 80                 OUT     MFG_PORT,AL             ;
03AB  E9 0000 E             JMP     PROC_SHUTDOWN           ; CAUSE A EARLY SHUTDOWN
03AE             EX_INT1:
03AE  EB FE                 JMP   EX_INT1                   ; STAY HERE TILL SHUTDOWN

03B0             BLOCKMOVE    ENDP
                 PAGE
                 ;--------------------------------------------------------------------
                 ; GATE_A20
                 ;      THIS ROUTINE CONTROLS A SIGNAL WHICH GATES ADDRESS BIT 20.
                 ;      THE GATE A20 SIGNAL IS AN OUTPUT OF THE 8042 SLAVE PROCCESSOR.
                 ;      ADDRESS BIT 20 SHOULD BE GATED ON BEFORE ENTERING PROTECTED MODE.
                 ;      IT SHOULD BE GATED OFF AFTER ENTERING REAL MODE FROM PROTECTED
                 ;      MODE.
                 ; INPUT
                 ;      (AH)=DDH ADDRESS BIT 20 GATE OFF. (A20 ALWAYS ZERO)
                 ;      (AH)=DFH ADDRESS BIT 20 GATE ON. (A20 CONTROLLED BY 80286)
                 ; OUTPUT
                 ;      (AL)=0 OPERATION SUCCESSFUL. 8042 HAS ACCEPTED COMMAND.
                 ;      (AL)=2 FAILURE--8042 UNABLE TO ACCEPT COMMAND.
                 ;--------------------------------------------------------------------
03B0             GATE_A20     PROC
03B0  FA                    CLI                     ;DISABLE INTERRUPTS WHILE USING 8042
03B1  E8 03C7 R             CALL    EMPTY_8042      ;INSURE 8042 INPUT BUFFER EMPTY
03B4  75 10                 JNZ     GATE_A20_RETURN ;RETURN IF 8042 UNABLE TO ACCEPT COMMAND
03B6  B0 D1                 MOV     AL,0D1H         ;8042 COMMAND TO WRITE OUTPUT PORT
03B8  E6 64                 OUT     STATUS_PORT,AL  ;OUTPUT COMMAND TO 8042
03BA  E8 03C7 R             CALL    EMPTY_8042      ;WAIT FOR 8042 TO ACCEPT COMMAND
03BD  75 07                         JNZ     GATE_A20_RETURN ;RETURN IF 8042 UNABLE TO ACCEPT COMMAND
03BF  8A C4                         MOV     AL,AH           ;8042 PORT DATA
03C1  E6 60                         OUT     PORT_A,AL       ;OUTPUT PORT DATA TO 8042
03C3  E8 03C7 R                     CALL    EMPTY_8042      ;WAIT FOR 8042 TO ACCEPT PORT DATA
                               ;
                               ;----- 8042 OUTPUT WILL SWITCH WITHIN 20 USEC OF ACCEPTING PORT DATA -----
                               ;
03C6                           GATE_A20_RETURN:
03C6  C3                               RET
                               ;--------------------------------------------------------------------
                               ; EMPTY_8042
                               ;       THIS ROUTINE WAITS FOR THE 8042 INPUT BUFFER TO EMPTY.
                               ; INPUT
                               ;       NONE
                               ; OUTPUT
                               ;       (AL)=0 8042 INPUT BUFFER EMPTY (ZERO FLAG SET)
                               ;       (AL)=2 TIME OUT, 8042 INPUT BUFFER FULL (NON-ZERO FLAG SET)
                               ;--------------------------------------------------------------------
03C7                           EMPTY_8042:
03C7  51                               PUSH    CX              ;SAVE CX
03C8  2B C9                            SUB     CX,CX           ;CX=0, WILL BE USED AS TIME OUT VALUE
03CA                           EMPTY_LOOP:
03CA  E4 64                            IN      AL,STATUS_PORT  ;READ 8042 STATUS PORT
03CC  24 02                            AND     AL,INPT_BUF_FULL;TEST INPUT BUFFER FULL FLAG (BIT 1)
03CE  E0 FA                            LOOPNZ  EMPTY_LOOP      ;LOOP UNTIL INPUT BUFFER EMPTY OR TIME OUT
03D0  59                               POP     CX              ;RESTORE CX
03D1  C3                               RET

03D2                           GATE_A20        ENDP
                               PAGE
                               ;------ INT 15 (FUNCTION 88H - IO MEMORY SIZE DETERMINE) --------
                               ; EXT_MEMORY                                                    :
                               ;       THIS ROUTINE RETURNS  THE AMOUNT OF MEMORY IN THE      :
                               ;       SYSTEM THAT IS LOCATED STARTING AT THE 1024K ADDRESSING :
                               ;       RANGE, AS DETERMINED BY THE POST ROUTINES.             :
                               ;       NOTE THAT THE SYSTEM MAY NOT BE ABLE TO USE I/O MEMORY :
                               ;       UNLESS THERE IS A FULL COMPLEMENT OF 512K OR 640 BYTES :
                               ;       ON THE PLANAR.  THIS SIZE IS STORED IN CMOS AT ADDRESS :
                               ;       30 AND 31.                                             :
                               ; INPUT                                                         :
                               ;       AH = 88H                                               :
                               ;                                                               :
                               ;       THE IO MEMORY SIZE VARIABLE IS SET DURING POWER ON     :
                               ;       DIAGNOSTICS ACCORDING TO THE FOLLOWING ASSUMPTIONS:    :
                               ;                                                               :
                               ;       3. ALL INSTALLED MEMORY IS FUNCTIONAL.                 :
                               ;                                                               :
                               ;       4. ALL MEMORY FROM 0 TO 640K MUST BE CONTIGUOUS.       :
                               ;                                                               :
                               ; OUTPUT                                                        :
                               ;       (AX) = NUMBER OF CONTIGUOUS 1K BLOCKS OF MEMORY A      :
                               ;              AVAILABLE STARTING AT ADDRESS 1024K.            :
                               ;                                                               :
                               ;---------------------------------------------------------------:
03D2                           EXT_MEMORY          PROC
03D2  FB                               STI                     ; INTERRUPTS BACK ON
03D3  B0 31                            MOV     AL,31H          ; GET THE HIGH BYTE OF IO MEMORY
03D5  E6 70                            OUT     CMOS_PORT,AL    ;
03D7  EB 00                            JMP     SHORT $+2       ; IO DELAY
03D9  E4 71                            IN      AL,CMOS_PORT+1
03DB  86 C4                            XCHG    AL,AH           ; PUT HIGH BYTE IN POSITION (AH)
03DD  B0 30                            MOV     AL,30H          ; GET THE LOW BYTE OF IO MEMORY
03DF  E6 70                            OUT     CMOS_PORT,AL    ;
```

SECTION 5

```
03E1  EB 00                JMP     SHORT $+2              ; IO DELAY
03E3  E4 71                IN      AL,CMOS_PORT+1         ;
03E5  CF                   IRET                           ; RETURN TO USER
03E6                  EXT_MEMORY      ENDP
                      PAGE
                      ;------ INT 15H (FUNCTION 89H) ------------------------------------
                      ; PURPOSE:                                                         :
                      ;       THIS BIOS FUNCTION PROVIDES A MEANS TO THE USER TO         :
                      ;       SWITCH INTO VIRTUAL (PROTECTED) MODE.  UPON COMPLETION     :
                      ;       OF THIS FUNCTION THE PROCESSOR WILL BE IN  VIRTUAL         :
                      ;       (PROTECTED) MODE AND CONTROL WILL BE TRANSFERED TO THE     :
                      ;       CODE SEGMENT THAT WAS SPECIFIED BY THE USER.               :
                      ;                                                                  :
                      ; ENTRY REQUIREMENTS:                                              :
                      ;                                                                  :
                      ;       ES:SI POINTS TO A DESCRIPTOR TABLE (GDT) BUILT BEFORE      :
                      ;       INTERRUPTING TO THIS FUNCTION.  THESE DESCRIPTORS ARE      :
                      ;       ARE USED BY THIS FUNCTION TO INITIALIZE THE IDTR, THE      :
                      ;       GDTR AND THE STACK SEGMENT SELECTOR.  THE DATA SEGMENT     :
                      ;       (DS) SELECTOR AND THE EXTRA SEGMENT (ES) SELECTOR WILL     :
                      ;       BE INITIALIZE TO DESCRIPTORS BUILT BY THE ROUTINE USING    :
                      ;       THIS FUNCTION.                                             :
                      ;       BH - OFFSET INTO THE INTERRUPT DESCRIPTOR TABLE            :
                      ;            STATING WHERE THE FIRST EIGHT HARDWARE INTERRUPTS     :
                      ;            WILL BEGIN. ( INTERRUPT LEVEL 1 )                     :
                      ;       BL - OFFSET INTO THE INTERRUPT DESCRIPTOR TABLE            :
                      ;            STATING WHERE THE SECOND EIGHT HARDWARE               :
                      ;            INTERRUPTS WILL BEGIN. ( INTERRUPT LEVEL 2 )          :
                      ;                                                                  :
                      ;                                                                  :
                      ; THE DESCRIPTORS ARE DEFINED AS FOLLOWS:                          :
                      ;                                                                  :
                      ;       1.   THE FIRST DESCRIPTOR IS THE REQUIRED DUMMY.           :
                      ;            (USER INITIALIZED TO 0)                               :
                      ;       2.   THE SECOND DESCRIPTOR POINTS TO THE GDT TABLE AS      :
                      ;            A DATA SEGMENT.                                       :
                      ;            (USER INITIALIZED)                                    :
                      ;       3.   THE THIRD DESCRIPTOR POINTS TO THE USER DEFINED       :
                      ;            INTERRUPT DESCRIPTOR TABLE (IDT).                     :
                      ;            (USER INITIALIZED)                                    :
                      ;       4.   THE FORTH DESCRIPTOR POINTS TO THE USER'S DATA        :
                      ;            SEGMENT (DS).                                         :
                      ;            (USER INITIALIZED)                                    :
                      ;       5.   THE FIFTH DESCRIPTOR POINTS TO THE USER'S EXTRA       :
                      ;            SEGMENT (ES).                                         :
                      ;            (USER INITIALIZED)                                    :
                      ;       6.   THE SIXTH DESCRIPTOR POINTS TO THE USER'S STACK       :
                      ;            SEGMENT (SS).                                         :
                      ;            (USER INITIALIZED)                                    :
                      ;       7.   THE SEVENTH DESCRIPTOR POINTS TO THE CODE SEGMENT     :
                      ;            THAT THIS FUNCTION WILL RETURN TO.                    :
                      ;            (USER INITIALIZED TO THE USER'S CODE SEGMENT.)        :
                      ;       8.   THE EIGTH DESCRIPTOR IS USED BY THIS FUNCTION TO      :
                      ;            ESTABLISH A CODE SEGMENT FOR ITSELF. THIS IS          :
                      ;            NEEDED SO THAT THIS FUNCTION CAN COMPLETE IT'S        :
                      ;            EXECUTION WHILE IN PROTECTED MODE.  WHEN CONTROL      :
                      ;            GETS PASSED TO THE USER'S CODE THIS DESCRIPTOR CAN    :
                      ;            BE USED BY HIM IN ANY WAY HE CHOOSES.                 :
                      ;                                                                  :
                      ;    NOTE -  EACH DESCRIPTOR MUST CONTAIN ALL THE NECESSARY        :
                      ;            DATA  I.E. THE LIMIT,  BASE ADDRESS AND THE ACCESS    :
                      ;            RIGHTS BYTE.                                          :
                      ;                                                                  :
                      ;       AH=88H  (FUNCTION CALL)                                    :
                      ;       ES:SI = LOCATION OF THE GDT TABLE BUILD BY ROUTINE         :
                      ;       USING THIS FUNCTION.                                       :
                      ;                                                                  :
                      ; EXIT PARAMETERS:                                                 :
                      ;                                                                  :
                      ;       AH = 0   IF SUCCESSFUL                                     :
                      ;       ALL SEGMENT REGISTERS ARE CHANGED, AX AND BP DESTROYED     :
                      ;                                                                  :
                      ; CONSIDERATIONS:                                                  :
                      ;                                                                  :
                      ;       1.   NO BIOS AVAILABLE TO USER.  USER MUST HANDLE ALL      :
                      ;            IO COMMANDS.                                          :
                      ;       2.   INTERRUPTS - INTERRUPT VECTOR LOCATIONS MUST BE       :
                      ;            MOVED,  DUE TO THE 286 RESERVED AREAS.  THE           :
                      ;            HARDWARE INTERRUPT CONTROLLERS MUST BE REINITIALIZED:
                      ;            TO DEFINE LOCATIONS THAT DO NOT RESIDE IN THE 286     :
                      ;            RESERVED AREAS.                                       :
                      ;       3.   EXCEPTION INTERRUPT TABLE AND HANDLER MUST BE         :
                      ;            INITIALIZED BY THE USER.                              :
                      ;       4.   THE INTERRUPT DESCRIPTOR TABLE MUST NOT OVERLAP       :
                      ;            THE REAL MODE BIOS INTERRUPT DESCRIPTOR TABLE.        :
                      ;       5.   THE FOLLOWING GIVES AN IDEA OF WHAT THE USER CODE     :
                      ;            SHOULD LOOK LIKE WHEN INVOKING THIS FUNCTION.         :
                      ;                                                                  :
                      ;       Real mode --->      "USER CODE"                            :
                      ;                    "      MOV    AX,GDT SEGMENT                  :
                      ;                    "      MOV    ES,AX                           :
                      ;                    "      MOV    SI,GDT OFFSET                   :
                      ;                    "      MOV    BH,HARDWARE INT LEVEL 1 OFFSET  :
                      ;                    "      MOV    BL,HARDWARE INT LEVEL 2 OFFSET  :
                      ;                    "      MOV    AH,88H                          :
                      ;                    "      INT    15H                             :
                      ;    Virtual mode --->      "USER CODE"                            :
                      ;                                                                  :
                      ;                                                                  :
                      ; DESCRIPTION:                                                     :
                      ;                                                                  :
                      ;       1.   CLI (NO INTERRUPTS ALLOWED) WHILE THIS FUNCTION IS    :
                      ;            EXECUTING.                                            :
                      ;       2.   ADDRESS LINE 20 IS GATED ACTIVE.                      :
                      ;       3.   THE CURRENT USER STACK SEGMENT DESCRIPTOR IS          :
                      ;            INITIALIZED.                                          :
                      ;       4.   THE GDTR IS LOADED WITH THE GDT BASE ADDRESS.         :
                      ;       5.   THE IDTR IS LOADED WITH THE IDT BASE ADDRESS.         :
                      ;       6.   THE 8259 IS REINITIALIZED WITH THE NEW INTERRUPT      :
                      ;            OFFSETS.                                              :
                      ;       7.   THE PROCESSOR IS PUT IN VIRTUAL MODE WITH THE CODE    :
                      ;            SEGMENT DESIGNATED FOR THIS FUNCTION.                 :
                      ;       8.   DATA SEGMENT IS LOADED WITH THE USER DEFINED          :
                      ;            SELECTOR FOR THE DS REGISTER.                         :
                      ;       9.   EXTRA SEGMENT IS LOADED WITH THE USER DEFINED         :
                      ;            SELECTOR FOR THE ES REGISTER.                         :
                      ;      10.   STACK SEGMENT IS LOADED WITH THE USER DEFINED         :
                      ;            SELECTOR FOR THE SS REGISTER.                         :
                      ;      11.   CODE SEGMENT DESCRIPTOR SELECTOR VALUE IS             :
                      ;            SUBSTITUTED ON THE STACK FOR RETURN TO USER.          :
                      ;      12.   WE TRANSFER CONTROL TO THE USER WITH INTERRUPTS       :
                      ;            DISABLED.                                             :
                      ;-------------------------------------------------------------------
                      page
                      ;       THE FOLLOWING DIAGRAM DEPICTS THE ORGANIZATION
                      ;       OF GDT.
```

```
                                        ;-----------------------------------------------------------------:
                                        ;                              G D T                                :
                                        ;                                                                   :
                                        ;                              .--------------.                     :
                                        ;                              v              |                     :
                                        ;    (ES:SI)-->>  +00 .---------------.       |                     :
                                        ;                     |     DUMMY      |      |                     :
                                        ;                     |                |      |                     :
                                        ;                 +08 |----------------|      |                     :
                                        ;                     |      GDT       | ---'                       :
                                        ;                     |                |                            :
                                        ;                 +10 |----------------|                            :
                                        ;                     |      IDT       |                            :
                                        ;                     |                |                            :
                                        ;                 +18 |----------------|                            :
                                        ;                     |      DS        |                            :
                                        ;                     |                |                            :
                                        ;                 +20 |----------------|                            :
                                        ;                     |      ES        |                            :
                                        ;                     |                |                            :
                                        ;                 +28 |----------------|                            :
                                        ;                     |      SS        |                            :
                                        ;                     |                |                            :
                                        ;                 +30 |----------------|                            :
                                        ;                     |      CS        |                            :
                                        ;                     |                |                            :
                                        ;                 +38 |----------------|                            :
                                        ;                     |   TEMP BIOS    |                            :
                                        ;                     |      CS        |                            :
                                        ;                     '----------------'                            :
                                        ;                                                                   :
                                        ;-----------------------------------------------------------------:

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

                                        VIRTUAL_ENABLE_GDT_DEF  STRUC

 0000  00 00 00 00 00 00                DUMY      DQ     0               ; FIRST DESCRIPTOR NOT ACCESSIBLE
       00 00
 0008  00 00 00 00 00 00                GDTPTR    DQ     0               ; GDT DESCRIPTOR
       00 00
 0010  00 00 00 00 00 00                IDTPTR    DQ     0               ; IDT DESCRIPTOR
       00 00
 0018  00 00 00 00 00 00                USER_DS   DQ     0               ; USER DATA SEGEMNT DESCRITOR
       00 00
 0020  00 00 00 00 00 00                USER_ES   DQ     0               ; USER EXTRA SEGMENT DESCRIPTOR
       00 00
 0028  00 00 00 00 00 00                USER_SS   DQ     0               ; USER STACK SEGMENT DESCRIPTOR
       00 00
 0030  00 00 00 00 00 00                USER_CS   DQ     0               ; USER CODE SEGMENT DESCRIPTOR
       00 00
 0038  00 00 00 00 00 00                BIO_CS    DQ     0               ; TEMPORARY BIOS DESCRIPTOR
       00 00

 0040                                   VIRTUAL_ENABLE_GDT_DEF  ENDS


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

                                                ASSUME  CS:CODE
                                                ASSUME  DS:DATA

 03E6                                   X_VIRTUAL       PROC    FAR
 03E6                                   SET_VMODE:

 03E6  FA                                       CLI                             ; NO INTERRUPTS ALLOWED

                                        ;------- ENABLE ADDRESS LATCH BIT 20

 03E7  B4 DF                                    MOV     AH,ENABLE_BIT20         ; ENABLE BIT 20 FOR ADDRESS GATE
 03E9  E8 03B0 R                                CALL    GATE_A20                ;
 03EC  3C 00                                    CMP     AL,0                    ; WAS THE COMMAND ACCEPTED?
 03EE  74 04                                    JZ      BIT20_ON                ; GO IF YES
 03F0  B4 FF                                    MOV     AH,0FFH                 ; SET THE ERROR FLAG
 03F2  F9                                       STC                             ; SET CARRY
 03F3  CF                                       IRET                            ; EARLY EXIT

 03F4                                   BIT20_ON:
                                                SEGOV   ES                      ; LOAD THE GLOBAL DESCRIPTOR TABLE REG
 03F4  26                             +         DB      026H
                                                LGDT    [SI].GDTPTR
 03F5  0F                             +         DB      00FH
 03F6                                 + ??0015  LABEL   BYTE
 03F6  8B 54 08                       +         MOV     DX,WORD PTR [SI].GDTPTR
 03F9                                 + ??0016  LABEL   BYTE
 03F6                                 +         ORG     OFFSET CS:??0015
 03F6  01                             +         DB      001H
 03F9                                 +         ORG     OFFSET CS:??0016
                                                SEGOV   ES                      ; LOAD THE INTERUPT DESCRIPTOR TABLE REG
 03F9  26                             +         DB      026H
                                                LIDT    [SI].IDTPTR
 03FA  0F                             +         DB      00FH
 03FB                                 + ??0018  LABEL   BYTE
 03FB  8B 5C 10                       +         MOV     BX,WORD PTR [SI].IDTPTR
 03FE                                 + ??0019  LABEL   BYTE
 03FB                                 +         ORG     OFFSET CS:??0018
 03FB  01                             +         DB      001H
 03FE                                 +         ORG     OFFSET CS:??0019

                                        ;-------------------------------------------------------------------------
                                        ; REINITIALIZE THE 8259 INTERRUPT CONTROLLER #1 TO THE USER SPECIFIED OFFSET  |
                                        ;-------------------------------------------------------------------------

 03FE  B0 11                                    MOV     AL,11H                  ; START INITIALIZATION SEQUENCE-ICW1
 0400  E6 20                                    OUT     INTA00,AL               ; EDGE,INTERVAL-8,MASTER,ICW4 NEEDED
 0402  EB 00                                    JMP     SHORT $+2
 0404  8A C7                                    MOV     AL,BH                   ; HARDWARE INT'S START AT INT # (BH)
 0406  E6 21                                    OUT     INTA01,AL               ; SEND ICW2
 0408  EB 00                                    JMP     SHORT $+2
 040A  B0 04                                    MOV     AL,04H                  ; SEND ICW3 - MASTER LEVEL 2
 040C  E6 21                                    OUT     INTA01,AL               ;
 040E  EB 00                                    JMP     SHORT $+2               ;
 0410  B0 01                                    MOV     AL,01H                  ; SEND ICW4 - MASTER,8086 MODE
 0412  E6 21                                    OUT     INTA01,AL               ;
 0414  EB 00                                    JMP     SHORT $+2               ;
 0416  B0 FF                                    MOV     AL,0FFH                 ; MASK OFF ALL INTERRUPTS
 0418  E6 21                                    OUT     INTA01,AL

                                        ;-------------------------------------------------------------------------
                                        ; REINITIALIZE THE 8259 INTERRUPT CONTROLLER #2 TO THE USER SPECIFIED OFFSET  |
                                        ;-------------------------------------------------------------------------

 041A  B0 11                                    MOV     AL,11H                  ; START INIT SEQUENCE-ICW1 FOR SLAVE
```

SECTION 5

```
041C  E6 A0                          OUT     INTB00,AL               ; EDGE,INTERVAL-8,MASTER,ICW4 NEEDED
041E  EB 00                          JMP     SHORT $+2               ;
0420  8A C3                          MOV     AL,BL                   ; HARDWARE INT'S START AT INT # (BL)
0422  E6 A1                          OUT     INTB01,AL               ; SEND ICW2
0424  B0 02                          MOV     AL,02H                  ;
0426  EB 00                          JMP     SHORT $+2               ;
0428  E6 A1                          OUT     INTB01,AL               ; SEND ICW3 - SLAVE LEVEL 2
042A  EB 00                          JMP     SHORT $+2               ;
042C  B0 01                          MOV     AL,01H                  ;
042E  E6 A1                          OUT     INTB01,AL               ; SEND ICW4 - SLAVE,8086 MODE
0430  EB 00                          JMP     SHORT $+2               ;
0432  B0 FF                          MOV     AL,0FFH                 ;
0434  E6 A1                          OUT     INTB01,AL               ; MASK OFF ALL INTERRUPTS

                           ;-----------------------------------------------------------------------
                           ; SETUP BIOS CODE SEGMENT DESCRIPTOR                                    |
                           ;-----------------------------------------------------------------------

0436  26: C7 44 38 FFFF              MOV     ES:[SI].BIO_CS.SEG_LIMIT,MAX_SEG_LEN     ; SET LENGTH
043C  26: C6 44 3C 0F                MOV     ES:[SI].BIO_CS.BASE_HI_BYTE,CSEG@_HI     ; SET HIGH BYTE OF CS=0F
0441  26: C7 44 3A 0000              MOV     ES:[SI].BIO_CS.BASE_LO_WORD,CSEG@_LO     ; SET LOW WORD OF CS=0
                                                                                      ; SET ACCESS RIGHTS BYTE
0447  26: C6 44 3D 9B                MOV     ES:[SI].BIO_CS.DATA_ACC_RIGHTS,CPL0_CODE_ACCESS
044C  26: C7 44 3E 0000              MOV     ES:[SI].BIO_CS.DATA_RESERVED,0           ; ZERO RESERVED AREA

                           ;-----------------------------------------------------------------------
                           ; ENABLE PROTECTED MODE                                                 |
                           ;-----------------------------------------------------------------------

0452  B8 0001                        MOV     AX,VIRTUAL_ENABLE       ; MACHINE STATUS WORD NEEDED TO
                                     LMSW    AX                      ;  SWITCH TO VIRTUAL MODE
0455  0F                   +         DB      00FH
0456                       + ??001A  LABEL   BYTE
0456  8B F0                +         MOV     SI,AX
0458                       + ??001B  LABEL   BYTE
0456                       +         ORG     OFFSET CS:??001A
0456  01                   +         DB      001H
0458                       +         ORG     OFFSET CS:??001B
                                     JUMPFAR VMODE,BIO_CS           ; MUST PURGE PRE-FETCH QUEUE
0458  EA                   +         DB      0EAH                    ; Jump far direct
0459  045D R               +         DW      (OFFSET VMODE)          ;   to this offset
045B  0038                 +         DW      BIO_CS          ;      in this segment

045D                       VMODE:
                           ;-----------------------------------------------------------------------
                           ; SETUP USER SEGMENT REGISTERS                                          |
                           ;-----------------------------------------------------------------------

045D  B8 0018                        MOV     AX,USER_DS              ; SETUP USER'S DATA SEGMENT
0460  8E D8                          MOV     DS,AX                   ;
0462  B8 0020                        MOV     AX,USER_ES              ; SETUP USER'S EXTRA SEGMENT
0465  8E C0                          MOV     ES,AX                   ;
0467  B8 0028                        MOV     AX,USER_SS              ; SETUP USER'S STACK SEGMENT
046A  8E D0                          MOV     SS,AX                   ;

                           ;-----------------------------------------------------------------------
                           ; PUT TRANSFER ADDRESS ON THE STACK AND RETURN TO THE USER              |
                           ;-----------------------------------------------------------------------

046C  5B                             POP     BX                      ; GET RETURN IP FROM THE STACK
046D  83 C4 04                       ADD     SP,4                    ; NORMALIZE STACK POINTER
                                     IPUSH   USER_CS                 ; SET STACK FOR A RETURN FAR
0470  68                   +         DB      068H
0471  0030                 +         DW      USER_CS
0473  53                             PUSH    BX                      ;
0474  CB                             RET                             ; RETURN TO USER IN VIRTUAL MODE

0475                       X_VIRTUAL         ENDP

                           ;--- DEVICE BUSY AND INTERRUPT COMPLETE -----------------------:
                           ;                                                              :
                           ;         THIS ROUTINE IS A TEMPORY HANDLER FOR DEVICE BUSY    :
                           ;         AND INTERRUPT COMPLETE                               :
                           ;                                                              :
                           ;         INPUT                                                :
                           ;         SEE PROLOG                                           :
                           ;                                                              :
                           ;--------------------------------------------------------------:
0475                       DEVICE_BUSY       PROC    NEAR
0475  F8                             CLC                             ; TURN CARRY OFF
0476  E9 004F R                      JMP     C1_F                    ; RETURN WITH CARRY FLAG
0479                       DEVICE_BUSY       ENDP

0479                       INT_COMPLETE      PROC    NEAR
0479  CF                             IRET                            ; RETURN
047A                       INT_COMPLETE      ENDP


047A                       CODE      ENDS
                                     END
```

```
                         TITLE 08-08-83 BIOS2  BIOS INTERRUPT
                         .LIST
                       C  INCLUDE SEGMENT.SRC
0000                   C  CODE SEGMENT BYTE PUBLIC
                       C
                          EXTRN   DDS:NEAR
                          PUBLIC  TIME_OF_DAY_1,TIMER_INT_1,PRINT_SCREEN_1
                          PUBLIC  RTC_INT
                          ;--- INT 1A -----------------------------------------------------
                          ; TIME_OF_DAY                                                    :
                          ;       THIS ROUTINE ALLOWS THE CLOCK TO BE SET/READ             :
                          ;                                                                :
                          ; INPUT                                                          :
                          ;   (AH) = 0    READ THE CURRENT CLOCK SETTING                   :
                          ;               RETURNS CX = HIGH PORTION OF COUNT               :
                          ;                       DX = LOW PORTION OF COUNT                :
                          ;                       AL = 0 IF TIMER HAS NOT PASSED 24 HOURS  :
                          ;                       SINCE LAST READ. <> 0 IF ON ANOTHER DAY  :
                          ;   (AH) = 1    SET THE CURRENT CLOCK                            :
                          ;       CX = HIGH PORTION OF COUNT                               :
                          ;       DX = LOW PORTION OF COUNT                                :
                          ;                                                                :
                          ; NOTE: COUNTS OCCUR AT THE RATE OF 1193180/65536 COUNTS/SEC     :
                          ;       (OR ABOUT 18.2 PER SECOND -- SEE EQUATES)                :
                          ;                                                                :
                          ;   (AH) = 2    READ THE REAL TIME CLOCK                         :
                          ;               RETURNS CH = HOURS IN BCD                        :
                          ;                       CL = MINUTES IN BCD                      :
                          ;                       DH = SECONDS IN BCD                      :
                          ;                                                                :
                          ;   (AH) = 3    SET THE REAL TIME CLOCK                          :
                          ;       CH = HOURS IN BCD                                        :
                          ;       CL = MINUTES IN BCD                                      :
                          ;       DH = SECONDS IN BCD                                      :
                          ;       DL = 1 IF DAYLIGHT SAVINGS TIME OPTION, ELSE 0           :
                          ;                                                                :
                          ;   (AH) = 4    READ THE DATE FROM THE REAL TIME CLOCK           :
                          ;               RETURNS CH = CENTURY IN BCD (19 OR 20)           :
                          ;                       CL = YEAR IN BCD                         :
                          ;                       DH = MONTH IN BCD                        :
                          ;                       DL = DAY IN BCD                          :
                          ;                                                                :
                          ;   (AH) = 5    SET THE DATE INTO THE REAL TIME CLOCK            :
                          ;       CH = CENTURY IN BCD (19 OR 20)                           :
                          ;       CL = YEAR IN BCD                                         :
                          ;       DH = MONTH IN BCD                                        :
                          ;       DL = DAY IN BCD                                          :
                          ;                                                                :
                          ;   (AH) = 6    SET THE ALARM                                    :
                          ;               THE ALARM CAN BE SET TO INTERRUPT UP TO          :
                          ;               23:59:59 FROM PRESENT TIME.                      :
                          ;               ONE ALARM FUNCTION MAY BE ACTIVE AT ANY TIME     :
                          ;                                                                :
                          ;       CH = HOURS IN BCD                                        :
                          ;       CL = MINUTES IN BCD                                      :
                          ;       DH = SECONDS IN BCD                                      :
                          ;                                                                :
                          ;   (AH) = 7    RESET THE ALARM                                  :
                          ;                                                                :
                          ; NOTE: FOR AH = 2, 4, 6 - CY FLAG SET IF CLOCK NOT OPERATING    :
                          ;       FOR AH = 6 - CY FLAG SET IF ALARM ALREADY ENABLED        :
                          ; NOTE: FOR THE ALARM FUNCTION (AH = 6) THE USER MUST CODE A     :
                          ;       ROUTINE AND PLACE THE CORRECT ADDRESS IN THE VECTOR      :
                          ;       TABLE FOR INT 4AH                                        :
                          ;----------------------------------------------------------------
                                  ASSUME  CS:CODE,DS:DATA

0000                      TIME_OF_DAY_1   PROC    FAR
0000  FB                          STI                         ; INTERRUPTS BACK ON
0001  1E                          PUSH    DS                  ; SAVE SEGMENT
0002  E8 0000 E                   CALL    DDS                 ; SET DATA SEGMENT
0005  0A E4                       OR      AH,AH               ; AH=0
0007  74 14                       JZ      T2                  ; READ_TIME
0009  FE CC                       DEC     AH                  ; AH=1
000B  74 23                       JZ      T3                  ; SET_TIME
000D  80 FC 07                    CMP     AH,7                ; CHECK IF VALID
0010  7D 03                       JGE     T1                  ; RETURN IF NOT VALID
0012  EB 2C 90                    JMP     RTC_0               ; GO CHECK OTHER FUNCTIONS
0015                      T1:                                 ; TOD_RETURN
0015  FB                          STI                         ; INTERRUPTS BACK ON
0016  1F                          POP     DS                  ; RECOVER SEGMENT
0017  CF                          IRET                        ; RETURN TO CALLER

0018                      T1_A:
0018  F9                          STC                         ; SET ERROR RETURN
0019  1F                          POP     DS
001A  CA 0002                     RET     2

001D                      T2:                                 ; READ_TIME
001D  FA                          CLI                         ; NO TIMER INTERRUPTS WHILE READING
001E  A0 0070 R                   MOV     AL,TIMER_OFL
0021  C6 06 0070 R 00             MOV     TIMER_OFL,0         ; GET OVERFLOW, AND RESET THE FLAG
0026  8B 0E 006E R                MOV     CX,TIMER_HIGH
002A  8B 16 006C R                MOV     DX,TIMER_LOW
002E  EB E5                       JMP     T1                  ; TOD_RETURN

0030                      T3:                                 ; SET_TIME
0030  FA                          CLI                         ; NO INTERRUPTS WHILE WRITING
0031  89 16 006C R                MOV     TIMER_LOW,DX
0035  89 0E 006E R                MOV     TIMER_HIGH,CX       ; SET THE TIME
0039  C6 06 0070 R 00             MOV     TIMER_OFL,0         ; RESET OVERFLOW
003E  EB D5                       JMP     T1                  ; TOD_RETURN

0040                      RTC_0:
0040  FE CC                       DEC     AH                  ; AH = 2
0042  74 07                       JZ      RTC_2               ; READ RTC TIME
0044  FE CC                       DEC     AH                  ; AH = 3
0046  74 26                       JZ      RTC_3               ; SET RTC TIME
0048  E9 00D7 R                   JMP     RTC_1               ; GO CHECK REMAINING FUNCTIONS
004B                      RTC_GET_TIME    PROC    NEAR

004B                      RTC_2:
004B  E8 01B7 R                   CALL    UPD_IN_PR           ; CHECK FOR UPDATE IN PROCESS
004E  73 02                       JNC     RTC_2A              ; GO AROUND IF OK
0050  EB C6                       JMP     T1_A                ; RETURN IF ERROR
0052                      RTC_2A:
0052  FA                          CLI                         ; INTERRUPTS OFF DURING READ
0053  B2 FE                       MOV     DL,-2
0055  E8 0192 R                   CALL    PORT_INC_2          ; SET ADDRESS OF SECONDS
0058  E4 71                       IN      AL,CMOS_PORT+1
005A  8A F0                       MOV     DH,AL               ; SAVE
005C  E8 0192 R                   CALL    PORT_INC_2          ; SET ADDRESS OF MINUTES
005F  E4 71                       IN      AL,CMOS_PORT+1      ;
0061  8A C8                       MOV     CL,AL               ; SAVE
0063  E8 0192 R                   CALL    PORT_INC_2          ; SET ADDRESS OF HOURS
0066  E4 71                       IN      AL,CMOS_PORT+1      ;
```

SECTION 5

```
0068  8A E8                    MOV      CH,AL                  ; SAVE
006A  B2 00                    MOV      DL,0                   ; SET DL TO ZERO
006C  EB A7                    JMP      T1                     ; RETURN
006E                RTC_GET_TIME ENDP
                    ;
006E                RTC_SET_TIME PROC   NEAR
006E                RTC_3:
006E  E8 01B7 R                CALL     UPD_IN_PR              ; CHECK FOR UPDATE IN PROCESS
0071  73 03                    JNC      RTC_3A                 ; GO AROUND IF CLOCK OPERATING
0073  E8 019A R                CALL     INITIALIZE_STATUS
0076                RTC_3A:
0076  FA                       CLI                             ; INTERRUPTS OFF DURING SET
0077  52                       PUSH     DX                     ; SAVE
0078  B2 FE                    MOV      DL,-2                  ; FIRST ADDRESS
007A  E8 0192 R                CALL     PORT_INC_2             ; UPDATE ADDRESS
007D  8A C6                    MOV      AL,DH                  ; GET TIME BYTE - SECONDS
007F  E6 71                    OUT      CMOS_PORT+1,AL         ; STORE TIME BYTE
0081  E8 0192 R                CALL     PORT_INC_2             ; UPDATE ADDRESS
0084  8A C1                    MOV      AL,CL                  ; GET TIME BYTE - MINUTES
0086  E6 71                    OUT      CMOS_PORT+1,AL         ; STORE TIME BYTE
0088  E8 0192 R                CALL     PORT_INC_2             ; UPDATE ADDRESS
008B  8A C5                    MOV      AL,CH                  ; GET TIME BYTE - HOURS
008D  E6 71                    OUT      CMOS_PORT+1,AL         ; STORE TIME BYTE
008F  B2 0A                    MOV      DL,0AH
0091  E8 018B R                CALL     PORT_INC
0094  5A                       POP      DX                     ; RESTORE
0095  E4 71                    IN       AL,CMOS_PORT+1         ; GET CURRENT VALUE
0097  24 23                    AND      AL,23H                 ; MASK FOR VALID BIT POSITIONS
0099  0A C2                    OR       AL,DL                  ; GET DST BIT
009B  0C 02                    OR       AL,02H                 ; TURN ON 24 HR MODE
009D  50                       PUSH     AX                     ;
009E  B2 0A                    MOV      DL,0AH                 ;
00A0  E8 018B R                CALL     PORT_INC               ;
00A3  58                       POP      AX                     ;
00A4  E6 71                    OUT      CMOS_PORT+1,AL
00A6  E9 0015 R                JMP      T1            ;DONE
00A9                RTC_SET_TIME ENDP

00A9                RTC_GET_DATE PROC   NEAR
00A9                RTC_4:
00A9  E8 01B7 R                CALL     UPD_IN_PR
00AC  73 03                    JNC      RTC_4A
00AE  E9 0018 R                JMP      T1_A                   ; RETURN ON ERROR
00B1                RTC_4A:
00B1  FA                       CLI                             ; INTERRUPTS OFF DURING READ
00B2  B2 06                    MOV      DL,6
00B4  E8 018B R                CALL     PORT_INC               ; POINT TO DAY
00B7  E4 71                    IN       AL,CMOS_PORT+1
00B9  8A E8                    MOV      CH,AL                  ; SAVE
00BB  E8 018B R                CALL     PORT_INC               ; POINT TO MONTH
00BE  E4 71                    IN       AL,CMOS_PORT+1
00C0  8A F0                    MOV      DH,AL                  ; SAVE
00C2  E8 018B R                CALL     PORT_INC               ; POINT TO YEAR
00C5  E4 71                    IN       AL,CMOS_PORT+1
00C7  8A C8                    MOV      CL,AL                  ; SAVE
00C9  B2 31                    MOV      DL,31H                 ; POINT TO CENTURY BYTE SAVE AREA
00CB  E8 018B R                CALL     PORT_INC               ;
00CE  E4 71                    IN       AL,CMOS_PORT+1         ; GET VALUE
00D0  8A D5                    MOV      DL,CH                  ; GET DAY BACK
00D2  8A E8                    MOV      CH,AL                  ;
00D4  E9 0015 R                JMP      T1                     ; FINISHED
00D7                RTC_GET_DATE ENDP
00D7                RTC_1:
00D7  FE CC                    DEC      AH                     ; AH = 4
00D9  74 CE                    JZ       RTC_4                  ; READ RTC DATE
00DB  FE CC                    DEC      AH                     ; AH = 5
00DD  74 07                    JZ       RTC_5                  ; SET RTC DATE
00DF  FE CC                    DEC      AH                     ; AH = 6
00E1  74 45                    JZ       RTC_6                  ; SET RTC ALARM
00E3  E9 0175 R                JMP      RTC_7                  ; RESET RTC ALARM

00E6                RTC_SET_DATE PROC   NEAR
00E6                RTC_5:
00E6  E8 01B7 R                CALL     UPD_IN_PR              ; CHECK FOR UPDATE IN PROCESS
00E9  73 03                    JNC      RTC_5A                 ; GO AROUND IF CLOCK UPDATING
00EB  E8 019A R                CALL     INITIALIZE_STATUS
00EE                RTC_5A:
00EE  FA                       CLI                             ; INTERRUPTS OFF DURING SET
00EF  51                       PUSH     CX                     ; SAVE
00F0  8A EA                    MOV      CH,DL                  ; SAVE DAY OF MONTH
00F2  B2 05                    MOV      DL,5                   ; ADDRESS OF DAY OF WEEK REGISTER
00F4  E8 018B R                CALL     PORT_INC
00F7  B0 00                    MOV      AL,00H
00F9  E6 71                    OUT      CMOS_PORT+1,AL         ; LOAD ZEROS TO 'DAY OF WEEK' BYTE
00FB  E8 018B R                CALL     PORT_INC               ; ADDRESS OF DAY OF MONTH REGISTER
00FE  8A C5                    MOV      AL,CH                  ; GET DAY OF MONTH BYTE
0100  E6 71                    OUT      CMOS_PORT+1,AL         ; STORE IT
0102  E8 018B R                CALL     PORT_INC               ; ADDRESS MONTH REGISTER
0105  8A C6                    MOV      AL,DH                  ; GET MONTH BYTE
0107  E6 71                    OUT      CMOS_PORT+1,AL         ; STORE IT
0109  E8 018B R                CALL     PORT_INC               ; ADDRESS OF YEAR REGISTER
010C  8A C1                    MOV      AL,CL                  ; GET YEAR BYTE
010E  E6 71                    OUT      CMOS_PORT+1,AL         ; STORE IT
0110  B2 0A                    MOV      DL,0AH
0112  E8 018B R                CALL     PORT_INC
0115  E4 71                    IN       AL,CMOS_PORT+1         ; GET CURRENT SETING
0117  24 7F                    AND      AL,07FH                ; CLEAR 'SET BIT'
0119  E6 71                    OUT      CMOS_PORT+1,AL         ; *AND START CLOCK UPDATING
011B  59                       POP      CX                     ; GET BACK
011C  B2 31                    MOV      DL,31H                 ; POINT TO SAVE AREA
011E  E8 018B R                CALL     PORT_INC
0121  8A C5                    MOV      AL,CH                  ; GET CENTURY BYTE
0123  E6 71                    OUT      CMOS_PORT+1,AL         ; SAVE IT
0125  E9 0015 R                JMP      T1                     ; RETURN
0128                RTC_SET_DATE ENDP

0128                RTC_SET_ALARM PROC  NEAR
0128                RTC_6:
0128  B2 0A                    MOV      DL,0AH                 ; CHECK FOR ALARM ALREADY ENABLED
012A  E8 018B R                CALL     PORT_INC               ;
012D  E4 71                    IN       AL,CMOS_PORT+1         ; GET CURRENT SETTING OF ALARM ENABLE
012F  A8 20                    TEST     AL,20H                 ;
0131  74 05                    JZ       RTC_6A                 ; ALARM NOT SET - GO PROCESS
0133  33 C0                    XOR      AX,AX                  ;
0135  E9 0018 R                JMP      T1_A                   ; RETURN IF ERROR
0138                RTC_6A:
0138  E8 01B7 R                CALL     UPD_IN_PR              ; CHECK FOR UPDATE IN PROCESS
013B  73 03                    JNC      RTC_6B                 ;
013D  E8 019A R                CALL     INITIALIZE_STATUS
0140                RTC_6B:
0140  FA                       CLI                             ; INTERRUPTS OFF DURING SET
0141  B2 FF                    MOV      DL,-1
0143  E8 0192 R                CALL     PORT_INC_2
0146  8A C6                    MOV      AL,DH                  ; GET SECONDS BYTE
0148  E6 71                    OUT      CMOS_PORT+1,AL         ; LOAD ALARM BYTE - SECONDS
```

```
014A  E8 0192 R                  CALL    PORT_INC_2
014D  8A C1                      MOV     AL,CL                ; GET MINUTES PARAMETER
014F  E6 71                      OUT     CMOS_PORT+1,AL       ; LOAD ALARM BYTE - MINUTES
0151  E8 0192 R                  CALL    PORT_INC_2
0154  8A C5                      MOV     AL,CH                ; GET HOURS PARAMETER
0156  E6 71                      OUT     CMOS_PORT+1,AL       ; LOAD ALARM BYTE - HOURS
0158  E4 A1                      IN      AL,0A1H              ;ENSURE INTERRUPT UNMASKED
015A  24 FE                      AND     AL,0FEH              ;
015C  E6 A1                      OUT     0A1H,AL              ;
015E  B2 0A                      MOV     DL,0AH
0160  E8 018B R                  CALL    PORT_INC
0163  E4 71                      IN      AL,CMOS_PORT+1       ; GET CURRENT VALUE
0165  24 7F                      AND     AL,07FH              ; ENSURE SET BIT TURNED OFF
0167  0C 20                      OR      AL,20H               ; TURN ON ALARM ENABLE
0169  50                         PUSH    AX                   ;
016A  B2 0A                      MOV     DL,0AH               ;
016C  E8 018B R                  CALL    PORT_INC             ;
016F  58                         POP     AX                   ;
0170  E6 71                      OUT     CMOS_PORT+1,AL       ; ENABLE ALARM
0172  E9 0015 R                  JMP     T1
0175                     RTC_SET_ALARM   ENDP

0175                     RTC_RESET_ALARM PROC    NEAR
0175                     RTC_7:
0175  FA                         CLI                          ; INTERRUPTS MASKED DURING RESET
0176  B2 0A                      MOV     DL,0AH
0178  E8 018B R                  CALL    PORT_INC
017B  E4 71                      IN      AL,CMOS_PORT+1       ; GET STATUS BYTE
017D  24 57                      AND     AL,57H               ; TURN OFF ALARM ENABLE
017F  50                         PUSH    AX                   ; SAVE
0180  B2 0A                      MOV     DL,0AH               ;
0182  E8 018B R                  CALL    PORT_INC             ;
0185  58                         POP     AX                   ;
0186  E6 71                      OUT     CMOS_PORT+1,AL       ; RESTORE
0188  E9 0015 R                  JMP     T1
018B                     RTC_RESET_ALARM ENDP

018B                     RTC_TIMEBIOS_SUBR       PROC    NEAR
018B                     PORT_INC:
018B  FE C2                      INC     DL                   ; INCREMENT ADDRESS
018D  8A C2                      MOV     AL,DL
018F  E6 70                      OUT     CMOS_PORT,AL
0191  C3                         RET
                         ;
0192                     PORT_INC_2:
0192  80 C2 02                   ADD     DL,2                 ; INCREMENT ADDRESS
0195  8A C2                      MOV     AL,DL
0197  E6 70                      OUT     CMOS_PORT,AL
0199  C3                         RET
                         ;
019A                     INITIALIZE_STATUS       PROC    NEAR
                         ;
019A  52                         PUSH    DX                   ; SAVE
019B  B2 09                      MOV     DL,09H
019D  E8 018B R                  CALL    PORT_INC
01A0  B0 26                      MOV     AL,26H
01A2  E6 71                      OUT     CMOS_PORT+1,AL       ; INITIALIZE 'A' REGISTER
01A4  E8 018B R                  CALL    PORT_INC
01A7  B0 82                      MOV     AL,82H               ; SET 'SET BIT' FOR CLOCK INITIALIZATION
                                                              ; AND 24 HOUR MODE
01A9  E6 71                      OUT     CMOS_PORT+1,AL       ; INITIALIZE 'B' REGISTER
01AB  E8 018B R                  CALL    PORT_INC
01AE  E4 71                      IN      AL,CMOS_PORT+1       ; READ REGISTER 'C' TO INITIALIZE
01B0  E8 018B R                  CALL    PORT_INC
01B3  E4 71                      IN      AL,CMOS_PORT+1       ; READ REGISTER 'D' TO INITIALIZE
01B5  5A                         POP     DX                   ; RESTORE
01B6  C3                         RET
                         ;
01B7                     INITIALIZE_STATUS       ENDP
                         ;
01B7                     UPD_IN_PR:
01B7  51                         PUSH    CX
01B8  B9 0258                    MOV     CX,600               ; SET LOOP COUNT
01BB                     UPDATE:
01BB  B0 0A                      MOV     AL,0AH               ; ADDRESS OF `A` REGISTER
01BD  E6 70                      OUT     CMOS_PORT,AL
01BF  EB 00                      JMP     $+2                  ; I/O TIME DELAY
01C1  E4 71                      IN      AL,CMOS_PORT+1       ; READ IN REGISTER 'A'
01C3  A8 80                      TEST    AL,80H               ; IF 8XH--> UIP BIT IS ON (CANNOT READ TIM
01C5  74 05                      JZ      UPD_IN_PREND
01C7  E2 F2                      LOOP    UPDATE
01C9  33 C0                      XOR     AX,AX                ;
01CB  F9                         STC                          ; SET CARRY FOR ERROR
01CC                     UPD_IN_PREND:
01CC  59                         POP     CX
01CD  C3                         RET                          ; RETURN
                         ;
01CE                     RTC_TIMEBIOS_SUBR       ENDP
01CE                     TIME_OF_DAY_1   ENDP
                         PAGE
                         ;--INT 50 (LEVEL 8)-------------------------------------------:
                         ; THIS ROUTINE HANDLES THE PERIODIC AND ALARM INTERRUPTS FROM   :
                         ; THE NON-VOLATILE TIMER.  INPUT FREQUENCY IS 1.024 KHZ         :
                         ; OR APPROXIMATELY 1024 INTERRUPTS EVERY SECOND FOR THE         :
                         ; PERIODIC INTERRUPT.  FOR THE ALARM FUNCTION, AN INTERRUPT WILL:
                         ; OCCUR AT THE DESIGNATED TIME.                                 :
                         ;                                                               :
                         ; THE INTERRUPT IS ENABLED ONLY WHEN EVENT OR ALARM FUNCTIONS   :
                         ; ARE ACTIVE.                                                   :
                         ; FOR THE EVENT INTERRUPT, THE HANDLER WILL DECREMENT THE       :
                         ; WAIT COUNTER AND WHEN IT EXPIRES WILL TURN ON THE HIGH ORDER  :
                         ; BIT OF THE DESIGNATED FLAG.                                   :
                         ; FOR THE ALARM INTERRUPT, THE USER ROUTINE WILL BE INVOKED     :
                         ; THROUGH INT 4AH.  THE USER MUST CODE A ROUTINE AND PLACE THE  :
                         ; CORRECT ADDRESS IN THE VECTOR TABLE.                          :
                         ;---------------------------------------------------------------:
01CE                     RTC_INT  PROC     FAR
01CE  FB                         STI                  ; INTERRUPTS BACK ON
01CF  1E                         PUSH    DS           ; SAVE REGISTERS
01D0  50                         PUSH    AX           ;
01D1  52                         PUSH    DX           ;
01D2  57                         PUSH    DI           ;
01D3  B2 0A                      MOV     DL,0AH       ; GET ENABLES
01D5  E8 018B R                  CALL    PORT_INC     ;
01D8  E4 71                      IN      AL,CMOS_PORT+1 ;
01DA  8A E0                      MOV     AH,AL        ; SAVE
01DC  E8 018B R                  CALL    PORT_INC     ; GET SOURCE
01DF  E4 71                      IN      AL,CMOS_PORT+1 ;
01E1  22 C4                      AND     AL,AH        ;
01E3  50                         PUSH    AX           ; SAVE
01E4  A8 40                      TEST    AL,040H      ; CHECK FOR PERIODIC INTERRUPT
01E6  74 2E                      JZ      RTC_INT_9    ; NO - GO AROUND
01E8  E8 0000 E                  CALL    DDS          ; ESTABLISH ADDRESSABILITY
01EB  81 2E 009C R 03D0          SUB     RTC_LOW,0976 ; DECREMENT COUNT
01F1  83 1E 009E R 00            SBB     RTC_HIGH,0   ;
01F6  77 1E                      JA      RTC_INT_9    ;
```

SECTION 5

```
01F8  B2 0A                       MOV    DL,0AH           ; TURN OFF PIE
01FA  E8 018B R                   CALL   PORT_INC         ;
01FD  E4 71                       IN     AL,CMOS_PORT+1   ;
01FF  24 BF                       AND    AL,0BFH          ;
0201  50                          PUSH   AX               ;
0202  B2 0A                       MOV    DL,0AH           ;
0204  E8 018B R                   CALL   PORT_INC         ;
0207  58                          POP    AX               ;
0208  E6 71                       OUT    CMOS_PORT+1,AL   ;
020A  C6 06 00A0 R 00             MOV    RTC_WAIT_FLAG,0 ; SET FUNCTION ACTIVE FLAG OFF
020F  C5 3E 0098 R                LDS    DI,DWORD PTR USER_FLAG ; SET UP DS,DI TO POINT TO USER FLAG
0213  C6 05 80                    MOV    BYTE PTR[DI],80H ; TURN ON USERS FLAG

0216                       RTC_INT_9:
0216  58                          POP    AX               ; GET INTERRUPT SOURCE BACK
0217  A8 20                       TEST   AL,20H           ; TEST FOR ALARM INTERRUPT
0219  74 02                       JZ     RTC_INT_10       ; NO - GO AROUND
021B  CD 4A                       INT    4AH              ; TRANSFER TO USER ROUTINE
021D                       RTC_INT_10:
021D  B0 20                       MOV    AL,EOI           ; END OF INTERRUPT TO 8259 - 2
021F  E6 A0                       OUT    0A0H,AL
0221  E6 20                       OUT    020H,AL          ; AND TO 8259 - 1
0223  5F                          POP    DI               ; RESTORE REGISTERS
0224  5A                          POP    DX               ;
0225  58                          POP    AX               ;
0226  1F                          POP    DS               ;
0227  CF                          IRET                    ; END OF INTERRUPT
0228                       RTC_INT ENDP
                           PAGE
                           ;-- INT 8 (LEVEL 0)------------------------------------------:
                           ; THIS ROUTINE HANDLES THE TIMER INTERRUPT FROM              :
                           ; CHANNEL 0 OF THE 8253 TIMER.  INPUT FREQUENCY IS 1.19318 MHZ :
                           ; AND THE DIVISOR IS 65536, RESULTING IN APPROX. 18.2 INTERRUPTS:
                           ; EVERY SECOND.                                              :
                           ;                                                            :
                           ; THE INTERRUPT HANDLER MAINTAINS A COUNT OF INTERRUPTS SINCE :
                           ; POWER ON TIME, WHICH MAY BE USED TO ESTABLISH TIME OF DAY.  :
                           ; THE INTERRUPT HANDLER ALSO DECREMENTS THE MOTOR CONTROL COUNT :
                           ; OF THE DISKETTE, AND WHEN IT EXPIRES, WILL TURN OFF THE     :
                           ; DISKETTE MOTOR(s), AND RESET THE MOTOR RUNNING FLAGS.      :
                           ; THE INTERRUPT HANDLER WILL ALSO INVOKE A USER ROUTINE THROUGH :
                           ; INTERRUPT 1CH AT EVERY TIME TICK.  THE USER MUST CODE A     :
                           ; ROUTINE AND PLACE THE CORRECT ADDRESS IN THE VECTOR TABLE.  :
                           ;------------------------------------------------------------:

0228                       TIMER_INT_1     PROC    FAR     ;
0228  FB                          STI                     ; INTERRUPTS BACK ON
0229  1E                          PUSH   DS
022A  50                          PUSH   AX
022B  52                          PUSH   DX               ; SAVE MACHINE STATE
022C  E8 0000 E                   CALL   DDS              ; ESTABLISH ADDRESSABILITY
022F  FF 06 006C R                INC    TIMER_LOW        ; INCREMENT TIME
0233  75 04                       JNZ    T4               ; TEST_DAY
0235  FF 06 006E R                INC    TIMER_HIGH       ; INCREMENT HIGH WORD OF TIME
0239                       T4:                            ; TEST_DAY
0239  83 3E 006E R 18             CMP    TIMER_HIGH,018H ; TEST FOR COUNT EQUALLING 24 HOURS
023E  75 15                       JNZ    T5               ; DISKETTE_CTL
0240  81 3E 006C R 00B0           CMP    TIMER_LOW,0B0H
0246  75 0D                       JNZ    T5               ; DISKETTE_CTL

                           ;------ TIMER HAS GONE 24 HOURS

0248  2B C0                       SUB    AX,AX
024A  A3 006E R                   MOV    TIMER_HIGH,AX
024D  A3 006C R                   MOV    TIMER_LOW,AX
0250  C6 06 0070 R 01             MOV    TIMER_OFL,1

                           ;------ TEST FOR DISKETTE TIME OUT

0255                       T5:                            ; DISKETTE_CTL
0255  FE 0E 0040 R                DEC    MOTOR_COUNT
0259  75 0B                       JNZ    T6               ; RETURN IF COUNT NOT OUT
025B  80 26 003F R F0             AND    MOTOR_STATUS,0F0H        ; TURN OFF MOTOR RUNNING BITS
0260  B0 0C                       MOV    AL,0CH
0262  BA 03F2                     MOV    DX,03F2H                 ; FDC CTL PORT
0265  EE                          OUT    DX,AL            ; TURN OFF THE MOTOR

0266                       T6:                            ; TIMER_RET:
0266  CD 1C                       INT    1CH              ; TRANSFER CONTROL TO A USER ROUTINE
0268  B0 20                       MOV    AL,EOI
026A  E6 20                       OUT    020H,AL          ; END OF INTERRUPT TO 8259
026C  5A                          POP    DX
026D  58                          POP    AX
026E  1F                          POP    DS               ; RESET MACHINE STATE
026F  CF                          IRET                    ; RETURN FROM INTERRUPT
0270                       TIMER_INT_1     ENDP

                           ;-- INT 5 -----------------------------------------------------------
                           ;      THIS LOGIC WILL BE INVOKED BY INTERRUPT 05H TO PRINT
                           ;      THE SCREEN. THE CURSOR POSITION AT THE TIME THIS ROUTINE
                           ;      IS INVOKED WILL BE SAVED AND RESTORED UPON COMPLETION. THE
                           ;      ROUTINE IS INTENDED TO RUN WITH INTERRUPTS ENABLED.
                           ;      IF A SUBSEQUENT 'PRINT SCREEN KEY IS DEPRESSED DURING THE
                           ;      TIME THIS ROUTINE IS PRINTING IT WILL BE IGNORED.
                           ;      ADDRESS 50:0 CONTAINS THE STATUS OF THE PRINT SCREEN:
                           ;
                           ;      50:0    =0      EITHER PRINT SCREEN HAS NOT BEEN CALLED
                           ;                      OR UPON RETURN FROM A CALL THIS INDICATES
                           ;                      A SUCCESSFUL OPERATION.
                           ;
                           ;              =1      PRINT SCREEN IS IN PROGRESS
                           ;
                           ;              =255    ERROR ENCOUNTERED DURING PRINTING
                           ;---------------------------------------------------------
                                  ASSUME  CS:CODE,DS:XXDATA


0270                       PRINT_SCREEN_1  PROC    FAR     ;
0270  FB                          STI                     ; MUST RUN WITH INTERRUPTS ENABLED
0271  1E                          PUSH   DS               ; MUST USE 50:0 FOR DATA AREA STORAGE
0272  50                          PUSH   AX
0273  53                          PUSH   BX
0274  51                          PUSH   CX               ; WILL USE THIS LATER FOR CURSOR LIMITS
0275  52                          PUSH   DX               ; WILL HOLD CURRENT CURSOR POSITION
0276  B8 ---- R                   MOV    AX,XXDATA        ; HEX 50
0279  8E D8                       MOV    DS,AX
027B  80 3E 0000 R 01             CMP    STATUS_BYTE,1    ; SEE IF PRINT ALREADY IN PROGRESS
0280  74 5F                       JZ     EXIT             ; JUMP IF PRINT ALREADY IN PROGRESS
0282  C6 06 0000 R 01             MOV    STATUS_BYTE,1    ; INDICATE PRINT NOW IN PROGRESS
0287  B4 0F                       MOV    AH,15            ; WILL REQUEST THE CURRENT SCREEN MODE
0289  CD 10                       INT    10H              ;       [AL]=MODE
                                                          ;       [AH]=NUMBER COLUMNS/LINE
                                                          ;       [BH]=VISUAL PAGE
                                  ; *************************************************************
```

```
                                    ;       AT THIS POINT WE KNOW THE COLUMNS/LINE ARE IN
                                    ;       [AX] AND THE PAGE IF APPLICABLE IS IN [BH]. THE STACK
                                    ;       HAS DS,AX,BX,CX,DX PUSHED. [AL] HAS VIDEO MODE
                                    ;
                                    ; ********************************************************
028B  8A CC                         MOV     CL,AH           ; WILL MAKE USE OF [CX] REGISTER TO
028D  B5 19                         MOV     CH,25           ; CONTROL ROW & COLUMNS
028F  E8 02E7 R                     CALL    CRLF            ; CARRIAGE RETURN LINE FEED ROUTINE
0292  51                            PUSH    CX              ; SAVE SCREEN BOUNDS
0293  B4 03                         MOV     AH,3            ; WILL NOW READ THE CURSOR.
0295  CD 10                         INT     10H             ; AND PRESERVE THE POSITION
0297  59                            POP     CX              ; RECALL SCREEN BOUNDS
0298  52                            PUSH    DX              ; RECALL [BH]=VISUAL PAGE
0299  33 D2                         XOR     DX,DX           ; WILL SET CURSOR POSITION TO [0,0]
                            ; ********************************************************
                            ;       THE LOOP FROM PRI10 TO THE INSTRUCTION PRIOR TO PRI20
                            ;       IS THE LOOP TO READ EACH CURSOR POSITION FROM THE SCREEN
                            ;       AND PRINT.
                            ; ********************************************************
029B  B4 02                 PRI10:  MOV     AH,2            ; TO INDICATE CURSOR SET REQUEST
029D  CD 10                         INT     10H             ; NEW CURSOR POSITION ESTABLISHED
029F  B4 08                         MOV     AH,8            ; TO INDICATE READ CHARACTER
02A1  CD 10                         INT     10H             ; CHARACTER NOW IN [AL]
02A3  0A C0                         OR      AL,AL           ; SEE IF VALID CHAR
02A5  75 02                         JNZ     PRI15           ; JUMP IF VALID CHAR
02A7  B0 20                         MOV     AL,' '          ; MAKE A BLANK
02A9                        PRI15:
02A9  52                            PUSH    DX              ; SAVE CURSOR POSITION
02AA  33 D2                         XOR     DX,DX           ; INDICATE PRINTER 1
02AC  32 E4                         XOR     AH,AH           ; TO INDICATE PRINT CHAR IN [AL]
02AE  CD 17                         INT     17H             ; PRINT THE CHARACTER
02B0  5A                            POP     DX              ; RECALL CURSOR POSITION
02B1  F6 C4 29                      TEST    AH, 29H         ; TEST FOR PRINTER ERROR
02B4  75 21                         JNZ     ERR10           ; JUMP IF ERROR DETECTED
02B6  FE C2                         INC     DL              ; ADVANCE TO NEXT COLUMN
02B8  3A CA                         CMP     CL,DL           ; SEE IF AT END OF LINE
02BA  75 DF                         JNZ     PRI10           ; IF NOT PROCEED
02BC  32 D2                         XOR     DL,DL           ; BACK TO COLUMN 0
02BE  8A E2                         MOV     AH,DL           ; [AH]=0
02C0  52                            PUSH    DX              ; SAVE NEW CURSOR POSITION
02C1  E8 02E7 R                     CALL    CRLF            ; LINE FEED CARRIAGE RETURN
02C4  5A                            POP     DX              ; RECALL CURSOR POSITION
02C5  FE C6                         INC     DH              ; ADVANCE TO NEXT LINE
02C7  3A EE                         CMP     CH,DH           ; FINISHED?
02C9  75 D0                         JNZ     PRI10           ; IF NOT CONTINUE
02CB  5A                    PRI20:  POP     DX              ; RECALL CURSOR POSITION
02CC  B4 02                         MOV     AH,2            ; TO INDICATE CURSOR SET REQUEST
02CE  CD 10                         INT     10H             ; CURSOR POSITION RESTORED
02D0  C6 06 0000 R 00               MOV     STATUS_BYTE,0   ; INDICATE FINISHED
02D5  EB 0A                         JMP     SHORT EXIT      ; EXIT THE ROUTINE
02D7  5A                    ERR10:  POP     DX              ; GET CURSOR POSITION
02D8  B4 02                         MOV     AH,2            ; TO REQUEST CURSOR SET
02DA  CD 10                         INT     10H             ; CURSOR POSITION RESTORED
02DC  C6 06 0000 R FF       ERR20:  MOV     STATUS_BYTE,0FFH        ; INDICATE ERROR

02E1  5A                    EXIT:   POP     DX              ; RESTORE ALL THE REGISTERS USED
02E2  59                            POP     CX
02E3  5B                            POP     BX
02E4  58                            POP     AX
02E5  1F                            POP     DS
02E6  CF                            IRET
02E7                        PRINT_SCREEN_1  ENDP

                            ;------ CARRIAGE RETURN, LINE FEED SUBROUTINE

02E7                        CRLF    PROC    NEAR
02E7  33 D2                         XOR     DX,DX           ; PRINTER 0
02E9  32 E4                         XOR     AH,AH           ; WILL NOW SEND INITIAL LF,CR TO PRINTER
02EB  B0 0A                         MOV     AL,12Q          ; LF
02ED  CD 17                         INT     17H             ; SEND THE LINE FEED
02EF  32 E4                         XOR     AH,AH           ; NOW FOR THE CR
02F1  B0 0D                         MOV     AL,15Q          ; CR
02F3  CD 17                         INT     17H             ; SEND THE CARRIAGE RETURN
02F5  C3                            RET
02F6                        CRLF    ENDP
02F6                        CODE    ENDS
                                    END
```

SECTION 5

```
                      TITLE 12/08/83 ORGS
                      .LIST
                    C INCLUDE SEGMENT.SRC
0000                C CODE SEGMENT BYTE PUBLIC
                    C
                      ASSUME CS:CODE, DS:DATA

                      EXTRN   K16:NEAR
                      EXTRN   INT_287:NEAR
                      EXTRN   DSKETTE_SETUP:NEAR
                      EXTRN   DISK_SETUP:NEAR
                      EXTRN   SEEK:NEAR
                      EXTRN   RTC_INT:NEAR
                      EXTRN   START_1:NEAR
                      EXTRN   NMI_INT_1:NEAR
                      EXTRN   BOOT_STRAP_1:NEAR
                      EXTRN   KEYBOARD_IO_1:NEAR
                      EXTRN   KB_INT_1:NEAR
                      EXTRN   DISKETTE_IO_1:NEAR
                      EXTRN   DISK_INT_1:NEAR
                      EXTRN   PRINTER_IO_1:NEAR
                      EXTRN   VIDEO_IO_1:NEAR
                      EXTRN   MEMORY_SIZE_DETERMINE_1:NEAR
                      EXTRN   EQUIPMENT_1:NEAR
                      EXTRN   CASSETTE_IO_1:NEAR
                      EXTRN   TIME_OF_DAY_1:NEAR
                      EXTRN   TIMER_INT_1:NEAR
                      EXTRN   D11:NEAR
                      EXTRN   RS232_IO_1:NEAR
                      EXTRN   DUMMY_RETURN_1:NEAR
                      EXTRN   PRINT_SCREEN_1:NEAR
                      EXTRN   C11:NEAR
                      EXTRN   C30:NEAR
                      EXTRN   TST4_B:NEAR
                      EXTRN   TST4_C:NEAR
                      EXTRN   TST4_D:NEAR
                      EXTRN   E30B:NEAR
                      EXTRN   E30C:NEAR
                      EXTRN   RE_DIRECT:NEAR

                      PUBLIC  BOOT_INVA
                      PUBLIC  TUTOR
                      PUBLIC  START
                      PUBLIC  C1
                      PUBLIC  C2
                      PUBLIC  C8042A
                      PUBLIC  OBF_42B
                      PUBLIC  OBF_42A
                      PUBLIC  C8042B
                      PUBLIC  C8042C
                      PUBLIC  E0
                      PUBLIC  E0_A
                      PUBLIC  E0_B
                      PUBLIC  VIR_ERR
                      PUBLIC  E1
                      PUBLIC  F3A
                      PUBLIC  D1
                      PUBLIC  D2
                      PUBLIC  D2A
                      PUBLIC  F3D
                      PUBLIC  F3D1
                      PUBLIC  F1
                      PUBLIC  F1_A
                      PUBLIC  F1_B
                      PUBLIC  F3
                      PUBLIC  LOCK
                      PUBLIC  CM1
                      PUBLIC  CM2
                      PUBLIC  CM3
                      PUBLIC  CM4

                      PUBLIC  CM4_A
                      PUBLIC  CM4_B
                      PUBLIC  CM4_C
                      PUBLIC  CM4_D
                      PUBLIC  F3B
                      PUBLIC  F4
                      PUBLIC  F4E
                      PUBLIC  E1_A
                      PUBLIC  E1_B
                      PUBLIC  E1_C
                      PUBLIC  ADERR
                      PUBLIC  ADERR1
                      PUBLIC  VECTOR_TABLE
                      PUBLIC  SLAVE_VECTOR_TABLE
                      PUBLIC  DISK_BASE
                      PUBLIC  VIDEO_PARMS
                      PUBLIC  M4
                      PUBLIC  M5
                      PUBLIC  M6
                      PUBLIC  M7
                      PUBLIC  CRT_CHAR_GEN
                      PUBLIC  PRINT_SCREEN
                      PUBLIC  A1
                      PUBLIC  K6
                      PUBLIC  K6L
                      PUBLIC  K7
                      PUBLIC  K8
                      PUBLIC  K9
                      PUBLIC  K10
                      PUBLIC  K11
                      PUBLIC  K12
                      PUBLIC  K13
                      PUBLIC  K14
                      PUBLIC  K15
                      PUBLIC  RS232_IO
                      PUBLIC  DUMMY_RETURN
                      PUBLIC  NMI_INT
                      PUBLIC  BOOT_STRAP
                      PUBLIC  KEYBOARD_IO
                      PUBLIC  KB_INT
                      PUBLIC  DISKETTE_IO
                      PUBLIC  DISK_INT
                      PUBLIC  PRINTER_IO
                      PUBLIC  VIDEO_IO
                      PUBLIC  MEMORY_SIZE_DETERMINE
                      PUBLIC  EQUIPMENT
                      PUBLIC  CASSETTE_IO
                      PUBLIC  TIME_OF_DAY
                      PUBLIC  TIMER_INT
                      PUBLIC  HRD
                      PUBLIC  FLOPPY
                      PUBLIC  SEEKS_1
                      PUBLIC  F1780
                      PUBLIC  F1781
                      PUBLIC  F1782
```

SECTION 5

```
                                        PUBLIC  F1790
                                        PUBLIC  F1791
                                        PUBLIC  FD_TBL

                                        ;-----------------------------------------------------------------------------
                                        ; THIS MODULE HAS BEEN ADDED TO FACILITATE THE EXPANSION OF THIS PROGRAM.    :
                                        ; IT ALLOWS FOR THE FIXED ORG STATEMENT ENTRY POINTS THAT HAVE TO REMAIN     :
                                        ; AT THE SAME ADDRESSES. ADDED ON 9/16/82                                    :
                                        ;-----------------------------------------------------------------------------

                                        ;--------------------
                                        ; COPYRIGHT NOTICE
                                        ;--------------------
                                        ;               ORG     0E000H

0000  36 31 38 31 30 32                                 DB      '6181028 COPR. IBM 1984'
      38 20 43 4F 50 52
      2E 20 49 42 4D 20
      31 39 38 34

                                        ;               ORG     0E05BH
005B                                                    ORG     0005BH
005B                                    RESET           LABEL   FAR
005B                                    START:
005B  E9 0000 E                                         JMP     START_1

                                        ;+++++++++++++++++++++++

                                        ;-------------------------
                                        ; TEMPORARY STACK FOR POST
                                        ;-------------------------
005E  0000 E                            C1              DW      C11
0060  0000 E                            C2              DW      C30
0062  0000 E                            C8042A          DW      TST4_B
0064  0000 E                            OBF_42A         DW      TST4_C
0066  0000 E                            C8042B          DW      TST4_D
0068  0000 E                            C8042C          DW      E30B
006A  0000 E                            OBF_42B         DW      E30C

                                        ;-------------------------
                                        ; POST ERROR MESSAGES
                                        ;-------------------------
006C  20 31 30 31 2D 53                 E0          DB    ' 101-System Board Error',13,10 ; INTERRUPT FAILUE
      79 73 74 65 6D 20
      42 6F 61 72 64 20
      45 72 72 6F 72 0D
      0A
0085  20 31 30 32 2D 53                 E0_A        DB    ' 102-System Board Error',13,10 ; TIMER FAILURE
      79 73 74 65 6D 20
      42 6F 61 72 64 20
      45 72 72 6F 72 0D
      0A
009E  20 31 30 33 2D 53                 E0_B        DB    ' 103-System Board Error',13,10 ; TIMER INTERRUPT FAILURE
      79 73 74 65 6D 20
      42 6F 61 72 64 20
      45 72 72 6F 72 0D
      0A
00B7  20 31 30 34 2D 53                 VIR_ERR     DB    ' 104-System Board Error',13,10 ; PROTECTED MODE FAILURE
      79 73 74 65 6D 20
      42 6F 61 72 64 20
      45 72 72 6F 72 0D
      0A
00D0  20 31 30 35 2D 53                 CM4         DB    ' 105-System Board Error',13,10 ; LAST 8042 COMMAND NOT ACCEPTED
      79 73 74 65 6D 20
      42 6F 61 72 64 20
      45 72 72 6F 72 0D
      0A
00E9  20 32 30 31 2D 4D                 E1          DB    ' 201-Memory Error',13,10
      65 6D 6F 72 79 20
      45 72 72 6F 72 0D
      0A
00FC  20 34 30 31 2D 43                 E1_B        DB    ' 401-CRT Error',13,10
      52 54 20 45 72 72
      6F 72 0D 0A
010C  20 35 30 31 2D 43                 E1_C        DB    ' 501-CRT Error',13,10
      52 54 20 45 72 72
      6F 72 0D 0A
011C  20 32 30 32 2D 4D                 ADERR1      DB    ' 202-Memory Address Error',13,10    ; LINE ERROR 00->15
      65 6D 6F 72 79 20
      41 64 64 72 65 73
      73 20 45 72 72 6F
      72 0D 0A
0137  20 32 30 33 2D 4D                 ADERR       DB    ' 203-Memory Address Error',13,10    ; LINE ERROR 16->23
      65 6D 6F 72 79 20
      41 64 64 72 65 73
      73 20 45 72 72 6F
      72 0D 0A
0152  52 4F 4D 20 45 72                 F3A         DB    'ROM Error',13,10                    ; ROM CHECKSUM
      72 6F 72 0D 0A
015D  20 4B 42 20 4F 4B                 F3B         DB    ' KB OK',13                          ; KB FOR MEMORY SIZE
      0D
0164  50 41 52 49 54 59                 D1          DB    'PARITY CHECK 2',13,10
      20 43 48 45 43 4B
      20 32 0D 0A
0174  50 41 52 49 54 59                 D2          DB    'PARITY CHECK 1',13,10
      20 43 48 45 43 4B
      20 31 0D 0A
0184  3F 3F 3F 3F 3F 0D                 D2A         DB    '?????',13,10
      0A
018B  20 28 52 45 53 55                 F3D         DB    ' (RESUME = "F1" KEY)',13,10
      4D 45 20 3D 20 22
      46 31 22 20 4B 45
      59 29 0D 0A
01A1  20 20 20 20 2D 55                 F3D1        DB    '    -Unlock System Unit Keylock',13,10
      6E 6C 6F 63 6B 20
      53 79 73 74 65 6D
      20 55 6E 69 74 20
      4B 65 79 6C 6F 63
      6B 0D 0A
01C2  20 33 30 31 2D 4B                 F1          DB    ' 301-Keyboard Error',13,10          ; KEYBOARD ERROR
      65 79 62 6F 61 72
      64 20 45 72 72 6F
      72 0D 0A
01D7  20 33 30 32 2D 53                 LOCK        DB    ' 302-System Unit Keylock is Locked',13,10   ; KEYBOARD LOCK ON
      79 73 74 65 6D 20
      55 6E 69 74 20 4B
      65 79 6C 6F 63 6B
      20 69 73 20 4C 6F
      63 6B 65 64 0D 0A
01FB  20 33 30 33 2D 4B                 F1_A        DB    ' 303-Keyboard Or System Unit Error',13,10
      65 79 62 6F 61 72
      64 20 4F 72 20 53
      79 73 74 65 6D 20
      55 6E 69 74 20 45
      72 72 6F 72 0D 0A
```

```
021F  20 36 30 31 2D 44   F3           DB      ' 601-Diskette Error',13,10          ; DISKETTE ERROR
      69 73 6B 65 74 74
      65 20 45 72 72 6F
      72 0D 0A
0234  20 31 36 31 2D 53   CM1          DB      ' 161-System Options Not Set-(Run SETUP)',13,10  ; DEAD BATTERY
      79 73 74 65 6D 20
      4F 70 74 69 6F 6E
      73 20 4E 6F 74 20
      53 65 74 2D 28 52
      75 6E 20 53 45 54
      55 50 29 0D 0A
025D  20 31 36 32 2D 53   CM2          DB      ' 162-System Options Not Set-(Run SETUP)',13,10
      79 73 74 65 6D 20
      4F 70 74 69 6F 6E
      73 20 4E 6F 74 20
      53 65 74 2D 28 52
      75 6E 20 53 45 54
      55 50 29 0D 0A
                                                                          ;CMOS CHECKSUM ERROR
0286  20 31 36 33 2D 54   CM3          DB      ' 163-Time & Date Not Set-(Run SETUP)',13,10
      69 6D 65 20 26 20
      44 61 74 65 20 4E
      6F 74 20 53 65 74
      2D 28 52 75 6E 20
      53 45 54 55 50 29
      0D 0A
                                                                    ; CLOCK NOT UPDATING
                          ;-------------------------
                          ; PRINTER TABLE
                          ;-------------------------
02AC                      F4           LABEL   WORD
02AC  03BC                             DW      3BCH
02AE  0378                             DW      378H
02B0  0278                             DW      278H
02B2                      F4E          LABEL   WORD

                          ;--------- NMI ENTRY

                          ;            ORG     0E2C3H
02C3                                   ORG     002C3H
= 02C3                    NMI_INT      EQU     $

02C3  E9 0000 E                        JMP     NMI_INT_1

02C6  20 31 30 36 2D 53   CM4_A        DB      ' 106-System Board Error',13,10 ; CONVERTING LOGIC TEST
      79 73 74 65 6D 20
      42 6F 61 72 64 20
      45 72 72 6F 72 0D
      0A
02DF  20 31 30 37 2D 53   CM4_B        DB      ' 107-System Board Error',13,10 ; HOT NMI TEST
      79 73 74 65 6D 20
      42 6F 61 72 64 20
      45 72 72 6F 72 0D
      0A
02F8  20 31 30 38 2D 53   CM4_C        DB      ' 108-System Board Error',13,10 ; TIMER BUS TEST
      79 73 74 65 6D 20
      42 6F 61 72 64 20
      45 72 72 6F 72 0D
      0A
0311  20 31 30 39 2D 53   CM4_D        DB      ' 109-System Board Error',13,10 ; LOW MEG CHIP SELECT TEST
      79 73 74 65 6D 20
      42 6F 61 72 64 20
      45 72 72 6F 72 0D

      0A
                              ;---------- MEMORY SIZE ERROR
032A  20 31 36 34 2D 4D       E1_A           DB      ' 164-Memory Size Error-(Run SETUP)',13,10
      65 6D 6F 72 79 20
      53 69 7A 65 20 45
      72 72 6F 72 2D 28
      52 75 6E 20 53 45
      54 55 50 29 0D 0A

                                                                    ; CMOS DOES NOT MATCH SYSTEM
                              ;---------- KEYBOARD/SYSTEM ERROR
034E  20 33 30 34 2D 4B       F1_B           DB      ' 304-Keyboard Or System Unit Error',13,10
      65 79 62 6F 61 72
      64 20 4F 72 20 53
      79 73 74 65 6D 20
      55 6E 69 74 20 45
      72 72 6F 72 0D 0A

                                                                    ; KEYBOARD CLOCK LINE HIGH

                              ;---------- DISKETTE BOOT RECORD IS NOT VALID
0372  20 36 30 32 2D 44       BOOT_INVA      DB      ' 602-Diskette Boot Record Error',13,10
      69 73 6B 65 74 74
      65 20 42 6F 6F 74
      20 52 65 63 6F 72
      64 20 45 72 72 6F
      72 0D 0A

                              ;---------- HARD FILE ERROR MSG
0393  31 37 38 30 2D 44       F1780   DB      '1780-Disk 0 Failure',0DH,0AH
      69 73 6B 20 30 20
      46 61 69 6C 75 72
      65 0D 0A
03A8  31 37 38 31 2D 44       F1781   DB      '1781-Disk 1 Failure',0DH,0AH
      69 73 6B 20 31 20
      46 61 69 6C 75 72
      65 0D 0A
03BD  31 37 38 32 2D 44       F1782   DB      '1782-Disk Controller Failure',0DH,0AH
      69 73 6B 20 43 6F
      6E 74 72 6F 6C 6C
      65 72 20 46 61 69
      6C 75 72 65 0D 0A
03DB  31 37 39 30 2D 44       F1790   DB      '1790-Disk 0 Error',0DH,0AH
      69 73 6B 20 30 20
      45 72 72 6F 72 0D
      0A
03EE  31 37 39 31 2D 44       F1791   DB      '1791-Disk 1 Error',0DH,0AH
      69 73 6B 20 31 20
      45 72 72 6F 72 0D
      0A
                              ;--------------------------------------------------------
                              ; INITIALIZE DRIVE CHARACTERISTICS                        :
                              ;                                                         :
                              ; FIXED DISK PARAMETER TABLE                              :
                              ;                                                         :
                              ;  -  THE TABLE IS COMPOSED OF A BLOCK DEFINED AS:        :
                              ;                                                         :
                              ;   +0    (1 WORD) - MAXIMUM NUMBER OF CYLINDERS          :
                              ;   +2    (1 BYTE) - MAXIMUM NUMBER OF HEADS              :
                              ;   +3    (1 WORD) - NOT USED/SEE PC-XT                   :
                              ;   +5    (1 WORD) - STARTING WRITE PRECOMPENSATION CYL   :
                              ;   +7    (1 BYTE) - NOT USED/SEE PC-XT                   :
                              ;   +8    (1 BYTE) - CONTROL BYTE                         :
                              ;                    BIT    7 DISABLE RETRIES -OR-        :
                              ;                    BIT    6 DISABLE RETRIES             :
```

SECTION 5

```
                                  ;                  BIT    3 MORE THAN 8 HEADS                 :
                                  ;   +9    (3 BYTES)- NOT USED/SEE PC-XT                       :
                                  ;  +12    (1 WORD) - LANDING ZONE                             :
                                  ;  +14    (1 BYTE) - NUMBER OF SECTORS/TRACK                  :
                                  ;  +15    (1 BYTE) - RESERVED FOR FUTURE USE                  :
                                  ;                                                             :
                                  ;           - TO DYNAMICALLY DEFINE A SET OF PARAMETERS       :
                                  ;             BUILD A TABLE FOR UP TO 15 TYPES AND PLACE      :
                                  ;             THE CORRESPONDING VECTOR INTO INTERRUPT 41      :
                                  ;             FOR DRIVE 0 AND INTERRUPT 46 FOR DRIVE 1.       :
                                  ;                                                             :
                                  ;--------------------------------------------------------------
0401                              FD_TBL:

                                  ;----- DRIVE TYPE 01

0401  0132                                DW      0306D                   ; CYLINDERS
0403  04                                  DB      04D                     ; HEADS
0404  0000                                DW      0
0406  0080                                DW      0128D                   ; WRITE PRE-COMPENSATION CYL
0408  00                                  DB      0
0409  00                                  DB      0                       ; CONTROL BYTE
040A  00 00 00                            DB      0,0,0
040D  0131                                DW      0305D                   ; LANDING ZONE
040F  11                                  DB      17D                     ; SECTORS/TRACK
0410  00                                  DB      0

                                  ;----- DRIVE TYPE 02

0411  0267                                DW      0615D                   ; CYLINDERS
0413  04                                  DB      04D                     ; HEADS
0414  0000                                DW      0
0416  012C                                DW      0300D                   ; WRITE PRE-COMPENSATION CYL
0418  00                                  DB      0
0419  00                                  DB      0                       ; CONTROL BYTE
041A  00 00 00                            DB      0,0,0
041D  0267                                DW      0615D                   ; LANDING ZONE
041F  11                                  DB      17D                     ; SECTORS/TRACK
0420  00                                  DB      0

                                  ;----- DRIVE TYPE 03

0421  0267                                DW      0615D                   ; CYLINDERS
0423  06                                  DB      06D                     ; HEADS
0424  0000                                DW      0
0426  012C                                DW      0300D                   ; WRITE PRE-COMPENSATION CYL
0428  00                                  DB      0
0429  00                                  DB      0                       ; CONTROL BYTE
042A  00 00 00                            DB      0,0,0
042D  0267                                DW      0615D                   ; LANDING ZONE
042F  11                                  DB      17D                     ; SECTORS/TRACK
0430  00                                  DB      0

                                  ;----- DRIVE TYPE 04

0431  03AC   Ø400                         DW      0940D                   ; CYLINDERS
0433  08                                  DB      08D                     ; HEADS
0434  0000                                DW      0
0436  0200                                DW      0512D                   ; WRITE PRE-COMPENSATION CYL
0438  00                                  DB      0
0439  00                                  DB      0                       ; CONTROL BYTE
043A  00 00 00                            DB      0,0,0
043D  03AC                                DW      0940D                   ; LANDING ZONE
043F  11                                  DB      17D                     ; SECTORS/TRACK
0440  00                                  DB      0

                                  ;----- DRIVE TYPE 05

0441  03AC                                DW      0940D                   ; CYLINDERS
0443  06                                  DB      06D                     ; HEADS
0444  0000                                DW      0
0446  0200                                DW      0512D                   ; WRITE PRE-COMPENSATION CYL
0448  00                                  DB      0
0449  00                                  DB      0                       ; CONTROL BYTE
044A  00 00 00                            DB      0,0,0
044D  03AC                                DW      0940D                   ; LANDING ZONE
044F  11                                  DB      17D                     ; SECTORS/TRACK
0450  00                                  DB      0

                                  ;----- DRIVE TYPE 06

0451  0267   Ø259                         DW      0615D                   ; CYLINDERS
0453  04      5                           DB      04D                     ; HEADS
0454  0000                                DW      0
0456  FFFF                                DW      0FFFFH                  ; WRITE PRE-COMPENSATION CYL
0458  00                                  DB      0
0459  00                                  DB      0                       ; CONTROL BYTE
045A  00 00 00                            DB      0,0,0
045D  0267                                DW      0615D                   ; LANDING ZONE
045F  11                                  DB      17D                     ; SECTORS/TRACK
0460  00                                  DB      0

                                  ;----- DRIVE TYPE 07

0461  01CE                                DW      0462D                   ; CYLINDERS
0463  08                                  DB      08D                     ; HEADS
0464  0000                                DW      0
0466  0100                                DW      0256D                   ; WRITE PRE-COMPENSATION CYL
0468  00                                  DB      0
0469  00                                  DB      0                       ; CONTROL BYTE
046A  00 00 00                            DB      0,0,0
046D  01FF                                DW      0511D                   ; LANDING ZONE
046F  11                                  DB      17D                     ; SECTORS/TRACK
0470  00                                  DB      0

                                  ;----- DRIVE TYPE 08

0471  02DD   Ø39D                         DW      0733D                   ; CYLINDERS
0473  05                                  DB      05D                     ; HEADS
0474  0000                                DW      0
0476  FFFF                                DW      0FFFFH                  ; NO WRITE PRE-COMPENSATION
0478  00                                  DB      0
0479  00                                  DB      0                       ; CONTROL BYTE
047A  00 00 00                            DB      0,0,0
047D  02DD                                DW      0733D                   ; LANDING ZONE
047F  11                                  DB      17D                     ; SECTORS/TRACK
0480  00                                  DB      0

                                  ;----- DRIVE TYPE 09

0481  0384                                DW      0900D                   ; CYLINDERS
0483  0F                                  DB      15D                     ; HEADS
0484  0000                                DW      0
0486  FFFF                                DW      0FFFFH                  ; NO WRITE PRE-COMPENSATION
0488  00                                  DB      0
```

```
0489  08                         DB        008H                  ; CONTROL BYTE
048A  00 00 00                   DB        0,0,0
048D  0385                       DW        0901D                 ; LANDING ZONE
048F  11                         DB        17D                   ; SECTORS/TRACK
0490  00                         DB        0

                       ;----- DRIVE TYPE 10

0491  0334     03D4              DW        0820D                 ; CYLINDERS
0493  03         5               DB        03D                   ; HEADS
0494  0000                       DW        0
0496  FFFF                       DW        0FFFFH                ; NO WRITE PRE-COMPENSATION
0498  00                         DB        0
0499  00                         DB        0                     ; CONTROL BYTE
049A  00 00 00                   DB        0,0,0
049D  0334                       DW        0820D                 ; LANDING ZONE
049F  11                         DB        17D                   ; SECTORS/TRACK
04A0  00                         DB        0

                       ;----- DRIVE TYPE 11

04A1  0357     039D              DW        0855D                 ; CYLINDERS
04A3  05         7               DB        05D                   ; HEADS
04A4  0000                       DW        0
04A6  FFFF                       DW        0FFFFH                ; NO WRITE PRE-COMPENSATION
04A8  00                         DB        0
04A9  00                         DB        0                     ; CONTROL BYTE
04AA  00 00 00                   DB        0,0,0
04AD  0357                       DW        0855D                 ; LANDING ZONE
04AF  11                         DB        17D                   ; SECTORS/TRACK
04B0  00                         DB        0

                       ;----- DRIVE TYPE 12

04B1  0357     039D              DW        0855D                 ; CYLINDERS
04B3  07        09               DB        07D                   ; HEADS
04B4  0000                       DW        0
04B6  FFFF                       DW        0FFFFH                ; NO WRITE PRE-COMPENSATION
04B8  00                         DB        0
04B9  00                         DB        0                     ; CONTROL BYTE
04BA  00 00 00                   DB        0,0,0
04BD  0357                       DW        0855D                 ; LANDING ZONE
04BF  11                         DB        17D                   ; SECTORS/TRACK
04C0  00                         DB        0

                       ;----- DRIVE TYPE 13

04C1  0132     0264              DW        0306D                 ; CYLINDERS
04C3  08                         DB        08D                   ; HEADS
04C4  0000                       DW        0
04C6  0080                       DW        0128D                 ; WRITE PRE-COMPENSATION CYL
04C8  00                         DB        0
04C9  00                         DB        0                     ; CONTROL BYTE
04CA  00 00 00                   DB        0,0,0
04CD  013F                       DW        0319D                 ; LANDING ZONE
04CF  11                         DB        17D                   ; SECTORS/TRACK
04D0  00                         DB        0

                       ;----- DRIVE TYPE 14

04D1  02DD     02F2              DW        0733D                 ; CYLINDERS
04D3  07        08               DB        07D                   ; HEADS
04D4  0000                       DW        0
04D6  FFFF                       DW        0FFFFH                ; WRITE PRE-COMPENSATION CYL
04D8  00                         DB        0
04D9  00                         DB        0                     ; CONTROL BYTE
04DA  00 00 00                   DB        0,0,0
04DD  02DD                       DW        0733D                 ; LANDING ZONE
04DF  11                         DB        17D                   ; SECTORS/TRACK
04E0  00                         DB        0

                       ;----- DRIVE TYPE 15     RESERVED     ****  DO NOT USE ****

04E1  0000                       DW        0000D                 ; CYLINDERS
04E3  00                         DB        00D                   ; HEADS
04E4  0000                       DW        0
04E6  0000                       DW        0000D                 ; WRITE PRE-COMPENSATION CYL
04E8  00                         DB        0
04E9  00                         DB        0                     ; CONTROL BYTE
04EA  00 00 00                   DB        0,0,0
04ED  0000                       DW        0000D                 ; LANDING ZONE
04EF  00                         DB        00D                   ; SECTORS/TRACK
04F0  00                         DB        0

                       ;---------- BOOT LOADER INTERRUPT

                       ;                  ORG      0E6F2H
06F2                                      ORG      006F2H
= 06F2                 BOOT_STRAP         EQU      $
06F2  E9 0000 E                           JMP      BOOT_STRAP_1

                       ;---------------BAUD RATE INIT

                       ;                  ORG      0E729H
0729                                      ORG      00729H
0729                   A1                 LABEL    WORD

0729  0417                       DW        1047      ; 110 BAUD  ; TABLE OF INIT VALUE
072B  0300                       DW        768       ; 150
072D  0180                       DW        384       ; 300
072F  00C0                       DW        192       ; 600
0731  0060                       DW        96        ; 1200
0733  0030                       DW        48        ; 2400
0735  0018                       DW        24        ; 4800
0737  000C                       DW        12        ; 9600

                       ;------------- RS232

                       ;                  ORG      0E739H
0739                                      ORG      00739H
= 0739                 RS232_IO           EQU      $
0739  E9 0000 E                           JMP      RS232_IO_1

                       ;-------------- KEYBOARD

                       ;                  ORG      0E82EH
082E                                      ORG      0082EH
= 082E                 KEYBOARD_IO        EQU      $
082E  E9 0000 E                           JMP      KEYBOARD_IO_1


                       ;                  ORG      0E87EH
087E                                      ORG      0087EH

                       ;------ TABLE OF SHIFT KEYS AND MASK VALUES (EARLY PC)
```

SECTION 5

```
087E                            K6      LABEL   BYTE
087E  52                                DB      INS_KEY                 ; INSERT KEY
087F  3A 45 46 38 1D                    DB      CAPS_KEY,NUM_KEY,SCROLL_KEY,ALT_KEY,CTL_KEY
0884  2A 36                             DB      LEFT_KEY,RIGHT_KEY
= 0008                          K6L     EQU     $-K6

                                ;------ SHIFT_MASK_TABLE

0886                            K7      LABEL   BYTE
0886  80                                DB      INS_SHIFT               ; INSERT MODE SHIFT
0887  40 20 10 08 04                    DB      CAPS_SHIFT,NUM_SHIFT,SCROLL_SHIFT,ALT_SHIFT,CTL_SHIFT
088C  02 01                             DB      LEFT_SHIFT,RIGHT_SHIFT

                                ;------ SCAN CODE TABLES

088E  1B FF 00 FF FF FF         K8              DB      27,-1,0,-1,-1,-1,30,-1
      1E FF
0896  FF FF FF 1F FF 7F                         DB      -1,-1,-1,31,-1,127,-1,17
      FF 11
089E  17 05 12 14 19 15                         DB      23,5,18,20,25,21,9,15
      09 0F
08A6  10 1B 1D 0A FF 01                         DB      16,27,29,10,-1,1,19
      13
08AD  04 06 07 08 0A 0B                         DB      4,6,7,8,10,11,12,-1,-1
      0C FF FF
08B6  FF FF 1C 1A 18 03                         DB      -1,-1,28,26,24,3,22,2
      16 02
08BE  0E 0D FF FF FF FF                         DB      14,13,-1,-1,-1,-1,-1,-1
      FF FF
08C6  20 FF                                     DB      ' ',-1
                                ;-------- CTL TABLE SCAN
08C8                            K9      LABEL   BYTE
08C8  5E 5F 60 61 62 63                         DB      94,95,96,97,98,99,100,101
      64 65
08D0  66 67 FF FF 77 FF                         DB      102,103,-1,-1,119,-1,132,-1
      84 FF
08D8  73 FF 74 FF 75 FF                         DB      115,-1,116,-1,117,-1,118,-1
      76 FF
08E0  FF                                        DB      -1
                                ;------- LC TABLE
08E1                            K10     LABEL   BYTE
08E1  1B 31 32 33 34 35                         DB      01BH,'1234567890-=',08H,09H
      36 37 38 39 30 2D
      3D 08 09
08F0  71 77 65 72 74 79                         DB      'qwertyuiop[]',0DH,-1,'asdfghjkl;',027H
      75 69 6F 70 5B 5D
      0D FF 61 73 64 66
      67 68 6A 6B 6C 3B
      27
0909  60 FF 5C 7A 78 63                         DB      60H,-1,5CH,'zxcvbnm,./',-1,'*',-1,' '
      76 62 6E 6D 2C 2E
      2F FF 2A FF 20
091A  FF                                        DB      -1

                                ;------ UC TABLE
091B                            K11     LABEL   BYTE
091B  1B 21 40 23 24 25                         DB      27,'!@#$',37,05EH,'&*()_+',08H,0
      5E 26 2A 28 29 5F
      2B 08 00
092A  51 57 45 52 54 59                         DB      'QWERTYUIOP{}',0DH,-1,'ASDFGHJKL:"'
      55 49 4F 50 7B 7D
      0D FF 41 53 44 46
      47 48 4A 4B 4C 3A
      22
0943  7E FF 7C 5A 58 43                         DB      07EH,-1,'|ZXCVBNM<>?',-1,0,-1,' ',-1
      56 42 4E 4D 3C 3E
      3F FF 00 FF 20 FF
                                ;------ UC TABLE SCAN
0955                            K12     LABEL   BYTE
0955  54 55 56 57 58 59                         DB      84,85,86,87,88,89,90
      5A
095C  5B 5C 5D                                  DB      91,92,93
                                ;------ ALT TABLE SCAN
095F                            K13     LABEL   BYTE
095F  68 69 6A 6B 6C                            DB      104,105,106,107,108
0964  6D 6E 6F 70 71                            DB      109,110,111,112,113

                                ;------ NUM STATE TABLE
0969                            K14     LABEL   BYTE
0969  37 38 39 2D 34 35                         DB      '789-456+1230.'
      36 2B 31 32 33 30
      2E

                                ;------ BASE CASE TABLE
0976                            K15     LABEL   BYTE
0976  47 48 49 FF 4B FF                         DB      71,72,73,-1,75,-1,77
      4D
097D  FF 4F 50 51 52 53                         DB      -1,79,80,81,82,83

                                ;------ KEYBOARD INTERRUPT

                                ;               ORG     0E987H
0987                                            ORG     00987H
= 0987                          KB_INT          EQU     $
0987  E9 0000 E                                 JMP     KB_INT_1

                                ;------- DISKETTE I/O

                                ;               ORG     0EC59H
0C59                                            ORG     00C59H
= 0C59                          DISKETTE_IO     EQU     $
0C59  E9 0000 E                                 JMP     DISKETTE_IO_1

                                ;-------- DISKETTE INTERRUPT

                                ;               ORG     0EF57H
0F57                                            ORG     00F57H
= 0F57                          DISK_INT        EQU     $
0F57  E9 0000 E                                 JMP     DISK_INT_1

                                ;-------- DISKETTE PARMS

                                ;               ORG     0EFC7H
0FC7                                            ORG     00FC7H

                                ;----------------------------------------------
                                ; DISK_BASE
                                ;   THIS IS THE SET OF PARAMETERS REQUIRED FOR
                                ;   DISKETTE OPERATION.  THEY ARE POINTED AT BY THE
                                ;   DATA VARIABLE DISK_POINTER.  TO MODIFY THE PARAMETERS,
                                ;   BUILD ANOTHER PARAMETER BLOCK AND POINT AT IT
                                ;----------------------------------------------

0FC7                            DISK_BASE       LABEL   BYTE

0FC7  DF                                DB      11011111B       ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
0FC8  02                                DB      2               ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
```

```
0FC9  25                          DB      MOTOR_WAIT      ; WAIT AFTER OPN TIL MOTOR OFF
0FCA  02                          DB      2               ; 512 BYTES/SECTOR
0FCB  0F                          DB      15              ; EOT ( LAST SECTOR ON TRACK)
0FCC  1B                          DB      01BH            ; GAP LENGTH
0FCD  FF                          DB      0FFH            ; DTL
0FCE  54                          DB      054H            ; GAP LENGTH FOR FORMAT
0FCF  F6                          DB      0F6H            ; FILL BYTE FOR FORMAT
0FD0  0F                          DB      15              ; HEAD SETTLE TIME (MILLISECONDS)
0FD1  08                          DB      8               ; MOTOR START TIME (1/8 SECONDS)

                          ;------- PRINTER IO

                          ;               ORG     0EFD2H
0FD2                                      ORG     00FD2H
= 0FD2                    PRINTER_IO      EQU     $
0FD2  E9 0000 E                           JMP     PRINTER_IO_1

                          ;-------- VIDEO IO

                          ;--------- ADDED FOR POSSIBLE COMPATABILITY ENTRY POINTS


                                          ;ORG     0F045H
1045                                      ORG     01045H
                                  ASSUME  CS:CODE,DS:DATA,ES:VIDEO_RAM

                          EXTRN SET_MODE:NEAR
                          EXTRN SET_CTYPE:NEAR
                          EXTRN SET_CPOS:NEAR
                          EXTRN READ_CURSOR:NEAR
                          EXTRN READ_LPEN:NEAR
                          EXTRN ACT_DISP_PAGE:NEAR
                          EXTRN SCROLL_UP:NEAR
                          EXTRN SCROLL_DOWN:NEAR
                          EXTRN READ_AC_CURRENT:NEAR
                          EXTRN WRITE_AC_CURRENT:NEAR
                          EXTRN WRITE_C_CURRENT:NEAR
                          EXTRN SET_COLOR:NEAR
                          EXTRN WRITE_DOT:NEAR
                          EXTRN READ_DOT:NEAR
                          EXTRN WRITE_TTY:NEAR
                          EXTRN VIDEO_STATE:NEAR

1045                      M1      LABEL   WORD     ;  TABLE OF ROUTINES WITHIN VIDEO I/O
1045  0000 E                      DW      OFFSET  SET_MODE
1047  0000 E                      DW      OFFSET  SET_CTYPE
1049  0000 E                      DW      OFFSET  SET_CPOS
104B  0000 E                      DW      OFFSET  READ_CURSOR
104D  0000 E                      DW      OFFSET  READ_LPEN
104F  0000 E                      DW      OFFSET  ACT_DISP_PAGE
1051  0000 E                      DW      OFFSET  SCROLL_UP
1053  0000 E                      DW      OFFSET  SCROLL_DOWN
1055  0000 E                      DW      OFFSET  READ_AC_CURRENT
1057  0000 E                      DW      OFFSET  WRITE_AC_CURRENT
1059  0000 E                      DW      OFFSET  WRITE_C_CURRENT
105B  0000 E                      DW      OFFSET  SET_COLOR
105D  0000 E                      DW      OFFSET  WRITE_DOT
105F  0000 E                      DW      OFFSET  READ_DOT
1061  0000 E                      DW      OFFSET  WRITE_TTY
1063  0000 E                      DW      OFFSET  VIDEO_STATE
= 0020                    M1L     EQU     $-M1
                          ;               ORG     0F065H
1065                                      ORG     01065H
= 1065                    VIDEO_IO        EQU     $
1065  E9 0000 E                           JMP     VIDEO_IO_1

                          ;--------- VIDEO PARMS

                          ;               ORG     0F0A4H
10A4                                      ORG     010A4H
10A4                      VIDEO_PARMS     LABEL   BYTE

                          ;----- INIT_TABLE
10A4  38 28 2D 0A 1F 06           DB      38H,28H,2DH,0AH,1FH,6,19H       ; SET UP FOR 40X25
      19
10AB  1C 02 07 06 07              DB      1CH,2,7,6,7
10B0  00 00 00 00                 DB      0,0,0,0
= 0010                    M4      EQU     $-VIDEO_PARMS

10B4  71 50 5A 0A 1F 06           DB      71H,50H,5AH,0AH,1FH,6,19H       ; SET UP FOR 80X25
      19
10BB  1C 02 07 06 07              DB      1CH,2,7,6,7
10C0  00 00 00 00                 DB      0,0,0,0

10C4  38 28 2D 0A 7F 06           DB      38H,28H,2DH,0AH,7FH,6,64H       ; SET UP FOR GRAPHICS
      64
10CB  70 02 01 06 07              DB      70H,2,1,6,7
10D0  00 00 00 00                 DB      0,0,0,0

10D4  61 50 52 0F 19 06           DB      61H,50H,52H,0FH,19H,6,19H       ; SET UP FOR 80X25 B&W CARD
      19
10DB  19 02 0D 0B 0C              DB      19H,2,0DH,0BH,0CH
10E0  00 00 00 00                 DB      0,0,0,0

10E4                      M5      LABEL   WORD                    ; TABLE OF REGEN LENGTHS
10E4  0800                        DW      2048            ; 40X25
10E6  1000                        DW      4096            ; 80X25
10E8  4000                        DW      16384           ; GRAPHICS
10EA  4000                        DW      16384           ;

                          ;------ COLUMNS
10EC                      M6      LABEL   BYTE
10EC  28 28 50 50 28 28           DB      40,40,80,80,40,40,80,80
      50 50

                          ;------ C_REG_TAB
10F4                      M7      LABEL   BYTE                    ; TABLE OF MODE SETS
10F4  2C 28 2D 29 2A 2E           DB      2CH,28H,2DH,29H,2AH,2EH,1EH,29H ;
      1E 29


                          ;-------- MEMORY SIZE

                          ;               ORG     0F841H
1841                                      ORG     01841H
= 1841                    MEMORY_SIZE_DETERMINE   EQU  $
1841  E9 0000 E                           JMP     MEMORY_SIZE_DETERMINE_1

                          ;--------- EQUIPMENT DETERMINE
```

SECTION 5

```
                                        ;               ORG     0F84DH
184D                                                    ORG     0184DH
= 184D                                  EQUIPMENT       EQU     $
184D  E9 0000 E                                         JMP     EQUIPMENT_1

                                        ;---------- CASSETTE (NO BIOS SUPPORT)

                                        ;               ORG     0F859H
1859                                                    ORG     01859H
= 1859                                  CASSETTE_IO     EQU     $
1859  E9 0000 E                                         JMP     CASSETTE_IO_1

                                        ;----------------------------------------------------------------------
                                        ; CHARACTER GENERATOR GRAPHICS FOR 320X200 AND 640X200 GRAPHICS       :
                                        ;----------------------------------------------------------------------
                                        ;               ORG     0FA6EH
1A6E                                                    ORG     01A6EH
1A6E                                    CRT_CHAR_GEN    LABEL   BYTE
1A6E  00 00 00 00 00 00                         DB      000H,000H,000H,000H,000H,000H,000H,000H ; D_00
      00 00
1A76  7E 81 A5 81 BD 99                         DB      07EH,081H,0A5H,081H,0BDH,099H,081H,07EH ; D_01
      81 7E
1A7E  7E FF DB FF C3 E7                         DB      07EH,0FFH,0DBH,0FFH,0C3H,0E7H,0FFH,07EH ; D_02
      FF 7E
1A86  6C FE FE FE 7C 38                         DB      06CH,0FEH,0FEH,0FEH,07CH,038H,010H,000H ; D_03
      10 00
1A8E  10 38 7C FE 7C 38                         DB      010H,038H,07CH,0FEH,07CH,038H,010H,000H ; D_04
      10 00
1A96  38 7C 38 FE FE 7C                         DB      038H,07CH,038H,0FEH,0FEH,07CH,038H,07CH ; D_05
      38 7C
1A9E  10 10 38 7C FE 7C                         DB      010H,010H,038H,07CH,0FEH,07CH,038H,07CH ; D_06
      38 7C
1AA6  00 00 18 3C 3C 18                         DB      000H,000H,018H,03CH,03CH,018H,000H,000H ; D_07
      00 00
1AAE  FF FF E7 C3 C3 E7                         DB      0FFH,0FFH,0E7H,0C3H,0C3H,0E7H,0FFH,0FFH ; D_08
      FF FF
1AB6  00 3C 66 42 42 66                         DB      000H,03CH,066H,042H,042H,066H,03CH,000H ; D_09
      3C 00
1ABE  FF C3 99 BD BD 99                         DB      0FFH,0C3H,099H,0BDH,0BDH,099H,0C3H,0FFH ; D_0A
      C3 FF
1AC6  0F 07 0F 7D CC CC                         DB      00FH,007H,00FH,07DH,0CCH,0CCH,0CCH,078H ; D_0B
      CC 78
1ACE  3C 66 66 66 3C 18                         DB      03CH,066H,066H,066H,03CH,018H,07EH,018H ; D_0C
      7E 18
1AD6  3F 33 3F 30 30 70                         DB      03FH,033H,03FH,030H,030H,070H,0F0H,0E0H ; D_0D
      F0 E0
1ADE  7F 63 7F 63 63 67                         DB      07FH,063H,07FH,063H,063H,067H,0E6H,0C0H ; D_0E
      E6 C0
1AE6  99 5A 3C E7 E7 3C                         DB      099H,05AH,03CH,0E7H,0E7H,03CH,05AH,099H ; D_0F
      5A 99

1AEE  80 E0 F8 FE F8 E0                         DB      080H,0E0H,0F8H,0FEH,0F8H,0E0H,080H,000H ; D_10
      80 00
1AF6  02 0E 3E FE 3E 0E                         DB      002H,00EH,03EH,0FEH,03EH,00EH,002H,000H ; D_11
      02 00
1AFE  18 3C 7E 18 18 7E                         DB      018H,03CH,07EH,018H,018H,07EH,03CH,018H ; D_12
      3C 18
1B06  66 66 66 66 66 00                         DB      066H,066H,066H,066H,066H,000H,066H,000H ; D_13
      66 00
1B0E  7F DB DB 7B 1B 1B                         DB      07FH,0DBH,0DBH,07BH,01BH,01BH,01BH,000H ; D_14
      1B 00
1B16  3E 63 38 6C 6C 38                         DB      03EH,063H,038H,06CH,06CH,038H,0CCH,078H ; D_15
      CC 78
1B1E  00 00 00 00 7E 7E                         DB      000H,000H,000H,000H,07EH,07EH,07EH,000H ; D_16
      7E 00
1B26  18 3C 7E 18 7E 3C                         DB      018H,03CH,07EH,018H,07EH,03CH,018H,0FFH ; D_17
      18 FF
1B2E  18 3C 7E 18 18 18                         DB      018H,03CH,07EH,018H,018H,018H,018H,000H ; D_18
      18 00
1B36  18 18 18 18 7E 3C                         DB      018H,018H,018H,018H,07EH,03CH,018H,000H ; D_19
      18 00
1B3E  00 18 0C FE 0C 18                         DB      000H,018H,00CH,0FEH,00CH,018H,000H,000H ; D_1A
      00 00
1B46  00 30 60 FE 60 30                         DB      000H,030H,060H,0FEH,060H,030H,000H,000H ; D_1B
      00 00
1B4E  00 00 C0 C0 C0 FE                         DB      000H,000H,0C0H,0C0H,0C0H,0FEH,000H,000H ; D_1C
      00 00
1B56  00 24 66 FF 66 24                         DB      000H,024H,066H,0FFH,066H,024H,000H,000H ; D_1D
      00 00
1B5E  00 18 3C 7E FF FF                         DB      000H,018H,03CH,07EH,0FFH,0FFH,000H,000H ; D_1E
      00 00
1B66  00 FF FF 7E 3C 18                         DB      000H,0FFH,0FFH,07EH,03CH,018H,000H,000H ; D_1F
      00 00

1B6E  00 00 00 00 00 00                         DB      000H,000H,000H,000H,000H,000H,000H,000H ; SP D_20
      00 00
1B76  30 78 78 30 30 00                         DB      030H,078H,078H,030H,030H,000H,030H,000H ; ! D_21
      30 00
1B7E  6C 6C 6C 00 00 00                         DB      06CH,06CH,06CH,000H,000H,000H,000H,000H ; " D_22
      00 00
1B86  6C 6C FE 6C FE 6C                         DB      06CH,06CH,0FEH,06CH,0FEH,06CH,06CH,000H ; # D_23
      6C 00
1B8E  30 7C C0 78 0C F8                         DB      030H,07CH,0C0H,078H,00CH,0F8H,030H,000H ; $ D_24
      30 00
1B96  00 C6 CC 18 30 66                         DB      000H,0C6H,0CCH,018H,030H,066H,0C6H,000H ; PER CENT D_25
      C6 00
1B9E  38 6C 38 76 DC CC                         DB      038H,06CH,038H,076H,0DCH,0CCH,076H,000H ; & D_26
      76 00
1BA6  60 60 C0 00 00 00                         DB      060H,060H,0C0H,000H,000H,000H,000H,000H ; ' D_27
      00 00
1BAE  18 30 60 60 60 30                         DB      018H,030H,060H,060H,060H,030H,018H,000H ; ( D_28
      18 00
1BB6  60 30 18 18 18 30                         DB      060H,030H,018H,018H,018H,030H,060H,000H ; ) D_29
      60 00
1BBE  00 66 3C FF 3C 66                         DB      000H,066H,03CH,0FFH,03CH,066H,000H,000H ; * D_2A
      00 00
1BC6  00 30 30 FC 30 30                         DB      000H,030H,030H,0FCH,030H,030H,000H,000H ; + D_2B
      00 00
1BCE  00 00 00 00 00 30                         DB      000H,000H,000H,000H,000H,030H,030H,060H ; , D_2C
      30 60
1BD6  00 00 00 FC 00 00                         DB      000H,000H,000H,0FCH,000H,000H,000H,000H ;- D_2D
      00 00
1BDE  00 00 00 00 00 30                         DB      000H,000H,000H,000H,000H,030H,030H,000H ; . D_2E
      30 00
1BE6  06 0C 18 30 60 C0                         DB      006H,00CH,018H,030H,060H,0C0H,080H,000H ; / D_2F
      80 00

1BEE  7C C6 CE DE F6 E6                         DB      07CH,0C6H,0CEH,0DEH,0F6H,0E6H,07CH,000H ; 0 D_30
      7C 00
1BF6  30 70 30 30 30 30                         DB      030H,070H,030H,030H,030H,030H,0FCH,000H ; 1 D_31
      FC 00
1BFE  78 CC 0C 38 60 CC                         DB      078H,0CCH,00CH,038H,060H,0CCH,0FCH,000H ; 2 D_32
      FC 00
1C06  78 CC 0C 38 0C CC                         DB      078H,0CCH,00CH,038H,00CH,0CCH,078H,000H ; 3 D_33
      78 00
```

```
1C0E  1C 3C 6C CC FE 0C      DB   01CH,03CH,06CH,0CCH,0FEH,00CH,01EH,000H ; 4 D_34
      1E 00
1C16  FC C0 F8 0C 0C CC      DB   0FCH,0C0H,0F8H,00CH,00CH,0CCH,078H,000H ; 5 D_35
      78 00
1C1E  38 60 C0 F8 CC CC      DB   038H,060H,0C0H,0F8H,0CCH,0CCH,078H,000H ; 6 D_36
      78 00
1C26  FC CC 0C 18 30 30      DB   0FCH,0CCH,00CH,018H,030H,030H,030H,000H ; 7 D_37
      30 00
1C2E  78 CC CC 78 CC CC      DB   078H,0CCH,0CCH,078H,0CCH,0CCH,078H,000H ; 8 D_38
      78 00
1C36  78 CC CC 7C 0C 18      DB   078H,0CCH,0CCH,07CH,00CH,018H,070H,000H ; 9 D_39
      70 00
1C3E  00 30 30 00 00 30      DB   000H,030H,030H,000H,000H,030H,030H,000H ; : D_3A
      30 00
1C46  00 30 30 00 00 30      DB   000H,030H,030H,000H,000H,030H,030H,060H ; ; D_3B
      30 60
1C4E  18 30 60 C0 60 30      DB   018H,030H,060H,0C0H,060H,030H,018H,000H ; < D_3C
      18 00
1C56  00 00 FC 00 00 FC      DB   000H,000H,0FCH,000H,000H,0FCH,000H,000H ; = D_3D
      00 00
1C5E  60 30 18 0C 18 30      DB   060H,030H,018H,00CH,018H,030H,060H,000H ; > D_3E
      60 00
1C66  78 CC 0C 18 30 00      DB   078H,0CCH,00CH,018H,030H,000H,030H,000H ; ? D_3F
      30 00

1C6E  7C C6 DE DE DE C0      DB   07CH,0C6H,0DEH,0DEH,0DEH,0C0H,078H,000H ; @ D_40
      78 00
1C76  30 78 CC CC FC CC      DB   030H,078H,0CCH,0CCH,0FCH,0CCH,0CCH,000H ; A D_41
      CC 00
1C7E  FC 66 66 7C 66 66      DB   0FCH,066H,066H,07CH,066H,066H,0FCH,000H ; B D_42
      FC 00
1C86  3C 66 C0 C0 C0 66      DB   03CH,066H,0C0H,0C0H,0C0H,066H,03CH,000H ; C D_43
      3C 00
1C8E  F8 6C 66 66 66 6C      DB   0F8H,06CH,066H,066H,066H,06CH,0F8H,000H ; D D_44
      F8 00
1C96  FE 62 68 78 68 62      DB   0FEH,062H,068H,078H,068H,062H,0FEH,000H ; E D_45
      FE 00
1C9E  FE 62 68 78 68 60      DB   0FEH,062H,068H,078H,068H,060H,0F0H,000H ; F D_46
      F0 00
1CA6  3C 66 C0 C0 CE 66      DB   03CH,066H,0C0H,0C0H,0CEH,066H,03EH,000H ; G D_47
      3E 00
1CAE  CC CC CC FC CC CC      DB   0CCH,0CCH,0CCH,0FCH,0CCH,0CCH,0CCH,000H ; H D_48
      CC 00
1CB6  78 30 30 30 30 30      DB   078H,030H,030H,030H,030H,030H,078H,000H ; I D_49
      78 00
1CBE  1E 0C 0C 0C CC CC      DB   01EH,00CH,00CH,00CH,0CCH,0CCH,078H,000H ; J D_4A
      78 00
1CC6  E6 66 6C 78 6C 66      DB   0E6H,066H,06CH,078H,06CH,066H,0E6H,000H ; K D_4B
      E6 00
1CCE  F0 60 60 60 62 66      DB   0F0H,060H,060H,060H,062H,066H,0FEH,000H ; L D_4C
      FE 00
1CD6  C6 EE FE FE D6 C6      DB   0C6H,0EEH,0FEH,0FEH,0D6H,0C6H,0C6H,000H ; M D_4D
      C6 00
1CDE  C6 E6 F6 DE CE C6      DB   0C6H,0E6H,0F6H,0DEH,0CEH,0C6H,0C6H,000H ; N D_4E
      C6 00
1CE6  38 6C C6 C6 C6 6C      DB   038H,06CH,0C6H,0C6H,0C6H,06CH,038H,000H ; O D_4F
      38 00

1CEE  FC 66 66 7C 60 60      DB   0FCH,066H,066H,07CH,060H,060H,0F0H,000H ; P D_50
      F0 00
1CF6  78 CC CC CC DC 78      DB   078H,0CCH,0CCH,0CCH,0DCH,078H,01CH,000H ; Q D_51
      1C 00
1CFE  FC 66 66 7C 6C 66      DB   0FCH,066H,066H,07CH,06CH,066H,0E6H,000H ; R D_52
      E6 00
1D06  78 CC E0 70 1C CC      DB   078H,0CCH,0E0H,070H,01CH,0CCH,078H,000H ; S D_53
      78 00
1D0E  FC B4 30 30 30 30      DB   0FCH,0B4H,030H,030H,030H,030H,078H,000H ; T D_54
      78 00
1D16  CC CC CC CC CC CC      DB   0CCH,0CCH,0CCH,0CCH,0CCH,0CCH,0FCH,000H ; U D_55
      FC 00
1D1E  CC CC CC CC CC 78      DB   0CCH,0CCH,0CCH,0CCH,0CCH,078H,030H,000H ; V D_56
      30 00
1D26  C6 C6 C6 D6 FE EE      DB   0C6H,0C6H,0C6H,0D6H,0FEH,0EEH,0C6H,000H ; W D_57
      C6 00
1D2E  C6 C6 6C 38 38 6C      DB   0C6H,0C6H,06CH,038H,038H,06CH,0C6H,000H ; X D_58
      C6 00
1D36  CC CC CC 78 30 30      DB   0CCH,0CCH,0CCH,078H,030H,030H,078H,000H ; Y D_59
      78 00
1D3E  FE C6 8C 18 32 66      DB   0FEH,0C6H,08CH,018H,032H,066H,0FEH,000H ; Z D_5A
      FE 00
1D46  78 60 60 60 60 60      DB   078H,060H,060H,060H,060H,060H,078H,000H ; [ D_5B
      78 00
1D4E  C0 60 30 18 0C 06      DB   0C0H,060H,030H,018H,00CH,006H,002H,000H ; BACKSLASH D_5C
      02 00
1D56  78 18 18 18 18 18      DB   078H,018H,018H,018H,018H,018H,078H,000H ; ] D_5D
      78 00
1D5E  10 38 6C C6 00 00      DB   010H,038H,06CH,0C6H,000H,000H,000H,000H ; CIRCUMFLEX D_5E
      00 00
1D66  00 00 00 00 00 00      DB   000H,000H,000H,000H,000H,000H,000H,0FFH ; _ D_5F
      00 FF

1D6E  30 30 18 00 00 00      DB   030H,030H,018H,000H,000H,000H,000H,000H ; ` D_60
      00 00
1D76  00 00 78 0C 7C CC      DB   000H,000H,078H,00CH,07CH,0CCH,076H,000H ; LOWER CASE A D_61
      76 00
1D7E  E0 60 60 7C 66 66      DB   0E0H,060H,060H,07CH,066H,066H,0DCH,000H ; L.C. B D_62
      DC 00
1D86  00 00 78 CC C0 CC      DB   000H,000H,078H,0CCH,0C0H,0CCH,078H,000H ; L.C. C D_63
      78 00
1D8E  1C 0C 0C 7C CC CC      DB   01CH,00CH,00CH,07CH,0CCH,0CCH,076H,000H ; L.C. D D_64
      76 00
1D96  00 00 78 CC FC C0      DB   000H,000H,078H,0CCH,0FCH,0C0H,078H,000H ; L.C. E D_65
      78 00
1D9E  38 6C 60 F0 60 60      DB   038H,06CH,060H,0F0H,060H,060H,0F0H,000H ; L.C. F D_66
      F0 00
1DA6  00 00 76 CC CC 7C      DB   000H,000H,076H,0CCH,0CCH,07CH,00CH,0F8H ; L.C. G D_67
      0C F8
1DAE  E0 60 6C 76 66 66      DB   0E0H,060H,06CH,076H,066H,066H,0E6H,000H ; L.C. H D_68
      E6 00
1DB6  30 00 70 30 30 30      DB   030H,000H,070H,030H,030H,030H,078H,000H ; L.C. I D_69
      78 00
1DBE  0C 00 0C 0C 0C CC      DB   00CH,000H,00CH,00CH,00CH,0CCH,0CCH,078H ; L.C. J D_6A
      CC 78
1DC6  E0 60 66 6C 78 6C      DB   0E0H,060H,066H,06CH,078H,06CH,0E6H,000H ; L.C. K D_6B
      E6 00
1DCE  70 30 30 30 30 30      DB   070H,030H,030H,030H,030H,030H,078H,000H ; L.C. L D_6C
      78 00
1DD6  00 00 CC FE FE D6      DB   000H,000H,0CCH,0FEH,0FEH,0D6H,0C6H,000H ; L.C. M D_6D
      C6 00
1DDE  00 00 F8 CC CC CC      DB   000H,000H,0F8H,0CCH,0CCH,0CCH,0CCH,000H ; L.C. N D_6E
      CC 00
1DE6  00 00 78 CC CC CC      DB   000H,000H,078H,0CCH,0CCH,0CCH,078H,000H ; L.C. O D_6F
      78 00

1DEE  00 00 DC 66 66 7C      DB   000H,000H,0DCH,066H,066H,07CH,060H,0F0H ; L.C. P D_70
      60 F0
```

SECTION 5

```
1DF6  00 00 76 CC CC 7C        DB      000H,000H,076H,0CCH,0CCH,07CH,00CH,01EH ; L.C. Q D_71
      0C 1E
1DFE  00 00 DC 76 66 60        DB      000H,000H,0DCH,076H,066H,060H,0F0H,000H ; L.C. R D_72
      F0 00
1E06  00 00 7C C0 78 0C        DB      000H,000H,07CH,0C0H,078H,00CH,0F8H,000H ; L.C. S D_73
      F8 00
1E0E  10 30 7C 30 30 34        DB      010H,030H,07CH,030H,030H,034H,018H,000H ; L.C. T D_74
      18 00
1E16  00 00 CC CC CC CC        DB      000H,000H,0CCH,0CCH,0CCH,0CCH,076H,000H ; L.C. U D_75
      76 00
1E1E  00 00 CC CC CC 78        DB      000H,000H,0CCH,0CCH,0CCH,078H,030H,000H ; L.C. V D_76
      30 00
1E26  00 00 C6 D6 FE FE        DB      000H,000H,0C6H,0D6H,0FEH,0FEH,06CH,000H ; L.C. W D_77
      6C 00
1E2E  00 00 C6 6C 38 6C        DB      000H,000H,0C6H,06CH,038H,06CH,0C6H,000H ; L.C. X D_78
      C6 00
1E36  00 00 CC CC CC 7C        DB      000H,000H,0CCH,0CCH,0CCH,07CH,00CH,0F8H ; L.C. Y D_79
      0C F8
1E3E  00 00 FC 98 30 64        DB      000H,000H,0FCH,098H,030H,064H,0FCH,000H ; L.C. Z D_7A
      FC 00
1E46  1C 30 30 E0 30 30        DB      01CH,030H,030H,0E0H,030H,030H,01CH,000H ; { D_7B
      1C 00
1E4E  18 18 18 00 18 18        DB      018H,018H,018H,000H,018H,018H,018H,000H ; | D_7C
      18 00
1E56  E0 30 30 1C 30 30        DB      0E0H,030H,030H,01CH,030H,030H,0E0H,000H ; } D_7D
      E0 00
1E5E  76 DC 00 00 00 00        DB      076H,0DCH,000H,000H,000H,000H,000H,000H ; ° D_7E
      00 00

1E66  00 10 38 6C C6 C6        DB      000H,010H,038H,06CH,0C6H,0C6H,0FEH,000H ; DELTA D_7F
      FE 00
                         .LIST

                         ;------ TIME OF DAY

                         ;              ORG     0FE6EH
1E6E                                    ORG     01E6EH
= 1E6E                   TIME_OF_DAY    EQU     $
1E6E  E9 0000 E                         JMP     TIME_OF_DAY_1

                         ;-------- TIMER INTERRUPT

                         ;              ORG     0FEA5H
1EA5                                    ORG     01EA5H
= 1EA5                   TIMER_INT      EQU     $
1EA5  E9 0000 E                         JMP     TIMER_INT_1

                         ;-------- VECTOR TABLE

                         ;              ORG     0FEF3H
1EF3                                    ORG     01EF3H
1EF3                     VECTOR_TABLE   LABEL   WORD
                                                                ; VECTOR TABLE
1EF3  1EA5 R                   DW       OFFSET TIMER_INT        ; INTERRUPT 8
1EF5  0987 R                   DW       OFFSET  KB_INT          ; INTERRUPT 9
1EF7  0000 E                   DW       OFFSET  D11             ; INTERRUPT A (SLAVE INPUT)
1EF9  0000 E                   DW       OFFSET  D11             ; INTERRUPT B
1EFB  0000 E                   DW       OFFSET  D11             ; INTERRUPT C
1EFD  0000 E                   DW       OFFSET  D11             ; INTERRUPT D
1EFF  0F57 R                   DW       OFFSET DISK_INT         ; INTERRUPT E
1F01  0000 E                   DW       OFFSET  D11             ; INTERRUPT F

                         ;------- SOFTWARE INTERRUPTS

1F03  1065 R                   DW       OFFSET VIDEO_IO         ; INT 10H
1F05  184D R                   DW       OFFSET EQUIPMENT        ; INT 11H
1F07  1841 R                   DW       OFFSET MEMORY_SIZE_DETERMINE    ; INT 12H
1F09  0C59 R                   DW       OFFSET DISKETTE_IO      ; INT 13H
1F0B  0739 R                   DW       OFFSET RS232_IO         ; INT 14H
1F0D  1859 R                   DW       CASSETTE_IO             ; INT 15H
1F0F  082E R                   DW       OFFSET KEYBOARD_IO      ; INT 16H
1F11  0FD2 R                   DW       OFFSET PRINTER_IO       ; INT 17H
1F13  0000                     DW       00000H                  ; INT 18H
                         ;     DW       0F600H                  ; MUST BE INSERTED INTO TABLE LATER
1F15  06F2 R                   DW       OFFSET BOOT_STRAP       ; INT 19H
1F17  1E6E R                   DW       TIME_OF_DAY             ; INT 1AH -- TIME OF DAY
1F19  1F53 R                   DW       DUMMY_RETURN            ; INT 1BH -- KEYBOARD BREAK ADDR
1F1B  1F53 R                   DW       DUMMY_RETURN            ; INT 1CH -- TIMER BREAK ADDR
1F1D  10A4 R                   DW       VIDEO_PARMS             ; INT 1DH -- VIDEO PARAMETERS
1F1F  0FC7 R                   DW       OFFSET DISK_BASE        ; INT 1EH -- DISK PARMS
1F21  0000                     DW       0                       ; INT 1FH -- POINTER TO VIDEO EXT

1F23                     SLAVE_VECTOR_TABLE LABEL WORD          ;(INTERRUPT 70 THRU 7F)

1F23  0000 E                   DW       OFFSET  RTC_INT         ; INT 70 REAL TIME CLOCK INTERRUPT VECTOR
1F25  0000 E                   DW       OFFSET  RE_DIRECT       ; INT 71 REDIRECT THIS TO INT A
1F27  0000 E                   DW       OFFSET  D11             ; INT 72
1F29  0000 E                   DW       OFFSET  D11             ; INT 73
1F2B  0000 E                   DW       OFFSET  D11             ; INT 74
1F2D  0000 E                   DW       OFFSET  INT_287         ; INT 75 MATH PROCESSOR INTERRUPT
1F2F  0000 E                   DW       OFFSET  D11             ; INT 76
1F31  0000 E                   DW       OFFSET  D11             ; INT 77

                         ;------- DUMMY INTERRUPT HANDLER

                         ;              ORG     0FF53H
1F53                                    ORG     01F53H
= 1F53                   DUMMY_RETURN   EQU     $

1F53  CF                                IRET                    ;

                         ;------- PRINT SCREEN

                         ;              ORG     0FF54H
1F54                                    ORG     01F54H
= 1F54                   PRINT_SCREEN   EQU     $
1F54  E9 0000 E                         JMP     PRINT_SCREEN_1
                         .LIST                                          ; TUTOR
```

```
                                         ;-----------------------------------------
                                         ;                                        :
                                         ;  POWER ON RESET VECTOR                 :
                                         ;                                        :
                                         ;-----------------------------------------
                                         ;        ORG      0FFF0H
1FF0                                              ORG      01FF0H
                                         PUBLIC    P_O_R
                                         ; ------ POWER ON RESET
1FF0                                     P_O_R     LABEL   FAR

1FF0  EA                                          DB       0EAH            ;HARD CODE JUMP
1FF1  005B R                                      DW       OFFSET  RESET   ;OFFSET
1FF3  F000                                        DW       0F000H          ;SEGMENT

1FF5  30 31 2F 31 30 2F                           DB       '01/10/84'      ;RELEASE MARKER
      38 34
1FFE                                              ORG      01FFEH          ;
1FFE  FC                                          DB       0FCH            ;THIS PC'S ID
1FFF                                     CODE     ENDS
                                                  END
```

SECTION 5

SECTION 6. INSTRUCTION SET

Contents

Notes:

Instruction Set

80286 Microprocessor Instruction Set

The following is an instruction set summary for the Intel 80286 microprocessor.

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

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

Push All

01100000

POP = Pop

Memory

10001111	mod000 r/m

Register

01011reg

Segment Register

000reg111 reg ≠ 0

POPA = Pop All

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

Translate Byte to AL

11010111

LEA = Load EA to Register

Load EA to Register

10001101	mod reg r/m

LDS = Load Pointer to DS

Load Pointer to DS

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

LES = Load Pointer to ES

Load Pointer to ES

11000100	mod reg r/m mod ≠ 11

LAHF = Load AH with Flags

Load AH with Flags

10011111

SAHF = Load AH with Flags

Store AH with Flags

10011110

PUSHF = Push Flags

Push Flags

10011100

POPF = Pop Flags

Pop Flags

10011101

Arithmetic

ADD = Add

Reg/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

Reg/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

Reg/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

Reg/Memory with Register to Either

000110dw	mod reg r/m

Immediate to Register Memory

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

Immediate to Accumulator

0001110w	data	data if w = 1

DEC = Decrement

Register/Memory

1111111w	mod001 r/m

Register

01001reg

CMP = Compare

Register/Memory with Register

0011101w	mod reg r/m

Register with Register/Memory

0011100w	mod reg r/m

Immediate with Register/Memory

100000sw	mod111 r/m	Data	Data if sw = 01

Immediate with Accumulator

0001110w	Data	Data if w = 1

NEG = Change Sign

Change Sign

1111011w	mod011 r/m

AAA = ASCII Adjust for Add

ASCII Adjust for Add

00110111

DEC = Decimal Adjust for Add

Decimal Adjust for Add

00100111

AAS = ASCII Adjust for Subtract

ASCII Adjust for Subtract
00111111

DAS = Decimal Adjust for Subtract

Decimal Adjust for Subtract
00110111

MUL = Multiply (Unsigned)

Multiply	
1111011w	mod100 r/m

IMUL = Integer Multiply (Signed)

Integer Multiply	
1111011w	mod101 r/m

IIMUL = Integer Immediate Multiply (Signed)

Integer Immediate Multiply			
011010s1	mod reg r/m	Data	Data if s = 0

DIV = Divide (Unsigned)

Divide	
1111011w	mod110 r/m

IDIV = Integer Divide (Signed)

Integer Divide	
1111011w	mod111 r/m

AAM = ASCII Adjust for Multiply

ASCII Adjust for Multiply	
11010100	00001010

AAD = ASCII Adjust for Divide

ASCII Adjust for Divide

11010101	00001010

CBW = Convert Byte to Word

Convert Byte to Word

10011000

CWD = Convert Word to Double Word

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

Reg/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

Reg/ 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

Reg/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

Invert Register/Memory	
1111011w	mod010 r/m

String Manipulation

MOVS = Move Byte Word

Move Byte Word
1010010w

CMPS = Compare Byte Word

Compare Byte Word
1010011w

SCAS = Scan Byte Word

Scan Byte Word
1010111w

LODS = Load Byte Word to AL/AX

Load Byte Word to AL/AX
1010110w

STOS = Store Byte Word from AL/AX

Store Byte Word from AL/AX
1010101w

INS = Input Byte from DX Port

Input Byte Word from DX Port
0110110w

OUTS = Output Byte to DX Port

Output Byte Word to DX Port
0110111w

MOVS = Move String

Move String	
11110010	1010010w

CMPS = Compare String

Compare String	
1111001z	1010011w

SCAS = Scan String

Scan String	
11110010	1010111w

LODS = Load String

Load String	
11110010	1010110w

STOS = Store String

Store String	
11110010	1010101w

INS = Input String

Input String	
11110010	0110110w

OUTS = Output String

Output String	
11110010	1010011w

Control Transfer

CALL = Call

Direct Within Segment

11101000	disp-low	disp-low

Register/Memory Indirect Within Segment

11111111	mod010 r/m

Direct Intersegment

10011010	Segment Offset	Segment Selector

Protected Mode Only (Direct Intersegment)

- Via call gate to same privilege level
- Via call gate to different privilege level, no parameters
- Via call gate to different privilege level, x parameters
- Via TSS
- Via task gate.

Indirect Intersegment

11111111	mod011 r/m (mod≠11)

Protected Mode Only (Indirect Intersegment)

- Via call gate to same privilege level
- Via call gate to different privilege level, no parameters
- Via call gate to different privilege level, x parameters
- Via TSS
- Via task gate.

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

Protected Mode Only (Direct Intersegment)

- Via call gate to same privilege level
- Via TSS
- Via task gate.

Indirect Intersegment

11111111	mod101 r/m (mod ≠ 11)

Protected Mode Only (Indirect Intersegment)

- Via call gate to same privilege level
- Via TSS
- Via task gate.

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

Protected Mode Only (RET)

- To Different Privilege Level

JE/JZ = Jump on Equal Zero

Jump on Equal Zero

01110100	disp

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

Jump on Less Not Greater, or Equal

01111100	disp

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

Jump on Less, or Equal Not Greater

01111110	disp

JB/JNAE = Jump on Less, or Equal Not Greater

Jump on Less, or Equal Not Greater

01110010	disp

JBE/JNA = Jump on Below, or Equal NotAbove

Jump on Below, or Equal Not Above

01110110	disp

JP/JPE = Jump on Parity Parity Even

Jump on Parity Parity Even

01111010	disp

JO = Jump on Overflow

Jump on Overflow

01110000	disp

JS = Jump on Sign

Jump on Sign

01111000	disp

JNE/JNZ = Jump on Not Equal Not Zero

Jump on Not Equal Not Zero	
01110101	disp

JNL/JGE = Jump on Not Less Greater or Equal

Jump on Not Less Greater or Zero	
01111101	disp

JNLE/JG = Jump on Not Less or Equal Greater

Jump on Not Less or Equal Greater	
01111111	disp

JNB/JAE = Jump on Not Below Above or Equal

Jump on Not Below Above or Equal	
01110011	disp

JNBE/JA = Jump on Not Below or Equal Above

Jump on Not Below or Equal Above	
01110111	disp

JNP/JPO = Jump on Not Parity Parity Odd

Jump on Not Parity Parity Odd	
01111011	disp

JNO = Jump on Not Overflow

Jump on Not Overflow	
01110001	disp

JNS = Jump on Not Sign

Jump on Not Sign	
01111011	disp

LOOP = Loop CX Times

Loop CX Times

11100010	disp

LOOPZ/LOOPE = Loop while Zero Equal

Loop while Zero Equal

11100001	disp

LOOPNZ/LOOPNE = Loop while Not Equal Zero

Loop while Not Equal Zero

11100000	disp

JCXZ = Jump on CX Zero

Jump on CX Zero

11100011	disp

ENTER = Enter Procedure

Enter Procedure

11001000	data-low	data-high	L

L=0

L=1

L>1

LEAVE = Leave Procedure

Leave Procedure

11001001

INT = Interrupt

Type Specified

11001101	Type

Type 3

11001100

INTO = Interrupt on Overflow

Interrupt on Overflow
11001110

Protected Mode Only

- Via interrupt or trap gate to same privilege level
- Via interrupt or trap gat to different privilege level
- Via task gate.

IRET = Interrupt Return

Interrupt Return
11001111

Protected Mode Only

- To same privilege level
- To different task (NT = 1).

BOUND = Detect Value Out of Range

Detect Value Out of Range	
01100010	mod reg r/m

Processor Control

CLC = Clear Carry

Clear Carry
1111100

CMC = Complement Carry

Complement Carry
11001111

STC = Set Carry

Set Carry
11111001

CLD = Clear Direction

Clear Direction
11111100

STD = Set Direction

Set Direction
11111101

CLI Clear Interrupt

Clear Interrupt
11111010

STI = Set Interrupt

Set Interrupt
11111011

HLT = Halt

Halt
11110100

WAIT = Wait

Wait
10011011

LOCK = Bus Lock Prefix

Bus Lock Prefix
11110000

CTS = Clear Task Switched Flag

Clear Task Switched Flag	
00001111	00000110

ESC = Processor Extension Escape

Processor Extension Escape

10011TTT	modLLL r/m

Protection Control

LGDT = Load Global Descriptor Table Register

Load Global Descriptor Table Register

00001111	00000001	mod010 r/m

SGDT = Store Global Descriptor Table Register

Store Global Descriptor Table Register

00001111	00000001	mod000 r/m

LIDT = Load Interrupt Descriptor Table Register

Load Interrupt Descriptor Table Register

00001111	00000001	mod011 r/m

SIDT = Store Interrupt Descriptor Table Register

Store Interrupt Descriptor Table Register

00001111	00000001	mod001 r/m

LLDT = Load Local Descriptor Table Register from Register Memory

Load Local Descriptor Table Register from Register Memory

00001111	00000000	mod010 r/m

SLDT = Store Local Descriptor Table Register from Register Memory

Store Local Descriptor Table Register from Register Memory

00001111	00000000	mod000 r/m

LTR = Load Task Register from Register Memory

Load Task Register from Register Memory

00001111	00000000	mod011 r/m

STR = Store Task Register to Register Memory

Store Task Register to Register Memory		
00001111	00000000	mod001 r/m

LMSW = Load Machine Status Word from Register Memory

Load Machine Status Word from Register Memory		
00001111	00000001	mod110 r/m

SMSW = Store Machine Status Word

Store Machine Status Word		
00001111	00000001	mod100 r/m

LAR = Load Access Rights from Register Memory

Load Access Rights from Register Memory		
00001111	00000010	mod reg r/m

LSL = Load Segment Limit from Register Memory

Load Segment Limit from Register Memory		
00001111	00000011	mod reg r/m

ARPL = Adjust Requested Privilege Level from Register Memory

Adjust Requested Privilege Level from Register Memory		
	01100011	mod reg r/m

VERR = Verify Read Access; Register Memory

Verify Read Access; Register Memory		
00001111	00000000	mod100 r/m

VERR = Verify Write Access

Verify Write Access		
00001111	00000000	mod101 r/m

Note: 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).

Segment Override Prefix

Segment Override Prefix
001reg001

reg is assigned as follows:

reg	Segment Register
00	ES
01	CS
10	SS
11	DS

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.

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)

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

	Compare ST(i) to ST(0) and pop twice
escape 110	11011001

FTST = Test ST(0)

	Test ST(0)
escape 001	11100100

FXAM = Examine ST(0)

	Examine ST(0)
escape 001	11100101

Constants

FLDZ = Load + 0.0 into ST(0)

	Load + 0.0 into ST(0)
escape 000	11101110

FLD1 = Load + 1.0 into ST(0)

	Load + 1.0 into ST(0)
escape 001	11101000

FLDP1 = Load π into ST(0) π into ST(0)

	Load
escape 001	11101011

FLDL2T = Load $\log_2$ 10 intoST(0) $_2$ 10 into ST(0)

	Load log
escape 001	11101001

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

Load log	
escape 001	11101100

FLDLN2 = Load $\log_{e}$ 2 into ST(0) $_{e}$ 2 into ST(0)

Load log	
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	1111 r r/m

FSQRT = Square Root of ST(0)

Square Root of ST(0)	
escape 001	11111010

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

Scale ST(0) by ST(1)	
escape 001	11111101

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

Partial Remainder of ST(0) + ST(1)	
escape 001	11111000

FRNDINT = Round ST(0) to Integer

Round ST(0) to Integer	
escape 001	11111100

FXTRACT = Extract Components of ST(0)

Extract Components of ST(0)	
escape 001	11110100

FABS = Absolute Value of ST(0)

Absolute Value of ST(0)	
escape 001	11100001

FCHS = Change Sign of ST(0)

Change Sign of ST(0)	
escape 001	11100000

Transcendental

FPTAN = Partial Tangent of ST(0)

Partial Tangent of ST(0)

escape 001	11110010

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

Partial Arctangent of ST(0) ÷ ST(1)

escape 001	11110011

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

escape 001	11110000

FYL2X = ST(1) x Log_2 [ST(0)] $_2$ [ST(0)]

ST(1) x log

escape 001	11110001

FYL2XP1 = ST(1) x Log_2 [ST(0) + 1] $_2$ [ST(0) + 1]

ST(1) x log

escape 001	11111001

Processor Control

FINT = Initialize NPX

Initialize NPX

escape 011	11100011

FSETPM = Enter Protected Mode

Enter Protected Mode

escape 011	11100100

FSTSWAX = Store Control Word

	Store Control Word
escape 111	11100000

FLDCW = Load Control Word

	Load Control Word
escape 001	mod 101 r/m

FSTCW = Store Control Word

	Store Control Word
escape 001	mod 111 r/m

FSTSW = Store Status Word

	Store Status Word
escape 101	mod 101 r/m

FCLEX = Clear Exceptions

	Clear Exceptions
escape 011	11100010

FSTENV = Store Environment

	Store Environment
escape 001	mod 110 r/m

FLDENV = Load Environment

	Load Environment
escape 001	mod 100 r/m

FSAVE = Save State

	Save State
escape 101	mod 110 r/m

FRSTOR = Restore State

Restore State

escape 101	mod 100 r/m

FINCSTP = Increment Stack Pointer

Increment Stack Pointer

escape 001	11110111

FDECSTP = Decrement Stack Pointer

Decrement Stack Pointer

escape 001	11110110

FFREE = Free ST(i)

Free ST(i)

escape 101	11000ST(i)

FNOP = No Operation

No Operation

escape 001	11010000

Notes:

SECTION 7. CHARACTERS, KEYSTROKES, AND COLORS

Contents

Notes:

Characters, Keystrokes, and Color

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 Green	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			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
28	43	+	+	Shift	Green	Light Cyan	High Intensity
2C	44	,	,		Green	Light Red	High Intensity
2D	45	—	—		Green	Light Magenta	High Intensity
2E	46	.	.	Note 2	Green	Yellow	High Intensity

Value		As Characters			As Text Attributes		
					Color/Graphics Monitor Adapter		IBM Monochrome Display Adapter
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground	
2F	47	/	/		Green	White	High Intensity
30	48	0	0	Note 3	Cyan	Black	Normal
31	49	1	1	Note 3	Cyan	Blue	Underline
32	50	2	2	Note 3	Cyan	Green	Normal
33	51	3	3	Note 3	Cyan	Cyan	Normal
34	52	4	4	Note 3	Cyan	Red	Normal
35	53	5	5	Note 3	Cyan	Magenta	Normal
36	54	6	6	Note 3	Cyan	Brown	Normal
37	55	7	7	Note 3	Cyan	Light Grey	Normal
38	56	8	8	Note 3	Cyan	Dark Grey	High Intensity
39	57	9	9	Note 3	Cyan	Light Blue	High Intensity Underline
3A	58	:	:	Shift	Cyan	Light Green	High Intensity
3B	59	;	;		Cyan	Light Cyan	High Intensity
3C	60	<	<	Shift	Cyan	Light Red	High Intensity
3D	61	=	=		Cyan	Light Magenta	High Intensity
3E	62	>	>	Shift	Cyan	Yellow	High Intensity
3F	63	?	?	Shift	Cyan	White	High Intensity
40	64	@	@	Shift	Red	Black	Normal
41	65	A	A	Note 4	Red	Blue	Underline
42	66	B	B	Note 4	Red	Green	Normal
43	67	C	C	Note 4	Red	Cyan	Normal
44	68	D	D	Note 4	Red	Red	Normal
45	69	E	E	Note 4	Red	Magenta	Normal
46	70	F	F	Note 4	Red	Brown	Normal
47	71	G	G	Note 4	Red	Light Grey	Normal
48	72	H	H	Note 4	Red	Dark Grey	High Intensity
49	73	I	I	Note 4	Red	Light Blue	High Intensity Underline
4A	74	J	J	Note 4	Red	Light Green	High Intensity

Value		As Characters			As Text Attributes		
					Color/Graphics Monitor Adapter		IBM Monochrome Display Adapter
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground	
4B	75	K	K	Note 4	Red	Light Cyan	High Intensity
4C	76	L	L	Note 4	Red	Light Red	High Intensity
4D	77	M	M	Note 4	Red	Light Magenta	High Intensity
4E	78	N	N	Note 4	Red	Yellow	High Intensity
4F	79	O	O	Note 4	Red	White	High Intensity
50	80	P	P	Note 4	Magenta	Black	Normal
51	81	Q	Q	Note 4	Magenta	Blue	Underline
52	82	R	R	Note 4	Magenta	Green	Normal
53	83	S	S	Note 4	Magenta	Cyan	Normal
54	84	T	T	Note 4	Magenta	Red	Normal
55	85	U	U	Note 4	Magenta	Magenta	Normal
56	86	V	V	Note 4	Magenta	Brown	Normal
57	87	W	W	Note 4	Magenta	Light Grey	Normal
58	88	X	X	Note 4	Magenta	Dark Grey	High Intensity
59	89	Y	Y	Note 4	Magenta	Light Blue	High Intensity Underline
5A	90	Z	Z	Note 4	Magenta	Light Green	High Intensity
5B	91	[	[		Magenta	Light Cyan	High Intensity
5C	92	\	\		Magenta	Light Red	High Intensity
5D	93	]	]		Magenta	Light Magenta	High Intensity
5E	94	^	^	Shift	Magenta	Yellow	High Intensity
5F	95	—	—	Shift	Magenta	White	High Intensity
60	96	`	`		Yellow	Black	Normal
61	97	a	a	Note 5	Yellow	Blue	Underline
62	98	b	b	Note 5	Yellow	Green	Normal
63	99	c	c	Note 5	Yellow	Cyan	Normal
64	100	d	d	Note 5	Yellow	Red	Normal
65	101	e	e	Note 5	Yellow	Magenta	Normal
66	102	f	f	Note 5	Yellow	Brown	Normal

Value		As Characters			As Text Attributes		
					Color/Graphics Monitor Adapter		IBM Monochrome Display Adapter
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground	
67	103	g	g	Note 5	Yellow	Light Grey	Normal
68	104	h	h	Note 5	Yellow	Dark Grey	High Intensity
69	105	i	i	Note 5	Yellow	Light Blue	High Intensity Underline
6A	106	j	j	Note 5	Yellow	Light Green	High Intensity
6B	107	k	k	Note 5	Yellow	Light Cyan	High Intensity
6C	108	l	l	Note 5	Yellow	Light Red	High Intensity
6D	109	m	m	Note 5	Yellow	Light Magenta	High Intensity
6E	110	n	n	Note 5	Yellow	Yellow	High Intensity
6F	111	o	o	Note 5	Yellow	White	High Intensity
70	112	p	p	Note 5	White	Black	Reverse Video
71	113	q	q	Note 5	White	Blue	Underline
72	114	r	r	Note 5	White	Green	Normal
73	115	s	s	Note 5	White	Cyan	Normal
74	116	f	f	Note 5	White	Red	Normal
75	117	u	u	Note 5	White	Magenta	Normal
76	118	v	v	Note 5	White	Brown	Normal
77	119	w	w	Note 5	White	Light Grey	Normal
78	120	x	x	Note 5	White	Dark Grey	Reverse Video
79	121	y	y	Note 5	White	Light Blue	High Intensity Underline
7A	122	z	z	Note 5	White	Light Green	High Intensity
7B	123	{	{	Shift	White	Light Cyan	High Intensity
7C	124	¦	¦	Shift	White	Light Red	High Intensity
7D	125	}	}	Shift	White	Light Magenta	High Intensity
7E	126	~	~	Shift	White	Yellow	High Intensity
7F	127	Δ	Ctrl ←		White	White	High Intensity

Value		As Characters			As Text Attributes		
					Color/Graphics Monitor Adapter		IBM Monochrome Display Adapter
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground	
* * * *	80 to FF Hex are Flashing in both Color & IBM Monochrome * * * *						
80	128	Ç	Alt 128	Note 6	Black	Black	Non-Display
81	129	ü	Alt 129	Note 6	Black	Blue	Underline
82	130	é	Alt 130	Note 6	Black	Green	Normal
83	131	â	Alt 131	Note 6	Black	Cyan	Normal
84	132	ä	Alt 132	Note 6	Black	Red	Normal
85	133	à	Alt 133	Note 6	Black	Magenta	Normal
86	134	å	Alt 134	Note 6	Black	Brown	Normal
87	135	ç	Alt 135	Note 6	Black	Light Grey	Normal
88	136	ê	Alt 136	Note 6	Black	Dark Grey	Non-Display
89	137	ë	Alt 137	Note 6	Black	Light Blue	High Intensity Underline
8A	138	è	Alt 138	Note 6	Black	Light Green	High Intensity
8B	139	ï	Alt 139	Note 6	Black	Light Cyan	High Intensity
8C	140	î	Alt 140	Note 6	Black	Light Red	High Intensity
8D	141	ì	Alt 141	Note 6	Black	Light Magenta	High Intensity
8E	142	Ä	Alt 142	Note 6	Black	Yellow	High Intensity
8F	143	Å	Alt 143	Note 6	Black	White	High Intensity
90	144	É	Alt 144	Note 6	Blue	Black	Normal
91	145	æ	Alt 145	Note 6	Blue	Blue	Underline
92	146	Æ	Alt 146	Note 6	Blue	Green	Normal
93	147	ô	Alt 147	Note 6	Blue	Cyan	Normal
94	148	ö	Alt 148	Note 6	Blue	Red	Normal
95	149	ò	Alt 149	Note 6	Blue	Magenta	Normal
96	150	û	Alt 150	Note 6	Blue	Brown	Normal
97	151	ù	Alt 151	Note 6	Blue	Light Grey	Normal
98	152	ÿ	Alt 152	Note 6	Blue	Dark Grey	High Intensity
99	153	Ö	Alt 153	Note 6	Blue	Light Blue	High Intensity Underline
9A	154	Ü	Alt 154	Note 6	Blue	Light Green	High Intensity

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

Value		As Characters			As Text Attributes		
					Color/Graphics Monitor Adapter		IBM Monochrome Display Adapter
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground	
B7	183	╖	Alt 183	Note 6	Cyan	Light Grey	Normal
B8	184	╕	Alt 184	Note 6	Cyan	Dark Grey	High Intensity
B9	185	╣	Alt 185	Note 6	Cyan	Light Blue	High Intensity Underline
BA	186	║	Alt 186	Note 6	Cyan	Light Green	High Intensity
BB	187	╗	Alt 187	Note 6	Cyan	Light Cyan	High Intensity
BC	188	╝	Alt 188	Note 6	Cyan	Light Red	High Intensity
BD	189	╜	Alt 189	Note 6	Cyan	Light Magenta	High Intensity
BE	190	╛	Alt 190	Note 6	Cyan	Yellow	High Intensity
BF	191	┐	Alt 191	Note 6	Cyan	White	High Intensity
C0	192	└	Alt 192	Note 6	Red	Black	Normal
C1	193	┴	Alt 193	Note 6	Red	Blue	Underline
C2	194	┬	Alt 194	Note 6	Red	Green	Normal
C3	195	├	Alt 195	Note 6	Red	Cyan	Normal
C4	196	─	Alt 196	Note 6	Red	Red	Normal
C5	197	┼	Alt 197	Note 6	Red	Magenta	Normal
C6	198	╞	Alt 198	Note 6	Red	Brown	Normal
C7	199	╟	Alt 199	Note 6	Red	Light Grey	Normal
C8	200	╚	Alt 200	Note 6	Red	Dark Grey	High Intensity
C9	201	╔	Alt 201	Note 6	Red	Light Blue	High Intensity Underline
CA	202	╩	Alt 202	Note 6	Red	Light Green	High Intensity
CB	203	╦	Alt 203	Note 6	Red	Light Cyan	High Intensity
CC	204	╠	Alt 204	Note 6	Red	Light Red	High Intensity
CD	205	═	Alt 205	Note 6	Red	Light Magenta	High Intensity
CE	206	╬	Alt 206	Note 6	Red	Yellow	High Intensity
CF	207	╧	Alt 207	Note 6	Red	White	High Intensity
D0	208	╨	Alt 208	Note 6	Magenta	Black	Normal

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	Yellow	Black	Normal
E1	225	β	Alt 225	Note 6	Yellow	Blue	Underline
E2	226	Γ	Alt 226	Note 6	Yellow	Green	Normal
E3	227	π	Alt 227	Note 6	Yellow	Cyan	Normal
E4	228	Σ	Alt 228	Note 6	Yellow	Red	Normal
E5	229	σ	Alt 229	Note 6	Yellow	Magenta	Normal
E6	230	μ	Alt 230	Note 6	Yellow	Brown	Normal
E7	231	τ	Alt 231	Note 6	Yellow	Light Grey	Normal
E8	232	Φ	Alt 232	Note 6	Yellow	Dark Grey	High Intensity
E9	233	θ	Alt 233	Note 6	Yellow	Light Blue	High Intensity Underline
EA	234	Ω	Alt 234	Note 6	Yellow	Light Green	High Intensity
EB	235	δ	Alt 235	Note 6	Yellow	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	Yellow	Light Red	High Intensity
ED	237	ϕ	Alt 237	Note 6	Yellow	Light Magenta	High Intensity
EE	238	ϵ	Alt 238	Note 6	Yellow	Yellow	High Intensity
EF	239	∩	Alt 239	Note 6	Yellow	White	High Intensity
F0	240	≡	Alt 240	Note 6	White	Black	Reverse Video
F1	241	±	Alt 241	Note 6	White	Blue	Underline
F2	242	≥	Alt 242	Note 6	White	Green	Normal
F3	243	≤	Alt 243	Note 6	White	Cyan	Normal
F4	244	⌠	Alt 244	Note 6	White	Red	Normal
F5	245	⌡	Alt 245	Note 6	White	Magenta	Normal
F6	246	÷	Alt 246	Note 6	White	Brown	Normal
F7	247	≈	Alt 247	Note 6	White	Light Grey	Normal
F8	248	○	Alt 248	Note 6	White	Dark Grey	Reverse Video
F9	249	●	Alt 249	Note 6	White	Light Blue	High Intensity Underline
FA	250	•	Alt 250	Note 6	White	Light Green	High Intensity
FB	251	√	Alt 251	Note 6	White	Light Cyan	High Intensity
FC	252	η	Alt 252	Note 6	White	Light Red	High Intensity
FD	253	2	Alt 253	Note 6	White	Light Magenta	High Intensity
FE	254	■	Alt 254	Note 6	White	Yellow	High Intensity
FF	255	BLANK	Alt 255	Note 6	White	White	High Intensity

NOTES

1. Asterisk (*) can be typed using two methods: press the PrtSc 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 0-255 may be entered in this fashion (with Caps Lock activated, character codes 97-122 will display uppercase.)

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	░	└	╨	α	≡
1	1	ü	æ	í	▒	┴	╤	β	±
2	2	é	Æ	ó	▓	┬	╥	Γ	≥
3	3	â	ô	ú	│	├	╙	π	≤
4	4	ä	ö	ñ	┤	─	╘	Σ	⌠
5	5	à	ò	Ñ	╡	┼	╒	σ	⌡
6	6	å	û	ª	╢	╞	╓	µ	÷
7	7	ç	ù	º	╖	╟	╫	τ	≈
8	8	ê	ÿ	¿	╕	╚	╪	Φ	°
9	9	ë	Ö	⌐	╣	╔	┘	Θ	•
10	A	è	Ü	¬	║	╩	┌	Ω	·
11	B	ï	¢	½	╗	╦	█	δ	√
12	C	î	£	¼	╝	╠	▄	∞	^{n}
13	D	ì	¥	¡	╜	═	▌	φ	^{2}
14	E	Ä	₧	«	╛	╬	▐	∈	■
15	F	Å	ƒ	»	┐	╧	▀	∩	BLANK 'FF'

Notes:

SECTION 8. COMMUNICATIONS

Contents

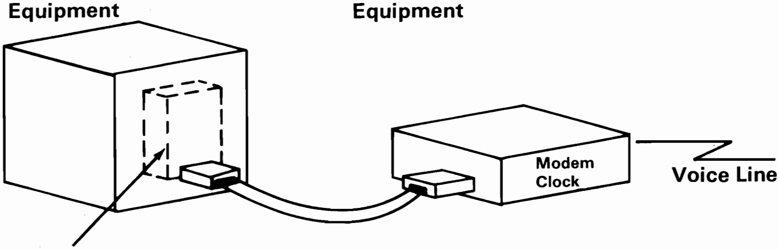
SECTION 8

Notes:

Communications

Information-processing equipment used for communication is called data terminal equipment (DTE.) Equipment used to connect the DTE to the communication line is called data communication equipment (DCE.)

An adapter connects the data terminal equipment to the data communication line as shown in the following figure:



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

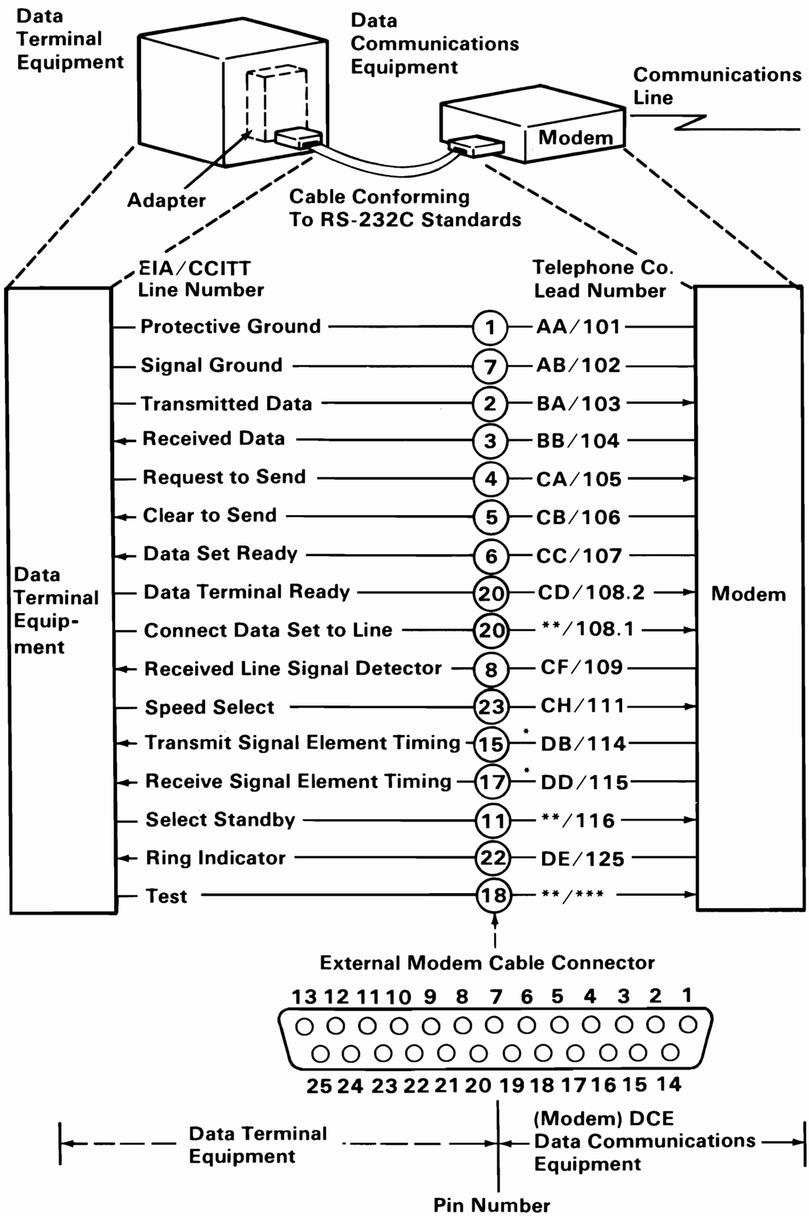
EIA standards are labeled RS-x (recommended standards-x), and CCITT standards are labeled V.x or X.x, where x is the number of the standard.

The EIA RS-232 interface standard defines the connector type, pin numbers, line names, and signal levels used to connect data terminal equipment to data communications equipment for the purpose of transmitting and receiving data. Since the RS-232 standard was developed, it has been revised three times. The three revised standards are RS-232A, RS-232B, and the presently used RS-232C.

The CCITT V.24 interface standard is equivalent to the RS-232C standard; therefore, the descriptions of the EIA standards also apply to the CCITT standards.

SECTION 8

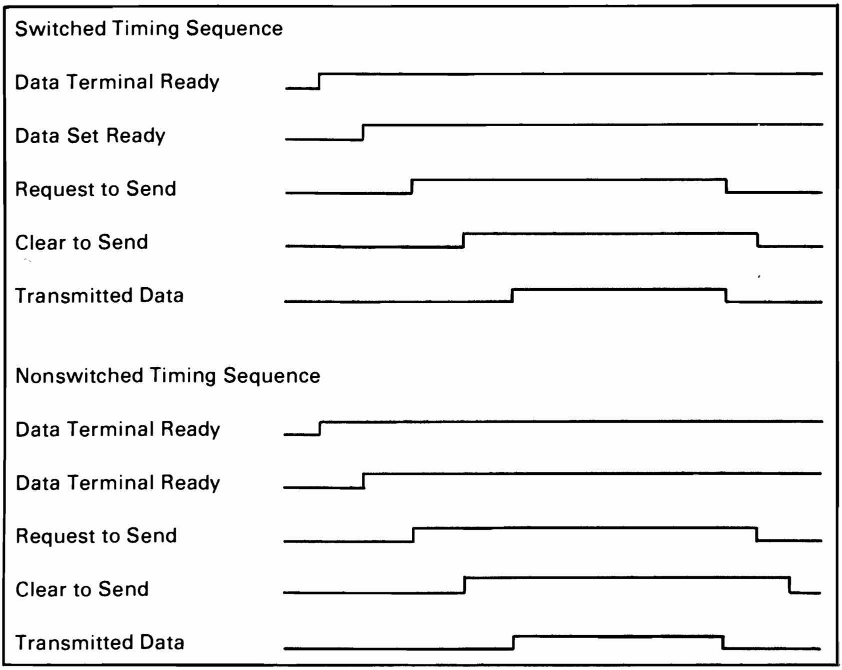
The following is an illustration of data terminal equipment connected to an external modem using connections defined by the RS-232C interface standard:



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

Establishing a Data Link

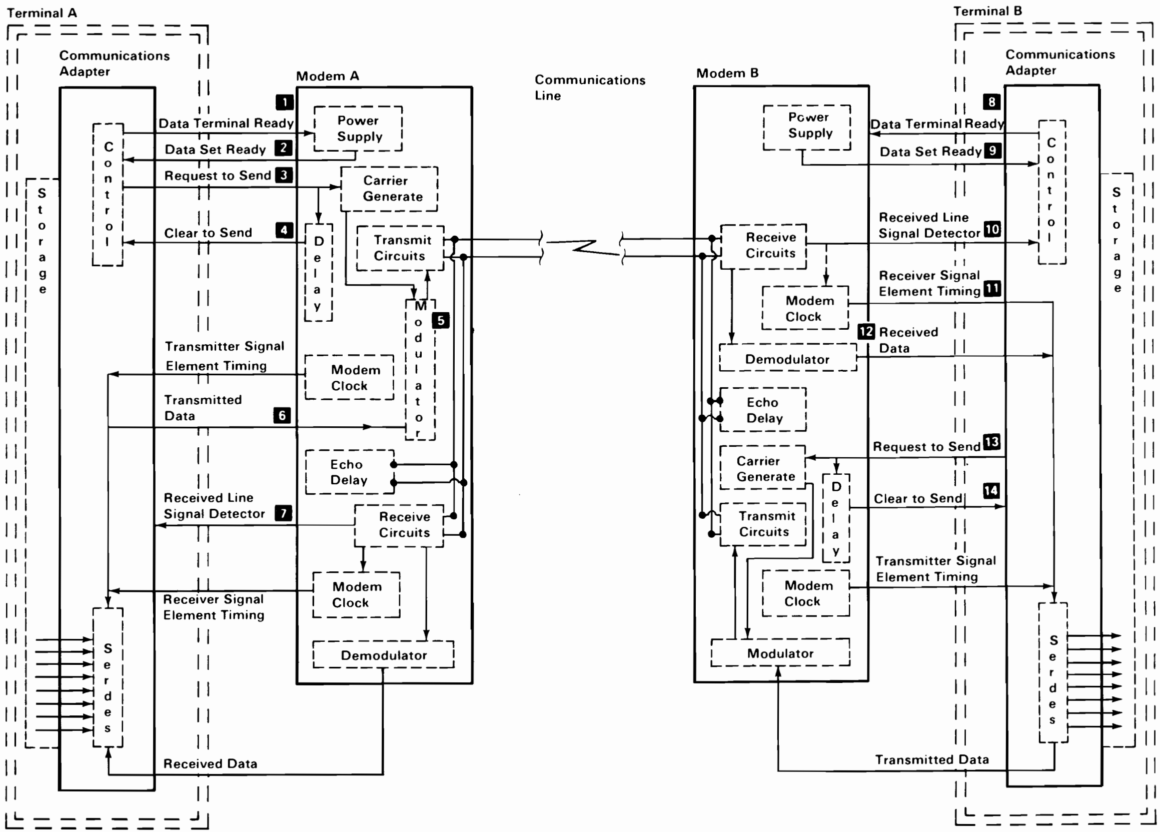
The following bar graphs represent normal timing sequences of operation during the establishment of communication for both switched (dial-up) and nonswitched (direct line) networks.



The following examples show how a link is established on a nonswitched point-to-point line, a nonswitched multipoint line, and a switched point-to-point line.

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

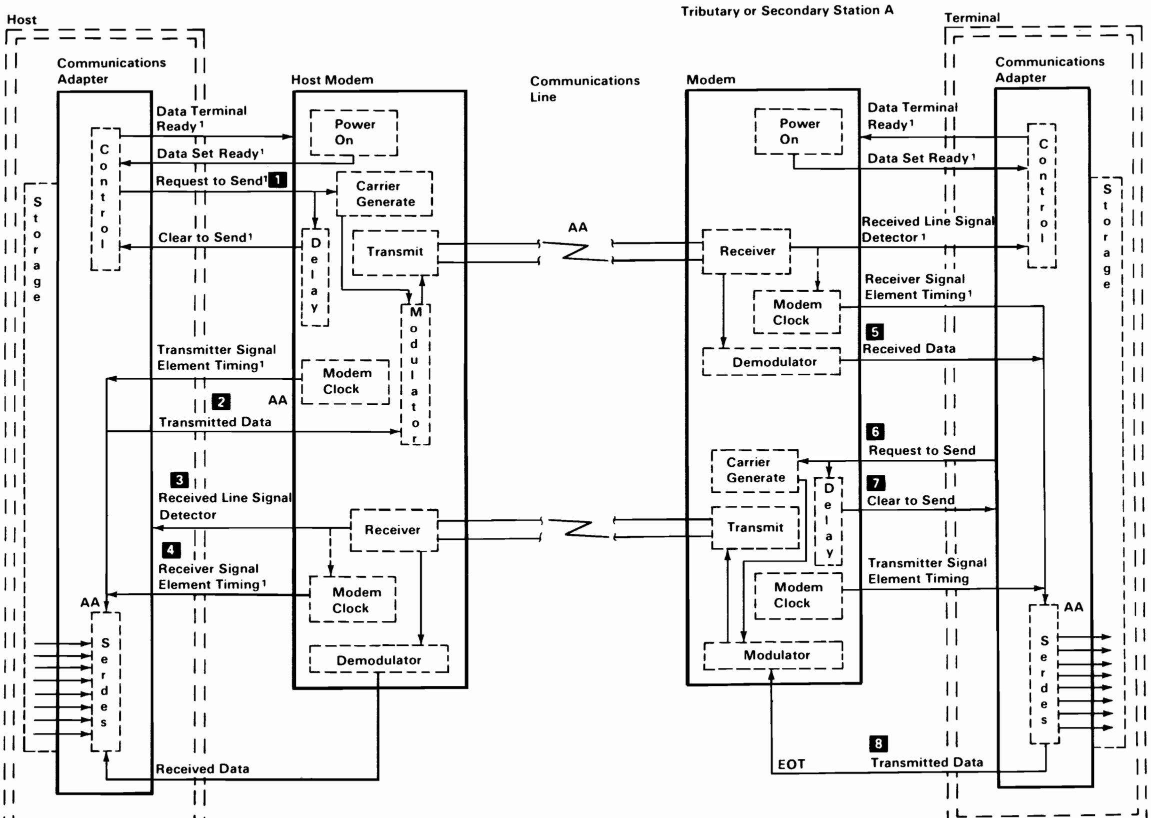
1. The terminals at both locations activate the 'data terminal ready' lines 1 and 8.
2. Normally the 'data set ready' lines 2 and 9 from the modems are active whenever the modems are powered on.
3. Terminal A activates the 'request to send' line 3, which causes the modem at terminal A to generate a carrier signal.
4. Modem B detects the carrier, and activates the 'received line signal detector' line (sometimes called data carrier detect) 10. Modem B also activates the 'receiver signal element timing' line (sometimes called receive clock) 11 to send receive clock signals to the terminal. Some modems activate the clock signals whenever the modem is powered on.
5. After a specified delay, modem A activates the 'clear to send' line 4, which indicates to terminal A that the modem is ready to transmit data.
6. Terminal A serializes the data to be transmitted (through the serdes) and transmits the data one bit at a time (synchronized by the transmit clock) onto the 'transmitted data' line 6 to the modem.
7. The modem modulates the carrier signal with the data and transmits it to the modem B 5.
8. Modem B demodulates the data from the carrier signal and sends it to terminal B on the 'received data' line 12.
9. Terminal B deserializes the data (through the serdes) using the receive clock signals (on the 'receiver signal element timing' line) 11 from the modem.
10. After terminal A completes its transmission, it deactivates the 'request to send' line 3, which causes the modem to turn off the carrier and deactivate the 'clear to send' line 4.
11. Terminal A and modem A now become receivers and wait for a response from terminal B, indicating that all data has reached terminal B. Modem A begins an echo delay (50 to 150 milliseconds) to ensure that all echoes on the line have diminished before it begins receiving. An echo is a reflection of the transmitted signal. If the transmitting modem changed to receive too soon, it could receive a reflection (echo) of the signal it just transmitted.
12. Modem B deactivates the 'received line signal detector' line 10 and, if necessary, deactivates the receive clock signals on the 'receiver signal element timing, line 11.
13. Terminal B now becomes the transmitter to respond to the request from terminal A. To transmit data, terminal B activates the 'request to send' line 13, which causes modem B to transmit a carrier to modem A.
14. Modem B begins a delay that is longer than the echo delay at modem A before turning on the 'clear to send' line. The longer delay (called request-to-send to clear-to-send delay) ensures that modem A is ready to receive when terminal B begins transmitting data. After the delay, modem B activates the 'clear to send' line 14 to indicate that terminal B can begin transmitting its response.
15. After the echo delay at modem A, modem A senses the carrier from modem B (the carrier was activated in step 13 when terminal B activated the 'request to send' line) and activates the 'received line signal detector' line 7 to terminal A.
16. Modem A and terminal A are now ready to receive the response from terminal B. Remember, the response was not transmitted until after the request-to-send to clear-to-send delay at modem B (step 14).



Terminal A
Communications Adapter
Modem A
Communications Line
Modem B
Terminal B
Communications Adapter
Storage
Control
Serdes
1
Data Terminal Ready
2
Data Set Ready
3
Request to Send
4
Clear to Send
5
6
7
Power Supply
Carrier Generate
Transmit Circuits
Delay
Modulator
Transmitter Signal Element Timing
Modem Clock
Transmitted Data
Echo Delay
Received Line Signal Detector
Receive Circuits
Modem Clock
Receiver Signal Element Timing
Demodulator
Received Data
8
Data Terminal Ready
9
Data Set Ready
Power Supply
Receive Circuits
10
Received Line Signal Detector
Modem Clock
11
Receiver Signal Element Timing
12
Received Data
Demodulator
Echo Delay
13
Request to Send
Carrier Generate
Delay
14
Clear to Send
Transmit Circuits
Transmitter Signal Element Timing
Modem Clock
Modulator
Transmitted Data
Control
Storage
Serdes

Establishing a Link on a Nonswitched Multipoint Line

1. The control station serializes the address for the tributary or secondary station (AA) and sends its address to the modem on the 'transmitted data' line 2.
2. Since the 'request to send' line and, therefore, the modem carrier, is active continuously 1, the modem immediately modulates the carrier with the address, and, thus, the address is transmitted to all modems on the line.
3. All tributary modems, including the modem for station A, demodulate the address and send it to their terminals on the 'received data' line 5.
4. Only station A responds to the address; the other stations ignore the address and continue monitoring their 'received data' line. To respond to the poll, station A activates its 'request to send' line 6 which causes the modem to begin transmitting a carrier signal.
5. The control station's modem receives the carrier and activates the 'received line signal detector' line 3 and the 'receiver signal element timing' line 4 (to send clock signals to the control station). Some modems activate the clock signals as soon as they are powered on.
6. After a short delay to allow the control station modem to receive the carrier, the tributary modem activates the 'clear to send' line 7.
7. When station A detects the active 'clear to send' line, it tansmits its response. (For this example, assume that station A has no data to send; therefore, it transmits an EOT 8.)
8. After transmitting the EOT, station A deactivates the 'request to send' line 6. This causes the modem to deactivate the carrier and the 'clear to send' line 7.
9. When the modem at the control station (host) detects the absence of the carrier, it deactivates the 'received line signal detector' line 3.
10. Tributary station A is now in receive mode waiting for the next poll or select transmission from the control station.

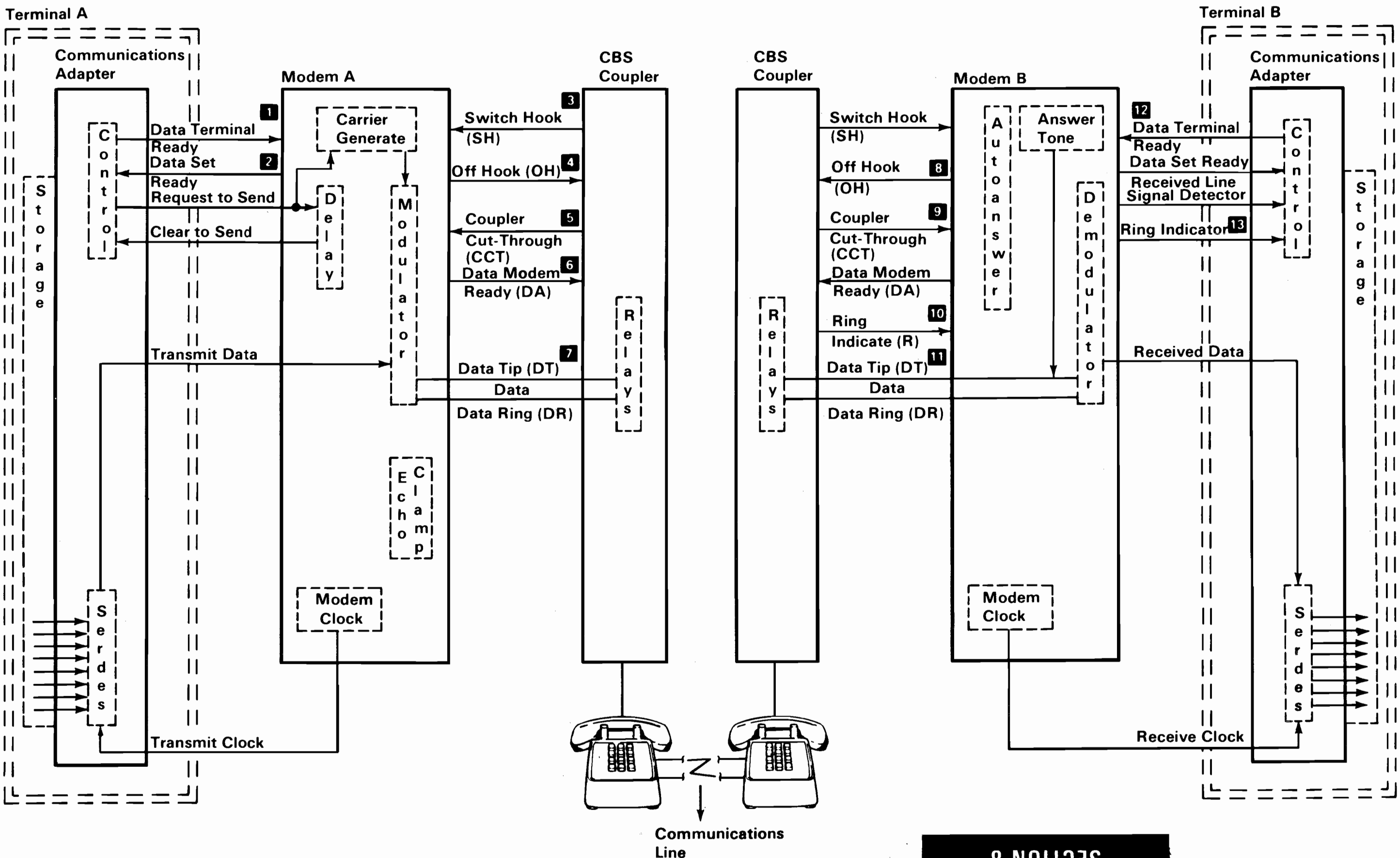


¹These lines are active continuously.

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

1. Terminal A is in communications mode; therefore, the 'data terminal ready' line **1** is active. Terminal B is in communication mode waiting for a call from terminal A.
2. When the terminal A operator lifts the telephone handset, the 'switch hook' line from the coupler is activated **3**.
3. Modem A detects the 'switch hook' line and activates the 'off hook' line **4**, which causes the coupler to connect the telephone set to the line and activate the 'coupler cut-through' line **5** to the modem.
4. Modem A activates the 'data modem ready' line **6** to the coupler (the 'data modem ready' line is on continuously in some modems).
5. The terminal A operator sets the exclusion key or talk/data switch to the talk position to connect the handset to the communications line. The operator then dials the terminal B number.
6. When the telephone at terminal B rings, the coupler activates the 'ring indicate' line to modem B **10**. Modem B indicates that the 'ring indicate' line was activated by activating the 'ring indicator' line **13** to terminal B.
7. Terminal B activates the 'data terminal ready' line to modem B **12**, which activates the autoanswer circuits in modem B. (The 'data terminal ready' line might already be active in some terminals.)
8. The autoanswer circuits in modem B activate the 'off hook' line to the coupler **8**.
9. The coupler connects modem B to the communications line through the 'data tip' and 'data ring' lines **11** and activates the 'coupler cut-through' line **9** to the modem. Modem B then transmits an answer tone to terminal A.
10. The terminal A operator hears the tone and sets the exclusion key or talk/data switch to the data position (or performs an equivalent operation) to connect modem A to the communications line through the 'data tip' and 'data ring' lines **7**.
11. The coupler at terminal A deactivates the 'switch hook' line **3**. This causes modem A to activate the 'data set ready' line **2** indicating to terminal A that the modem is connected to the communications line.

The sequence of the remaining steps to establish the data link is the same as the sequence required on a nonswitched point-to-point line. When the terminals have completed their transmission, they both deactivate the 'data terminal ready' line to disconnect the modems from the line.



Terminal A
Communications Adapter
Control
Storage
Serdes
Data Terminal Ready
Data Set Ready
Request to Send
Clear to Send
Transmit Data
Transmit Clock
Modem A
Carrier Generate
Delay
Modulator
Echo Clamp
Modem Clock
Switch Hook (SH)
Off Hook (OH)
Coupler Cut-Through (CCT)
Data Modem Ready (DA)
Data Tip (DT)
Data
Data Ring (DR)
CBS Coupler
Relays
Communications Line
Terminal B
Modem B
Autoanswer
Answer Tone
Demodulator
Ring Indicate (R)
Data Terminal Ready
Data Set Ready
Received Line Signal Detector
Ring Indicator
Received Data
Receive Clock
1
2
3
4
5
6
7
8
9
10
11
12
13

Notes:

SECTION 9. IBM PERSONAL COMPUTER COMPATIBILITY

Contents

Notes:

This section shows the differences between the IBM Personal Computer AT and the rest of the IBM Personal Computer family. It also contains information necessary to design hardware and programs that will be compatible with all IBM Personal Computers.

Hardware Considerations

In order to design compatible hardware or programs, hardware differences between the IBM Personal Computers must be considered. The following are hardware features of the IBM Personal Computer AT that are not supported by the rest of the IBM Personal Computer Family.

System Board

The IBM Personal Computer AT system board uses an Intel 80286 microprocessor which is generally compatible with the Intel 8088 microprocessor used in the rest of the IBM Personal Computers. Programming considerations because of the faster processing capability of the 80286 are discussed later in "Application Guidelines."

The system board expansion slots in the IBM Personal Computer AT have a 36-pin connector in addition to the 62-pin connector. Adapters designed to make use of the 36-pin connector are not compatible with the rest of the IBM Personal Computers.

On the I/O channel:

- The system clock signal should only be used for synchronization and not for applications requiring a fixed frequency.
- The 14.31818 MHz oscillator is not synchronous with the system clock.
- 'ALE' is activated during DMA cycles.

SECTION 9

- The 'I/O write' signal is not active during refresh cycles.
- Pin B04 supports IRQ 9.

20Mb Fixed Disk Drive

The fixed disk drive used in the IBM Personal Computer AT can store up to 20Mb of data. Reading from and writing to this drive is initiated in the same way as with the Personal Computer XT; however, the IBM Personal Computer AT Fixed Disk and Diskette Drive Adapter may be addressed from different BIOS locations.

High Capacity Diskette Drive

This diskette drive is capable of reading and writing diskettes in 160/180Kb, 320/360Kb, and 1.2Mb mode. However, if a diskette, formatted in either the 160/180Kb or 320/360Kb mode is written on by this diskette drive, that information may only be read by a high capacity diskette drive.

> **Note:** Diskettes, designed for use in this drive, in the 1.2Mb mode may not be used in either a 160/180Kb or a 320/360Kb diskette drive.

Adapters

The IBM Personal Computer AT 128KB Memory Expansion Option, the IBM Personal Computer AT 512KB Memory Expansion Option, the IBM Personal Computer AT Prototype Adapter, and the IBM Personal Computer AT Fixed Disk and Diskette Drive Adapter use the additional 36 pin system board expansion slot and are not compatible with the rest of the IBM Personal Computer Family.

Keyboard

The IBM Personal Computer AT Keyboard is an 84-key unit that can perform all functions of the other IBM Personal Computer keyboards, but is not plug-compatible with any of the other keyboards.

The IBM Personal Computer AT Does Not Support

- Expansion Unit
- IBM Asynchronous Communications Adapter
- IBM 64/256KB Memory Expansion Adapter
- IBM Printer Adapter
- Other keyboards

SECTION 9

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, COBAL, 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").

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 "Assembly Language Programming" 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 the IBM Personal Computer AT than they do on other IBM Personal Computers.

- If the system microprocessor executes a POPF instruction in either the real or the virtual address mode with CPL≤IOPL, then a pending maskable interrupt (the INTR pin active) may be improperly recognized after executing the POPF instruction even if maskable interrupts were disabled before the POPF instruction and the value popped had IF=0. If the interrupt is improperly recognized, the interrupt is still correctly executed. This errata has no effect when interrupts are enabled in either real or virtual address mode. This errata has no effect in the virtual address mode when CPL>IOPL.

  The POPF instruction may be simulated with the following code macro:

```
POPFF       Macro        ;use POPFF instead of POPF
                         ;simulate popping flags
                         ;using IRET
EB 01       JMP $+3      ;jump around IRET
CF          IRET         ;POP CS, IP, flags
0E          PUSH CS      ;push CS
E8 FB FF    CALL $-2     ;CALL within segment
                         ;program will continue here
```

- PUSH SP pushes the current stack pointer. The microprocessor used in the IBM Personal Computer and the IBM Personal Computer XT pushes the new stack pointer.

- Single step interrupt (when TF=1) does not occur on the interrupt instruction (OP code hex CC,CD). The microprocessor in the IBM Personal Computer and the IBM Personal Computer XT does interrupt on the INT instruction.

- The divide error exception (interrupt 0) pushes the CS:IP of the instruction, causing the exception. The IBM Personal Computer and the IBM Personal Computer XT push the CS:IP following the instruction, causing the exception.

- Shift counts are masked to 5 bits. Shift counts greater than 31 are treated mod 32, that is, a shift count of 36 shifts the operand 4 places.

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

- Program interrupts are used for access to these functions. This practice removes the absolute addressing from the program. Only the interrupt number is required.

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

- Back to back I/O commands to the same I/O ports will not permit enough recovery time for I/O chips. To insure 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 the IBM Personal Computer AT IRQ 9 is redirected to INT hex 0A (hardware IRQ 2). This insures that hardware designed to use IRQ 2 will operate in the IBM Personal Computer AT.

- The system can mask hardware sensitivity. New devices can change the ROM BIOS to accept the same programming interface on the new device.

- In cases where BIOS provides parameter tables, such as for video or diskette, a program may substitute new parameter values by building a new copy of the table and changing the vector to point to that table. However, the program should copy the current table, using the current vector, and then modify those locations in the table that need to be changed. In this way, the program will not inadvertently change any values that should be left the same.

- Disk_Base consists of 11 parameters required for diskette operation. They are pointed at by the data variable, Disk_Pointer, at absolute address 0:78. It is strongly recommended that the values supplied in ROM be used. If it becomes necessary to modify any of the parameters, build another parameter block and modify the address in Disk–Pointer to point to the new block. The parameters were established to operate both the High Capacity Diskette Drive and the Double Sided Diskette Drive. Three of the parameters in this table are under control of BIOS in the following situations. The Gap Length Parameter is no longer retrieved from the parameter block. Gap length used during diskette read, write, and verify operations is derived from within diskette BIOS. Gap length for format operations is still obtained from the parameter block. Special considerations are required for formatting operations. See the prologue 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 or 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. If a successful read occurs, a high capacity diskette is in the drive.

  3. If Step 1 fails, issue the reset function (AH=0) to diskette BIOS and retry. If a successful read cannot be done, the media needs to be formatted or is defective.

ROM BIOS and DOS do not provide for all functions. The following are the allowable I/O operations with which IBM will maintain compatibility in future systems.

- Control of the sound using port hex 61, and the sound channel of the timer/counter. A program can control timer/counter channels 0 and 2, ports hex 40, 42, and 43. A program must not change the value in port hex 41, because this port controls the dynamic-memory refresh. Channel 0 provides the time-of-day interrupt, and can also be used for timing short intervals. Channel 2 of the timer/counter is the output for the speaker and cassette ports. This channel may also be used for timing short intervals, although it cannot interrupt at the end of the period.

- Control of the Game Control Adapter, port hex 201

  Note: Programs should use the timer for delay on the paddle input rather than a program loop.

- Interrupt Mask Register (IMR), port hex 21, can be used to selectively mask and unmask the hardware features.

The following information pertains to absolute memory locations.

- Interrupt Vectors (hex 0)--A program may change these to point at different processing routines. When an interrupt vector is modified, the original value should be retained. If the interrupt, either hardware or program, is not directed toward this device handler, the request should be passed to the next item in the list.

- Video Display Buffers (hex B0000 and B8000)-- For each mode of operation defined in the video display BIOS, the memory map will remain the same. For example, the bit map for the 320 x 200 medium-resolution graphics mode of the Color/Graphics Monitor adapter will be retained on any future adapter that supports that mode. If the bit map is modified, a different mode number will be used.

- ROM BIOS Data Area (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 under 'Copy Protection' later in this section.

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.

Multi-tasking Provisions

The IBM Personal Computer AT BIOS contains a feature to assist multi-tasking implementation. "Hooks" are provided for a multi-tasking dispatcher. Whenever a busy (wait) loop occurs in the BIOS, a hook is provided for the system to break out of the loop. Also, whenever an interrupt is serviced by the BIOS, which causes a corresponding wait loop to be exited, another hook is provided for the system.

Thus a system may be written which employs the bulk of the device driver code. The following is valid only in the microprocessor's real address mode. Several steps must be taken by the system code in order to allow this support. First, the system is responsible for the serialization of access to the device driver. The BIOS code is not reentrant. Second, the system is responsible for matching corresponding wait and post calls.

Interfaces

There are four interfaces to be used by the multi-tasking dispatcher:

Startup

The first thing to be done is for the startup code to hook interrupt hex 15. The dispatcher is responsible to check for function codes AH = hex 90 and 91. The "Wait" and "Post" sections describe these codes. The dispatcher must pass all other functions through to the previous user of interrupt hex 15. This can be done via a JMP or a CALL. If the function code is hex 90 or 91, then the dispatcher should do the appropriate processing and return via the IRET instruction.

Serialization

It is up to the multi-tasking system to insure that the device driver code is used in a serial fashion. Multiple entries into the code can result in very serious errors.

Wait (Busy)

Whenever the BIOS is about to enter a busy loop, it first issues an interrupt 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 is an outline of the code which has been added to the BIOS to implement this function.

```
EXAMPLE DEVICE BUSY LOOP

      DO UNTIL
```

MOV AX, hex 90XX ;WAIT code in AH and

```
                              ;TYPE code in AL
INT hex 15                    ;issue call
JC TIMEOUT                    ;optional: for timeout or
                              ;if carry is set, timeout
                              ;occurred
NORMAL TIMEOUT LOGIC          ;normal timeout

     UNTIL INTERRUPT COMPLETE FLAG IS SET
```

POST (Interrupt)

Whenever the BIOS has set an interrupt flag for a corresponding busy loop, an interrupt 15 occurs with a function code 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 BIOS has been added to code to implement this function.

```
INTERRUPT PROCESSING

SET INTERRUPT COMPLETE FLAG FOR BUSY LOOP


MOV AX,hex 91XX               ; post code AH and
                              ; type code AL
INT hex 15                    ; issue call
```

Classes

The following types of wait loops are supported:

- The class for 0->7Fh 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 80h->BFh is reentrant. There is no restriction on the number of tasks which may access the device.
- The class for C0h->FFh 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->0FFh	wait only calls; there is no complementary "POST" for these waits--these are timeout only. Times are function number dependent.

Function Code Assignments

The following are specific assignments for the IBM Personal Computer AT BIOS. They are grouped according to the classes described under "Function Code Classes".

Type Code (AL)	Timeout	Description
00H	yes (6 sec)	IBM Personal Computer AT fixed disk

01H	yes (2 sec)	IBM Personal Computer AT diskette
02H	no	IBM Personal Computer AT keyboard
0FDH	yes (1 sec-write)	diskette motor start
--	(625 msec-read)	--
0FEH	yes (?? sec)	printer

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

Timeouts

In order to support timeouts properly, it is necessary for the multi-tasking dispatcher to 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 timeout occurred.

SYS REQ Key

The following describes the use of the SYS REQ key in a multi-tasking environment. It assumes that tasks used are cooperative in some manner. The system must employ a task monitor to allow the user to select various tasks. This selection may be for starting tasks, terminating tasks, supplying input to tasks from the keyboard, or any other function that requires user input.

Subsystem Structure

The following figure shows three subsystems which have multiple tasks. They are arranged in order of hierarchy. Tasks in subsystem B can only run when Task "Other" A is active in subsystem A and tasks in subsystem C can only run when Task "Other" B is active in subsystem B.

<table>
<tr><td>Task 1A</td><td>Task 2A</td><td>Task 3A</td><td colspan="3">Task "Other" A</td></tr>
<tr><td colspan="3">Subsystem B Inhibited</td><td>Task 1B</td><td>Task 2B</td><td rowspan="2">Task B
"Other"
Task 1C
Task 2C</td></tr>
<tr><td colspan="5">Subsystem C Inhibited</td></tr>
</table>

Multiple Task Subsystems

The order in which subsystems were installed (loaded into main storage) determines their priority. The first one installed is higher on the hierarchy. An inhibit mechanism provided at startup time enforces the hierarchy. As a subsystem starts, it broadcasts to the rest of the subsystems, previously installed, that it is starting and at the same time, provides the address of a lock. This lock must be set (incremented) by subsystems higher in the hierarchy whenever they wish to run one of their own tasks. This flag must be set for each subsystem lower on the hierarchy, for example, when subsystem A is about to start Task 2A, the dispatcher must set subsystem B inhibit and subsystem C inhibit.

Subsystem Startup and Lockout

In order for multiple subsystems to cooperate, there must be communication between subsystems when a subsystem is loaded into storage and initialized.

The subsystem being loaded tells the previously loaded subsystems that it is being loaded and broadcasts the address of its synchronization lock. Higher priority subsystems use this lock to exclude the new subsystem from accessing any system resources (DOS, interrupts, etc.).

After a subsystem is loaded, it must "listen" for any subsystems that may be loaded later so that it can lock them out when it is running. The following describes the code sequence for startup.

Startup Interface

MOV AX,SEG SYSLOCK	;segment of lock
MOV ES,AX	--
MOV BX,OFFSET SYSLOCK	;offset of lock
MOV AX,2000H	;AH=20H, AL=0
INT 15H	--

Lockout Interface

The register ES:BX points to a byte which initially contains a value of 0. Whenever a higher priority subsystem wishes to run, it increments the lock. When it completes running, it decrements the lock. This allows proper synchronization of resources and subsystems.

SYS REQ Key Functions

During initialization, the subsystem also needs to connect to the SYS REQ key function. It is necessary for the SYS key code to be included in each subsystem. This startup section determines if the SYS support is already loaded and loads the support if necessary.

The SYS functions provide a means for the subsystem's main screen or menu to be displayed. If the subsystem requires no user action, then these functions need not be provided.

SYS Key Modes

There are two SYS key modes: multiple press and super shift.

Multiple Press Mode: This mode allows the user to sequence through subsystems. Subsystems are displayed in the reverse order of their installation.

Super Shift Mode: This mode allows the user direct access to any subsystem regardless of the priority. The user activates this mode by holding the SYS key pressed and pressing another key which designates another subsystem.

Multiple Key Sequence

If a subsystem is to be used on the IBM Personal Computer and the IBM Personal Computer XT, a multiple key sequence must be used to access the SYS key functions.

SYS Key Interfaces

There are four interfaces needed by the SYS code to support a subsystem: startup, activation, cancellation, and completion. The subsystem activates two of these: startup and completion. The SYS code in conjunction with user input activates the other two.

The following is a description, in tabular form, of the states, transitions, and actions needed to implement the SYS REQ functions.

Subsystem Entry Points

subsys A subsys B subsys C	code A code B code C

Entry Points

```
# subsystems          current subsystem #

  num                       cur
```

State/Transition Table

Current State	Input	Next State	Action
Idle	SYS REQ	Active	activate subsys 'cur'
	SYS code	Active Super	activate subsys 'code'
	Startup	Idle	increment 'num'
			set 'cur' to 'num'
			insert entry point and code
Active	SYS REQ	Active	cancel subsys 'cur'
			decrement 'cur'
			activate subsys 'cur'
	Completion 'cur'	Idle	set 'cur' to 'num'
	Startup	Active	increment 'num'
			insert entry point and code
	SYS code	Active Super	activate subsys 'code'
Active Super	Completion 'cur'	Idle	set 'cur' to 'num'
	Startup	Active	increment 'num'
			insert entry point and code

SECTION 9

Startup

At startup, a call is issued to determine if the SYS REQ key support is already loaded and to initialize the support for the new subsystem.

The parameters for the startup routine are the address of the entry point and the function code (direct-access mode). If the operation was successful, the carry flag is set.

The following shows the calling sequence.

MOV AX,SEG entry__point	;address for SYS to call
MOV ES,AX	;
MOV BX,OFFSET entry__point	;
MOV CX,XXXX	;super shift mode code
MOV AX,2010H	;AH=20H, AL=10
INT 15H	;

If the carry flag is not set, the initialization code needs to hook the vector for interrupt 15H, save the previous address, and reissue the initialization call.

Activation

This is a signal from the SYS REQ processing module that a subsystem's monitor is to be activated.

This entry into the subsystem dispatcher signals that the monitor task should be activated. It should be treated as a signal to set a flag for the subsystem rather than an opportunity to gain control of the system asynchronously as it may not be a proper time for the subsystem to run. The subsystem may have to wait until a higher priority subsystem allows it to have control before the subsystem's monitor gets control. The subsystem entry point is CALLED with the AH register set to 0.

Cancellation

This signal from the SYS REQ processing module tells the subsystem monitor to ignore the previous activation signal and take the necessary action to return to its previous state.

This entry into the subsystem dispatcher signals that the monitor task should be deactivated. The subsystem may not have control of the system. It is necessary for the subsystem to note that a cancellation has occurred and to wait until it has a valid opportunity to run through its dispatcher code in a normal fashion. The subsystem entry point is CALLED with the AH register set to 1.

Completion

The following call signals completion. Completion constitutes any action taken by the user when the subsystem's menu is displayed.

The completion call causes the activation pointer to be reset to the lowest priority subsystem. All lower priority subsystems also receive a cancellation notification.

```
MOV AX,SEG entry_point         ;address for SYS to call

MOV ES,AX                      ;

MOV BX,OFFSET entry_point      ;ES:BX must contain the same

                               ;values as the startup call

MOV AX,2011H                   ;AH=20H, AL=11H

INT 15H                        ;
```

SECTION 9

Copy Protection

Some modes of copy protection will not work on the IBM Personal Computer AT due to the following conditions:

- Bypassing BIOS
- Diskette drive differences
- Write current differences

Bypassing BIOS

Copy protection, which depends on the following will not work on the IBM Personal Computer AT:

Track Density: The High Capacity Diskette Drive records tracks at a density of 96TPI. This drive has to double step in the 48TPI 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 Differences

Copy protection, which depends on the following will not work on the High Capacity Diskette Drive:

Rotational Speed: Copy protection using the time between two events on a diskette will not work on the High Capacity Diskette Drive.

Access Time: Diskette BIOS must set the track to track access time for the different types of media used on the IBM Personal Computer AT.

Head Geometry: See 'High Capacity Diskette Drive' earlier in this section.

Diskette Change Signal: Copy protection may not be able to reset this signal.

Write Current

The IBM Personal Computer AT Fixed Disk and Diskette Drive Adapter selects the proper write current for the media being used.

Machine-Sensitive Code

Programs may program for machine specific features, but they must test for specific machine type. Location hex 0FFFF:0E contains the machine identification:

Hex	Machine Identification
0FF	IBM Personal Computer
0FE	IBM Personal Computer XT
0FD	IBM PC*jr*
0FC	IBM Personal Computer AT

Machine Identification Code

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

Notes:

Glossary

μ. Prefix micro; 0.000 001.

μs. Microsecond; 0.000 001 second.

A. Ampere.

ac. Alternating current.

accumulator. A register in which the result of an operation is formed.

active high. Designates a signal that has to go high to produce an effect. Synonymous with positive true.

active low. Designates a signal that has to go low to produce an effect. Synonymous with negative true.

adapter. An auxiliary device or unit used to extend the operation of another system.

address bus. One or more conductors used to carry the binary-coded address from the processor throughout the rest of the system.

algorithm. A finite set of well-defined rules for the solution of a problem in a finite number of steps.

all points addressable (APA). A mode in which all points of a displayable image can be controlled by the user.

alphameric. Synonym for alphanumeric.

alphanumeric (A/N). Pertaining to a character set that contains letters, digits, and usually other characters, such as punctuation marks. Synonymous with alphameric.

alternating current (ac). A current that periodically reverses its direction of flow.

American National Standard Code for Information Exchange (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 Exchange.

assemble. To translate a program expressed in an assembler language into a computer language.

assembler. A computer program used to assemble.

assembler language. A computer-oriented language whose instructions are usually in one-to-one correspondence with computer instructions.

asynchronous transmission. (1) Transmission in which the time of occurrence of the start of each character, or block of characters, is arbitrary; once started, the time of occurrence of each signal representing a bit within a character, or block, has the same relationship to significant instants of a fixed time frame. (2) Transmission in which each information character is individually transmitted (usually timed by the use of start elements and stop elements).

audio frequencies. Frequencies that can be heard by the human ear (approximately 15 hertz to 20 000 hertz).

auxiliary storage. (1) A storage device that is not main storage. (2) Data storage other than main storage; for example, storage on magnetic disk. (3) Contrast with main storage.

BASIC. Beginner's all-purpose symbolic instruction code.

basic input/output system (BIOS). The feature of the IBM Personal Computer that provides the level control of the major I/O devices, and relieves the programmer from concern about hardware device characteristics.

baud. (1) A unit of signaling speed equal to the number of discrete conditions or signal events per second. For example, one baud equals one bit per second in a train of binary signals, one-half dot cycle per second in Morse code, and one 3-bit value per second in a train of signals each of which can assume one of eight different states. (2) In asynchronous transmission, the unit of modulation rate corresponding to one unit of interval per second; that is, if the duration of the unit interval is 20 milliseconds, the modulation rate is 50 baud.

BCC. Block-check character.

beginner's all-purpose symbolic instruction code (BASIC). A programming language with a small repertoire of commands and a simple syntax, primarily designed for numeric applications.

binary. (1) Pertaining to a selection, choice, or condition that has two possible values or states. (2) Pertaining to a fixed radix numeration system having a radix of 2.

binary digit. (1) In binary notation, either of the characters 0 or 1. (2) Synonymous with bit.

binary notation. Any notation that uses two different characters, usually the binary digits 0 and 1.

binary synchronous communications (BSC). A uniform procedure, using a standardized set of control characters and control character sequences for synchronous transmission of binary–coded data between stations.

BIOS. Basic input/output system.

bit. Synonym for binary digit

bits per second (bps). A unit of measurement representing the number of discrete binary digits transmitted by a device in one second.

block. (1) A string of records, a string of words, or a character string formed for technical or logic reasons to be treated as an entity. (2) A set of things, such as words, characters, or digits, treated as a unit.

block-check character (BCC). In cyclic redundancy checking, a character that is transmitted by the sender after each message block and is compared with a block-check character computed by the receiver to determine if the transmission was successful.

boolean operation. (1) Any operation in which each of the operands and the result take one of two values. (2) An operation that follows the rules of boolean algebra.

bootstrap. A technique or device designed to bring itself into a desired state by means of its own action; for example, a machine routine whose first few instructions are sufficient to bring the rest of itself into the computer from an input device.

bps. Bits per second.

BSC. Binary synchronous communications.

buffer. (1) An area of storage that is temporarily reserved for use in performing an input/output operation, into which data is read or from which data is written. Synonymous with I/O area. (2) A portion of storage for temporarily holding input or output data.

bus. One or more conductors used for transmitting signals or power.

byte. (1) A sequence of eight adjacent binary digits that are operated upon as a unit. (2) A binary character operated upon as a unit. (3) The representation of a character.

C. Celsius.

capacitor. An electronic circuit component that stores an electric charge.

CAS. Column address strobe.

cathode ray tube (CRT). A vacuum tube in which a stream of electrons is projected onto a fluorescent screen producing a luminous spot. The location of the spot can be controlled.

cathode ray tube display (CRT display). (1) A CRT used for displaying data. For example, the electron beam can be controlled to form alphanumeric data by use of a dot matrix. (2) The data display produced by the device as in (1).

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.

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.

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.

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 intervention by a human operator 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.

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 a 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. An electronic means of overcoming the make/break bounce of switches to obtain one smooth change of signal level.

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 unit.

disk drive. A mechanism for moving a disk pack and controlling its movements.

disk pack. A removable assembly of magnetic disks.

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 mechanism for moving a diskette and controlling its movements.

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.

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.

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 memory 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.

fixed disk. In the IBM Personal Computer, synonym for disk drive.

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.

frame. (1) In SDLC, the vehicle for every command, every response, and all information that is transmitted using SDLC procedures. Each frame begins and ends with a flag. (2) In data transmission, the sequence of contiguous bits bracketed by and including beginning and ending flag sequences.

g. Gram.

G. (1) Prefix giga; 1 000 000 000. (2) When referring to computer storage capacity, 1 073 741 824. (1 073 741 824 = 2 to the 30th power.)

gate. (1) A combinational logic circuit having one output channel and one or more input channels, such that the output channel state is completely determined by the input channel states. (2) A signal that enables the passage of other signals through a circuit.

Gb.

1 073 741 824 bytes.

general-purpose register. A register, usually explicitly addressable within a set of registers, that can be used for different purposes; for example, as an accumulator, as an index register, or as a special handler of data.

giga (G). Prefix 1 000 000 000.

gram (g). A unit of weight (equivalent to 0.035 ounces).

graphic. A symbol produced by a process such as handwriting, drawing, or printing.

graphic character. A character, other than a control character, that is normally represented by a graphic.

half-duplex. (1) In data communication, pertaining to an alternate, one way at a time, independent transmission. (2) Contrast with duplex.

hardware. (1) Physical equipment used in data processing, as opposed to programs, procedures, rules, and associated documentation. (2) Contrast with software.

head. A device that reads, writes, or erases data on a storage medium; for example, a small electromagnet used to read, write, or erase data on a magnetic disk.

hertz (Hz). A unit of frequency equal to one cycle per second.

hex. Common abbreviation for hexadecimal.

hexadecimal. (1) Pertaining to a selection, choice, or condition that has 16 possible different values or states. These values or states are usually symbolized by the ten digits 0 through 9 and the six letters A through F. (2) Pertaining to a fixed radix numeration system having a radix of 16.

high impedance state. A state in which the output of a device is effectively isolated from the circuit.

highlighting. In computer graphics, emphasizing a given display group by changing its attributes relative to other display groups in the same display field.

high-order position. The leftmost position in a string of characters. See also most-significant digit.

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 "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.

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.

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.

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.

m. (1) Prefix milli; 0.001. (2) Meter.

M. (1) Prefix mega; 1 000 000. (2) When referring to computer storage capacity, 1 048 576. (1 048 576 = 2 to the 20th power.)

mA. Milliampere; 0.001 ampere.

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

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

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

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

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

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

masked. Synonym for disabled.

matrix. (1) A rectangular array of elements, arranged in rows and columns, that may be manipulated according to the rules of matrix algebra. (2) In computers, a logic network in the form of an array of input leads and output leads with logic elements connected at some of their intersections.

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

Mb. 1 048 576 bytes.

mega (M). Prefix 1 000 000.

megahertz (MHz). 1 000 000 hertz.

memory. Term for main storage.

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

MFM. Modified frequency modulation.

MHz. Megahertz; 1 000 000 hertz.

micro (μ). Prefix 0.000 001.

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

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

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

microsecond (μs). 0.000 001 second.

milli (m). Prefix 0.001.

milliampere (mA). 0.001 ampere.

millisecond (ms). 0.001 second.

mnemonic. A symbol chosen to assist the human memory; for example, an abbreviation such as "mpy" for "multiply".

mode. (1) A method of operation; for example, the binary mode, the interpretive mode, the alphanumeric mode. (2) The most frequent value in the statistical sense.

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.

monitor. (1) A device that observes and verifies the operation of a data processing system and indicates any significant departure from the norm. (2) Software or hardware that observes, supervises, controls, or verifies the operations of a system.

most-significant digit. The leftmost (non-zero) digit. See also high-order position.

ms. Millisecond; 0.001 second.

multiplexer. A device capable of interleaving the events of two or more activities, or capable of distributing the events of an interleaved sequence to the respective activities.

multiprogramming. (1) Pertaining to the concurrent execution of two or more computer programs by a computer. (2) A mode of operation that provides for the interleaved execution of two or more computer programs by a single processor.

n. Prefix nano; 0.000 000 001.

NAND. A logic operator having the property that if P is a statement, Q is a statement, R is a statement,..., then the NAND of P, Q ,R,... is true if at least one statement is false, false if all statements are true.

NAND gate. A gate in which the output is 0 only if all inputs are 1.

nano (n). Prefix 0.000 000 001.

nanosecond (ns). 0.000 000 001 second.

negative true. Synonym for active low.

negative-going edge. The edge of a pulse or signal changing in a negative direction. Synonymous with falling edge.

non-return-to-zero change-on-ones recording (NRZI). A transmission encoding method in which the data terminal equipment changes the signal to the opposite state to send a binary 1 and leaves it in the same state to send a binary 0.

non-return-to-zero (inverted) recording (NRZI). Deprecated term for non-return-to-zero change-on-ones recording.

NOR. A logic operator having the property that if P is a statement, Q is a statement, R is a statement,..., then the NOR of P, Q, R,... is true if all statements are false, false if at least one statement is true.

NOR gate. A gate in which the output is 0 only if at least one input is 1.

NOT. A logical operator having the property that if P is a statement, then the NOT of P is true if P is false, false if P is true.

NRZI. Non-return-to-zero change-on-ones recording.

ns. Nanosecond; 0.000 000 001 second.

NUL. The null character.

null character (NUL). A control character that is used to accomplish media-fill or time-fill, and that may be inserted into or removed from, a sequence of characters without affecting the meaning of the sequence; however, the control of the equipment or the format may be affected by this character.

odd-even check. Synonym for parity check.

offline. Pertaining to the operation of a functional unit without the continual control of a computer.

one-shot. A circuit that delivers one output pulse of desired duration for each input (trigger) pulse.

open circuit. (1) A discontinuous circuit; that is, one that is broken at one or more points and, consequently, cannot conduct current. Contrast with closed circuit. (2) Pertaining to a no-load condition; for example, the open-circuit voltage of a power supply.

open collector. A switching transistor without an internal connection between its collector and the voltage supply. A connection from the collector to the voltage supply is made through an external (pull-up) resistor.

operand. (1) An entity to which an operation is applied. (2) That which is operated upon. An operand is usually identified by an address part of an instruction.

operating system. Software that controls the execution of programs; an operating system may provide services such as resource allocation, scheduling, input/output control, and data management.

OR. A logic operator having the property that if P is a statement, Q is a statement, R is a statement,..., then the OR of P, Q, R,...is true if at least one statement is true, false if all statements are false.

OR gate. A gate in which the output is 1 only if at least one input is 1.

output. Pertaining to a device, process, or channel involved in an output process, or to the data or states involved in an output process.

output process. (1) The process that consists of the delivery of data from a data processing system, or from any part of it. (2) The return of information from a data processing system to an end user, including the translation of data from a machine language to a language that the end user can understand.

overcurrent. A current of higher than specified strength.

overflow indicator. (1) An indicator that signifies when the last line on a page has been printed or passed. (2) An indicator that is set on if the result of an arithmetic operation exceeds the capacity of the accumulator.

overrun. Loss of data because a receiving device is unable to accept data at the rate it is transmitted.

overvoltage. A voltage of higher than specified value.

parallel. (1) Pertaining to the concurrent or simultaneous operation of two or more devices, or to the concurrent performance of two or more activities. (2) Pertaining to the concurrent or simultaneous occurrence of two or more related activities in multiple devices or channels. (3) Pertaining to the simultaneity of two or more processes. (4) Pertaining to the simultaneous processing of the individual parts of a whole, such as the bits of a character and the characters of a word, using separate facilities for the various parts. (5) Contrast with serial.

parameter. (1) A variable that is given a constant value for a specified application and that may denote the application. (2) A name in a procedure that is used to refer to an argument passed to that procedure.

parity bit. A binary digit appended to a group of binary digits to make the sum of all the digits either always odd (odd parity) or always even (even parity).

parity check. (1) A redundancy check that uses a parity bit. (2) Synonymous with odd-even check.

PEL. Picture element.

personal computer. A small home or business computer that has a processor and keyboard and that can be connected to a television or some other monitor. An optional printer is usually available.

phototransistor. A transistor whose switching action is controlled by light shining on it.

picture element (PEL). The smallest displayable unit on a display.

polling. (1) Interrogation of devices for purposes such as to avoid contention, to determine operational status, or to determine readiness to send or receive data. (2) The process whereby stations are invited, one at a time, to transmit.

port. An access point for data entry or exit.

positive true. Synonym for active high.

positive-going edge. The edge of a pulse or signal changing in a positive direction. Synonymous with rising edge.

potentiometer. A variable resistor with three terminals, one at each end and one on a slider (wiper).

power supply. A device that produces the power needed to operate electronic equipment.

printed circuit. A pattern of conductors (corresponding to the wiring of an electronic circuit) formed on a board of insulating material.

printed-circuit board. A usually copper-clad plastic board used to make a printed circuit.

priority. A rank assigned to a task that determines its precedence in receiving system resources.

processing program. A program that performs such functions as compiling, assembling, or translating for a particular programming language.

processing unit. A functional unit that consists of one or more processors and all or part of internal storage.

processor. (1) In a computer, a functional unit that interprets and executes instructions. (2) A functional unit, a part of another unit such as a terminal or a processing unit, that interprets and executes instructions. (3) Deprecated term for processing program. (4) See microprocessor.

program. (1) A series of actions designed to achieve a certain result. (2) A series of instructions telling the computer how to handle a problem or task. (3) To design, write, and test computer programs.

programmable read-only memory (PROM). A read-only memory that can be programmed by the user.

programming language. (1) An artificial language established for expressing computer programs. (2) A set of characters and rules with meanings assigned prior to their use, for writing computer programs.

programming system. One or more programming languages and the necessary software for using these languages with particular automatic data-processing equipment.

PROM. Programmable read-only memory.

propagation delay. (1) The time necessary for a signal to travel from one point on a circuit to another. (2) The time delay between a signal change at an input and the corresponding change at an output.

protocol. (1) A specification for the format and relative timing of information exchanged between communicating parties. (2) The set of rules governing the operation of functional units of a communication system that must be followed if communication is to be achieved.

pulse. A variation in the value of a quantity, short in relation to the time schedule of interest, the final value being the same as the initial value.

radio frequency (RF). An ac frequency that is higher than the highest audio frequency. So called because of the application to radio communication.

radix. (1) In a radix numeration system, the positive integer by which the weight of the digit place is multiplied to obtain the weight of the digit place with the next higher weight; for example, in the decimal numeration system the radix of each digit place is 10. (2) Another term for base.

radix numeration system. A positional representation system in which the ratio of the weight of any one digit place to the weight of the digit place with the next lower weight is a positive integer (the radix). The permissible values of the character in any digit place range from 0 to one less than the radix.

RAM. Random access memory. Read/write memory.

random access memory (RAM). Read/write memory.

RAS. In the IBM Personal Computer, row address strobe.

raster. In computer graphics, a predetermined pattern of lines that provides uniform coverage of a display space.

read. To acquire or interpret data from a storage device, from a data medium, or from another source.

read-only memory (ROM). A storage device whose contents cannot be modified. The memory is retained when power is removed.

read/write memory. A storage device whose contents can be modified. Also called RAM.

recoverable error. An error condition that allows continued execution of a program.

red-green-blue-intensity (RGBI). The description of a direct-drive color monitor that accepts input signals of red, green, blue, and intensity.

redundancy check. A check that depends on extra characters attached to data for the detection of errors. See cyclic redundancy check.

register. (1) A storage device, having a specified storage capacity such as a bit, a byte, or a computer word, and usually intended for a special purpose. (2) A storage device in which specific data is stored.

retry. To resend the current block of data (from the last EOB or ETB) a prescribed number of times, or until it is entered correctly or accepted.

reverse video. A form of highlighting a character, field, or cursor by reversing the color of the character, field, or cursor with its background; for example, changing a red character on a black background to a black character on a red background.

RF. Radio frequency.

RF modulator. The device used to convert the composite video signal to the antenna level input of a home TV.

RGBI. Red-green-blue-intensity.

rising edge. Synonym for positive-going edge.

ROM. Read-only memory.

ROM/BIOS. The ROM resident basic input/output system, which provides the level control of the major I/O devices in the computer system.

row address strobe (RAS). A signal that latches the row address in a memory chip.

RS-232C. A standard by the EIA for communication between computers and external equipment.

RTS. Request to send. Associated with modem control.

run. A single continuous performance of a computer program or routine.

schematic. The representation, usually in a drawing or diagram form, of a logical or physical structure.

Schottky TTL. A version (S series) of TTL with faster switching speed, but requiring more power. See also transistor-transistor logic and low power Schottky TTL.

SDLC. Synchronous Data Link Control

sector. That part of a track or band on a magnetic drum, a magnetic disk, or a disk pack that can be accessed by the magnetic heads in the course of a predetermined rotational displacement of the particular device.

SERDES. Serializer/deserializer.

serial. (1) Pertaining to the sequential performance of two or more activities in a single device. In English, the modifiers serial and parallel usually refer to devices, as opposed to sequential and consecutive, which refer to processes. (2) Pertaining to the sequential or consecutive occurrence of two or more related activities in a single device or channel. (3) Pertaining to the sequential processing of the individual parts of a whole, such as the bits of a character or the characters of a word, using the same facilities for successive parts. (4) Contrast with parallel.

serializer/deserializer (SERDES). A device that serializes output from, and deserializes input to, a business machine.

setup. (1) In a computer that consists of an assembly of individual computing units, the arrangement of interconnections between the units, and the adjustments needed for the computer to operate. (2) The preparation of a computing system to perform a job or job step. Setup is usually performed by an operator and often involves performing routine functions, such as mounting tape reels. (3) The preparation of the system for normal operation.

short circuit. A low-resistance path through which current flows, rather than through a component or circuit.

signal. A variation of a physical quantity, used to convey data.

sink. A device or circuit into which current drains.

software. (1) Computer programs, procedures, and rules concerned with the operation of a data processing system. (2) Contrast with hardware.

source. The origin of a signal or electrical energy.

square wave. An alternating or pulsating current or voltage whose waveshape is square.

square wave generator. A signal generator delivering an output signal having a square waveform.

SS. Start-stop.

start bit. (1) A signal to a receiving mechanism to get ready to receive data or perform a function. (2) In a start-stop system, a signal preceding a character or block that prepares the receiving device for the reception of the code elements.

start-of-text (STX). A transmission control character that precedes a text and may be used to terminate the message heading.

start-stop system. A data transmission system in which each character is preceded by a start bit and is followed by a stop bit.

start-stop (SS) transmission. (1) Asynchronous transmission such that a group of signals representing a character is preceded by a start bit and followed by a stop bit. (2) Asynchronous transmission in which a group of bits is preceded by a start bit that prepares the receiving mechanism for the reception and registration of a character and is followed by at least one stop bit that enables the receiving mechanism to come to an idle condition pending the reception of the next character.

static memory. RAM memory 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 or a representation of something by reason of relationship, association, or convention. (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.

V. Volt.

video. Computer data or graphics displayed on a cathode ray tube, monitor, or display.

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.

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.

GLOSSARY

Notes:

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

INDEX

B

C

D

INDEX

E

F

G

H

I

J

INDEX

K

L

M

N

O

P

R

S

INDEX

T

U

V

W

X

Z

Numerals

The Personal Computer
Hardware Reference Library

Reader's Comment Form

Technical Reference 1502494

Your comments assist us in improving the usefulness of our publication; they are an important part of the input used for revisions.

IBM may use and distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Please do not use this form for technical questions regarding the IBM Personal Computer or programs for the IBM Personal Computer, or for requests for additional publications; this only delays the response. Instead, direct your inquiries or request to your authorized IBM Personal Computer dealer.

Comments:

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 321 BOCA RATON, FLORIDA 33432

POSTAGE WILL BE PAID BY ADDRESSEE

IBM PERSONAL COMPUTER
SALES & SERVICE
P.O. BOX 1328-C
BOCA RATON, FLORIDA 33432

Fold here