IBM

Personal Computer
Hardware Reference
Library

Technical Reference

6139821

IBM

Personal Computer Hardware Reference Library

Technical Reference

Revised Edition (March 1986)

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

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

Products are not stocked at the address below. Requests for copies of this publication and for technical information about IBM Personal Computer products should be made to your authorized IBM Personal Computer dealer, IBM Product Center, or your IBM Marketing Representative.

The following paragraph applies only to the United States and Puerto Rico: A Reader's Comment Form is provided at the back of this publication. If the form has been removed, address comments to: IBM Corporation, Personal Computer, P.O. Box 1328-C, Boca Raton, Florida 33429-1328. IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligations whatever.

Federal Communications Commission Radio Frequency Interference Statement

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

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

Notes:

Preface

This publication describes the various components of the IBM Personal Computer XT and IBM Portable Personal Computer; and the interaction of each.

The information in this publication is for reference, and is intended for hardware and program designers, programmers, engineers, and anyone else with a knowledge of electronics and/or programming who needs to understand the design and operation of the IBM Personal Computer XT or IBM Portable Personal Computer.

This publication consists of two parts: a system manual and an options and adapters manual.

The system manual is divided into the following sections:

Section 1, "System Board", discusses the component layout, circuitry, and function of the system board.

Section 2, "Coprocessor", describes the Intel 8087 coprocessor and provides programming and hardware interface information.

Section 3, "Power Supply", provides electrical input/output specifications as well as theory of operation for both the IBM Personal Computer XT power supply and the IBM Portable Personal Computer power supply.

Section 4, "Keyboard", discusses the hardware makeup, function, and layouts of the IBM Personal Computer XT 83-key and 101/102-key keyboards and the IBM Portable Personal Computer keyboard. In addition, keyboard encoding and usage is discussed.

Section 5, "System Bios", describes the basic input/output system and its use. This section also contains the software interrupt listing, a BIOS memory map, descriptions of vectors with special meanings, and a set of low memory maps.

Section 6, "Instruction Set", provides a quick reference for the 8088 and 8087 assembly instruction set.

Section 7, "Characters, Keystrokes, and Colors", supplies the decimal and hexadecimal values for characters and text attributes.

A glossary, bibliography, and index are also provided.

The *Technical Reference* Options and Adapters manual provides information, logic diagrams, and specifications pertaining to the options and adapters available for the IBM Personal Computer family of products. The manual is modular in format, with each module providing information about a specific option or adapter. Modules having a large amount of text contain individual indexes. The modules are grouped by type of device into the following categories:

- Expansion Unit
- Displays
- Printers
- Storage Devices
- Memory Expansion
- Adapters
- Miscellaneous
- Cables and Connectors.

Full-length hard-tab pages with the above category descriptions, separate the groups of modules.

The term "*Technical Reference* manual" in the Options and Adapters manual, refers to the:

- IBM Personal Computer XT/IBM Portable Personal Computer *Technical Reference* manual
- IBM Personal Computer *Technical Reference* manual
- IBM Personal Computer AT *Technical Reference* manual.

The term "*Guide to Operations* manual" in the Options and Adapters manual, refers to the:

- IBM Personal Computer *Guide to Operations* manual
- IBM Personal Computer XT *Guide to Operations* manual
- IBM Portable Personal Computer *Guide to Operations* manual
- IBM Personal Computer AT *Guide to Operations* manual.

Prerequisite Publications

- IBM Personal Computer XT *Guide to Operations*
- IBM Portable Personal Computer *Guide to Operations*.

Suggested Reading

- *BASIC for the IBM Personal Computer*
- *Disk Operating System (DOS)*
- *Hardware Maintenance Service* manual
- *Hardware Maintenance Reference* manual
- *Macro Assembler for the IBM Personal Computer.*

Notes:

Contents

SECTION 1. SYSTEM BOARD 1-1
Description 1-3
Microprocessor 1-4
Data Flow Diagrams 1-5
System Memory Map 1-8
System Timers 1-10
System Interrupts 1-11
System Boards 1-12
RAM 1-12
ROM 1-13
DMA 1-14
I/O Channel 1-15
System Board Diagram 1-19
I/O Channel Description 1-20
I/O Address Map 1-24
Other Circuits 1-26
Speaker Circuit 1-26
8255A I/O Bit Map 1-27
Specifications 1-29
System Unit 1-29
Card Specifications 1-31
Connectors 1-32
Logic Diagrams - 64/256K 1-34
Logic Diagrams - 256/640K 1-46

SECTION 2. COPROCESSOR 2-1
Description 2-3
Programming Interface 2-4
Hardware Interface 2-4

SECTION 3. POWER SUPPLIES 3-1
IBM Personal Computer XT Power Supply 3-3
Description 3-3
Input Requirements 3-4
Outputs 3-4
Overvoltage/Overcurrent Protection 3-5
Power Good Signal 3-5

Connector Specifications and Pin Assignments 3-6
IBM Portable Personal Computer Power Supply 3-7
Description 3-7
Voltage and Current Requirements 3-7
Power Good Signal 3-8
Connector Specifications and Pin Assignments 3-9

SECTION 4. KEYBOARDS 4-1
Introduction 4-3
83-Key Keyboard Description 4-3
Block Diagram 4-5
Keyboard Encoding and Usage 4-6
Extended Codes 4-9
Keyboard Layouts 4-12
Connector Specifications 4-19
Keyboard Logic Diagram 4-21
101/102-Key Keyboard 4-22
Description 4-22
Power-On Routine 4-25
Commands from the System 4-26
Commands to the System 4-26
Keyboard Scan Codes 4-28
Clock and Data Signals 4-32
Keyboard Encoding and Usage 4-33
Keyboard Layouts 4-44
Specifications 4-51
Logic Diagram 4-52

SECTION 5. SYSTEM BIOS 5-1
System BIOS Usage 5-3
System BIOS Listing - 11/22/85 5-11
Quick Reference - 256/640K Board 5-11
System BIOS Listing - 11/8/82 5-111
Quick Reference - 64/256K Board 5-111

SECTION 6. INSTRUCTION SET 6-1
8088 Register Model 6-3
Operand Summary 6-4
Second Instruction Byte Summary 6-4
Memory Segmentation Model 6-5
Segment Override Prefix 6-6
Use of Segment Override 6-6
8088 Instruction Set 6-7

Data Transfer 6-7
Arithmetic 6-10
Logic 6-13
String Manipulation 6-15
Control Transfer 6-16
8088 Instruction Set Matrix 6-20
8088 Conditional Transfer Operations 6-22
Processor Control 6-23
8087 Coprocessor Instruction Set 6-24
Data Transfer 6-24
Comparison 6-25
Arithmetic 6-26
Transcendental 6-28
Constants 6-28
Processor Control 6-29

SECTION 7. CHARACTERS, KEYSTROKES, AND COLORS 7-1
Character Codes 7-3
Quick Reference 7-14

Glossary Glossary-1

Bibliography Bibliography-1

Index Index-1

Notes:

INDEX TAB LISTING

Section 1. System Board . . . . SECTION 1

Section 2. Coprocessor . . . . SECTION 2

Section 3. Power Supplies . . . . SECTION 3

Section 4. Keyboards . . . . SECTION 4

Section 5. System BIOS . . . . SECTION 5

Section 6. Instruction Set . . . . SECTION 6

Notes:

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

Glossary . . . .

Bibliography . . . .

Index . . . .

SECTION 7

GLOSSARY

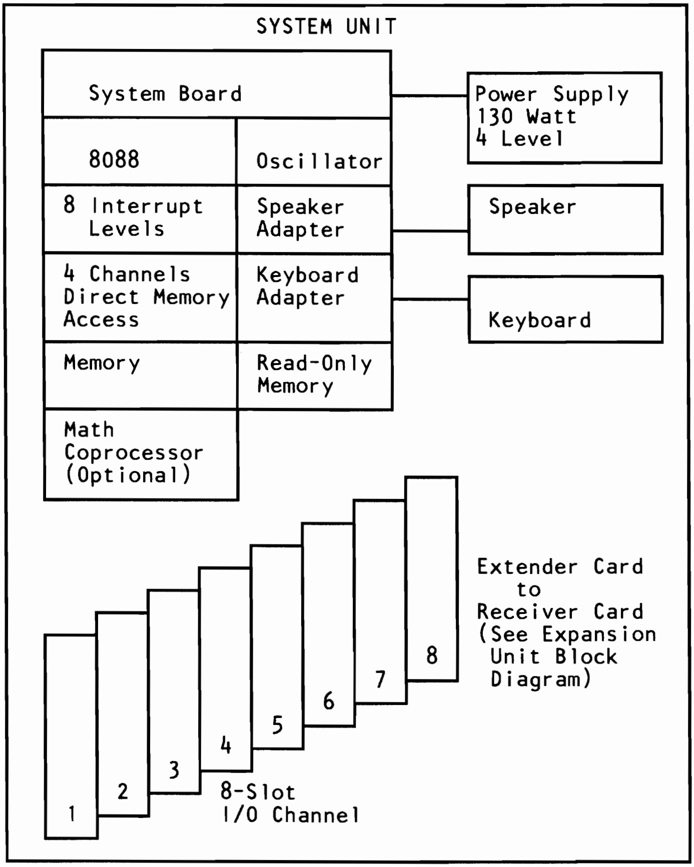
BIBLIOGRAPHY

INDEX

Notes:

System Block Diagram (XT)

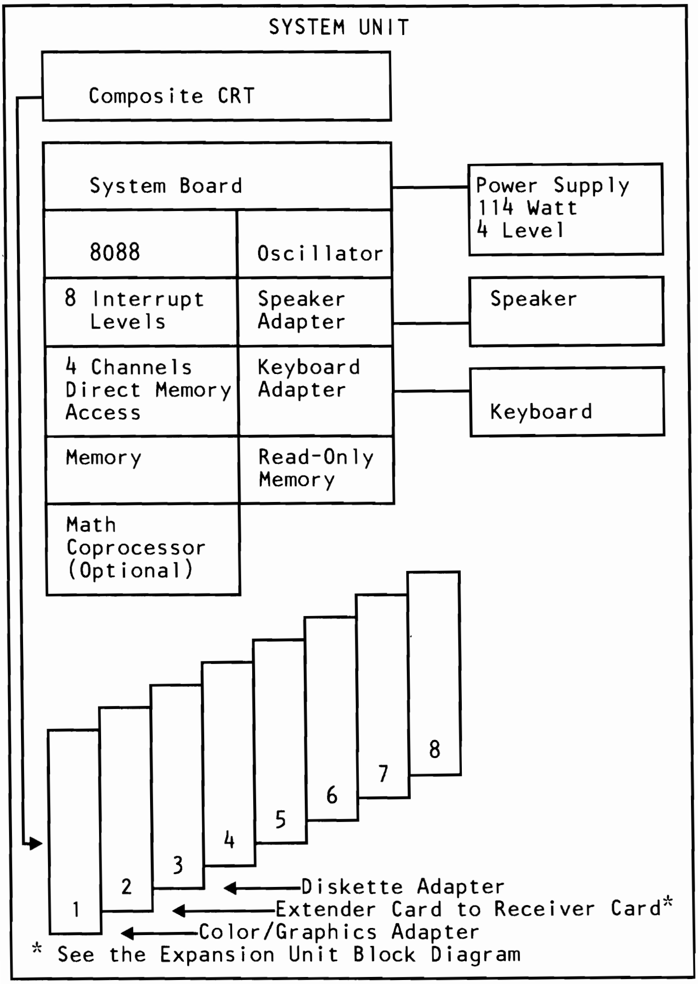
The following is a system block diagram of the IBM Personal Computer XT.



Note: A "System to Adapter Compatibility Chart," to identify the adapters supported by each system, and an "Option to Adapter Compatibility Chart," to identify the options supported by each adapter, can be found in the front matter of the *Technical Reference* Options and Adapters manual, Volume 1.

System Block Diagram (Portable)

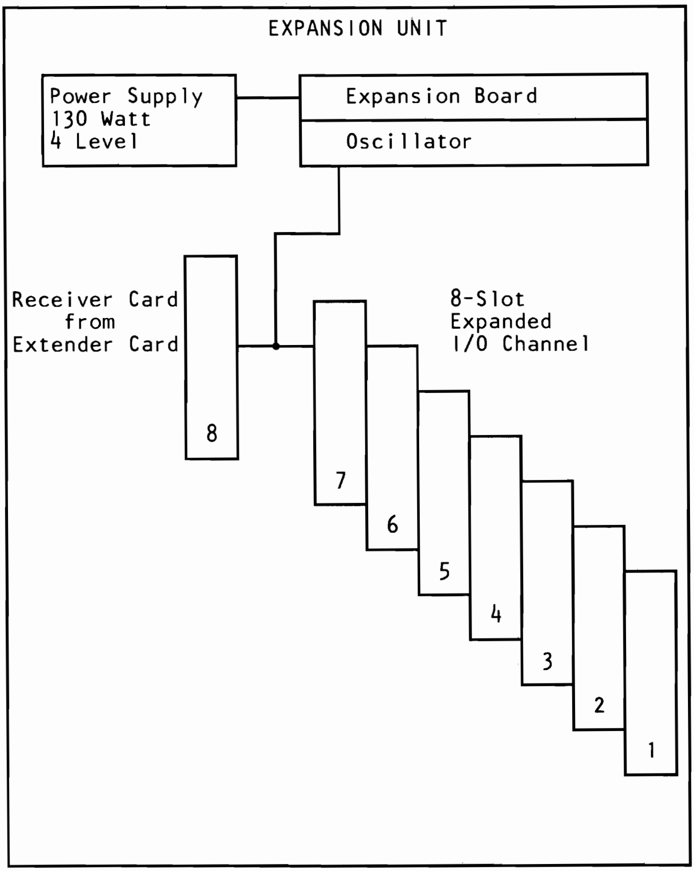
The following is a system block diagram of the IBM Portable Personal Computer.



Note: A "System to Adapter Compatibility Chart," to identify the adapters supported by each system, and an "Option to Adapter Compatibility Chart," to identify the options supported by each adapter, can be found in the front matter of the *Technical Reference* Options and Adapters manual, Volume 1.

Expansion Unit Block Diagram

The following is an expansion unit block diagram for the IBM Portable Personal Computer and IBM Personal Computer XT with the 64/256K system board.



Note: A "System to Adapter Compatibility Chart," to identify the adapters supported by each system, and an "Option to Adapter Compatibility Chart," to identify the options supported by each adapter, can be found in the front matter of the *Technical Reference* Options and Adapters manual, Volume 1.

Notes:

SECTION 1. SYSTEM BOARD

Description 1-3
Microprocessor 1-4
Data Flow Diagrams 1-5
System Memory Map 1-8
System Timers 1-10
System Interrupts 1-11
System Boards 1-12
    RAM 1-12
        64/256K System Board 1-12
        256/640K System Board 1-13
    ROM 1-13
DMA 1-14
I/O Channel 1-15
System Board Diagram 1-19
I/O Channel Description 1-20
I/O Address Map 1-24
Other Circuits 1-26
    Speaker Circuit 1-26
    8255A I/O Bit Map 1-27
Specifications 1-29
    System Unit 1-29
        Size 1-29
        Weight 1-29
        Power Cable 1-29
        Environment 1-29
        Heat Output 1-30
        Noise Level 1-30
        Electrical 1-30
    Card Specifications 1-31
Connectors 1-32
Logic Diagrams - 64/256K 1-34
Logic Diagrams - 256/640K 1-46

Notes:

Description

The system board fits horizontally in the base of the system unit of the Personal Computer XT and Portable Personal Computer and is approximately 215 mm by 304 mm (8-1/2 x 12 in.). It is a multilayer, single-land-per-channel design with ground and internal planes provided. DC power and a signal from the power supply enter the board through two 6-pin connectors. Other connectors on the board are for attaching the keyboard and speaker. Eight 62-pin card-edge sockets are also mounted on the board. The I/O channel is bussed across these eight I/O slots. Slot J8 is slightly different from the others in that any card placed in it is expected to respond with a 'card selected' signal whenever the card is selected.

A dual in-line package (DIP) switch (one 8-switch pack) is mounted on the board and can be read under program control. The DIP switch provides the system programs with information about the installed options, how much storage the system board has, what type of display adapter is installed, whether or not the coprocessor is installed, what operational modes are desired when power is switched on (color or black-and-white, 80- or 40-character lines), and the number of diskette drives attached.

The system board contains the adapter circuits for attaching the serial interface from the keyboard. These circuits generate an interrupt to the microprocessor when a complete scan code is received. The interface can request execution of a diagnostic test in the keyboard.

The system board consists of five functional areas: the processor subsystem and its support elements, the ROM subsystem, the read/write (R/W) memory subsystem, integrated I/O adapters, and the I/O channel. All are described in this section.

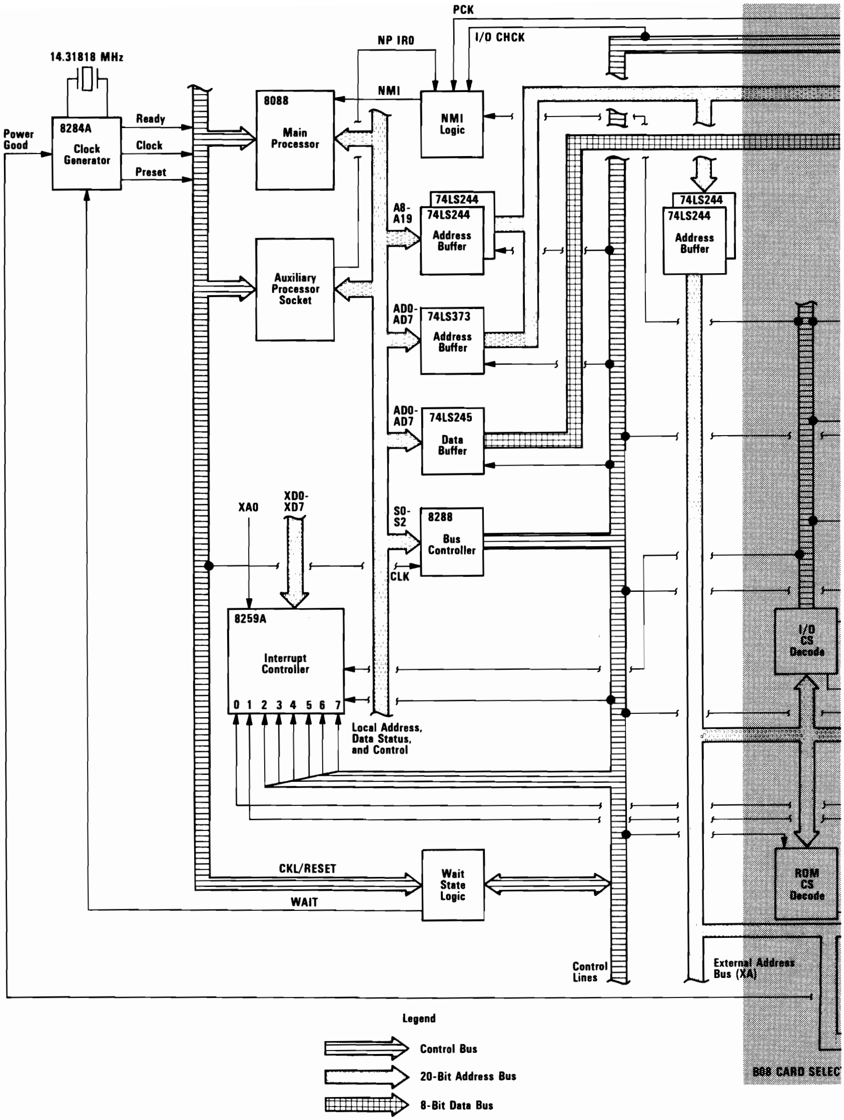
Microprocessor

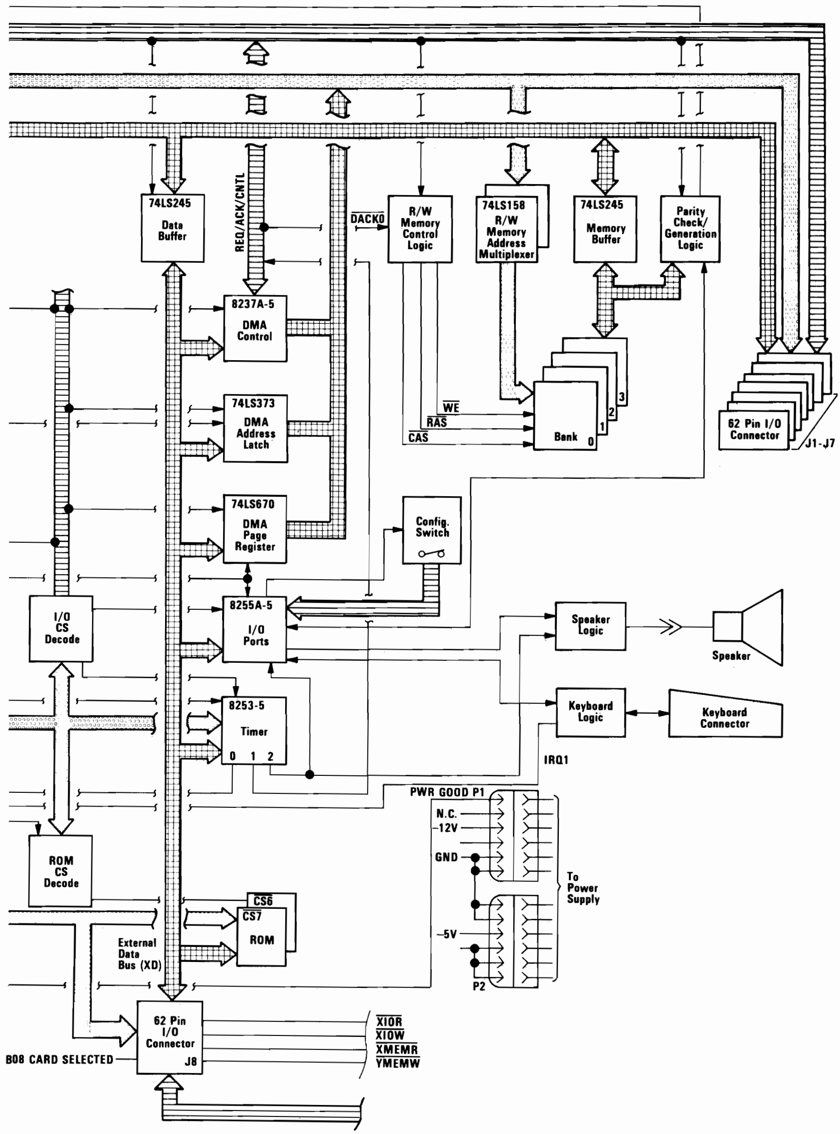
The heart of the system board is the Intel 8088 Microprocessor. This is an 8-bit external-bus version of Intel's 16-bit 8086 Microprocessor, and is software-compatible with the 8086. Thus, the 8088 supports 16-bit operations, including multiply and divide, and 20 bits of addressing (1M byte of storage). It also operates in maximum mode, so a coprocessor can be added as a feature. The microprocessor operates at 4.77MHz. This frequency is derived from a 14.31818MHz crystal, the frequency of which is divided by 3 for the microprocessor clock, and divided by 4 to obtain the 3.58MHz color-burst signal required for color televisions.

At the 4.77MHz clock rate, the 8088 bus cycles are four clocks of 210 nanoseconds (ns) each, or 840ns total. Some I/O cycles take five 210ns clocks or 1.05 microseconds (μs).

Data Flow Diagrams

The system board data flow diagram starts on the next page.


PCK
NP IRQ
I/O CHCK
14.31818 MHz
NMI
8088
Main Processor
NMI Logic
Power Good
8284A
Clock Generator
Ready
Clock
Preset
A8-A19
74LS244
74LS244
Address Buffer
Auxiliary Processor Socket
AD0-AD7
74LS373
Address Buffer
AD0-AD7
74LS245
Data Buffer
XA0
XD0-XD7
S0-S2
8288
Bus Controller
CLK
8259A
Interrupt Controller
0 1 2 3 4 5 6 7
Local Address, Data Status, and Control
I/O CS Decode
ROM CS Decode
CKL/RESET
WAIT
Wait State Logic
Control Lines
External Address Bus (XA)
808 CARD SELEC
Legend
Control Bus
20-Bit Address Bus
8-Bit Data Bus


74LS245
Data Buffer
REQ/ACK/CNTL
DACK0
R/W Memory Control Logic
74LS158
R/W Memory Address Multiplexer
74LS245
Memory Buffer
Parity Check/ Generation Logic
8237A-5
DMA Control
74LS373
DMA Address Latch
74LS670
DMA Page Register
WE
RAS
CAS
Bank 0
1
2
3
62 Pin I/O Connector
J1-J7
Config. Switch
8255A-5
I/O Ports
Speaker Logic
Speaker
Keyboard Logic
Keyboard Connector
I/O CS Decode
8253-5
Timer
0 1 2
IRQ1
PWR GOOD P1
N.C.
-12V
GND
-5V
P2
To Power Supply
ROM CS Decode
CS6
CS7
ROM
External Data Bus (XD)
62 Pin I/O Connector
J8
B08 CARD SELECTED
XIOR
XIOW
XMEMR
YMEMW

System Memory Map

<table>
<tr><th colspan="2">Start Address</th><th colspan="2">Function</th></tr>
<tr><th>Decimal</th><th>Hex</th><th>64/256K</th><th>256/640K</th></tr>
<tr><td>0K
16K
32K
48K</td><td>00000
04000
08000
0C000</td><td rowspan="4">128-256K
Read/Write
Memory
on the
System Board</td><td rowspan="10">256-640K
Read/Write
Memory
on the
System Board</td></tr>
<tr><td>64K
80K
86K
112K</td><td>10000
14000
18000
1C000</td></tr>
<tr><td>128K
144K
160K
176K</td><td>20000
24000
28000
2C000</td></tr>
<tr><td>192K
208K
224K
240K</td><td>30000
34000
38000
3C000</td></tr>
<tr><td>256K
272K
288K
304K</td><td>40000
44000
48000
4C000</td><td rowspan="6">384K R/W
Memory
Expansion
in the
I/O Channel</td></tr>
<tr><td>320K
336K
352K
368K</td><td>50000
54000
58000
5C000</td></tr>
<tr><td>384K
400K
416K
432K</td><td>60000
64000
68000
6C000</td></tr>
<tr><td>448K
464K
480K
496K</td><td>70000
74000
78000
7C000</td></tr>
<tr><td>512K
528K
544K
560K</td><td>80000
84000
88000
8C000</td></tr>
<tr><td>576K
592K
608K
624K</td><td>90000
94000
98000
9C000</td></tr>
</table>

System Memory Map (Part 1 of 2)

Start Address		Function
Decimal	Hex	64/256K & 256/640K
640K 656K 672K 688K	A0000 A4000 A8000 AC000	128K Reserved
704K	B0000	Monochrome
736K	B8000	Color/Graphics
752K	BC000	
768K	C0000	Enhanced Graphics
784K	C6000	Professional Graphics
800K	C8000	Fixed Disk Control
816K	CC000	PC Network
832K	D0000	Cluster
848K 864K 880K 896K 912K 928K 944K	D4000 D8000 DC000 E0000 E4000 E8000 EC000	192K Read Only Memory Expansion and Control
960K 976K 992K 1008K	F0000 F4000 F8000 FC000	64K Base system BIOS and BASIC ROM

System Memory Map (Part 2 of 2)

System Timers

Three programmable timer/counters are used by the system as follows: Channel 0 is used as a general-purpose timer providing a constant time base for implementing a time-of-day clock.

Channel 0	**System Timer**
GATE 0	Tied on
CLK IN 0	1.193182 MHz OSC
CLK OUT 0	8259A IRQ 0

Channel 1 is used to time and request refresh cycles from the DMA channel.

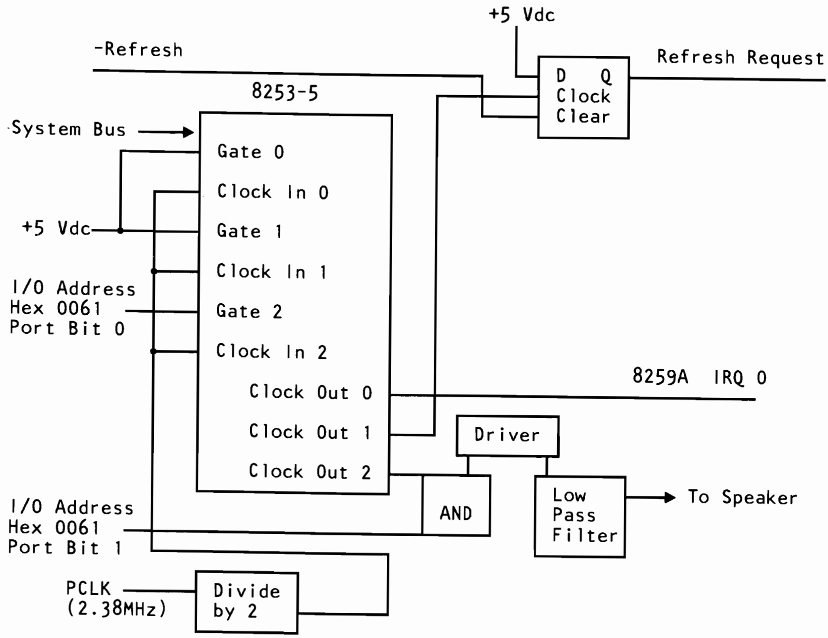
Channel 1	**Refresh Request Generator**
GATE 1	Tied on
CLK IN 1	1.193182 MHz OSC
CLK OUT 1	Request refresh cycle

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

Channel 2 is used to support the tone generation for the audio speaker. Each channel has a minimum timing resolution of 1.05μs.

Channel 2	**Tone Generation for Speaker**
GATE 2	Controlled by bit 0 of port hex 61, PPI bit
CLK IN 2	1.193182 MHz OSC
CLK OUT 2	Used to drive the speaker

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



System-Timer Block Diagram

System Interrupts

Of the eight prioritized levels of interrupt, six are bussed to the system expansion slots for use by feature cards. Two levels are used on the system board. Level 0, the higher priority, is attached to Channel 0 of the timer/counter and provides a periodic interrupt for the time-of-day clock. Level 1 is attached to the keyboard adapter circuits and receives an interrupt for each scan code sent by the keyboard.

The non-maskable interrupt (NMI) of the 8088 is used to report memory parity errors.

The following diagram contains the System Interrupt Listing.

Number	Usage
NMI	Parity 8087
0	Timer
1	Keyboard
2	EGA Display, PC Net, 3278/79
3	Asynchronous Communications (Alternate) PC Net(Alternate) 3278/79(Alternate) SDLC Communications BSC Communications Cluster (Primary)
4	Asynchronous Communications (Primary) SDLC Communications BSC Communications Voice Communications Adapter *
5	Fixed Disk
6	Diskette
7	Printer Cluster (Alternate)
* Jumper selectable to 2, 3, 4, 7.	

8088 Hardware Interrupt Listing

System Boards

There are two types of system boards, 64/256K and 256/640K.

RAM

64/256K System Board

The 64/256K system board has either 128K or 256K of R/W memory. Memory greater than the system board's maximum of 256K is obtained by adding memory cards in the expansion slots. The memory consists of dynamic 64K by 1 bit chips with an access time of 200ns and a cycle time of 345ns. All R/W memory is parity-checked.

256/640K System Board

The 256/640K system board has either 256K, 512K or 640K of R/W memory. The memory consists of dynamic 64K by 1 bit chips in Banks 2 and 3 and dynamic 256K by 1 bit chips in Banks 0 and 1 with an access time of 200ns and a cycle time of 345ns. All R/W memory is parity-checked.

System Board	Minimum Storage	Maximum Storage	Memory Modules	Pluggable (Banks 0-1)	Pluggable (Banks 2-3)
64/256K	64K	256K	64K by 1 bit	2 Banks of 9	2 Banks of 9
256/640K	256K	640K	256K by 1 bit and 64K by 1 bit	2 Banks of 9	2 Banks of 9

ROM

The system board supports both read only memory (ROM) and R/W memory. It has space for 64K by 8 of ROM or erasable programmable read-only memory (EPROM). Two module sockets are provided, each of which can accept a 32K or 8K device. On the 64/256K system board, one socket has 32K by 8 bits of ROM, the other 8K by 8 bits. On the 256/640K system board, both sockets have 32K by 8 bits of ROM installed. This ROM contains the power-on self test, I/O drivers, dot patterns for 128 characters in graphics mode, and a diskette bootstrap loader. The ROM is packaged in 28-pin modules and has an access time and a cycle time of 250ns each.

DMA

The microprocessor is supported by a set of high-function support devices providing four channels of 20-bit direct-memory access (DMA), three 16-bit timer/counter channels, and eight prioritized interrupt levels.

Three of the four DMA channels are available on the I/O bus and support high-speed data transfers between I/O devices and memory without microprocessor intervention. The fourth DMA channel is programmed to refresh the system's dynamic memory. This is done by programming a channel of the timer/counter device to periodically request a dummy DMA transfer. This action creates a memory-read cycle, which is available to refresh dynamic memory both on the system board and in the system expansion slots. DMA data transfers take five clock cycles of 210ns, or 1.05μs. (See I/O CH RDY on page 1-22.) Refresh cycles occur once every 72 clocks (approximately 15μs) and require four clocks or approximately 5.6% of the bus bandwidth.

The following formula determines the percentage of bandwidth used for refresh.

```
                        64K X 1

 % Bandwidth used     4 cycles X 128     512
   for Refresh      = -------------- = ----- = 5.6%
                      1.93ms/210ns      9190

                       256K X 1

 % Bandwidth used     4 cycles X 256     1024
   for Refresh      = -------------- = ----- = 5.6%
                      3.86ms/210ns      19048
```

I/O Channel

The I/O channel is an extension of the 8088 microprocessor bus. It is, however, demultiplexed, repowered, and enhanced by the addition of interrupts and direct memory access (DMA) functions.

The I/O channel contains an 8-bit, bidirectional data bus, 20 address lines, 6 levels of interrupt, control lines for memory and I/O read or write, clock and timing lines, 3 channels of DMA control lines, memory refresh-timing control lines, a 'channel check' line, and power and ground for the adapters. Four voltage levels are provided for I/O cards: +5 Vdc ± 5%, -5 Vdc ± 10%, +12 Vdc ± 5%, and -12 Vdc ± 10%. These functions are provided in a 62-pin connector with 100-mil card tab spacing.

An 'I/O channel ready' line (I/O CH RDY) is available on the I/O channel to allow operation with slow I/O or memory devices. These devices can pull I/O CH RDY low to add wait states to the following operations:

- Normal memory read and write cycles take four 210ns clocks for a cycle time of 840ns/byte.
- Microprocessor-generated I/O read and write cycles require five clocks for a cycle time of 1.05μs/byte.
- DMA transfers require five clocks for a cycle time of 1.05μs/byte.

I/O devices are addressed using I/O mapped address space. The channel is designed so that 768 I/O device addresses are available to the I/O channel cards.

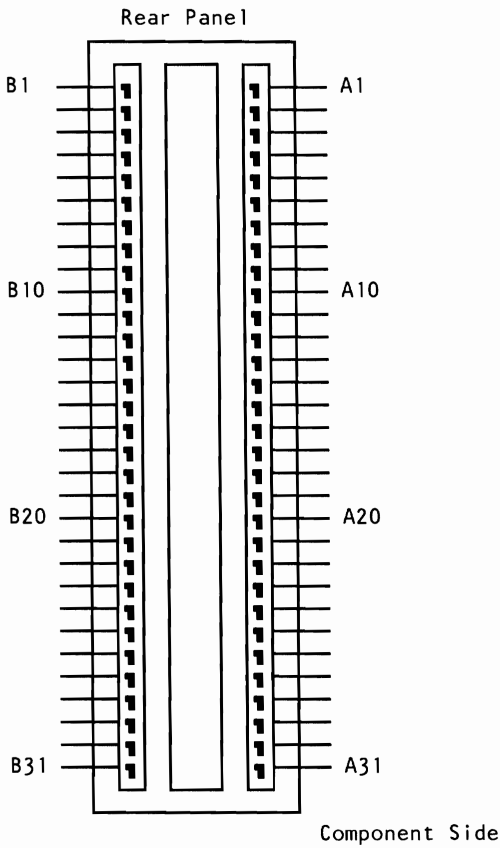
A 'channel check' line exists for reporting error conditions to the microprocessor. Activating this line results in a non-maskable interrupt (NMI) to the 8088 microprocessor. Memory expansion options use this line to report parity errors.

The I/O channel is repowered to provide sufficient drive to power all eight (J1 through J8) expansion slots, assuming two low-power

Schottky (LS) loads per slot. The IBM I/O adapters typically use only one load.

Timing requirements on slot J8 are much stricter than those on slots J1 through J7. Slot J8 also requires the card to provide a signal designating when the card is selected.

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



I/O Channel Pin Numbering (J1-J8)

The following figures show signals and voltages for the I/O channel connectors.

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

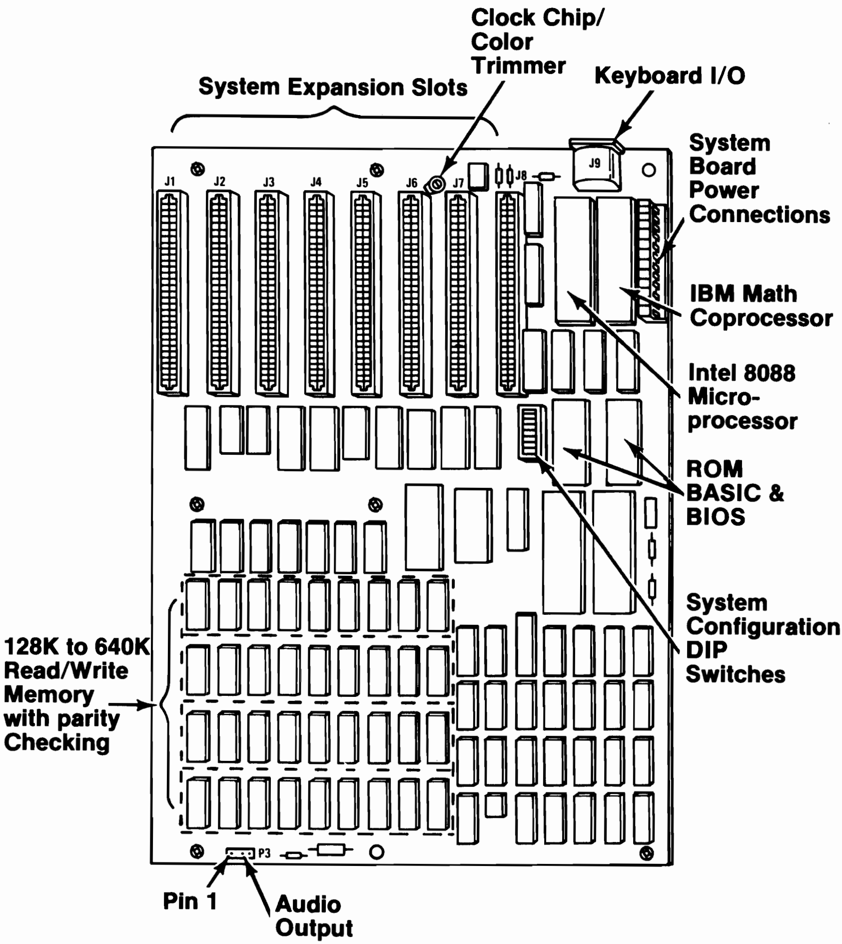
I/O Channel (A-Side, J1 through J8)

I/O Pin	Signal Name	I/O
B1	GND	Ground
B2	RESET DRV	O
B3	+5 Vdc	Power
B4	IRQ 2	I
B5	-5 Vdc	Power
B6	DRQ2	I
B7	-12 Vdc	Power
B8	-CARD SLCTD	I
B9	+12 Vdc	Power
B10	GND	Ground
B11	-MEMW	O
B12	-MEMR	O
B13	-IOW	I/O
B14	-IOR	I/O
B15	-DACK3	O
B16	DRQ3	I
B17	-DACK1	O
B18	DRQ1	I
B19	-DACK0	I/O
B20	CLK	O
B21	IRQ7	I
B22	IRQ6	I
B23	IRQ5	I
B24	IRQ4	I
B25	IRQ3	I
B26	-DACK2	O
B27	T/C	O
B28	ALE	O
B29	+5Vdc	Power
B30	OSC	O
B31	GND	Ground

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

System Board Diagram

The following diagram shows the component layout for the system board. All system board switch settings for total system memory, number of diskette drives, and types of display adapters are shown on page 1-27.



System Board Component Diagram

I/O Channel Description

The following is a description of the I/O Channel. All lines are TTL-compatible.

A0–A19 (O)

Address bits 0 to 19: These lines are used to address memory and I/O devices within the system. The 20 address lines allow access of up to 1M byte of memory. A0 is the least significant bit (LSB) and A19 is the most significant bit (MSB). These lines are generated by either the microprocessor or DMA controller. They are active high.

AEN (O)

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

ALE (O)

Address Latch Enable: This line is provided by the 8288 Bus Controller and is used on the system board to latch valid addresses from the microprocessor. It is available to the I/O channel as an indicator of a valid microprocessor address (when used with AEN). Microprocessor addresses are latched with the falling edge of ALE.

-CARD SLCTD (I)

-Card Selected: This line is activated by cards in expansion slot J8. It signals the system board that the card has been selected and that appropriate drivers on the system board should be directed to either read from, or write to, expansion slot J8. Connectors J1 through J8 are tied together at this pin, but the system board does not use their signal. This line should be driven by an open collector device.

CLK (O)

System clock: It is a divide-by-3 of the oscillator and has a period of 210ns (4.77MHz). The clock has a 33% duty cycle.

D0—D7 I/O

Data Bits 0 to 7: These lines provide data bus bits 0 to 7 for the microprocessor, memory, and I/O devices. D0 is the LSB and D7 is the MSB. These lines are active high.

-DACK0 to -DACK3 (O)

-DMA Acknowledge 0 to 3: These lines are used to acknowledge DMA requests (DRQ1—DRQ3) and refresh system dynamic memory (-DACK0). They are active low.

DRQ1–DRQ3 (I)

DMA Request 1 to 3: These lines are asynchronous channel requests used by peripheral devices to gain DMA service. They are prioritized with DRQ3 being the lowest and DRQ1 being the highest. A request is generated by bringing a DRQ line to an active level (high). A DRQ line must be held high until the corresponding DACK line goes active.

-I/O CH CK (I)

-I/O Channel Check: This line provides the microprocessor with parity (error) information on memory or devices in the I/O channel. When this signal is active low, a parity error is indicated.

I/O CH RDY (I)

I/O Channel Ready: This line, normally high (ready), is pulled low (not ready) by a memory or I/O device to lengthen I/O or memory cycles. It allows slower devices to attach to the I/O channel with a minimum of difficulty. Any slow device using this line should drive it low immediately upon detecting a valid address and a Read or Write command. This line should never be held low longer than 10 clock cycles. Machine cycles (I/O or memory) are extended by an integral number of clock cycles (210ns).

-IOR (O)

-I/O Read Command: This command line instructs an I/O device to drive its data onto the data bus. It may be driven by the microprocessor or the DMA controller. This signal is active low.

-IOW (O)

-I/O Write Command: This command line instructs an I/O device to read the data on the data bus. It may be driven by the microprocessor or the DMA controller. This signal is active low.

IRQ2—IRQ7 (I)

Interrupt Request 2 to 7: These lines are used to signal the microprocessor that an I/O device requires attention. They are prioritized with IRQ2 as the highest priority and IRQ7 as the lowest. An interrupt request is generated by raising an IRQ line (low to high) and holding it high until it is acknowledged by the microprocessor (interrupt service routine).

-MEMR (O)

-Memory Read: This command line instructs the memory to drive its data onto the data bus. It may be driven by the microprocessor or the DMA controller. This signal is active low.

-MEMW (O)

-Memory Write: This command line instructs the memory to store the data present on the data bus. It may be driven by the microprocessor or the DMA controller. This signal is active low.

OSC (O)

Oscillator: High-speed clock with a 70ns period (14.31818MHz). It has a 50% duty cycle.

RESET DRV (O)

Reset Drive: This line is used to reset or initialize system logic upon power-up or during a low line-voltage outage. This signal is synchronized to the falling edge of CLK and is active high.

T/C (O)

Terminal Count: This line provides a pulse when the terminal count for any DMA channel is reached. This signal is active high.

I/O Address Map

The following pages contain the planar and channel I/O Address Maps.

Hex Range*	Device
000-01F	DMA controller, 8237A-5
020-03F	Interrupt controller, 8259A
040-05F	Timer, 8253-5
060-06F	PPI 8255A-5
080-09F	DMA page registers
0AX**	NMI Mask Registers

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.

* These are the addresses decoded by the current set of adapter cards. IBM may use any of the unlisted addresses for future use.

** At power-on-time, the Non Mask Interrupt into the 8088 is masked off. This mask bit can be set and reset through system software as follows:

Set mask: Write hex 80 to I/O Address hex A0(enable NMI)

Clear mask: Write hex 00 to I/O Address hex A0(disable NMI)

Planar I/O Address Map

Hex Range*	Device
200-20F	Game Control
201	Game I/O
20C-20D	Reserved
210-217	Expansion Unit
21F	Reserved
278-27F	Parallel printer port 2
2B0-2DF	Alternate Enhanced Graphics Adapter
2E1	GPIB (Adapter 0)
2E2 & 2E3	Data Acquisition (Adapter 0)
2F8-2FF	Serial port 2
300-31F	Prototype card
320-32F	Fixed Disk
348-357	DCA 3278
360-367	PC Network (low address)
368-36F	PC Network (high address)
378-37F	Parallel printer port 1
380-38F***	SDLC, bisynchronous 2
390-393	Cluster
3A0-3AF	Bisynchronous 1
3B0-3BF	Monochrome Display and Printer Adapter
3C0-3CF	Enhanced Graphics Adapter
3D0-3DF	Color/Graphics Monitor Adapter
3F0-3F7	Diskette controller
3F8-3FF	Serial port 1
6E2 & 6E3	Data Acquisition (Adapter 1)
790-793	Cluster (Adapter 1)
AE2 & AE3	Data Acquisition (Adapter 2)
B90-B93	Cluster (Adapter 2)
EE2 & EE3	Data Acquisition (Adapter 3)
1390-1393	Cluster (Adapter 3)
22E1	GPIB (Adapter 1)
2390-2393	Cluster (Adapter 4)
42E1	GPIB (Adapter 2)
62E1	GPIB (Adapter 3)
82E1	GPIB (Adapter 4)
A2E1	GPIB (Adapter 5)
C2E1	GPIB (Adapter 6)
E2E1	GPIB (Adapter 7)

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

* These are the addresses decoded by the current set of adapter cards. IBM may use any of the unlisted addresses for future use.

*** SDLC Communication and Secondary Binary Synchronous Communications cannot be used together because their hex addresses overlap.

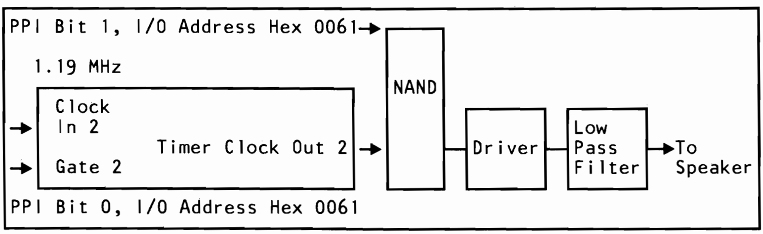
Channel I/O Address Map

Other Circuits

Speaker Circuit

The system unit has a 57.15 mm (2-1/4 in.) audio speaker. The speaker's control circuits and driver are on the system board. The speaker connects through a 2-wire interface that attaches to a 3-pin connector on the system board.

The speaker drive circuit is capable of approximately 1/2 watt of power. The control circuits allow the speaker to be driven three different ways: 1.) a direct program control register bit may be toggled to generate a pulse train; 2.) the output from Channel 2 of the timer/counter may be programmed to generate a waveform to the speaker; 3.) the clock input to the timer/counter can be modulated with a program-controlled I/O register bit. All three methods may be performed simultaneously.



Speaker Drive System Block Diagram

```
Channel 2 (Tone generation for speaker)
     Gate 2       -- Controlled by 8255A-5 PPI Bit (See 8255 Map)
     Clock In 2   -- 1.19318 MHz OSC
     Clock Out 2 -- Used to drive speaker
```

Speaker Tone Generation

The speaker connection is a 4-pin Berg connector.

	Pin	Function
P3	1	Data out
	2	Key
	3	Ground
	4	+5 Vdc

Speaker Connector (P3)

8255A I/O Bit Map

The 8255A I/O Bit Map shows the inputs and outputs for the Command/Mode register on the system board. Also shown are the switch settings for the memory, display, and number of diskette drives. The following page contains the I/O bit map.

Hex Port Number		Bit	Function
0060	INPUT	PA0–PA7	+ Keyboard Scan Code 0–7 Or Diagnostic Outputs 0–7
0061	OUTPUT	PB0	+ Timer 2 Gate Speaker
		1	+ Speaker Data
		2	Spare
		3	Read High Switches or Read Low Switches
		4	- Enable RAM Parity Check
		5	- Enable I/O Channel Check
		6	- Hold Keyboard Clock Low
		7	- (Enable Keyboard or + (Clear Keyboard)
0062	INPUT	PC0	Loop on POST — Sw-1 Or Display 0 — **Sw-5
		1	+ CoProcessor Installed — Sw-2 Or Display 1 — **Sw-6
		2	+ Planar RAM Size 0 — * Sw-3 Or # Drives — ***Sw-7
		3	+ Planar RAM Size 1 — * Sw-4 Or # Drives 1 — ***Sw-8
		4	Spare
		5	+ Timer Channel 2 Out
		6	+ I/O Channel Check
		7	+ RAM Parity Check

0063 Command/Mode Register Hex 99

Mode Register Value

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

* Sw-4	Sw-3	Amount of Memory on System Board - 64/256K
0	0	64K
0	1	128K
1	0	192K
1	1	256K

* Sw-4	Sw-3	Amount of Memory on System Board - 256/640K
0	0	256K
0	1	512K
1	0	576K
1	1	640K

** Sw-6	Sw-5	Display at Power-Up Mode
0	0	Reserved
0	1	Color 40 X 25 (BW Mode)
1	0	Color 80 X 25 (BW Mode)
1	1	IBM Monochrome 80 X 25

*** Sw-8	Sw-7	Number of Diskette Drives in System
0	0	1
0	1	2
1	0	3
1	1	4

Notes:
PA Bit = 0 implies switch "ON". PA Bit = 1 implies switch "OFF".
A plus (+) indicates a bit value of 1 performs the specified function.
A minus (-) indicates a bit value of 0 performs the specified function.

8255A I/O Bit Map

Specifications

System Unit

Size

- Length: 498 millimeters (19.6 inches)
- Depth: 411 millimeters (16.2 inches)
- Height: 147 millimeters (5.8 inches)

Weight

- 14.2 kilograms (31.6 pounds)

Power Cable

- Length: 1.8 meters (6 feet)

Environment

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

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

Heat Output

- 1229 British Thermal Units (BTU) per hour

Noise Level

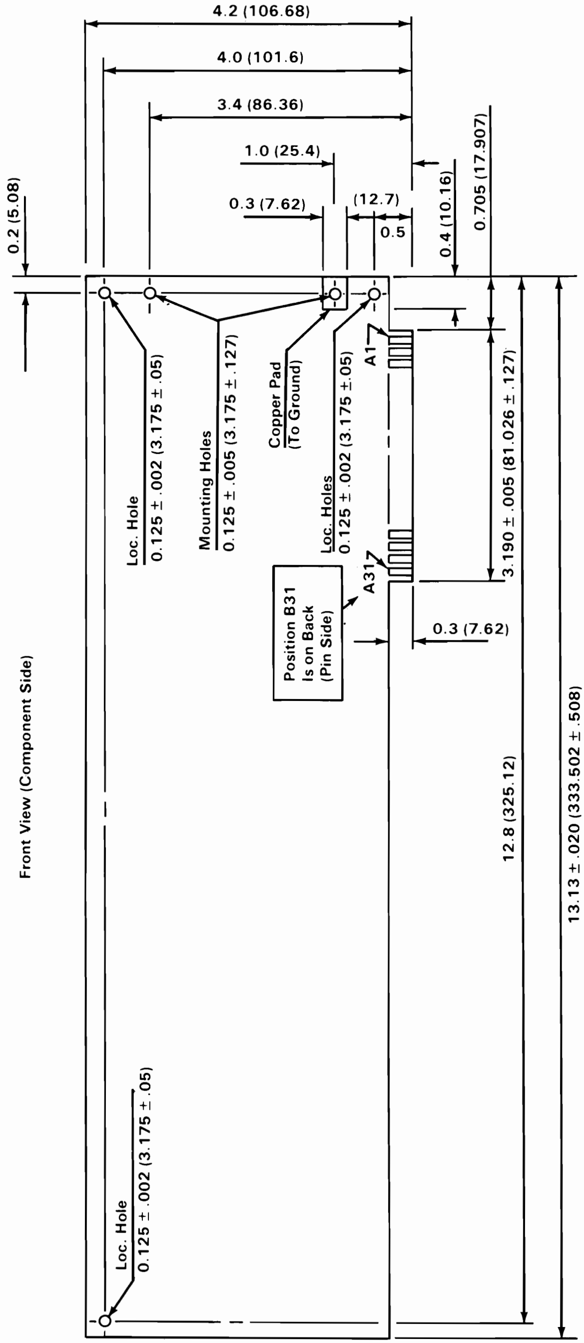
- 43 decibels average-noise rating (without printer)

Electrical

- Power: 450 VA
- Input
  - Nominal: 115 Vac
  - Minimum: 100 Vac
  - Maximum: 125 Vac

Card Specifications

The specifications for option cards follow.



Notes:

1. All Card Dimensions are ± .010 (.254) Tolerance (With Exceptions Indicated on Drawing or in Notes).
2. Max. Card Length is 13.15 (334.01) Smaller Length is Permissible.
3. Loc. and Mounting Holes are Non-Plated Thru. (Loc. 3X, Mtg. 2X).
4. 31 Gold Tabs Each Side, 0.100 ± .0005 (2.54 ± .0127) Center to Center, 0.06 ± .0005 (1.524 ± .0127) Width.
5. Numbers in Parentheses are in Millimeters. All Others are in Inches.

Connectors

The system board has the following additional connectors:

- Two power-supply connectors (P1 and P2)
- Speaker connector (J19)
- Keyboard connector (J22)

The pin assignments for the power-supply connectors, P1 and P2, are as follows. The pins are numbered 1 through 6 from the rear of the system.

Connector	Pin	Assignments
P1	1	Power Good
	2	Key
	3	+12 Vdc
	4	-12 Vdc
	5	Ground
	6	Ground
P2	1	Ground
	2	Ground
	3	-5 Vdc
	4	+5 Vdc
	5	+5 Vdc
	6	+5 Vdc

Power Supply Connectors (P1, P2)

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

Connector	Pin	Function
J19	1	Data out
	2	Key
	3	Ground
	4	+5 Vdc

Speaker Connector (J19)

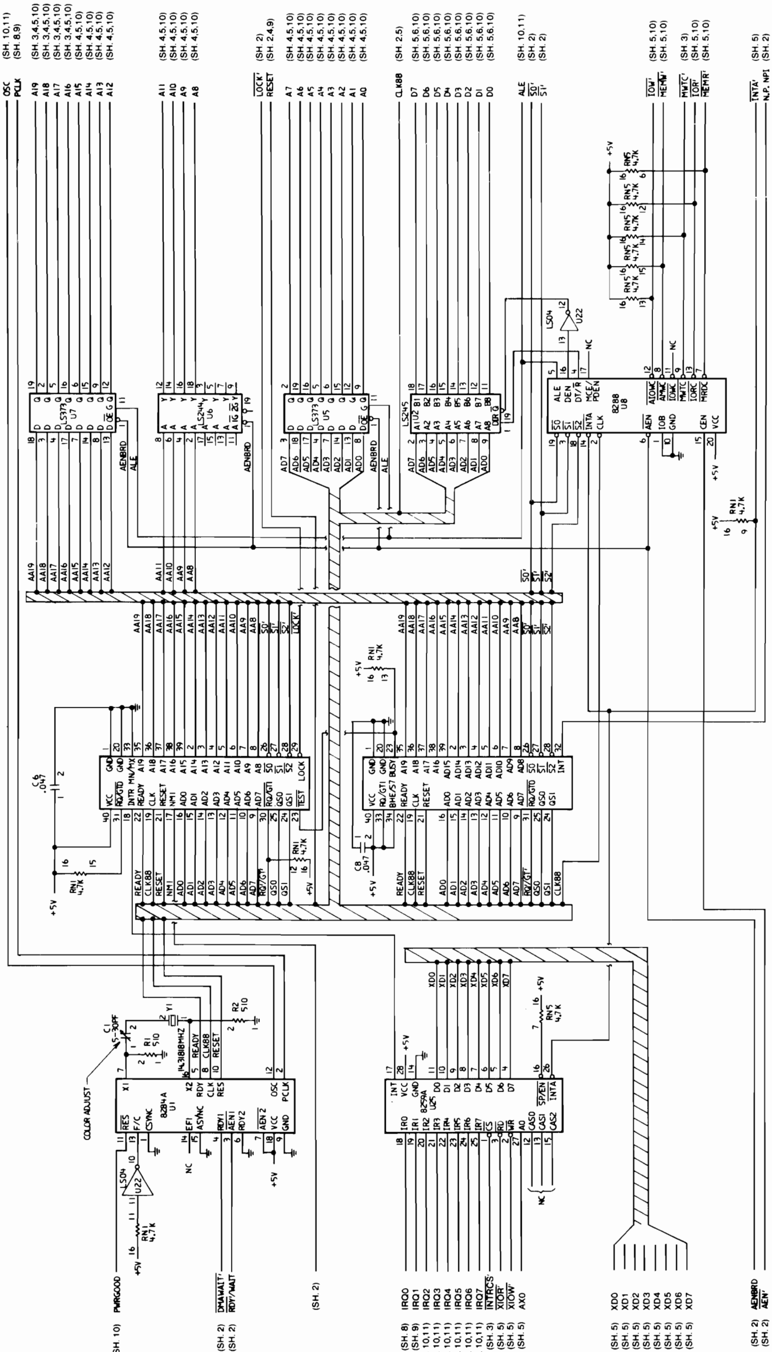
The keyboard connector, J22, is a 5-pin, 90-degree printed circuit board (PCB) mounting, DIN connector. For pin numbering, see the "Keyboard" section. The pin assignments are as follows:

Connector	Pin	Assignments
J22	1	Keyboard Clock
	2	Keyboard Data
	3	Reserved
	4	Ground
	5	+5 Vdc

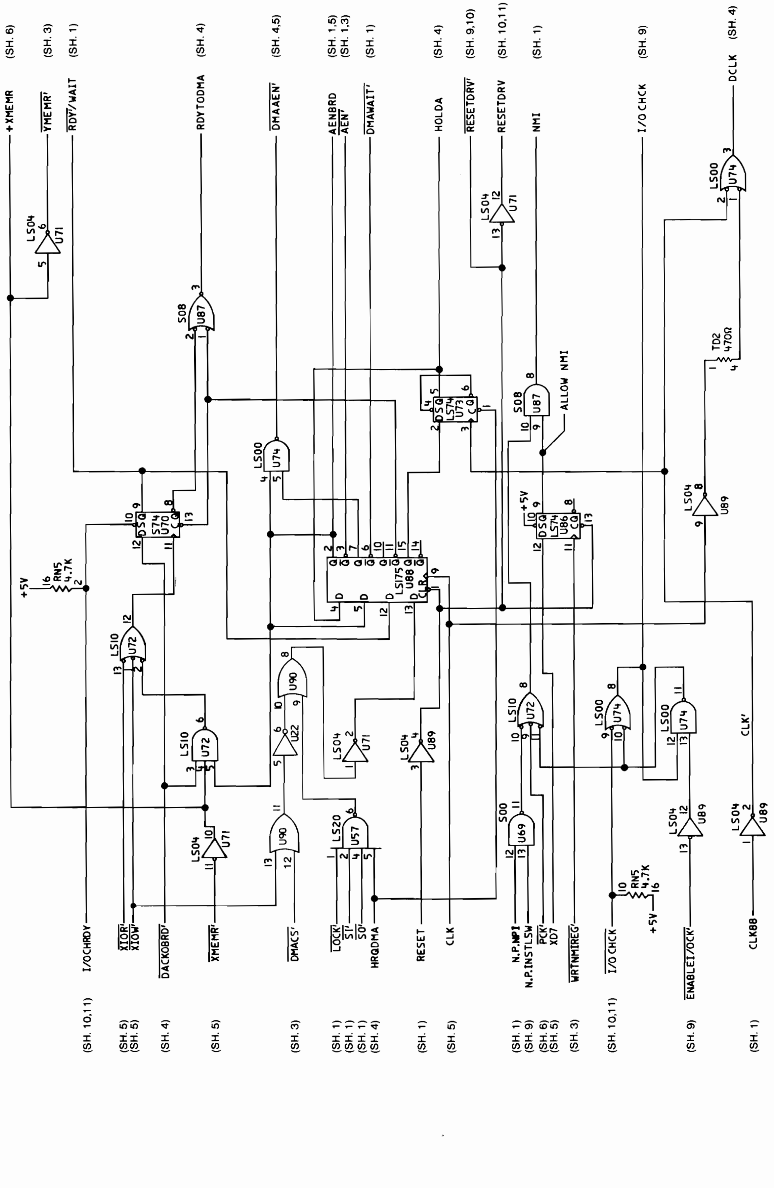
Keyboard Connector (J22)

Logic Diagrams - 64/256K

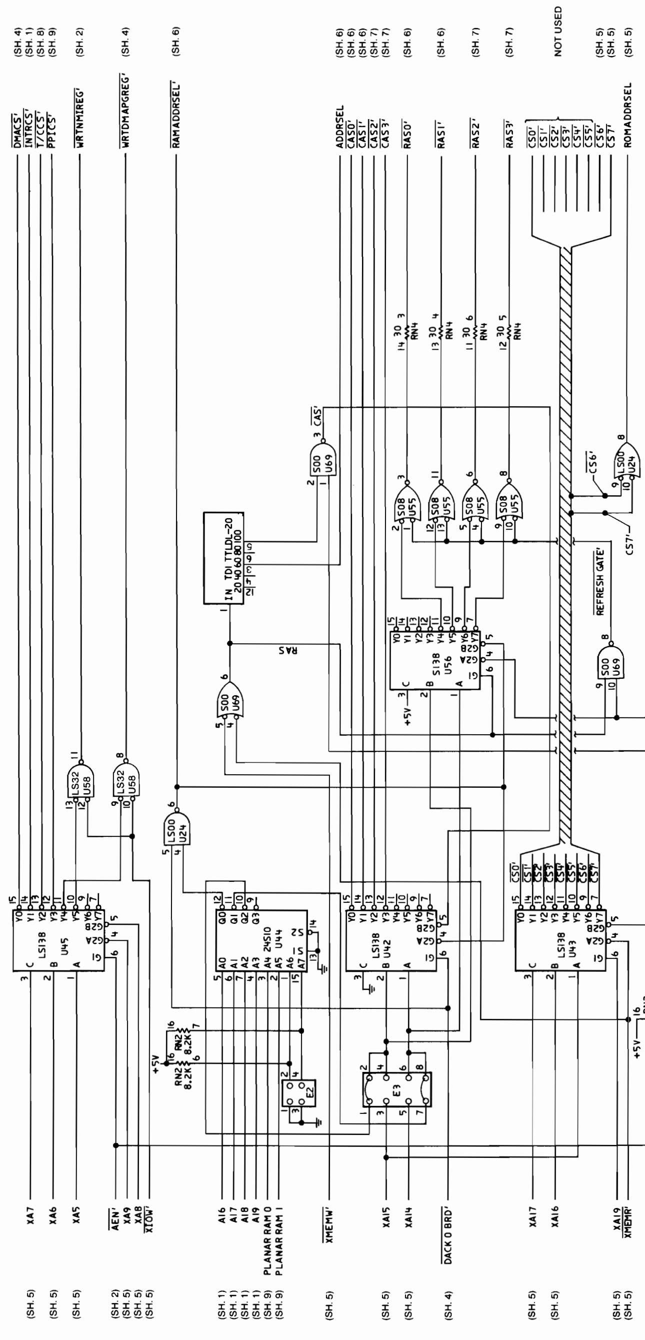
The following pages contain the logic diagrams for the 64/256K system board.



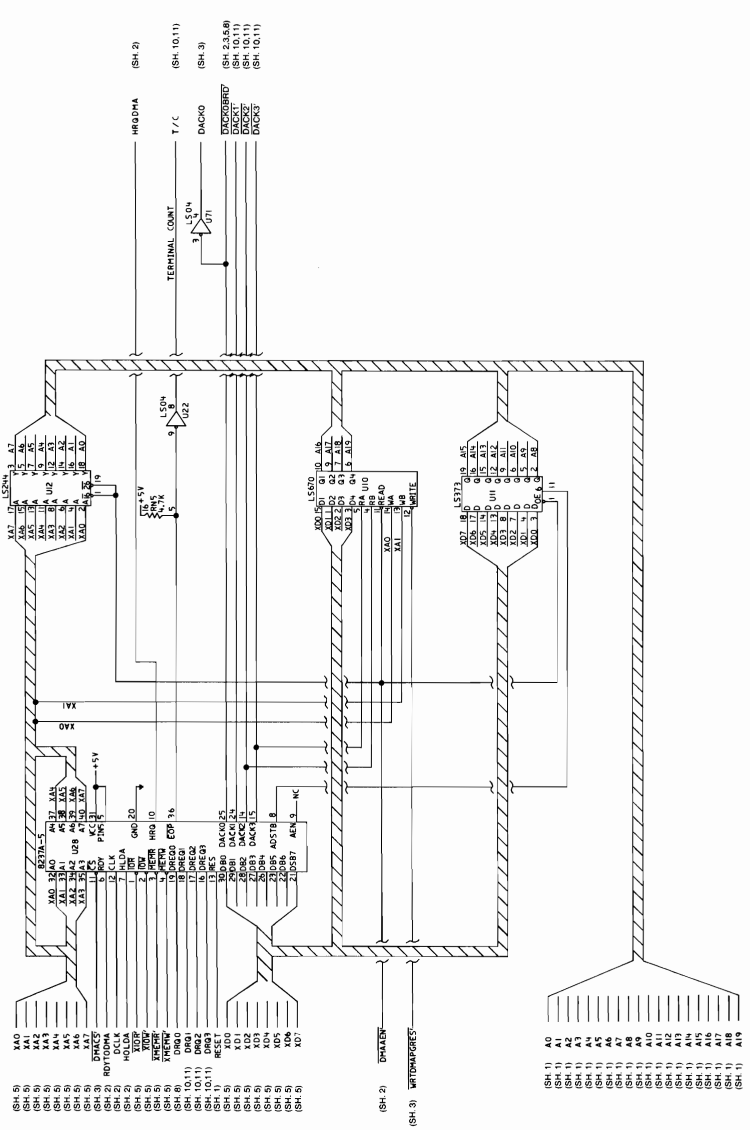
64/256K System Board (Sheet 1 of 11)



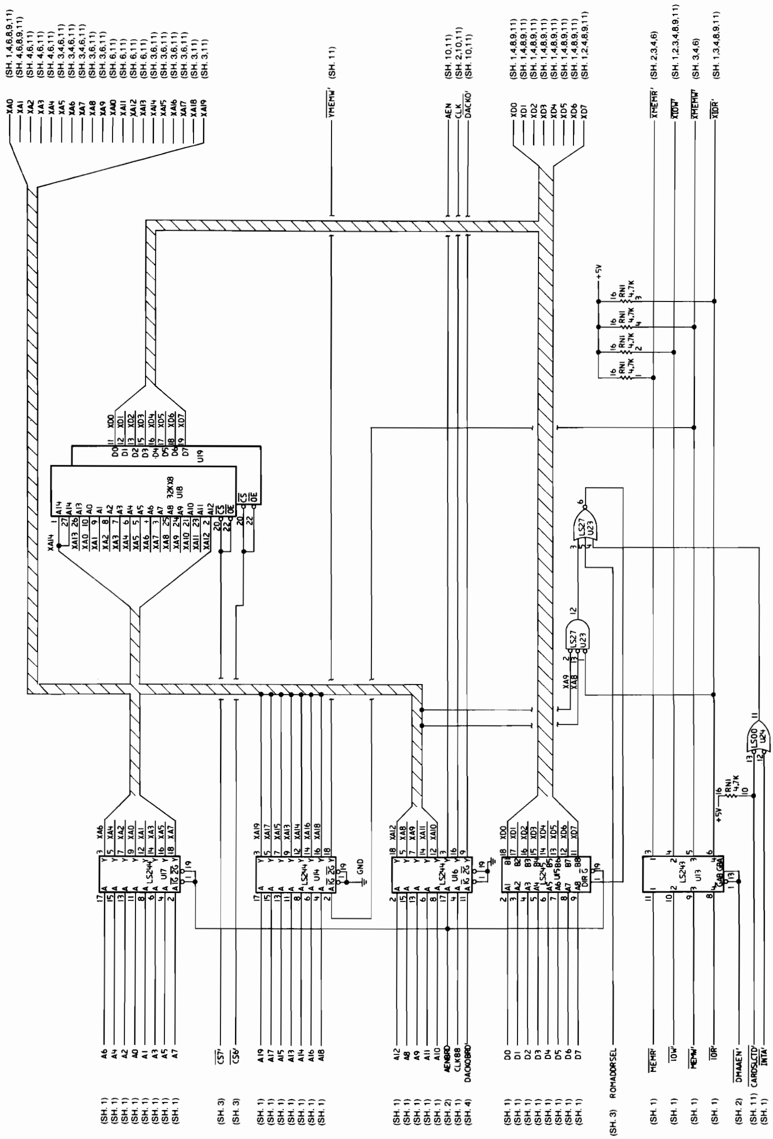
64/256K System Board (Sheet 2 of 11)



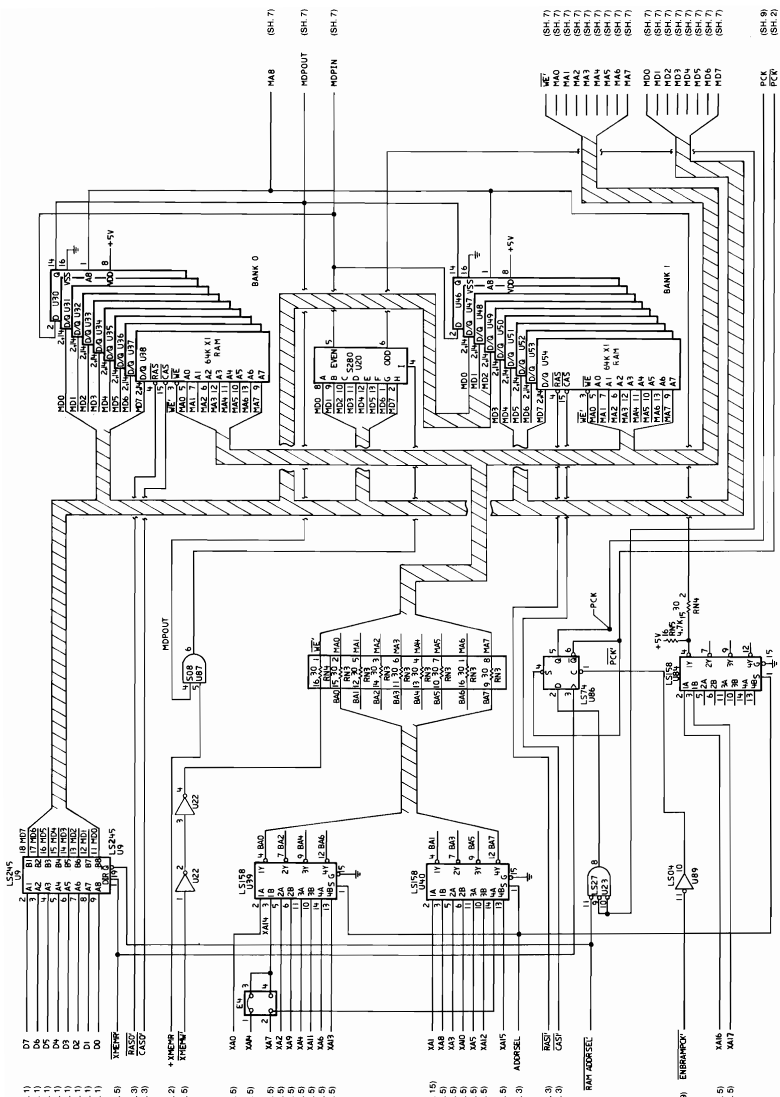
64/256K System Board (Sheet 3 of 11)



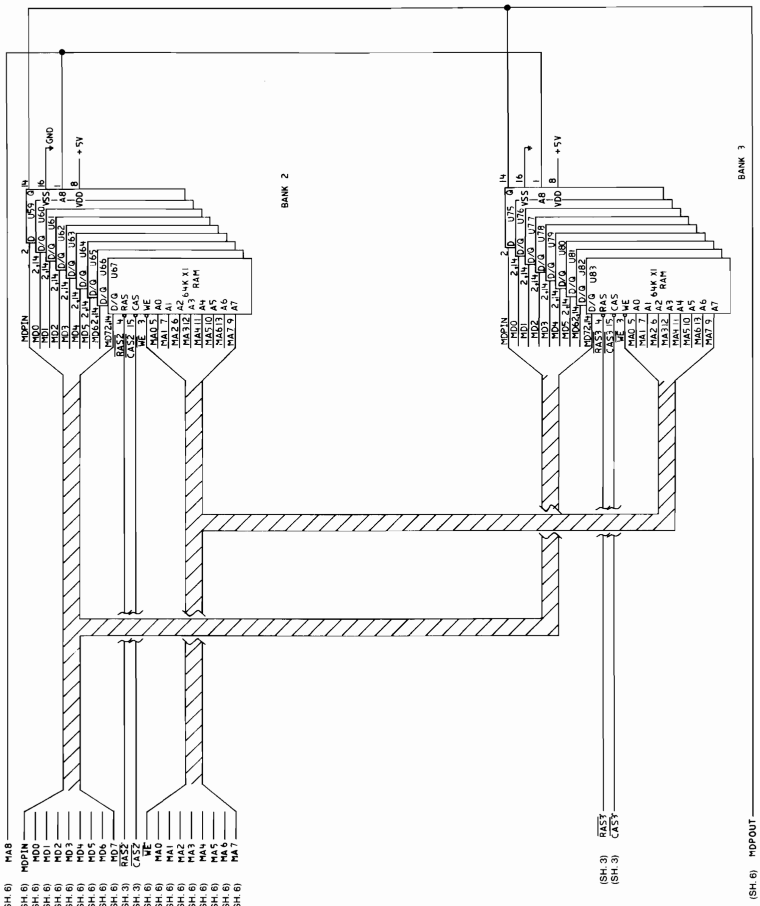
64/256K System Board (Sheet 4 of 11)



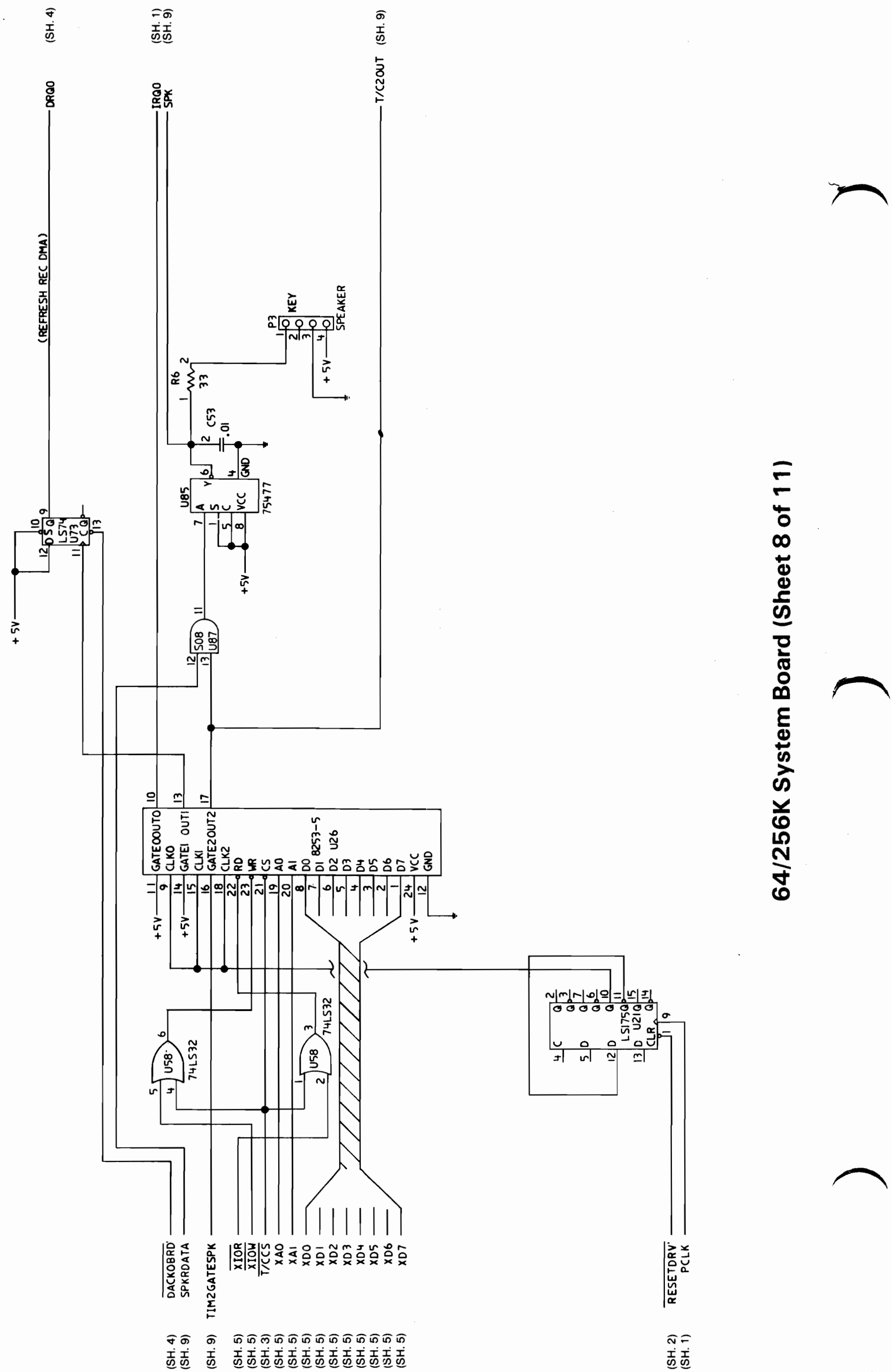
64/256K System Board (Sheet 5 of 11)



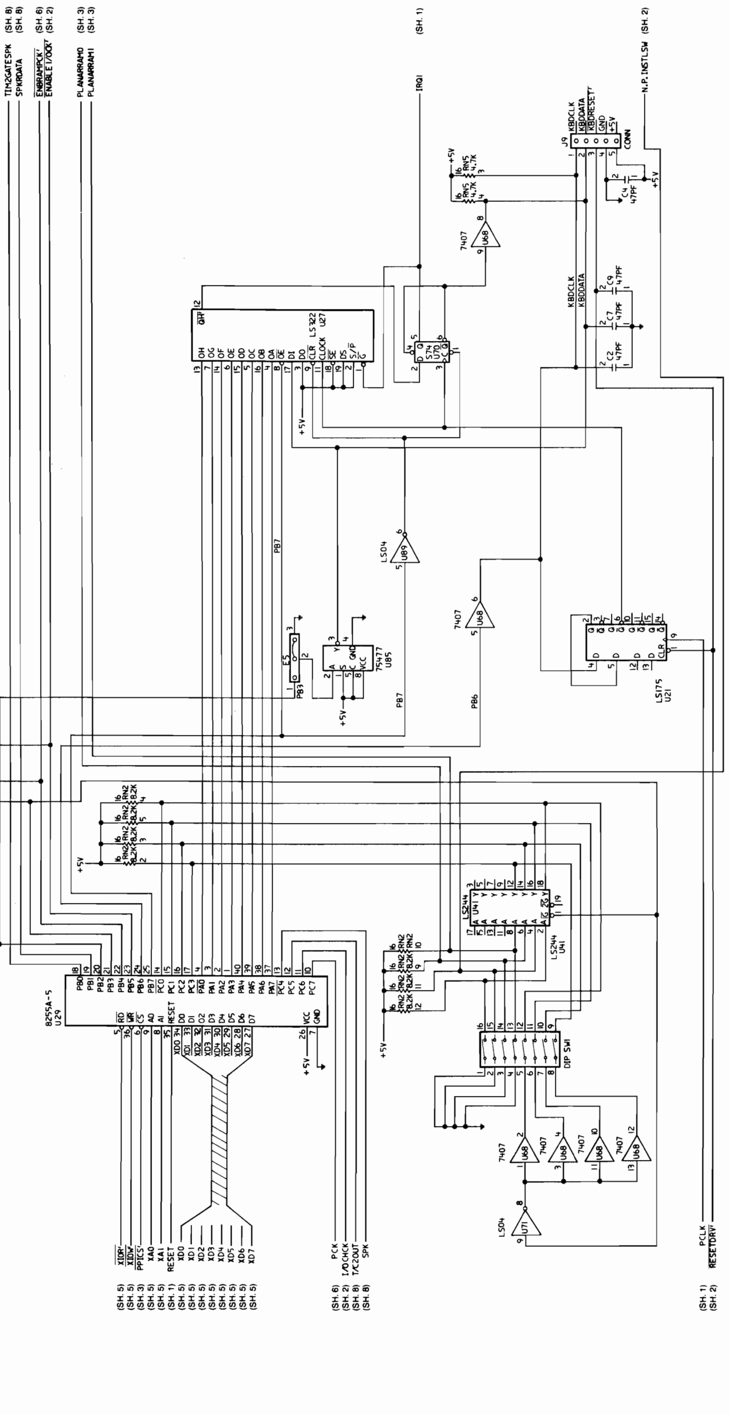
64/256K System Board (Sheet 6 of 11)



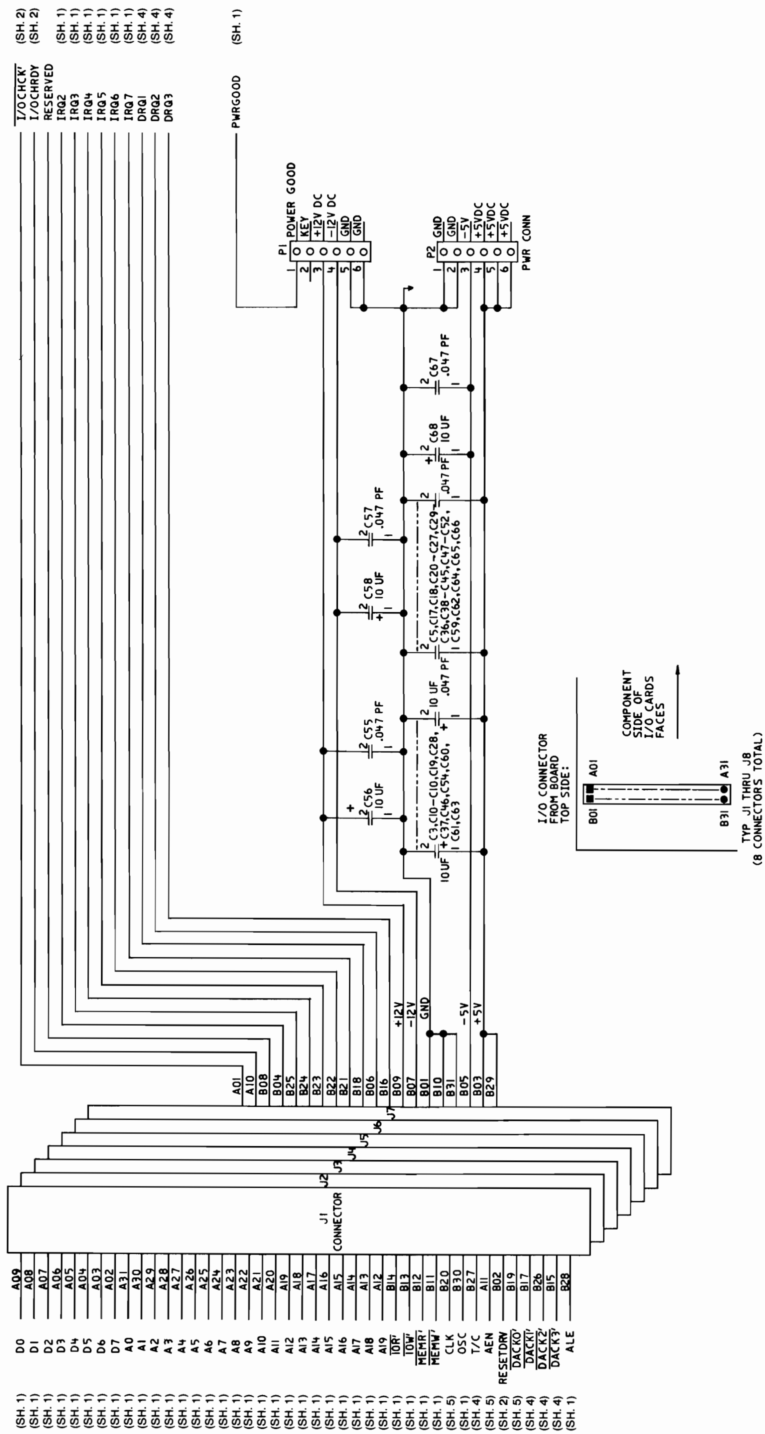
64/256K System Board (Sheet 7 of 11)



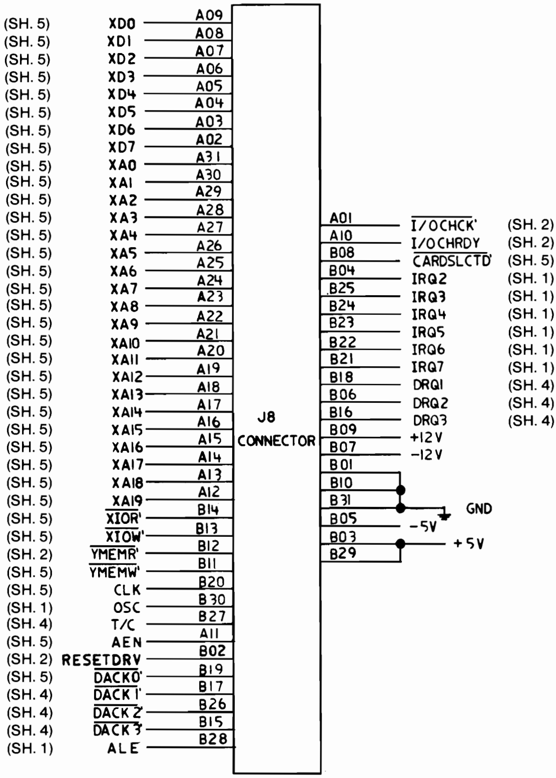
64/256K System Board (Sheet 8 of 11)



64/256K System Board (Sheet 9 of 11)



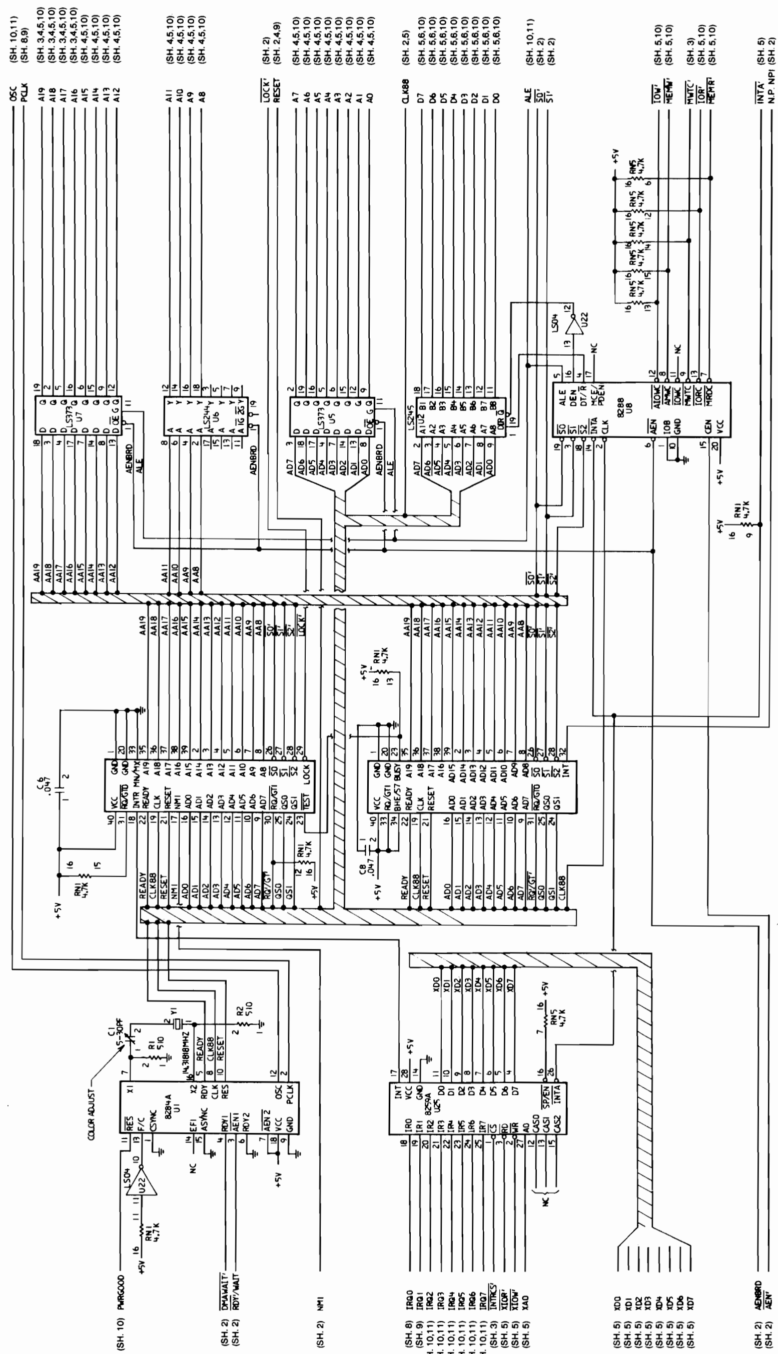
64/256K System Board (Sheet 10 of 11)



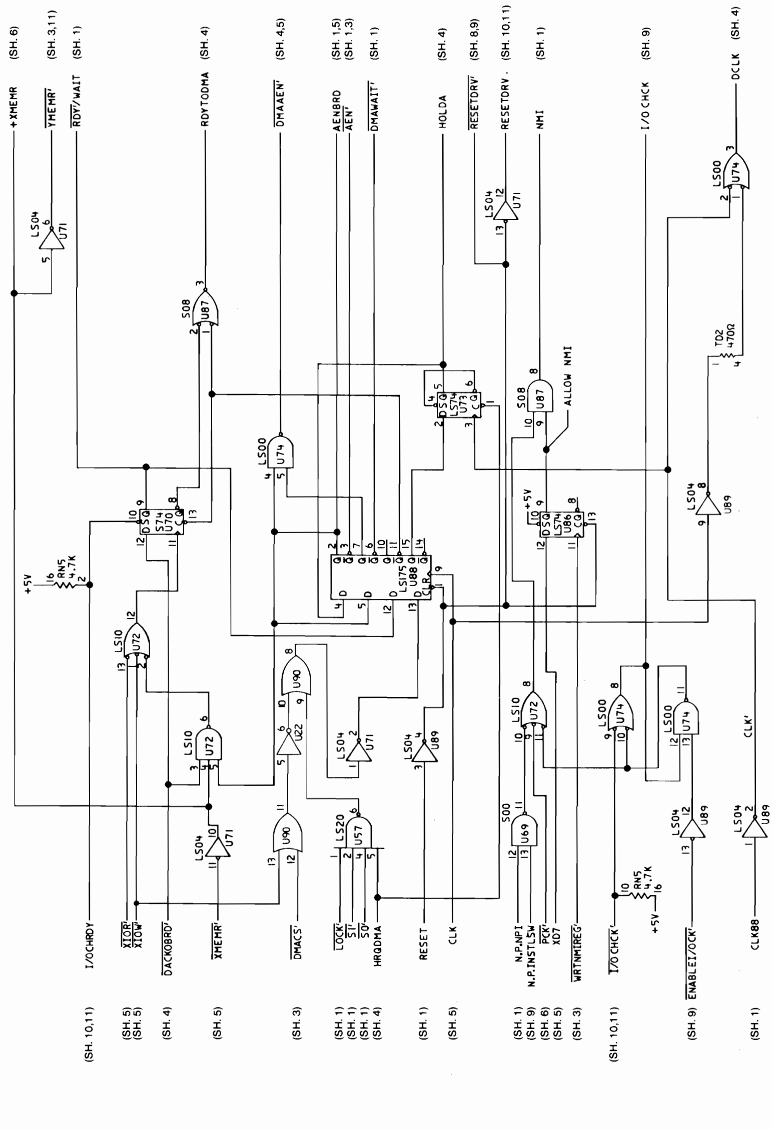
64/256K System Board (Sheet 11 of 11)

Logic Diagrams - 256/640K

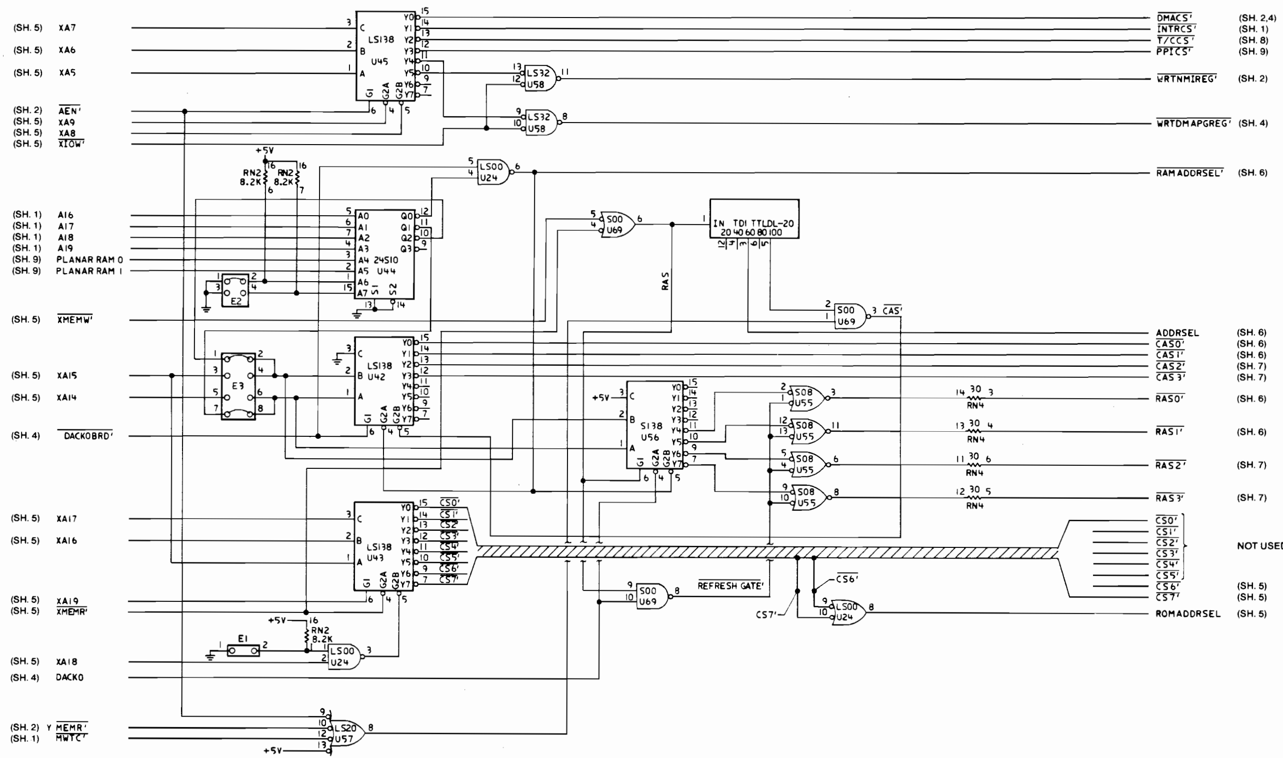
The following pages contain the logic diagrams for the 256/640K system board.



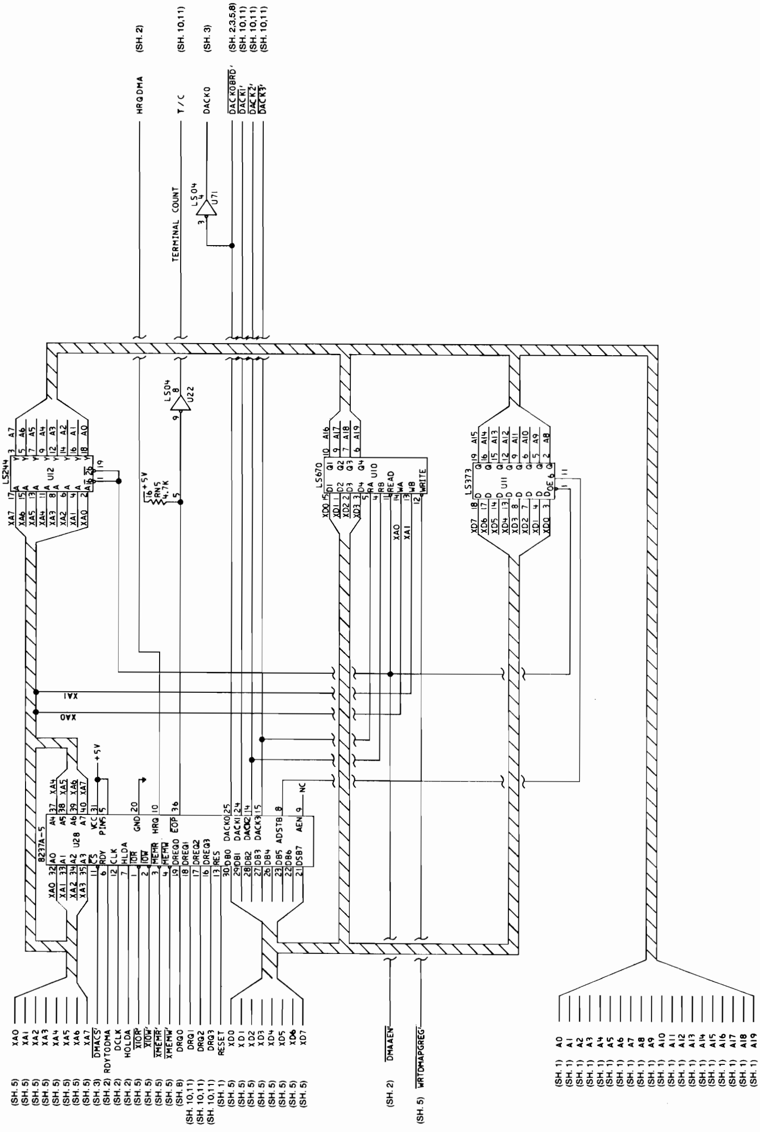
256/640K System Board (Sheet 1 of 11)



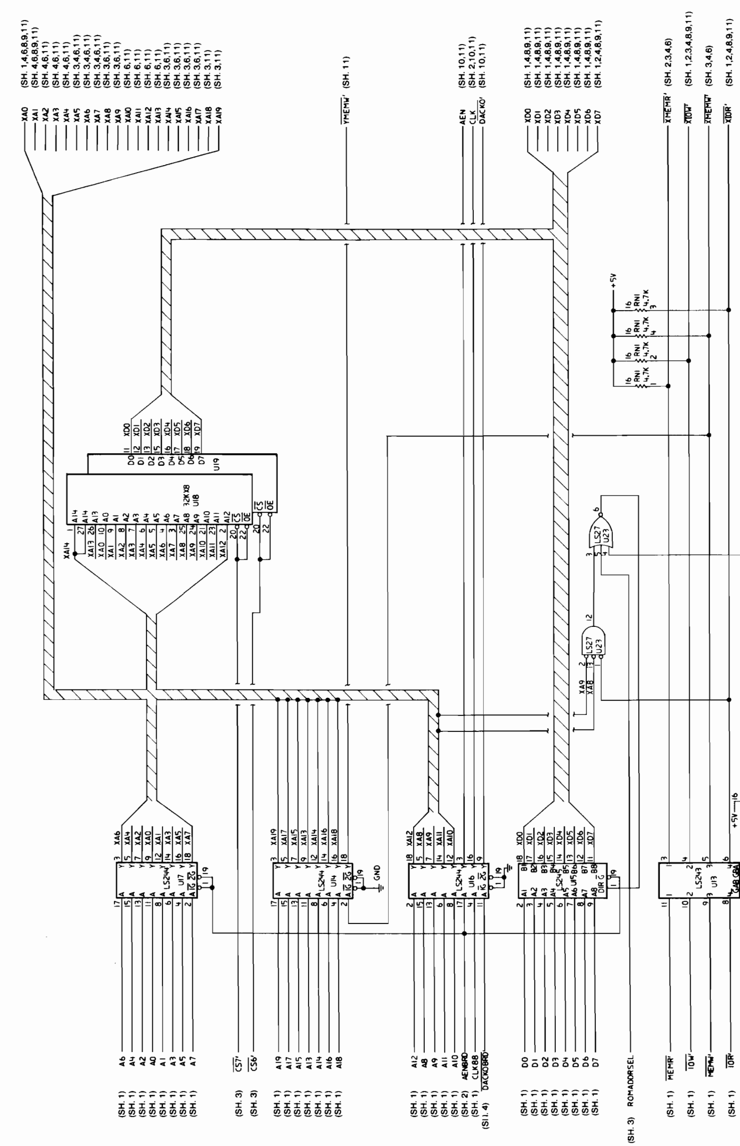
256/640K System Board (Sheet 2 of 11)



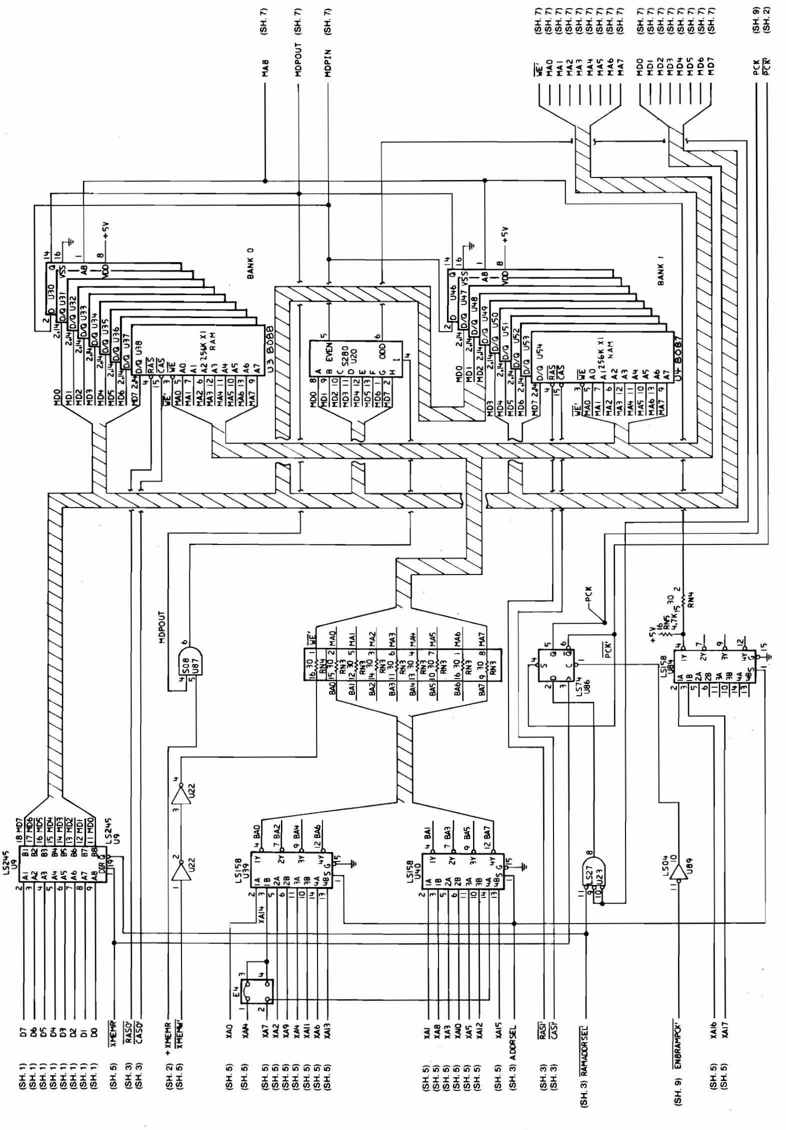
256/640K System Board (Sheet 3 of 11)



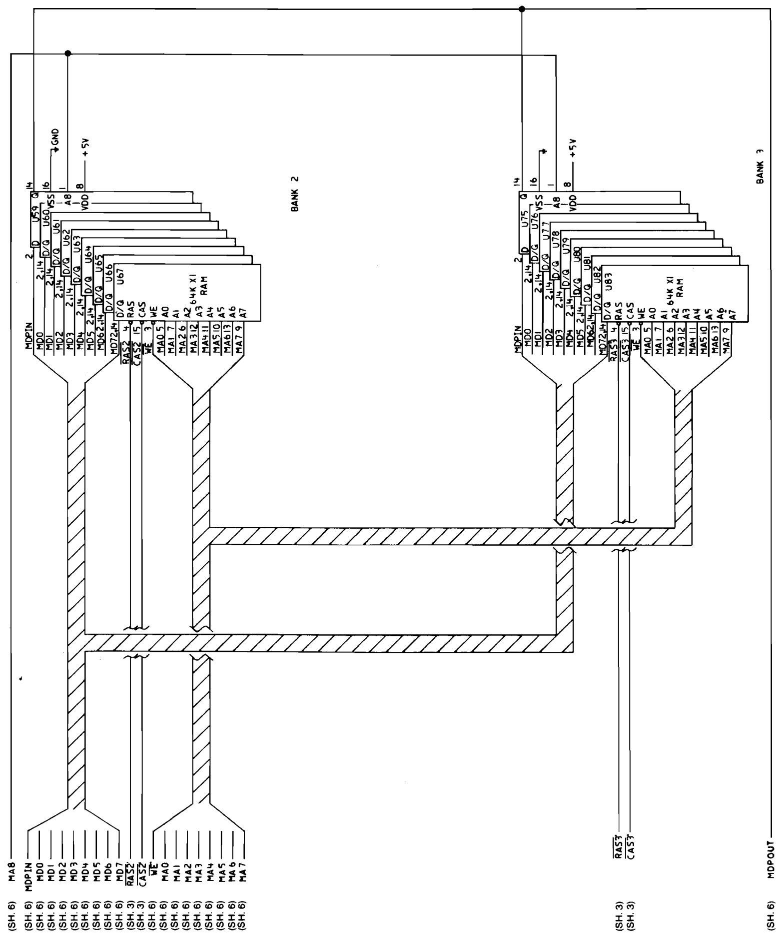
256/640K System Board (Sheet 4 of 11)



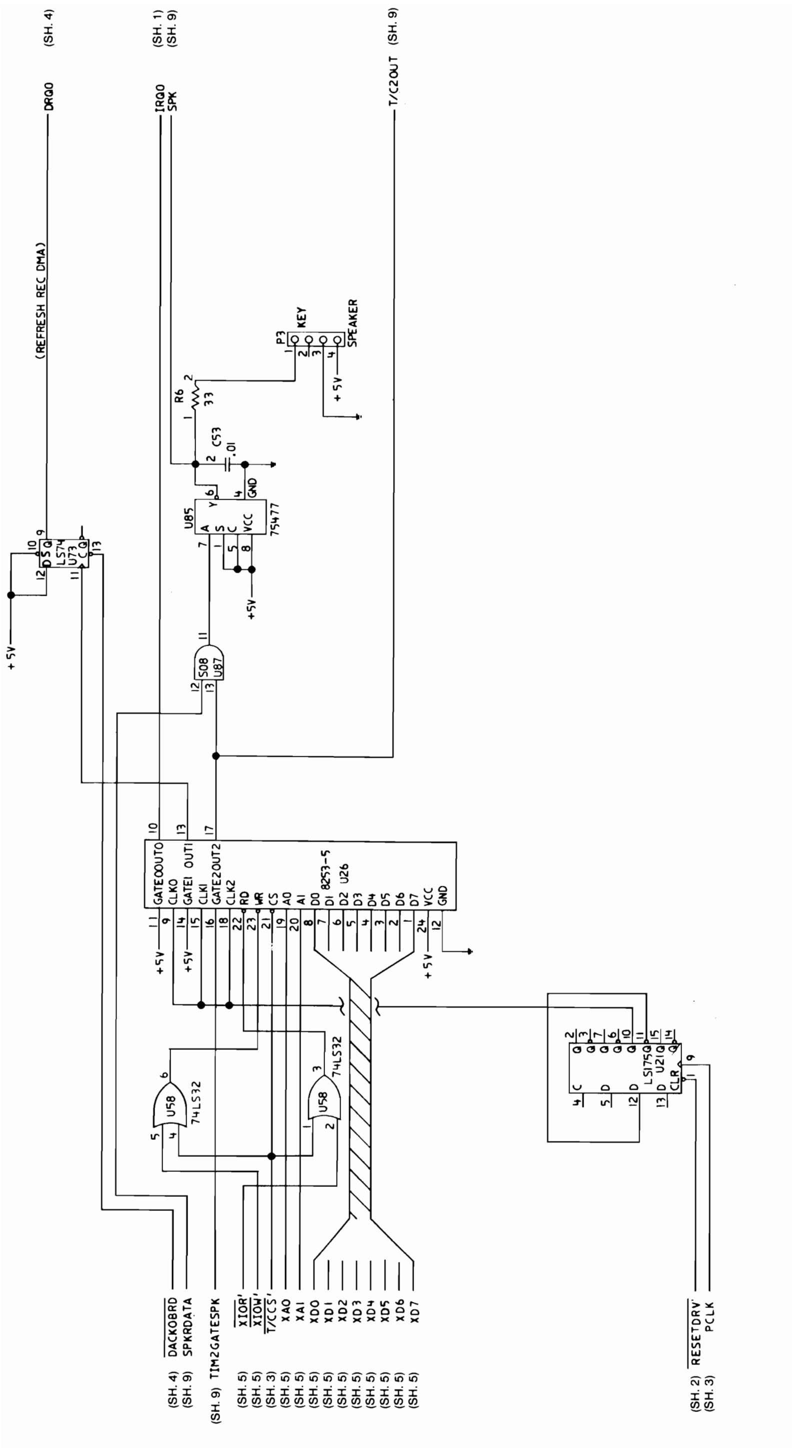
256/640K System Board (Sheet 5 of 11)



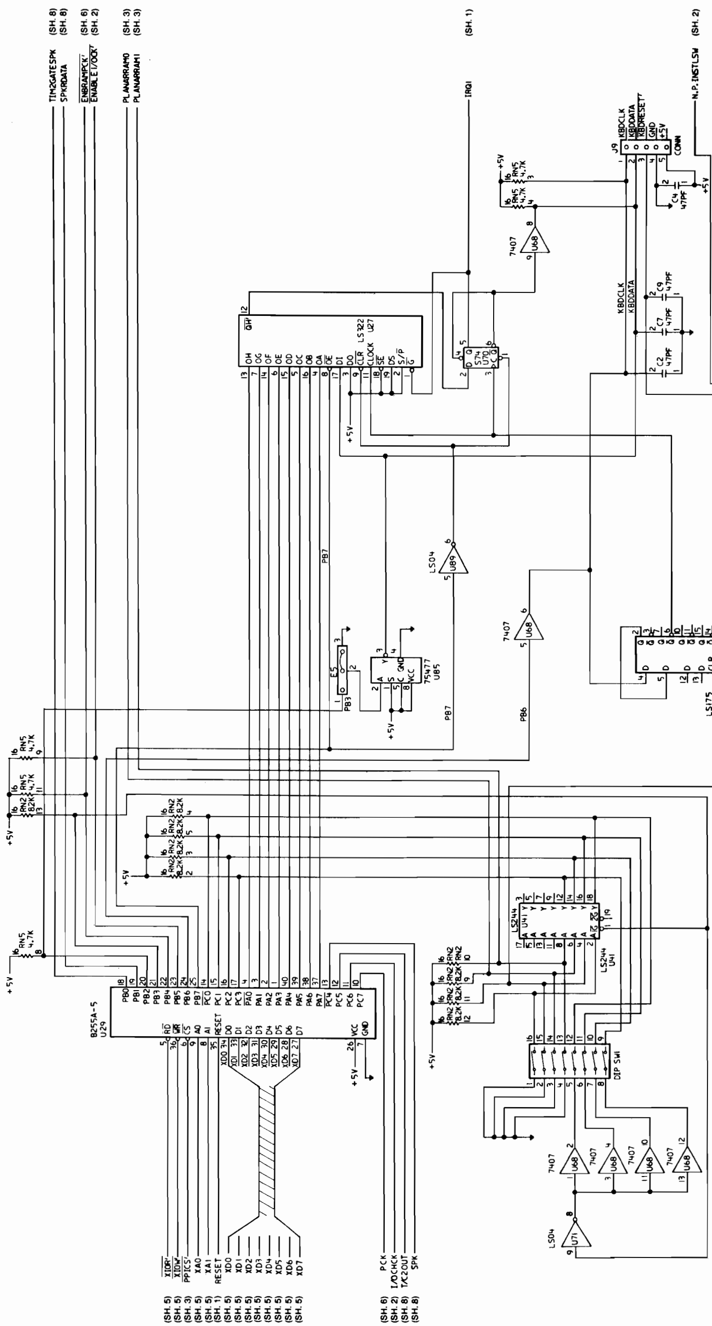
256/640K System Board (Sheet 6 of 11)



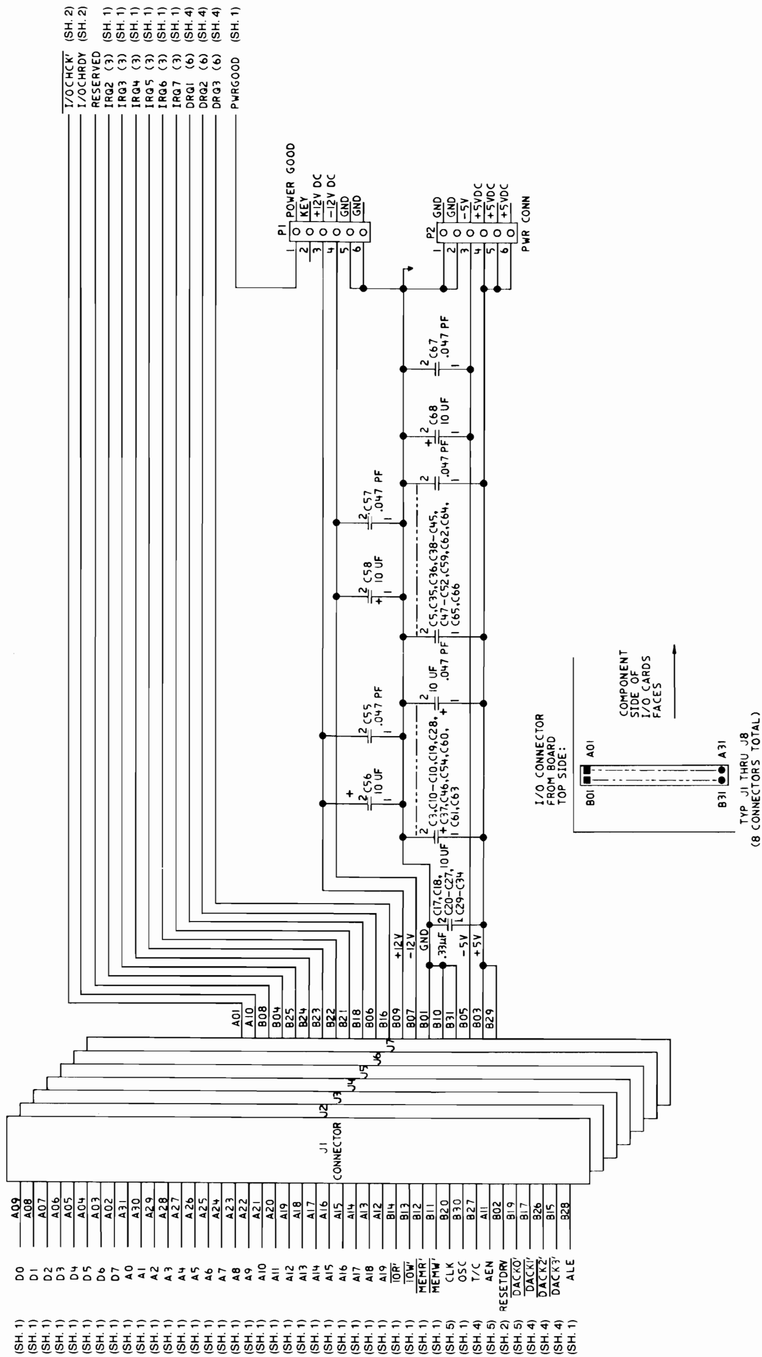
256/640K System Board (Sheet 7 of 11)



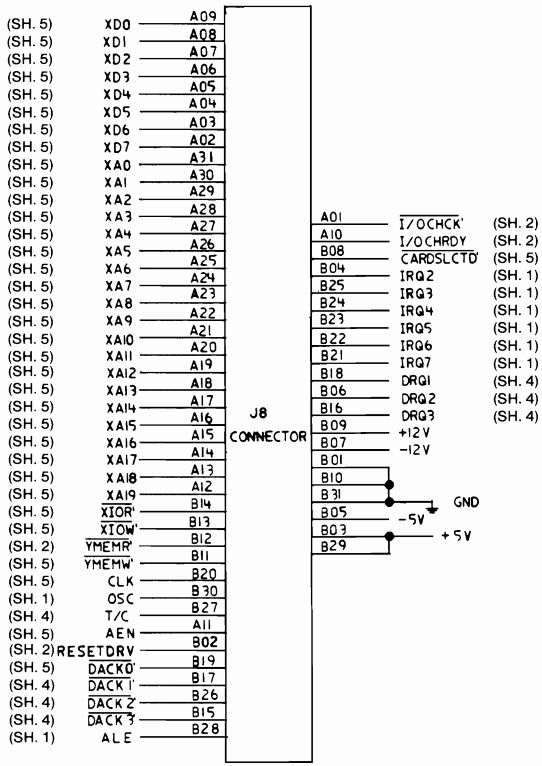
256/640K System Board (Sheet 8 of 11)



256/640K System Board (Sheet 9 of 11)



256/640K System Board (Sheet 10 of 11)



256/640K System Board (Sheet 11 of 11)

Notes:

SECTION 2. COPROCESSOR

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

SECTION 2

Notes:

Description

The Math Coprocessor (8087) enables the IBM Personal Computer to perform high-speed arithmetic, logarithmic functions, and trigonometric operations with extreme accuracy.

The 8087 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 first five bits of every instruction's operation code for the coprocessor are identical (binary 11011). When the microprocessor and the coprocessor see this operation code, the microprocessor calculates the address of any variables in memory, while the coprocessor checks the instruction. The coprocessor takes the memory address from the microprocessor if necessary. To gain access to locations in memory, the coprocessor takes the local bus from the microprocessor when the microprocessor finishes its current instruction. When the coprocessor is finished with the memory transfer, it returns the local bus to the microprocessor.

The IBM Math Coprocessor works with seven numeric data types divided into the three classes listed below.

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

Programming Interface

The coprocessor extends the 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 found 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 figure below 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	$-2\times10^{9} \leq X \leq +2\times10^{9}$
Long Integer	64	18	$-9\times10^{18} \leq X \leq +9\times10^{18}$
Packed Decimal	80	18	-9..99 ≤ X ≤ +9..99 (18 digits)
Short Real *	32	6-7	$8.43\times10^{-37} \leq \|X\| \leq 3.37\times10^{38}$
Long Real *	64	15-16	$4.19\times10^{-307} \leq \|X\| \leq 1.67\times10^{308}$
Temporary Real	80	19	$3.4\times10^{-4932} \leq \|X\| \leq 1.2\times10^{4932}$

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

Data Types

Hardware Interface

The coprocessor uses the same clock generator and system bus interface components as the microprocessor. The microprocessor's queue status lines (QS0 and QS1) enable the coprocessor to obtain and decode instructions simultaneously with the microprocessor. 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 (WAIT FOR NOT BUSY).

When an incorrect instruction is sent to the coprocessor (for example, divide by 0 or load a full register), the coprocessor can signal the microprocessor with an interrupt. There are three conditions that will disable the coprocessor interrupt to the microprocessor:

1. Exception and interrupt-enable bits of the control word are set to 1's

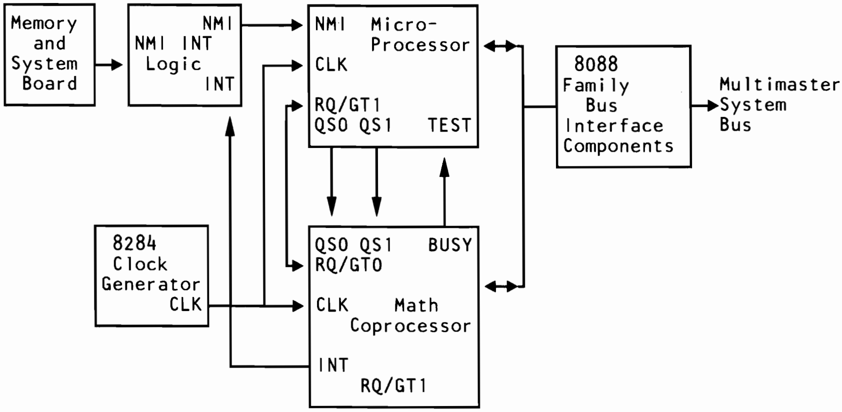
2. System-board switch-block 1, switch 2, set in the On position

3. Non-maskable interrupt register (NMI REG) is set to zero.

At power-on time, the NMI REG is cleared to disable the NMI. Any program using the coprocessor's interrupt capability must ensure that conditions 2 and 3 are never met during the operation of the software or an "Endless WAIT" will occur. An "Endless WAIT" will have the microprocessor waiting for the 'not busy' signal from the coprocessor while the coprocessor is waiting for the microprocessor to interrupt.

Because a memory parity error may also cause an interrupt to the microprocessor NMI line, the program should check the coprocessor status for an exception condition. If a coprocessor exception condition is not found, control should be passed to the normal NMI handler. If an 8087 exception condition is found, the program may clear the exception by executing the FNSAVE or the FNCLEX instruction, and the exception can be identified and acted upon.

The NMI REG and the coprocessor's interrupt are tied to the NMI line through the NMI interrupt logic. Minor modifications to programs designed for use with a coprocessor must be made before the programs will be compatible with the IBM Personal Computer Math Coprocessor.



Coprocessor Interconnection

Detailed information for the internal functions of the Intel 8087 Coprocessor can be found in the books listed in the Bibliography.

SECTION 3. POWER SUPPLIES

IBM Personal Computer XT Power Supply 3-3
Description 3-3
Input Requirements 3-4
Outputs 3-4
Overvoltage/Overcurrent Protection 3-5
Power Good Signal 3-5
Connector Specifications and Pin Assignments 3-6
IBM Portable Personal Computer Power Supply 3-7
Description 3-7
Voltage and Current Requirements 3-7
Power Good Signal 3-8
Connector Specifications and Pin Assignments 3-9

SECTION 3

Notes:

IBM Personal Computer XT Power Supply

Description

The system dc power supply is a 130-watt, 4 voltage-level switching regulator. It is integrated into the system unit and supplies power for the system unit, its options, and the keyboard. The supply provides 15 A of +5 Vdc, plus or minus 5%, 4.2 A of +12 Vdc, plus or minus 5%, 300 mA of -5 Vdc, plus or minus 10%, and 250 mA of -12 Vdc, plus or minus 10%. All power levels are regulated with overvoltage and overcurrent protection. There are two power supplies, 120 Vac and 220/240 Vac. Both are fused. If dc overcurrent or overvoltage conditions exist, the supply automatically shuts down until the condition is corrected. The supply is designed for continuous operation at 130 watts.

The system board takes approximately 2 to 4 A of +5 Vdc, thus allowing approximately 11 A of +5 Vdc for the adapters in the system expansion slots. The +12 Vdc power level is designed to power the internal diskette drives and the 10M or 20M fixed disk drive. The -5 Vdc level is used for analog circuits in the diskette adapter's phase-lock loop. The +12 Vdc and -12 Vdc are used for powering the Electronic Industries Association (EIA) drivers for the communications adapters. All four power levels are bussed across the eight system expansion slots.

The IBM Monochrome Display has its own power supply, receiving its ac power from the system unit's power system. The ac output for the display is switched on and off with the Power switch and is a nonstandard connector.

Input Requirements

The nominal power requirements and output voltages are listed in the following tables.

Voltage @ 50/60 Hz ± 3 Hz		
Nominal Vac	Minimum Vac	Maximum Vac
110 220/240	90 180	137 259
Current: 4.1 A max at 90 Vac		

Input Requirements

Outputs

Nominal Output(Vdc)	Load Current (A) Min	Max	Regulation Tolerance
+5 Vdc	2.3	15.0	+5% to -4%
-5 Vdc	0.0	0.3	+10% to -8%
+12 Vdc	0.4	4.2	+5% to -4%
-12 Vdc	0.0	0.25	+10% to -9%

Vdc Output

Nominal Output(Vac)	Load Current (A) Min	Max	Voltage Limits Min	Max
120	0.0	1.0	90	137
220/240	0.0	0.5	180	259

Vac Output

The sense levels of the dc outputs are:

Output(Vdc)	Minimum (Vdc)	Sense Voltage Nominal (Vdc)	Maximum (Vdc)
+5 Vdc	+4.5	+5.0	+5.5
-5 Vdc	-4.3	-5.0	-5.5
+12 Vdc	+10.8	+12.0	+13.2
-12 Vdc	-10.2	-12.0	-13.2

Vdc Sense Levels

Overvoltage/Overcurrent Protection

Voltage Nominal(Vac)	Type Protection	Rating Amps
110 220/240	Fuse Fuse	5.0 3.5

Voltage and Current Protection

Power Good Signal

When the supply is switched off for a minimum of 1.0 second, and then switched on, the 'power good' signal will be regenerated.

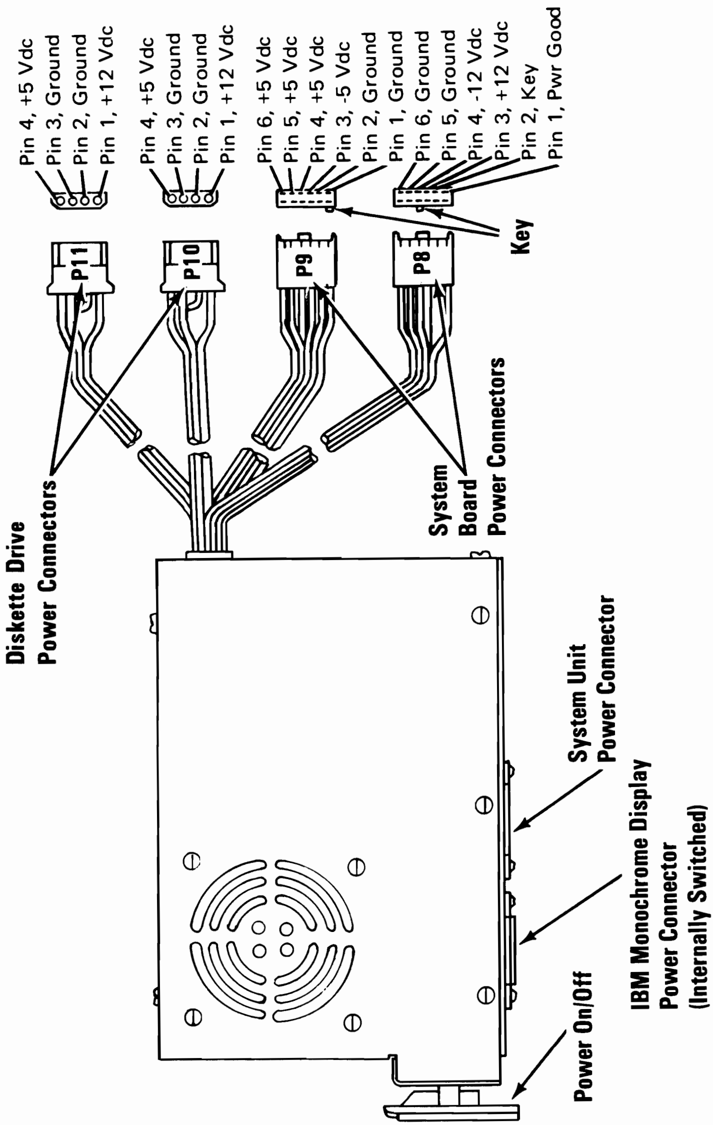
The 'power good' signal indicates that there is adequate power to continue processing. If the power goes below the specified levels, the 'power good' signal triggers a system shutdown.

This signal is the logical AND of the dc output-voltage 'sense' signal and the ac input-voltage 'fail' signal. This signal is TTL-compatible up-level for normal operation or down-level for fault conditions. The ac 'fail' signal causes 'power good' to go to a down level when any output voltage falls below the regulation limits.

The dc output-voltage 'sense' signal holds the 'power good' signal at a down level (during power-on) until all output voltages have reached their respective minimum sense levels. The 'power good' signal has a turn-on delay of at least 100 ms but no greater than 500 ms.

Connector Specifications and Pin Assignments

The power connector on the system board is a 12-pin male connector that plugs into the power-supply connectors. The pin assignments and locations are shown below.



Power Supply and Connectors

IBM Portable Personal Computer Power Supply

Description

The system unit's power supply is a 114-watt, switching regulator that provides five outputs. It supplies power for the system unit and its options, the power supply fan, the diskette drive, the composite display, and the keyboard. All power levels are protected against overvoltage and overcurrent conditions. The input voltage selector switch has 115 Vac and 230 Vac positions. If a dc overload or overvoltage condition exists, the power supply automatically shuts down until the condition is corrected, and the power supply is switched off and then on.

The internal 5-1/4 inch diskette drive uses the +5 Vdc and the +12 Vdc power levels. Both the +12 Vdc and -12 Vdc power levels are used in the drivers and receivers of the optional communications adapters. The display uses a separate +12 Vdc power level. The +5 Vdc, -5 Vdc, +12 Vdc, and -12 Vdc power levels are bussed across the system expansion slots.

SECTION 3

Voltage and Current Requirements

Voltage @ 50/60 Hz ± 3 Hz		
Nominal Vac	Minimum Vac	Maximum Vac
110 220/240	90 180	137 259
Current: 3.5 A max at 90 Vac		

Note: Input voltage to be 50 or 60 hertz, ± 3 hertz.

Nominal Output(Vdc)	Load Current (A) Min	Max	Regulation Tolerance
+5 Vdc	2.3	11.2	+5% to -4%
-5 Vdc	0.0	0.3	+10% to -8%
+12 Vdc	0.04	2.9	+5% to -4%
-12 Vdc	0.0	0.25	+10% to -9%
+12 Vdc (display)	0.5	1.5	+10% to -9%

Vdc Output

Output(Vdc)	Minimum (Vdc)	Sense Voltage Nominal (Vdc)	Maximum (Vdc)
+5 Vdc	+4.5	+5.0	+6.5
-5 Vdc	-4.3	-5.0	-6.5
+12 Vdc	+10.8	+12.0	+15.6
-12 Vdc	-10.2	-12.0	-15.6
+12 Vdc (display)	+10.8	+12.0	+15.6

Vdc Sense Levels

Voltage Nominal(Vac)	Type Protection	Rating Amps
110	Fuse	5.0
220/240	Fuse	2.5

Voltage and Current Protection

Power Good Signal

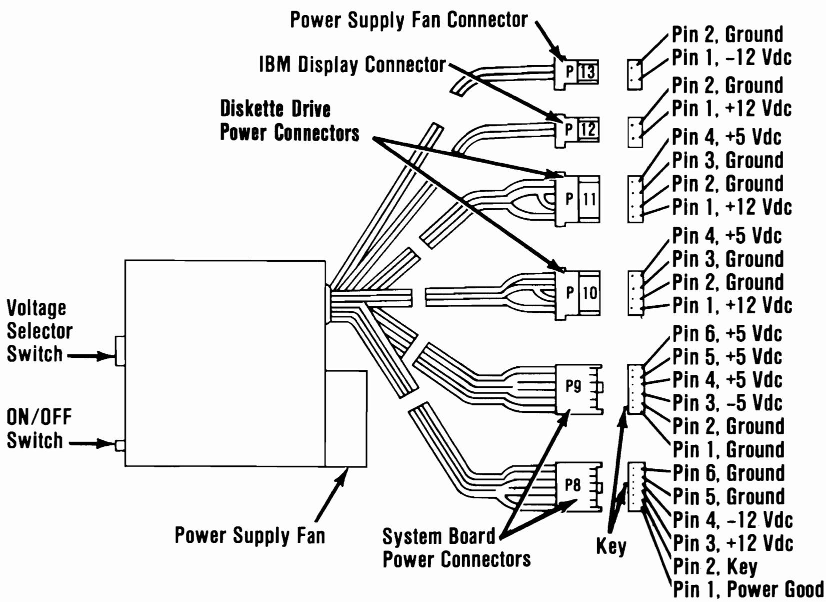
When the power supply is switched off for a minimum of 1 second and then switched on, the 'power good' signal is regenerated.

This signal is the logical **AND** of the dc output-voltage sense signal and the ac input-voltage fail signal. This signal is **TTL**-compatible up-level for normal operation or down-level for fault conditicns. The ac 'fail' signal causes 'power good' to go to a down-level when any output voltage falls below the sense voltage limits.

When power is switched on, the dc output-voltage sense signal holds the 'power good' signal at a down level until all output voltages reach their minimum sense levels. The 'power good' signal has a turn-on delay of 100 to 500 milliseconds.

Connector Specifications and Pin Assignments

The power connector on the system board is a 12-pin connector that plugs into the power supply connectors, P8 and P9. The Input Voltage Selector switch and the pin assignment locations follow.



Power Supply and Connectors

SECTION 3

Notes:

SECTION 4. KEYBOARDS

Introduction 4-3
83-Key Keyboard Description 4-3
Block Diagram 4-5
Keyboard Encoding and Usage 4-6
Encoding 4-6
Character Codes 4-6
Extended Codes 4-9
Extended Functions 4-9
Shift States 4-9
Special Handling 4-11
Extended Functions 4-12
Keyboard Layouts 4-12
French Keyboard 4-13
German Keyboard 4-14
Italian Keyboard 4-15
Spanish Keyboard 4-16
UK Keyboard 4-17
US Keyboard 4-18
Connector Specifications 4-19
Keyboard Logic Diagram 4-21
101/102-Key Keyboard 4-22
Description 4-22
Cables and Connectors 4-23
Sequencing Key-Code Scanning 4-23
Keyboard Buffer 4-24
Keys 4-24
Power-On Routine 4-25
Power-On Reset 4-25
Basic Assurance Test 4-25
Commands from the System 4-26
Reset (Hex FF) 4-26
Commands to the System 4-26
BAT Completion Code (Hex AA) 4-26
BAT Failure Code (Hex FC) 4-26
Key Detection Error (Hex FF) 4-27
Overrun (Hex FF) 4-27
Keyboard Scan Codes 4-28

SECTION 4

Scan Code Tables 4-28
Clock and Data Signals 4-32
Data Stream 4-33
Keyboard Data Output 4-33
Keyboard Encoding and Usage 4-33
Character Codes 4-34
Extended Functions 4-38
Shift States 4-40
Special Handling 4-42
Keyboard Layouts 4-44
French Keyboard 4-45
German Keyboard 4-46
Italian Keyboard 4-47
Spanish Keyboard 4-48
UK English Keyboard 4-49
US English Keyboard 4-50
Specifications 4-51
Power Requirements 4-51
Size 4-51
Weight 4-51
Logic Diagram 4-52

Introduction

Three keyboards are discussed in this section. The 83-key keyboard information for the Personal Computer XT and Portable Personal Computer begins below. Information about the IBM Enhanced Personal Computer Keyboard, hereafter referred to as the 101/102-Key Keyboard, begins on page 4-22.

83-Key Keyboard Description

The Personal Computer XT keyboard has a permanently attached cable that connects to a DIN connector at the rear of the system unit. This shielded 5-wire cable has power (+5 Vdc), ground, and two bidirectional signal lines. The cable is approximately 183 cm (6 ft) long and is coiled, like that of a telephone handset.

The IBM Portable Personal Computer keyboard cable is a detachable, 4-wire, shielded cable that connects to a modular connector in the front panel of the system unit. The cable has power (+5 Vdc), ground, and two bidirectional signal lines in it. It is 762 mm (30 in.) long and is coiled.

Both keyboards use a capacitive technology with a microprocessor (Intel 8048) performing the keyboard scan function. The keyboard has two tilt positions for operator comfort (5- or 15-degree tilt orientations for the Personal Computer XT and 5- or 12-degree tilt orientations for the IBM Portable Personal Computer).

Note: The following descriptions are common to both the Personal Computer XT and IBM Portable Personal Computer.

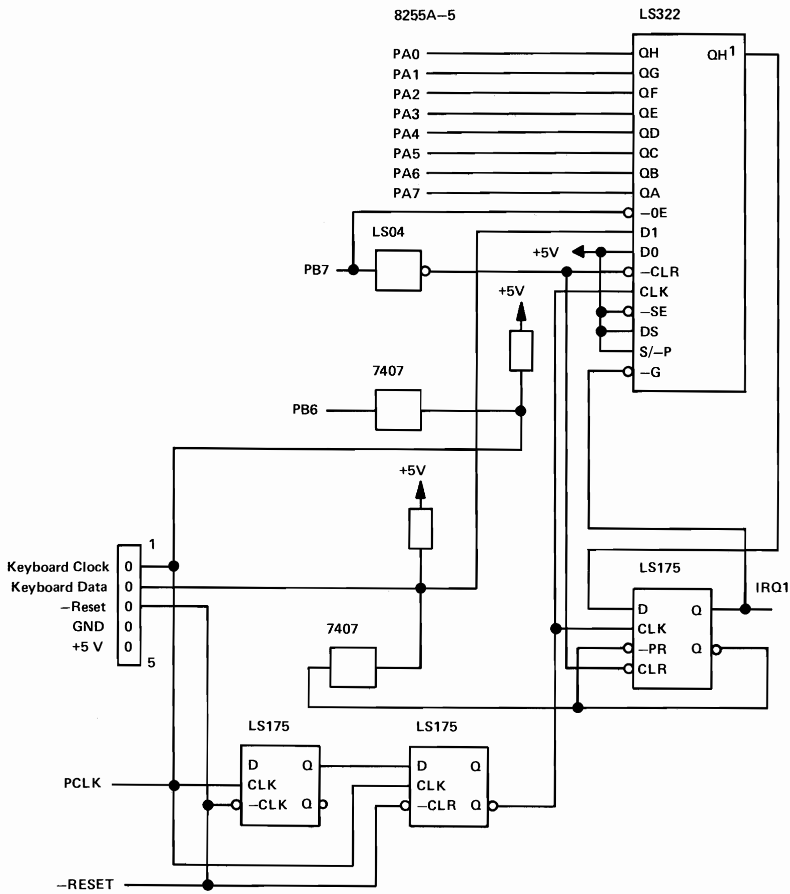
The keyboard has 83 keys arranged in three major groupings. The central portion of the keyboard is a standard typewriter keyboard layout. On the left side are 10 function keys. These keys are user-defined by the software. On the right is a 15-key keypad. These keys are also defined by the software, but have legends for the functions of numeric entry, cursor control, calculator pad, and screen edit.

The keyboard interface is defined so that system software has maximum flexibility in defining certain keyboard operations. This is accomplished by having the keyboard return scan codes rather than American Standard Code for Information Interchange (ASCII) codes. In addition, all keys are typematic (if held down, they will repeat) and generate both a make and a break scan code. For example, key 1 produces scan code hex 01 on make and code hex 81 on break. Break codes are formed by adding hex 80 to make codes. The keyboard I/O driver can define keyboard keys as shift keys or typematic, as required by the application.

The keyboard microprocessor (Intel 8048) performs several functions, including a power-on self test when requested by the system unit. This test checks the keyboard's ROM, tests memory, and checks for stuck keys. Additional functions are keyboard scanning, buffering of up to 16 key scan codes, maintaining bidirectional serial communications with the system unit, and executing the handshake protocol required by each scan-code transfer.

Several different keyboard arrangements are available. These are illustrated on the following pages. For information about the keyboard routines required to implement non-US keyboards, refer to the *Guide to Operations* and *DOS* manuals.

Block Diagram



Keyboard Interface Block Diagram

SECTION 4

Keyboard Encoding and Usage

Encoding

The keyboard routine provided by IBM in the ROM BIOS is responsible for converting the keyboard scan codes into what will be termed "Extended ASCII."

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

Character Codes

The 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 AL.

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

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

Character Codes (Part 1 of 2)

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

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

Character Codes (Part 2 of 2)

Keys 71 through 83 have meaning only in base case, in Num Lock (or shifted) states, or in Ctrl state. Note that the Shift key temporarily reverses the current Num Lock state.

Extended Codes

Extended Functions

For certain functions that cannot be represented in 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 that 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.

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 set of active shift states is available by calling an entry point in the ROM keyboard routine. The key numbers are shown on the keyboard diagrams beginning on page 4-12. The following keys result in altered shift states:

Shift

This key temporarily shifts keys 2–13, 15–27, 30–41, 43–53, 55, 59–68 to uppercase (base case if in Caps Lock state). Also, the Shift key temporarily reverses the Num Lock or non-Num-Lock state of keys 71–73, 75, 77, and 79–83.

Ctrl

This key temporarily shifts keys 3, 7, 12, 14, 16–28, 30–38, 43–50, 55, 59–71, 73, 75, 77, 79, and 81 to the Ctrl state. Also, the Ctrl key used with the Alt and Del keys causes the system reset function; with the Scroll Lock key, the break function; and with the Num Lock key,the pause function. The system reset, break, and pause functions are described in "Special Handling" on the following pages.

Alt

This key temporarily shifts keys 2–13, 16–25, 30–38, 44–50, and 59–68 to the Alt state. Also, the Alt key is used with the Ctrl and Del keys to cause the system reset function described in "Special Handling" on the following pages.

The Alt key has another use. This key allows the user to enter any ASCII character code from 1 to 255 into the system from the keyboard. The user holds down the Alt key and types the decimal value of the characters desired using the numeric keypad (keys 71–73, 75–77, and 79–82). 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 transmitted through the keyboard routine to the system or application program. Alt is handled within the keyboard routine.

Caps Lock

This key shifts keys 16–25, 30–38, and 44–50 to uppercase. Pressing the Caps Lock key a second time reverses the action. Caps Lock is handled within the keyboard routine.

Scroll Lock

This key is interpreted by appropriate application programs as indicating that use of the cursor-control keys should cause windowing over the text rather than cursor movement. Pressing the Scroll Lock key a second time reverses the action. The keyboard routine simply records the current shift state of the Scroll Lock key. It is the responsibility of the system or application program to perform the function.

Shift Key Priorities and Combinations

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

Special Handling

System Reset

The combination of the Alt, Ctrl, and Del keys will result in the keyboard routine initiating the equivalent of a system reset. System reset is handled within the keyboard routine.

Break

The combination of the Ctrl and Break keys will result in the keyboard routine signaling interrupt hex 1B. Also the extended characters (AL = hex 00, AH = hex 00) will be returned.

Pause

The combination of the Ctrl and Num Lock keys will cause the keyboard interrupt routine to loop, waiting for any key except the Num Lock key to be pressed. This provides a system- or application-transparent method of temporarily suspending list, print, and so on, and then resuming the operation. The "unpause" key is thrown away. Pause is handled within the keyboard routine.

Print Screen

The combination of the Shift and PrtSc keys will result in an interrupt invoking the print screen routine. This routine works in the alphameric or graphics mode, with unrecognizable characters printing as blanks.

Extended Functions

The keyboard routine does its own buffering. The keyboard buffer is large enough that few typists will ever fill it. However, if a key is pressed when the buffer is full, the key will be ignored and the "bell" will sound.

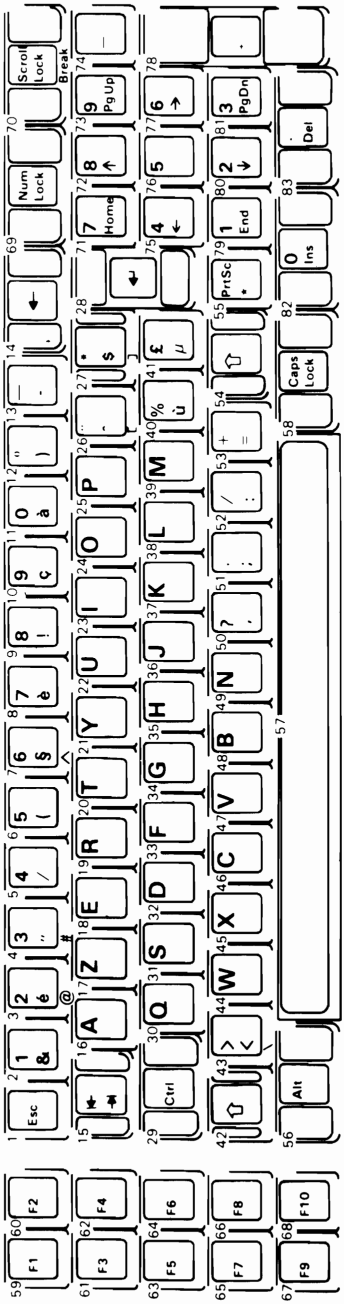
Also, the keyboard routine suppresses the typematic action of the following keys: Ctrl, Shift, Alt, Num Lock, Scroll Lock, Caps Lock, and Ins.

Keyboard Layouts

The IBM Personal Computer keyboard is available in six different layouts as shown on the following pages:

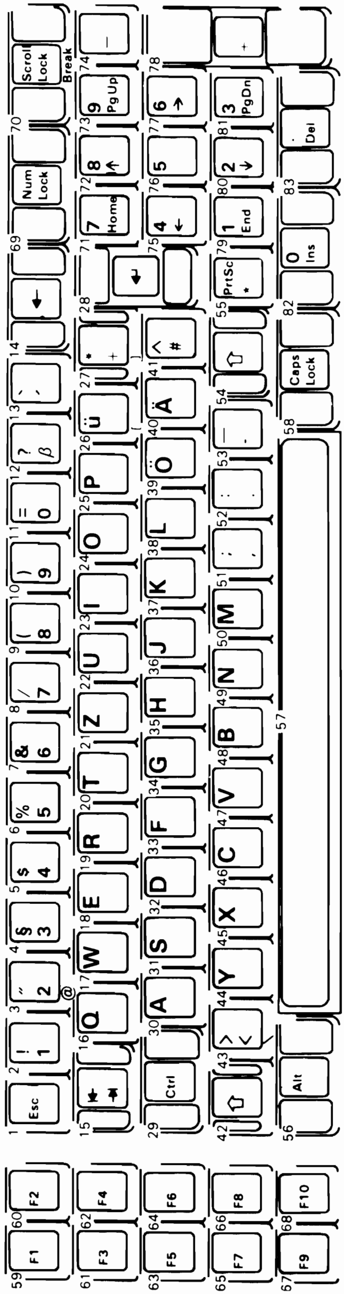
- French
- German
- Italian
- Spanish
- UK English
- US English

French Keyboard



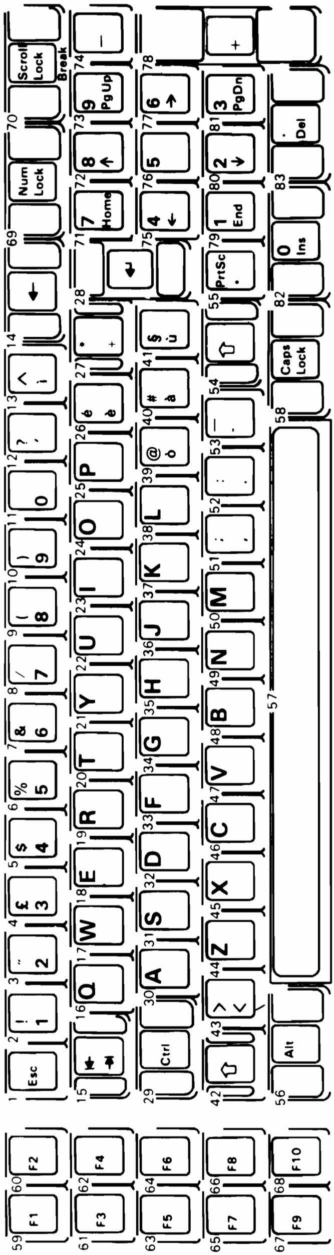
Note: Nomenclature is on both the top and front face of keybuttons as shown. The number to the upper left designates the button position.

German Keyboard



Note: Nomenclature is on both the top and front face of keybuttons as shown. The number to the upper left designates the button position.

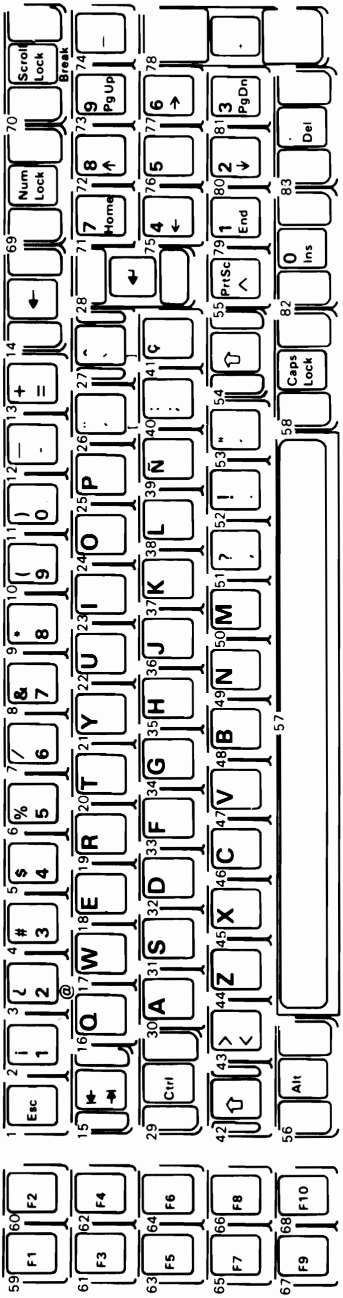
Italian Keyboard



Note: Nomenclature is on both the top and front face of keybuttons as shown. The number to the upper left designates the button position.

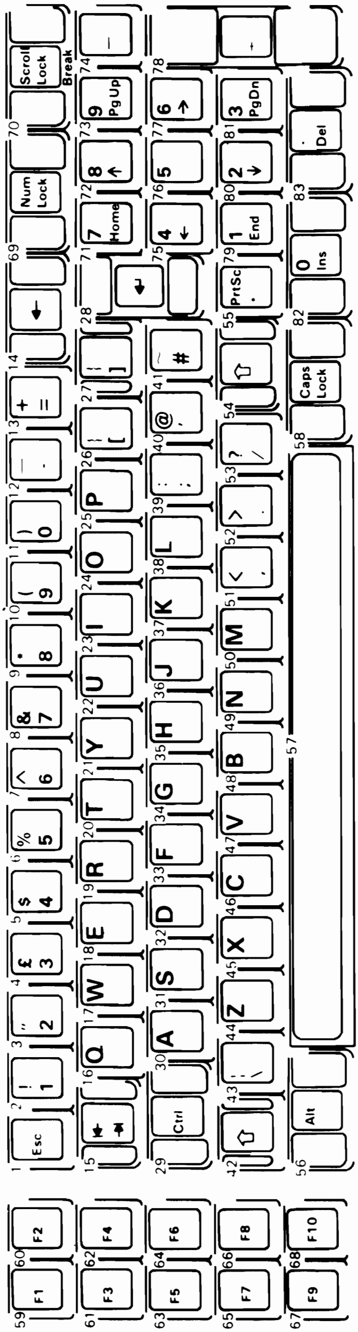
SECTION 4

Spanish Keyboard



Note: Nomenclature is on both the top and front face of keybuttons as shown. The number to the upper left designates the button position.

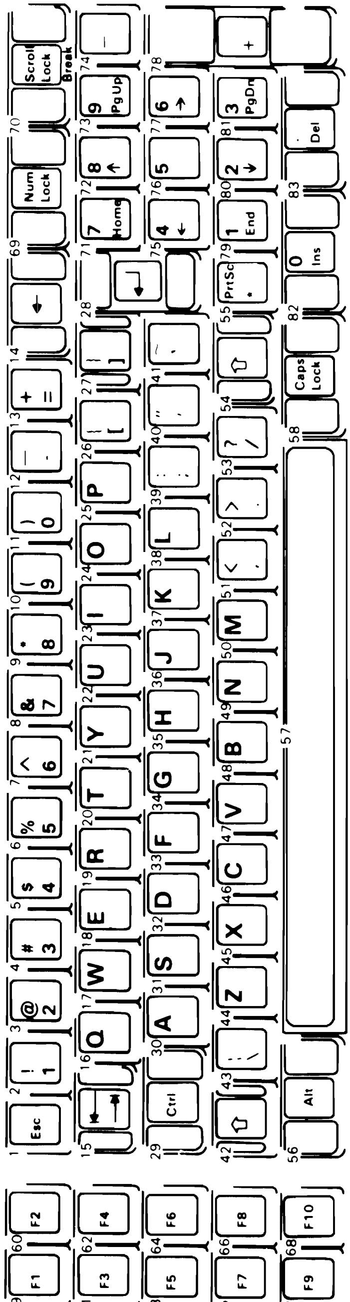
UK Keyboard



Note: Nomenclature is on both the top and front face of keybuttons as shown. The number to the upper left designates the button position.

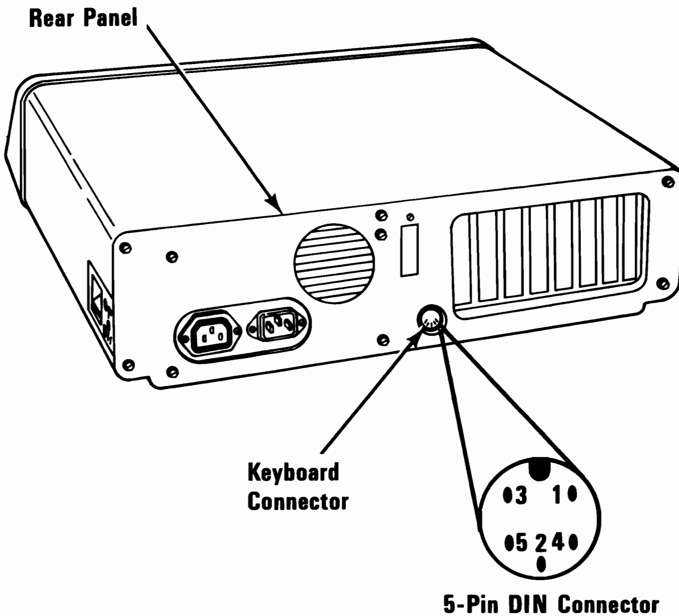
SECTION 4

US Keyboard



Note: Nomenclature is on both the top and front face of keybuttons as shown. The number to the upper left designates the button position.

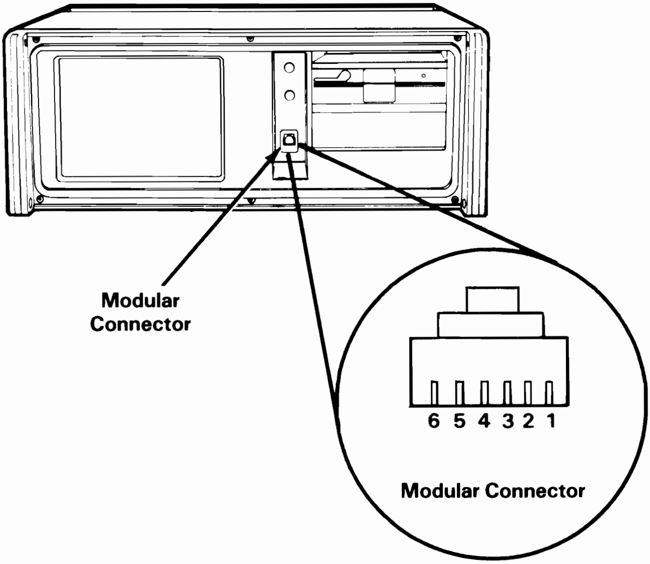
Connector Specifications



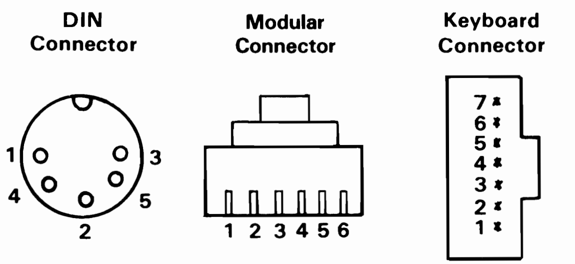
Pin	TTL Signal	Signal Level
1 2 3	+ Keyboard Clock + Keyboard Data - Keyboard Reset (Not used by keyboard	+ 5 Vdc + 5 Vdc
	Power Supply Voltages	Voltage
4 5	Ground + 5 Volts	0 + 5 Vdc

Keyboard Interface Connector Specifications

SECTION 4



Keyboard Cable Connections

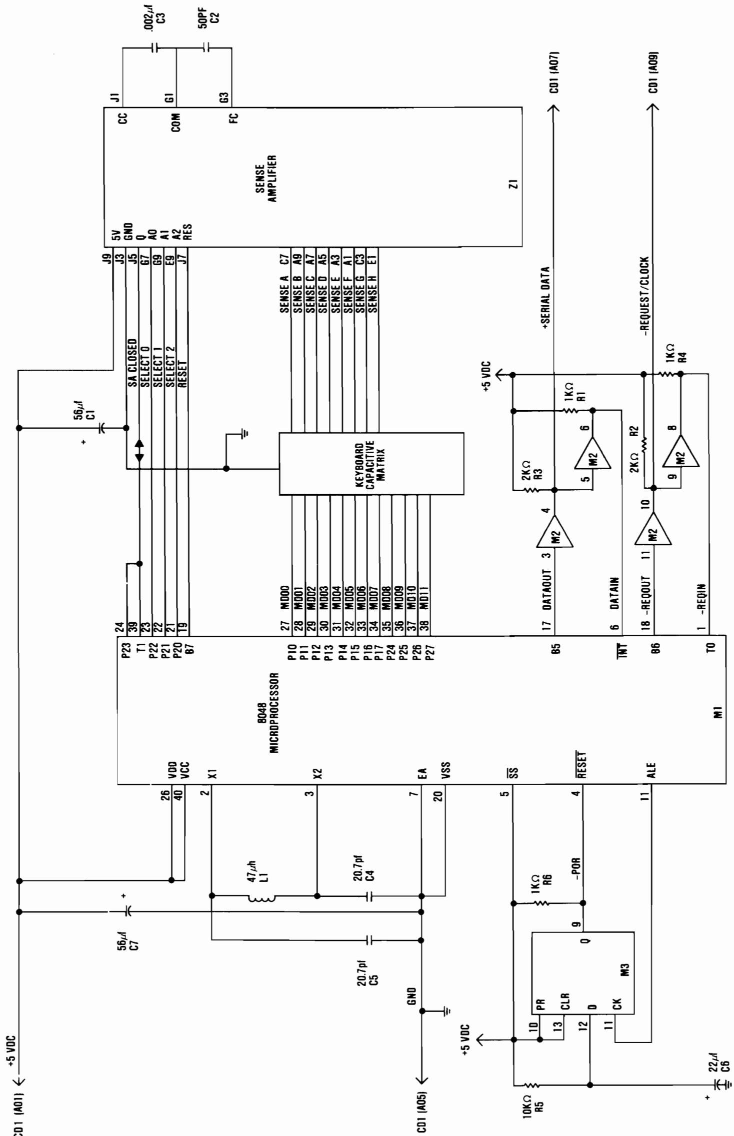


	Pin Side	Pin Side	Wire Side
Clock	1	4	6
Data	2	5	5
Ground	4	3	4
+5 Volts	5	2	2

Modular connector pin 1 is connected to the ground wire going to the chassis.

The ground wire at the keyboard connector is attached to the ground screw on the keyboard logic board.

Keyboard Logic Diagram



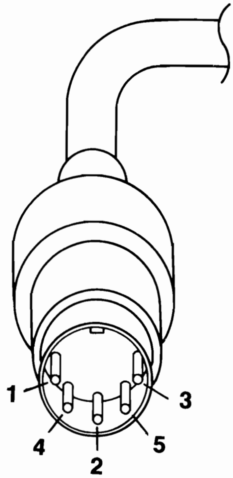
101/102-Key Keyboard

Description

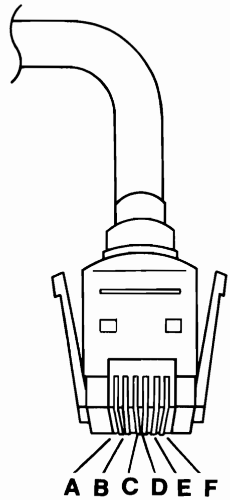
The keyboard has 101 keys (102 in countries outside the U. S.). At system power-on, the keyboard monitors the signals on the 'clock' and 'data' lines and establishes its line protocol.

Cables and Connectors

The keyboard cable connects to the system with a 5-pin DIN connector, and to the keyboard with a 6-position SDL connector. The following table shows the pin configuration and signal assignments.



DIN Connector



SDL Connector

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

Sequencing Key-Code Scanning

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

SECTION 4

Keyboard Buffer

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

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

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

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

Keys

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

Additionally, except for the Pause key, all keys are *typematic*. When a key is pressed and held down, the keyboard sends the make code for that key, delays 500 milliseconds ± 20%, and begins sending a make code for that key at a rate of 10.9 characters per second ± 20%.

If two or more keys are held down, only the last key pressed repeats at the typematic rate. Typematic operation stops when the last key pressed is released, even if other keys are still held down. If a key is pressed and held down while keyboard transmission is inhibited, only the first make code is stored in the buffer. This prevents buffer overflow as a result of typematic action.

Power-On Routine

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

Power-On Reset

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

Basic Assurance Test

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

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

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

Commands from the System

Reset (Hex FF)

The system lowers the 'clock' line for a minimum of 12.5 milliseconds. The keyboard then begins to clock bits on the 'data' line. The result is a Reset command causing the keyboard to reset itself, perform a BAT, and return the appropriate completion code.

Commands to the System

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

Command	Hex Value
BAT Completion Code	AA
BAT Failure Code	FC
Key Detection Error/Overrun	FF

The commands the keyboard sends to the system are described below, in alphabetic order.

BAT Completion Code (Hex AA)

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

BAT Failure Code (Hex FC)

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

Key Detection Error (Hex FF)

The keyboard sends a key detection error character (hex FF) if conditions in the keyboard make it impossible to identify a switch closure.

Overrun (Hex FF)

An overrun character (hex FF) is placed in the keyboard buffer and replaces the last code when the buffer capacity has been exceeded. The code is sent to the system when it reaches the top of the buffer queue.

SECTION 4

Keyboard Scan Codes

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

Scan Code Tables

The following keys send the codes as shown, regardless of any shift states in the keyboard or the system. Refer to "Keyboard Layouts" beginning on page 4-44 to determine the character associated with each key number.

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

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

SECTION 4

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

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

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

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

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

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

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

SECTION 4

Clock and Data Signals

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

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

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

When the keyboard sends data to the system, it generates the 'clock' signal to time the data. The system can prevent the keyboard from sending data by forcing the 'clock' line to an inactive level, or by holding the 'data' line at an inactive level.

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

Data Stream

Data transmissions from the keyboard consist of a 9-bit data stream sent serially over the 'data' line. A logical 1 is sent at an active (high) level. The following table shows the functions of the bits.

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

Keyboard Data Output

When the keyboard is ready to send data, it first checks the status of the keyboard 'clock' line. If the line is active (high), the keyboard issues a request-to-send (RTS) by making the 'clock' line inactive (low). The system must respond with a clear-to-send (CTS), generated by allowing the 'data' line to become active, within 250 microseconds after RTS, or data will be stored in the keyboard buffer. After receiving CTS, the keyboard begins sending the 9 serial bits. The leading edge of the first clock pulse will follow CTS by 60 to 120 microseconds. During each clock cycle, the keyboard clock is active for 25 to 50 microseconds. Each data bit is valid from 2.5 microseconds before the leading edge until 2.5 microseconds after the trailing edge of each keyboard clock cycle.

Keyboard Encoding and Usage

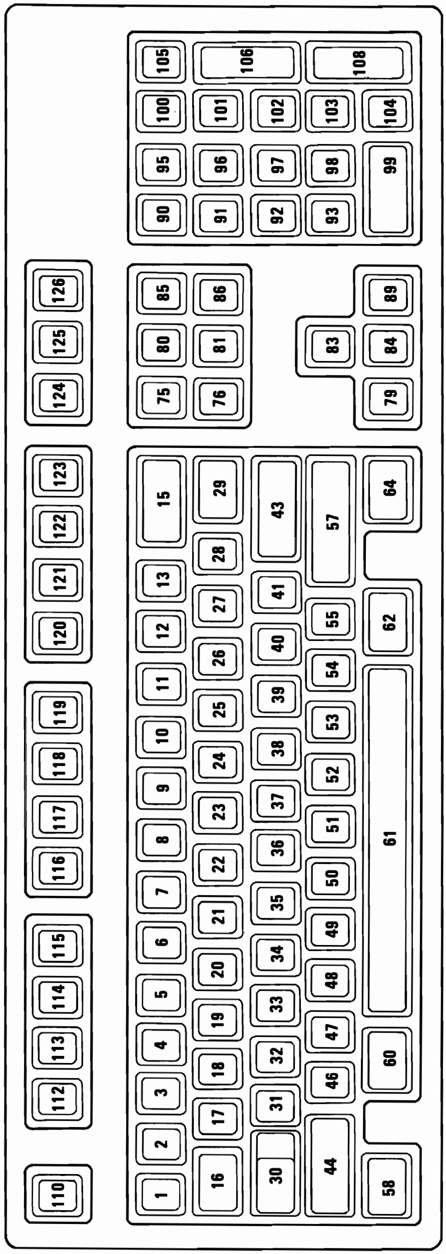
The keyboard routine, provided by IBM in the ROM BIOS, is responsible for converting the keyboard scan codes into what will be termed *Extended ASCII*. The extended ASCII codes returned by the ROM routine are mapped to the US English keyboard layout. Some operating systems may make provisions for alternate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

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

Character Codes

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

The following figure shows the keyboard layout and key positions.



SECTION 4

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

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

Character Codes (Part 1 of 2)

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

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

Character Codes (Part 2 of 2)

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

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

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

Special Character Codes

Extended Functions

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

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

Second Code	Function
1	Alt Esc
3	Nul Character
14	Alt Backspace
15	|← (Back-tab)
16-25	Alt Q, W, E, R, T, Y, U, I, O, P
26-28	Alt [ ] ↵
30-38	Alt A, S, D, F, G, H, J, K, L
39-41	Alt ; ' '
43	Alt \
44-50	Alt Z, X, C, V, B, N, M
51-53	Alt , . /
55	Alt Keypad *
59-68	F1 to F10 Function Keys (Base Case)
71	Home
72	↑ (Cursor Up)
73	Page Up
74	Alt Keypad -
75	← (Cursor Left)
76	Center Cursor
77	→ (Cursor Right)
78	Alt Keypad +
79	End
80	↓ (Cursor Down)
81	Page Down
82	Ins (Insert)
83	Del (Delete)
84-93	Shift F1 to F10
94-103	Ctrl F1 to F10
104-113	Alt F1 to F10
114	Ctrl PrtSc (Start/Stop Echo to Printer)
115	Ctrl ← (Reverse Word)
116	Ctrl → (Advance Word)
117	Ctrl End (Erase to End of Line-EOL)
118	Ctrl PgDn (Erase to End of Screen-EOS)
119	Ctrl Home (Clear Screen and Home)
120-131	Alt 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, -, = keys 2-13
132	Ctrl PgUp (Top 25 Lines of Text and Cursor Home)
133-134	F11, F12
135-136	Shift F11, F12
137-138	Ctrl F11, F12
139-140	Alt F11, F12
141	Ctrl Up/8
142	Ctrl Keypad -
143	Ctrl Keypad 5
144	Ctrl Keypad +
145	Ctrl Down/2
146	Ctrl Ins/0
147	Ctrl Del/.
148	Ctrl Tab
149	Ctrl Keypad /
150	Ctrl Keypad *

Keyboard Extended Functions (Part 1 of 2)

SECTION 4

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

Keyboard Extended Functions (Part 2 of 2)

Shift States

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

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

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

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

The Alt key also allows the user to enter any character code from 0 to 255. The user holds down the Alt key and types the decimal value of the characters desired on the numeric keypad (keys 91 through 93, 96 through 99, and 101 through 103). The Alt key is then released. If the number is greater than 255, a modulo-256 value is used. This value is interpreted as a character code and is sent through the keyboard routine to the system or application program. Alt is handled internal to the keyboard routine.

Caps Lock: This key shifts keys 17 through 26, 31 through 39, and 46 through 52 to uppercase. When Caps Lock is pressed again, it reverses the action. Caps Lock is handled internal to the keyboard routine.

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

Num Lock: This key shifts keys 91 through 93, 96 through 99, and 101 through 104 to uppercase. When Num Lock is pressed again, it reverses the action. Num Lock is handled internal to the keyboard routine.

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

Special Handling

System Reset

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

Break

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

Pause

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

Print Screen

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

System Request

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

Save the previous address.

Overlay interrupt vector hex 15.

Check AH for a value of hex 85:

If yes, process may begin.
If no, go to previous address.

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

Other Characteristics

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

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

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

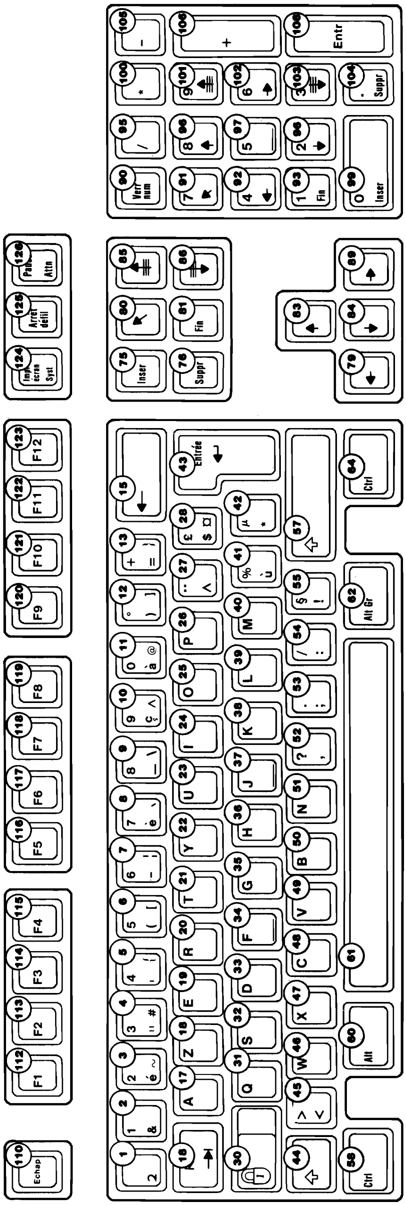
Keyboard Layouts

The keyboard is available in six layouts:

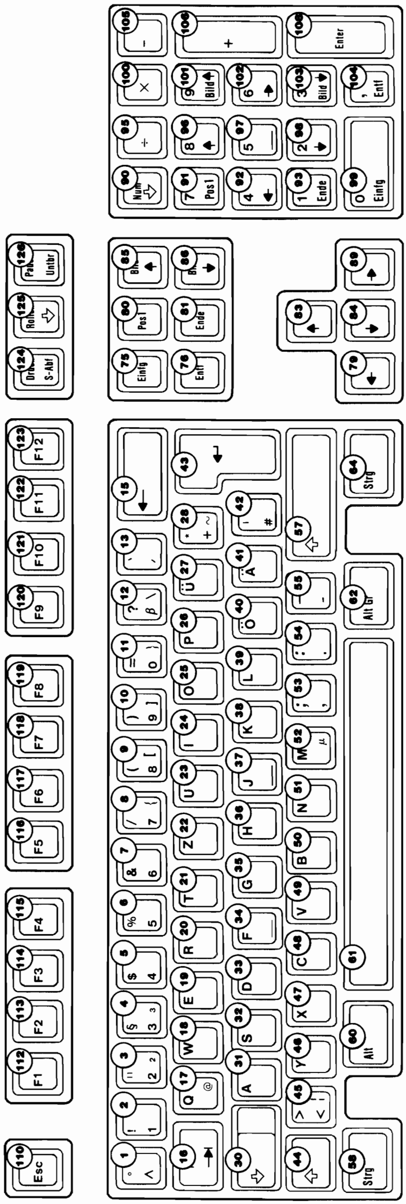
- French
- German
- Italian
- Spanish
- UK English
- US English

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

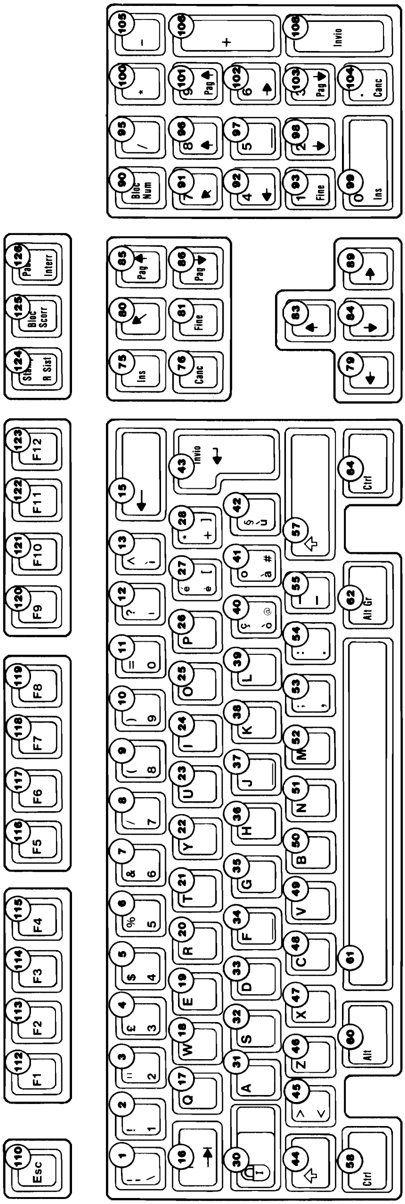
French Keyboard



German Keyboard

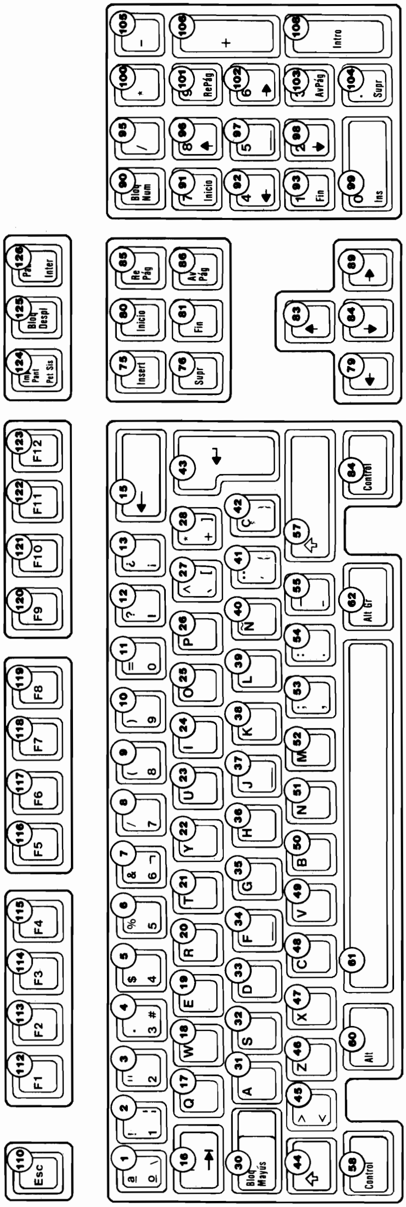


Italian Keyboard

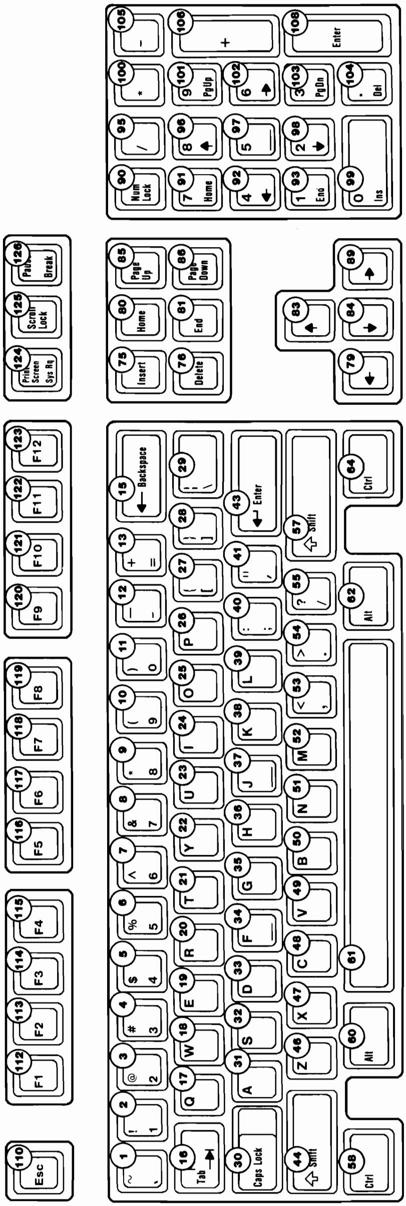


SECTION 4

Spanish Keyboard

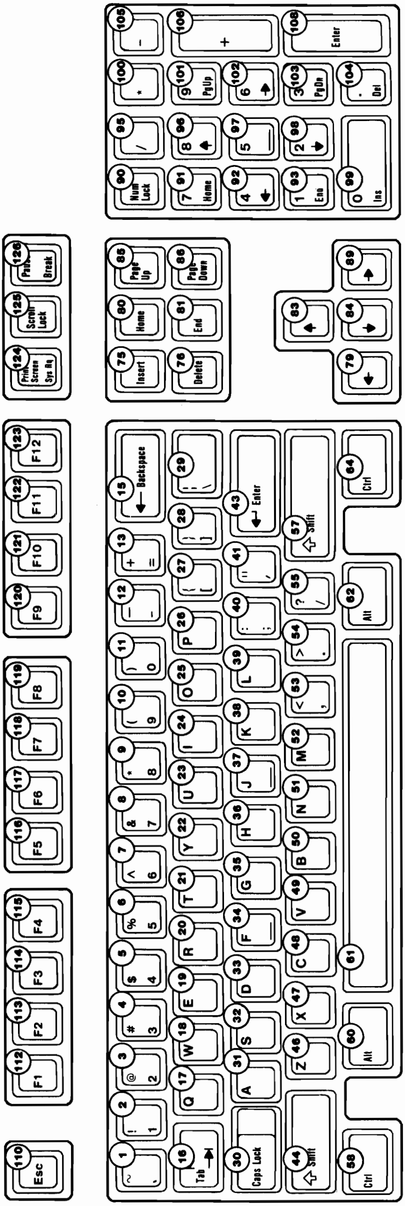


UK English Keyboard



SECTION 4

US English Keyboard



Specifications

The specifications for the keyboard follow.

Power Requirements

- +5 Vdc ± 10%
- Current cannot exceed 275 mA.

Size

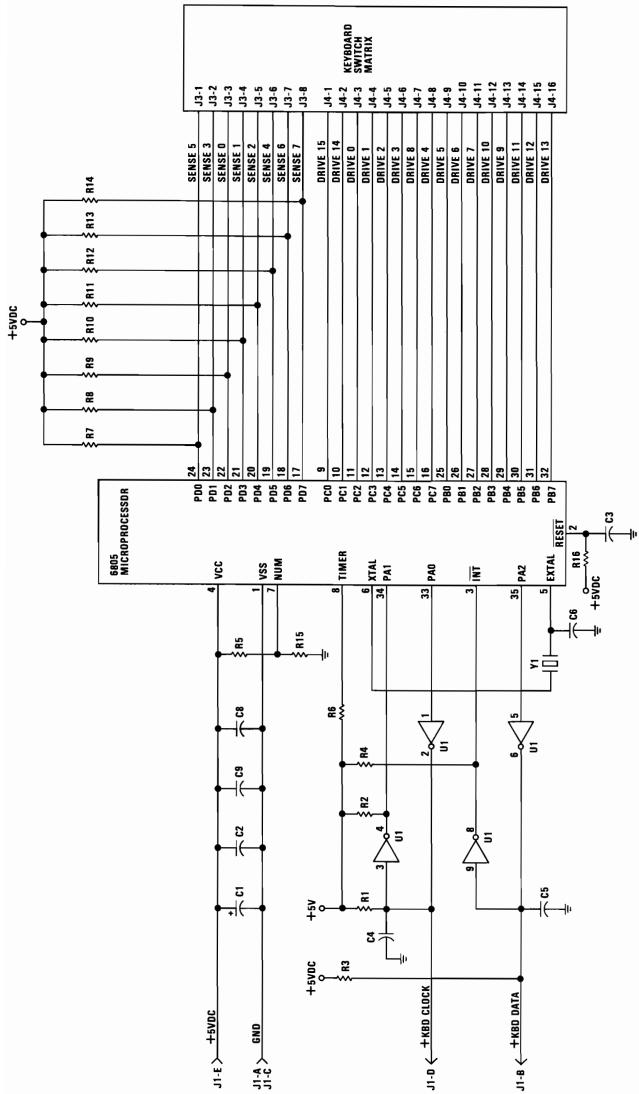
- Length: 492 millimeters (19.4 inches)
- Depth: 210 millimeters (8.3 inches)
- Height: 58 millimeters (2.3 inches), legs extended

Weight

2.25 kilograms (5.0 pounds)

SECTION 4

Logic Diagram



101/102-KEY KEYBOARD

SECTION 5. SYSTEM BIOS

System BIOS Usage 5-3
Vectors with Special Meanings 5-5
System BIOS Listing - 11/22/85 5-11
Quick Reference - 256/640K Board 5-11
System BIOS Listing - 11/8/82 5-111
Quick Reference - 64/256K Board 5-111

SECTION 5

Notes:

System BIOS Usage

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

> **Note:** BIOS listings for both the 256/640 and 64/256 system boards are included in this manual.

The goal is to provide an operational interface to the system and relieve the programmer of the concern about the characteristics of hardware devices. The BIOS interface insulates the user from the hardware, thus allowing new devices to be added to the system, yet retaining the BIOS level interface to the device. In this manner, user programs become transparent to hardware modifications and enhancements.

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 in this section.

Access to the BIOS is through the 8088 software interrupts. Each BIOS entry point is available through its own interrupt.

The software interrupts, hex 10 through hex 1A, each access a different BIOS routine. For example, to determine the amount of memory available in the system,

INT 12H

invokes the BIOS routine for determining memory size and returns the value to the caller.

Parameter Passing

All parameters passed to and from the BIOS routines go through the 8088 registers. The prologue of each BIOS function indicates the registers used on the call and the return. For the memory size example, no parameters are passed. The memory size, in 1K-byte increments, is returned in the AX register.

If a BIOS function has several possible operations, the AH register is used as 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.
```

Generally, 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 is in the prologue of each BIOS function.

Int	Address	Name	BIOS Entry
0	0-3	Divide by Zero	D11
1	4-7	Single Step	D11
2	8-B	Nonmaskable	NMI_INT
3	C-F	Breakpoint	D11
4	10-13	Overflow	D11
5	14-17	Print Screen	PRINT_SCREEN
6	18-1B	Reserved	D11
7	1C-1F	Reserved	D11
8	20-23	Timer	TIMER_INT
9	24-27	Keyboard	KB_INT
A	28-2B	Reserved	D11
B	2C-2F	Communications	D11
C	30-33	Communications	D11

8088 Software Interrupt Listing (Part 1 of 2)

Int	Address	Name	BIOS Entry
D	34-37	Alternate Printer	D11
E	38-3B	Diskette	DISK_INT
F	3C-3F	Printer	D11
10	40-43	Video	VIDEO_IO
11	44-47	Equipment Check	EQUIPMENT
12	48-4B	Memory	MEMORY_SIZE_DETERMINE
13	4C-4F	Diskette	DISKETTE_IO
14	50-53	Communications	RS232_IO
15	54-57	Cassette	CASSETTE_IO
16	58-5B	Keyboard	KEYBOARD_IO
17	5C-5F	Printer	PRINTER_IO
18	60-63	Resident BASIC	F600:0000
19	64-67	Bootstrap	BOOTSTRAP
1A	68-6B	Time of Day	TIME_OF_DAY
1B	6C-6F	Keyboard Break	DUMMY_RETURN
1C	70-73	Timer Tick	DUMMY_RETURN
1D	74-77	Video Initialization	VIDEO_PARMS
1E	78-7B	Diskette Parameters	DISK_BASE
1F	7C-7F	Video Graphics Chars	0
40	100-103	Diskette pointer save area for Fixed Disk	
41	104-107	Fixed Disk Parameters	FD_TBL
5A	168-16B	Cluster	D000:XXXX
5B	16C-16F	Used by Cluster Program	
60-67	180-19F	Reserved for User Programs	

8088 Software Interrupt Listing (Part 2 of 2)

Note: For BIOS index, see the BIOS Quick Reference on page 5-11 or 5-111.

Vectors with Special Meanings

Interrupt Hex 1B - Keyboard Break Address

This vector points to the code to be used when the Ctrl and Break keys are pressed on the keyboard. The vector is invoked while responding to the keyboard interrupt, and control should be returned through an IRET instruction. The power-on routines initialize this vector 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. Also, all I/O devices should be reset in case an operation was underway at that time.

Interrupt Hex 1C - Timer Tick

This vector points to the code to be executed on 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. It is the responsibility of the application to save and restore all registers that will be modified.

Interrupt Hex 1D - Video Parameters

This vector points to a data region containing the parameters required for the initialization of the 6845 on the video card. Note 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 Hex 1E - Diskette Parameters

This vector points to a data region containing the parameters required for the diskette drive. The power-on routines initialize the 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 the other drives attached.

Interrupt Hex 1F - Graphics Character Extensions

When operating in the graphics modes of the IBM Color/Graphics Monitor Adapter (320 by 200 or 640 by 200), the read/write character interface forms the character from the ASCII code point, using a set of dot patterns. The dot patterns for the first 128 code points are contained in ROM. To access the second 128 code points, this vector must be established to point at a table of up to 1K bytes, where each code point is represented by eight bytes of graphic information. At power-on, this vector is initialized to 000:0, and it is the responsibility of the user to change this vector if additional code points are required.

Interrupt Hex 40 - Reserved

When an IBM Fixed Disk Adapter is installed, the BIOS routines use interrupt hex 30 to revector the diskette pointer.

Interrupt Hex 41 - Fixed Disk Parameters

This vector points to a data region containing the parameters required for the fixed disk drive. The power-on routines initialize the vector to point to the parameters contained in the ROM disk routine. These default parameters represent the specified values for any IBM fixed disk drives attached to the system. Changing this parameter block may be necessary to reflect the specifications of the other fixed disk 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 hex 407 contain the base addresses of any RS-232C cards attached to the system. Locations hex 408 to hex 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 in a different area, the area must be set by the application.

Interrupt	Address	Function
20	80-83	DOS program terminate
21	84-87	DOS function call
22	88-8B	DOS terminate address
23	8C-8F	DOS Ctrl Break exit address
24	90-93	DOS fatal error vector
25	94-97	DOS absolute disk read
26	98-9B	DOS absolute disk write
27	9C-9F	DOS terminate, fix in storage
28-3F	A0-FF	Reserved for DOS
40-5F	100-17F	Reserved for BIOS
60-67	180-19F	Reserved for user program interrupts
68-6F	1A0-1BF	Not used
80-85	200-217	Reserved for BASIC
86-F0	218-3C3	Used by BASIC interpreter while BASIC is running
F1-FF	3C4-3FF	Not used

Hardware, Basic, and DOS Interrupts

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 DEF SEG (Default workspace segment):

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

Basic Workspace Variables

Example

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

L	H
Hex 64	Hex 00

Starting Address	
00000	BIOS interrupt vectors
00080	Available interrupt vectors
00400	BIOS data area
00500	User read/write memory
C8000	Disk Adapter
F0000	Read only memory
FE000	BIOS program area

BIOS Memory Map

BIOS Programming Hints

The BIOS code is invoked through software interrupts. The programmer should not "hard code" BIOS addresses into application programs. 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 on diskette reads to ensure the problem is not due to motor startup.

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

Adapter Cards with System-Accessible ROM Modules

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

The absolute addresses hex C8000 through hex F4000 are scanned in 2K blocks in search of a valid adapter card 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 (length/512). A checksum is also done to test the integrity of the ROM module. Each byte in the defined ROM is summed modulo hex 100. This sum must be 0 for the module to be deemed 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 card may now perform its power-on initialization tasks. The feature ROM should return control to the BIOS routines by executing a far return.

System BIOS Listing - 01/10/86

Quick Reference - 256/640K Board

Map **5-13**

Header **5-14**
EQUATES 5-15
ABS0 5-19
DATA Segment 5-20

Diskette **5-23**
INT 13H 5-23
Drive Type 5-25
Diskette__IO__1 5-25
DMA__Setup 5-36
Motor__On 5-40
Disk__Int 5-44
Diskette__Setup 5-45

Keyboard BIOS **5-46**
Scan Tables 5-56

Printer BIOS **5-57**

RS232 BIOS **5-59**

Video BIOS **5-62**

BIOS1 .... **5-80**
INT 15H 5-80
Joystick Support 5-82

POST **5-84**
Determine Configuration 5-87
8259 Test 5-89
Keyboard Test 5-90
Expansion Test 5-91
Boot__Strap (INT 19H) 5-94
Time__of__Day (INT 1AH) 5-95
Beep 5-96

SECTION 5

STGTST_CNT 5-97
Disk_Base 5-99
NMI 5-100
DDS 5-103
Timer_Int 5-103
Character Generator 5-104
D11 5-107
Print Screen 5-108

Address	Publics by Name
F000:E729	A1
F000:15CC	ACT_DISP_PAGE
F000:6000	BASIC
F000:EC5C	BEEP
F000:1C4F	CASSETTE_IO_1
F000:E73C	CONF_TBL
F000:FA6E	CRT_CHAR_GEN
F000:FA12	DDS
F000:0062	DISKETTE_IO_1
F000:EFC7	DISK_BASE
F000:0BC4	DISK_INT_1
F000:0BDB	DSKETTE_SETUP
F000:1D37	FILL
F000:0000	HEADER
F000:0D78	KB_INT_1
F000:0C57	KEYBOARD_IO_1
F000:F0E4	M5
F000:F0EC	M6
F000:F0F4	M7
F000:EF79	MD_TBL1
F000:EF86	MD_TBL2
F000:EF93	MD_TBL3
F000:EFA0	MD_TBL4
F000:EFAD	MD_TBL5
F000:EFBA	MD_TBL6
F000:0A40	NEC_OUTPUT
F000:12BE	PRINTER_IO_1
F000:FFF0	P_O_R
F000:1725	READ_AC_CURRENT
F000:15B5	READ_CURSOR
F000:1865	READ_DOT
F000:1BAB	READ_LPEN
F000:E05B	RESET
F000:0B32	RESULTS
F000:1344	RS232_IO_1
F000:16D3	SCROLL_DOWN
F000:1635	SCROLL_UP
F000:0A64	SEEK
F000:15EE	SET_COLOR
F000:158D	SET_CPOS
F000:156C	SET_CTYPE
F000:1485	SET_MODE
F000:144E	VIDEO_IO_1
F000:F0A4	VIDEO_PARMS
F000:1563	VIDEO_RETURN
F000:1614	VIDEO_STATE
F000:ECA0	WAITF
F000:1782	WRITE_AC_CURRENT
F000:17B4	WRITE_C_CURRENT
F000:1876	WRITE_DOT
F000:17E1	WRITE_STRING
F000:1B24	WRITE_TTY

Address	Publics by Value
F000:0000	HEADER
F000:0062	DISKETTE_IO_1
F000:0A40	NEC_OUTPUT
F000:0A64	SEEK
F000:0B32	RESULTS
F000:0BC4	DISK_INT_1
F000:0BDB	DSKETTE_SETUP
F000:0C57	KEYBOARD_IO_1
F000:0D78	KB_INT_1
F000:12BE	PRINTER_IO_1
F000:1344	RS232_IO_1
F000:144E	VIDEO_IO_1
F000:1485	SET_MODE
F000:1563	VIDEO_RETURN
F000:156C	SET_CTYPE
F000:158D	SET_CPOS
F000:15B5	READ_CURSOR
F000:15CC	ACT_DISP_PAGE
F000:15EE	SET_COLOR
F000:1614	VIDEO_STATE
F000:1635	SCROLL_UP
F000:16D3	SCROLL_DOWN
F000:1725	READ_AC_CURRENT
F000:1782	WRITE_AC_CURRENT
F000:17B4	WRITE_C_CURRENT
F000:17E1	WRITE_STRING
F000:1865	READ_DOT
F000:1876	WRITE_DOT
F000:1B24	WRITE_TTY
F000:1BAB	READ_LPEN
F000:1C4F	CASSETTE_IO_1
F000:1D37	FILL
F000:6000	BASIC
F000:E05B	RESET
F000:E729	A1
F000:E73C	CONF_TBL
F000:EC5C	BEEP
F000:ECA0	WAITF
F000:EF79	MD_TBL1
F000:EF86	MD_TBL2
F000:EF93	MD_TBL3
F000:EFA0	MD_TBL4
F000:EFAD	MD_TBL5
F000:EFBA	MD_TBL6
F000:EFC7	DISK_BASE
F000:F0A4	VIDEO_PARMS
F000:F0E4	M5
F000:F0EC	M6
F000:F0F4	M7
F000:FA12	DDS
F000:FA6E	CRT_CHAR_GEN
F000:FFF0	P_O_R

SECTION 5

```
PAGE 118,121
TITLE HEADER --- 01/08/86  POWER ON SELF TEST (POST)

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



;---------------------------------------------------------------------------
;   MODULE REFERENCE                                                       :
;                                                                          :
;    HEADER.ASM     -->  DEFINITIONS                                       :
;      DSEG.INC     -->   DATA SEGMENT LOCATIONS                           :
;      POSTEQU.INC  -->   COMMON EQUATES FOR POST AND BIOS                 :
;                                                                          :
;    DSKETTE.ASM    -->  DISKETTE BIOS                                     :
;                          DISKETTE_IO_1 - INT 13H BIOS ENTRY (40H)   -INT 13H :
;                          DISK_INT_1    - HARDWARE INTERRUPT HANDLER -INT 0EH :
;                          DSKETTE_SETUP - POST SETUP DRIVE TYPES          :
;                                                                          :
;    KEYBRD.ASM     -->  KEYBOARD BIOS                                     :
;                          KEYBOARD_IO_1 - INT 16H BIOS ENTRY         -INT 16H :
;                          KB_INT_1      - HARDWARE INTERRUPT         -INT 09H :
;                          SND_DATA      - KEYBOARD TRANSMISSION           :
;                                                                          :
;    PRT.ASM        -->  PRINTER ADAPTER BIOS                         -INT 17H :
;                                                                          :
;    RS232.ASM      -->  COMMUNICATIONS BIOS FOR RS232                -INT 14H :
;                                                                          :
;    VIDEO.ASM      -->  VIDEO BIOS                                   -INT 10H :
;                                                                          :
;    BIOS1.ASM      -->  INTERRUPT 15H BIOS ROUTINES                  -INT 15H :
;                          DEV_OPEN      - NULL DEVICE OPEN HANDLER        :
;                          DEV_CLOSE     - NULL DEVICE CLOSE HANDLER       :
;                          PROG_TERM     - NULL PROGRAM TERMINATION        :
;                          JOY_STICK     - JOYSTICK PORT HANDLER           :
;                          SYS_REQ       - NULL SYSTEM REQUEST KEY         :
;                          EXT_MEMORY    - EXTENDED MEMORY SIZE DETERMINE  :
;                          DEVICE_BUSY   - NULL DEVICE BUSY HANDLER        :
;                          INT_COMPLETE  - NULL INTERRUPT COMPLETE HANDLER :
;                                                                          :
;    POST.ASM       -->  BIOS INTERRUPT ROUTINES                           :
;                          POST          - POWER ON SELF TEST & INITIALIZATION :
;                          TIME_OF_DAY_1 - TIME OF DAY ROUTINES       -INT 1AH :
;                          PRINT_SCREEN1 - PRINT SCREEN ROUTINE       -INT 05H :
;                          TIMER_INT_1   - TIMER1 INTERRUPT HANDLER  ->INT 1CH :
;                          DDS           - LOAD (DS:) WITH DATA SEGMENT    :
;                          BEEP          - SPEAKER BEEP CONTROL ROUTINE    :
;                          WAITF         - FIXED TIME WAIT ROUTINE         :
;                                                                          :
;---------------------------------------------------------------------------
.LIST
```

```
                            PAGE
                        C   INCLUDE POSTEQU.INC
                        C   ;---------------------------------------
                        C   ;        EQUATES USED BY POST AND BIOS      ;
                        C   ;---------------------------------------
                        C
 = 0000                 C   SYSTEM          EQU     0               ; 0 PC-XT, 1 PC-AT
 = 00FB                 C   MODEL_BYTE      EQU     0FBH            ; SYSTEM MODEL BYTE
 = 0000                 C   SUB_MODEL_BYTE  EQU     000H            ; SYSTEM SUB-MODEL TYPE
 = 0001                 C   BIOS_LEVEL      EQU     001H            ; BIOS REVISION LEVEL
                        C   ENDIF
                        C
                        C   ;--------- KEYBOARD INTERFACE AND DIAGNOSTIC CONTROL REGISTERS -----------------
 = 0060                 C   PORT_A          EQU     060H            ; KEYBOARD SCAN CODE/CONTROL PORT
 = 0061                 C   PORT_B          EQU     061H            ; PORT B READ/WRITE DIAGNOSTIC REGISTER
 = 0062                 C   PORT_C          EQU     062H            ; 8255 PORT C ADDR
 = 0063                 C   CMD_PORT        EQU     063H
                        C
                        C
 = 0060                 C   KB_DATA         EQU     60H             ; KEYBOARD SCAN CODE PORT
 = 0061                 C   KB_CTL          EQU     61H             ; CONTROL BITS FOR KEYBOARD SENSE DATA
 = 0054                 C   ID_2A           EQU     054H            ; ALTERNATE 2ND ID CHAR FOR KBX
 = 00E0                 C   MC_E0           EQU     224             ; GENERAL MARKER CODE
 = 00E1                 C   MC_E1           EQU     225             ; PAUSE KEY MARKER CODE
                        C
                        C
 = 00F3                 C   RAM_PAR_ON      EQU     11110011B       ; AND MASK FOR PARITY CHECKING ENABLE ON
 = 000C                 C   RAM_PAR_OFF     EQU     00001100B       ; OR MASK FOR PARITY CHECKING ENABLE OFF
 = 00C0                 C   PARITY_ERR      EQU     11000000B       ; R/W MEMORY - I/O CHANNEL PARITY ERROR
 = 0001                 C   GATE2           EQU     00000001B       ; TIMER 2 INPUT GATE CLOCK BIT
 = 0002                 C   SPK2            EQU     00000010B       ; SPEAKER OUTPUT DATA ENABLE BIT
 = 0010                 C   REFRESH_BIT     EQU     00010000B       ; REFRESH TEST BIT
 = 0020                 C   OUT2            EQU     00100000B       ; SPEAKER TIMER OUT2 INPUT BIT
 = 0040                 C   IO_CHECK        EQU     01000000B       ; I/O (MEMORY) CHECK OCCURRED BIT MASK
 = 0080                 C   PARITY_CHECK    EQU     10000000B       ; MEMORY PARITY CHECK OCCURRED BIT MASK
                        C   ENDIF
```

```
                          C  PAGE
                          C  ;---------- KEYBOARD RESPONSE ------------------------------------------------
 = 00AA                   C  KB_OK           EQU     0AAH            ; RESPONSE FROM SELF DIAGNOSTIC
 = 00FA                   C  KB_ACK          EQU     0FAH            ; ACKNOWLEDGE FROM TRANSMISSION
 = 00FE                   C  KB_RESEND       EQU     0FEH            ; RESEND REQUEST
 = 00FF                   C  KB_OVER_RUN     EQU     0FFH            ; OVER RUN SCAN CODE
                          C
                          C  ;---------- FLAG EQUATES WITHIN @KB_FLAG --------------------------------------
 = 0001                   C  RIGHT_SHIFT     EQU     00000001B       ; RIGHT SHIFT KEY DEPRESSED
 = 0002                   C  LEFT_SHIFT      EQU     00000010B       ; LEFT SHIFT KEY DEPRESSED
 = 0004                   C  CTL_SHIFT       EQU     00000100B       ; CONTROL SHIFT KEY DEPRESSED
 = 0008                   C  ALT_SHIFT       EQU     00001000B       ; ALTERNATE SHIFT KEY DEPRESSED
 = 0010                   C  SCROLL_STATE    EQU     00010000B       ; SCROLL LOCK STATE HAS BEEN TOGGLED
 = 0020                   C  NUM_STATE       EQU     00100000B       ; NUM LOCK STATE HAS BEEN TOGGLED
 = 0040                   C  CAPS_STATE      EQU     01000000B       ; CAPS LOCK STATE HAS BEEN TOGGLED
 = 0080                   C  INS_STATE       EQU     10000000B       ; INSERT STATE IS ACTIVE
                          C
                          C  ;---------- FLAG EQUATES WITHIN @KB_FLAG_1 ------------------------------------
 = 0004                   C  SYS_SHIFT       EQU     00000100B       ; SYSTEM KEY DEPRESSED AND HELD
 = 0008                   C  HOLD_STATE      EQU     00001000B       ; SUSPEND KEY HAS BEEN TOGGLED
 = 0010                   C  SCROLL_SHIFT    EQU     00010000B       ; SCROLL LOCK KEY IS DEPRESSED
 = 0020                   C  NUM_SHIFT       EQU     00100000B       ; NUM LOCK KEY IS DEPRESSED
 = 0040                   C  CAPS_SHIFT      EQU     01000000B       ; CAPS LOCK KEY IS DEPRESSED
 = 0080                   C  INS_SHIFT       EQU     10000000B       ; INSERT KEY IS DEPRESSED
                          C
                          C  ;---------- FLAGS EQUATES WITHIN @KB_FLAG_2 -----------------------------------
 = 0007                   C  KB_LEDS         EQU     00000111B       ; KEYBOARD LED STATE BITS
                          C  ;               EQU     00001000B       ; RESERVED (MUST BE ZERO)
 = 0010                   C  KB_FA           EQU     00010000B       ; ACKNOWLEDGMENT RECEIVED
 = 0020                   C  KB_FE           EQU     00100000B       ; RESEND RECEIVED FLAG
 = 0040                   C  KB_PR_LED       EQU     01000000B       ; MODE INDICATOR UPDATE
 = 0080                   C  KB_ERR          EQU     10000000B       ; KEYBOARD TRANSMIT ERROR FLAG
                          C
                          C  ;---------- FLAGS EQUATES WITHIN @KB_FLAG_3 -----------------------------------
 = 0001                   C  LC_E1           EQU     00000001B       ; LAST CODE WAS THE E1 HIDDEN CODE
 = 0002                   C  LC_E0           EQU     00000010B       ; LAST CODE WAS THE E0 HIDDEN CODE
 = 0004                   C  R_CTL_SHIFT     EQU     00000100B       ; RIGHT CTL KEY DOWN
 = 0008                   C  GRAPH_ON        EQU     00001000B       ; ALL GRAPHICS KEY DOWN (W.T. ONLY)
                          C  ;               EQU     00011000B       ; RESERVED (MUST BE ZERO)
 = 0010                   C  KBX             EQU     00010000B       ; KBX INSTALLED
 = 0020                   C  SET_NUM_LK      EQU     00100000B       ; FORCE NUM LOCK IF READ ID AND KBX
 = 0040                   C  LC_AB           EQU     01000000B       ; LAST CHARACTER WAS FIRST ID CHARACTER
 = 0080                   C  RD_ID           EQU     10000000B       ; DOING A READ ID (MUST BE BIT0)
                          C
                          C  ;---------- KEYBOARD SCAN CODES -----------------------------------------------
 = 00AB                   C  ID_1            EQU     0ABH            ; 1ST ID CHARACTER FOR KBX
 = 0041                   C  ID_2            EQU     041H            ; 2ND ID CHARACTER FOR KBX
 = 0038                   C  ALT_KEY         EQU     56              ; SCAN CODE FOR  ALTERNATE SHIFT KEY
 = 001D                   C  CTL_KEY         EQU     29              ; SCAN CODE FOR  CONTROL KEY
 = 003A                   C  CAPS_KEY        EQU     58              ; SCAN CODE FOR  SHIFT LOCK KEY
 = 0053                   C  DEL_KEY         EQU     83              ; SCAN CODE FOR  DELETE KEY
 = 0052                   C  INS_KEY         EQU     82              ; SCAN CODE FOR  INSERT KEY
 = 002A                   C  LEFT_KEY        EQU     42              ; SCAN CODE FOR  LEFT SHIFT
 = 0045                   C  NUM_KEY         EQU     69              ; SCAN CODE FOR  NUMBER LOCK KEY
 = 0036                   C  RIGHT_KEY       EQU     54              ; SCAN CODE FOR  RIGHT SHIFT
 = 0046                   C  SCROLL_KEY      EQU     70              ; SCAN CODE FOR  SCROLL LOCK KEY
 = 0054                   C  SYS_KEY         EQU     84              ; SCAN CODE FOR  SYSTEM KEY
 = 0057                   C  F11_M           EQU     87              ; F11 KEY MAKE
 = 0058                   C  F12_M           EQU     88              ; F12 KEY MAKE
```

```
                        C  PAGE
                        C  ENDIF
                        C
                        C  ;---------- DISKETTE EQUATES --------------------------------------------------
 = 0050                 C  CARD_ID         EQU     01010000B       ; CONTROLLER CARD I.D. BIT
 = 0001                 C  DUAL            EQU     00000001B       ; MASK FOR FDC ADAPTER I.D.
 = 0080                 C  INT_FLAG        EQU     10000000B       ; INTERRUPT OCCURRENCE FLAG
 = 0080                 C  DSK_CHG         EQU     10000000B       ; DISKETTE CHANGE FLAG MASK BIT
 = 0010                 C  DETERMINED      EQU     00010000B       ; SET STATE DETERMINED IN STATE BITS
 = 0010                 C  HOME            EQU     00010000B       ; TRACK 0 MASK
 = 0004                 C  SENSE_DRV_ST    EQU     00000100B       ; SENSE DRIVE STATUS COMMAND
 = 0030                 C  TRK_SLAP        EQU     030H            ; CRASH STOP (48 TPI DRIVES)
 = 000A                 C  QUIET_SEEK      EQU     00AH            ; SEEK TO TRACK 10
 = 0002                 C  MAX_DRV         EQU     2               ; MAX NUMBER OF DRIVES
 = 000F                 C  HD12_SETTLE     EQU     15              ; 1.2 M HEAD SETTLE TIME
 = 0014                 C  HD320_SETTLE    EQU     20              ; 320 K HEAD SETTLE TIME
 = 0025                 C  MOTOR_WAIT      EQU     37              ; 2 SECONDS OF COUNTS FOR MOTOR TURN OFF
                        C
                        C  ;---------- DISKETTE ERRORS ---------------------------------------------------
 = 0080                 C  TIME_OUT        EQU     080H            ; ATTACHMENT FAILED TO RESPOND
 = 0040                 C  BAD_SEEK        EQU     040H            ; SEEK OPERATION FAILED
 = 0020                 C  BAD_NEC         EQU     020H            ; DISKETTE CONTROLLER HAS FAILED
 = 0010                 C  BAD_CRC         EQU     010H            ; BAD CRC ON DISKETTE READ
 = 000C                 C  MED_NOT_FND     EQU     00CH            ; MEDIA TYPE FOUND
 = 0009                 C  DMA_BOUNDARY    EQU     009H            ; ATTEMPT TO DMA ACROSS 64K BOUNDARY
 = 0008                 C  BAD_DMA         EQU     008H            ; DMA OVERRUN ON OPERATION
 = 0006                 C  MEDIA_CHANGE    EQU     006H            ; MEDIA REMOVED ON DUAL ATTACH CARD
 = 0004                 C  RECORD_NOT_FND  EQU     004H            ; REQUESTED SECTOR NOT FOUND
 = 0003                 C  WRITE_PROTECT   EQU     003H            ; WRITE ATTEMPTED ON WRITE PROTECT DISK
 = 0002                 C  BAD_ADDR_MARK   EQU     002H            ; ADDRESS MARK NOT FOUND
 = 0001                 C  BAD_CMD         EQU     001H            ; BAD COMMAND PASSED TO DISKETTE I/O
                        C
                        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
                        C  ;---------- MEDIA/DRIVE STATE INDICATORS -------------------------------------
 = 0001                 C  TRK_CAPA        EQU     00000001B       ; 80 TRACK CAPABILITY
 = 0002                 C  FMT_CAPA        EQU     00000010B       ; MULTIPLE FORMAT CAPABILITY (1.2M)
 = 0004                 C  DRV_DET         EQU     00000100B       ; DRIVE DETERMINED
 = 0010                 C  MED_DET         EQU     00010000B       ; MEDIA DETERMINED BIT
 = 0020                 C  DBL_STEP        EQU     00100000B       ; DOUBLE STEP BIT
 = 00C0                 C  RATE_MSK        EQU     11000000B       ; MASK FOR CLEARING ALL BUT RATE
 = 0000                 C  RATE_500        EQU     00000000B       ; 500 KBS DATA RATE
 = 0040                 C  RATE_300        EQU     01000000B       ; 300 KBS DATA RATE
 = 0080                 C  RATE_250        EQU     10000000B       ; 250 KBS DATA RATE
 = 000C                 C  STRT_MSK        EQU     00001100B       ; OPERATION START RATE MASK
 = 00C0                 C  SEND_MSK        EQU     11000000B       ; MASK FOR SEND RATE BITS
                        C
                        C  ;---------- MEDIA/DRIVE STATE INDICATORS COMPATIBILITY -----------------------
 = 0000                 C  M3D3U           EQU     00000000B       ; 360 MEDIA/DRIVE NOT ESTABLISHED
 = 0001                 C  M3D1U           EQU     00000001B       ; 360 MEDIA,1.2DRIVE NOT ESTABLISHED
 = 0002                 C  M1D1U           EQU     00000010B       ; 1.2 MEDIA/DRIVE NOT ESTABLISHED
 = 0007                 C  MED_UNK         EQU     00000111B       ; NONE OF THE ABOVE
```

SECTION 5

```
                    C   PAGE
                    C   ;---------- INTERRUPT EQUATES ---------------------------------------------------
 = 0020             C   EOI             EQU     020H            ; END OF INTERRUPT COMMAND TO 8259
 = 0020             C   INTA00          EQU     020H            ; 8259 PORT
 = 0021             C   INTA01          EQU     021H            ; 8259 PORT
 = 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   ;-------------------------------------------------------------------------------
 = 0008             C   DMA08           EQU     008H            ; DMA STATUS REGISTER PORT ADDRESS
 = 0000             C   DMA             EQU     000H            ; DMA CH.0 ADDRESS REGISTER PORT ADDRESS
 = 00D0             C   DMA18           EQU     0D0H            ; 2ND DMA STATUS PORT ADDRESS
 = 00C0             C   DMA1            EQU     0C0H            ; 2ND DMA CH.0 ADDRESS REGISTER ADDRESS
                    C   ;-------------------------------------------------------------------------------
 = 0040             C   TIMER           EQU     040H            ; 8253 TIMER - BASE ADDRESS
 = 0043             C   TIM_CTL         EQU     043H            ; 8253 TIMER CONTROL PORT ADDR
 = 0040             C   TIMER0          EQU     040H            ; 8253 TIMER/CNTER 0 PORT ADDR
                    C
                    C   ;--------- MANUFACTURING PORT ---------------------------------------------------
 = 0080             C   MFG_PORT        EQU     80H             ; MANUFACTURING AND POST CHECKPOINT PORT
                    C                                           ;  DMA CHANNEL 0 PAGE REGISTER ADDRESS
                    C
                    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)
 = 0080             C   KEY_FAIL        EQU     10000000B       ; KEYBOARD LOCKED               (ERROR 302)
                    C
                    C   ;-------------------------------------------------------------------------------
 = 0081             C   DMA_PAGE        EQU     081H            ; START OF DMA PAGE REGISTERS
 = 008F             C   LAST_DMA_PAGE   EQU     08FH            ; LAST DMA PAGE REGISTER
                    C
                    C   ;-------------------------------------------------------------------------------
                    C   ;X287            EQU     0F0H            ; MATH COPROCESSOR CONTROL PORT
                    C
                    C   ;-------------------------------------------------------------------------------
 = 0000             C   POST_SS         EQU     00000H          ; POST STACK SEGMENT
 = 8000             C   POST_SP         EQU     08000H          ; POST STACK POINTER
 = 0030             C   STACK_SS        EQU     30H             ; STACK SEGMENT USED DURING POST
 = 0100             C   TOS             EQU     100H            ; STACK -- USED DURING POST ONLY
                    C                                           ;   USE WILL OVERLAY INTERRUPTS VECTORS
                    C
                    C
                    C   ;-------------------------------------------------------------------------------
 = 000D             C   CR              EQU     000DH           ; CARRIAGE RETURN CHARACTER
 = 000A             C   LF              EQU     000AH           ; LINE FEED CHARACTER
 = 0008             C   RVRT            EQU     00001000B       ; VIDEO VERTICAL RETRACE BIT
 = 0001             C   RHRZ            EQU     00000001B       ; VIDEO HORIZONTAL RETRACE BIT
 = 0100             C   H               EQU     256             ; HIGH BYTE FACTOR (X 100H)
 = 0101             C   X               EQU     H+1             ; HIGH AND LOW BYTE FACTOR (X 101H)

                        .LIST
```

```
                         PAGE
                       C INCLUDE DSEG.INC
                       C ELSE
                       C ;--------------------------------------
                       C ;        8088 INTERRUPT LOCATIONS     :
                       C ;        REFERENCED BY POST & BIOS    :
                       C ;--------------------------------------
                       C ENDIF
                       C
0000                   C ABS0           SEGMENT AT 0           ; ADDRESS= 0000:0000
                       C
0000 ??                C @STG_LOC0      DB      ?              ; START OF INTERRUPT VECTOR TABLE
                       C
0008                   C                ORG     4*002H
0008 ????????          C @NMI_PTR       DD      ?              ; NON-MASKABLE INTERRUPT VECTOR
                       C
0014                   C                ORG     4*005H
0014 ????????          C @INT5_PTR      DD      ?              ; PRINT SCREEN INTERRUPT VECTOR
                       C
0020                   C                ORG     4*008H
0020 ????????          C @INT_PTR       DD      ?              ; HARDWARE INTERRUPT POINTER (8-F)
                       C
0040                   C                ORG     4*010H
0040 ????????          C @VIDEO_INT     DD      ?              ; VIDEO I/O INTERRUPT VECTOR
                       C
004C                   C                ORG     4*013H
004C ????????          C @ORG_VECTOR    DD      ?              ; DISKETTE/DISK INTERRUPT VECTOR
                       C
0060                   C                ORG     4*018H
0060 ????????          C @BASIC_PTR     DD      ?              ; POINTER TO CASSETTE BASIC
                       C
0074                   C                ORG     4*01DH
0074 ????????          C @PARM_PTR      DD      ?              ; POINTER TO VIDEO PARAMETERS
                       C
0078                   C                ORG     4*01EH
0078 ????????          C @DISK_POINTER  DD      ?              ; POINTER TO DISKETTE PARAMETER TABLE
                       C
007C                   C                ORG     4*01FH
007C ????????          C @EXT_PTR       DD      ?              ; POINTER TO GRAPHIC CHARACTERS 128-255
                       C
0100                   C                ORG     4*040H
0100 ????????          C @DISK_VECTOR   DD      ?              ; POINTER TO DISKETTE INTERRUPT CODE
                       C
0104                   C                ORG     4*041H
0104 ????????          C @HF_TBL_VEC    DD      ?              ; POINTER TO FIRST DISK PARAMETER TABLE
                       C
0118                   C                ORG     4*046H
0118 ????????          C @HF1_TBL_VEC   DD      ?              ; POINTER TO SECOND DISK PARAMETER TABLE
                       C
01C0                   C                ORG     4*070H
01C0 ????????          C @SLAVE_INT_PTR DD      ?              ; POINTER TO SLAVE INTERRUPT HANDLER
                       C
01D8                   C                ORG     4*076H
01D8 ????????          C @HDISK_INT     DD      ?              ; POINTER TO FIXED DISK INTERRUPT CODE
                       C
0400                   C        ORG     400H
0400                   C DATA_AREA      LABEL   BYTE           ; ABSOLUTE LOCATION OF DATA SEGMENT
0400                   C DATA_WORD      LABEL   WORD
                       C
0500                   C                ORG     0500H
0500                   C @MFG_TEST_RTN  LABEL   FAR            ; LOAD LOCATION FOR MANUFACTURING TESTS
                       C
7C00                   C                ORG     7C00H
7C00                   C @BOOT_LOCN     LABEL   FAR            ; BOOT STRAP CODE LOAD LOCATION
                       C
7C00                   C ABS0           ENDS
```

SECTION 5

```
                        C   PAGE
                        C   ;--------------------------------------------
                        C   ;       ROM BIOS DATA AREAS                  :
                        C   ;--------------------------------------------
                        C
0000                    C   DATA            SEGMENT AT 40H          ; ADDRESS= 0040:0000
0000                    C   DATA40          LABEL   BYTE
0000 ????               C   @RS232_BASE     DW      ?               ; BASE ADDRESSES OF RS232 ADAPTERS
0002 ????               C                   DW      ?               ;  SECOND LOGICAL RS232 ADAPTER
0004 ????               C                   DW      ?               ;   RESERVED
0006 ????               C                   DW      ?               ;   RESERVED
0008 ????               C   @PRINTER_BASE   DW      ?               ; BASE ADDRESSES OF PRINTER ADAPTERS
000A ????               C                   DW      ?               ;  SECOND LOGICAL PRINTER ADAPTER
000C ????               C                   DW      ?               ;  THIRD LOGICAL PRINTER ADAPTER
000E ????               C                   DW      ?               ;   RESERVED
0010 ????               C   @EQUIP_FLAG     DW      ?               ; INSTALLED HARDWARE FLAGS
0012 ??                 C   @MFG_TST        DB      ?               ; INITIALIZATION FLAGS
0013 ????               C   @MEMORY_SIZE    DW      ?               ; BASE MEMORY SIZE IN K BYTES  (X 1024)
0015 ??                 C   @MFG_ERR_FLAG   DB      ?               ; SCRATCHPAD FOR MANUFACTURING
0016 ??                 C                   DB      ?               ;  ERROR CODES
                        C
                        C   ;--------------------------------------------
                        C   ;       KEYBOARD DATA AREAS                  :
                        C   ;--------------------------------------------
                        C
0017 ??                 C   @KB_FLAG        DB      ?               ; KEYBOARD SHIFT STATE AND STATUS FLAGS
0018 ??                 C   @KB_FLAG_1      DB      ?               ; SECOND BYTE OF KEYBOARD STATUS
0019 ??                 C   @ALT_INPUT      DB      ?               ; STORAGE FOR ALTERNATE KEY PAD ENTRY
001A ????               C   @BUFFER_HEAD    DW      ?               ; POINTER TO HEAD OF KEYBOARD BUFFER
001C ????               C   @BUFFER_TAIL    DW      ?               ; POINTER TO TAIL OF KEYBOARD BUFFER
                        C
                        C   ;------     HEAD = TAIL    INDICATES THAT THE BUFFER IS EMPTY
                        C
001E    10 [            C   @KB_BUFFER      DW      16 DUP(?)       ; ROOM FOR 15 SCAN CODE ENTRIES
              ????      C
                    ]   C
                        C
                        C   ;--------------------------------------------
                        C   ;       DISKETTE DATA AREAS                  :
                        C   ;--------------------------------------------
                        C
003E ??                 C   @SEEK_STATUS    DB      ?               ; DRIVE RECALIBRATION STATUS
                        C                                           ; BIT 3-0 = DRIVE 3-0 RECALIBRATION
                        C                                           ; BEFORE NEXT SEEK IF BIT IS = 0
003F ??                 C   @MOTOR_STATUS   DB      ?               ; MOTOR STATUS
                        C                                           ; BIT 3-0 = DRIVE 3-0 CURRENTLY RUNNING
                        C                                           ; BIT 7 = CURRENT OPERATION IS A WRITE
0040 ??                 C   @MOTOR_COUNT    DB      ?               ; TIME OUT COUNTER FOR MOTOR(S) TURN OFF
0041 ??                 C   @DSKETTE_STATUS DB      ?               ; RETURN CODE STATUS BYTE
                        C                                           ; CMD_BLOCK  IN STACK FOR DISK OPERATION
0042    07 [            C   @NEC_STATUS     DB      7 DUP(?)        ; STATUS BYTES FROM DISKETTE OPERATION
              ??        C
                    ]   C
                        C
                        C
                        C   ;--------------------------------------------
                        C   ;       VIDEO DISPLAY DATA AREA              :
                        C   ;--------------------------------------------
                        C
0049 ??                 C   @CRT_MODE       DB      ?               ; CURRENT DISPLAY MODE (TYPE)
004A ????               C   @CRT_COLS       DW      ?               ; NUMBER OF COLUMNS ON SCREEN
004C ????               C   @CRT_LEN        DW      ?               ; LENGTH OF REGEN BUFFER IN BYTES
004E ????               C   @CRT_START      DW      ?               ; STARTING ADDRESS IN REGEN BUFFER
0050    08 [            C   @CURSOR_POSN    DW      8 DUP(?)        ; CURSOR FOR EACH OF UP TO 8 PAGES
              ????      C
                    ]   C
                        C
0060 ????               C   @CURSOR_MODE    DW      ?               ; CURRENT CURSOR MODE SETTING
0062 ??                 C   @ACTIVE_PAGE    DB      ?               ; CURRENT PAGE BEING DISPLAYED
0063 ????               C   @ADDR_6845      DW      ?               ; BASE ADDRESS FOR ACTIVE DISPLAY CARD
0065 ??                 C   @CRT_MODE_SET   DB      ?               ; CURRENT SETTING OF THE 3X8 REGISTER
0066 ??                 C   @CRT_PALETTE    DB      ?               ; CURRENT PALETTE SETTING - COLOR CARD
                        C
                        C   ;--------------------------------------------
                        C   ;       POST AND BIOS WORK DATA AREA         :
                        C   ;--------------------------------------------
                        C
                        C                                           ; STACK SAVE, ETC.
0067 ????               C   @IO_ROM_INIT    DW      ?               ; POINTER TO ROM INITIALIZATION ROUTINE
0069 ????               C   @IO_ROM_SEG     DW      ?               ; POINTER TO I/O ROM SEGMENT
006B ??                 C   @INTR_FLAG      DB      ?               ; FLAG INDICATING AN INTERRUPT HAPPENED
                        C
                        C   ;--------------------------------------------
                        C   ;       TIMER DATA AREA                      :
                        C   ;--------------------------------------------
                        C
006C ????               C   @TIMER_LOW      DW      ?               ; LOW WORD OF TIMER COUNT
006E ????               C   @TIMER_HIGH     DW      ?               ; HIGH WORD OF TIMER COUNT
0070 ??                 C   @TIMER_OFL      DB      ?               ; TIMER HAS ROLLED OVER SINCE LAST READ
                        C
                        C   ;--------------------------------------------
                        C   ;       SYSTEM DATA AREA                     :
                        C   ;--------------------------------------------
                        C
0071 ??                 C   @BIOS_BREAK     DB      ?               ; BIT 7=1 IF BREAK KEY HAS BEEN PRESSED
0072 ????               C   @RESET_FLAG     DW      ?               ; WORD=1234H IF KEYBOARD RESET UNDERWAY
                        C
                        C   ;--------------------------------------------
                        C   ;       FIXED DISK DATA AREAS                :
                        C   ;--------------------------------------------
                        C
0074 ??                 C   @DISK_STATUS1   DB      ?               ; FIXED DISK STATUS
0075 ??                 C   @HF_NUM         DB      ?               ; COUNT OF FIXED DISK DRIVES
0076 ??                 C   @CONTROL_BYTE   DB      ?               ; HEAD CONTROL BYTE
0077 ??                 C   @PORT_OFF       DB      ?               ;  RESERVED (PORT OFFSET)
```

```
                       C  PAGE
                       C  ;----------------------------------------
                       C  ;        TIME-OUT VARIABLES             :
                       C  ;----------------------------------------
                       C
0078 ??                C  @PRINT_TIM_OUT  DB     ?          ; TIME OUT COUNTERS FOR PRINTER RESPONSE
0079 ??                C                  DB     ?          ;  SECOND LOGICAL PRINTER ADAPTER
007A ??                C                  DB     ?          ;  THIRD LOGICAL PRINTER ADAPTER
007B ??                C                  DB     ?          ;   RESERVED
007C ??                C  @RS232_TIM_OUT  DB     ?          ; TIME OUT COUNTERS FOR RS232 RESPONSE
007D ??                C                  DB     ?          ;  SECOND LOGICAL RS232 ADAPTER
007E ??                C                  DB     ?          ;   RESERVED
007F ??                C                  DB     ?          ;   RESERVED
                       C
                       C  ;----------------------------------------
                       C  ;    ADDITIONAL KEYBOARD DATA AREA      :
                       C  ;----------------------------------------
                       C
                       C                                    ; BUFFER LOCATION WITHIN SEGMENT 40H
0080 ????              C  @BUFFER_START   DW     ?          ; OFFSET OF KEYBOARD BUFFER START
0082 ????              C  @BUFFER_END     DW     ?          ; OFFSET OF END OF BUFFER
                       C
                       C  ;----------------------------------------
                       C  ;       EGA/PGA DISPLAY WORK AREA       :
                       C  ;----------------------------------------
                       C
0084 ??                C  @ROWS           DB     ?          ; ROWS ON THE ACTIVE SCREEN (LESS 1)
0085 ????              C  @POINTS         DW     ?          ; BYTES PER CHARACTER
0087 ??                C  @INFO           DB     ?          ; MODE OPTIONS
0088 ??                C  @INFO_3         DB     ?          ; FEATURE BIT SWITCHES
0089 ??                C                  DB     ?          ; RESERVED FOR DISPLAY ADAPTERS
008A ??                C                  DB     ?          ; RESERVED FOR DISPLAY ADAPTERS
                       C
                       C  ;----------------------------------------
                       C  ;        ADDITIONAL MEDIA DATA          :
                       C  ;----------------------------------------
                       C
008B ??                C  @LASTRATE       DB     ?          ; LAST DISKETTE DATA RATE SELECTED
008C ??                C  @HF_STATUS      DB     ?          ; STATUS REGISTER
008D ??                C  @HF_ERROR       DB     ?          ; ERROR REGISTER
008E ??                C  @HF_INT_FLAG    DB     ?          ; FIXED DISK INTERRUPT FLAG
008F ??                C  @HF_CNTRL       DB     ?          ; BIT 0-> PC-1/DUAL FDC ADAPTER CARD
0090 ??                C  @DSK_STATE      DB     ?          ; DRIVE 0 MEDIA STATE
0091 ??                C                  DB     ?          ; DRIVE 1 MEDIA STATE
0092 ??                C                  DB     ?          ; DRIVE 0 OPERATION START STATE
0093 ??                C                  DB     ?          ; DRIVE 1 OPERATION START STATE
0094 ??                C  @DSK_TRK        DB     ?          ; DRIVE 0 PRESENT CYLINDER
0095 ??                C                  DB     ?          ; DRIVE 1 PRESENT CYLINDER
                       C
                       C  ;----------------------------------------
                       C  ;      ADDITIONAL KEYBOARD FLAGS        :
                       C  ;----------------------------------------
                       C
0096 ??                C  @KB_FLAG_3      DB     ?          ; KEYBOARD MODE STATE AND TYPE FLAGS
0097 ??                C  @KB_FLAG_2      DB     ?          ; KEYBOARD LED FLAGS
                       C
                       C  IFE SYSTEM
                       C  ;----------------------------------------
                       C  ;      REAL TIME CLOCK DATA AREA        :
                       C  ;----------------------------------------
                       C
0098 ????              C  @USER_FLAG      DW     ?          ; OFFSET ADDRESS OF USERS WAIT FLAG
009A ????              C  @USER_FLAG_SEG  DW     ?          ; SEGMENT ADDRESS OF USER WAIT FLAG
009C ????              C  @RTC_LOW        DW     ?          ; LOW WORD OF USER WAIT FLAG
009E ????              C  @RTC_HIGH       DW     ?          ; HIGH WORD OF USER WAIT FLAG
00A0 ??                C  @RTC_WAIT_FLAG  DB     ?          ; WAIT ACTIVE FLAG (01=BUSY, 80=POSTED)
                       C  ENDIF
                       C                                    ;                (00=POST ACKNOWLEDGED)
                       C  ;----------------------------------------
                       C  ;      AREA FOR NETWORK ADAPTER         :
                       C  ;----------------------------------------
                       C
00A1    07 [           C  @NET            DB     7 DUP(?)   ; RESERVED FOR NETWORK ADAPTERS
              ??       C
                  ]    C
                       C
                       C  ;----------------------------------------
                       C  ;       EGA/PGA PALETTE POINTER         :
                       C  ;----------------------------------------
                       C
00A8 ????????          C  @SAVE_PTR       DD     ?          ; POINTER TO EGA PARAMETER CONTROL BLOCK
                       C
                       C  ;----------------------------------------
                       C  ;        TIMER DATA                     :
                       C  ;----------------------------------------
                       C
00CE                   C                  ORG    0CEH
00CE ????              C  @DAY_COUNT      DW     ?          ; COUNT OF DAYS FROM 1-1-80
                       C
                       C                                    ; RESERVED
                       C  ;----------------------------------------
                       C  ;      DATA AREA - PRINT SCREEN         :
                       C  ;----------------------------------------
                       C
0100                   C                  ORG    100H       ; ADDRESS= 0040:0100   (REF 0050:0000)
                       C
0100 ??                C  @STATUS_BYTE    DB     ?          ; PRINT SCREEN STATUS BYTE
                       C                                    ;  00=READY/OK,  01=BUSY,  FF=ERROR
                       C
0101                   C  DATA            ENDS              ; END OF BIOS DATA SEGMENT

                          .LIST
```

```
                              PAGE
0000                          CODE    SEGMENT WORD PUBLIC

                                      PUBLIC  HEADER

                                      ASSUME  CS:CODE,DS:NOTHING,ES:NOTHING,SS:NOTHING

0000                          HEADER  PROC    NEAR

= 0000                        BEGIN   EQU     $
0000 36 32 58 30 38 35                DB      '62X0854 COPR. IBM CORP. 1981,1986 '          ;COPYRIGHT NOTICE
     34 20 43 4F 50 52
     2E 20 49 42 4D 20
     43 4F 52 50 2E 20
     31 39 38 31 2C 31
     39 38 36 20
                                      EVEN                                                  ;EVEN BOUNDARY
0022 20 20 20 20 20 20                DB      '                                 '           ;PAD
     20 20 20 20 20 20
     20 20 20 20 20 20
     20 20 20 20 20
0039 20 20 20 20 20 20                DB      '                                 '           ;PAD
     20 20 20 20 20 20
     20 20 20 20 20 20
     20 20 20 20 20

0050                          HEADER  ENDP
0050                          CODE    ENDS
                                      END
```

```
PAGE 118,121
TITLE DSKETTE -- 01/10/86  DISKETTE ADAPTER BIOS
.LIST
;-- INT 13 -----------------------------------------------------------------
; DISKETTE I/O
;       THIS INTERFACE PROVIDES DISK ACCESS TO THE 5.25 INCH 360 KB,
;       1.2 MB, AND 720 KB 80 TRACK DISKETTE DRIVES.
; INPUT
;       (AH)=0  RESET DISKETTE SYSTEM
;               HARD RESET TO NEC, PREPARE COMMAND, RECALIBRATE REQUIRED
;               ON ALL DRIVES
;--------------------------------------------------------------------------
;       (AH)=1  READ THE STATUS OF THE SYSTEM INTO (AH)
;               @DISKETTE_STATUS FROM LAST OPERATION 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
;               720K       720K              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
;               720K       720K              1-9
;       (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
;               720K       720K                 9
;
;       (ES:BX) - ADDRESS OF BUFFER (NOT 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
;               (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, IF THERE EXISTS MORE THAN
;               ONE SUPPORTED MEDIA FORMAT TYPE WITHIN THE DRIVE IN QUESTION,
;               THEN "SET DASD TYPE" (INT 13H, AH = 17H) OR "SET MEDIA TYPE"
;               (INT 13H, AH = 18H) MUST BE CALLED TO SET THE DISKETTE TYPE
;               THAT IS TO BE FORMATED.  IF "SET DASD TYPE" OR "SET MEDIA TYPE"
;               IS NOT CALLED, THE FORMAT ROUTINE WILL ASSUME THE MEDIA FORMAT
;               TO BE THE MAXIMUM CAPACITY OF THE DRIVE.
;
;               THESE PARAMETERS OF DISK_BASE MUST BE CHANGED IN ORDER TO
;               FORMAT THE FOLLOWING MEDIAS:
;               --------------------------------------------
;               : MEDIA  :      DRIVE       : PARM 1 : PARM 2 :
;               --------------------------------------------
;               : 320K   : 320K/360K/1.2M :  50H   :   8    :
;               : 360K   : 320K/360K/1.2M :  50H   :   9    :
;               : 1.2M   : 1.2M           :  54H   :  15    :
;               : 720K   : 720K           :  50H   :   9    :
;               --------------------------------------------
;               NOTES: - PARM 1 = GAP LENGTH FOR FORMAT
;                      - PARM 2 = EOT (LAST SECTOR ON TRACK)
;                      - DISK_BASE IS POINTED TO BY DISK POINTER LOCATED
;                        AT ABSOLUTE ADDRESS 0:78H.
;                      - WHEN FORMAT OPERATIONS ARE COMPLETE, THE PARAMETERS
;                        SHOULD BE RESTORED TO THEIR RESPECTIVE INITIAL VALUES.
;
;--------------------------------------------------------------------------
;       (AH)=8  READ DRIVE PARAMETERS
;       REGISTERS
;         INPUT
;           (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)
;         OUTPUT
;           (ES:DI) POINTS TO DRIVE PARAMETERS TABLE
;           (CH) - LOW ORDER 8 OF 10 BITS MAXIMUM NUMBER OF TRACKS
;           (CL) - BITS 7 & 6 - HIGH ORDER TWO BITS OF MAXIMUM TRACKS
;                - BITS 5 THRU 0 - MAXIMUM SECTORS PER TRACK
;           (DH) - MAXIMUM HEAD NUMBER
;           (DL) - NUMBER OF DISKETTE DRIVES INSTALLED
;           (BH) - 0
;           (BL) - BITS 7 THRU 4 - 0
;                  BITS 3 THRU 0 - VALID DRIVE TYPE VALUE IN CMOS
;           (AX) - 0
;         UNDER THE FOLLOWING CIRCUMSTANCES:
;          (1) THE DRIVE NUMBER IS INVALID,
;          (2) THE DRIVE TYPE IS UNKNOWN AND CMOS IS NOT PRESENT,
;          (3) THE DRIVE TYPE IS UNKNOWN AND CMOS IS BAD,
;          (4) OR THE DRIVE TYPE IS UNKNOWN AND THE CMOS DRIVE TYPE IS INVALID
;          THEN ES,AX,BX,CX,DH,DI=0 ; DL=NUMBER OF DRIVES.
;          IF NO DRIVES ARE PRESENT THEN: ES,AX,BX,CX,DX,DI=0.
;          @DISKETTE_STATUS = 0 AND CY IS RESET.
;--------------------------------------------------------------------------
;       (AH)=15 READ DASD TYPE
;       OUTPUT 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 - RESERVED
;       (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)
;
```

```
;--------------------------------------------------------------------------------
;       (AH)=16 DISK CHANGE LINE STATUS
;       OUTPUT 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
;       INPUT REGISTERS
;       (AL) -    00 - NOT USED
;                 01 - DISKETTE 320/360K IN 360K DRIVE
;                 02 - DISKETTE 360K IN 1.2M DRIVE
;                 03 - DISKETTE 1.2M IN 1.2M DRIVE
;                 04 - DISKETTE 720K IN 720K DRIVE
;       (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED;
;              DO NOT USE WHEN DISKETTE ATTACH CARD USED)
;--------------------------------------------------------------------------------
;       (AH)=18 SET MEDIA TYPE FOR FORMAT
;       INPUT REGISTERS
;       (CH) - LOW ORDER 8 OF 10 BITS MAXIMUM NUMBER OF TRACKS
;       (CL) - BITS 7 & 6 - HIGH ORDER TWO BITS OF MAXIMUM TRACKS
;            - BITS 5 THRU 0 - MAXIMUM SECTORS PER TRACK
;       (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)
;       OUTPUT REGISTERS
;       (ES:DI) - POINTER TO DRIVE PARAMETERS TABLE FOR THIS MEDIA TYPE,
;                 UNCHANGED IF (AH) IS NON-ZERO
;       (AH) - 00H, CY = 0, TRACK AND SECTORS/TRACK COMBINATION IS SUPPORTED
;            - 01H, CY = 1, FUNCTION IS NOT AVAILABLE
;            - 0CH, CY = 1, TRACK AND SECTORS/TRACK COMBINATION IS NOT SUPPORTED
;--------------------------------------------------------------------------------
;       DISK CHANGE STATUS IS ONLY CHECKED WHEN A MEDIA SPECIFIED IS OTHER
;       THAN 360 KB DRIVE.  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 FOR ALL FUNCTIONS
;       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,CX 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.
;--------------------------------------------------------------------------------
.LIST
; DISKETTE STATE MACHINE - ABSOLUTE ADDRESS 40:90 (DRIVE A) & 91 (DRIVE B)
.LIST
;   -----------------------------------------------------------------
;   |       |       |       |       |       |       |       |       |
;   |   7   |   6   |   5   |   4   |   3   |   2   |   1   |   0   |
;   |       |       |       |       |       |       |       |       |
;   -----------------------------------------------------------------
;       |       |       |       |       |       |       |       |
;       |       |       |       |       |       -----------------
;       |       |       |       |       |               |
;       |       |       |       |    RESERVED           |
;       |       |       |       |                 PRESENT STATE
;       |       |       |       |          000: 360K IN 360K DRIVE UNESTABLISHED
;       |       |       |       |          001: 360K IN 1.2M DRIVE UNESTABLISHED
;       |       |       |       |          010: 1.2M IN 1.2M DRIVE UNESTABLISHED
;       |       |       |       |          011: 360K IN 360K DRIVE ESTABLISHED
;       |       |       |       |          100: 360K IN 1.2M DRIVE ESTABLISHED
;       |       |       |       |          101: 1.2M IN 1.2M DRIVE ESTABLISHED
;       |       |       |       |          110: RESERVED
;       |       |       |       |          111: NONE OF THE ABOVE
;       |       |       |       |
;       |       |       |       ------> MEDIA/DRIVE ESTABLISHED
;       |       |       |
;       |       |       --------------> DOUBLE STEPPING REQUIRED (360K IN 1.2M
;       |       |                          DRIVE)
;       |       |
;       ------------------------------> DATA TRANSFER RATE FOR THIS DRIVE:
;
;                                               00: 500 KBS
;                                               01: 300 KBS
;                                               10: 250 KBS
;                                               11: RESERVED
;

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

```
                                PAGE

                                MD_STRUC        STRUC
 0000 ??                        MD_SPEC1        DB      ?       ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
 0001 ??                        MD_SPEC2        DB      ?       ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
 0002 ??                        MD_OFF_TIM      DB      ?       ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
 0003 ??                        MD_BYT_SEC      DB      ?       ; 512 BYTES/SECTOR
 0004 ??                        MD_SEC_TRK      DB      ?       ; EOT ( LAST SECTOR ON TRACK)
 0005 ??                        MD_GAP          DB      ?       ; GAP LENGTH
 0006 ??                        MD_DTL          DB      ?       ; DTL
 0007 ??                        MD_GAP3         DB      ?       ; GAP LENGTH FOR FORMAT
 0008 ??                        MD_FIL_BYT      DB      ?       ; FILL BYTE FOR FORMAT
 0009 ??                        MD_HD_TIM       DB      ?       ; HEAD SETTLE TIME (MILLISECONDS)
 000A ??                        MD_STR_TIM      DB      ?       ; MOTOR START TIME (1/8 SECONDS)
 000B ??                        MD_MAX_TRK      DB      ?       ; MAX. TRACK NUMBER
 000C ??                        MD_RATE         DB      ?       ; DATA TRANSFER RATE
 000D                           MD_STRUC        ENDS

 = 007F                         BIT7OFF         EQU     7FH
 = 0080                         BIT7ON          EQU     80H

                                        PUBLIC  DISK_INT_1
                                        PUBLIC  DSKETTE_SETUP
                                        PUBLIC  DISKETTE_IO_1
                                        PUBLIC  NEC_OUTPUT
                                        PUBLIC  RESULTS
                                        PUBLIC  SEEK

                                        EXTRN   DDS:NEAR
                                        EXTRN   DISK_BASE:NEAR
                                        EXTRN   WAITF:NEAR
                                        EXTRN   MD_TBL1:NEAR
                                        EXTRN   MD_TBL2:NEAR
                                        EXTRN   MD_TBL3:NEAR
                                        EXTRN   MD_TBL4:NEAR
                                        EXTRN   MD_TBL5:NEAR
                                        EXTRN   MD_TBL6:NEAR

 0000                           CODE    SEGMENT BYTE PUBLIC

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

                                ;---------------------------------------------------------------
                                ;       DRIVE TYPE TABLE                                       :
                                ;---------------------------------------------------------------
 0000                           DR_TYPE         LABEL   BYTE
 0000 01                                        DB      01              ; DRIVE TYPE, MEDIA TABLE
 0001 0000 E                                    DW      OFFSET MD_TBL1
 0003 82                                        DB      02+BIT7ON
 0004 0000 E                                    DW      OFFSET MD_TBL2
 0006 02                                        DB      02
 0007 0000 E                                    DW      OFFSET MD_TBL3
 0009 03                                        DB      03
 000A 0000 E                                    DW      OFFSET MD_TBL4
 000C 84                                        DB      04+BIT7ON
 000D 0000 E                                    DW      OFFSET MD_TBL5
 000F 04                                        DB      04
 0010 0000 E                                    DW      OFFSET MD_TBL6
 = 0012                         DR_TYPE_E       =$                      ; END OF TABLE
 = 0006                         DR_CNT          EQU     (DR_TYPE_E-DR_TYPE)/3   ; NUMBER OF DRIVE TYPES

 0012                           DISKETTE_IO_1   PROC    FAR             ;>>> ENTRY POINT FOR ORG 0EC59H
 0012 FB                                STI                             ; INTERRUPTS BACK ON
 0013 55                                PUSH    BP                      ; USER REGISTER
 0014 57                                PUSH    DI                      ; USER REGISTER
 0015 52                                PUSH    DX                      ; HEAD #, DRIVE # OR USER REGISTER
 0016 53                                PUSH    BX                      ; BUFFER OFFSET PARAMETER OR REGISTER
 0017 51                                PUSH    CX                      ; TRACK #-SECTOR # OR USER REGISTER
 0018 8B EC                             MOV     BP,SP                   ; BP     => PARAMETER LIST DEP. ON AH
                                                                        ; [BP]    = SECTOR #
                                                                        ; [BP+1]  = TRACK #
                                                                        ; [BP+2]  = BUFFER OFFSET
                                                                        ; FOR RETURN OF DRIVE PARAMETERS:
                                                                        ; CL/[BP] = BITS 7&6 HI BITS OF MAX CYL
                                                                        ;           BITS 0-5 MAX SECTORS/TRACK
                                                                        ; CH/[BP+1] = LOW 8 BITS OF MAX CYL.
                                                                        ; BL/[BP+2] = BITS 7-4 = 0
                                                                        ;             BITS 3-0 = VALID CMOS TYPE
                                                                        ; BH/[BP+3] = 0
                                                                        ; DL/[BP+4] = # DRIVES INSTALLED
                                                                        ; DH/[BP+5] = MAX HEAD #
                                                                        ; DI/[BP+6] = OFFSET TO DISK BASE
 001A 1E                                PUSH    DS                      ; BUFFER SEGMENT PARM OR USER REGISTER
 001B 56                                PUSH    SI                      ; USER REGISTERS
 001C E8 0000 E                         CALL    DDS                     ; SEGMENT OF BIOS DATA AREA TO DS
 001F 80 FC 19                          CMP     AH,(FNC_TAE-FNC_TAB)/2  ; CHECK FOR > LARGEST FUNCTION
 0022 72 02                             JB      OK_FUNC                 ; FUNCTION OK
 0024 B4 14                             MOV     AH,14H                  ; REPLACE WITH KNOWN INVALID FUNCTION
 0026                           OK_FUNC:
 0026 80 FC 01                          CMP     AH,1                    ; RESET OR STATUS ?
 0029 76 0C                             JBE     OK_DRV                  ; IF RESET OR STATUS DRIVE ALWAYS OK
 002B 80 FC 08                          CMP     AH,8                    ; READ DRIVE PARMS ?
 002E 74 07                             JZ      OK_DRV                  ; IF SO DRIVE CHECKED LATER
 0030 80 FA 03                          CMP     DL,3                    ; DRIVES 0,1,2 AND 3 OK
 0033 76 02                             JBE     OK_DRV                  ; IF 0 OR 1 THEN JUMP
 0035 B4 14                             MOV     AH,14H                  ; REPLACE WITH KNOWN INVALID FUNCTION
 0037                           OK_DRV:
 0037 8A CC                             MOV     CL,AH                   ; CL = FUNCTION
 0039 32 ED                             XOR     CH,CH                   ; CX = FUNCTION
 003B D0 E1                             SHL     CL,1                    ; FUNCTION TIMES 2
 003D BB 0060 R                         MOV     BX,OFFSET FNC_TAB       ; LOAD START OF FUNCTION TABLE
 0040 03 D9                             ADD     BX,CX                   ; ADD OFFSET INTO TABLE => ROUTINE
 0042 8A E6                             MOV     AH,DH                   ; AX = HEAD #,# OF SECTORS OR DASD TYPE
 0044 32 F6                             XOR     DH,DH                   ; DX = DRIVE #
 0046 8B F0                             MOV     SI,AX                   ; SI = HEAD #,# OF SECTORS OR DASD TYPE
 0048 8B FA                             MOV     DI,DX                   ; DI = DRIVE #
 004A 8A 26 0041 R                      MOV     AH,@DSKETTE_STATUS      ; LOAD STATUS TO AH FOR STATUS FUNCTION
 004E C6 06 0041 R 00                   MOV     @DSKETTE_STATUS,0       ; INITIALIZE FOR ALL OTHERS

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

SECTION 5

```
                                ;               SI-HI  : HEAD #
                                ;               SI-LOW : # OF SECTORS OR DASD TYPE FOR FORMAT
                                ;               ES     : BUFFER SEGMENT
                                ;               [BP]   : SECTOR #
                                ;               [BP+1] : TRACK #
                                ;               [BP+2] : BUFFER OFFSET
                                ;
                                ;       ACROSS CALLS TO SUBROUTINES THE CARRY FLAG (CY=1), WHERE INDICATED IN
                                ;       SUBROUTINE PROLOGUES, REPRESENTS AN EXCEPTION RETURN (NORMALLY AN ERROR
                                ;       CONDITION). IN MOST CASES, WHEN CY = 1, @DSKETTE_STATUS CONTAINS THE
                                ;       SPECIFIC ERROR CODE.
                                                                ; (AH) = @DSKETTE_STATUS
 0053 2E: FF 17                         CALL    WORD PTR CS:[BX]        ; CALL THE REQUESTED FUNCTION

 0056 5E                                POP     SI                      ; RESTORE ALL REGISTERS
 0057 1F                                POP     DS
 0058 59                                POP     CX
 0059 5B                                POP     BX
 005A 5A                                POP     DX
 005B 5F                                POP     DI
 005C 5D                                POP     BP
 005D CA 0002                           RET     2                       ; THROW AWAY SAVED FLAGS

                                ;----------------------------------------------------------------
 0060 0092 R                    FNC_TAB DW      DISK_RESET              ; AH = 00; RESET
 0062 00EA R                            DW      DISK_STATUS             ; AH = 01; STATUS
 0064 00F6 R                            DW      DISK_READ               ; AH = 02; READ
 0066 0102 R                            DW      DISK_WRITE              ; AH = 03; WRITE
 0068 010E R                            DW      DISK_VERF               ; AH = 04; VERIFY
 006A 011A R                            DW      DISK_FORMAT             ; AH = 05; FORMAT
 006C 017D R                            DW      FNC_ERR                 ; AH = 06; INVALID
 006E 017D R                            DW      FNC_ERR                 ; AH = 07; INVALID
 0070 0187 R                            DW      DISK_PARMS              ; AH = 08; READ DRIVE PARAMETERS
 0072 017D R                            DW      FNC_ERR                 ; AH = 09; INVALID
 0074 017D R                            DW      FNC_ERR                 ; AH = 0A; INVALID
 0076 017D R                            DW      FNC_ERR                 ; AH = 0B; INVALID
 0078 017D R                            DW      FNC_ERR                 ; AH = 0C; INVALID
 007A 017D R                            DW      FNC_ERR                 ; AH = 0D; INVALID
 007C 017D R                            DW      FNC_ERR                 ; AH = 0E; INVALID
 007E 017D R                            DW      FNC_ERR                 ; AH = 0F; INVALID
 0080 017D R                            DW      FNC_ERR                 ; AH = 10; INVALID
 0082 017D R                            DW      FNC_ERR                 ; AH = 11; INVALID
 0084 017D R                            DW      FNC_ERR                 ; AH = 12; INVALID
 0086 017D R                            DW      FNC_ERR                 ; AH = 13; INVALID
 0088 017D R                            DW      FNC_ERR                 ; AH = 14; INVALID
 008A 027C R                            DW      DISK_TYPE               ; AH = 15; READ DASD TYPE
 008C 02B0 R                            DW      DISK_CHANGE             ; AH = 16; CHANGE STATUS
 008E 02E5 R                            DW      FORMAT_SET              ; AH = 17; SET DASD TYPE
 0090 034D R                            DW      SET_MEDIA               ; AH = 18; SET MEDIA TYPE
 = 0092                         FNC_TAE EQU     $                       ; END
 0092                           DISKETTE_IO_1   ENDP

                                ;----------------------------------------------------------------
                                ; DISK_RESET                                                    :
                                ;       RESET THE DISKETTE SYSTEM.                              :
                                ;                                                               :
                                ; ON EXIT:      @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION :
                                ;----------------------------------------------------------------
 0092                           DISK_RESET      PROC    NEAR
 0092 BA 03F2                           MOV     DX,03F2H                ; ADAPTER CONTROL PORT
 0095 FA                                CLI                             ; NO INTERRUPTS
 0096 A0 003F R                         MOV     AL,@MOTOR_STATUS        ; GET DIGITAL OUTPUT REGISTER REFLECTION
 0099 24 3F                             AND     AL,00111111B            ; KEEP SELECTED AND MOTOR ON BITS
 009B D0 C0                             ROL     AL,1                    ; MOTOR VALUE TO HIGH NIBBLE
 009D D0 C0                             ROL     AL,1                    ; DRIVE SELECT TO LOW NIBBLE
 009F D0 C0                             ROL     AL,1
 00A1 D0 C0                             ROL     AL,1
 00A3 0C 08                             OR      AL,00001000B            ; TURN ON INTERRUPT ENABLE
 00A5 EE                                OUT     DX,AL                   ; RESET THE ADAPTER
 00A6 C6 06 003E R 00                   MOV     @SEEK_STATUS,0          ; SET RECALIBRATE REQUIRED ON ALL DRIVES
 00AB EB 00                             JMP     $+2                     ; WAIT FOR I/O
 00AD 0C 04                             OR      AL,00000100B            ; TURN OFF RESET BIT
 00AF EE                                OUT     DX,AL                   ; RESET THE ADAPTER
 00B0 FB                                STI                             ; ENABLE THE INTERRUPTS
 00B1 E8 0ABA R                         CALL    WAIT_INT                ; WAIT FOR THE INTERRUPT
 00B4 72 2D                             JC      DR_ERR                  ; IF ERROR, RETURN IT
 00B6 B9 00C0                           MOV     CX,11000000B            ; CL = EXPECTED @NEC_STATUS

 00B9                           NXT_DRV:
 00B9 51                                PUSH    CX                      ; SAVE FOR CALL
 00BA B8 00E2 R                         MOV     AX,OFFSET DR_POP_ERR    ; LOAD NEC_OUTPUT ERROR ADDRESS
 00BD 50                                PUSH    AX                      ; "
 00BE B4 08                             MOV     AH,08H                  ; SENSE INTERRUPT STATUS COMMAND
 00C0 E8 09F0 R                         CALL    NEC_OUTPUT
 00C3 58                                POP     AX                      ; THROW AWAY ERROR RETURN
 00C4 E8 0AE2 R                         CALL    RESULTS                 ; READ IN THE RESULTS
 00C7 59                                POP     CX                      ; RESTORE AFTER CALL
 00C8 72 19                             JC      DR_ERR                  ; ERROR RETURN
 00CA 3A 0E 0042 R                      CMP     CL,@NEC_STATUS          ; TEST FOR DRIVE READY TRANSITION
 00CE 75 13                             JNZ     DR_ERR                  ; EVERYTHING OK
 00D0 FE C1                             INC     CL                      ; NEXT EXPECTED @NEC_STATUS
 00D2 80 F9 C3                          CMP     CL,11000011B            ; ALL POSSIBLE DRIVES CLEARED
 00D5 76 E2                             JBE     NXT_DRV                 ; FALL THRU IF 11000100B OR >

                                ;-----  SEND SPECIFY COMMAND TO NEC

 00D7                           J7:
 00D7 E8 03D1 R                         CALL    SEND_SPEC
 00DA                           RESBAC:
 00DA E8 0832 R                         CALL    SETUP_END               ; VARIOUS CLEANUPS
 00DD 8B DE                             MOV     BX,SI                   ; GET SAVED AL TO BL
 00DF 8A C3                             MOV     AL,BL                   ; PUT BACK FOR RETURN
 00E1 C3                                RET

 00E2                           DR_POP_ERR:
 00E2 59                                POP     CX                      ; CLEAR STACK
 00E3                           DR_ERR:                                 ;
 00E3 80 0E 0041 R 20                   OR      @DSKETTE_STATUS,BAD_NEC ; SET ERROR CODE
 00E8 EB F0                             JMP     SHORT RESBAC            ; RETURN FROM RESET
 00EA                           DISK_RESET      ENDP

                                ;----------------------------------------------------------------
                                ; DISK_STATUS                                                   :
                                ;       DISKETTE STATUS.                                        :
                                ; ON ENTRY:     AH = STATUS OF PREVIOUS OPERATION               :
```

```
                                ;                                                               :
                                ; ON EXIT:      @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION :
                                ;----------------------------------------------------------------
 00EA                           DISK_STATUS     PROC    NEAR
 00EA 88 26 0041 R                      MOV     @DSKETTE_STATUS,AH      ; PUT BACK FOR SETUP_END
 00EE E8 0832 R                         CALL    SETUP_END               ; VARIOUS CLEANUPS
 00F1 8B DE                             MOV     BX,SI                   ; GET SAVED AL TO BL
 00F3 8A C4                             MOV     AL,AH                   ; STORE STATUS IN AL
 00F5 C3                                RET
 00F6                           DISK_STATUS     ENDP
                                ;----------------------------------------------------------------
                                ; DISK_READ                                                     :
                                ;       DISKETTE READ.                                          :
                                ; ON ENTRY:     DI      = DRIVE #                               :
                                ;               SI-HI   = HEAD #                                :
                                ;               SI-LOW  = # OF SECTORS                          :
                                ;               ES      = BUFFER SEGMENT                        :
                                ;               [BP]    = SECTOR #                              :
                                ;               [BP+1]  = TRACK #                               :
                                ;               [BP+2]  = BUFFER OFFSET                         :
                                ;                                                               :
                                ; ON EXIT:      @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION :
                                ;----------------------------------------------------------------
 00F6                           DISK_READ       PROC    NEAR
 00F6 80 26 003F R 7F                   AND     @MOTOR_STATUS,01111111B ; INDICATE A READ OPERATION
 00FB B8 E646                           MOV     AX,0E646H               ; AX = NEC COMMAND, DMA COMMAND
 00FE E8 04B3 R                         CALL    RD_WR_VF                ; COMMON READ/WRITE/VERIFY
 0101 C3                                RET
 0102                           DISK_READ       ENDP
                                ;----------------------------------------------------------------
                                ; DISK_WRITE                                                    :
                                ;       DISKETTE WRITE.                                         :
                                ; ON ENTRY:     DI      = DRIVE #                               :
                                ;               SI-HI   = HEAD #                                :
                                ;               SI-LOW  = # OF SECTORS                          :
                                ;               ES      = BUFFER SEGMENT                        :
                                ;               [BP]    = SECTOR #                              :
                                ;               [BP+1]  = TRACK #                               :
                                ;               [BP+2]  = BUFFER OFFSET                         :
                                ;                                                               :
                                ; ON EXIT:      @DSKETTE_STATUS, CY REFLECT STATUScOF OPERATION :
                                ;----------------------------------------------------------------
 0102                           DISK_WRITE      PROC    NEAR
 0102 B8 C54A                           MOV     AX,0C54AH               ; AX = NEC COMMAND, DMA COMMAND
 0105 80 0E 003F R 80                   OR      @MOTOR_STATUS,10000000B ; INDICATE WRITE OPERATION
 010A E8 04B3 R                         CALL    RD_WR_VF                ; COMMON READ/WRITE/VERIFY
 010D C3                                RET
 010E                           DISK_WRITE      ENDP
                                ;----------------------------------------------------------------
                                ; DISK_VERF                                                     :
                                ;       DISKETTE VERIFY.                                        :
                                ; ON ENTRY:     DI      = DRIVE #                               :
                                ;               SI-HI   = HEAD #                                :
                                ;               SI-LOW  = # OF SECTORS                          :
                                ;               ES      = BUFFER SEGMENT                        :
                                ;               [BP]    = SECTOR #                              :
                                ;               [BP+1]  = TRACK #                               :
                                ;               [BP+2]  = BUFFER OFFSET                         :
                                ;                                                               :
                                ; ON EXIT:      @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION :
                                ;----------------------------------------------------------------
 010E                           DISK_VERF       PROC    NEAR
 010E 80 26 003F R 7F                   AND     @MOTOR_STATUS,01111111B ; INDICATE A READ OPERATION
 0113 B8 E642                           MOV     AX,0E642H               ; AX = NEC COMMAND, DMA COMMAND
 0116 E8 04B3 R                         CALL    RD_WR_VF                ; COMMON READ/WRITE/VERIFY
 0119 C3                                RET
 011A                           DISK_VERF       ENDP
                                ;----------------------------------------------------------------
                                ; DISK_FORMAT                                                   :
                                ;       DISKETTE FORMAT.                                        :
                                ; ON ENTRY:     DI      = DRIVE #                               :
                                ;               SI-HI   = HEAD #                                :
                                ;               SI-LOW  = # OF SECTORS                          :
                                ;               ES      = BUFFER SEGMENT                        :
                                ;               [BP]    = SECTOR #                              :
                                ;               [BP+1]  = TRACK #                               :
                                ;               [BP+2]  = BUFFER OFFSET                         :
                                ;               @DISK_POINTER POINTS TO THE PARAMETER TABLE OF  :
                                ;                       THIS DRIVE                              :
                                ;                                                               :
                                ; ON EXIT:      @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION :
                                ;----------------------------------------------------------------
 011A                           DISK_FORMAT     PROC    NEAR
 011A E8 0404 R                         CALL    XLAT_NEW                ; TRANSLATE STATE TO PRESENT ARCH.
 011D E8 05A0 R                         CALL    FMT_INIT                ; ESTABLISH STATE IF UNESTABLISHED
 0120 80 0E 003F R 80                   OR      @MOTOR_STATUS,10000000B ; INDICATE WRITE OPERATION
 0125 F6 06 008F R 01                   TEST    @HF_CNTRL,DUAL          ; TEST CONTROLLER I.D.
 012A 74 05                             JZ      NO_CHG_CHECK
 012C E8 05F5 R                         CALL    MED_CHANGE              ; CHECK MEDIA CHANGE AND RESET IF SO
 012F 72 41                             JC      FM_DON                  ; MEDIA CHANGED, SKIP
 0131                           NO_CHG_CHECK:
 0131 E8 0658 R                         CALL    CHK_LASTRATE            ; ZF=1 ATTEMPT RATE IS SAME AS LAST RATE
 0134 74 06                             JZ      FM_WR                   ; YES, SKIP SPECIFY COMMAND
 0136 E8 03D1 R                         CALL    SEND_SPEC               ; SEND SPECIFY COMMAND TO NEC
 0139 E8 0637 R                         CALL    SEND_RATE               ; SEND DATA RATE TO CONTROLLER
 013C                           FM_WR:
 013C B0 4A                             MOV     AL,04AH                 ; WILL WRITE TO THE DISKETTE
 013E E8 0668 R                         CALL    DMA_SETUP               ; SET UP THE DMA
 0141 72 2F                             JC      FM_DON                  ; RETURN WITH ERROR
 0143 B4 4D                             MOV     AH,04DH                 ; ESTABLISH THE FORMAT COMMAND
 0145 E8 06CB R                         CALL    NEC_INIT                ; INITIALIZE THE NEC
 0148 72 28                             JC      FM_DON
 014A B8 0172 R                         MOV     AX,OFFSET FM_DON        ; LOAD ERROR ADDRESS
 014D 50                                PUSH    AX                      ; PUSH NEC_OUT ERROR RETURN
 014E B2 03                             MOV     DL,3                    ; BYTES/SECTOR VALUE TO NEC
 0150 E8 08FE R                         CALL    GET_PARM
 0153 E8 09F0 R                         CALL    NEC_OUTPUT
 0156 B2 04                             MOV     DL,4                    ; SECTORS/TRACK VALUE TO NEC
 0158 E8 08FE R                         CALL    GET_PARM
 015B E8 09F0 R                         CALL    NEC_OUTPUT
 015E B2 07                             MOV     DL,7                    ; GAP LENGTH VALUE TO NEC
 0160 E8 08FE R                         CALL    GET_PARM
 0163 E8 09F0 R                         CALL    NEC_OUTPUT
 0166 B2 08                             MOV     DL,8                    ; FILLER BYTE TO NEC
```

SECTION 5

```
0168 E8 08FE R                CALL   GET_PARM
016B E8 09F0 R                CALL   NEC_OUTPUT
016E 58                       POP    AX                        ; THROW AWAY ERROR
016F E8 0727 R                CALL   NEC_TERM                  ; TERMINATE, RECEIVE STATUS, ETC.
0172                   FM_DON:
0172 E8 0432 R                CALL   XLAT_OLD                  ; TRANSLATE STATE TO COMPATIBLE MODE
0175 E8 0832 R                CALL   SETUP_END                 ; VARIOUS CLEANUPS
0178 8B DE                    MOV    BX,SI                     ; GET SAVED AL TO BL
017A 8A C3                    MOV    AL,BL                     ; PUT BACK FOR RETURN
017C C3                       RET
017D                   DISK_FORMAT    ENDP
                       ;-------------------------------------------------------------
                       ; FNC_ERR                                                    :
                       ;         INVALID FUNCTION REQUESTED OR INVALID DRIVE;       :
                       ;         SET BAD COMMAND IN STATUS.                         :
                       ;                                                            :
                       ; ON EXIT:      @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION :
                       ;-------------------------------------------------------------
017D                   FNC_ERR PROC   NEAR                     ; INVALID FUNCTION REQUEST
017D 8B C6                    MOV    AX,SI                     ; RESTORE AL
017F B4 01                    MOV    AH,BAD_CMD                ; SET BAD COMMAND ERROR
0181 88 26 0041 R             MOV    @DSKETTE_STATUS,AH        ; STORE IN DATA AREA
0185 F9                       STC                              ; SET CARRY INDICATING ERROR
0186 C3                       RET
0187                   FNC_ERR ENDP
                       ;-------------------------------------------------------------
                       ; DISK_PARMS                                                 :
                       ;         READ DRIVE PARAMETERS.                             :
                       ; ON ENTRY:                                                  :
                       ;         DI = DRIVE #                                       :
                       ; ON EXIT:                                                   :
                       ;         CL/[BP]   = BITS 7 & 6 HIGH 2 BITS OF MAX CYLINDER :
                       ;                     BITS 0-5 MAX SECTORS/TRACK             :
                       ;         CH/[BP+1] = LOW 8 BITS OF MAX CYLINDER             :
                       ;         BL/[BP+2] = BITS 7-4 = 0                           :
                       ;                     BITS 3-0 = VALID CMOS DRIVE TYPE       :
                       ;         BH/[BP+3] = 0                                      :
                       ;         DL/[BP+4] = # DRIVES INSTALLED                     :
                       ;         DH/[BP+5] = MAX HEAD #                             :
                       ;         DI/[BP+6] = OFFSET OF MEDIA/DRIVE PARAMETER TABLE  :
                       ;         ES        = SEGMENT OF MEDIA/DRIVE PARAMETER TABLE :
                       ;         AX        = 0                                      :
                       ;                                                            :
                       ; NOTE : THE ABOVE INFORMATION IS STORED IN THE USERS STACK AT :
                       ;        THE LOCATIONS WHERE THE MAIN ROUTINE WILL POP THEM  :
                       ;        INTO THE APPROPRIATE REGISTERS BEFORE RETURNING TO THE :
                       ;        CALLER.                                             :
                       ;-------------------------------------------------------------
0187                   DISK_PARMS     PROC    NEAR
0187 81 FF 0080               CMP    DI,80H                    ; CHECK FOR FIXED MEDIA TYPE REQUEST
018B 72 06                    JB     DISK_P2                   ; CONTINUE IF NOT REQUEST FALL THROUGH

                       ;----- FIXED DISK REQUEST FALL THROUGH ERROR

018D 8B C6                    MOV    AX,SI                     ; RESTORE AL WITH CALLERS VALUE
018F B4 01                    MOV    AH,BAD_CMD                ; SET BAD COMMAND ERROR IN (AH)
0191 F9                       STC                              ; SET ERROR RETURN CODE
0192 C3                       RET

0193                   DISK_P2:
0193 E8 0404 R                CALL   XLAT_NEW                  ; TRANSLATE STATE TO PRESENT ARCH.
0196 C7 46 02 0000            MOV    WORD PTR [BP+2],0         ; DRIVE TYPE = 0
019B A1 0010 R                MOV    AX,@EQUIP_FLAG            ; LOAD EQUIPMENT FLAG FOR # DISKETTES
019E 24 C1                    AND    AL,11000001B              ; KEEP DISKETTE DRIVE BITS
01A0 D0 E8                    SHR    AL,1                      ; ARE THERE ANY DRIVES INSTALLED?
01A2 73 7C                    JNC    NON_DRV                   ; NC-->NO DRIVES, ZERO PARAMETERS
01A4 D0 C0                    ROL    AL,1                      ; ROTATE TO ORIGINAL POSITION
01A6 D0 C0                    ROL    AL,1                      ; ROTATE BITS 6 AND 7 TO 0 AND 1
01A8 D0 C0                    ROL    AL,1
01AA FE C0                    INC    AL                        ; CONVERT TO RELATIVE 1
01AC 88 46 04                 MOV    [BP+4],AL                 ; STORE NUMBER OF DRIVES
01AF F6 06 008F R 01          TEST   @HF_CNTRL,DUAL            ; CHECK CONTROLLER I.D.
01B4 75 03                    JNZ    DP1_CONT                  ; CONTINUE WITH USUAL PARMS CHECK
01B6 E9 0256 R                JMP    DET_PARMS                 ; RETURN THIS CONTROLLERS PARMS
01B9                   DP1_CONT:
01B9 83 FF 01                 CMP    DI,1                      ; CHECK FOR VALID DRIVE
01BC 77 66                    JA     NON_DRV1                  ; DRIVE INVALID
01BE C6 46 05 01              MOV    BYTE PTR[BP+5],1          ; MAXIMUM HEAD NUMBER = 1
01C2 E8 08CF R                CALL   CMOS_TYPE                 ; RETURN DRIVE TYPE IN AL
01C5 72 16                    JC     CHK_EST                   ; ON CMOS BAD CHECK ESTABLISHED
01C7 0A C0                    OR     AL,AL                     ; TEST FOR NO DRIVE TYPE
01C9 74 12                    JZ     CHK_EST                   ; JUMP IF SO
01CB E8 03B1 R                CALL   DR_TYPE_CHECK             ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
01CE 72 0D                    JC     CHK_EST                   ; TYPE NOT IN TABLE (POSSIBLE BAD CMOS)
01D0 88 46 02                 MOV    [BP+2],AL                 ; STORE VALID CMOS DRIVE TYPE
01D3 2E: 8A 4F 04             MOV    CL,CS:[BX].MD_SEC_TRK     ; GET SECTOR/TRACK
01D7 2E: 8A 6F 0B             MOV    CH,CS:[BX].MD_MAX_TRK     ; GET MAX. TRACK NUMBER
01DB EB 32                    JMP    SHORT STO_CX              ; CMOS GOOD, USE CMOS

01DD                   CHK_EST:
01DD 8A A5 0090 R             MOV    AH,@DSK_STATE[DI]         ; LOAD STATE FOR THIS DRIVE
01E1 F6 C4 10                 TEST   AH,MED_DET                ; CHECK FOR ESTABLISHED STATE
01E4 74 3E                    JZ     NON_DRV1                  ; CMOS BAD/INVALID AND UNESTABLISHED

01E6                   USE_EST:
01E6 80 E4 C0                 AND    AH,RATE_MSK               ; ISOLATE STATE
01E9 80 FC 80                 CMP    AH,RATE_250               ; RATE 250 ?
01EC 75 54                    JNE    USE_EST2                  ; NO, GO CHECK OTHER RATE

                       ;--- DATA RATE IS 250 KBS, TRY 360 KB TABLE FIRST

01EE B0 01                    MOV    AL,01                     ; DRIVE TYPE 1 (360KB)
01F0 E8 03B1 R                CALL   DR_TYPE_CHECK             ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
01F3 2E: 8A 4F 04             MOV    CL,CS:[BX].MD_SEC_TRK     ; GET SECTOR/TRACK
01F7 2E: 8A 6F 0B             MOV    CH,CS:[BX].MD_MAX_TRK     ; GET MAX. TRACK NUMBER
01FB F6 85 0090 R 01          TEST   @DSK_STATE[DI],TRK_CAPA   ; 80 TRACK ?
0200 74 0D                    JZ     STO_CX                    ; MUST BE 360KB DRIVE

                       ;--- IT IS HIGH DATA RATE/80 TRACK DRIVE

0202                   PARM_HDR_80T:
0202 B0 04                    MOV    AL,04                     ; DRIVE TYPE 4
0204 E8 03B1 R                CALL   DR_TYPE_CHECK             ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
0207 2E: 8A 4F 04             MOV    CL,CS:[BX].MD_SEC_TRK     ; GET SECTOR/TRACK
```

```
0208 2E: 8A 6F 0B              MOV     CH,CS:[BX].MD_MAX_TRK   ; GET MAX. TRACK NUMBER

020F                    STO_CX:
020F 89 4E 00                  MOV     [BP],CX                 ; SAVE IN STACK FOR RETURN
0212                    ES_DI:
0212 89 5E 06                  MOV     [BP+6],BX               ; ADDRESS OF MEDIA/DRIVE PARM TABLE
0215 8C C8                     MOV     AX,CS                   ; SEGMENT MEDIA/DRIVE PARAMETER TABLE
0217 8E C0                     MOV     ES,AX                   ; ES IS SEGMENT OF TABLE

0219                    DP_OUT:
0219 E8 0432 R                 CALL    XLAT_OLD                ; TRANSLATE STATE TO COMPATIBLE MODE
021C 33 C0                     XOR     AX,AX                   ; CLEAR
021E F8                        CLC
021F C3                        RET

                        ;----- NO DRIVE PRESENT HANDLER

0220                    NON_DRV:
0220 C6 46 04 00               MOV     BYTE PTR [BP+4],0       ; CLEAR NUMBER OF DRIVES

0224                    NON_DRV1:
0224 81 FF 0080                CMP     DI,80H                  ; CHECK FOR FIXED MEDIA TYPE REQUEST
0228 72 09                     JB      NON_DRV2                ; CONTINUE IF NOT REQUEST FALL THROUGH

                        ;----- FIXED DISK REQUEST FALL THROUGH ERROR

022A                    FD_REQ_ERR:
022A E8 0432 R                 CALL    XLAT_OLD                ; ELSE TRANSLATE TO COMPATIBLE MODE
022D 8B C6                     MOV     AX,SI                   ; RESTORE AL
022F B4 01                     MOV     AH,BAD_CMD              ; SET BAD COMMAND ERROR
0231 F9                        STC                             ; SET ERROR RETURN CODE
0232 C3                        RET

0233                    NON_DRV2:
0233 33 C0                     XOR     AX,AX                   ; CLEAR PARMS IF NO DRIVES OR CMOS BAD
0235 89 46 00                  MOV     [BP],AX                 ; TRACKS, SECTORS/TRACK = 0
0238 88 66 05                  MOV     [BP+5],AH               ; HEAD = 0
023B 89 46 06                  MOV     [BP+6],AX               ; OFFSET TO DISK_BASE = 0
023E 8E C0                     MOV     ES,AX                   ; ES IS SEGMENT OF TABLE
0240 EB D7                     JMP     SHORT DP_OUT

                        ;--- DATA RATE IS EITHER 300 KBS OR 500 KBS, TRY 1.2 MB TABLE FIRST

0242                    USE_EST2:
0242 B0 02                     MOV     AL,02                   ; DRIVE TYPE 2 (1.2MB)
0244 E8 03B1 R                 CALL    DR_TYPE_CHECK           ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
0247 2E: 8A 4F 04              MOV     CL,CS:[BX].MD_SEC_TRK   ; GET SECTOR/TRACK
024B 2E: 8A 6F 0B              MOV     CH,CS:[BX].MD_MAX_TRK   ; GET MAX. TRACK NUMBER
024F 80 FC 40                  CMP     AH,RATE_300             ; RATE 300 ?
0252 74 BB                     JE      STO_CX                  ; MUST BE 1.2MB DRIVE
0254 EB AC                     JMP     SHORT PARM_HDR_80T      ; ELSE, HIGH DATA RATE/80 TRACK DRIVE
0256                    DET_PARMS:
0256 83 FF 03                  CMP     DI,3                    ; REQUEST FOR FIXED DISK?
0259 77 D8                     JA      NON_DRV2                ; YES-->DRIVE NUMBER INVALID
025B B1 09                     MOV     CL,9
025D F6 85 0090 R 01           TEST    @DSK_STATE[DI],TRK_CAPA ; IS DRIVE 80 TRACKS?(RELATIVE ZERO)
0262 B0 01                     MOV     AL,1                    ; SET CMOS TYPE 1
0264 B5 27                     MOV     CH,39                   ; NUMBER OF TRACKS (RELATIVE ZERO)
0266 74 04                     JZ      SET_TYP1                ; IF ZERO TYPE =1
0268 B0 03                     MOV     AL,3                    ; SET CMOS TYPE 3
026A B5 4F                     MOV     CH,79                   ; NUMBER OF TRACKS (RELATIVE ZERO)
026C                    SET_TYP1:
026C 88 46 02                  MOV     [BP+2],AL               ; STORE TYPE
026F C6 46 03 00               MOV     BYTE PTR [BP+3],0
0273 C6 46 05 01               MOV     BYTE PTR [BP+5],1       ; MAXIMUM HEAD NUMBER = 1
0277 E8 03B1 R                 CALL    DR_TYPE_CHECK           ; ADDRESS OF DISK BASE
027A EB 93                     JMP     STO_CX                  ; GO SET TRKS/SEC,CYL,ES:BX AND EXIT

027C                    DISK_PARMS      ENDP

                        ;------------------------------------------------------------
                        ; DISK_TYPE                                                 :
                        ;       THIS ROUTINE RETURNS THE TYPE OF MEDIA INSTALLED.   :
                        ;  ON ENTRY:    DI = DRIVE #                                :
                        ;                                                           :
                        ;  ON EXIT:     AH = DRIVE TYPE, CY=0                       :
                        ;------------------------------------------------------------
027C                    DISK_TYPE       PROC    NEAR
027C F6 06 008F R 01           TEST    @HF_CNTRL,DUAL          ; CHECK CONTROLLER I.D.
0281 74 22                     JZ      NO_CHNG
0283 E8 0404 R                 CALL    XLAT_NEW                ; TRANSLATE STATE TO PRESENT ARCH.
0286 8A 85 0090 R              MOV     AL,@DSK_STATE[DI]       ; GET PRESENT STATE INFORMATION
028A 0A C0                     OR      AL,AL                   ; CHECK FOR NO DRIVE
028C 74 13                     JZ      NO_DRV
028E B4 01                     MOV     AH,NOCHGLN              ; NO CHANGE LINE FOR 40 TRACK DRIVE
0290 A8 01                     TEST    AL,TRK_CAPA             ; IS THIS DRIVE AN 80 TRACK DRIVE?
0292 74 02                     JZ      DT_BACK                 ; IF NO JUMP
0294 B4 02                     MOV     AH,CHGLN                ; CHANGE LINE FOR 80 TRACK DRIVE

0296                    DT_BACK:
0296 50                        PUSH    AX                      ; SAVE RETURN VALUE
0297 E8 0432 R                 CALL    XLAT_OLD                ; TRANSLATE STATE TO COMPATIBLE MODE
029A 58                        POP     AX                      ; RESTORE RETURN VALUE
029B                    DISK_TYPE_EX:                          ; EXIT DISK TYPE FUNCTION
029B F8                        CLC                             ; NO ERROR
029C 8B DE                     MOV     BX,SI                   ; GET SAVED AL TO BL
029E 8A C3                     MOV     AL,BL                   ; PUT BACK FOR RETURN
02A0 C3                        RET
02A1                    NO_DRV:
02A1 32 E4                     XOR     AH,AH                   ; NO DRIVE PRESENT OR UNKNOWN
02A3 EB F1                     JMP     SHORT DT_BACK
02A5                    NO_CHNG:
02A5 A1 0010 R                 MOV     AX,@EQUIP_FLAG          ; LOAD EQUIPMENT FLAG FOR # DISKETTES
02A8 D0 E8                     SHR     AL,1                    ; SHIFT DRIVES PRESENT BIT INTO CARRY
02AA 73 F5                     JNC     NO_DRV                  ; NO DRIVE IN SYSTEM
02AC B4 01                     MOV     AH,1                    ; DISKETTE NO CHANGE LINE AVAILABLE
02AE EB EB                     JMP     DISK_TYPE_EX
02B0                    DISK_TYPE ENDP

                        ;------------------------------------------------------------
                        ; DISK_CHANGE                                               :
                        ;       THIS ROUTINE RETURNS THE STATE OF THE DISK CHANGE LINE. :
                        ;                                                           :
                        ; ON ENTRY:     DI = DRIVE #                                :
                        ;                                                           :
```

SECTION 5

```
                                ; ON EXIT:      AH = @DSKETTE_STATUS                               :
                                ;                       00 - DISK CHANGE LINE INACTIVE, CY = 0     :
                                ;                       06 - DISK CHANGE LINE ACTIVE, CY = 1       :
                                ;-------------------------------------------------------------------
 02B0                           DISK_CHANGE     PROC    NEAR
 02B0 F6 06 008F R 01                   TEST    @HF_CNTRL,DUAL          ; TEST CONTROLLER I.D.
 02B5 75 03                             JNZ     DC1
 02B7 E9 017D R                         JMP     FNC_ERR                 ; ERROR FOR THIS KIND OF CONTROLLER
 02BA                           DC1:
 02BA E8 0404 R                         CALL    XLAT_NEW                ; TRANSLATE STATE TO PRESENT ARCH.
 02BD 8A 85 0090 R                      MOV     AL,@DSK_STATE[DI]       ; GET MEDIA STATE INFORMATION
 02C1 0A C0                             OR      AL,AL                   ; DRIVE PRESENT ?
 02C3 74 19                             JZ      DC_NON                  ; JUMP IF NO DRIVE
 02C5 A8 01                             TEST    AL,TRK_CAPA             ; 80 TRACK DRIVE ?
 02C7 74 05                             JZ      SETIT                   ; IF SO , CHECK CHANGE LINE

 02C9 E8 0B21 R                 DC0:    CALL    READ_DSKCHNG            ; GO CHECK STATE OF DISK CHANGE LINE
 02CC 74 05                             JZ      FINIS                   ; CHANGE LINE NOT ACTIVE

 02CE C6 06 0041 R 06           SETIT:  MOV     @DSKETTE_STATUS,MEDIA_CHANGE    ; INDICATE MEDIA REMOVED

 02D3 E8 0432 R                 FINIS:  CALL    XLAT_OLD                ; TRANSLATE STATE TO COMPATIBLE MODE
 02D6 E8 0832 R                         CALL    SETUP_END               ; VARIOUS CLEANUPS
 02D9 8B DE                             MOV     BX,SI                   ; GET SAVED AL TO BL
 02DB 8A C3                             MOV     AL,BL                   ; PUT BACK FOR RETURN
 02DD C3                                RET

 02DE                           DC_NON:
 02DE 80 0E 0041 R 80                   OR      @DSKETTE_STATUS,TIME_OUT        ; SET TIMEOUT, NO DRIVE
 02E3 EB EE                             JMP     SHORT FINIS
 02E5                           DISK_CHANGE     ENDP

                                ;-------------------------------------------------------------------
                                ; FORMAT_SET                                                        :
                                ;       THIS ROUTINE IS USED TO ESTABLISH THE TYPE OF               :
                                ;       MEDIA TO BE USED FOR THE FOLLOWING FORMAT OPERATION.        :
                                ;                                                                   :
                                ; ON ENTRY:     SI LOW = DASD TYPE FOR FORMAT                       :
                                ;               DI     = DRIVE #                                    :
                                ;                                                                   :
                                ; ON EXIT:      @DSKETTE_STATUS REFLECTS STATUS                     :
                                ;               AH = @DSKETTE_STATUS                                :
                                ;               CY = 1 IF ERROR                                     :
                                ;-------------------------------------------------------------------
 02E5                           FORMAT_SET      PROC    NEAR
 02E5 E8 0404 R                         CALL    XLAT_NEW                ; TRANSLATE STATE TO PRESENT ARCH.
 02E8 56                                PUSH    SI                      ; SAVE DASD TYPE
 02E9 8B C6                             MOV     AX,SI                   ; AH = ? , AL = DASD TYPE
 02EB 32 E4                             XOR     AH,AH                   ; AH = 0 , AL = DASD TYPE
 02ED 8B F0                             MOV     SI,AX                   ; SI = DASD TYPE
 02EF 80 A5 0090 R 0F                   AND     @DSK_STATE[DI],NOT MED_DET+DBL_STEP+RATE_MSK    ; CLEAR STATE
 02F4 4E                                DEC     SI                      ; CHECK FOR 320/360K MEDIA & DRIVE
 02F5 75 07                             JNZ     NOT_320                 ; BYPASS IF NOT
 02F7 80 8D 0090 R 90                   OR      @DSK_STATE[DI],MED_DET+RATE_250 ; SET TO 320/360
 02FC EB 3E                             JMP     SHORT S0

 02FE                           NOT_320:
 02FE F6 06 008F R 01                   TEST    @HF_CNTRL,DUAL          ; TEST CONTROLLER I.D.
 0303 74 0A                             JZ      S3
 0305 E8 05F5 R                         CALL    MED_CHANGE              ; CHECK FOR TIME_OUT
 0308 80 3E 0041 R 80                   CMP     @DSKETTE_STATUS,TIME_OUT
 030D 74 2D                             JZ      S0                      ; IF TIME OUT TELL CALLER

 030F 4E                        S3:     DEC     SI                      ; CHECK FOR 320/360K IN 1.2M DRIVE
 0310 75 07                             JNZ     NOT_320_12              ; BYPASS IF NOT
 0312 80 8D 0090 R 70                   OR      @DSK_STATE[DI],MED_DET+DBL_STEP+RATE_300 ; SET STATE
 0317 EB 23                             JMP     SHORT S0

 0319                           NOT_320_12:
 0319 4E                                DEC     SI                      ; CHECK FOR 1.2M MEDIA IN 1.2M DRIVE
 031A 75 07                             JNZ     NOT_12                  ; BYPASS IF NOT
 031C 80 8D 0090 R 10                   OR      @DSK_STATE[DI],MED_DET+RATE_500  ; SET STATE VARIABLE
 0321 EB 19                             JMP     SHORT S0                ; RETURN TO CALLER

 0323                           NOT_12:
 0323 4E                                DEC     SI                      ; CHECK FOR SET DASD TYPE 04
 0324 75 20                             JNZ     FS_ERR                  ; BAD COMMAND EXIT IF NOT VALID TYPE

 0326 F6 85 0090 R 04                   TEST    @DSK_STATE[DI],DRV_DET  ; DRIVE DETERMINED ?
 032B 74 09                             JZ      ASSUME                  ; IF STILL NOT DETERMINED ASSUME
 032D B0 50                             MOV     AL,MED_DET+RATE_300
 032F F6 85 0090 R 02                   TEST    @DSK_STATE[DI],FMT_CAPA ; MULTIPLE FORMAT CAPABILITY ?
 0334 75 02                             JNZ     OR_IT_IN                ; IF 1.2 M THEN DATA RATE 300

 0336                           ASSUME:
 0336 B0 90                             MOV     AL,MED_DET+RATE_250     ; SET UP

 0338                           OR_IT_IN:
 0338 08 85 0090 R                      OR      @DSK_STATE[DI],AL       ; OR IN THE CORRECT STATE

 033C                           S0:
 033C E8 0432 R                         CALL    XLAT_OLD                ; TRANSLATE STATE TO COMPATIBLE MODE
 033F E8 0832 R                         CALL    SETUP_END               ; VARIOUS CLEANUPS
 0342 5B                                POP     BX                      ; GET SAVED AL TO BL
 0343 8A C3                             MOV     AL,BL                   ; PUT BACK FOR RETURN
 0345 C3                                RET

 0346                           FS_ERR:
 0346 C6 06 0041 R 01                   MOV     @DSKETTE_STATUS,BAD_CMD ; UNKNOWN STATE,BAD COMMAND
 034B EB EF                             JMP     SHORT S0

 034D                           FORMAT_SET      ENDP

                                ;-------------------------------------------------------------------
                                ; SET_MEDIA                                                         :
                                ;       THIS ROUTINE SETS THE TYPE OF MEDIA AND DATA RATE           :
                                ;       TO BE USED FOR THE FOLLOWING FORMAT OPERATION.              :
                                ; ON ENTRY:                                                         :
                                ;       [BP]    = SECTOR PER TRACK                                  :
                                ;       [BP+1]  = TRACK #                                           :
                                ;       DI      = DRIVE #                                           :
                                ; ON EXIT:                                                          :
                                ;       @DSKETTE_STATUS REFLECTS STATUS                             :
                                ;       IF NO ERROR:                                                :
```

```
                                  ;       AH = 0                                             :
                                  ;       CY = 0                                             :
                                  ;       ES = SEGMENT OF MEDIA/DRIVE PARAMETER TABLE        :
                                  ;       DI/[BP+6] = OFFSET OF MEDIA/DRIVE PARAMETER TABLE  :
                                  ;     IF ERROR:                                            :
                                  ;       AH = @DSKETTE_STATUS                               :
                                  ;       CY = 1                                             :
                                  ;------------------------------------------------------------
 034D                             SET_MEDIA       PROC    NEAR
 034D E8 0404 R                           CALL    XLAT_NEW                ; TRANSLATE STATE TO PRESENT ARCH.
 0350 33 DB                               XOR     BX,BX                   ; ZERO INDEX POINTER
 0352 80 7E 01 27                         CMP     BYTE PTR [BP+1],39      ; MAX. TRACK = 40 ?
 0356 75 0B                               JNE     TBL_CHK1
 0358 F6 85 0090 R 01                     TEST    @DSK_STATE[DI],TRK_CAPA ; 80 TRACK DRIVE ?
 035D 74 16                               JZ      MD_FND                  ; POINT TO TABLE ENTRY 1
 035F B3 03                               MOV     BL,3                    ; POINT TO TABLE ENTRY 2
 0361 EB 12                               JMP     SHORT MD_FND
 0363                             TBL_CHK1:
 0363 B3 06                               MOV     BL,6                    ; POINT TO TABLE ENTRY 3
 0365 80 7E 00 0F                         CMP     BYTE PTR [BP],15        ; SECTORS/TRACK = 15 ?
 0369 74 0A                               JE      MD_FND
 036B B3 12                               MOV     BL,18                   ; POINT TO TABLE ENTRY 6
 036D 80 7E 00 12                         CMP     BYTE PTR [BP],18        ; SECTORS/TRACK = 18 ?
 0371 74 02                               JE      MD_FND
 0373 B3 0C                               MOV     BL,12                   ; POINT TO TABLE ENTRY 4
 0375                             MD_FND:
 0375 2E: 8B 9F 0001 R                    MOV     BX,CS:WORD PTR DR_TYPE[BX+1]    ; DI = MEDIA/DRIVE PARAMETER TAB
                      LE
 037A 2E: 8A 47 04                        MOV     AL,CS:[BX].MD_SEC_TRK   ; GET SECTOR/TRACK
 037E 2E: 8A 67 0B                        MOV     AH,CS:[BX].MD_MAX_TRK   ; GET MAX. TRACK #
 0382 39 46 00                            CMP     [BP],AX                 ; MATCH ?
 0385 75 23                               JNE     ER_RTN                  ; NOT SUPPORTED
 0387 2E: 8A 47 0C                        MOV     AL,CS:[BX].MD_RATE      ; GET RATE
 038B 3C 40                               CMP     AL,RATE_300             ; DOUBLE STEP REQUIRED FOR RATE 300
 038D 75 02                               JNE     MD_SET
 038F 0C 20                               OR      AL,DBL_STEP
 0391                             MD_SET:
 0391 89 5E 06                            MOV     [BP+6],BX               ; SAVE TABLE POINTER IN STACK
 0394 0C 10                               OR      AL,MED_DET              ; SET MESIA ESTABLISHED
 0396 80 A5 0090 R 0F                     AND     @DSK_STATE[DI],NOT MED_DET+DBL_STEP+RATE_MSK    ; CLEAR STATE
 039B 08 85 0090 R                        OR      @DSK_STATE[DI],AL       ; SET STATE
 039F 8C C8                               MOV     AX,CS                   ; SEGMENT MEDIA/DRIVE PARAMETER TABLE
 03A1 8E C0                               MOV     ES,AX                   ; ES IS SEGMENT OF TABLE
 03A3                             SM_RTN:
 03A3 E8 0432 R                           CALL    XLAT_OLD                ; TRANSLATE STATE TO COMPATIBLE MODE
 03A6 E8 0832 R                           CALL    SETUP_END               ; VARIOUS CLEANUPS
 03A9 C3                                  RET
 03AA                             ER_RTN:
 03AA C6 06 0041 R 0C                     MOV     @DSKETTE_STATUS,MED_NOT_FND     ; ERROR, MEDIA TYPE NOT FOUND
 03AF EB F2                               JMP     SM_RTN
 03B1                             SET_MEDIA       ENDP

                                  ;------------------------------------------------------------
                                  ; DR_TYPE_CHECK                                             :
                                  ;       CHECK IF THE GIVEN DRIVE TYPE IN REGISTER (AL)      :
                                  ;       IS SUPPORTED IN BIOS DRIVE TYPE TABLE               :
                                  ; ON ENTRY:                                                 :
                                  ;       AL = DRIVE TYPE                                     :
                                  ; ON EXIT:                                                  :
                                  ;       CS = SEGMENT OF MEDIA/DRIVE PARAMETER TABLE (CODE)  :
                                  ;       CY = 0   DRIVE TYPE SUPPORTED                       :
                                  ;            BX = OFFSET TO MEDIA/DRIVE PARAMETER TABLE     :
                                  ;       CY = 1   DRIVE TYPE NOT SUPPORTED                   :
                                  ; REGISTERS ALTERED:  BX                                    :
                                  ;------------------------------------------------------------
 03B1                             DR_TYPE_CHECK   PROC    NEAR
 03B1 50                                  PUSH    AX
 03B2 51                                  PUSH    CX
 03B3 33 DB                               XOR     BX,BX                   ; BX = INDEX TO DR_TYPE TABLE
 03B5 B9 0006                             MOV     CX,DR_CNT               ; CX = LOOP COUNT
 03B8                             TYPE_CHK:
 03B8 2E: 8A A7 0000 R                    MOV     AH,CS:DR_TYPE[BX]       ; GET DRIVE TYPE
 03BD 3A C4                               CMP     AL,AH                   ; DRIVE TYPE MATCH ?
 03BF 74 08                               JE      DR_TYPE_VALID           ; YES, RETURN WITH CARRY RESET
 03C1 83 C3 03                            ADD     BX,3                    ; CHECK NEXT DRIVE TYPE
 03C4 E2 F2                               LOOP    TYPE_CHK
 03C6 F9                                  STC                             ; DRIVE TYPE NOT FOUND IN TABLE
 03C7 EB 05                               JMP     SHORT TYPE_RTN
 03C9                             DR_TYPE_VALID:
 03C9 2E: 8B 9F 0001 R                    MOV     BX,CS:WORD PTR DR_TYPE[BX+1]    ; BX = MEDIA TABLE
 03CE                             TYPE_RTN:
 03CE 59                                  POP     CX
 03CF 58                                  POP     AX
 03D0 C3                                  RET
 03D1                             DR_TYPE_CHECK   ENDP

                                  ;------------------------------------------------------------
                                  ; SEND_SPEC                                                 :
                                  ;       SEND THE SPECIFY COMMAND TO CONTROLLER USING DATA FROM :
                                  ;       THE DRIVE PARAMETER TABLE POINTED BY @DISK_POINTER  :
                                  ; ON ENTRY:    @DISK_POINTER = DRIVE PARAMETER TABLE        :
                                  ; ON EXIT :    NONE                                         :
                                  ; REGISTERS ALTERED: AX                                     :
                                  ;------------------------------------------------------------
 03D1                             SEND_SPEC       PROC    NEAR
 03D1 B8 03EB R                           MOV     AX,OFFSET SPECBAC       ; LOAD ERROR ADDRESS
 03D4 50                                  PUSH    AX                      ; PUSH NEC_OUT ERROR RETURN
 03D5 B4 03                               MOV     AH,03H                  ; SPECIFY COMMAND
 03D7 E8 09F0 R                           CALL    NEC_OUTPUT              ; OUTPUT THE COMMAND
 03DA 2A D2                               SUB     DL,DL                   ; FIRST SPECIFY BYTE
 03DC E8 08FE R                           CALL    GET_PARM                ; GET PARAMETER TO AH
 03DF E8 09F0 R                           CALL    NEC_OUTPUT              ; OUTPUT THE COMMAND
 03E2 B2 01                               MOV     DL,1                    ; SECOND SPECIFY BYTE
 03E4 E8 08FE R                           CALL    GET_PARM                ; GET PARAMETER TO AH
 03E7 E8 09F0 R                           CALL    NEC_OUTPUT              ; OUTPUT THE COMMAND
 03EA 58                                  POP     AX                      ; POP ERROR RETURN
 03EB                             SPECBAC:
 03EB C3                                  RET
 03EC                             SEND_SPEC       ENDP

                                  ;------------------------------------------------------------
                                  ; SEND_SPEC_MD                                              :
                                  ;       SEND THE SPECIFY COMMAND TO CONTROLLER USING DATA FROM :
                                  ;       THE MEDIA/DRIVE PARAMETER TABLE POINTED BY (CS:BX)  :
```

SECTION 5

```
                        ; ON ENTRY:     CS:BX = MEDIA/DRIVE PARAMETER TABLE            :
                        ; ON EXIT :     NONE                                           :
                        ; REGISTERS ALTERED:  AX,CX,DX                                 :
                        ;---------------------------------------------------------------
0 3EC                   SEND_SPEC_MD    PROC    NEAR
03EC B8 0403 R                  MOV     AX,OFFSET SPEC_ESBAC    ; LOAD ERROR ADDRESS
03EF 50                         PUSH    AX                      ; PUSH NEC_OUT ERROR RETURN
03F0 B4 03                      MOV     AH,03H                  ; SPECIFY COMMAND
03F2 E8 09F0 R                  CALL    NEC_OUTPUT              ; OUTPUT THE COMMAND
03F5 2E: 8A 27                  MOV     AH,CS:[BX].MD_SPEC1     ; GET 1ST SPECIFY BYTE
03F8 E8 09F0 R                  CALL    NEC_OUTPUT              ; OUTPUT THE COMMAND
03FB 2E: 8A 67 01               MOV     AH,CS:[BX].MD_SPEC2     ; GET SECOND SPECIFY BYTE
03FF E8 09F0 R                  CALL    NEC_OUTPUT              ; OUTPUT THE COMMAND
0402 58                         POP     AX                      ; POP ERROR RETURN
0403                    SPEC_ESBAC:
0403 C3                         RET
0404                    SEND_SPEC_MD    ENDP
                        ;---------------------------------------------------------------
                        ; XLAT_NEW                                                     :
                        ;       TRANSLATES DISKETTE STATE LOCATIONS FROM COMPATIBLE    :
                        ;       MODE TO NEW ARCHITECTURE.                              :
                        ;                                                              :
                        ; ON ENTRY:     DI : DRIVE                                     :
                        ;---------------------------------------------------------------
0404                    XLAT_NEW        PROC    NEAR            ;
0404 F6 06 008F R 01            TEST    @HF_CNTRL,DUAL          ; TEST CONTROLLER I.D.
0409 74 22                      JZ      XN_OUT
040B 83 FF 01                   CMP     DI,1                    ; VALID DRIVE ?
040E 77 1D                      JA      XN_OUT                  ; IF INVALID BACK
0410 80 BD 0090 R 00            CMP     @DSK_STATE[DI],0        ; NO DRIVE ?
0415 74 17                      JZ      DO_DET                  ; IF NO DRIVE ATTEMPT DETERMINE
0417 8B CF                      MOV     CX,DI                   ; CX = DRIVE NUMBER
0419 D0 E1                      SHL     CL,1                    ; CL = SHIFT COUNT, A=0, B=4
041B D0 E1                      SHL     CL,1
041D A0 008F R                  MOV     AL,@HF_CNTRL            ; DRIVE INFORMATION
0420 D2 C8                      ROR     AL,CL                   ; TO LOW NIBBLE
0422 24 07                      AND     AL,DRV_DET+FMT_CAPA+TRK_CAPA    ; KEEP DRIVE BITS
0424 80 A5 0090 R F8            AND     @DSK_STATE[DI],NOT DRV_DET+FMT_CAPA+TRK_CAPA
0429 08 85 0090 R               OR      @DSK_STATE[DI],AL       ; UPDATE DRIVE STATE
042D                    XN_OUT:
042D C3                         RET                             ;

042E                    DO_DET:
042E E8 0B2B R                  CALL    DRIVE_DET               ; TRY TO DETERMINE
0431 C3                         RET

0432                    XLAT_NEW        ENDP
                        ;---------------------------------------------------------------
                        ; XLAT_OLD                                                     :
                        ;       TRANSLATES DISKETTE STATE LOCATIONS FROM NEW           :
                        ;       ARCHITECTURE TO COMPATIBLE MODE.                       :
                        ;                                                              :
                        ; ON ENTRY:     DI : DRIVE                                     :
                        ;---------------------------------------------------------------
0432                    XLAT_OLD        PROC    NEAR
0432 F6 06 008F R 01            TEST    @HF_CNTRL,DUAL          ; TEST CONTROLLER I.D.
0437 74 79                      JZ      XO_OUT
0439 83 FF 01                   CMP     DI,1                    ; VALID DRIVE ?
043C 77 74                      JA      XO_OUT                  ; IF INVALID BACK
043E 80 BD 0090 R 00            CMP     @DSK_STATE[DI],0        ; NO DRIVE ?
0443 74 6D                      JZ      XO_OUT                  ; IF NO DRIVE TRANSLATE DONE

                        ;-----  TEST FOR SAVED DRIVE INFORMATION ALREADY SET

0445 8B CF                      MOV     CX,DI                   ; CX = DRIVE NUMBER
0447 D0 E1                      SHL     CL,1                    ; CL = SHIFT COUNT, A=0, B=4
0449 D0 E1                      SHL     CL,1
044B B4 02                      MOV     AH,FMT_CAPA             ; LOAD MULTI DATA RATE BIT MASK
044D D2 CC                      ROR     AH,CL                   ; ROTATE BY MASK
044F 84 26 008F R               TEST    @HF_CNTRL,AH            ; MULTI-DATA RATE DETERMINED ?
0453 75 16                      JNZ     SAVE_SET                ; IF SO, NO NEED TO RE-SAVE

                        ;-----  ERASE DRIVE BITS IN @HF_CNTRL FOR THIS DRIVE

0455 B4 07                      MOV     AH,DRV_DET+FMT_CAPA+TRK_CAPA    ; MASK TO KEEP
0457 D2 CC                      ROR     AH,CL                   ; FIX MASK TO KEEP
0459 F6 D4                      NOT     AH                      ; TRANSLATE MASK
045B 20 26 008F R               AND     @HF_CNTRL,AH            ; KEEP BITS FROM OTHER DRIVE INTACT

                        ;-----  ACCESS CURRENT DRIVE BITS AND STORE IN @HF_CNTRL

045F 8A 85 0090 R               MOV     AL,@DSK_STATE[DI]       ; ACCESS STATE
0463 24 07                      AND     AL,DRV_DET+FMT_CAPA+TRK_CAPA    ; KEEP DRIVE BITS
0465 D2 C8                      ROR     AL,CL                   ; FIX FOR THIS DRIVE
0467 08 06 008F R               OR      @HF_CNTRL,AL            ; UPDATE SAVED DRIVE STATE

                        ;-----  TRANSLATE TO COMPATIBILITY MODE

046B                    SAVE_SET:
046B 8A A5 0090 R               MOV     AH,@DSK_STATE[DI]       ; ACCESS STATE
046F 8A FC                      MOV     BH,AH                   ; TO BH FOR LATER
0471 80 E4 C0                   AND     AH,RATE_MSK             ; KEEP ONLY RATE
0474 80 FC 00                   CMP     AH,RATE_500             ; RATE 500 ?
0477 74 10                      JZ      CHK_HDR_80T             ; YES 1.2/1.2 OR HIGH DATA RATE 80 TRK
0479 B0 01                      MOV     AL,M3D1U                ; AL = 360 IN 1.2 UNESTABLISHED
047B 80 FC 40                   CMP     AH,RATE_300             ; RATE 300 ?
047E 75 16                      JNZ     CHK_250                 ; NO, 360/360 ,720/720
0480 F6 C7 20                   TEST    BH,DBL_STEP             ; YES, DOUBLE STEP ?
0483 75 1D                      JNZ     TST_DET                 ; YES, MUST BE 360 IN 1.2

0485                    UNKNO:
0485 B0 07                      MOV     AL,MED_UNK              ; 'NONE OF THE ABOVE'
0487 EB 20                      JMP     SHORT AL_SET            ; PROCESS COMPLETE

0489                    CHK_HDR_80T:
0489 E8 08CF R                  CALL    CMOS_TYPE               ; RETURN DRIVE TYPE IN (AL)
048C 72 F7                      JC      UNKNO                   ; ERROR, SET 'NONE OF THE ABOVE'
048E 3C 02                      CMP     AL,02                   ; 1.2MB DRIVE ?
0490 75 F3                      JNE     UNKNO                   ; NO, GO SET 'NONE OF THE ABOVE'
0492 B0 02                      MOV     AL,M1D1U                ; AL = 1.2 IN 1.2 UNESTABLISHED
0494 EB 0C                      JMP     SHORT TST_DET

0496                    CHK_250:
0496 B0 00                      MOV     AL,M3D3U                ; AL = 360 IN 360 UNESTABLISHED
```

```
0498 80 FC 80                  CMP     AH,RATE_250             ; RATE 250 ?
049B 75 E8                     JNZ     UNKNO                   ; IF SO FALL THRU
049D F6 C7 01                  TEST    BH,TRK_CAPA             ; 80 TRACK CAPABILITY ?
04A0 75 E3                     JNZ     UNKNO                   ; IF SO JUMP, FALL THRU TEST DET

04A2                   TST_DET:
04A2 F6 C7 10                  TEST    BH,MED_DET              ; DETERMINED ?
04A5 74 02                     JZ      AL_SET                  ; IF NOT THEN SET
04A7 04 03                     ADD     AL,3                    ; MAKE DETERMINED/ESTABLISHED

04A9                   AL_SET:
04A9 80 A5 0090 R F8           AND     @DSK_STATE[DI],NOT DRV_DET+FMT_CAPA+TRK_CAPA  ; CLEAR DRIVE
04AE 08 85 0090 R              OR      @DSK_STATE[DI],AL       ; REPLACE WITH COMPATIBLE MODE
04B2                   XO_OUT:
04B2 C3                        RET
04B3                   XLAT_OLD        ENDP

                       ;----------------------------------------------------------------
                       ; RD_WR_VF                                                       :
                       ;       COMMON READ, WRITE AND VERIFY;                           :
                       ;       MAIN LOOP FOR STATE RETRIES.                             :
                       ;                                                                :
                       ; ON ENTRY:     AH : READ/WRITE/VERIFY NEC PARAMETER             :
                       ;               AL : READ/WRITE/VERIFY DMA PARAMETER             :
                       ;                                                                :
                       ; ON EXIT:      @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION  :
                       ;----------------------------------------------------------------
04B3                   RD_WR_VF        PROC    NEAR
04B3 50                        PUSH    AX                      ; SAVE DMA, NEC PARAMETERS
04B4 E8 0404 R                 CALL    XLAT_NEW                ; TRANSLATE STATE TO PRESENT ARCH.
04B7 E8 0561 R                 CALL    SETUP_STATE             ; INITIALIZE START AND END RATE
04BA 58                        POP     AX                      ; RESTORE READ/WRITE/VERIFY

04BB                   DO_AGAIN:
04BB F6 06 008F R 01           TEST    @HF_CNTRL,DUAL          ; TEST CONTROLLER I.D.
04C0 74 0A                     JZ      RWV
04C2 50                        PUSH    AX                      ; SAVE READ/WRITE/VERIFY PARAMETER
04C3 E8 05F5 R                 CALL    MED_CHANGE              ; MEDIA CHANGE AND RESET IF CHANGED
04C6 58                        POP     AX                      ; RESTORE READ/WRITE/VERIFY
04C7 73 03                     JNC     RWV
04C9 E9 0552 R                 JMP     RWV_END
04CC                   RWV:
04CC 50                        PUSH    AX                      ; SAVE READ/WRITE/VERIFY PARAMETER

04CD 8A B5 0090 R              MOV     DH,@DSK_STATE[DI]       ; GET RATE STATE OF THIS DRIVE
04D1 80 E6 C0                  AND     DH,RATE_MSK             ; KEEP ONLY RATE

04D4 E8 08CF R                 CALL    CMOS_TYPE               ; RETURN DRIVE TYPE IN (AL)
04D7 0A C0                     OR      AL,AL                   ; TEST FOR NO DRIVE
04D9 74 2F                     JZ      RWV_ASSUME              ; ASSUME TYPE, USE MAX TRACK
04DB E8 03B1 R                 CALL    DR_TYPE_CHECK           ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
04DE 72 2A                     JC      RWV_ASSUME              ; TYPE NOT IN TABLE ASSUME DEFAULT

                       ;--- SEARCH FOR MEDIA/DRIVE PARAMETER TABLE

04E0 57                        PUSH    DI                      ; SAVE DRIVE #
04E1 33 DB                     XOR     BX,BX                   ; BX = INDEX TO DR_TYPE TABLE
04E3 B9 0006                   MOV     CX,DR_CNT               ; CX = LOOP COUNT
04E6                   RWV_DR_SEARCH:
04E6 2E: 8A A7 0000 R          MOV     AH,CS:DR_TYPE[BX]       ; GET DRIVE TYPE
04EB 80 E4 7F                  AND     AH,BIT7OFF              ; MASK OUT MSB
04EE 3A C4                     CMP     AL,AH                   ; DRIVE TYPE MATCH ?
04F0 75 0B                     JNE     RWV_NXT_MD              ; NO, CHECK NEXT DRIVE TYPE
04F2                   RWV_DR_FND:
04F2 2E: 8B BF 0001 R          MOV     DI,WORD PTR CS:DR_TYPE[BX+1]  ; DI = MEDIA/DRIVE PARAMETER TABLE
04F7                   RWV_MD_SEARCH:
04F7 2E: 3A 75 0C              CMP     DH,CS:[DI].MD_RATE      ; MATCH ?
04FB 74 1D                     JE      RWV_MD_FND              ; YES, GO GET 1ST SPECIFY BYTE
04FD                   RWV_NXT_MD:
04FD 83 C3 03                  ADD     BX,3                    ; CHECK NEXT DRIVE TYPE
0500 E2 E4                     LOOP    RWV_DR_SEARCH
0502 C6 06 0041 R FF           MOV     @DSKETTE_STATUS,0FFH    ; FORCE IT TO RETRY
0507 5F                        POP     DI                      ; RESTORE DRIVE #
0508 EB 3F                     JMP     SHORT CHK_RET           ; GO RETRY

                       ;--- ASSUME PRIMARY DRIVE IS INSTALLED AS SHIPPED

050A                   RWV_ASSUME:
050A BB 0000 E                 MOV     BX,OFFSET MD_TBL1       ; POINT TO 40 TK 250 KBS
050D F6 85 0090 R 01           TEST    @DSK_STATE[DI],TRK_CAPA ; TEST FOR 80 TRACK
0512 74 09                     JZ      RWV_MD_FND1             ; MUST BE 40 TRACK
0514 BB 0000 E                 MOV     BX,OFFSET MD_TBL3       ; POINT TO 80 TK 500 KBS
0517 EB 04 90                  JMP     RWV_MD_FND1             ; GO SET SPECIFY PARAMETERS

                       ;--- CS:BX POINTS TO MEDIA/DRIVE PARAMETER TABLE

051A                   RWV_MD_FND:
051A 8B DF                     MOV     BX,DI                   ; BX = MEDIA/DRIVE PARAMETER TABLE
051C 5F                        POP     DI                      ; RESTORE DRIVE #
051D                   RWV_MD_FND1:

                       ;--- SEND THE SPECIFY COMMAND TO THE CONTROLLER

051D E8 03EC R                 CALL    SEND_SPEC_MD
0520 E8 0658 R                 CALL    CHK_LASTRATE            ; ZF=1 ATTEMPT RATE IS SAME AS LAST RATE
0523 74 03                     JZ      RWV_DBL                 ; YES, SKIP SEND RATE COMMAND
0525 E8 0637 R                 CALL    SEND_RATE               ; SEND DATA RATE TO NEC
0528                   RWV_DBL:
0528 53                        PUSH    BX                      ; SAVE MEDIA/DRIVE PARAM ADDRESS
0529 E8 084C R                 CALL    SETUP_DBL               ; CHECK FOR DOUBLE STEP
052C 5B                        POP     BX                      ; RESTORE ADDRESS
052D 72 1A                     JC      CHK_RET                 ; ERROR FROM READ ID, POSSIBLE RETRY
052F 58                        POP     AX                      ; RESTORE NEC,DMA COMMAND
0530 50                        PUSH    AX                      ; SAVE NEC COMMAND
0531 53                        PUSH    BX                      ; SAVE MEDIA/DRIVE PARAM ADDRESS
0532 E8 0668 R                 CALL    DMA_SETUP               ; SET UP THE DMA
0535 5B                        POP     BX                      ; RESTORE ADDRESS
0536 58                        POP     AX                      ; RESTORE NEC COMMAND
0537 72 1F                     JC      RWV_BAC                 ; CHECK FOR DMA BOUNDARY ERROR
0539 50                        PUSH    AX                      ; SAVE NEC COMMAND
053A 53                        PUSH    BX                      ; SAVE MEDIA/DRIVE PARAM ADDRESS
053B E8 06CB R                 CALL    NEC_INIT                ; INITIALIZE NEC
053E 5B                        POP     BX                      ; RESTORE ADDRESS
```

SECTION 5

```
053F 72 08                      JC      CHK_RET                 ; IF ERROR DO NOT SEND MORE COMMANDS
0541 E8 06F1 R                  CALL    RWV_COM                 ; OP CODE COMMON TO READ/WRITE/VERIFY
0544 72 03                      JC      CHK_RET                 ; IF ERROR DO NOT SEND MORE COMMANDS
0546 E8 0727 R                  CALL    NEC_TERM                ; TERMINATE, GET STATUS, ETC.

0549                    CHK_RET:
0549 E8 07BE R                  CALL    RETRY                   ; CHECK FOR, SETUP RETRY
054C 58                         POP     AX                      ; RESTORE READ/WRITE/VERIFY PARAMETER
054D 73 03                      JNC     RWV_END                 ; CY = 0 NO RETRY
054F E9 04BB R                  JMP     DO_AGAIN                ; CY = 1 MEANS RETRY

0552                    RWV_END:
0552 E8 077B R                  CALL    DSTATE                  ; ESTABLISH STATE IF SUCCESSFUL
0555 E8 0805 R                  CALL    NUM_TRANS               ; AL = NUMBER TRANSFERRED

0558                    RWV_BAC:                                ; BAD DMA ERROR ENTRY
0558 50                         PUSH    AX                      ; SAVE NUMBER TRANSFERRED
0559 E8 0432 R                  CALL    XLAT_OLD                ; TRANSLATE STATE TO COMPATIBLE MODE
055C 58                         POP     AX                      ; RESTORE NUMBER TRANSFERRED
055D E8 0832 R                  CALL    SETUP_END               ; VARIOUS CLEANUPS
0560 C3                         RET
0561                    RD_WR_VF        ENDP
                        ;---------------------------------------------------------------
                        ; SETUP_STATE:  INITIALIZES START AND END RATES.               :
                        ;---------------------------------------------------------------
0561                    SETUP_STATE     PROC    NEAR
0561 F6 06 008F R 01            TEST    @HF_CNTRL,DUAL          ; TEST CONTROLLER I.D.
0566 74 37                      JZ      JIC
0568 F6 85 0090 R 10            TEST    @DSK_STATE[DI],MED_DET  ; MEDIA DETERMINED ?
056D 75 30                      JNZ     JIC                     ; NO STATES IF DETERMINED
056F B8 4080                    MOV     AX,RATE_300*H+RATE_250  ; AH = START RATE,  AL = END RATE
0572 F6 85 0090 R 04            TEST    @DSK_STATE[DI],DRV_DET  ; DRIVE ?
0577 74 0C                      JZ      AX_SET                  ; DO NOT KNOW DRIVE
0579 B0 00                      MOV     AL,RATE_500             ; SET UP FOR 1.2 M END RATE
057B F6 85 0090 R 02            TEST    @DSK_STATE[DI],FMT_CAPA ; 1.2 M ?
0580 75 03                      JNZ     AX_SET                  ; JUMP WITH FIXED END RATE
0582 B8 8080                    MOV     AX,RATE_250*X           ; START & END RATE = 250 FOR 360 DRIVE

0585                    AX_SET:
0585 80 A5 0090 R 1F            AND     @DSK_STATE[DI],NOT RATE_MSK+DBL_STEP ; TURN OFF THE RATE
058A 08 A5 0090 R               OR      @DSK_STATE[DI],AH       ; RATE FIRST TO TRY
058E 80 26 008B R F3            AND     @LASTRATE,NOT STRT_MSK  ; ERASE LAST TO TRY RATE BITS
0593 D0 C8                      ROR     AL,1                    ; TO OPERATION LAST RATE LOCATION
0595 D0 C8                      ROR     AL,1
0597 D0 C8                      ROR     AL,1
0599 D0 C8                      ROR     AL,1
059B 08 06 008B R               OR      @LASTRATE,AL            ; LAST RATE
059F                    JIC:
059F C3                         RET
05A0                    SETUP_STATE     ENDP
                        ;---------------------------------------------------------------
                        ;  FMT_INIT: ESTABLISH STATE IF UNESTABLISHED AT FORMAT TIME.   :
                        ;---------------------------------------------------------------
05A0                    FMT_INIT        PROC    NEAR
05A0 F6 06 008F R 01            TEST    @HF_CNTRL,DUAL          ; TEST CONTROLLER I.D.
05A5 74 49                      JZ      FI_OUT
05A7 F6 85 0090 R 10            TEST    @DSK_STATE[DI],MED_DET  ; IS MEDIA ESTABLISHED
05AC 75 42                      JNZ     FI_OUT                  ; IF SO RETURN
05AE E8 08CF R                  CALL    CMOS_TYPE               ; RETURN DRIVE TYPE IN AL
05B1 72 3E                      JC      CL_DRV                  ; ERROR IN CMOS ASSUME NO DRIVE
05B3 FE C8                      DEC     AL                      ; MAKE ZERO ORIGIN
05B5 78 3A                      JS      CL_DRV                  ; NO DRIVE IF AL 0
05B7 8A A5 0090 R               MOV     AH,@DSK_STATE[DI]       ; AH = CURRENT STATE
05BB 80 E4 0F                   AND     AH,NOT MED_DET+DBL_STEP+RATE_MSK        ; CLEAR
05BE 0A C0                      OR      AL,AL                   ; CHECK FOR 360
05C0 75 05                      JNZ     N_360                   ; IF 360 WILL BE 0
05C2 80 CC 90                   OR      AH,MED_DET+RATE_250     ; ESTABLISH MEDIA
05C5 EB 25                      JMP     SHORT SKP_STATE         ; SKIP OTHER STATE PROCESSING

05C7                    N_360:
05C7 FE C8                      DEC     AL                      ; 1.2 M DRIVE
05C9 75 05                      JNZ     N_12                    ; JUMP IF NOT
05CB 80 CC 10           FI_RATE:OR      AH,MED_DET+RATE_500     ; SET FORMAT RATE
05CE EB 1C                      JMP     SHORT SKP_STATE         ; SKIP OTHER STATE PROCESSING

05D0                    N_12:
05D0 FE C8                      DEC     AL                      ; CHECK FOR TYPE 3
05D2 75 0F                      JNZ     N_720                   ; JUMP IF NOT
05D4 F6 C4 04                   TEST    AH,DRV_DET              ; IS DRIVE DETERMINED
05D7 74 10                      JZ      ISNT_12                 ; TREAT AS NON 1.2 DRIVE
05D9 F6 C4 02                   TEST    AH,FMT_CAPA             ; IS 1.2M
05DC 74 0B                      JZ      ISNT_12                 ; JUMP IF NOT
05DE 80 CC 50                   OR      AH,MED_DET+RATE_300     ; RATE 300
05E1 EB 09                      JMP     SHORT SKP_STATE         ; CONTINUE
05E3                    N_720:
05E3 FE C8                      DEC     AL                      ; CHECK FOR TYPE 4
05E5 75 0A                      JNZ     CL_DRV                  ; NO DRIVE, CMOS BAD
05E7 EB E2                      JMP     SHORT FI_RATE
05E9                    ISNT_12:
05E9 80 CC 90                   OR      AH,MED_DET+RATE_250     ; MUST BE RATE 250
05EC                    SKP_STATE:
05EC 88 A5 0090 R               MOV     @DSK_STATE[DI],AH       ; STORE AWAY
05F0                    FI_OUT:
05F0 C3                         RET
05F1                    CL_DRV:
05F1 32 E4                      XOR     AH,AH                   ; CLEAR STATE
05F3 EB F7                      JMP     SHORT SKP_STATE         ; SAVE IT
05F5                    FMT_INIT        ENDP
                        ;---------------------------------------------------------------
                        ; MED_CHANGE                                                   :
                        ;       CHECKS FOR MEDIA CHANGE, RESETS MEDIA CHANGE,          :
                        ;       CHECKS MEDIA CHANGE AGAIN.                             :
                        ;                                                              :
                        ; ON EXIT:      CY = 1 MEANS MEDIA CHANGE OR TIMEOUT           :
                        ;               @DSKETTE_STATUS = ERROR CODE                   :
                        ;---------------------------------------------------------------
05F5                    MED_CHANGE      PROC    NEAR
05F5 F6 06 008F R 01            TEST    @HF_CNTRL,DUAL          ; TEST CONTROLLER I.D.
05FA 74 37                      JZ      OK2
05FC E8 0B21 R                  CALL    READ_DSKCHNG            ; READ DISK CHANGE LINE STATE
05FF 74 34                      JZ      MC_OUT                  ; BYPASS HANDLING DISK CHANGE LINE
0601 80 A5 0090 R EF            AND     @DSK_STATE[DI],NOT MED_DET      ; CLEAR STATE FOR THIS DRIVE

                        ;       THIS SEQUENCE ENSURES WHENEVER A DISKETTE IS CHANGED THAT
```

```
                                ;       ON THE NEXT OPERATION THE REQUIRED MOTOR START UP TIME WILL
                                ;       BE WAITED. (DRIVE MOTOR MAY GO OFF UPON DOOR OPENING).

0606 8B CF                              MOV     CX,DI                   ; CL = DRIVE #
0608 B0 01                              MOV     AL,1                    ; MOTOR ON BIT MASK
060A D2 E0                              SHL     AL,CL                   ; TO APPROPRIATE POSITION
060C F6 D0                              NOT     AL                      ; KEEP ALL BUT MOTOR ON
060E FA                                 CLI                             ; NO INTERRUPTS
060F 20 06 003F R                       AND     @MOTOR_STATUS,AL        ; TURN MOTOR OFF INDICATOR
0613 FB                                 STI                             ; INTERRUPTS ENABLED
0614 E8 0913 R                          CALL    MOTOR_ON                ; TURN MOTOR ON

                                ;-----  THIS SEQUENCE OF SEEKS IS USED TO RESET DISKETTE CHANGE SIGNAL

0617 E8 0092 R                          CALL    DISK_RESET              ; RESET NEC
061A B5 01                              MOV     CH,01H                  ; MOVE TO CYLINDER 1
061C E8 0A14 R                          CALL    SEEK                    ; ISSUE SEEK
061F 32 ED                              XOR     CH,CH                   ; MOVE TO CYLINDER 0
0621 E8 0A14 R                          CALL    SEEK                    ; ISSUE SEEK
0624 C6 06 0041 R 06                    MOV     @DSKETTE_STATUS,MEDIA_CHANGE    ; STORE IN STATUS

0629 E8 0B21 R                  OK1:    CALL    READ_DSKCHNG            ; CHECK MEDIA CHANGED AGAIN
062C 74 05                              JZ      OK2                     ; IF ACTIVE, NO DISKETTE, TIMEOUT

062E C6 06 0041 R 80            OK4:    MOV     @DSKETTE_STATUS,TIME_OUT; TIMEOUT IF DRIVE EMPTY

0633 F9                         OK2:    STC                             ; MEDIA CHANGED, SET CY
0634 C3                                 RET
0635                            MC_OUT:
0635 F8                                 CLC                             ; NO MEDIA CHANGED, CLEAR CY
0636 C3                                 RET
0637                            MED_CHANGE      ENDP
                                ;----------------------------------------------------------------
                                ; SEND_RATE                                                      :
                                ;       SENDS DATA RATE COMMAND TO NEC                           :
                                ; ON ENTRY:     DI = DRIVE #                                     :
                                ; ON EXIT:      NONE                                             :
                                ; REGISTERS ALTERED: NONE                                        :
                                ;----------------------------------------------------------------
0637                            SEND_RATE       PROC    NEAR            ;
0637 F6 06 008F R 01                    TEST    @HF_CNTRL,DUAL          ; TEST CONTROLLER I.D.
063C 74 19                              JZ      C_S_OUT
063E 50                                 PUSH    AX                      ; SAVE REG.
063F 80 26 008B R 3F                    AND     @LASTRATE,NOT SEND_MSK  ; ELSE CLEAR LAST RATE ATTEMPTED
0644 8A 85 0090 R                       MOV     AL,@DSK_STATE[DI]       ; GET RATE STATE OF THIS DRIVE
0648 24 C0                              AND     AL,SEND_MSK             ; KEEP ONLY RATE BITS
064A 08 06 008B R                       OR      @LASTRATE,AL            ; SAVE NEW RATE FOR NEXT CHECK
064E D0 C0                              ROL     AL,1                    ; MOVE TO BIT OUTPUT POSITIONS
0650 D0 C0                              ROL     AL,1
0652 BA 03F7                            MOV     DX,03F7H                ; OUTPUT NEW DATA RATE
0655 EE                                 OUT     DX,AL
0656 58                                 POP     AX                      ; RESTORE REG.
0657                            C_S_OUT:
0657 C3                                 RET
0658                            SEND_RATE       ENDP

                                ;----------------------------------------------------------------
                                ; CHK_LASTRATE                                                   :
                                ;       CHECK PREVIOUS DATA RATE SENT TO THE CONTROLLER.         :
                                ; ON ENTRY:                                                      :
                                ;       DI = DRIVE #                                             :
                                ; ON EXIT:                                                       :
                                ;       ZF = 1  DATA RATE IS THE SAME AS LAST RATE SENT TO NEC   :
                                ;       ZF = 0  DATA RATE IS DIFFERENT FROM LAST RATE            :
                                ; REGISTERS ALTERED: NONE                                        :
                                ;----------------------------------------------------------------
0658                            CHK_LASTRATE    PROC    NEAR
0658 50                                 PUSH    AX                      ; SAVE REG
0659 8A 26 008B R                       MOV     AH,@LASTRATE            ; GET LAST DATA RATE SELECTED
065D 8A 85 0090 R                       MOV     AL,@DSK_STATE[DI]       ; GET RATE STATE OF THIS DRIVE
0661 25 C0C0                            AND     AX,SEND_MSK*X           ; KEEP ONLY RATE BITS OF BOTH
0664 3A C4                              CMP     AL,AH                   ; COMPARE TO PREVIOUSLY TRIED
                                                                        ; ZF = 1  RATE IS THE SAME
0666 58                                 POP     AX                      ; RESTORE REG.
0667 C3                                 RET
0668                            CHK_LASTRATE    ENDP
```

SECTION 5

```
                                PAGE
                                ;--------------------------------------------------------------
                                ; DMA_SETUP                                                    :
                                ;       THIS ROUTINE SETS UP THE DMA FOR READ/WRITE/VERIFY     :
                                ;       OPERATIONS.                                            :
                                ;                                                              :
                                ; ON ENTRY:     AL = DMA COMMAND                               :
                                ;                                                              :
                                ; ON EXIT:      @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION :
                                ;--------------------------------------------------------------
 0668                           DMA_SETUP       PROC    NEAR
 0668 FA                                CLI                             ; DISABLE INTERRUPTS DURING DMA SET-UP
 0669 E6 0C                             OUT     DMA+12,AL               ; SET THE FIRST/LAST F/F
 066B EB 00                             JMP     $+2                     ; WAIT FOR I/O
 066D E6 0B                             OUT     DMA+11,AL               ; OUTPUT THE MODE BYTE
 066F 3C 42                             CMP     AL,42H                  ; DMA VERIFY COMMAND
 0671 75 04                             JNE     NOT_VERF                ; NO
 0673 33 C0                             XOR     AX,AX                   ; START ADDRESS
 0675 EB 15                             JMP     SHORT J33
 0677                           NOT_VERF:
 0677 8C C0                             MOV     AX,ES                   ; GET THE ES VALUE
 0679 D1 C0                             ROL     AX,1                    ; ROTATE LEFT
 067B D1 C0                             ROL     AX,1
 067D D1 C0                             ROL     AX,1
 067F D1 C0                             ROL     AX,1
 0681 8A E8                             MOV     CH,AL                   ; GET HIGHEST NIBBLE OF ES TO CH
 0683 24 F0                             AND     AL,11110000B            ; ZERO THE LOW NIBBLE FROM SEGMENT
 0685 03 46 02                          ADD     AX,[BP+2]               ; TEST FOR CARRY FROM ADDITION
 0688 73 02                             JNC     J33
 068A FE C5                             INC     CH                      ; CARRY MEANS HIGH 4 BITS MUST BE INC
 068C                           J33:
 068C 50                                PUSH    AX                      ; SAVE START ADDRESS
 068D E6 04                             OUT     DMA+4,AL                ; OUTPUT LOW ADDRESS
 068F EB 00                             JMP     $+2                     ; WAIT FOR I/O
 0691 8A C4                             MOV     AL,AH
 0693 E6 04                             OUT     DMA+4,AL                ; OUTPUT HIGH ADDRESS
 0695 8A C5                             MOV     AL,CH                   ; GET HIGH 4 BITS
 0697 EB 00                             JMP     $+2                     ; I/O WAIT STATE
 0699 24 0F                             AND     AL,00001111B
 069B E6 81                             OUT     081H,AL                 ; OUTPUT HIGH 4 BITS TO PAGE REGISTER

                                ;-----  DETERMINE COUNT

 069D 8B C6                             MOV     AX,SI                   ; AL = # OF SECTORS
 069F 86 C4                             XCHG    AL,AH                   ; AH = # OF SECTORS
 06A1 2A C0                             SUB     AL,AL                   ; AL = 0, AX = # OF SECTORS * 256
 06A3 D1 E8                             SHR     AX,1                    ; AX = # SECTORS * 128
 06A5 50                                PUSH    AX                      ; SAVE # OF SECTORS * 128
 06A6 B2 03                             MOV     DL,3                    ; GET BYTES/SECTOR PARAMETER
 06A8 E8 08FE R                         CALL    GET_PARM                ; "
 06AB 8A CC                             MOV     CL,AH                   ; SHIFT COUNT (0=128, 1=256 ETC)
 06AD 58                                POP     AX                      ; AX = # OF SECTORS * 128
 06AE D3 E0                             SHL     AX,CL                   ; SHIFT BY PARAMETER VALUE
 06B0 48                                DEC     AX                      ;-1 FOR DMA VALUE
 06B1 50                                PUSH    AX                      ; SAVE COUNT VALUE
 06B2 E6 05                             OUT     DMA+5,AL                ; LOW BYTE OF COUNT
 06B4 EB 00                             JMP     $+2                     ; WAIT FOR I/O
 06B6 8A C4                             MOV     AL,AH
 06B8 E6 05                             OUT     DMA+5,AL                ; HIGH BYTE OF COUNT
 06BA FB                                STI                             ; RE-ENABLE INTERRUPTS
 06BB 59                                POP     CX                      ; RECOVER COUNT VALUE
 06BC 58                                POP     AX                      ; RECOVER ADDRESS VALUE
 06BD 03 C1                             ADD     AX,CX                   ; ADD, TEST FOR 64K OVERFLOW
 06BF B0 02                             MOV     AL,2                    ; MODE FOR 8237
 06C1 E6 0A                             OUT     DMA+10,AL               ; INITIALIZE THE DISKETTE CHANNEL

 06C3 73 05                             JNC     NO_BAD                  ; CHECK FOR ERROR
 06C5 C6 06 0041 R 09                   MOV     @DSKETTE_STATUS,DMA_BOUNDARY    ; SET ERROR

 06CA                           NO_BAD:
 06CA C3                                RET                             ; CY SET BY ABOVE IF ERROR
 06CB                           DMA_SETUP       ENDP
                                ;--------------------------------------------------------------
                                ; NEC_INIT                                                     :
                                ;       THIS ROUTINE SEEKS TO THE REQUESTED TRACK AND          :
                                ;       INITIALIZES THE NEC FOR THE READ/WRITE/VERIFY/FORMAT   :
                                ;       OPERATION.                                             :
                                ;                                                              :
                                ; ON ENTRY:     AH : NEC COMMAND TO BE PERFORMED               :
                                ;                                                              :
                                ; ON EXIT:      @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION :
                                ;--------------------------------------------------------------
 06CB                           NEC_INIT        PROC    NEAR
 06CB 50                                PUSH    AX                      ; SAVE NEC COMMAND
 06CC E8 0913 R                         CALL    MOTOR_ON                ; TURN MOTOR ON FOR SPECIFIC DRIVE

                                ;-----  DO THE SEEK OPERATION

 06CF 8A 6E 01                          MOV     CH,[BP+1]               ; CH = TRACK #
 06D2 E8 0A14 R                         CALL    SEEK                    ; MOVE TO CORRECT TRACK
 06D5 58                                POP     AX                      ; RECOVER COMMAND
 06D6 72 18                             JC      ER_1                    ; ERROR ON SEEK
 06D8 BB 06F0 R                         MOV     BX,OFFSET ER_1          ; LOAD ERROR ADDRESS
 06DB 53                                PUSH    BX                      ; PUSH NEC_OUT ERROR RETURN

                                ;-----  SEND OUT THE PARAMETERS TO THE CONTROLLER

 06DC E8 09F0 R                         CALL    NEC_OUTPUT              ; OUTPUT THE OPERATION COMMAND
 06DF 8B C6                             MOV     AX,SI                   ; AH = HEAD #
 06E1 8B DF                             MOV     BX,DI                   ; BL = DRIVE #
 06E3 D0 E4                             SAL     AH,1                    ; MOVE IT TO BIT 2
 06E5 D0 E4                             SAL     AH,1
 06E7 80 E4 04                          AND     AH,00000100B            ; ISOLATE THAT BIT
 06EA 0A E3                             OR      AH,BL                   ; OR IN THE DRIVE NUMBER
 06EC E8 09F0 R                         CALL    NEC_OUTPUT              ; FALL THRU CY SET IF ERROR
 06EF 5B                                POP     BX                      ; THROW AWAY ERROR RETURN
 06F0                           ER_1:
 06F0 C3                                RET
 06F1                           NEC_INIT        ENDP
                                ;--------------------------------------------------------------
                                ; RWV_COM                                                      :
                                ;       THIS ROUTINE SENDS PARAMETERS TO THE NEC SPECIFIC      :
                                ;       TO THE READ/WRITE/VERIFY OPERATIONS.                   :
                                ;                                                              :
```

```
                              ; ON ENTRY:     CS:BX = ADDRESS OF MEDIA/DRIVE PARAMETER TABLE  :
                              ; ON EXIT :     @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION :
                              ;-----------------------------------------------------------------
 06F1                         RWV_COM PROC    NEAR
 06F1 B8 0726 R                       MOV     AX,OFFSET ER_2          ; LOAD ERROR ADDRESS
 06F4 50                              PUSH    AX                      ; PUSH NEC_OUT ERROR RETURN
 06F5 8A 66 01                        MOV     AH,[BP+1]               ; OUTPUT TRACK #
 06F8 E8 09F0 R                       CALL    NEC_OUTPUT
 06FB 8B C6                           MOV     AX,SI                   ; OUTPUT HEAD #
 06FD E8 09F0 R                       CALL    NEC_OUTPUT
 0700 8A 66 00                        MOV     AH,[BP]                 ; OUTPUT SECTOR #
 0703 E8 09F0 R                       CALL    NEC_OUTPUT
 0706 B2 03                           MOV     DL,3                    ; BYTES/SECTOR PARAMETER FROM BLOCK
 0708 E8 08FE R                       CALL    GET_PARM                ; . TO THE NEC
 070B E8 09F0 R                       CALL    NEC_OUTPUT              ; OUTPUT TO CONTROLLER
 070E B2 04                           MOV     DL,4                    ; EOT PARAMETER FROM BLOCK
 0710 E8 08FE R                       CALL    GET_PARM                ; . TO THE NEC
 0713 E8 09F0 R                       CALL    NEC_OUTPUT              ; OUTPUT TO CONTROLLER
 0716 2E: 8A 67 05                    MOV     AH,CS:[BX].MD_GAP        ; GET GAP LENGTH
 071A                         R15:
 071A E8 09F0 R                       CALL    NEC_OUTPUT
 071D B2 06                           MOV     DL,6                    ; DTL PARAMETER FROM BLOCK
 071F E8 08FE R                       CALL    GET_PARM                ;  TO THE NEC
 0722 E8 09F0 R                       CALL    NEC_OUTPUT              ; OUTPUT TO CONTROLLER
 0725 58                              POP     AX                      ; THROW AWAY ERROR EXIT
 0726                         ER_2:
 0726 C3                              RET
 0727                         RWV_COM ENDP
                              ;-----------------------------------------------------------------
                              ; NEC_TERM                                                        :
                              ;       THIS ROUTINE WAITS FOR THE OPERATION THEN ACCEPTS         :
                              ;       THE STATUS FROM THE NEC FOR THE READ/WRITE/VERIFY/        :
                              ;       FORMAT OPERATION.                                         :
                              ;                                                                 :
                              ; ON EXIT:      @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION :
                              ;-----------------------------------------------------------------
 0727                         NEC_TERM        PROC    NEAR

                              ;----- LET THE OPERATION HAPPEN

 0727 56                              PUSH    SI                      ; SAVE HEAD #, # OF SECTORS
 0728 E8 0ABA R                       CALL    WAIT_INT                ; WAIT FOR THE INTERRUPT
 072B 9C                              PUSHF
 072C E8 0AE2 R                       CALL    RESULTS                 ; GET THE NEC STATUS
 072F 72 47                           JC      SET_END_POP
 0731 9D                              POPF
 0732 72 3C                           JC      SET_END                 ; LOOK FOR ERROR

                              ;----- CHECK THE RESULTS RETURNED BY THE CONTROLLER

 0734 FC                              CLD                             ; SET THE CORRECT DIRECTION
 0735 BE 0042 R                       MOV     SI,OFFSET @NEC_STATUS   ; POINT TO STATUS FIELD
 0738 AC                              LODS    @NEC_STATUS             ; GET ST0
 0739 24 C0                           AND     AL,11000000B            ; TEST FOR NORMAL TERMINATION
 073B 74 33                           JZ      SET_END                 ;
 073D 3C 40                           CMP     AL,01000000B            ; TEST FOR ABNORMAL TERMINATION
 073F 75 29                           JNZ     J18                     ; NOT ABNORMAL, BAD NEC

                              ;----- ABNORMAL TERMINATION, FIND OUT WHY

 0741 AC                              LODS    @NEC_STATUS             ; GET ST1
 0742 D0 E0                           SAL     AL,1                    ; TEST FOR EOT FOUND
 0744 B4 04                           MOV     AH,RECORD_NOT_FND
 0746 72 24                           JC      J19
 0748 D0 E0                           SAL     AL,1
 074A D0 E0                           SAL     AL,1
 074C B4 10                           MOV     AH,BAD_CRC
 074E 72 1C                           JC      J19
 0750 D0 E0                           SAL     AL,1                    ; TEST FOR DMA OVERRUN
 0752 B4 08                           MOV     AH,BAD_DMA
 0754 72 16                           JC      J19
 0756 D0 E0                           SAL     AL,1                    ; TEST FOR RECORD NOT FOUND
 0758 D0 E0                           SAL     AL,1
 075A B4 04                           MOV     AH,RECORD_NOT_FND
 075C 72 0E                           JC      J19
 075E D0 E0                           SAL     AL,1
 0760 B4 03                           MOV     AH,WRITE_PROTECT        ; TEST FOR WRITE_PROTECT
 0762 72 08                           JC      J19
 0764 D0 E0                           SAL     AL,1                    ; TEST MISSING ADDRESS MARK
 0766 B4 02                           MOV     AH,BAD_ADDR_MARK
 0768 72 02                           JC      J19

                              ;-----
                              J18:    NEC MUST HAVE FAILED
 076A
 076A B4 20                   J19:    MOV     AH,BAD_NEC
 076C
 076C 08 26 0041 R            SET_END:OR      @DSKETTE_STATUS,AH
 0770
 0770 80 3E 0041 R 01                 CMP     @DSKETTE_STATUS,1       ; SET ERROR CONDITION
 0775 F5                              CMC                             ;
 0776 5E                              POP     SI                      ; RESTORE HEAD #, # OF SECTORS
 0777 C3                              RET

 0778                         SET_END_POP:
 0778 9D                              POPF
 0779 EB F5                           JMP     SHORT SET_END
 077B                         NEC_TERM        ENDP
                              ;-----------------------------------------------------------------
                              ; DSTATE:       ESTABLISH STATE UPON SUCCESSFUL OPERATION.       :
                              ;-----------------------------------------------------------------
 077B                         DSTATE  PROC    NEAR
 077B F6 06 008F R 01                 TEST    @HF_CNTRL,DUAL          ; TEST CONTROLLER I.D.
 0780 74 3B                           JZ      SETBAC
 0782 80 3E 0041 R 00                 CMP     @DSKETTE_STATUS,0       ; CHECK FOR ERROR
 0787 75 34                           JNZ     SETBAC                  ; IF ERROR JUMP
 0789 80 8D 0090 R 10                 OR      @DSK_STATE[DI],MED_DET  ; NO ERROR, MARK MEDIA AS DETERMINED
 078E F6 85 0090 R 04                 TEST    @DSK_STATE[DI],DRV_DET  ; DRIVE DETERMINED ?
 0793 75 28                           JNZ     SETBAC                  ; IF DETERMINED NO TRY TO DETERMINE
 0795 8A 85 0090 R                    MOV     AL,@DSK_STATE[DI]       ; LOAD STATE
 0799 24 C0                           AND     AL,RATE_MSK             ; KEEP ONLY RATE
 079B 3C 80                           CMP     AL,RATE_250             ; RATE 250 ?
 079D 75 19                           JNE     M_12                    ; NO MUST BE 1.2M OR HI DATA RATE 80 TRK

                              ;--- CHECK FOR HIGH DATA RATE 80 TRACK
```

SECTION 5

```

079F E8 08CF R               CALL   CMOS_TYPE                 ; RETURN DRIVE TYPE IN (AL)
07A2 72 14                   JC     M_12                      ; CMOS BAD ASSUME DEFAULT
07A4 3C 02                   CMP    AL,2                      ; TYPE 2 DRIVE ?
07A6 74 10                   JE     M_12                      ; YES-->ASSUME MULTI FORMAT CAPABILITY
07A8 3C 04                   CMP    AL,4                      ; TYPE 4 DRIVE ?
07AA 74 0C                   JE     M_12                      ; YES-->ASSUME MULTI FORMAT CAPABILITY
07AC                  M_720:
07AC 80 A5 0090 R FD         AND    @DSK_STATE[DI],NOT FMT_CAPA      ; TURN OFF FORMAT CAPABILITY
07B1 80 8D 0090 R 04         OR     @DSK_STATE[DI],DRV_DET    ; MARK DRIVE DETERMINED
07B6 EB 05                   JMP    SHORT SETBAC              ; BACK

07B8                  M_12:
07B8 80 8D 0090 R 06         OR     @DSK_STATE[DI],DRV_DET+FMT_CAPA  ; TURN ON DETERMINED & FMT CAPA

07BD                  SETBAC:
07BD C3                      RET
07BE                  DSTATE ENDP
                      ;-------------------------------------------------------------
                      ; RETRY                                                       :
                      ;       DETERMINES WHETHER A RETRY IS NECESSARY. IF RETRY IS   :
                      ;       REQUIRED THEN STATE INFORMATION IS UPDATED FOR RETRY.  :
                      ;                                                             :
                      ; ON EXIT:      CY = 1 FOR RETRY, CY = 0 FOR NO RETRY         :
                      ;-------------------------------------------------------------
07BE                  RETRY  PROC   NEAR
07BE 80 3E 0041 R 00         CMP    @DSKETTE_STATUS,0         ; GET STATUS OF OPERATION
07C3 74 3E                   JZ     NO_RETRY                  ; SUCCESSFUL OPERATION
07C5 80 3E 0041 R 80         CMP    @DSKETTE_STATUS,TIME_OUT       ; IF TIME OUT NO RETRY
07CA 74 37                   JZ     NO_RETRY                  ;
07CC 8A A5 0090 R            MOV    AH,@DSK_STATE[DI]         ; GET MEDIA STATE OF DRIVE
07D0 F6 C4 10                TEST   AH,MED_DET                ; ESTABLISHED/DETERMINED ?
07D3 75 2E                   JNZ    NO_RETRY                  ; IF ESTABLISHED STATE THEN TRUE ERROR
07D5 80 E4 C0                AND    AH,RATE_MSK               ; ISOLATE RATE
07D8 8A 2E 008B R            MOV    CH,@LASTRATE              ; GET START OPERATION STATE
07DC D0 C5                   ROL    CH,1                      ; TO CORRESPONDING BITS
07DE D0 C5                   ROL    CH,1
07E0 D0 C5                   ROL    CH,1
07E2 D0 C5                   ROL    CH,1
07E4 80 E5 C0                AND    CH,RATE_MSK               ; ISOLATE RATE BITS
07E7 3A EC                   CMP    CH,AH                     ; ALL RATES TRIED
07E9 74 18                   JE     NO_RETRY                  ; IF YES, THEN TRUE ERROR

                      ;       SETUP STATE INDICATOR FOR RETRY ATTEMPT TO NEXT RATE
                      ;         00000000B (500) -> 10000000B (250)
                      ;         10000000B (250) -> 01000000B (300)
                      ;         01000000B (300) -> 00000000B (500)

07EB 80 FC 01                CMP    AH,RATE_500+1             ; SET CY FOR RATE 500
07EE D0 DC                   RCR    AH,1                      ; TO NEXT STATE
07F0 80 E4 C0                AND    AH,RATE_MSK               ; KEEP ONLY RATE BITS
07F3 80 A5 0090 R 1F         AND    @DSK_STATE[DI],NOT RATE_MSK+DBL_STEP    ; RATE, DBL STEP OFF
07F8 08 A5 0090 R            OR     @DSK_STATE[DI],AH         ; TURN ON NEW RATE
07FC C6 06 0041 R 00         MOV    @DSKETTE_STATUS,0         ; RESET STATUS FOR RETRY
0801 F9                      STC                              ; SET CARRY FOR RETRY
0802 C3                      RET                              ; RETRY RETURN

0803                  NO_RETRY:
0803 F8                      CLC                              ; CLEAR CARRY NO RETRY
0804 C3                      RET                              ; NO RETRY RETURN
0805                  RETRY  ENDP
                      ;-------------------------------------------------------------
                      ; NUM_TRANS                                                   :
                      ;       THIS ROUTINE CALCULATES THE NUMBER OF SECTORS THAT    :
                      ;       WERE ACTUALLY TRANSFERRED TO/FROM THE DISKETTE.       :
                      ;                                                             :
                      ; ON ENTRY:     [BP+1] = TRACK                                :
                      ;               SI-HI  = HEAD                                 :
                      ;               [BP]   = START SECTOR                         :
                      ;                                                             :
                      ; ON EXIT:      AL = NUMBER ACTUALLY TRANSFERRED              :
                      ;-------------------------------------------------------------
0805                  NUM_TRANS     PROC   NEAR
0805 32 C0                   XOR    AL,AL                     ; CLEAR FOR ERROR
0807 80 3E 0041 R 00         CMP    @DSKETTE_STATUS,0         ; CHECK FOR ERROR
080C 75 23                   JNZ    NT_OUT                    ; IF ERROR 0 TRANSFERRED
080E B2 04                   MOV    DL,4                      ; SECTORS/TRACK OFFSET TO DL
0810 E8 08FE R               CALL   GET_PARM                  ; AH = SECTORS/TRACK
0813 8A 1E 0047 R            MOV    BL,@NEC_STATUS+5          ; GET ENDING SECTOR
0817 8B CE                   MOV    CX,SI                     ; CH = HEAD # STARTED
0819 3A 2E 0046 R            CMP    CH,@NEC_STATUS+4          ; GET HEAD ENDED UP ON
081D 75 0B                   JNZ    DIF_HD                    ; IF ON SAME HEAD, THEN NO ADJUST

081F 8A 2E 0045 R            MOV    CH,@NEC_STATUS+3          ; GET TRACK ENDED UP ON
0823 3A 6E 01                CMP    CH,[BP+1]                 ; IS IT ASKED FOR TRACK
0826 74 04                   JZ     SAME_TRK                  ; IF SAME TRACK NO INCREASE

0828 02 DC                   ADD    BL,AH                     ; ADD SECTORS/TRACK
082A                  DIF_HD:
082A 02 DC                   ADD    BL,AH                     ; ADD SECTORS/TRACK
082C                  SAME_TRK:
082C 2A 5E 00                SUB    BL,[BP]                   ; SUBTRACT START FROM END
082F 8A C3                   MOV    AL,BL                     ; TO AL

0831                  NT_OUT:
0831 C3                      RET
0832                  NUM_TRANS     ENDP
                      ;-------------------------------------------------------------
                      ; SETUP_END                                                   :
                      ;       RESTORES @MOTOR_COUNT TO PARAMETER PROVIDED IN TABLE  :
                      ;       AND LOADS @DSKETTE_STATUS TO AH, AND SETS CY.         :
                      ; ON EXIT:                                                    :
                      ;       AH, @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION   :
                      ;-------------------------------------------------------------
0832                  SETUP_END     PROC   NEAR
0832 B2 02                   MOV    DL,2                      ; GET THE MOTOR WAIT PARAMETER
0834 50                      PUSH   AX                        ; SAVE NUMBER TRANSFERRED
0835 E8 08FE R               CALL   GET_PARM
0838 88 26 0040 R            MOV    @MOTOR_COUNT,AH           ; STORE UPON RETURN
083C 58                      POP    AX                        ; RESTORE NUMBER TRANSFERRED
083D 8A 26 0041 R            MOV    AH,@DSKETTE_STATUS        ; GET STATUS OF OPERATION
0841 0A E4                   OR     AH,AH                     ; CHECK FOR ERROR
0843 74 02                   JZ     NUN_ERR                   ; NO ERROR
0845 32 C0                   XOR    AL,AL                     ; CLEAR NUMBER RETURNED
```

```
0847                        NUN_ERR:
0847 80 FC 01                       CMP     AH,1                    ; SET THE CARRY FLAG TO INDICATE
084A F5                             CMC                             ; SUCCESS OR FAILURE
084B C3                             RET
084C                        SETUP_END       ENDP
                            ;---------------------------------------------------------------
                            ; SETUP_DBL                                                    :
                            ;       CHECK DOUBLE STEP.                                     :
                            ; ON ENTRY:     DI = DRIVE                                     :
                            ;                                                              :
                            ; ON EXIT :     CY = 1 MEANS ERROR                             :
                            ;---------------------------------------------------------------
084C                        SETUP_DBL       PROC    NEAR
084C F6 06 008F R 01                TEST    @HF_CNTRL,DUAL          ; TEST CONTROLLER I.D.
0851 74 65                          JZ      NO_DBL                  ; NO DOUBLE STEPPING REQUIRED
0853 8A A5 0090 R                   MOV     AH,@DSK_STATE[DI]       ; ACCESS STATE
0857 F6 C4 10                       TEST    AH,MED_DET              ; ESTABLISHED STATE ?
085A 75 5C                          JNZ     NO_DBL                  ; IF ESTABLISHED THEN DOUBLE DONE

                            ;-----  CHECK FOR TRACK 0 TO SPEED UP ACKNOWLEDGE OF UNFORMATTED DISKETTE

085C C6 06 003E R 00                MOV     @SEEK_STATUS,0          ; SET RECALIBRATE REQUIRED ON ALL DRIVES
0861 E8 0913 R                      CALL    MOTOR_ON                ; ENSURE MOTOR STAY ON
0864 B5 00                          MOV     CH,0                    ; LOAD TRACK 0
0866 E8 0A14 R                      CALL    SEEK                    ; SEEK TO TRACK 0
0869 E8 08BA R                      CALL    READ_ID                 ; READ ID FUNCTION
086C 72 35                          JC      SD_ERR                  ; IF ERROR NO TRACK 0

                            ;-----  INITIALIZE START AND MAX TRACKS (TIMES 2 FOR BOTH HEADS)

086E B9 0450                        MOV     CX,0450H                ; START, MAX TRACKS
0871 F6 85 0090 R 01                TEST    @DSK_STATE[DI],TRK_CAPA ; TEST FOR 80 TRACK CAPABILITY
0876 74 02                          JZ      CNT_OK                  ; IF NOT COUNT IS SETUP
0878 B1 A0                          MOV     CL,0A0H                 ; MAXIMUM TRACK 1.2 MB

                            ;       ATTEMPT READ ID OF ALL TRACKS, ALL HEADS UNTIL SUCCESS; UPON SUCCESS,
                            ;       MUST SEE IF ASKED FOR TRACK IN SINGLE STEP MODE = TRACK ID READ; IF NOT
                            ;       THEN SET DOUBLE STEP ON.

087A                        CNT_OK:
087A 51                             PUSH    CX                      ; SAVE TRACK, COUNT
087B C6 06 0041 R 00                MOV     @DSKETTE_STATUS,0       ; CLEAR STATUS, EXPECT ERRORS
0880 33 C0                          XOR     AX,AX                   ; CLEAR AX
0882 D0 ED                          SHR     CH,1                    ; HALVE TRACK, CY = HEAD
0884 D0 D0                          RCL     AL,1                    ; AX = HEAD IN CORRECT BIT
0886 D0 D0                          RCL     AL,1
0888 D0 D0                          RCL     AL,1
088A 50                             PUSH    AX                      ; SAVE HEAD
088B E8 0A14 R                      CALL    SEEK                    ; SEEK TO TRACK
088E 58                             POP     AX                      ; RESTORE HEAD
088F 0B F8                          OR      DI,AX                   ; DI = HEAD OR'ED DRIVE
0891 E8 08BA R                      CALL    READ_ID                 ; READ ID HEAD 0
0894 9C                             PUSHF                           ; SAVE RETURN FROM READ_ID
0895 81 E7 00FB                     AND     DI,11111011B            ; TURN OFF HEAD 1 BIT
0899 9D                             POPF                            ; RESTORE ERROR RETURN
089A 59                             POP     CX                      ; RESTORE COUNT
089B 73 08                          JNC     DO_CHK                  ; IF OK, ASKED = RETURNED TRACK ?
089D FE C5                          INC     CH                      ; INC FOR NEXT TRACK
089F 3A E9                          CMP     CH,CL                   ; REACHED MAXIMUM YET
08A1 75 D7                          JNZ     CNT_OK                  ; CONTINUE TILL ALL TRIED

                            ;-----  FALL THRU, READ ID FAILED FOR ALL TRACKS

08A3                        SD_ERR:
08A3 F9                             STC                             ; SET CARRY FOR ERROR
08A4 C3                             RET                             ; SETUP_DBL ERROR EXIT

08A5                        DO_CHK:
08A5 8A 0E 0045 R                   MOV     CL,@NEC_STATUS+3        ; LOAD RETURNED TRACK
08A9 88 8D 0094 R                   MOV     @DSK_TRK[DI],CL         ; STORE TRACK NUMBER
08AD D0 ED                          SHR     CH,1                    ; HALVE TRACK
08AF 3A E9                          CMP     CH,CL                   ; IS IT THE SAME AS ASKED FOR TRACK
08B1 74 05                          JZ      NO_DBL                  ; IF SAME THEN NO DOUBLE STEP
08B3 80 8D 0090 R 20                OR      @DSK_STATE[DI],DBL_STEP ; TURN ON DOUBLE STEP REQUIRED

08B8                        NO_DBL:
08B8 F8                             CLC                             ; CLEAR ERROR FLAG
08B9 C3                             RET
08BA                        SETUP_DBL       ENDP
                            ;---------------------------------------------------------------
                            ; READ_ID                                                      :
                            ;       READ ID FUNCTION.                                      :
                            ; ON ENTRY:     DI = BIT 2 = HEAD; BITS 1,0 = DRIVE            :
                            ;                                                              :
                            ; ON EXIT:      DI = BIT 2 IS RESET, BITS 1,0 = DRIVE          :
                            ;               @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION :
                            ;---------------------------------------------------------------
08BA                        READ_ID PROC    NEAR
08BA B8 08CE R                      MOV     AX,OFFSET ER_3          ; MOVE NEC OUTPUT ERROR ADDRESS
08BD 50                             PUSH    AX
08BE B4 4A                          MOV     AH,4AH                  ; READ ID COMMAND
08C0 E8 09F0 R                      CALL    NEC_OUTPUT              ; TO CONTROLLER
08C3 8B C7                          MOV     AX,DI                   ; DRIVE # TO AH, HEAD 0
08C5 8A E0                          MOV     AH,AL
08C7 E8 09F0 R                      CALL    NEC_OUTPUT              ; TO CONTROLLER
08CA E8 0727 R                      CALL    NEC_TERM                ; WAIT FOR OPERATION, GET STATUS
08CD 58                             POP     AX                      ; THROW AWAY ERROR ADDRESS
08CE                        ER_3:
08CE C3                             RET
08CF                        READ_ID ENDP
                            ;---------------------------------------------------------------
                            ; CMOS_TYPE                                                    :
                            ;       RETURNS DISKETTE TYPE FROM CMOS                        :
                            ;                                                              :
                            ; ON ENTRY:     DI = DRIVE #                                   :
                            ;                                                              :
                            ; ON EXIT:      AL = TYPE; CY REFLECTS STATUS                  :
                            ;---------------------------------------------------------------
08CF                        CMOS_TYPE       PROC    NEAR
08CF A0 0010 R                      MOV     AL,BYTE PTR @EQUIP_FLAG ; LOAD EQUIPMENT FLAG FOR # DISKETTES
08D2 24 C1                          AND     AL,11000001B            ; KEEP DISKETTE DRIVE BITS
08D4 D0 E8                          SHR     AL,1                    ; ARE THERE ANY DRIVES INSTALLED?
08D6 73 20                          JNC     TYP_ZERO                ; NC-->NO DRIVES TYPE ZERO
```

SECTION 5

```
08D8 D0 C0                      ROL     AL,1                    ; ROTATE TO ORIGINAL POSITION
08DA D0 C0                      ROL     AL,1                    ; ROTATE BITS 6 AND 7 TO 0 AND 1
08DC D0 C0                      ROL     AL,1
08DE 32 E4                      XOR     AH,AH                   ; AX=NUMBER OF DRIVES
08E0 3B C7                      CMP     AX,DI                   ; IS DRIVE REQUESTED PRESENT
08E2 72 14                      JC      TYP_ZERO                ; C-->REQUESTED DRIVE NOT PRESENT
08E4 F6 06 008F R 01            TEST    @HF_CNTRL,DUAL          ; TEST CONTROLLER I.D.
08E9 75 10                      JNZ     CR2
08EB F6 85 0090 R 01            TEST    @DSK_STATE[DI],TRK_CAPA ; TEST FOR 80 TRACKS
08F0 B0 01                      MOV     AL,1                    ; DRIVE TYPE HAS 40 TRACKS
08F2 74 06                      JZ      CR1
08F4 B0 03                      MOV     AL,3                    ; DRIVE TYPE HAS 80 TRACKS
08F6 EB 02                      JMP     SHORT CR1
08F8                    TYP_ZERO:
08F8 32 C0                      XOR     AL,AL                   ; DRIVE TYPE 0
08FA                    CR1:                                    ; EXIT WITH AL=TYPE ACCORDING TO TRACKS
08FA C3                         RET
08FB                    CR2:
08FB F9                         STC
08FC EB FC                      JMP     CR1                     ; EXIT WITH CARRY IF DUAL CARD
08FE                    CMOS_TYPE       ENDP

                        ;---------------------------------------------------------------
                        ; GET_PARM                                                     :
                        ;       THIS ROUTINE FETCHES THE INDEXED POINTER FROM THE      :
                        ;       DISK_BASE BLOCK POINTED TO BY THE DATA VARIABLE        :
                        ;       @DISK_POINTER. A BYTE FROM THAT TABLE IS THEN MOVED    :
                        ;       INTO AH, THE INDEX OF THAT BYTE BEING THE PARAMETER    :
                        ;       IN DL.                                                 :
                        ;                                                              :
                        ; ON ENTRY:     DL = INDEX OF BYTE TO BE FETCHED               :
                        ;                                                              :
                        ; ON EXIT:      AH = THAT BYTE FROM BLOCK                      :
                        ;               AL,DH DESTROYED                                :
                        ;---------------------------------------------------------------
08FE                    GET_PARM        PROC    NEAR
08FE 1E                         PUSH    DS
08FF 56                         PUSH    SI
0900 2B C0                      SUB     AX,AX                   ; DS = 0 , BIOS DATA AREA
0902 8E D8                      MOV     DS,AX
0904 87 D3                      XCHG    DX,BX                   ; BL = INDEX
0906 2A FF                      SUB     BH,BH                   ; BX = INDEX
                                ASSUME  DS:ABS0
0908 C5 36 0078 R               LDS     SI,@DISK_POINTER        ; POINT TO BLOCK
090C 8A 20                      MOV     AH,[SI+BX]              ; GET THE WORD
090E 87 D3                      XCHG    DX,BX                   ; RESTORE BX
0910 5E                         POP     SI
0911 1F                         POP     DS
0912 C3                         RET
                                ASSUME  DS:DATA
0913                    GET_PARM        ENDP

                        ;-----------------------------------------------------------------------
                        ; MOTOR_ON                                                             :
                        ;       TURN MOTOR ON AND WAIT FOR MOTOR START UP TIME. THE @MOTOR_COUNT:
                        ;       IS REPLACED WITH A SUFFICIENTLY HIGH NUMBER (0FFH) TO ENSURE   :
                        ;       THAT THE MOTOR DOES NOT GO OFF DURING THE OPERATION. IF THE    :
                        ;       MOTOR NEEDED TO BE TURNED ON, THE MULTITASKING HOOK FUNCTION   :
                        ;       (AX=90FDH, INT 15H) IS CALLED TELLING THE OPERATING SYSTEM     :
                        ;       THAT THE BIOS IS ABOUT TO WAIT FOR MOTOR START UP. IF THIS     :
                        ;       FUNCTION RETURNS WITH CY = 1, IT MEANS THAT THE MINIMUM WAIT   :
                        ;       HAS BEEN COMPLETED. AT THIS POINT A CHECK IS MADE TO ENSURE    :
                        ;       THAT THE MOTOR WASN'T TURNED OFF BY THE TIMER. IF THE HOOK DID :
                        ;       NOT WAIT, THE WAIT FUNCTION (AH=086H) IS CALLED TO WAIT THE    :
                        ;       PRESCRIBED AMOUNT OF TIME. IF THE CARRY FLAG IS SET ON RETURN, :
                        ;       IT MEANS THAT THE FUNCTION IS IN USE AND DID NOT PERFORM THE   :
                        ;       WAIT. A TIMER 1 WAIT LOOP WILL THEN DO THE WAIT.               :
                        ;                                                                      :
                        ; ON ENTRY:     DI = DRIVE #                                           :
                        ;                                                                      :
                        ; ON EXIT:      AX,CX,DX DESTROYED                                     :
                        ;-----------------------------------------------------------------------
0913                    MOTOR_ON        PROC    NEAR
0913 53                         PUSH    BX                      ; SAVE REG.
0914 E8 095E R                  CALL    TURN_ON                 ; TURN ON MOTOR
0917 72 43                      JC      MOT_IS_ON               ; IF CY=1 NO WAIT
0919 E8 0432 R                  CALL    XLAT_OLD                ; TRANSLATE STATE TO COMPATIBLE MODE
091C B8 90FD                    MOV     AX,090FDH               ; LOAD WAIT CODE & TYPE
091F CD 15                      INT     15H                     ; TELL OPERATING SYSTEM ABOUT TO DO WAIT
0921 9C                         PUSHF                           ; SAVE CY FOR TEST
0922 E8 0404 R                  CALL    XLAT_NEW                ; TRANSLATE STATE TO PRESENT ARCH.
0925 9D                         POPF                            ; RESTORE CY FOR TEST
0926 73 05                      JNC     M_WAIT                  ; BYPASS LOOP IF OP SYSTEM HANDLED WAIT
0928 E8 095E R                  CALL    TURN_ON                 ; CHECK AGAIN IF MOTOR ON
092B 72 2F                      JC      MOT_IS_ON               ; IF NO WAIT MEANS IT IS ON

092D                    M_WAIT:
092D B2 0A                      MOV     DL,10                   ; GET THE MOTOR WAIT PARAMETER
092F E8 08FE R                  CALL    GET_PARM
0932 8A C4                      MOV     AL,AH                   ; AL = MOTOR WAIT PARAMETER
0934 32 E4                      XOR     AH,AH                   ; AX = MOTOR WAIT PARAMETER
0936 3C 08                      CMP     AL,8                    ; SEE IF AT LEAST A SECOND IS SPECIFIED
0938 73 02                      JAE     GP2                     ; IF YES, CONTINUE
093A B0 08                      MOV     AL,8                    ; ONE SECOND WAIT FOR MOTOR START UP

                        ;-----  AX CONTAINS NUMBER OF 1/8 SECONDS (125000 MICROSECONDS) TO WAIT

093C 50                 GP2:    PUSH    AX                      ; SAVE WAIT PARAMETER
093D BA F424                    MOV     DX,62500                ; LOAD LARGEST POSSIBLE MULTIPLIER
0940 F7 E2                      MUL     DX                      ; MULTIPLY BY HALF OF WHAT'S NECESSARY
0942 8B CA                      MOV     CX,DX                   ; CX = HIGH WORD
0944 8B D0                      MOV     DX,AX                   ; CX,DX = 1/2 * (# OF MICROSECONDS)
0946 F8                         CLC                             ; CLEAR CARRY FOR ROTATE
0947 D1 D2                      RCL     DX,1                    ; DOUBLE LOW WORD, CY CONTAINS OVERFLOW
0949 D1 D1                      RCL     CX,1                    ; DOUBLE HI, INCLUDING LOW WORD OVERFLOW
094B B4 86                      MOV     AH,86H                  ; LOAD WAIT CODE
094D CD 15                      INT     15H                     ; PERFORM WAIT
094F 58                         POP     AX                      ; RESTORE WAIT PARAMETER
0950 73 0A                      JNC     MOT_IS_ON               ; CY MEANS WAIT COULD NOT BE DONE

                        ;-----  FOLLOWING LOOPS REQUIRED WHEN RTC WAIT FUNCTION IS ALREADY IN USE

0952                    J13:                                    ;       WAIT FOR 1/8 SECOND PER (AL)
0952 B9 205E                    MOV     CX,8286                 ; COUNT FOR 1/8 SECOND AT 15.085737 US
0955 E8 0000 E                  CALL    WAITF                   ; GO TO FIXED WAIT ROUTINE
0958 FE C8                      DEC     AL                      ; DECREMENT TIME VALUE
```

```
095A 75 F6                      JNZ     J13                     ; ARE WE DONE YET

095C                    MOT_IS_ON:
095C 5B                         POP     BX                      ; RESTORE REG.
095D C3                         RET
095E                    MOTOR_ON        ENDP
                        ;---------------------------------------------------------
                        ; TURN_ON                                                 :
                        ;       TURN MOTOR ON AND RETURN WAIT STATE.              :
                        ;                                                         :
                        ; ON ENTRY:     DI = DRIVE #                              :
                        ;                                                         :
                        ; ON EXIT:      CY = 0 MEANS WAIT REQUIRED                :
                        ;               CY = 1 MEANS NO WAIT REQUIRED             :
                        ;               AX,BX,CX,DX DESTROYED                     :
                        ;---------------------------------------------------------
095E                    TURN_ON PROC    NEAR
095E 8B DF                      MOV     BX,DI                   ; BX = DRIVE #
0960 8A CB                      MOV     CL,BL                   ; CL = DRIVE #
0962 D0 C3                      ROL     BL,1                    ; BL = DRIVE SELECT
0964 D0 C3                      ROL     BL,1
0966 D0 C3                      ROL     BL,1
0968 D0 C3                      ROL     BL,1
096A FA                         CLI                             ; NO INTERRUPTS WHILE DETERMINING STATUS
096B C6 06 0040 R FF            MOV     @MOTOR_COUNT,0FFH       ; ENSURE MOTOR STAYS ON FOR OPERATION
0970 A0 003F R                  MOV     AL,@MOTOR_STATUS        ; GET DIGITAL OUTPUT REGISTER REFLECTION
0973 24 30                      AND     AL,00110000B            ; KEEP ONLY DRIVE SELECT BITS
0975 B4 01                      MOV     AH,1                    ; MASK FOR DETERMINING MOTOR BIT
0977 D2 E4                      SHL     AH,CL                   ; AH = MOTOR ON, A=00000001, B=00000010

                        ;  AL = DRIVE SELECT FROM @MOTOR_STATUS
                        ;  BL = DRIVE SELECT DESIRED
                        ;  AH = MOTOR ON MASK DESIRED

0979 3A C3                      CMP     AL,BL                   ; REQUESTED DRIVE ALREADY SELECTED ?
097B 75 06                      JNZ     TURN_IT_ON              ; IF NOT SELECTED JUMP
097D 84 26 003F R               TEST    AH,@MOTOR_STATUS        ; TEST MOTOR ON BIT
0981 75 31                      JNZ     NO_MOT_WAIT             ; JUMP IF MOTOR ON AND SELECTED

0983                    TURN_IT_ON:
0983 0A E3                      OR      AH,BL                   ; AH = DRIVE SELECT AND MOTOR ON
0985 8A 3E 003F R               MOV     BH,@MOTOR_STATUS        ; SAVE COPY OF @MOTOR_STATUS BEFORE
0989 80 E7 0F                   AND     BH,00001111B            ; KEEP ONLY MOTOR BITS
098C 80 26 003F R C0            AND     @MOTOR_STATUS,11000000B ; CLEAR OUT DRIVE SELECT AND MOTORS
0991 08 26 003F R               OR      @MOTOR_STATUS,AH        ; OR IN DRIVE SELECTED
0995 A0 003F R                  MOV     AL,@MOTOR_STATUS        ; GET DIGITAL OUTPUT REGISTER REFLECTION
0998 8A D8                      MOV     BL,AL                   ; BL=@MOTOR_STATUS AFTER, BH=BEFORE
099A 80 E3 0F                   AND     BL,00001111B            ; KEEP ONLY MOTOR BITS
099D FB                         STI                             ; ENABLE INTERRUPTS AGAIN
099E 24 3F                      AND     AL,00111111B            ; STRIP AWAY UNWANTED BITS
09A0 D0 C0                      ROL     AL,1                    ; PUT BITS IN DESIRED POSITIONS
09A2 D0 C0                      ROL     AL,1
09A4 D0 C0                      ROL     AL,1
09A6 D0 C0                      ROL     AL,1
09A8 0C 0C                      OR      AL,00001100B            ; NO RESET, ENABLE DMA/INTERRUPT
09AA BA 03F2                    MOV     DX,03F2H                ; SELECT DRIVE AND TURN ON MOTOR
09AD EE                         OUT     DX,AL                   ; "
09AE 3A DF                      CMP     BL,BH                   ; NEW MOTOR TURNED ON ?
09B0 74 02                      JZ      NO_MOT_WAIT             ; NO WAIT REQUIRED IF JUST SELECT
09B2 F8                         CLC                             ; SET CARRY MEANING WAIT
09B3 C3                         RET

09B4                    NO_MOT_WAIT:
09B4 F9                         STC                             ; SET NO WAIT REQUIRED
09B5 FB                         STI                             ; INTERRUPTS BACK ON
09B6 C3                         RET
09B7                    TURN_ON ENDP
                        ;---------------------------------------------------------
                        ; HD_WAIT                                                 :
                        ;       WAIT FOR HEAD SETTLE TIME.                        :
                        ;                                                         :
                        ; ON ENTRY:     DI : DRIVE #                              :
                        ;                                                         :
                        ; ON EXIT:      AX,BX,CX,DX DESTROYED                     :
                        ;---------------------------------------------------------
09B7                    HD_WAIT         PROC    NEAR
09B7 B2 09                      MOV     DL,9                    ; POINT TO HEAD SETTLE PARAMETER
09B9 E8 08FE R                  CALL    GET_PARM                ; GET PARAMETER
09BC F6 06 003F R 80            TEST    @MOTOR_STATUS,10000000B ; SEE IF A WRITE OPERATION
09C1 74 09                      JZ      ISNT_WRITE              ; IF NOT, DO NOT ENFORCE ANY VALUES
09C3 80 FC 0F                   CMP     AH,15                   ; IS WAIT 15 MILLISECONDS OR GREATER?
09C6 73 08                      JAE     DO_WAT                  ; IF THERE DO NOT ENFORCE
09C8 B4 0F                      MOV     AH,15                   ; HEAD SETTLE MINIMUM
09CA EB 04                      JMP     SHORT DO_WAT            ; DO WAIT OPERATION
09CC                    ISNT_WRITE:
09CC 0A E4                      OR      AH,AH                   ; CHECK FOR WAIT TO BE ZERO
09CE 74 1F                      JZ      HW_DONE                 ; IF NOT WRITE AND 0 THEN EXIT

                        ;-----  AH CONTAINS NUMBER OF MILLISECONDS TO WAIT

09D0                    DO_WAT:
09D0 8A C4                      MOV     AL,AH                   ; AL = # MILLISECONDS
09D2 32 E4                      XOR     AH,AH                   ; AX = # MILLISECONDS
09D4 50                         PUSH    AX                      ; SAVE HEAD SETTLE PARAMETER
09D5 BA 03E8                    MOV     DX,1000                 ; SET UP FOR MULTIPLY TO MICROSECONDS
09D8 F7 E2                      MUL     DX                      ; DX,AX = # MICROSECONDS
09DA 8B CA                      MOV     CX,DX                   ; CX,AX = # MICROSECONDS
09DC 8B D0                      MOV     DX,AX                   ; CX,DX = # MICROSECONDS
09DE B4 86                      MOV     AH,86H                  ; LOAD WAIT CODE
09E0 CD 15                      INT     15H                     ; PERFORM WAIT
09E2 58                         POP     AX                      ; RESTORE HEAD SETTLE PARAMETER
09E3 73 0A                      JNC     HW_DONE                 ; CHECK FOR EVENT WAIT ACTIVE

09E5                    J29:                                    ;       1 MILLISECOND LOOP
09E5 B9 0042                    MOV     CX,66                   ; COUNT AT 15.085737 US PER COUNT
09E8 E8 0000 E                  CALL    WAITF                   ; DELAY FOR 1 MILLISECOND
09EB FE C8                      DEC     AL                      ; DECREMENT THE COUNT
09ED 75 F6                      JNZ     J29                     ; DO AL MILLISECOND # OF TIMES
09EF                    HW_DONE:
09EF C3                         RET
09F0                    HD_WAIT         ENDP
                        ;---------------------------------------------------------
                        ; NEC_OUTPUT                                              :
                        ;       THIS ROUTINE SENDS A BYTE TO THE NEC CONTROLLER AFTER :
```

SECTION 5

```
                                ;       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.                                         :
                                ;                                                              :
                                ; ON ENTRY:                                                    :
                                ;       AH = BYTE TO BE OUTPUT                                 :
                                ; ON EXIT:                                                     :
                                ;       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.                      :
                                ;       AX,CX,DX DESTROYED                                     :
                                ;---------------------------------------------------------------
 09F0                           NEC_OUTPUT      PROC    NEAR
 09F0 53                                PUSH    BX                      ; SAVE REG.
 09F1 BA 03F4                           MOV     DX,03F4H                ; STATUS PORT
 09F4 B3 02                             MOV     BL,2                    ; HIGH ORDER COUNTER
 09F6 33 C9                             XOR     CX,CX                   ; COUNT FOR TIME OUT

 09F8 EC                        J23:    IN      AL,DX                   ; GET STATUS
 09F9 24 C0                             AND     AL,11000000B            ; KEEP STATUS AND DIRECTION
 09FB 3C 80                             CMP     AL,10000000B            ; STATUS 1 AND DIRECTION 0 ?
 09FD 74 0F                             JZ      J27                     ; STATUS AND DIRECTION OK
 09FF E2 F7                             LOOP    J23                     ; CONTINUE TILL CX EXHAUSTED

 0A01 FE CB                             DEC     BL                      ; DECREMENT COUNTER
 0A03 75 F3                             JNZ     J23                     ; REPEAT TILL DELAY FINISHED, CX = 0

                                ;-----  FALL THRU TO ERROR RETURN

 0A05 80 0E 0041 R 80                   OR      @DSKETTE_STATUS,TIME_OUT

 0A0A 5B                                POP     BX                      ; RESTORE REG.

 0A0B 58                                POP     AX                      ; DISCARD THE RETURN ADDRESS
 0A0C F9                                STC                             ; INDICATE ERROR TO CALLER
 0A0D C3                                RET

                                ;-----  DIRECTION AND STATUS OK; OUTPUT BYTE

 0A0E 8A C4                     J27:    MOV     AL,AH                   ; GET BYTE TO OUTPUT
 0A10 42                                INC     DX                      ; DATA PORT = STATUS PORT + 1
 0A11 EE                                OUT     DX,AL                   ; OUTPUT THE BYTE

 0A12 5B                                POP     BX                      ; RESTORE REG.
 0A13 C3                                RET                             ; CY = 0 FROM TEST INSTRUCTION
 0A14                           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.                                  :
                                ;                                                              :
                                ; ON ENTRY:     DI = DRIVE #                                   :
                                ;               CH = TRACK #                                   :
                                ;                                                              :
                                ; ON EXIT:      @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION.:
                                ;               AX,BX,CX,DX DESTROYED                          :
                                ;---------------------------------------------------------------
 0A14                           SEEK    PROC    NEAR
 0A14 8B DF                             MOV     BX,DI                   ; BX = DRIVE #
 0A16 BA 0A7B R                         MOV     DX,OFFSET NEC_ERR       ; LOAD RETURN ADDRESS
 0A19 52                                PUSH    DX                      ; ON STACK FOR NEC_OUTPUT ERROR
 0A1A B0 01                             MOV     AL,1                    ; ESTABLISH MASK FOR RECALIBRATE TEST
 0A1C 86 CB                             XCHG    CL,BL                   ; GET DRIVE VALUE INTO CL
 0A1E D2 C0                             ROL     AL,CL                   ; SHIFT MASK BY THE DRIVE VALUE
 0A20 86 CB                             XCHG    CL,BL                   ; RECOVER TRACK VALUE
 0A22 84 06 003E R                      TEST    AL,@SEEK_STATUS         ; TEST FOR RECALIBRATE REQUIRED
 0A26 75 21                             JNZ     J28A                    ; JUMP IF RECALIBRATE NOT REQUIRED

 0A28 08 06 003E R                      OR      @SEEK_STATUS,AL         ; TURN ON THE NO RECALIBRATE BIT IN FLAG
 0A2C E8 0A7C R                         CALL    RECAL                   ; RECALIBRATE DRIVE
 0A2F 73 0A                             JNC     AFT_RECAL               ; RECALIBRATE DONE

                                ;-----  ISSUE RECALIBRATE FOR 80 TRACK DISKETTES

 0A31 C6 06 0041 R 00                   MOV     @DSKETTE_STATUS,0       ; CLEAR OUT INVALID STATUS
 0A36 E8 0A7C R                         CALL    RECAL                   ; RECALIBRATE DRIVE
 0A39 72 3F                             JC      RB                      ; IF RECALIBRATE FAILS TWICE THEN ERROR
                                ;
 0A3B                           AFT_RECAL:
 0A3B 83 FF 01                          CMP     DI,1                    ; IF REQUEST FOR DRIVE > 2
 0A3E 77 21                             JA      R8                      ; DO SEEK EVERY TIME
 0A40 C6 85 0094 R 00                   MOV     @DSK_TRK[DI],0          ; SAVE NEW CYLINDER AS PRESENT POSITION
 0A45 0A ED                             OR      CH,CH                   ; CHECK FOR SEEK TO TRACK 0
 0A47 74 2C                             JZ      DO_WAIT                 ; HEAD SETTLE, CY = 0 IF JUMP

                                ;-----  DRIVE IS IN SYNCHRONIZATION WITH CONTROLLER, SEEK TO TRACK

 0A49                           J28A:
 0A49 83 FF 01                          CMP     DI,1                    ; IF REQUEST FOR DRIVE > 2
 0A4C 77 13                             JA      R8                      ; DO SEEK EVERY TIME
 0A4E F6 85 0090 R 20                   TEST    @DSK_STATE[DI],DBL_STEP ; CHECK FOR DOUBLE STEP REQUIRED
 0A53 74 02                             JZ      R7                      ; SINGLE STEP REQUIRED BYPASS DOUBLE
 0A55 D0 E5                             SHL     CH,1                    ; DOUBLE NUMBER OF STEP TO TAKE

 0A57 3A AD 0094 R              R7:     CMP     CH,@DSK_TRK[DI]         ; SEE IF ALREADY AT THE DESIRED TRACK
 0A5B 74 1D                             JE      RB                      ; IF YES, DO NOT NEED TO SEEK

 0A5D 88 AD 0094 R                      MOV     @DSK_TRK[DI],CH         ; SAVE NEW CYLINDER AS PRESENT POSITION
 0A61                           R8:
 0A61 51                                PUSH    CX                      ; SAVE CYLINDER #
 0A62 B4 0F                             MOV     AH,0FH                  ; SEEK COMMAND TO NEC
 0A64 E8 09F0 R                         CALL    NEC_OUTPUT
 0A67 8B DF                             MOV     BX,DI                   ; BX = DRIVE #
 0A69 8A E3                             MOV     AH,BL                   ; OUTPUT DRIVE NUMBER
 0A6B E8 09F0 R                         CALL    NEC_OUTPUT
 0A6E 58                                POP     AX                      ; RESTORE CYLINDER # FOR NEC_OUTPUT
 0A6F E8 09F0 R                         CALL    NEC_OUTPUT
 0A72 E8 0A93 R                         CALL    CHK_STAT_2              ; ENDING INTERRUPT AND SENSE STATUS
```

```

                              ;----- WAIT FOR HEAD SETTLE

0A75                          DO_WAIT:
0A75 9C                               PUSHF                           ; SAVE STATUS
0A76 E8 09B7 R                        CALL    HD_WAIT                 ; WAIT FOR HEAD SETTLE TIME
0A79 9D                               POPF                            ; RESTORE STATUS
0A7A                          RB:
0A7A 58                               POP     AX                      ; CLEAR ERROR RETURN FROM NEC_OUTPUT
0A7B                          NEC_ERR:
0A7B C3                               RET                             ; RETURN TO CALLER
0A7C                          SEEK    ENDP

                              ;---------------------------------------------------------------
                              ; RECAL                                                        :
                              ;       RECALIBRATE DRIVE                                      :
                              ;                                                              :
                              ; ON ENTRY      DI = DRIVE #                                   :
                              ;                                                              :
                              ; ON EXIT:      CY  REFLECTS STATUS OF OPERATION.              :
                              ;---------------------------------------------------------------
0A7C                          RECAL   PROC    NEAR
0A7C 51                               PUSH    CX
0A7D B8 0A91 R                        MOV     AX,OFFSET RC_BACK       ; LOAD NEC_OUTPUT ERROR
0A80 50                               PUSH    AX
0A81 B4 07                            MOV     AH,07H                  ; RECALIBRATE COMMAND
0A83 E8 09F0 R                        CALL    NEC_OUTPUT
0A86 8B DF                            MOV     BX,DI                   ; BX = DRIVE #
0A88 8A E3                            MOV     AH,BL
0A8A E8 09F0 R                        CALL    NEC_OUTPUT              ; OUTPUT THE DRIVE NUMBER
0A8D E8 0A93 R                        CALL    CHK_STAT_2              ; GET THE INTERRUPT AND SENSE INT STATUS
0A90 58                               POP     AX                      ; THROW AWAY ERROR
0A91                          RC_BACK:
0A91 59                               POP     CX
0A92 C3                               RET
0A93                          RECAL   ENDP

                              ;---------------------------------------------------------------
                              ; CHK_STAT_2                                                   :
                              ;       THIS ROUTINE HANDLES THE INTERRUPT RECEIVED AFTER      :
                              ;       RECALIBRATE, SEEK, OR RESET TO THE ADAPTER. THE        :
                              ;       INTERRUPT IS WAITED FOR, THE INTERRUPT STATUS SENSED,  :
                              ;       AND THE RESULT RETURNED TO THE CALLER.                 :
                              ;                                                              :
                              ; ON EXIT:      @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION.:
                              ;---------------------------------------------------------------
0A93                          CHK_STAT_2      PROC    NEAR
0A93 B8 0AB1 R                        MOV     AX,OFFSET CS_BACK       ; LOAD NEC_OUTPUT ERROR ADDRESS
0A96 50                               PUSH    AX
0A97 E8 0ABA R                        CALL    WAIT_INT                ; WAIT FOR THE INTERRUPT
0A9A 72 14                            JC      J34                     ; IF ERROR, RETURN IT
0A9C B4 08                            MOV     AH,08H                  ; SENSE INTERRUPT STATUS COMMAND
0A9E E8 09F0 R                        CALL    NEC_OUTPUT
0AA1 E8 0AE2 R                        CALL    RESULTS                 ; READ IN THE RESULTS
0AA4 72 0A                            JC      J34
0AA6 A0 0042 R                        MOV     AL,@NEC_STATUS          ; GET THE FIRST STATUS BYTE
0AA9 24 60                            AND     AL,01100000B            ; ISOLATE THE BITS
0AAB 3C 60                            CMP     AL,01100000B            ; TEST FOR CORRECT VALUE
0AAD 74 03                            JZ      J35                     ; IF ERROR, GO MARK IT
0AAF F8                               CLC                             ; GOOD RETURN
0AB0                          J34:
0AB0 58                               POP     AX                      ; THROW AWAY ERROR RETURN
0AB1                          CS_BACK:
0AB1 C3                               RET

0AB2                          J35:
0AB2 80 0E 0041 R 40                  OR      @DSKETTE_STATUS,BAD_SEEK
0AB7 F9                               STC                             ; ERROR RETURN CODE
0AB8 EB F6                            JMP     SHORT J34
0ABA                          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.             :
                              ;                                                              :
                              ; ON EXIT:      @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION.:
                              ;---------------------------------------------------------------
0ABA                          WAIT_INT        PROC    NEAR
0ABA FB                               STI                             ; TURN ON INTERRUPTS, JUST IN CASE
0ABB F8                               CLC                             ; CLEAR TIMEOUT INDICATOR
0ABC B8 9001                          MOV     AX,09001H               ; LOAD WAIT CODE AND TYPE
0ABF CD 15                            INT     15H                     ; PERFORM OTHER FUNCTION
0AC1 72 11                            JC      J36A                    ; BYPASS TIMING LOOP IF TIMEOUT DONE

0AC3 B3 04                            MOV     BL,4                    ; CLEAR THE COUNTERS
0AC5 33 C9                            XOR     CX,CX                   ; FOR 2 SECOND WAIT
0AC7                          J36:
0AC7 F6 06 003E R 80                  TEST    @SEEK_STATUS,INT_FLAG   ; TEST FOR INTERRUPT OCCURRING
0ACC 75 0C                            JNZ     J37
0ACE E2 F7                            LOOP    J36                     ; COUNT DOWN WHILE WAITING
0AD0 FE CB                            DEC     BL                      ; SECOND LEVEL COUNTER
0AD2 75 F3                            JNZ     J36

0AD4 80 0E 0041 R 80          J36A:   OR      @DSKETTE_STATUS,TIME_OUT        ; NOTHING HAPPENED
0AD9 F9                               STC                             ; ERROR RETURN
0ADA                          J37:
0ADA 9C                               PUSHF                           ; SAVE CURRENT CARRY
0ADB 80 26 003E R 7F                  AND     @SEEK_STATUS,NOT INT_FLAG       ; TURN OFF INTERRUPT FLAG
0AE0 9D                               POPF                            ; RECOVER CARRY
0AE1 C3                               RET                             ; GOOD RETURN CODE
0AE2                          WAIT_INT        ENDP

                              ;---------------------------------------------------------------
                              ; RESULTS                                                      :
                              ;       THIS ROUTINE WILL READ ANYTHING THAT THE NEC CONTROLLER :
                              ;       RETURNS FOLLOWING AN INTERRUPT.                        :
                              ;                                                              :
                              ; ON EXIT:      @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION.:
                              ;               AX,BX,CX,DX DESTROYED                          :
                              ;---------------------------------------------------------------
0AE2                          RESULTS PROC    NEAR
0AE2 57                               PUSH    DI
0AE3 BF 0042 R                        MOV     DI,OFFSET @NEC_STATUS   ; POINTER TO DATA AREA
0AE6 B3 07                            MOV     BL,7                    ; MAX STATUS BYTES
0AE8 BA 03F4                          MOV     DX,03F4H                ; STATUS PORT

```

SECTION 5

```
                                ;-----  WAIT FOR REQUEST FOR MASTER

 0AEB B7 02                     R10:    MOV     BH,2                    ; HIGH ORDER COUNTER
 0AED 33 C9                             XOR     CX,CX                   ; COUNTER
 0AEF                           J39:                                    ; WAIT FOR MASTER
 0AEF EC                                IN      AL,DX                   ; GET STATUS
 0AF0 24 C0                             AND     AL,11000000B            ; KEEP ONLY STATUS AND DIRECTION
 0AF2 3C C0                             CMP     AL,11000000B            ; STATUS 1 AND DIRECTION 0 ?
 0AF4 74 0E                             JZ      J42                     ; STATUS AND DIRECTION OK
 0AF6 E2 F7                             LOOP    J39                     ; LOOP TILL TIMEOUT

 0AF8 FE CF                             DEC     BH                      ; DECREMENT HIGH ORDER COUNTER
 0AFA 75 F3                             JNZ     J39                     ; REPEAT TILL DELAY DONE

 0AFC 80 0E 0041 R 80                   OR      @DSKETTE_STATUS,TIME_OUT
 0B01 F9                                STC                             ; SET ERROR RETURN
 0B02 EB 1B                             JMP     SHORT POPRES            ; POP REGISTERS AND RETURN

                                ;-----  READ IN THE STATUS

 0B04                           J42:
 0B04 42                                INC     DX                      ; POINT AT DATA PORT
 0B05 EC                                IN      AL,DX                   ; GET THE DATA
 0B06 88 05                             MOV     [DI],AL                 ; STORE THE BYTE
 0B08 47                                INC     DI                      ; INCREMENT THE POINTER

 0B09 B9 0002                           MOV     CX,2                    ; MINIMUM 12 MICROSECONDS FOR NEC
 0B0C E8 0000 E                         CALL    WAITF                   ; WAIT 15 TO 30 MICROSECONDS
 0B0F 4A                                DEC     DX                      ; POINT AT STATUS PORT
 0B10 EC                                IN      AL,DX                   ; GET STATUS
 0B11 A8 10                             TEST    AL,00010000B            ; TEST FOR NEC STILL BUSY
 0B13 74 0A                             JZ      POPRES                  ; RESULTS DONE ?

 0B15 FE CB                             DEC     BL                      ; DECREMENT THE STATUS COUNTER
 0B17 75 D2                             JNZ     R10                     ; GO BACK FOR MORE
 0B19 80 0E 0041 R 20                   OR      @DSKETTE_STATUS,BAD_NEC ; TOO MANY STATUS BYTES
 0B1E F9                                STC                             ; SET ERROR FLAG

                                ;-----  RESULT OPERATION IS DONE

 0B1F                           POPRES:
 0B1F 5F                                POP     DI
 0B20 C3                                RET                             ; RETURN WITH CARRY SET
 0B21                           RESULTS ENDP
                                ;----------------------------------------------------------------
                                ; READ_DSKCHNG                                                  :
                                ;         READS THE STATE OF THE DISK CHANGE LINE.              :
                                ;                                                               :
                                ; ON ENTRY:       DI = DRIVE #                                  :
                                ;                                                               :
                                ; ON EXIT:        DI = DRIVE #                                  :
                                ;                 ZF = 0 : DISK CHANGE LINE INACTIVE            :
                                ;                 ZF = 1 : DISK CHANGE LINE ACTIVE              :
                                ;                 AX,CX,DX DESTROYED                            :
                                ;----------------------------------------------------------------
 0B21                           READ_DSKCHNG    PROC    NEAR
 0B21 E8 0913 R                         CALL    MOTOR_ON                ; TURN ON THE MOTOR IF OFF
 0B24 BA 03F7                           MOV     DX,03F7H                ; ADDRESS DIGITAL INPUT REGISTER
 0B27 EC                                IN      AL,DX                   ; INPUT DIGITAL INPUT REGISTER
 0B28 A8 80                             TEST    AL,DSK_CHG              ; CHECK FOR DISK CHANGE LINE ACTIVE
 0B2A C3                                RET                             ; RETURN TO CALLER WITH ZERO FLAG SET
 0B2B                           READ_DSKCHNG    ENDP
                                ;----------------------------------------------------------------
                                ; DRIVE_DET                                                     :
                                ;       DETERMINES WHETHER DRIVE IS 80 OR 40 TRACKS AND         :
                                ;       UPDATES STATE INFORMATION ACCORDINGLY.                  :
                                ;                                                               :
                                ; ON ENTRY:       DI = DRIVE #                                  :
                                ;----------------------------------------------------------------
 0B2B                           DRIVE_DET       PROC    NEAR
 0B2B E8 0913 R                         CALL    MOTOR_ON                ; TURN ON MOTOR IF NOT ALREADY ON
 0B2E E8 0A7C R                         CALL    RECAL                   ; RECALIBRATE DRIVE
 0B31 72 3E                             JC      DD_BAC                  ; ASSUME NO DRIVE PRESENT
 0B33 B5 30                             MOV     CH,TRK_SLAP             ; SEEK TO TRACK 48
 0B35 E8 0A14 R                         CALL    SEEK                    ; "
 0B38 72 37                             JC      DD_BAC                  ; ERROR NO DRIVE
 0B3A B5 0B                             MOV     CH,QUIET_SEEK+1         ; SEEK TO TRACK 10
 0B3C                           SK_GIN:
 0B3C FE CD                             DEC     CH                      ; DECREMENT TO NEXT TRACK
 0B3E 78 26                             JS      IS_40                   ; END LOOP IF CYLINDER COUNT NEGATIVE
 0B40 51                                PUSH    CX                      ; SAVE TRACK
 0B41 E8 0A14 R                         CALL    SEEK                    ; "
 0B44 72 2C                             JC      POP_BAC                 ; POP AND RETURN
 0B46 B8 0B71 R                         MOV     AX,OFFSET DD_BAC        ; LOAD NEC OUTPUT ERROR ADDRESS
 0B49 50                                PUSH    AX                      ; "
 0B4A B4 04                             MOV     AH,SENSE_DRV_ST         ; SENSE DRIVE STATUS COMMAND BYTE
 0B4C E8 09F0 R                         CALL    NEC_OUTPUT              ; OUTPUT TO NEC
 0B4F 8B C7                             MOV     AX,DI                   ; AL = DRIVE
 0B51 8A E0                             MOV     AH,AL                   ; AH = DRIVE
 0B53 E8 09F0 R                         CALL    NEC_OUTPUT              ; OUTPUT TO NEC
 0B56 E8 0AE2 R                         CALL    RESULTS                 ; GO GET STATUS
 0B59 58                                POP     AX                      ; THROW AWAY ERROR ADDRESS
 0B5A 59                                POP     CX                      ; RESTORE TRACK
 0B5B F6 06 0042 R 10                   TEST    @NEC_STATUS,HOME        ; TRACK 0 ?
 0B60 74 DA                             JZ      SK_GIN                  ; GO TILL TRACK 0
 0B62 0A ED                             OR      CH,CH                   ; IS HOME AT TRACK 0 ?
 0B64 74 06                             JZ      IS_80                   ; MUST BE 80 TRACK DRIVE

                                ;       DRIVE IS A 360; SET DRIVE TO DETERMINED;
                                ;       SET MEDIA TO DETERMINED AT RATE 250.
 0B66                           IS_40:
 0B66 80 8D 0090 R 94                   OR      @DSK_STATE[DI],DRV_DET+MED_DET+RATE_250
 0B6B C3                                RET                             ; ALL INFORMATION SET

 0B6C                           IS_80:
 0B6C 80 8D 0090 R 01                   OR      @DSK_STATE[DI],TRK_CAPA ; SETUP 80 TRACK CAPABILITY
 0B71                           DD_BAC:
 0B71 C3                                RET

 0B72                           POP_BAC:
 0B72 59                                POP     CX                      ; THROW AWAY
 0B73 C3                                RET

 0B74                           DRIVE_DET       ENDP
```

```
                               ;-------------------------------------------------------------------
                               ; DISK_INT                                                         :
                               ;       THIS ROUTINE HANDLES THE DISKETTE INTERRUPT.               :
                               ;                                                                  :
                               ; ON EXIT:      THE INTERRUPT FLAG IS SET IN @SEEK_STATUS.         :
                               ;-------------------------------------------------------------------
 0B74                          DISK_INT_1      PROC    FAR             ; ENTRY POINT FOR ORG 0EF57H
 0B74 FB                               STI                             ; RE-ENABLE INTERRUPTS
 0B75 50                               PUSH    AX                      ; SAVE WORK REGISTER
 0B76 1E                               PUSH    DS                      ; SAVE REGISTERS
 0B77 E8 0000 E                        CALL    DDS                     ; SETUP DATA ADDRESSING
 0B7A 80 0E 003E R 80                  OR      @SEEK_STATUS,INT_FLAG   ; TURN ON INTERRUPT OCCURRED
 0B7F 1F                               POP     DS                      ; RESTORE USER (DS)
 0B80 B0 20                            MOV     AL,EOI                  ; END OF INTERRUPT MARKER
 0B82 E6 20                            OUT     INTA00,AL               ; INTERRUPT CONTROL PORT
 0B84 B8 9101                          MOV     AX,09101H               ; INTERRUPT POST CODE AND TYPE
 0B87 CD 15                            INT     15H                     ; GO PERFORM OTHER TASK
 0B89 58                               POP     AX                      ; RECOVER REGISTER
 0B8A CF                               IRET                            ; RETURN FROM INTERRUPT
 0B8B                          DISK_INT_1      ENDP
                               ;-------------------------------------------------------------------
                               ; DSKETTE_SETUP                                                    :
                               ;       THIS ROUTINE DOES A PRELIMINARY CHECK TO SEE WHAT TYPE     :
                               ;       OF DISKETTE DRIVES ARE ATTACHED TO THE SYSTEM.             :
                               ;-------------------------------------------------------------------
 0B8B                          DSKETTE_SETUP PROC      NEAR
 0B8B 50                               PUSH    AX                      ; SAVE REGISTERS
 0B8C 53                               PUSH    BX
 0B8D 51                               PUSH    CX
 0B8E 52                               PUSH    DX
 0B8F 57                               PUSH    DI
 0B90 56                               PUSH    SI
 0B91 1E                               PUSH    DS
 0B92 E8 0000 E                        CALL    DDS                     ; POINT DATA SEGMENT TO BIOS DATA AREA
 0B95 80 0E 00A0 R 01                  OR      @RTC_WAIT_FLAG,01       ; NO RTC WAIT, FORCE USE OF LOOP
 0B9A C7 06 0090 R 0000                MOV     WORD PTR @DSK_STATE,0   ; INITIALIZE STATES
 0BA0 80 26 008B R 33                  AND     @LASTRATE,NOT STRT_MSK+SEND_MSK  ; CLEAR START & SEND
 0BA5 80 0E 008B R C0                  OR      @LASTRATE,SEND_MSK      ; INITIALIZE SENT TO IMPOSSIBLE
 0BAA C6 06 003E R 00                  MOV     @SEEK_STATUS,0          ; INDICATE RECALIBRATE NEEDED
 0BAF C6 06 0040 R 00                  MOV     @MOTOR_COUNT,0          ; INITIALIZE MOTOR COUNT
 0BB4 C6 06 003F R 00                  MOV     @MOTOR_STATUS,0         ; INITIALIZE DRIVES TO OFF STATE
 0BB9 C6 06 0041 R 00                  MOV     @DSKETTE_STATUS,0       ; NO ERRORS
 0BBE A0 0010 R                        MOV     AL,BYTE PTR @EQUIP_FLAG ; GET EQUIPMENT STATUS
 0BC1 D0 C0                            ROL     AL,1                    ; SHIFT BITS 7,6 TO 1,0
 0BC3 D0 C0                            ROL     AL,1
 0BC5 24 03                            AND     AL,3                    ; MASK DRIVE BITS
 0BC7 32 E4                            XOR     AH,AH                   ; AX=NUMBER OF DRIVES(RELATIVE ZERO)
 0BC9 33 FF                            XOR     DI,DI                   ; DI=INITIAL DRIVE TO BE ESTABLISHED
 0BCB BE 0010                          MOV     SI,HOME                 ; SI=HOME MASK FOR ALL DRIVES
 0BCE                          SUP0:
 0BCE F6 06 008F R 01                  TEST    @HF_CNTRL,DUAL          ; TEST CONTROLLER TYPE
 0BD3 75 05                            JNZ     SUP1
 0BD5 C6 85 0090 R 94                  MOV     @DSK_STATE[DI],DRV_DET+MED_DET+RATE_250
 0BDA                          SUP1:
 0BDA 50                               PUSH    AX                      ; SAVE DRIVE COUNT
 0BDB E8 0B2B R                        CALL    DRIVE_DET               ; DETERMINE DRIVE
 0BDE E8 0432 R                        CALL    XLAT_OLD                ; TRANSLATE STATE TO COMPATIBLE MODE
 0BE1 23 36 0042 R                     AND     SI,WORD PTR @NEC_STATUS ; AND @NEC_STATUS WITH HOME MASK
 0BE5 58                               POP     AX                      ; RESTORE DRIVE COUNT
 0BE6 47                               INC     DI                      ; POINT TO NEXT DRIVE
 0BE7 3B F8                            CMP     DI,AX
 0BE9 76 E3                            JNA     SUP0                    ; REPEAT FOR EACH DRIVE
 0BEB                          SUP2:
 0BEB C6 06 003E R 00                  MOV     @SEEK_STATUS,0          ; FORCE RECALIBRATE
 0BF0 80 26 00A0 R FE                  AND     @RTC_WAIT_FLAG,0FEH     ; ALLOW FOR RTC WAIT
 0BF5 E8 0832 R                        CALL    SETUP_END               ; VARIOUS CLEANUPS
 0BF8 72 05                            JC      HOME_OK                 ; EXIT WITH CY FLAG FROM SETUP_END
 0BFA 0B F6                            OR      SI,SI                   ; TEST HOME INDICATORS FOR ALL DRIVES
 0BFC 75 01                            JNZ     HOME_OK
 0BFE F9                               STC                             ; ERROR-->HOME INDICATOR BAD
 0BFF                          HOME_OK:
 0BFF 1F                               POP     DS                      ; RESTORE CALLERS RESISTERS
 0C00 5E                               POP     SI
 0C01 5F                               POP     DI
 0C02 5A                               POP     DX
 0C03 59                               POP     CX
 0C04 5B                               POP     BX
 0C05 58                               POP     AX
 0C06 C3                               RET
 0C07                          DSKETTE_SETUP ENDP
 0C07                          CODE    ENDS
                                       END
```

SECTION 5

```
                                PAGE 118,121
                                TITLE KEYBRD --- 01/10/86  KEYBOARD ADAPTER BIOS
                                ;---- 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                               :
                                ;       (AH)=5  PLACE ASCII CHARACTER/SCAN CODE COMBINATION IN KEYBOARD :
                                ;               BUFFER AS IF STRUCK FROM KEYBOARD                      :
                                ;                                                                      :
                                ;               ENTRY:  (CL) = ASCII CHARACTER                         :
                                ;                       (CH) = SCAN CODE                               :
                                ;                                                                      :
                                ;               EXIT:   (AL) = 00H = SUCCESSFUL OPERATION              :
                                ;                       (AL) = 01H = UNSUCCESSFUL - BUFFER FULL        :
                                ;                                                                      :
                                ;                                                                      :
                                ;       (AH)=10H EXTENDED READ INTERFACE FOR THE ENHANCED KEYBOARD     :
                                ;       (AH)=11H EXTENDED ASCII STATUS FOR THE ENHANCED KEYBOARD,      :
                                ;               OTHERWISE SAME AS FUNCTION AH=1                        :
                                ;       (AH)=12H RETURN THE EXTENDED SHIFT STATUS IN AX REGISTER       :
                                ;               AL = BITS FROM @KB_FLAG, AH = BITS FOR LEFT AND RIGHT  :
                                ;               CTL AND ALT KEYS FROM @KB_FLAG_1 AND @KB_FLAG_3        :
                                ;                                                                      :
                                ;               EXIT:                                                  :
                                ;          ---------------                                             :
                                ;          |7|6|5|4|3|2|1|0|     AH REGISTER                           :
                                ;          ---------------                                             :
                                ;           | | | | | | | |                                            :
                                ;           | | | | | | | '----- LEFT CONTROL KEY IS DEPRESSED         :
                                ;           | | | | | | '------- LEFT ALTERNATE SHIFT KEY IS DEPRESSED :
                                ;           | | | | | '--------- RIGHT CONTROL KEY IS DEPRESSED        :
                                ;           | | | | '----------- RIGHT ALTERNATE SHIFT KEY IS DEPRESSED :
                                ;           | | | '------------- SCROLL LOCK KEY IS DEPRESSED          :
                                ;           | | '--------------- NUM LOCK KEY IS DEPRESSED             :
                                ;           | '----------------- CAPS LOCK KEY IS DEPRESSED            :
                                ;           '------------------- SYSTEM KEY IS DEPRESSED               :
                                ;                                                                      :
                                ;          ---------------                                             :
                                ;          |7|6|5|4|3|2|1|0|     AL REGISTER                           :
                                ;          ---------------                                             :
                                ;           | | | | | | | |                                            :
                                ;           | | | | | | | '----- RIGHT SHIFT KEY IS DEPRESSED           :
                                ;           | | | | | | '------- LEFT SHIFT KEY IS DEPRESSED            :
                                ;           | | | | | '--------- CONTROL SHIFT KEY IS DEPRESSED         :
                                ;           | | | | '----------- ALTERNATE SHIFT KEY IS DEPRESSED       :
                                ;           | | | '------------- SCROLL LOCK STATE HAS BEEN TOGGLED     :
                                ;           | | '--------------- NUM LOCK STATE HAS BEEN TOGGLED        :
                                ;           | '----------------- CAPS LOCK STATE HAS BEEN TOGGLED       :
                                ;           '------------------- INSERT STATE IS ACTIVE                :
                                ;                                                                      :
                                ; OUTPUT                                                               :
                                ;       AS NOTED ABOVE, ONLY AX AND FLAGS CHANGED                      :
                                ;       ALL OTHER REGISTERS PRESERVED                                  :
                                ;-----------------------------------------------------------------------

                                EXTRN   DDS:NEAR
                                EXTRN   RESET:NEAR
                                EXTRN   BEEP:NEAR

                                PUBLIC  KEYBOARD_IO_1
                                PUBLIC  KB_INT_1

                                .LIST

 0000                           CODE    SEGMENT BYTE    PUBLIC
                                        ASSUME  CS:CODE,DS:DATA
 0000                           KEYBOARD_IO_1   PROC    FAR     ;
 0000  FB                               STI                     ; INTERRUPTS BACK ON
 0001  1E                               PUSH    DS              ; SAVE CURRENT DS
 0002  53                               PUSH    BX              ; SAVE BX TEMPORARILY
 0003  51                               PUSH    CX
 0004  E8 0000 E                        CALL    DDS             ; ESTABLISH POINTER TO DATA REGION
 0007  0A E4                            OR      AH,AH           ; AH=0
 0009  74 26                            JZ      K1              ; ASCII_READ
 000B  FE CC                            DEC     AH              ; AH=1
 000D  74 37                            JZ      K2              ; ASCII_STATUS
 000F  FE CC                            DEC     AH              ; AH=2
 0011  74 64                            JZ      K3              ; SHIFT_STATUS
 0013  80 EC 03                         SUB     AH,3            ; AH=5
 0016  74 64                            JZ      K500            ; KEYBOARD WRITE
 0018  80 EC 0B                         SUB     AH,0BH          ; AH=10
 001B  74 0C                            JZ      K1E             ; EXTENDED_ASCII_READ
 001D  FE CC                            DEC     AH              ; AH=11
 001F  74 1A                            JZ      K2E             ; EXTENDED_ASCII_STATUS
 0021  FE CC                            DEC     AH              ; AH=12
 0023  74 39                            JZ      K3E             ; EXTENDED_SHIFT_STATUS
 0025                           KIO_EXIT:
 0025  59                               POP     CX
 0026  5B                               POP     BX              ; RECOVER REGISTER
 0027  1F                               POP     DS              ; RECOVER SEGMENT
 0028  CF                               IRET                    ; INVALID COMMAND

                                ;------ ASCII CHARACTER

 0029  E8 009E R                K1E:    CALL    K1S             ; GET A CHARACTER FROM THE BUFFER (EXTENDED)
 002C  E8 00D1 R                        CALL    KIO_E_XLAT      ; ROUTINE TO XLATE FOR EXTENDED CALLS
 002F  EB F4                            JMP     KIO_EXIT        ; GIVE IT TO THE CALLER

 0031  E8 009E R                K1:     CALL    K1S             ; GET A CHARACTER FROM THE BUFFER
 0034  E8 00DC R                        CALL    KIO_S_XLAT      ; ROUTINE TO XLATE FOR STANDARD CALLS
 0037  72 F8                            JC      K1              ; CARRY SET MEANS THROW CODE AWAY
 0039  EB EA                            JMP     KIO_EXIT
```

```
                            ;------ ASCII STATUS

0038 E8 00C4 R              K2E:    CALL    K2S             ; TEST FOR CHARACTER IN BUFFER (EXTENDED)
003E 74 18                          JZ      K2B             ; RETURN IF BUFFER EMPTY
0040 9C                             PUSHF                   ; SAVE ZF FROM TEST
0041 E8 00D1 R                      CALL    KIO_E_XLAT      ; ROUTINE TO XLATE FOR EXTENDED CALLS
0044 EB 11                          JMP     SHORT K2A       ; GIVE IT TO THE CALLER

0046 E8 00C4 R              K2:     CALL    K2S             ; TEST FOR CHARACTER IN BUFFER
0049 74 0D                          JZ      K2B             ; RETURN IF BUFFER EMPTY
004B 9C                             PUSHF                   ; SAVE ZF FROM TEST
004C E8 00DC R                      CALL    KIO_S_XLAT      ; ROUTINE TO XLATE FOR STANDARD CALLS
004F 73 06                          JNC     K2A             ; CARRY CLEAR MEANS PASS VALID CODE
0051 9D                             POPF                    ;  INVALID CODE FOR THIS TYPE OF CALL
0052 E8 009E R                      CALL    K1S             ;  THROW THE CHARACTER AWAY
0055 EB EF                          JMP     K2              ; GO LOOK FOR NEXT CHAR, IF ANY

0057 9D                     K2A:    POPF                    ; RESTORE ZF FROM TEST
0058 59                     K2B:    POP     CX
0059 5B                             POP     BX              ; RECOVER REGISTER
005A 1F                             POP     DS              ; RECOVER SEGMENT
005B CA 0002                        RET     2               ; THROW AWAY FLAGS

                            ;------ SHIFT STATUS

005E                        K3E:                            ; GET THE EXTENDED SHIFT STATUS FLAGS
005E 8A 26 0018 R                   MOV     AH,@KB_FLAG_1   ; GET SYSTEM SHIFT KEY STATUS
0062 80 E4 04                       AND     AH,SYS_SHIFT    ; MASK ALL BUT SYS KEY BIT
0065 B1 05                          MOV     CL,5            ; SHIFT THE SYSTEM KEY BIT OVER TO
0067 D2 E4                          SHL     AH,CL           ;  BIT 7 POSITION
0069 A0 0018 R                      MOV     AL,@KB_FLAG_1   ; GET SHIFT STATES BACK
006C 24 73                          AND     AL,01110011B    ; ELIMINATE SYS_SHIFT, HOLD_STATE, AND INS_SHIFT
006E 0A E0                          OR      AH,AL           ; MERGE THE REMAINING BITS INTO AH
0070 A0 0096 R                      MOV     AL,@KB_FLAG_3   ; GET RIGHT CTL AND ALT
0073 24 0C                          AND     AL,00001100B    ; ELIMINATE LC_E0 AND LC_E1
0075 0A E0                          OR      AH,AL           ; OR THE SHIFT FLAGS TOGETHER
0077 A0 0017 R              K3:     MOV     AL,@KB_FLAG     ; GET THE SHIFT STATUS FLAGS
007A EB A9                          JMP     KIO_EXIT        ; RETURN TO CALLER

                            ;------ WRITE TO KEYBOARD BUFFER

007C                        K500:
007C 56                             PUSH    SI
007D FA                             CLI
007E 8B 1E 001C R                   MOV     BX,[@BUFFER_TAIL]; GET THE "IN TO" POINTER TO THE BUFFER
0082 8B F3                          MOV     SI,BX           ; SAVE A COPY IN CASE BUFFER NOT FULL
0084 E8 0114 R                      CALL    K4              ; BUMP THE POINTER TO SEE IF BUFFER IS FULL
0087 3B 1E 001A R                   CMP     BX,[@BUFFER_HEAD]; WILL THE BUFFER OVERRUN IF WE STORE THIS?
008B 74 0B                          JE      K502            ;  YES - INFORM CALLER OF ERROR
008D 89 0C                          MOV     [SI],CX         ;  NO  - PUT THE ASCII/SCAN CODE INTO BUFFER
008F 89 1E 001C R                   MOV     [@BUFFER_TAIL],BX; ADJUST "IN TO" POINTER TO REFLECT CHANGE
0093 2A C0                          SUB     AL,AL           ; TELL CALLER THAT OPERATION WAS SUCCESSFUL
0095 EB 03 90                       JMP     K504            ; SUB INSTRUCTION ALSO RESETS CARRY FLAG
0098                        K502:
0098 B0 01                          MOV     AL,01H          ; BUFFER FULL INDICATION
009A                        K504:
009A FB                             STI
009B 5E                             POP     SI
009C EB 87                          JMP     KIO_EXIT        ; RETURN TO CALLER WITH STATUS IN AL

009E                        KEYBOARD_IO_1 ENDP
```

SECTION 5

```
                                PAGE
                                ;------ READ THE KEY TO FIGURE OUT WHAT TO DO ------------------------

 009E                           K1S     PROC    NEAR
 009E 8B 1E 001A R                      MOV     BX,@BUFFER_HEAD ; GET POINTER TO HEAD OF BUFFER
 00A2 3B 1E 001C R                      CMP     BX,@BUFFER_TAIL ; TEST END OF BUFFER
 00A6 75 05                             JNE     K1T             ; IF ANYTHING IN BUFFER DONT DO INTERRUPT

 00A8 B8 9002                           MOV     AX,09002H       ; MOVE IN WAIT CODE & TYPE
 00AB CD 15                             INT     15H             ; PERFORM OTHER FUNCTION
 00AD                           K1T:                            ; ASCII READ
 00AD FB                                STI                     ; INTERRUPTS BACK ON DURING LOOP
 00AE 90                                NOP                     ; ALLOW AN INTERRUPT TO OCCUR
 00AF FA                                CLI                     ; INTERRUPTS BACK OFF
 00B0 8B 1E 001A R                      MOV     BX,@BUFFER_HEAD ; GET POINTER TO HEAD OF BUFFER
 00B4 3B 1E 001C R                      CMP     BX,@BUFFER_TAIL ; TEST END OF BUFFER
 00B8 74 F3                             JE      K1T             ; LOOP UNTIL SOMETHING IN BUFFER
 00BA 8B 07                             MOV     AX,[BX]         ; GET SCAN CODE AND ASCII CODE
 00BC E8 0114 R                         CALL    K4              ; MOVE POINTER TO NEXT POSITION
 00BF 89 1E 001A R                      MOV     @BUFFER_HEAD,BX ; STORE VALUE IN VARIABLE
 00C3 C3                                RET                     ; RETURN
 00C4                           K1S     ENDP


                                ;------ READ THE KEY TO SEE IF ONE IS PRESENT ------------------------

 00C4                           K2S     PROC    NEAR
 00C4 FA                                CLI                     ; INTERRUPTS OFF
 00C5 8B 1E 001A R                      MOV     BX,@BUFFER_HEAD ; GET HEAD POINTER
 00C9 3B 1E 001C R                      CMP     BX,@BUFFER_TAIL ; IF EQUAL (Z=1) THEN NOTHING THERE
 00CD 8B 07                             MOV     AX,[BX]
 00CF FB                                STI                     ; INTERRUPTS BACK ON
 00D0 C3                                RET                     ; RETURN
 00D1                           K2S     ENDP


                                ;------ ROUTINE TO TRANSLATE SCAN CODE PAIRS FOR EXTENDED CALLS

 00D1                           KIO_E_XLAT:
 00D1 3C F0                             CMP     AL,0F0h         ; IS IT ONE OF THE FILL-INs?
 00D3 75 06                             JNE     KIO_E_RET       ;  NO, PASS IT ON
 00D5 0A E4                             OR      AH,AH           ; AH = 0 IS SPECIAL CASE
 00D7 74 02                             JZ      KIO_E_RET       ;  PASS THIS ON UNCHANGED
 00D9 32 C0                             XOR     AL,AL           ;  OTHERWISE SET AL = 0
 00DB                           KIO_E_RET:
 00DB C3                                RET                     ; GO BACK


                                ;------ ROUTINE TO TRANSLATE SCAN CODE PAIRS FOR STANDARD CALLS

 00DC                           KIO_S_XLAT:
 00DC 80 FC E0                          CMP     AH,0E0h         ; IS IT KEYPAD ENTER OR / ?
 00DF 75 12                             JNE     KIO_S2          ;  NO, CONTINUE
 00E1 3C 0D                             CMP     AL,0Dh          ; KEYPAD ENTER CODE?
 00E3 74 09                             JE      KIO_S1          ;  YES, MASSAGE A BIT
 00E5 3C 0A                             CMP     AL,0Ah          ; CTRL KEYPAD ENTER CODE?
 00E7 74 05                             JE      KIO_S1          ;  YES, MASSAGE THE SAME
 00E9 B4 35                             MOV     AH,35h          ;  NO, MUST BE KEYPAD /
 00EB EB 23 90                          JMP     KIO_USE         ; GIVE TO CALLER
 00EE B4 1C                     KIO_S1: MOV     AH,1Ch          ; CONVERT TO COMPATIBLE OUTPUT
 00F0 EB 1E 90                          JMP     KIO_USE         ; GIVE TO CALLER

 00F3 80 FC 84                  KIO_S2: CMP     AH,84h          ; IS IT ONE OF THE EXTENDED ONES?
 00F6 77 1A                             JA      KIO_DIS         ;  YES, THROW AWAY AND GET ANOTHER CHAR

 00F8 3C F0                             CMP     AL,0F0h         ; IS IT ONE OF THE FILL-INs?
 00FA 75 07                             JNE     KIO_S3          ;  NO, TRY LAST TEST
 00FC 0A E4                             OR      AH,AH           ; AH = 0 IS SPECIAL CASE
 00FE 74 10                             JZ      KIO_USE         ;  PASS THIS ON UNCHANGED
 0100 EB 10 90                          JMP     KIO_DIS         ;  THROW AWAY THE REST

 0103 3C E0                     KIO_S3: CMP     AL,0E0h         ; IS IT AN EXTENSION OF A PREVIOUS ONE?
 0105 75 09                             JNE     KIO_USE         ;  NO, MUST BE A STANDARD CODE
 0107 0A E4                             OR      AH,AH           ; AH = 0 IS SPECIAL CASE
 0109 74 05                             JZ      KIO_USE         ; JUMP IF AH = 0
 010B 32 C0                             XOR     AL,AL           ; CONVERT TO COMPATIBLE OUTPUT
 010D EB 01 90                          JMP     KIO_USE         ; PASS IT ON TO CALLER

 0110                           KIO_USE:
 0110 F8                                CLC                     ; CLEAR CARRY TO INDICATE GOOD CODE
 0111 C3                                RET                     ; RETURN
 0112                           KIO_DIS:
 0112 F9                                STC                     ; SET CARRY TO INDICATE DISCARD CODE
 0113 C3                                RET                     ; RETURN
```

```
                                PAGE
                                ;----------------------------------------------------------------
                                ;       INCREMENT BUFFER POINTER ROUTINE                        -
                                ;----------------------------------------------------------------

0114                            K4      PROC    NEAR
0114 43                                 INC     BX                      ; MOVE TO NEXT WORD IN LIST
0115 43                                 INC     BX

0116 3B 1E 0082 R                       CMP     BX,@BUFFER_END          ; AT END OF BUFFER?
011A 72 04                              JB      K5                      ; NO, CONTINUE
011C 8B 1E 0080 R                       MOV     BX,@BUFFER_START        ; YES, RESET TO BUFFER BEGINNING
0120 C3                         K5:     RET
0121                            K4      ENDP


                                ;----- KEYBOARD INTERRUPT ROUTINE

0121                            KB_INT_1 PROC   FAR
0121 50                                 PUSH    AX              ; SAVE THE STI UNTIL AFTER KEYBOARD RESET
0122 53                                 PUSH    BX
0123 51                                 PUSH    CX
0124 52                                 PUSH    DX
0125 56                                 PUSH    SI
0126 57                                 PUSH    DI
0127 1E                                 PUSH    DS
0128 06                                 PUSH    ES
0129 FC                                 CLD                     ; FORWARD DIRECTION
012A E8 0000 E                          CALL    DDS             ; SET UP ADDRESSING TO DATA SEGMENT
012D E4 60                              IN      AL,KB_DATA      ; READ IN THE CHARACTER
012F 93                                 XCHG    BX,AX           ; SAVE IT

                                ;-----  RESET THE SHIFT REGISTER ON THE PLANAR IF ENABLED, OR DO NOTHING IF
                                ;-----  IT IS DISABLED

0130 E4 61                              IN      AL,KB_CTL       ; GET THE CONTROL PORT
0132 8A E0                              MOV     AH,AL           ; SAVE VALUE
0134 0C 80                              OR      AL,80H          ; RESET BIT FOR KEYBOARD
0136 E6 61                              OUT     KB_CTL,AL
0138 86 E0                              XCHG    AH,AL           ; GET BACK ORIGINAL CONTROL
013A E6 61                              OUT     KB_CTL,AL       ; KB HAS BEEN RESET
013C FB                                 STI
013D 93                                 XCHG    AX,BX           ; RESTORE DATA IN

                                ;-----  SYSTEM HOOK  INT 15H - FUNCTION 4FH  (ON HARDWARE INTERRUPT LEVEL 9H)

013E B4 4F                              MOV     AH,04FH         ; SYSTEM INTERCEPT - KEY CODE FUNCTION
0140 F9                                 STC                     ; SET CY= 1 (IN CASE OF IRET)
0141 CD 15                              INT     15H             ; CASSETTE CALL    (AL)= KEY SCAN CODE
                                                                ;  RETURNS CY= 1 FOR INVALID FUNCTION
0143 72 03                              JC      KB_INT_PC       ; CONTINUE IF CARRY FLAG SET ((AL)=CODE)
0145 E9 02CA R                          JMP     K26             ; EXIT IF SYSTEM HANDLED SCAN CODE
                                                                ;  EXIT HANDLES HARDWARE EOI AND ENABLE
0148                            KB_INT_PC:
0148 8A E0                              MOV     AH,AL           ; SAVE SCAN CODE IN AH ALSO

                                ;------ TEST FOR OVERRUN SCAN CODE FROM KEYBOARD

014A 3C FF                              CMP     AL,0FFH                 ; IS THIS AN OVERRUN CHAR
014C 75 03                              JNZ     K16                     ; NO, TEST FOR SHIFT KEY
014E E9 0540 R                          JMP     K62                     ; BUFFER_FULL_BEEP

0151 0E                         K16:    PUSH    CS
0152 07                                 POP     ES                      ; ESTABLISH ADDRESS OF TABLES
0153 8A 3E 0096 R                       MOV     BH,@KB_FLAG_3           ; LOAD FLAGS FOR TESTING

0157                            TEST_E0:
0157 3C E0                              CMP     AL,MC_E0                ; IS THIS THE GENERAL MARKER CODE?
0159 75 07                              JNE     TEST_E1                 ;
015B 80 0E 0096 R 12                    OR      @KB_FLAG_3,LC_E0+KBX    ; SET FLAG BIT, SET KBX, AND
0160 EB 09                              JMP     SHORT EXIT_K            ; THROW AWAY THIS CODE

0162                            TEST_E1:
0162 3C E1                              CMP     AL,MC_E1                ; IS THIS THE PAUSE KEY?
0164 75 08                              JNE     NOT_HC                  ;
0166 80 0E 0096 R 11                    OR      @KB_FLAG_3,LC_E1+KBX    ; SET FLAG, PAUSE KEY MARKER CODE
016B E9 02CF R                  EXIT_K: JMP     K26A                    ; THROW AWAY THIS CODE

016E                            NOT_HC:
016E 24 7F                              AND     AL,07FH                 ; TURN OFF THE BREAK BIT
0170 F6 C7 02                           TEST    BH,LC_E0                ; WAS LAST CODE THE E0 MARKER CODE?
0173 74 0C                              JZ      NOT_LC_E0               ; JUMP IF NOT

0175 B9 0002                            MOV     CX,2                    ; LENGTH OF SEARCH
0178 BF 0555 R                          MOV     DI,OFFSET K6+6          ; IS THIS A SHIFT KEY?
017B F2/ AE                             REPNE   SCASB                   ; CHECK IT
017D 75 54                              JNE     K16A                    ; NO, CONTINUE KEY PROCESSING
017F EB 3D                              JMP     SHORT K16B              ; YES, THROW AWAY & RESET FLAG

0181                            NOT_LC_E0:
0181 F6 C7 01                           TEST    BH,LC_E1                ; WAS LAST CODE THE E1 MARKER CODE?
0184 74 16                              JZ      T_SYS_KEY               ; JUMP IF NOT

0186 B9 0004                            MOV     CX,4                    ; LENGTH OF SEARCH
0189 BF 0553 R                          MOV     DI,OFFSET K6+4          ; IS THIS AN ALT, CTL, OR SHIFT?
018C F2/ AE                             REPNE   SCASB                   ; CHECK IT
018E 74 DB                              JE      EXIT_K                  ; THROW AWAY IF SO

0190 3C 45                              CMP     AL,NUM_KEY              ; IS IT THE PAUSE KEY?
0192 75 2A                              JNE     K16B                    ; NO, THROW AWAY & RESET FLAG
0194 F6 C4 80                           TEST    AH,80H                  ; YES, IS IT THE BREAK OF THE KEY?
0197 75 25                              JNZ     K16B                    ;  YES, THROW THIS AWAY, TOO
0199 E9 03FF R                          JMP     K39P                    ;  NO, THIS IS THE REAL PAUSE STATE
```

```
                                PAGE
                                ;------ TEST FOR SYSTEM KEY

 019C                           T_SYS_KEY:
 019C 3C 54                             CMP     AL,SYS_KEY              ; IS IT THE SYSTEM KEY?
 019E 75 33                             JNE     K16A                    ; CONTINUE IF NOT

 01A0 F6 C4 80                          TEST    AH,080H                 ; CHECK IF THIS A BREAK CODE
 01A3 75 1C                             JNZ     K16C                    ; DONT TOUCH SYSTEM INDICATOR IF TRUE

 01A5 F6 06 0018 R 04                   TEST    @KB_FLAG_1,SYS_SHIFT    ; SEE IF IN SYSTEM KEY HELD DOWN
 01AA 75 12                             JNZ     K16B                    ; IF YES, DONT PROCESS SYSTEM INDICATOR

 01AC 80 0E 0018 R 04                   OR      @KB_FLAG_1,SYS_SHIFT    ; INDICATE SYSTEM KEY DEPRESSED
 01B1 B0 20                             MOV     AL,EOI                  ; END OF INTERRUPT COMMAND
 01B3 E6 20                             OUT     020H,AL                 ; SEND COMMAND TO INTERRUPT CONTROL PORT
                                                                        ; INTERRUPT-RETURN-NO-EOI
 01B5 B8 8500                           MOV     AX,08500H               ; FUNCTION VALUE FOR MAKE OF SYSTEM KEY
 01B8 FB                                STI                             ; MAKE SURE INTERRUPTS ENABLED
 01B9 CD 15                             INT     15H                     ; USER INTERRUPT
 01BB E9 02D4 R                         JMP     K27                     ; END PROCESSING

 01BE E9 02CA R                 K16B:   JMP     K26                     ; IGNORE SYSTEM KEY

 01C1 80 26 0018 R FB           K16C:   AND     @KB_FLAG_1,NOT SYS_SHIFT; TURN OFF SHIFT KEY HELD DOWN
 01C6 B0 20                             MOV     AL,EOI                  ; END OF INTERRUPT COMMAND
 01C8 E6 20                             OUT     020H,AL                 ; SEND COMMAND TO INTERRUPT CONTROL PORT
                                                                        ; INTERRUPT-RETURN-NO-EOI
 01CA B8 8501                           MOV     AX,08501H               ; FUNCTION VALUE FOR BREAK OF SYSTEM KEY
 01CD FB                                STI                             ; MAKE SURE INTERRUPTS ENABLED
 01CE CD 15                             INT     15H                     ; USER INTERRUPT
 01D0 E9 02D4 R                         JMP     K27                     ; IGNORE SYSTEM KEY

                                ;------ TEST FOR SHIFT KEYS

 01D3 8A 1E 0017 R              K16A:   MOV     BL,@KB_FLAG             ; PUT STATE FLAGS IN BL
 01D7 BF 054F R                         MOV     DI,OFFSET K6            ; SHIFT KEY TABLE
 01DA B9 0008 90                        MOV     CX,K6L                  ; LENGTH
 01DE F2/ AE                            REPNE   SCASB                   ; LOOK THROUGH THE TABLE FOR A MATCH
 01E0 8A C4                             MOV     AL,AH                   ; RECOVER SCAN CODE
 01E2 74 03                             JE      K17                     ; JUMP IF MATCH FOUND
 01E4 E9 02B6 R                         JMP     K25                     ; IF NO MATCH, THEN SHIFT NOT FOUND

                                ;------ SHIFT KEY FOUND

 01E7 81 EF 0550 R              K17:    SUB     DI,OFFSET K6+1          ; ADJUST PTR TO SCAN CODE MTCH
 01EB 2E: 8A A5 0557 R                  MOV     AH,CS:K7[DI]            ; GET MASK INTO AH
 01F0 B1 02                             MOV     CL,2                    ; SET UP COUNT FOR FLAG SHIFTS
 01F2 A8 80                             TEST    AL,80H                  ; TEST FOR KEY BREAK
 01F4 74 03                             JZ      K17C
 01F6 EB 6E 90                          JMP     K23                     ; JUMP IF BREAK

                                ;------ SHIFT MAKE FOUND, DETERMINE SET OR TOGGLE

 01F9 80 FC 10                  K17C:   CMP     AH,SCROLL_SHIFT
 01FC 73 21                             JAE     K18                     ; IF SCROLL SHIFT OR ABOVE, TOGGLE KEY

                                ;------ PLAIN SHIFT KEY, SET SHIFT ON

 01FE 08 26 0017 R                      OR      @KB_FLAG,AH             ; TURN ON SHIFT BIT
 0202 F6 C4 0C                          TEST    AH,CTL_SHIFT+ALT_SHIFT  ; IS IT ALT OR CTRL?
 0205 75 03                             JNZ     K17D                    ;  YES, MORE FLAGS TO SET
 0207 E9 02CA R                         JMP     K26                     ;  NO, INTERRUPT_RETURN
 020A F6 C7 02                  K17D:   TEST    BH,LC_E0                ; IS THIS ONE OF THE NEW KEYS?
 020D 74 07                             JZ      K17E                    ;  NO, JUMP
 020F 08 26 0096 R                      OR      @KB_FLAG_3,AH           ; SET BITS FOR RIGHT CTRL, ALT
 0213 E9 02CA R                         JMP     K26                     ; INTERRUPT_RETURN
 0216 D2 EC                     K17E:   SHR     AH,CL                   ; MOVE FLAG BITS TWO POSITIONS
 0218 08 26 0018 R                      OR      @KB_FLAG_1,AH           ; SET BITS FOR LEFT CTRL, ALT
 021C E9 02CA R                         JMP     K26                     ; INTERRUPT_RETURN

                                ;------ TOGGLED SHIFT KEY, TEST FOR 1ST MAKE OR NOT

 021F                           K18:                                    ; SHIFT-TOGGLE
 021F F6 C3 04                          TEST    BL,CTL_SHIFT            ; CHECK CTL SHIFT STATE
 0222 74 03                             JZ      K18A                    ; JUMP IF NOT CTL STATE
 0224 E9 02B6 R                         JMP     K25                     ; JUMP IF CTL STATE
 0227 3C 52                     K18A:   CMP     AL,INS_KEY              ; CHECK FOR INSERT KEY
 0229 75 21                             JNE     K22                     ; JUMP IF NOT INSERT KEY
 022B F6 C3 08                          TEST    BL,ALT_SHIFT            ; CHECK FOR ALTERNATE SHIFT
 022E 74 03                             JZ      K18B                    ; JUMP IF NOT ALTERNATE SHIFT
 0230 E9 02B6 R                         JMP     K25                     ; JUMP IF ALTERNATE SHIFT
 0233 F6 C7 02                  K18B:   TEST    BH,LC_E0                ; IS THIS THE NEW INSERT KEY?
 0236 75 14                             JNZ     K22                     ; YES, THIS ONE'S NEVER A "0"
 0238 F6 C3 20                  K19:    TEST    BL,NUM_STATE            ; CHECK FOR BASE STATE
 023B 75 0A                             JNZ     K21                     ; JUMP IF NUM LOCK IS ON
 023D F6 C3 03                          TEST    BL,LEFT_SHIFT+RIGHT_SHIFT  ; TEST FOR SHIFT STATE
 0240 74 0A                             JZ      K22                        ; JUMP IF BASE STATE
 0242                           K20:
 0242 8A E0                             MOV     AH,AL                   ; PUT SCAN CODE BACK INTO AH
 0244 EB 70 90                          JMP     K25                     ; NUMERAL "0", STNDRD. PROCESSING

 0247 F6 C3 03                  K21:    TEST    BL,LEFT_SHIFT+RIGHT_SHIFT  ; MIGHT BE NUMERIC
 024A 74 F6                             JZ      K20                        ; IS NUMERIC, STD. PROC.

 024C                           K22:                                    ; SHIFT TOGGLE KEY HIT; PROCESS IT
 024C 84 26 0018 R                      TEST    AH,@KB_FLAG_1           ; IS KEY ALREADY DEPRESSED
 0250 74 03                             JZ      K22A
 0252 EB 76 90                          JMP     K26                     ; JUMP IF KEY ALREADY DEPRESSED
 0255 08 26 0018 R              K22A:   OR      @KB_FLAG_1,AH           ; INDICATE THAT THE KEY IS DEPRESSED
 0259 30 26 0017 R                      XOR     @KB_FLAG,AH             ; TOGGLE THE SHIFT STATE
 025D 3C 52                             CMP     AL,INS_KEY              ; TEST FOR 1ST MAKE OF INSERT KEY
 025F 75 69                             JNE     K26                     ; JUMP IF NOT INSERT KEY
 0261 8A E0                             MOV     AH,AL                   ; SCAN CODE IN BOTH HALVES OF AX
 0263 EB 78 90                          JMP     K28                     ; FLAGS UPDATED, PROC. FOR BUFFER

                                ;------ BREAK SHIFT FOUND

 0266                           K23:                                    ; BREAK-SHIFT-FOUND
 0266 80 FC 10                          CMP     AH,SCROLL_SHIFT         ; IS THIS A TOGGLE KEY?
 0269 F6 D4                             NOT     AH                      ; INVERT MASK
 026B 73 43                             JAE     K24                     ; YES, HANDLE BREAK TOGGLE
 026D 20 26 0017 R                      AND     @KB_FLAG,AH             ; TURN OFF SHIFT BIT
 0271 80 FC FB                          CMP     AH,NOT CTL_SHIFT        ; IS THIS ALT OR CTL?
```

```
0274 77 26                      JA      K23D                    ; NO, ALL DONE

0276 F6 C7 02                   TEST    BH,LC_E0                ; 2ND ALT OR CTL?
0279 74 06                      JZ      K23A                    ; NO, HANDLE NORMALLY
027B 20 26 0096 R               AND     @KB_FLAG_3,AH           ; RESET BIT FOR RIGHT ALT OR CTL
027F EB 06                      JMP     SHORT K23B              ; CONTINUE
0281 D2 FC              K23A:   SAR     AH,CL                   ; MOVE THE MASK BIT TWO POSITIONS
0283 20 26 0018 R               AND     @KB_FLAG_1,AH           ; RESET BIT FOR LEFT ALT OR CTL
0287 8A E0              K23B:   MOV     AH,AL                   ; SAVE SCAN CODE
0289 A0 0096 R                  MOV     AL,@KB_FLAG_3           ; GET RIGHT ALT & CTRL FLAGS
028C D2 E8                      SHR     AL,CL                   ; MOVE TO BITS 1 & 0
028E 0A 06 0018 R               OR      AL,@KB_FLAG_1           ; PUT IN LEFT ALT & CTL FLAGS
0292 D2 E0                      SHL     AL,CL                   ; MOVE BACK TO BITS 3 & 2
0294 24 0C                      AND     AL,ALT_SHIFT+CTL_SHIFT  ; FILTER OUT OTHER GARBAGE
0296 08 06 0017 R               OR      @KB_FLAG,AL             ; PUT RESULT IN THE REAL FLAGS
029A 8A C4                      MOV     AL,AH                   ; RECOVER SAVED SCAN CODE

029C 3C B8              K23D:   CMP     AL,ALT_KEY+80H          ; IS THIS ALTERNATE SHIFT RELEASE
029E 75 2A                      JNE     K26                     ; INTERRUPT_RETURN

                        ;------ ALTERNATE SHIFT KEY RELEASED, GET THE VALUE INTO BUFFER

02A0 A0 0019 R                  MOV     AL,@ALT_INPUT
02A3 B4 00                      MOV     AH,0                    ; SCAN CODE OF 0
02A5 88 26 0019 R               MOV     @ALT_INPUT,AH           ; ZERO OUT THE FIELD
02A9 3C 00                      CMP     AL,0                    ; WAS THE INPUT = 0?
02AB 74 1D                      JE      K26                     ; INTERRUPT_RETURN
02AD E9 0519 R                  JMP     K61                     ; IT WASN'T, SO PUT IN BUFFER

02B0                    K24:                                    ; BREAK-TOGGLE
02B0 20 26 0018 R               AND     @KB_FLAG_1,AH           ; INDICATE NO LONGER DEPRESSED
02B4 EB 14                      JMP     SHORT K26               ; INTERRUPT_RETURN

                        ;------ TEST FOR HOLD STATE
                                                                ; AL, AH = SCAN CODE
02B6                    K25:                                    ; NO-SHIFT-FOUND
02B6 3C 80                      CMP     AL,80H                  ; TEST FOR BREAK KEY
02B8 73 10                      JAE     K26                     ; NOTHING FOR BREAK CHARS FROM HERE ON
02BA F6 06 0018 R 08            TEST    @KB_FLAG_1,HOLD_STATE   ; ARE WE IN HOLD STATE
02BF 74 1C                      JZ      K28                     ; BRANCH AROUND TEST IF NOT
02C1 3C 45                      CMP     AL,NUM_KEY
02C3 74 05                      JE      K26                     ; CAN'T END HOLD ON NUM_LOCK
02C5 80 26 0018 R F7            AND     @KB_FLAG_1,NOT HOLD_STATE       ; TURN OFF THE HOLD STATE BIT

02CA                    K26:
02CA 80 26 0096 R FC            AND     @KB_FLAG_3,NOT LC_E0+LC_E1   ; RESET LAST CHAR H.C. FLAG

02CF                    K26A:                                   ; INTERRUPT-RETURN
02CF FA                         CLI                             ; TURN OFF INTERRUPTS
02D0 B0 20                      MOV     AL,EOI                  ; END OF INTERRUPT COMMAND
02D2 E6 20                      OUT     020H,AL                 ; SEND COMMAND TO INTERRUPT CONTROL PORT

02D4                    K27:                                    ; INTERRUPT-RETURN-NO-EOI
02D4 07                         POP     ES                      ; RESTORE REGISTERS
02D5 1F                         POP     DS
02D6 5F                         POP     DI
02D7 5E                         POP     SI
02D8 5A                         POP     DX
02D9 59                         POP     CX
02DA 5B                         POP     BX
02DB 58                         POP     AX
02DC CF                         IRET                            ; RETURN, INTERRUPTS BACK ON
                                                                ;  WITH FLAG CHANGE
```

SECTION 5

```
                                PAGE
                                ;------ NOT IN HOLD STATE
                                                                ; AL, AH = SCAN CODE (ALL MAKES)
 02DD                           K28:                            ; NO-HOLD-STATE
 02DD 3C 58                             CMP     AL,88           ; TEST FOR OUT-OF-RANGE SCAN CODES
 02DF 77 E9                             JA      K26             ; IGNORE IF OUT-OF-RANGE
 02E1 F6 C3 08                          TEST    BL,ALT_SHIFT    ; ARE WE IN ALTERNATE SHIFT?
 02E4 74 0C                             JZ      K28A            ; JUMP IF NOT ALTERNATE
 02E6 F6 C7 10                          TEST    BH,KBX          ; IS THIS THE ENHANCED KEYBOARD?
 02E9 74 0A                             JZ      K29             ; NO, ALT STATE IS REAL
 02EB F6 06 0018 R 04                   TEST    @KB_FLAG_1,SYS_SHIFT    ; YES, IS SYSREQ KEY DOWN?
 02F0 74 03                             JZ      K29             ;  NO, ALT STATE IS REAL
 02F2 E9 03CC R                 K28A:   JMP     K38             ;  YES, THIS IS PHONY ALT STATE
                                                                ;       DUE TO PRESSING SYSREQ
                                ;------ TEST FOR RESET KEY SEQUENCE (CTL ALT DEL)
 02F5                           K29:                            ; TEST-RESET
 02F5 F6 C3 04                          TEST    BL,CTL_SHIFT    ; ARE WE IN CONTROL SHIFT ALSO?
 02F8 74 37                             JZ      K31             ; NO_RESET
 02FA 3C 53                             CMP     AL,DEL_KEY      ; SHIFT STATE IS THERE, TEST KEY
 02FC 75 33                             JNE     K31             ; NO_RESET, IGNORE
                                ;------ CTL-ALT-DEL HAS BEEN FOUND, DO I/O CLEANUP
 02FE C7 06 0072 R 1234                 MOV     @RESET_FLAG,1234H       ; SET FLAG FOR RESET FUNCTION
 0304 81 26 0096 R 0010                 AND     WORD PTR @KB_FLAG_3,KBX ; CLEAR ALL FLAG BITS EXCEPT KBX
 030A E9 0000 E                         JMP     RESET           ; JUMP TO POWER ON DIAGNOSTICS
                                ;------ ALT-INPUT-TABLE
 030D                           K30     LABEL   BYTE
 030D 52 4F 50 51 4B                    DB      82,79,80,81,75
 0312 4C 4D 47 48 49                    DB      76,77,71,72,73  ; 10 NUMBERS ON KEYPAD
                                ;------ SUPER-SHIFT-TABLE
 0317 10 11 12 13 14 15                 DB      16,17,18,19,20,21       ; A-Z TYPEWRITER CHARS
 031D 16 17 18 19 1E 1F                 DB      22,23,24,25,30,31
 0323 20 21 22 23 24 25                 DB      32,33,34,35,36,37
 0329 26 2C 2D 2E 2F 30                 DB      38,44,45,46,47,48
 032F 31 32                             DB      49,50
                                ;------ IN ALTERNATE SHIFT, RESET NOT FOUND
 0331                           K31:                            ; NO-RESET
 0331 3C 39                             CMP     AL,57           ; TEST FOR SPACE KEY
 0333 75 05                             JNE     K311            ; NOT THERE
 0335 B0 20                             MOV     AL,' '          ; SET SPACE CHAR
 0337 E9 050D R                         JMP     K57             ; BUFFER_FILL
 033A                           K311:
 033A 3C 0F                             CMP     AL,15           ; TEST FOR TAB KEY
 033C 75 06                             JNE     K312            ; NOT THERE
 033E B8 A500                           MOV     AX,0A500h       ; SET SPECIAL CODE FOR ALT-TAB
 0341 E9 050D R                         JMP     K57             ; BUFFER_FILL
 0344                           K312:
 0344 3C 4A                             CMP     AL,74           ; TEST FOR KEYPAD -
 0346 74 79                             JE      K37B            ; GO PROCESS
 0348 3C 4E                             CMP     AL,78           ; TEST FOR KEYPAD +
 034A 74 75                             JE      K37B            ; GO PROCESS
                                ;------ LOOK FOR KEY PAD ENTRY
 034C                           K32:                            ; ALT-KEY-PAD
 034C BF 030D R                         MOV     DI,OFFSET K30   ; ALT-INPUT-TABLE
 034F B9 000A                           MOV     CX,10           ; LOOK FOR ENTRY USING KEYPAD
 0352 F2/ AE                            REPNE   SCASB           ; LOOK FOR MATCH
 0354 75 18                             JNE     K33             ; NO_ALT_KEYPAD
 0356 F6 C7 02                          TEST    BH,LC_E0        ; IS THIS ONE OF THE NEW KEYS?
 0359 75 6B                             JNZ     K37C            ;  YES, JUMP, NOT NUMPAD KEY
 035B 81 EF 030E R                      SUB     DI,OFFSET K30+1 ; DI NOW HAS ENTRY VALUE
 035F A0 0019 R                         MOV     AL,@ALT_INPUT   ; GET THE CURRENT BYTE
 0362 B4 0A                             MOV     AH,10           ; MULTIPLY BY 10
 0364 F6 E4                             MUL     AH
 0366 03 C7                             ADD     AX,DI           ; ADD IN THE LATEST ENTRY
 0368 A2 0019 R                         MOV     @ALT_INPUT,AL   ; STORE IT AWAY
 036B E9 02CA R                 K32A:   JMP     K26             ; THROW AWAY THAT KEYSTROKE
                                ;------ LOOK FOR SUPERSHIFT ENTRY
 036E                           K33:                            ; NO-ALT-KEYPAD
 036E C6 06 0019 R 00                   MOV     @ALT_INPUT,0    ; ZERO ANY PREVIOUS ENTRY INTO INPUT
 0373 B9 001A                           MOV     CX,26           ; DI,ES ALREADY POINTING
 0376 F2/ AE                            REPNE   SCASB           ; LOOK FOR MATCH IN ALPHABET
 0378 74 42                             JE      K37A            ; MATCH FOUND, GO FILL THE BUFFER
                                ;------ LOOK FOR TOP ROW OF ALTERNATE SHIFT
 037A                           K34:                            ; ALT-TOP-ROW
 037A 3C 02                             CMP     AL,2            ; KEY WITH '1' ON IT
 037C 72 43                             JB      K37B            ; NOT ONE OF INTERESTING KEYS
 037E 3C 0D                             CMP     AL,13           ; IS IT IN THE REGION
 0380 77 05                             JA      K35             ; ALT-FUNCTION
 0382 80 C4 76                          ADD     AH,118          ; CONVERT PSEUDO SCAN CODE TO RANGE
 0385 EB 35                             JMP     SHORT K37A      ; GO FILL THE BUFFER
                                ;------ TRANSLATE ALTERNATE SHIFT PSEUDO SCAN CODES
 0387                           K35:                            ; ALT-FUNCTION
 0387 3C 57                             CMP     AL,F11_M        ; IS IT F11?
 0389 72 09                             JB      K35A            ;  NO, BRANCH
 038B 3C 58                             CMP     AL,F12_M        ; IS IT F12?
 038D 77 05                             JA      K35A            ;  NO, BRANCH
 038F 80 C4 34                          ADD     AH,52           ; CONVERT TO PSEUDO SCAN CODE
 0392 EB 28                             JMP     SHORT K37A      ; GO FILL THE BUFFER
 0394 F6 C7 02                  K35A:   TEST    BH,LC_E0        ; DO WE HAVE ONE OF THE NEW KEYS?
 0397 74 18                             JZ      K37             ;  NO, JUMP
 0399 3C 1C                             CMP     AL,28           ; TEST FOR KEYPAD ENTER
 039B 75 06                             JNE     K35B            ; NOT THERE
 039D B8 A600                           MOV     AX,0A600h       ; SPECIAL CODE
 03A0 E9 050D R                         JMP     K57             ; BUFFER FILL
 03A3 3C 53                     K35B:   CMP     AL,83           ; TEST FOR DELETE KEY
 03A5 74 1F                             JE      K37C            ; HANDLE WITH OTHER EDIT KEYS
```

```
03A7 3C 35                      CMP     AL,53                   ; TEST FOR KEYPAD /
03A9 75 C0                      JNE     K32A                    ; NOT THERE, NO OTHER E0 SPECIALS
03AB B8 A400                    MOV     AX,0A400h               ; SPECIAL CODE
03AE E9 050D R                  JMP     K57                     ; BUFFER FILL
03B1 3C 3B              K37:    CMP     AL,59                   ; TEST FOR IN TABLE
03B3 72 0C                      JB      K37B                    ; ALT-CONTINUE
03B5 3C 44                      CMP     AL,68                   ; IN KEYPAD REGION?
                                                                ; OR NUMLOCK, SCROLLOCK?
03B7 77 B2                      JA      K32A                    ; IF SO, IGNORE
03B9 80 C4 2D                   ADD     AH,45                   ; CONVERT TO PSEUDO SCAN CODE
03BC B0 00              K37A:   MOV     AL,0                    ; ASCII CODE OF ZERO
03BE E9 050D R                  JMP     K57                     ; PUT IT IN THE BUFFER
03C1 B0 F0              K37B:   MOV     AL,0F0h                 ; USE SPECIAL ASCII CODE
03C3 E9 050D R                  JMP     K57                     ; PUT IT IN THE BUFFER
03C6 04 50              K37C:   ADD     AL,80                   ; CONVERT SCAN CODE (EDIT KEYS)
03C8 8A E0                      MOV     AH,AL                   ; (SCAN CODE NOT IN AH FOR INSERT)
03CA EB F0                      JMP     K37A                    ; PUT IT IN THE BUFFER
                        ;------ NOT IN ALTERNATE SHIFT
03CC                    K38:                                    ; NOT-ALT-SHIFT
                                                                ; BL STILL HAS SHIFT FLAGS
03CC F6 C3 04                   TEST    BL,CTL_SHIFT            ; ARE WE IN CONTROL SHIFT?
03CF 75 03                      JNZ     K38A                    ;  YES, START PROCESSING
03D1 E9 0454 R                  JMP     K44                     ; NOT-CTL-SHIFT
                        ;------ CONTROL SHIFT, TEST SPECIAL CHARACTERS
                        ;------ TEST FOR BREAK
03D4 3C 46              K38A:   CMP     AL,SCROLL_KEY           ; TEST FOR BREAK
03D6 75 1E                      JNE     K39                     ; JUMP, NO-BREAK
03D8 F6 C7 10                   TEST    BH,KBX                  ; IS THIS THE ENHANCED KEYBOARD?
03DB 74 05                      JZ      K38B                    ;  NO, BREAK IS VALID
03DD F6 C7 02                   TEST    BH,LC_E0                ;  YES, WAS LAST CODE AN E0?
03E0 74 14                      JZ      K39                     ;   NO-BREAK, TEST FOR PAUSE
03E2 8B 1E 001A R       K38B:   MOV     BX,@BUFFER_HEAD         ; RESET BUFFER TO EMPTY
03E6 89 1E 001C R               MOV     @BUFFER_TAIL,BX         ;
03EA C6 06 0071 R 80            MOV     @BIOS_BREAK,80H         ; TURN ON BIOS_BREAK BIT
03EF CD 1B                      INT     1BH                     ; BREAK INTERRUPT VECTOR
03F1 2B C0                      SUB     AX,AX                   ; PUT OUT DUMMY CHARACTER
03F3 E9 050D R                  JMP     K57                     ; BUFFER_FILL
                        ;-------- TEST FOR PAUSE
03F6                    K39:                                    ; NO-BREAK
03F6 F6 C7 10                   TEST    BH,KBX                  ; IS THIS THE ENHANCED KEYBOARD?
03F9 75 25                      JNZ     K41                     ; YES, THEN THIS CAN'T BE PAUSE
03FB 3C 45                      CMP     AL,NUM_KEY              ; LOOK FOR PAUSE KEY
03FD 75 21                      JNE     K41                     ; NO-PAUSE
03FF 80 0E 0018 R 08    K39P:   OR      @KB_FLAG_1,HOLD_STATE   ; TURN ON THE HOLD FLAG
0404 B0 20                      MOV     AL,EOI                  ; END OF INTERRUPT TO CONTROL PORT
0406 E6 20                      OUT     020H,AL                 ; ALLOW FURTHER KEYSTROKE INTS
                        ;------ DURING PAUSE INTERVAL, TURN CRT BACK ON
0408 80 3E 0049 R 07            CMP     @CRT_MODE,7             ; IS THIS BLACK AND WHITE CARD
040D 74 07                      JE      K40                     ; YES, NOTHING TO DO
040F BA 03D8                    MOV     DX,03D8H                ; PORT FOR COLOR CARD
0412 A0 0065 R                  MOV     AL,@CRT_MODE_SET        ; GET THE VALUE OF THE CURRENT MODE
0415 EE                         OUT     DX,AL                   ; SET THE CRT MODE, SO THAT CRT IS ON
0416                    K40:                                    ; PAUSE-LOOP
0416 F6 06 0018 R 08            TEST    @KB_FLAG_1,HOLD_STATE
041B 75 F9                      JNZ     K40                     ; LOOP UNTIL FLAG TURNED OFF
041D E9 02D4 R                  JMP     K27                     ; INTERRUPT_RETURN_NO_EOI
                        ;------ TEST SPECIAL CASE KEY 55
0420                    K41:                                    ; NO-PAUSE
0420 3C 37                      CMP     AL,55                   ; TEST FOR */PRTSC KEY
0422 75 10                      JNE     K42                     ; NOT-KEY-55
0424 F6 C7 10                   TEST    BH,KBX                  ; IS THIS THE ENHANCED KEYBOARD?
0427 74 05                      JZ      K41A                    ;  NO, CTL-PRTSC IS VALID
0429 F6 C7 02                   TEST    BH,LC_E0                ;  YES, WAS LAST CODE AN E0?
042C 74 20                      JZ      K42B                    ;   NO, TRANSLATE TO A FUNCTION
042E B8 7200            K41A:   MOV     AX,114*256              ; START/STOP PRINTING SWITCH
0431 E9 050D R                  JMP     K57                     ; BUFFER_FILL
                        ;------ SET UP TO TRANSLATE CONTROL SHIFT
0434                    K42:                                    ; NOT-KEY-55
0434 3C 0F                      CMP     AL,15                   ; IS IT THE TAB KEY?
0436 74 16                      JE      K42B                    ;  YES, XLATE TO FUNCTION CODE
0438 3C 35                      CMP     AL,53                   ; IS IT THE / KEY?
043A 75 0B                      JNE     K42A                    ;  NO, NO MORE SPECIAL CASES
043C F6 C7 02                   TEST    BH,LC_E0                ;  YES, IS IT FROM THE KEYPAD?
043F 74 06                      JZ      K42A                    ;   NO, JUST TRANSLATE
0441 B8 9500                    MOV     AX,9500h                ;   YES, SPECIAL CODE FOR THIS ONE
0444 E9 050D R                  JMP     K57                     ;   BUFFER FILL
0447 BB 055F R          K42A:   MOV     BX,OFFSET K8            ; SET UP TO TRANSLATE CTL
044A 3C 3B                      CMP     AL,59                   ; IS IT IN CHARACTER TABLE?
044C 72 57                      JB      K45F                    ;  YES, GO TRANSLATE CHAR
044E BB 055F R          K42B:   MOV     BX,OFFSET K8            ; SET UP TO TRANSLATE CTL
0451 E9 04FC R                  JMP     K64                     ;  NO, GO TRANSLATE_SCAN
                        ;------ NOT IN CONTROL SHIFT
0454 3C 37              K44:    CMP     AL,55                   ; PRINT SCREEN KEY?
0456 75 1F                      JNE     K45                     ; NOT-PRINT-SCREEN
0458 F6 C7 10                   TEST    BH,KBX                  ; IS THIS ENHANCED KEYBOARD?
045B 74 07                      JZ      K44A                    ; NO, TEST FOR SHIFT STATE
045D F6 C7 02                   TEST    BH,LC_E0                ; YES, LAST CODE A MARKER?
0460 75 07                      JNZ     K44B                    ;  YES, IS PRINT SCREEN
0462 EB 34                      JMP     SHORT K45C              ;  NO, XLATE TO "*" CHARACTER
0464 F6 C3 03           K44A:   TEST    BL,LEFT_SHIFT+RIGHT_SHIFT ;NOT 101 KBD, SHIFT KEY DOWN?
0467 74 2F                      JZ      K45C                    ; NO, XLATE TO "*" CHARACTER
                        ;------ ISSUE INTERRUPT TO PERFORM PRINT SCREEN FUNCTION
```

SECTION 5

```
0469 B0 20              K44B:   MOV     AL,EOI                  ; END OF CURRENT INTERRUPT
046B E6 20                      OUT     020H,AL                 ;  SO FURTHER THINGS CAN HAPPEN
046D CD 05                      INT     5H                      ; ISSUE PRINT SCREEN INTERRUPT
046F 80 26 0096 R FC            AND     @KB_FLAG_3,NOT LC_E0+LC_E1;ZERO OUT THESE FLAGS
0474 E9 02D4 R                  JMP     K27                     ; GO BACK WITHOUT EOI OCCURRING


                        ;------ HANDLE THE IN-CORE KEYS
0477                    K45:                                    ; NOT-PRINT-SCREEN
0477 3C 3A                      CMP     AL,58                   ; TEST FOR IN-CORE AREA
0479 77 2C                      JA      K46                     ; JUMP IF NOT

047B 3C 35                      CMP     AL,53                   ; IS THIS THE "/" KEY?
047D 75 05                      JNE     K45A                    ;  NO, JUMP
047F F6 C7 02                   TEST    BH,LC_E0                ; WAS LAST CODE THE MARKER?
0482 75 14                      JNZ     K45C                    ;  YES, TRANSLATE TO CHARACTER

0484 B9 001A            K45A:   MOV     CX,26                   ; LENGTH OF SEARCH
0487 BF 0317 R                  MOV     DI,OFFSET K30+10        ; POINT TO TABLE OF A-Z CHARS
048A F2/ AE                     REPNE   SCASB                   ; IS THIS A LETTER KEY?
048C 75 05                      JNE     K45B                    ;  NO, SYMBOL KEY

048E F6 C3 40                   TEST    BL,CAPS_STATE           ; ARE WE IN CAPS_LOCK?
0491 75 0A                      JNZ     K45D                    ; TEST FOR SURE
0493 F6 C3 03           K45B:   TEST    BL,LEFT_SHIFT+RIGHT_SHIFT ; ARE WE IN SHIFT STATE?
0496 75 0A                      JNZ     K45E                    ; YES, UPPERCASE
                                                                ; NO, LOWERCASE
0498 BB 05B7 R          K45C:   MOV     BX,OFFSET K10           ; TRANSLATE TO LOWERCASE LETTERS
049B EB 50                      JMP     SHORT K56
049D                    K45D:                                   ; ALMOST-CAPS-STATE
049D F6 C3 03                   TEST    BL,LEFT_SHIFT+RIGHT_SHIFT ; CL ON. IS SHIFT ON, TOO?
04A0 75 F6                      JNZ     K45C                    ; SHIFTED TEMP OUT OF CAPS STATE
04A2 BB 060F R          K45E:   MOV     BX,OFFSET K11           ; TRANSLATE TO UPPERCASE LETTERS
04A5 EB 46              K45F:   JMP     SHORT K56


                        ;------ TEST FOR KEYS F1 - F10
04A7                    K46:                                    ; NOT IN-CORE AREA
04A7 3C 44                      CMP     AL,68                   ; TEST FOR F1 - F10
04A9 77 02                      JA      K47                     ; JUMP IF NOT
04AB EB 36                      JMP     SHORT K53               ; YES, GO DO FN KEY PROCESS


                        ;------ HANDLE THE NUMERIC PAD KEYS

04AD                    K47:                                    ; NOT F1 - F10
04AD 3C 53                      CMP     AL,83                   ; TEST FOR NUMPAD KEYS
04AF 77 2C                      JA      K52                     ; JUMP IF NOT

                        ;------ KEYPAD KEYS, MUST TEST NUM LOCK FOR DETERMINATION
04B1 3C 4A              K48:    CMP     AL,74                   ; SPECIAL CASE FOR MINUS
04B3 74 ED                      JE      K45E                    ; GO TRANSLATE
04B5 3C 4E                      CMP     AL,78                   ; SPECIAL CASE FOR PLUS
04B7 74 E9                      JE      K45E                    ; GO TRANSLATE
04B9 F6 C7 02                   TEST    BH,LC_E0                ; IS THIS ONE OF THE NEW KEYS?
04BC 75 0A                      JNZ     K49                     ;  YES, TRANSLATE TO BASE STATE

04BE F6 C3 20                   TEST    BL,NUM_STATE            ; ARE WE IN NUM_LOCK?
04C1 75 13                      JNZ     K50                     ; TEST FOR SURE
04C3 F6 C3 03                   TEST    BL,LEFT_SHIFT+RIGHT_SHIFT  ; ARE WE IN SHIFT STATE?
04C6 75 13                      JNZ     K51                        ; IF SHIFTED, REALLY NUM STATE

                        ;------ BASE CASE FOR KEYPAD
04C8 3C 4C              K49:    CMP     AL,76                   ; SPECIAL CASE FOR BASE STATE 5
04CA 75 05                      JNE     K49A                    ; CONTINUE IF NOT KEYPAD 5
04CC B0 F0                      MOV     AL,0F0h                 ; SPECIAL ASCII CODE
04CE EB 3D 90                   JMP     K57                     ; BUFFER FILL
04D1 BB 05B7 R          K49A:   MOV     BX,OFFSET K10           ; BASE CASE TABLE
04D4 EB 26                      JMP     SHORT K64               ; CONVERT TO PSEUDO SCAN

                        ;------ MIGHT BE NUM LOCK, TEST SHIFT STATUS
04D6 F6 C3 03           K50:    TEST    BL,LEFT_SHIFT+RIGHT_SHIFT        ;ALMOST-NUM-STATE
04D9 75 ED                      JNZ     K49                     ; SHIFTED TEMP OUT OF NUM STATE
04DB EB C5              K51:    JMP     SHORT K45E              ; REALLY_NUM_STATE


                        ;------ TEST FOR THE NEW KEY ON WT KEYBOARDS

04DD                    K52:                                    ; NOT A NUMPAD KEY
04DD 3C 56                      CMP     AL,86                   ; IS IT THE NEW WT KEY?
04DF 75 02                      JNE     K53                     ; JUMP IF NOT
04E1 EB B0                      JMP     SHORT K45B              ; HANDLE WITH REST OF LETTER KEYS


                        ;------ MUST BE F11 OR F12
                                                                ; F1 - F10 COME HERE, TOO
04E3 F6 C3 03           K53:    TEST    BL,LEFT_SHIFT+RIGHT_SHIFT ;TEST SHIFT STATE
04E6 74 E0                      JZ      K49                     ; JUMP, LOWERCASE PSEUDO SC'S

04E8 BB 060F R                  MOV     BX,OFFSET K11           ; UPPER CASE PSEUDO SCAN CODES
04EB EB 0F                      JMP     SHORT K64               ; TRANSLATE_SCAN

                        ;------ TRANSLATE THE CHARACTER

04ED                    K56:                                    ; TRANSLATE-CHAR
04ED FE C8                      DEC     AL                      ; CONVERT ORIGIN
04EF 2E: D7                     XLAT    CS:K11                  ; CONVERT THE SCAN CODE TO ASCII
04F1 F6 06 0096 R 02            TEST    @KB_FLAG_3,LC_E0        ; IS THIS A NEW KEY?
04F6 74 15                      JZ      K57                     ;  NO, GO FILL BUFFER
04F8 B4 E0                      MOV     AH,MC_E0                ;  YES, PUT SPECIAL MARKER IN AH
04FA EB 11                      JMP     SHORT K57               ; PUT IT INTO THE BUFFER

                        ;------ TRANSLATE SCAN FOR PSEUDO SCAN CODES

04FC                    K64:                                    ; TRANSLATE-SCAN-ORGD
04FC FE C8                      DEC     AL                      ; CONVERT ORIGIN
04FE 2E: D7                     XLAT    CS:K8                   ; CTL TABLE SCAN
0500 8A E0                      MOV     AH,AL                   ; PUT VALUE INTO AH
0502 B0 00                      MOV     AL,0                    ; ZERO ASCII CODE
0504 F6 06 0096 R 02            TEST    @KB_FLAG_3,LC_E0        ; IS THIS A NEW KEY?
0509 74 02                      JZ      K57                     ;  NO, GO FILL BUFFER
050B B0 E0                      MOV     AL,MC_E0                ;  YES, PUT SPECIAL MARKER IN AL

                        ;------ PUT CHARACTER INTO BUFFER
```

```

0500D                 K57:                                          ; BUFFER-FILL
050D 3C FF                     CMP     AL,-1                        ; IS THIS AN IGNORE CHAR
050F 74 05                     JE      K59                          ; YES, DO NOTHING WITH IT
0511 80 FC FF                  CMP     AH,-1                        ; LOOK FOR -1 PSEUDO SCAN
0514 75 03                     JNE     K61                          ; NEAR_INTERRUPT_RETURN

0516                  K59:                                          ; NEAR-INTERRUPT-RETURN
0516 E9 02CA R                 JMP     K26                          ; INTERRUPT_RETURN

0519                  K61:
0519 FA                        CLI                                  ; TURN OFF INTERRUPTS
051A 8B 1E 001C R              MOV     BX,@BUFFER_TAIL              ; GET THE END POINTER TO THE BUFFER
051E 8B F3                     MOV     SI,BX                        ; SAVE THE VALUE
0520 E8 0114 R                 CALL    K4                           ; ADVANCE THE TAIL
0523 3B 1E 001A R              CMP     BX,@BUFFER_HEAD              ; HAS THE BUFFER WRAPPED AROUND
0527 74 17                     JE      K62                          ; BUFFER_FULL_BEEP
0529 89 04                     MOV     [SI],AX                      ; STORE THE VALUE
052B 89 1E 001C R              MOV     @BUFFER_TAIL,BX              ; MOVE THE POINTER UP
052F B0 20                     MOV     AL,EOI                       ; END OF INTERRUPT COMMAND
0531 E6 20                     OUT     020H,AL                      ; SEND COMMAND TO INTERRUPT CONTROL PORT
0533 B8 9102                   MOV     AX,09102H                    ; MOVE IN POST CODE & TYPE
0536 CD 15                     INT     15H                          ; PERFORM OTHER FUNCTION
0538 80 26 0096 R FC           AND     @KB_FLAG_3,NOT LC_E0+LC_E1     ; RESET LAST CHAR H.C. FLAG
053D E9 02D4 R                 JMP     K27                          ; INTERRUPT_RETURN

                      ;------ BUFFER IS FULL SOUND THE BEEPER

0540                  K62:
0540 B0 20                     MOV     AL,EOI                       ; ENABLE INTERRUPT CONTROLLER CHIP
0542 E6 20                     OUT     INTA00,AL                    ;
0544 B9 02A6                   MOV     CX,678                       ; DIVISOR FOR 1760 HZ
0547 B3 04                     MOV     BL,4                         ; SHORT BEEP COUNT (1/16 + 1/64 DELAY)
0549 E8 0000 E                 CALL    BEEP                         ; GO TO COMMON BEEP HANDLER
054C E9 02D4 R                 JMP     K27                          ; EXIT

054F                  KB_INT_1  ENDP
```

SECTION 5

```
                                PAGE
                                ;---------------------------------------------------------------
                                ;          KEY IDENTIFICATION SCAN TABLES                       -
                                ;---------------------------------------------------------------
                                ;-------------- TABLE OF SHIFT KEYS AND MASK VALUES ---------------
                                ;------ KEY_TABLE
 054F                           K6      LABEL   BYTE
 054F 52                                DB      INS_KEY                 ; INSERT KEY
 0550 3A 45 46 38 1D                    DB      CAPS_KEY,NUM_KEY,SCROLL_KEY,ALT_KEY,CTL_KEY
 0555 2A 36                             DB      LEFT_KEY,RIGHT_KEY
 = 0008                         K6L     EQU     $-K6

                                ;------ MASK_TABLE
 0557                           K7      LABEL   BYTE
 0557 80                                DB      INS_SHIFT               ; INSERT MODE SHIFT
 0558 40 20 10 08 04                    DB      CAPS_SHIFT,NUM_SHIFT,SCROLL_SHIFT,ALT_SHIFT,CTL_SHIFT
 055D 02 01                             DB      LEFT_SHIFT,RIGHT_SHIFT

                                ;---------------------- TABLES FOR CTRL CASE ----------------------

 055F                           K8      LABEL   BYTE                    ;-------- CHARACTERS ---------
 055F 1B FF 00 FF FF FF                 DB      27,-1,00,-1,-1,-1       ; Esc, 1, 2, 3, 4, 5
 0565 1E FF FF FF FF 1F                 DB      30,-1,-1,-1,-1,31       ; 6, 7, 8, 9, 0, -
 056B FF 7F 94 11 17 05                 DB      -1,127,148,17,23,5      ; =, Bksp, Tab, Q, W, E
 0571 12 14 19 15 09 0F                 DB      18,20,25,21,09,15       ; R, T, Y, U, I, O
 0577 10 1B 1D 0A FF 01                 DB      16,27,29,10,-1,01       ; P, [,], Enter, Ctrl, A
 057D 13 04 06 07 08 0A                 DB      19,04,06,07,08,10       ; S, D, F, G, H, J
 0583 0B 0C FF FF FF FF                 DB      11,12,-1,-1,-1,-1       ; K, L, ;, ', `, LShift
 0589 1C 1A 18 03 16 02                 DB      28,26,24,03,22,02       ; \, Z, X, C, V, B
 058F 0E 0D FF FF FF FF                 DB      14,13,-1,-1,-1,-1       ; N, M, ,, ., /, RShift
 0595 96 FF 20 FF                       DB      150,-1,' ',-1           ; *, Alt, Space, CL
                                                                        ;--------- FUNCTIONS ---------
 0599 5E 5F 60 61 62 63                 DB      94,95,96,97,98,99       ; F1 - F6
 059F 64 65 66 67 FF FF                 DB      100,101,102,103,-1,-1   ; F7 - F10, NL, SL
 05A5 77 8D 84 8E 73 8F                 DB      119,141,132,142,115,143 ; Home, Up, PgUp, -, Left, Pad5
 05AB 74 90 75 91 76 92                 DB      116,144,117,145,118,146 ; Right, +, End, Down, PgDn, Ins
 05B1 93 FF FF FF 89 8A                 DB      147,-1,-1,-1,137,138    ; Del, SysReq, Undef, WT, F11, F12
                                ;---------------------- TABLES FOR LOWER CASE ----------------------

 05B7                           K10     LABEL   BYTE
 05B7 1B 31 32 33 34 35                 DB      27,'12345'
 05BD 36 37 38 39 30 2D                 DB      '67890-'
 05C3 3D 08 09 71 77 65                 DB      '=',08,09,'qwe'
 05C9 72 74 79 75 69 6F                 DB      'rtyuio'
 05CF 70 5B 5D 0D FF 61                 DB      'p[]',0DH,-1,'a'        ; LETTERS, Return, Ctrl
 05D5 73 64 66 67 68 6A                 DB      'sdfghj'
 05DB 6B 6C 3B 27 60 FF                 DB      "kl;'`",-1              ; LETTERS, L Shift
 05E1 5C 7A 78 63 76 62                 DB      '\zxcvb'
 05E7 6E 6D 2C 2E 2F                    DB      'nm,./'
 05EC FF 2A FF 20 FF                    DB      -1,'*',-1,' ',-1        ; R Shift,*, Alt, Space, CL

                                ;------ LC TABLE SCAN
 05F1 3B 3C 3D 3E 3F                    DB      59,60,61,62,63          ; BASE STATE OF F1 - F10
 05F6 40 41 42 43 44                    DB      64,65,66,67,68
 05FB FF FF                             DB      -1,-1                   ; NL, SL

                                ;------ KEYPAD TABLE
 05FD                           K15     LABEL   BYTE
 05FD 47 48 49 FF 4B FF                 DB      71,72,73,-1,75,-1       ; BASE STATE OF KEYPAD KEYS
 0603 4D FF 4F 50 51 52                 DB      77,-1,79,80,81,82
 0609 53                                DB      83
 060A FF FF 5C 85 86                    DB      -1,-1,'\',133,134       ; SysRq, Undef, WT, F11, F12

                                ;---------------------- TABLES FOR UPPER CASE ----------------------

 060F                           K11     LABEL   BYTE
 060F 1B 21 40 23 24 25                 DB      27,'!@#$%'
 0615 5E 26 2A 28 29 5F                 DB      '^&*()_'
 061B 2B 08 00 51 57 45                 DB      '+',08,00,'QWE'
 0621 52 54 59 55 49 4F                 DB      'RTYUIO'
 0627 50 7B 7D 0D FF 41                 DB      'P{}',0DH,-1,'A'        ; LETTERS, Return, Ctrl
 062D 53 44 46 47 48 4A                 DB      'SDFGHJ'
 0633 4B 4C 3A 22 7E FF                 DB      'KL:"~',-1              ; LETTERS, L Shift
 0639 7C 5A 58 43 56 42                 DB      '|ZXCVB'
 063F 4E 4D 3C 3E 3F                    DB      'NM<>?'
 0644 FF 2A FF 20 FF                    DB      -1,'*',-1,' ',-1        ; R Shift,*, Alt, Space, CL

                                ;------ UC TABLE SCAN
 0649                           K12     LABEL   BYTE
 0649 54 55 56 57 58                    DB      84,85,86,87,88          ; SHIFTED STATE OF F1 - F10
 064E 59 5A 5B 5C 5D                    DB      89,90,91,92,93
 0653 FF FF                             DB      -1,-1                   ; NL, SL

                                ;------ NUM STATE TABLE
 0655                           K14     LABEL   BYTE
 0655 37 38 39 2D 34 35                 DB      '789-456+1230.'         ; NUMLOCK STATE OF KEYPAD KEYS
      36 2B 31 32 33 30
      2E
 0662 FF FF 7C 87 88                    DB      -1,-1,'|',135,136       ; SysRq, Undef, WT, F11, F12
 0667                           CODE    ENDS
                                END
```

```
                                PAGE 118,121
                                TITLE PRT ------ 01/10/86  PRINTER ADAPTER BIOS
                                .LIST

 0000                           CODE    SEGMENT BYTE PUBLIC

                                        PUBLIC  PRINTER_IO_1

                                        EXTRN   DDS:NEAR

                                ;--- INT  17 H -----------------------------------------------------------------
                                ; PRINTER_IO                                                                  :
                                ;       THIS ROUTINE PROVIDES COMMUNICATION WITH THE PRINTER                  :
                                ; INPUT                                                                       :
                                ;       (AH)= 00H  PRINT THE CHARACTER IN (AL)                                :
                                ;                   ON RETURN, (AH)= 1 IF CHARACTER NOT PRINTED (TIME OUT)    :
                                ;                   OTHER BITS SET AS ON NORMAL STATUS CALL                   :
                                ;       (AH)= 01H  INITIALIZE THE PRINTER PORT                                :
                                ;                   RETURNS WITH (AH) SET WITH PRINTER STATUS                 :
                                ;       (AH)= 02H  READ THE PRINTER STATUS INTO (AH)                          :
                                ;                   7      6      5      4      3      2-1    0               :
                                ;                   |      |      |      |      |      |      |_ 1 = 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                                                  :
                                ;                                                                             :
                                ; REGISTERS     (AH) IS MODIFIED WITH STATUS INFORMATION                      :
                                ;               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  52                               PUSH    DX                      ; SAVE WORK REGISTERS
 0002  53                               PUSH    BX
 0003  83 FA 03                         CMP     DX,03H                  ; CHECK FOR PRINTER NUMBER VALID 0-3
 0006  77 25                            JA      B10                     ; ERROR EXIT IF OUT OF RANGE

 0008  8A F8                            MOV     BH,AL                   ; SAVE CHARACTER TO BE PRINTED
 000A  1E                               PUSH    DS                      ; SAVE SEGMENT
 000B  E8 0000 E                        CALL    DDS                     ; ADDRESS DATA SEGMENT

 000E  56                               PUSH    SI                      ; SAVE WORK POINTER REGISTER
 000F  8B F2                            MOV     SI,DX                   ; GET PRINTER PARAMETER
 0011  8A 9C 0078 R                     MOV     BL,@PRINT_TIM_OUT[SI]   ; LOAD TIMEOUT VALUE
 0015  D1 E6                            SHL     SI,1                    ; WORD OFFSET INTO TABLE INTO (SI)
 0017  8B 94 0008 R                     MOV     DX,@PRINTER_BASE[SI]    ; GET BASE ADDRESS FOR PRINTER CARD
 001B  5E                               POP     SI                      ; RECOVER CALLERS (SI) REGISTER
 001C  1F                               POP     DS                      ;  AND (DS) SEGMENT REGISTER
                                        ASSUME  DS:NOTHING
 001D  0B D2                            OR      DX,DX                   ; TEST DX = ZERO, INDICATING NO PRINTER
 001F  74 0C                            JZ      B10                     ; EXIT,  NO PRINTER ADAPTER AT OFFSET

 0021  0A E4                            OR      AH,AH                   ; TEST FOR (AH)= 00H
 0023  74 0D                            JZ      B30                     ;  PRINT CHARACTER IN (AL)

 0025  FE CC                            DEC     AH                      ; TEST FOR (AH)= 01H
 0027  74 4B                            JZ      B80                     ;  INITIALIZE PRINTER

 0029  FE CC                            DEC     AH                      ; TEST FOR (AH)= 02H
 002B  74 39                            JZ      B60                     ;  GET PRINTER STATUS

 002D                           B10:
 002D  B4 29                            MOV     AH,029H                 ; RETURN ERROR BITS FOR INVALID CALLS

 002F                           B20:
 002F  5B                               POP     BX                      ; RETURN
 0030  5A                               POP     DX                      ; RECOVER REGISTERS
 0031  CF                               IRET                            ; RETURN TO CALLING PROGRAM (AH)= STATUS


                                ;-----  PRINT THE CHARACTER IN (AL)

 0032                           B30:
 0032  EE                               OUT     DX,AL                   ; OUTPUT CHARACTER TO DATA PORT
 0033  42                               INC     DX                      ; POINT TO STATUS PORT
```

SECTION 5

```
                          PAGE
                          ;-----  CHECK FOR PRINTER BUSY

0034 EC                           IN      AL,DX          ; PRE-CHARGE +BUSY LINE IF FLOATING
0035 EC                           IN      AL,DX          ; GET STATUS PORT VALUE
0036 A8 80                        TEST    AL,80H         ; IS THE PRINTER CURRENTLY BUSY
0038 75 05                        JNZ     B40            ; SKIP SYSTEM DEVICE BUSY CALL IF NOT

                          ;-----  INT 15 H -- DEVICE BUSY

003A B8 90FE                      MOV     AX,90FEH       ; FUNCTION 90 PRINTER ID
003D CD 15                        INT     15H            ; SYSTEM CALL

                          ;-----  WAIT BUSY

003F                      B40:
003F 51                           PUSH    CX             ; SAVE CALLERS (CX) REGISTER
0040 2B C9                        SUB     CX,CX          ; INNER LOOP (64K)
0042                      B45:
0042 EC                           IN      AL,DX          ; GET STATUS
0043 8A E0                        MOV     AH,AL          ; STATUS TO (AH) ALSO
0045 A8 80                        TEST    AL,80H         ; IS THE PRINTER CURRENTLY BUSY
0047 75 0F                        JNZ     B50            ; GO TO OUTPUT STROBE

0049 E2 F7                        LOOP    B45            ; LOOP IF NOT

004B FE CB                        DEC     BL             ; DECREMENT OUTER LOOP COUNT
004D 75 F3                        JNZ     B45            ; MAKE ANOTHER PASS IF NOT ZERO

004F 59                           POP     CX             ; RESTORE (CX) WITH CALLERS VALUE
0050 80 CC 01                     OR      AH,1           ; SET ERROR FLAG
0053 80 E4 F9                     AND     AH,0F9H        ; TURN OFF THE UNUSED BITS
0056 EB 15                        JMP     SHORT B70      ; RETURN WITH ERROR FLAG SET

0058                      B50:                           ;       SEND STROBE PULSE
0058 59                           POP     CX             ; RESTORE (CX) WITH CALLERS VALUE
0059 B0 0D                        MOV     AL,0DH         ; SET THE STROBE LOW (BIT ON)
005B 42                           INC     DX             ; OUTPUT STROBE TO CONTROL PORT
005C FA                           CLI                    ; PREVENT INTERRUPT PULSE STRETCHING
005D EE                           OUT     DX,AL          ; OUTPUT STROBE BIT   > 1us  < 5us
005E EB 00                        JMP     $+2            ; I/O DELAY TO ALLOW FOR LINE LOADING
                                                         ;  AND FOR CORRECT PULSE WIDTH
0060 B0 0C                        MOV     AL,0CH         ; SET THE -STROBE HIGH
0062 EE                           OUT     DX,AL
0063 FB                           STI                    ; INTERRUPTS BACK ON
0064 4A                           DEC     DX             ; ADJUST BACK TO BASE ADDRESS
0065 4A                           DEC     DX             ;  FOR STATUS ROUTINE EXIT


                          ;-----  PRINTER STATUS

0066                      B60:
0066 42                           INC     DX             ; POINT TO CONTROL PORT
0067 EC                           IN      AL,DX          ; PRE-CHARGE +BUSY LINE IF FLOATING
0068 EC                           IN      AL,DX          ; GET PRINTER STATUS HARDWARE BITS
0069 24 F8                        AND     AL,0F8H        ; TURN OFF UNUSED BITS
006B 8A E0                        MOV     AH,AL          ; SAVE
006D                      B70:
006D 8A C7                        MOV     AL,BH          ; RECOVER CHARACTER INTO (AL) REGISTER
006F 80 F4 48                     XOR     AH,48H         ; FLIP A COUPLE OF BITS IN STATUS
0072 EB BB                        JMP     B20            ; RETURN FROM ROUTINE WITH STATUS IN AH


                          ;-----  INITIALIZE THE PRINTER PORT

0074                      B80:
0074 42                           INC     DX             ; POINT TO OUTPUT PORT
0075 42                           INC     DX
0076 B0 08                        MOV     AL,8           ; SET INIT LINE LOW
0078 EE                           OUT     DX,AL
0079 B8 03E8                      MOV     AX,1000        ; ADJUST FOR INITIALIZATION DELAY LOOP
007C                      B90:
007C 48                           DEC     AX             ; DECREMENT DELAY COUNTER
007D 75 FD                        JNZ     B90            ; LOOP FOR RESET TO TAKE

007F B0 0C                        MOV     AL,0CH         ; NO INTERRUPTS, NON AUTO LF, INIT HIGH
0081 EE                           OUT     DX,AL          ; SET DEFAULT INITIAL OUTPUTS
0082 4A                           DEC     DX             ; ADJUST BACK TO BASE ADDRESS
0083 4A                           DEC     DX             ;  FOR STATUS ROUTINE EXIT
0084 EB E0                        JMP     B60            ; EXIT THROUGH STATUS ROUTINE

0086                      PRINTER_IO_1    ENDP

0086                              CODE    ENDS
                                  END
```

```
                          PAGE 118,121
                          TITLE RS232 ---- 01/10/86  COMMUNICATIONS BIOS (RS232)
                          .LIST
0000                      CODE    SEGMENT BYTE PUBLIC

                                  PUBLIC  RS232_IO_1
                                  EXTRN   A1:NEAR
                                  EXTRN   DDS:NEAR

                          ;--- INT 14 H -----------------------------------------------------------------
                          ;RS232_IO                                                                     :
                          ;       THIS ROUTINE PROVIDES BYTE STREAM I/O TO THE COMMUNICATIONS           :
                          ;       PORT ACCORDING TO THE PARAMETERS:                                     :
                          ;                                                                             :
                          ;       (AH)= 00H  INITIALIZE THE COMMUNICATIONS PORT                         :
                          ;               (AL) HAS PARAMETERS FOR INITIALIZATION                        :
                          ;                                                                             :
                          ;            7      6      5      4     3       2        1      0             :
                          ;            ----- BAUD RATE --    -PARITY--   STOPBIT   --WORD LENGTH--      :
                          ;                                                                             :
                          ;            000 - 110             X0 - NONE    0 - 1     10 - 7 BITS         :
                          ;            001 - 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=03H)    :
                          ;                                                                             :
                          ;       (AH)= 01H  SEND THE CHARACTER IN (AL) OVER THE COMMO LINE             :
                          ;                  (AL) REGISTER IS PRESERVED                                 :
                          ;                  ON EXIT, BIT 7 OF AH IS SET IF THE ROUTINE WAS UNABLE TO   :
                          ;                       TO TRANSMIT THE BYTE OF DATA OVER THE LINE.           :
                          ;                       IF BIT 7 OF AH IS NOT SET, THE                        :
                          ;                       REMAINDER OF (AH) IS SET AS IN A STATUS REQUEST,      :
                          ;                       REFLECTING THE CURRENT STATUS OF THE LINE.            :
                          ;       (AH)= 02H  RECEIVE A CHARACTER IN (AL) FROM COMMO LINE BEFORE         :
                          ;                       RETURNING TO CALLER                                   :
                          ;                  ON EXIT, (AH) HAS THE CURRENT LINE STATUS, AS SET BY THE   :
                          ;                       THE STATUS ROUTINE, EXCEPT THAT THE ONLY BITS         :
                          ;                       LEFT ON ARE THE ERROR BITS (7,4,3,2,1)                :
                          ;                       IF (AH) HAS BIT 7 ON (TIME OUT) THE REMAINING         :
                          ;                       BITS ARE NOT PREDICTABLE.                             :
                          ;                       THUS, (AH) IS NON ZERO ONLY WHEN AN ERROR OCCURRED.   :
                          ;       (AH)= 03H  RETURN THE COMMO PORT STATUS IN (AX)                       :
                          ;                   (AH) CONTAINS THE LINE CONTROL STATUS                     :
                          ;                         BIT 7 = TIME OUT                                    :
                          ;                         BIT 6 = TRANSMIT SHIFT REGISTER EMPTY               :
                          ;                         BIT 5 = TRANSMIT HOLDING REGISTER EMPTY             :
                          ;                         BIT 4 = BREAK DETECT                                :
                          ;                         BIT 3 = FRAMING ERROR                               :
                          ;                         BIT 2 = PARITY ERROR                                :
                          ;                         BIT 1 = OVERRUN ERROR                               :
                          ;                         BIT 0 = DATA READY                                  :
                          ;                   (AL) CONTAINS THE MODEM STATUS                            :
                          ;                         BIT 7 = RECEIVE LINE SIGNAL DETECT                  :
                          ;                         BIT 6 = RING INDICATOR                              :
                          ;                         BIT 5 = DATA SET READY                              :
                          ;                         BIT 4 = CLEAR TO SEND                               :
                          ;                         BIT 3 = DELTA RECEIVE LINE SIGNAL DETECT            :
                          ;                         BIT 2 = TRAILING EDGE RING DETECTOR                 :
                          ;                         BIT 1 = DELTA DATA SET READY                        :
                          ;                         BIT 0 = DELTA CLEAR TO SEND                         :
                          ;                                                                             :
                          ;       (DX) = PARAMETER INDICATING WHICH RS232 CARD (0,1 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 LABEL @RS232_TIM_OUT (BYTE) CONTAINS OUTER LOOP COUNT       :
                          ;       VALUE FOR TIMEOUT (DEFAULT=1)                                         :
                          ;OUTPUT                                                                       :
                          ;            AX MODIFIED ACCORDING TO PARAMETERS OF CALL                      :
                          ;            ALL OTHERS UNCHANGED                                             :
                          ;-------------------------------------------------------------------------------
                                  ASSUME  CS:CODE,DS:DATA

0000                      RS232_IO_1      PROC    FAR
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 83 FA 03                     CMP     DX,03H                  ; CHECK FOR ADAPTER NUMBER VALID 0-3
000A 77 24                        JA      A3E                     ; ERROR EXIT IF OUT OF RANGE
000C 8B F2                        MOV     SI,DX                   ; RS232 VALUE TO (SI)
000E 8B FA                        MOV     DI,DX                   ; AND TO (DI) (FOR TIMEOUTS)
0010 D1 E6                        SHL     SI,1                    ; WORD OFFSET
0012 E8 0000 E                    CALL    DDS
0015 8B 94 0000 R                 MOV     DX,@RS232_BASE[SI]      ; GET BASE ADDRESS
0019 0B D2                        OR      DX,DX                   ; TEST FOR 0 BASE ADDRESS
001B 74 13                        JZ      A3E                     ; RETURN
001D 0A E4                        OR      AH,AH                   ; TEST FOR (AH)= 00H
001F 74 18                        JZ      A4                      ; COMMO INITIALIZATION
0021 FE CC                        DEC     AH                      ; TEST FOR (AH)= 01H
0023 74 4B                        JZ      A5                      ; SEND (AL)
0025 FE CC                        DEC     AH                      ; TEST FOR (AH)= 02H
0027 74 70                        JZ      A12                     ; RECEIVE INTO (AL)
0029                      A2:
0029 FE CC                        DEC     AH                      ; TEST FOR (AH)= 03H
002B 75 03                        JNZ     A3E                     ;  ERROR IF BAD COMMAND
002D E9 00BB R                    JMP     A18                     ; COMMUNICATION STATUS
0030                      A3E:
0030 B4 80                        MOV     AH,080H                 ; SET ERROR RETURN CODE
0032                      A3:
0032 5B                           POP     BX                      ; RETURN FROM RS232
0033 59                           POP     CX
0034 5F                           POP     DI
0035 5E                           POP     SI
0036 5A                           POP     DX
0037 1F                           POP     DS
0038 CF                           IRET                            ; RETURN TO CALLER, NO ACTION
```

SECTION 5

```
                                PAGE
                                ;-----  INITIALIZE THE COMMUNICATIONS PORT

 0039                           A4:
 0039 8A E0                             MOV     AH,AL                   ; SAVE INITIALIZATION PARAMETERS IN (AH)
 003B 83 C2 03                          ADD     DX,3                    ; POINT TO 8250 CONTROL REGISTER
 003E B0 80                             MOV     AL,80H
 0040 EE                                OUT     DX,AL                   ; SET DLAB=1

                                ;-----  DETERMINE BAUD RATE DIVISOR

 0041 8A D4                             MOV     DL,AH                   ; GET PARAMETERS TO (DL)
 0043 B1 04                             MOV     CL,4
 0045 D2 C2                             ROL     DL,CL
 0047 81 E2 000E                        AND     DX,0EH                  ; ISOLATE THEM
 004B BF 0000 E                         MOV     DI,OFFSET A1            ; BASE OF TABLE
 004E 03 FA                             ADD     DI,DX                   ; PUT INTO INDEX REGISTER
 0050 8B 94 0000 R                      MOV     DX,@RS232_BASE[SI]      ; POINT TO HIGH ORDER OF DIVISOR
 0054 42                                INC     DX
 0055 2E: 8A 45 01                      MOV     AL,CS:[DI]+1            ; GET HIGH ORDER OF DIVISOR
 0059 EE                                OUT     DX,AL                   ; SET ms OF DIVISOR TO 0
 005A 4A                                DEC     DX
 005B 90                                NOP                             ; I/O DELAY
 005C 2E: 8A 05                         MOV     AL,CS:[DI]              ; GET LOW ORDER OF DIVISOR
 005F EE                                OUT     DX,AL                   ; SET LOW OF DIVISOR
 0060 83 C2 03                          ADD     DX,3
 0063 8A C4                             MOV     AL,AH                   ; GET PARAMETERS BACK
 0065 24 1F                             AND     AL,01FH                 ; STRIP OFF THE BAUD BITS
 0067 EE                                OUT     DX,AL                   ; LINE CONTROL TO 8 BITS
 0068 4A                                DEC     DX
 0069 4A                                DEC     DX
 006A 90                                NOP                             ; I/O DELAY
 006B B0 00                             MOV     AL,0
 006D EE                                OUT     DX,AL                   ; INTERRUPT ENABLES ALL OFF
 006E EB 4B                             JMP     SHORT A18               ; COM_STATUS

                                ;-----  SEND CHARACTER IN (AL) OVER COMMO LINE

 0070                           A5:
 0070 50                                PUSH    AX                      ; SAVE CHAR TO SEND
 0071 83 C2 04                          ADD     DX,4                    ; MODEM CONTROL REGISTER
 0074 B0 03                             MOV     AL,3                    ; DTR AND RTS
 0076 EE                                OUT     DX,AL                   ; DATA TERMINAL READY, REQUEST TO SEND
 0077 42                                INC     DX                      ; MODEM STATUS REGISTER
 0078 42                                INC     DX
 0079 B7 30                             MOV     BH,30H                  ; DATA SET READY & CLEAR TO SEND
 007B E8 00CA R                         CALL    WAIT_FOR_STATUS         ; ARE BOTH TRUE
 007E 74 08                             JE      A9                      ; YES, READY TO TRANSMIT CHAR
 0080                           A7:
 0080 59                                POP     CX
 0081 8A C1                             MOV     AL,CL                   ; RELOAD DATA BYTE
 0083                           A8:
 0083 80 CC 80                          OR      AH,80H                  ; INDICATE TIME OUT
 0086 EB AA                             JMP     A3                      ; RETURN

 0088                           A9:                                     ; CLEAR_TO_SEND
 0088 4A                                DEC     DX                      ; LINE_STATUS REGISTER
 0089                           A10:                                    ; WAIT_SEND
 0089 B7 20                             MOV     BH,20H                  ; IS TRANSMITTER READY
 008B E8 00CA R                         CALL    WAIT_FOR_STATUS         ; TEST FOR TRANSMITTER READY
 008E 75 F0                             JNZ     A7                      ; RETURN WITH TIME OUT SET
 0090                           A11:                                    ; OUT_CHAR
 0090 83 EA 05                          SUB     DX,5                    ; DATA_PORT
 0093 59                                POP     CX                      ; RECOVER IN CX TEMPORARILY
 0094 8A C1                             MOV     AL,CL                   ; MOVE CHAR TO AL FOR OUT, STATUS IN AH
 0096 EE                                OUT     DX,AL                   ; OUTPUT CHARACTER
 0097 EB 99                             JMP     A3                      ; RETURN

                                ;-----  RECEIVE CHARACTER FROM COMMO LINE

 0099                           A12:
 0099 83 C2 04                          ADD     DX,4                    ; MODEM CONTROL REGISTER
 009C B0 01                             MOV     AL,1                    ; DATA TERMINAL READY
 009E EE                                OUT     DX,AL
 009F 42                                INC     DX                      ; MODEM STATUS REGISTER
 00A0 42                                INC     DX
 00A1                           A13:                                    ; WAIT_DSR
 00A1 B7 20                             MOV     BH,20H                  ; DATA_SET READY
 00A3 E8 00CA R                         CALL    WAIT_FOR_STATUS         ; TEST FOR DSR
 00A6 75 DB                             JNZ     A8                      ; RETURN WITH ERROR
 00A8                           A15:                                    ; WAIT_DSR_END
 00A8 4A                                DEC     DX                      ; LINE_STATUS REGISTER
 00A9                           A16:                                    ; WAIT_RECV
 00A9 B7 01                             MOV     BH,1                    ; RECEIVE BUFFER FULL
 00AB E8 00CA R                         CALL    WAIT_FOR_STATUS         ; TEST FOR RECEIVE BUFFER FULL
 00AE 75 D3                             JNZ     A8                      ; SET TIME OUT ERROR
 00B0                           A17:                                    ; GET_CHAR
 00B0 80 E4 1E                          AND     AH,00011110B            ; TEST FOR ERROR CONDITIONS ON RECEIVE

 00B3 8B 94 0000 R                      MOV     DX,@RS232_BASE[SI]      ; DATA PORT
 00B7 EC                                IN      AL,DX                   ; GET CHARACTER FROM LINE
 00B8 E9 0032 R                         JMP     A3                      ; RETURN

                                ;-----  COMMO PORT STATUS ROUTINE

 00BB                           A18:
 00BB 8B 94 0000 R                      MOV     DX,@RS232_BASE[SI]
 00BF 83 C2 05                          ADD     DX,5                    ; CONTROL PORT
 00C2 EC                                IN      AL,DX                   ; GET LINE CONTROL STATUS
 00C3 8A E0                             MOV     AH,AL                   ; PUT IN (AH) FOR RETURN
 00C5 42                                INC     DX                      ; POINT TO MODEM STATUS REGISTER
 00C6 EC                                IN      AL,DX                   ; GET MODEM CONTROL STATUS
 00C7 E9 0032 R                         JMP     A3                      ; RETURN
```

```
                                PAGE
                                ;----------------------------------------
                                ;       WAIT FOR STATUS ROUTINE          :
                                ;ENTRY: (BH)= STATUS BIT(S) TO LOOK FOR :
                                ;       (DX)= ADDRESS OF STATUS REG      :
                                ;EXIT:  ZERO FLAG ON = STATUS FOUND      :
                                ;       ZERO FLAG OFF = TIMEOUT.         :
                                ;       (AH)= LAST STATUS READ           :
                                ;----------------------------------------

00CA                            WAIT_FOR_STATUS PROC    NEAR

00CA 8A 9D 007C R                       MOV     BL,@RS232_TIM_OUT[DI]   ; LOAD OUTER LOOP COUNT
00CE                            WFS0:
00CE 2B C9                              SUB     CX,CX
00D0                            WFS1:
00D0 EC                                 IN      AL,DX                   ; GET STATUS
00D1 8A E0                              MOV     AH,AL                   ; MOVE TO (AH)
00D3 22 C7                              AND     AL,BH                   ; ISOLATE BITS TO TEST
00D5 3A C7                              CMP     AL,BH                   ; EXACTLY = TO MASK
00D7 74 08                              JE      WFS_END                 ; RETURN WITH ZERO FLAG ON

00D9 E2 F5                              LOOP    WFS1                    ; TRY AGAIN

00DB FE CB                              DEC     BL                      ; DECREMENT LOOP COUNTER
00DD 75 EF                              JNZ     WFS0

00DF 0A FF                              OR      BH,BH                   ; SET ZERO FLAG OFF
00E1                            WFS_END:
00E1 C3                                 RET

00E2                            WAIT_FOR_STATUS ENDP

00E2                            RS232_IO_1      ENDP

00E2                            CODE    ENDS
                                        END
```

SECTION 5

```
                        PAGE 118,121
                        TITLE VIDEO ---- 01/10/86  VIDEO DISPLAY BIOS
                        .LIST
 0000                   CODE    SEGMENT BYTE PUBLIC

                                PUBLIC  ACT_DISP_PAGE
                                PUBLIC  READ_AC_CURRENT
                                PUBLIC  READ_CURSOR
                                PUBLIC  READ_DOT
                                PUBLIC  READ_LPEN
                                PUBLIC  SCROLL_DOWN
                                PUBLIC  SCROLL_UP
                                PUBLIC  SET_COLOR
                                PUBLIC  SET_CPOS
                                PUBLIC  SET_CTYPE
                                PUBLIC  SET_MODE
                                PUBLIC  WRITE_AC_CURRENT
                                PUBLIC  WRITE_C_CURRENT
                                PUBLIC  WRITE_DOT
                                PUBLIC  WRITE_TTY
                                PUBLIC  VIDEO_IO_1
                                PUBLIC  VIDEO_STATE

                                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
                                PUBLIC  VIDEO_RETURN
                                PUBLIC  VIDEO_RETURN
                                PUBLIC  VIDEO_RETURN
                                PUBLIC  WRITE_STRING

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

                        ;--- INT 10 H -------------------------------------------------------------
                        ;                                                                        :
                        ; VIDEO_IO                                                               :
                        ;       THESE ROUTINES PROVIDE THE CRT DISPLAY INTERFACE                 :
                        ;       THE FOLLOWING FUNCTIONS ARE PROVIDED:                            :
                        ;                                                                        :
                        ;    (AH)= 00H  SET MODE (AL) CONTAINS MODE VALUE                        :
                        ;               (AL) = 00H  40X25 BW MODE (POWER ON DEFAULT)             :
                        ;               (AL) = 01H  40X25 COLOR                                  :
                        ;               (AL) = 02H  80X25 BW                                     :
                        ;               (AL) = 03H  80X25 COLOR                                  :
                        ;                           GRAPHICS MODES                               :
                        ;               (AL) = 04H  320X200 COLOR                                :
                        ;               (AL) = 05H  320X200 BW MODE                              :
                        ;               (AL) = 06H  640X200 BW MODE                              :
                        ;               (AL) = 07H   80X25 MONOCHROME (USED INTERNAL TO VIDEO ONLY) :
                        ;               *** NOTES -BW MODES OPERATE SAME AS COLOR MODES, BUT COLOR :
                        ;                          BURST IS NOT ENABLED                          :
                        ;                         -CURSOR IS NOT DISPLAYED IN GRAPHICS MODE      :
                        ;    (AH)= 01H  SET CURSOR TYPE                                          :
                        ;               (CH) =  BITS 4-0 = START LINE FOR CURSOR                 :
                        ;                       ** HARDWARE WILL ALWAYS CAUSE BLINK              :
                        ;                       ** SETTING BIT 5 OR 6 WILL CAUSE ERRATIC BLINKING :
                        ;                          OR NO CURSOR AT ALL                           :
                        ;               (CL) =  BITS 4-0 = END LINE FOR CURSOR                   :
                        ;    (AH)= 02H  SET CURSOR POSITION                                      :
                        ;               (DH,DL) = ROW,COLUMN  (00H,00H) IS UPPER LEFT            :
                        ;               (BH) = PAGE NUMBER (MUST BE 00H FOR GRAPHICS MODES)      :
                        ;    (AH)= 03H  READ CURSOR POSITION                                     :
                        ;               (BH) = PAGE NUMBER (MUST BE 00H FOR GRAPHICS MODES)      :
                        ;               ON EXIT (DH,DL) = ROW,COLUMN OF CURRENT CURSOR           :
                        ;                       (CH,CL) = CURSOR MODE CURRENTLY SET              :
                        ;    (AH)= 04H  READ LIGHT PEN POSITION                                  :
                        ;               ON EXIT:                                                 :
                        ;               (AH) = 00H -- LIGHT PEN SWITCH NOT DOWN/NOT TRIGGERED    :
                        ;               (AH) = 01H -- VALID LIGHT PEN VALUE IN REGISTERS         :
                        ;                       (DH,DL) = ROW,COLUMN OF CHARACTER LP POSITION    :
                        ;                       (CH) = RASTER LINE (0-199)                       :
                        ;                       (BX) = PIXEL COLUMN (0-319,639)                  :
                        ;    (AH)= 05H  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)= 06H  SCROLL ACTIVE PAGE UP                                    :
                        ;               (AL) = NUMBER OF LINES, ( LINES BLANKED AT BOTTOM OF WINDOW ) :
                        ;                       (AL) = 00H MEANS BLANK ENTIRE WINDOW             :
                        ;               (CH,CL) = ROW,COLUMN OF UPPER LEFT CORNER OF SCROLL      :
                        ;               (DH,DL) = ROW,COLUMN OF LOWER RIGHT CORNER OF SCROLL     :
                        ;               (BH) = ATTRIBUTE TO BE USED ON BLANK LINE                :
                        ;    (AH)= 07H  SCROLL ACTIVE PAGE DOWN                                  :
                        ;               (AL) = NUMBER OF LINES, INPUT LINES BLANKED AT TOP OF WINDOW :
                        ;                       (AL) = 00H MEANS BLANK ENTIRE WINDOW             :
                        ;               (CH,CL) = ROW,COLUMN OF UPPER LEFT CORNER OF SCROLL      :
                        ;               (DH,DL) = ROW,COLUMN OF LOWER RIGHT CORNER OF SCROLL     :
                        ;               (BH) = ATTRIBUTE TO BE USED ON BLANK LINE                :
                        ;                                                                        :
                        ;  CHARACTER HANDLING ROUTINES                                           :
                        ;                                                                        :
                        ;    (AH)= 08H  READ ATTRIBUTE/CHARACTER AT CURRENT CURSOR POSITION      :
                        ;               (BH) = DISPLAY PAGE (VALID FOR ALPHA MODES ONLY)         :
                        ;               ON EXIT:                                                 :
                        ;               (AL) = CHAR READ                                         :
                        ;               (AH) = ATTRIBUTE OF CHARACTER READ (ALPHA MODES ONLY)    :
                        ;    (AH)= 09H  WRITE ATTRIBUTE/CHARACTER AT CURRENT CURSOR POSITION     :
                        ;               (BH) = DISPLAY PAGE (VALID FOR ALPHA MODES ONLY)         :
```

```
                        ;                (CX) = COUNT OF CHARACTERS TO WRITE                         ;
                        ;                (AL) = CHAR TO WRITE                                        ;
                        ;                (BL) = ATTRIBUTE OF CHARACTER (ALPHA)/COLOR OF CHAR (GRAPHICS);
                        ;                       SEE NOTE ON WRITE DOT FOR BIT 7 OF BL = 1.           ;
                        ;       (AH)= 0AH  WRITE CHARACTER ONLY AT CURRENT CURSOR POSITION           ;
                        ;                (BH) = DISPLAY PAGE (VALID FOR ALPHA MODES ONLY)            ;
                        ;                (CX) = COUNT OF CHARACTERS TO WRITE                         ;
                        ;                (AL) = CHAR TO WRITE                                        ;
                        ;                       NOTE: USE FUNCTION (AH)= 09H IN GRAPHICS MODES       ;
                        ;          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)= 0BH  SET COLOR PALETTE                                         ;
                        ;                (BH) = PALETTE 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 PALETTE 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     ;
                        ;                             PALETTE COLOR 0 INDICATES THE BORDER COLOR     ;
                        ;                             TO BE USED (VALUES 0-31, WHERE 16-31 SELECT    ;
                        ;                             THE HIGH INTENSITY BACKGROUND SET.             ;
                        ;       (AH)= 0CH  WRITE DOT                                                 ;
                        ;                (DX) = ROW NUMBER                                           ;
                        ;                (CX) = COLUMN NUMBER                                        ;
                        ;                (AL) = COLOR VALUE                                          ;
                        ;                       IF BIT 7 OF AL = 1, THEN THE COLOR VALUE IS EXCLUSIVE ;
                        ;                       ORed WITH THE CURRENT CONTENTS OF THE DOT            ;
                        ;       (AH)= 0DH  READ DOT                                                  ;
                        ;                (DX) = ROW NUMBER                                           ;
                        ;                (CX) = COLUMN NUMBER                                        ;
                        ;                (AL) RETURNS THE DOT READ                                   ;
                        ;                                                                            ;
                        ;       ASCII TELETYPE ROUTINE FOR OUTPUT                                    ;
                        ;                                                                            ;
                        ;       (AH)= 0EH  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)= 0FH  CURRENT VIDEO STATE                                       ;
                        ;                RETURNS THE CURRENT VIDEO STATE                             ;
                        ;                (AL) = MODE CURRENTLY SET ( SEE (AH)= 00H FOR EXPLANATION)  ;
                        ;                (AH) = NUMBER OF CHARACTER COLUMNS ON SCREEN                ;
                        ;                (BH) = CURRENT ACTIVE DISPLAY PAGE                          ;
                        ;       (AH)= 10H  RESERVED                                                  ;
                        ;       (AH)= 11H  RESERVED                                                  ;
                        ;       (AH)= 12H  RESERVED                                                  ;
                        ;       (AH)= 13H  WRITE STRING                                              ;
                        ;                         ES:BP   -  POINTER TO STRING TO BE WRITTEN         ;
                        ;                         CX      -  LENGTH OF CHARACTER STRING TO WRITTEN   ;
                        ;                         DX      -  CURSOR POSITION FOR STRING TO BE WRITTEN ;
                        ;                         BH      -  PAGE NUMBER                             ;
                        ;              (AL)= 00H  WRITE CHARACTER STRING                             ;
                        ;                         BL      -  ATTRIBUTE                               ;
                        ;                         STRING IS  <CHAR,CHAR, ... ,CHAR>                  ;
                        ;                         CURSOR NOT MOVED                                   ;
                        ;              (AL)= 01H  WRITE CHARACTER STRING AND MOVE CURSOR             ;
                        ;                         BL      -  ATTRIBUTE                               ;
                        ;                         STRING IS  <CHAR,CHAR, ... ,CHAR>                  ;
                        ;                         CURSOR IS MOVED                                    ;
                        ;              (AL)= 02H  WRITE CHARACTER AND ATTRIBUTE STRING               ;
                        ;                               (VALID FOR ALPHA MODES ONLY)                 ;
                        ;                         STRING IS  <CHAR,ATTR,CHAR,ATTR ..  ,CHAR,ATTR>    ;
                        ;                         CURSOR IS NOT MOVED                                ;
                        ;              (AL)= 03H  WRITE CHARACTER AND ATTRIBUTE STRING AND MOVE CURSOR ;
                        ;                               (VALID FOR ALPHA MODES ONLY)                 ;
                        ;                         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. ;
                        ;                                                                            ;
                        ;         BX,CX,DX,SI,DI,BP,SP,DS,ES,SS PRESERVED DURING CALLS EXCEPT FOR    ;
                        ;         BX,CX,DX RETURN VALUES ON FUNCTIONS 03H,04H,0DH AND 0DH. ON ALL CALLS ;
                        ;         AX IS MODIFIED.                                                    ;
                        ;-----------------------------------------------------------------------------

                                  ASSUME  CS:CODE,DS:DATA,ES:NOTHING

 0000 005F R            M1        DW      OFFSET  SET_MODE        ;  TABLE OF ROUTINES WITHIN VIDEO I/O
 0002 0146 R                      DW      OFFSET  SET_CTYPE
 0004 0167 R                      DW      OFFSET  SET_CPOS
 0006 018F R                      DW      OFFSET  READ_CURSOR
 0008 0785 R                      DW      OFFSET  READ_LPEN
 000A 01A6 R                      DW      OFFSET  ACT_DISP_PAGE
 000C 020F R                      DW      OFFSET  SCROLL_UP
 000E 02AD R                      DW      OFFSET  SCROLL_DOWN
 0010 02FF R                      DW      OFFSET  READ_AC_CURRENT
 0012 035C R                      DW      OFFSET  WRITE_AC_CURRENT
 0014 038E R                      DW      OFFSET  WRITE_C_CURRENT
 0016 01C8 R                      DW      OFFSET  SET_COLOR
 0018 0450 R                      DW      OFFSET  WRITE_DOT
 001A 043F R                      DW      OFFSET  READ_DOT
 001C 06FE R                      DW      OFFSET  WRITE_TTY
 001E 01EE R                      DW      OFFSET  VIDEO_STATE
 0020 013D R                      DW      OFFSET  VIDEO_RETURN    ; RESERVED
 0022 013D R                      DW      OFFSET  VIDEO_RETURN    ; RESERVED
 0024 013D R                      DW      OFFSET  VIDEO_RETURN    ; RESERVED
 0026 03BB R                      DW      OFFSET  WRITE_STRING    ; CASE 13H, WRITE STRING
 = 0028                 M1L       EQU     $-M1
```

SECTION 5

```
0028                    VIDEO_IO_1      PROC    NEAR            ;       ENTRY POINT FOR ORG 0F065H
0028 FB                         STI                             ; INTERRUPTS BACK ON
0029 FC                         CLD                             ; SET DIRECTION FORWARD
002A 80 FC 14                   CMP     AH,M1L/2                ; TEST FOR WITHIN TABLE RANGE
002D 73 2F                      JNB     M4                      ; BRANCH TO EXIT IF NOT A VALID COMMAND

002F 06                         PUSH    ES
0030 1E                         PUSH    DS                      ; SAVE WORK AND PARAMETER REGISTERS
0031 52                         PUSH    DX
0032 51                         PUSH    CX
0033 53                         PUSH    BX
0034 56                         PUSH    SI
0035 57                         PUSH    DI
0036 55                         PUSH    BP
0037 BE ---- R                  MOV     SI,DATA                 ; POINT DS: TO DATA SEGMENT
003A 8E DE                      MOV     DS,SI
003C 8B F0                      MOV     SI,AX                   ; SAVE COMMAND/DATA INTO (SI) REGISTER
003E A0 0010 R                  MOV     AL,BYTE PTR @EQUIP_FLAG ; GET THE EQUIPMENT FLAG VIDEO BITS
0041 24 30                      AND     AL,30H                  ; ISOLATE CRT SWITCHES
0043 3C 30                      CMP     AL,30H                  ; IS SETTING FOR MONOCHROME CARD?
0045 BF B800                    MOV     DI,0B800H               ; GET SEGMENT FOR COLOR CARD
0048 75 03                      JNE     M2                      ; SKIP IF NOT MONOCHROME CARD
004A BF B000                    MOV     DI,0B000H               ; ELSE GET SEGMENT FOR MONOCHROME CARD
004D                    M2:
004D 8E C7                      MOV     ES,DI                   ; SET UP TO POINT AT VIDEO MEMORY AREAS
004F 8A C4                      MOV     AL,AH                   ; PLACE COMMAND IN LOW BYTE OF (AX)
0051 98                         CBW                             ; AND FORM BYTE OFFSET WITH COMMAND
0052 D1 E0                      SAL     AX,1                    ; TIMES 2 FOR WORD TABLE LOOKUP
0054 96                         XCHG    SI,AX                   ; MOVE OFFSET INTO LOOK UP REGISTER (SI)
                                                                ;  AND RESTORE COMMAND/DATA INTO (AX)
0055 8A 26 0049 R               MOV     AH,@CRT_MODE            ; MOVE CURRENT MODE INTO (AH) REGISTER

0059 2E: FF A4 0000 R           JMP     WORD PTR CS:[SI+OFFSET M1]      ; GO TO SELECTED FUNCTION

005E                    M4:                                     ;       COMMAND NOT VALID
005E CF                         IRET                            ; DO NOTHING IF NOT IN VALID RANGE
005F                    VIDEO_IO_1      ENDP
                        ;--------------------------------------------------------
                        ; SET_MODE                                              :
                        ;       THIS ROUTINE INITIALIZES THE ATTACHMENT TO      :
                        ;       THE SELECTED MODE.  THE SCREEN IS BLANKED.      :
                        ; INPUT                                                 :
                        ;       @EQUIP_FLAG BITS 5-4 = MODE/WIDTH               :
                        ;          11 = MONOCHROME (FORCES MODE 7)              :
                        ;          01 = COLOR ADAPTER 40x25 (MODE 0 DEFAULT)    :
                        ;          10 = COLOR ADAPTER 80x25 (MODE 2 DEFAULT)    :
                        ;        (AL) = COLOR MODE REQUESTED ( RANGE  0 - 6 )   :
                        ; OUTPUT                                                :
                        ;       NONE                                            :
                        ;--------------------------------------------------------
005F                    SET_MODE        PROC    NEAR
005F BA 03D4                    MOV     DX,03D4H                ; ADDRESS OF COLOR CARD
0062 8B 3E 0010 R               MOV     DI,@EQUIP_FLAG          ; GET EQUIPMENT FLAGS SETTING
0066 81 E7 0030                 AND     DI,30H                  ; ISOLATE CRT SWITCHES
006A 83 FF 30                   CMP     DI,30H                  ; IS BW CARD INSTALLED AS PRIMARY
006D 75 06                      JNE     M8C                     ; SKIP AND CHECK IF COLOR
006F B0 07                      MOV     AL,7                    ; ELSE INDICATE INTERNAL BW CARD MODE
0071 B2 B4                      MOV     DL,0B4H                 ; SET ADDRESS OF BW (MONOCHROME) CARD
0073 EB 0D                      JMP     SHORT M8                ; CONTINUE WITH FORCED MODE 7
0075                    M8C:
0075 3C 07                      CMP     AL,7                    ; CHECK FOR VALID COLOR MODES 0-6
0077 72 09                      JB      M8                      ; CONTINUE IF BELOW MODE 7
0079 B0 00                      MOV     AL,0                    ; FORCE DEFAULT 40x25 BW MODE
007B 83 FF 20                   CMP     DI,20H                  ; CHECK FOR @EQUIP_FLAG AT 80x25 BW
007E 74 02                      JE      M8                      ; CONTINUE WITH MODE 0 IF NOT
0080 B0 02                      MOV     AL,2                    ; ELSE FORCE MODE 2
0082                    M8:
0082 A2 0049 R                  MOV     @CRT_MODE,AL            ; SAVE MODE IN GLOBAL VARIABLE
0085 89 16 0063 R               MOV     @ADDR_6845,DX           ; SAVE ADDRESS OF BASE
0089 C6 06 0084 R 18            MOV     @ROWS,25-1              ; INITIALIZE DEFAULT ROW COUNT OF 25
008E 1E                         PUSH    DS                      ; SAVE POINTER TO DATA SEGMENT
008F 50                         PUSH    AX                      ; SAVE MODE NUMBER (AL)
0090 98                         CBW                             ; CLEAR HIGH BYTE OF MODE
0091 8B F0                      MOV     SI,AX                   ; SET TABLE POINTER, INDEXED BY MODE
0093 2E: 8A 84 0000 E           MOV     AL,CS:[SI + OFFSET M7]  ; GET THE MODE SET VALUE FROM TABLE
0098 A2 0065 R                  MOV     @CRT_MODE_SET,AL        ; SAVE THE MODE SET VALUE
009B 24 37                      AND     AL,037H                 ; VIDEO OFF, SAVE HIGH RESOLUTION BIT
009D 52                         PUSH    DX                      ; SAVE OUTPUT PORT VALUE
009E 83 C2 04                   ADD     DX,4                    ; POINT TO CONTROL REGISTER
00A1 EE                         OUT     DX,AL                   ; RESET VIDEO TO OFF TO SUPPRESS ROLLING
00A2 5A                         POP     DX                      ; BACK TO BASE REGISTER
                                ASSUME  DS:ABS0
00A3 2B DB                      SUB     BX,BX                   ; SET UP FOR ABS0 SEGMENT
00A5 8E DB                      MOV     DS,BX                   ; ESTABLISH VECTOR TABLE ADDRESSING
00A7 C5 1E 0074 R               LDS     BX,@PARM_PTR            ; GET POINTER TO VIDEO PARMS
                                ASSUME  DS:CODE
00AB 58                         POP     AX                      ; RECOVER MODE NUMBER IN (AL)
00AC B9 0010                    MOV     CX,16                   ; LENGTH OF EACH ROW OF TABLE
00AF 3C 02                      CMP     AL,2                    ; DETERMINE WHICH ONE TO USE
00B1 72 0E                      JC      M9                      ; MODE IS 0 OR 1
00B3 03 D9                      ADD     BX,CX                   ; NEXT ROW OF INITIALIZATION TABLE
00B5 3C 04                      CMP     AL,4
00B7 72 08                      JC      M9                      ; MODE IS 2 OR 3
00B9 03 D9                      ADD     BX,CX                   ; MOVE TO GRAPHICS ROW OF INIT_TABLE
00BB 3C 07                      CMP     AL,7
00BD 72 02                      JC      M9                      ; MODE IS 4,5, OR 6
00BF 03 D9                      ADD     BX,CX                   ; MOVE TO BW CARD ROW OF INIT_TABLE

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

00C1                    M9:                                     ; OUT_INIT
00C1 50                         PUSH    AX                      ; SAVE MODE IN (AL)
00C2 8B 47 0A                   MOV     AX,[BX+10]              ; GET THE CURSOR MODE FROM THE TABLE
00C5 86 E0                      XCHG    AH,AL                   ; PUT CURSOR MODE IN CORRECT POSITION
00C7 1E                         PUSH    DS                      ; SAVE TABLE SEGMENT POINTER
                                ASSUME  DS:DATA
00C8 E8 0000 E                  CALL    DDS                     ; POINT DS TO DATA SEGMENT
00CB A3 0060 R                  MOV     @CURSOR_MODE,AX         ; PLACE INTO BIOS DATA SAVE AREA
                                ASSUME  DS:CODE
00CE 1F                         POP     DS                      ; RESTORE THE TABLE SEGMENT POINTER
00CF 32 E4                      XOR     AH,AH                   ; AH IS REGISTER NUMBER DURING LOOP

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

```
00D1                    M10:                                            ; INITIALIZATION LOOP
00D1 8A C4                      MOV     AL,AH                           ; GET 6845 REGISTER NUMBER
00D3 EE                         OUT     DX,AL
00D4 42                         INC     DX                              ; POINT TO DATA PORT
00D5 FE C4                      INC     AH                              ; NEXT REGISTER VALUE
00D7 8A 07                      MOV     AL,[BX]                         ; GET TABLE VALUE
00D9 EE                         OUT     DX,AL                           ; OUT TO CHIP
00DA 43                         INC     BX                              ; NEXT IN TABLE
00DB 4A                         DEC     DX                              ; BACK TO POINTER REGISTER
00DC E2 F3                      LOOP    M10                             ; DO THE WHOLE TABLE
00DE 58                         POP     AX                              ; GET MODE BACK INTO (AL)
00DF 1F                         POP     DS                              ; RECOVER SEGMENT VALUE
                                ASSUME  DS:DATA
                        ;-----  FILL REGEN AREA WITH BLANK
00E0 33 FF                      XOR     DI,DI                           ; SET UP POINTER FOR REGEN
00E2 89 3E 004E R               MOV     @CRT_START,DI                   ; START ADDRESS SAVED IN GLOBAL
00E6 C6 06 0062 R 00            MOV     @ACTIVE_PAGE,0                  ; SET PAGE VALUE
00EB B9 2000                    MOV     CX,8192                         ; NUMBER OF WORDS IN COLOR CARD
00EE 3C 04                      CMP     AL,4                            ; TEST FOR GRAPHICS
00F0 72 0A                      JC      M12                             ; NO_GRAPHICS_INIT
00F2 3C 07                      CMP     AL,7                            ; TEST FOR BW CARD
00F4 74 04                      JE      M11                             ; BW_CARD_INIT
00F6 33 C0                      XOR     AX,AX                           ; FILL FOR GRAPHICS MODE
00F8 EB 05                      JMP     SHORT M13                       ; CLEAR_BUFFER
00FA                    M11:                                            ; BW_CARD_INIT
00FA B5 08                      MOV     CH,08H                          ; BUFFER SIZE ON BW CARD (2048)
00FC                    M12:                                            ; NO_GRAPHICS_INIT
00FC B8 0720                    MOV     AX,' '+7*H                      ; FILL CHAR FOR ALPHA + ATTRIBUTE
00FF                    M13:                                            ; CLEAR_BUFFER
00FF F3/ AB                     REP     STOSW                           ; FILL THE REGEN BUFFER WITH BLANKS
                        ;-----  ENABLE VIDEO AND CORRECT PORT SETTING
0101 8B 16 0063 R               MOV     DX,@ADDR_6845                   ; PREPARE TO OUTPUT TO VIDEO ENABLE PORT
0105 83 C2 04                   ADD     DX,4                            ; POINT TO THE MODE CONTROL REGISTER
0108 A0 0065 R                  MOV     AL,@CRT_MODE_SET                ; GET THE MODE SET VALUE
010B EE                         OUT     DX,AL                           ; SET VIDEO ENABLE PORT
                        ;-----  DETERMINE NUMBER OF COLUMNS, BOTH FOR ENTIRE DISPLAY
                        ;-----  AND THE NUMBER TO BE USED FOR TTY INTERFACE
010C 2E: 8A 84 0000 E           MOV     AL,CS:[SI + OFFSET M6]          ; GET NUMBER OF COLUMNS ON THIS SCREEN
0111 98                         CBW                                     ; CLEAR HIGH BYTE
0112 A3 004A R                  MOV     @CRT_COLS,AX                    ; INITIALIZE NUMBER OF COLUMNS COUNT
                        ;-----  SET CURSOR POSITIONS
0115 81 E6 000E                 AND     SI,000EH                        ; WORD OFFSET INTO CLEAR LENGTH TABLE
0119 2E: 8B 84 0000 E           MOV     AX,CS:[SI + OFFSET M5]          ; LENGTH TO CLEAR
011E A3 004C R                  MOV     @CRT_LEN,AX                     ; SAVE LENGTH OF CRT -- NOT USED FOR BW
0121 B9 0008                    MOV     CX,8                            ; CLEAR ALL CURSOR POSITIONS
0124 BF 0050 R                  MOV     DI,OFFSET @CURSOR_POSN
0127 1E                         PUSH    DS                              ; ESTABLISH SEGMENT
0128 07                         POP     ES                              ;   ADDRESSING
0129 33 C0                      XOR     AX,AX
012B F3/ AB                     REP     STOSW                           ; FILL WITH ZEROES
                        ;-----  SET UP OVERSCAN REGISTER
012D 42                         INC     DX                              ; SET OVERSCAN PORT TO A DEFAULT
012E B0 30                      MOV     AL,30H                          ; 30H VALUE FOR ALL MODES EXCEPT 640X200
0130 80 3E 0049 R 06            CMP     @CRT_MODE,6                     ; SEE IF THE MODE IS 640X200 BW
0135 75 02                      JNZ     M14                             ; IF NOT 640X200, THEN GO TO REGULAR
0137 B0 3F                      MOV     AL,3FH                          ; IF IT IS 640X200, THEN PUT IN 3FH
0139                    M14:
0139 EE                         OUT     DX,AL                           ; OUTPUT THE CORRECT VALUE TO 3D9 PORT
013A A2 0066 R                  MOV     @CRT_PALETTE,AL                 ; SAVE THE VALUE FOR FUTURE USE
                        ;-----  NORMAL RETURN FROM ALL VIDEO RETURNS
013D                    VIDEO_RETURN:
013D 5D                         POP     BP
013E 5F                         POP     DI
013F 5E                         POP     SI
0140 5B                         POP     BX
0141                    M15:                                            ; VIDEO_RETURN_C
0141 59                         POP     CX
0142 5A                         POP     DX
0143 1F                         POP     DS
0144 07                         POP     ES                              ; RECOVER SEGMENTS
0145 CF                         IRET                                    ; ALL DONE
0146                    SET_MODE        ENDP
                        ;------------------------------------------------
                        ; SET_CTYPE
                        ;       THIS ROUTINE SETS THE CURSOR VALUE
                        ; INPUT
                        ;       (CX) HAS CURSOR VALUE CH-START LINE, CL-STOP LINE
                        ; OUTPUT
                        ;       NONE
                        ;------------------------------------------------
0146                    SET_CTYPE       PROC    NEAR
0146 B4 0A                      MOV     AH,10                           ; 6845 REGISTER FOR CURSOR SET
0148 89 0E 0060 R               MOV     @CURSOR_MODE,CX                 ; SAVE IN DATA AREA
014C E8 0151 R                  CALL    M16                             ; OUTPUT CX REGISTER
014F EB EC                      JMP     VIDEO_RETURN
                        ;-----  THIS ROUTINE OUTPUTS THE CX REGISTER TO THE 6845 REGISTERS NAMED IN (AH)
0151                    M16:
0151 8B 16 0063 R               MOV     DX,@ADDR_6845                   ; ADDRESS REGISTER
0155 8A C4                      MOV     AL,AH                           ; GET VALUE
0157 EE                         OUT     DX,AL                           ; REGISTER SET
0158 42                         INC     DX                              ; DATA REGISTER
0159 8A C5                      MOV     AL,CH                           ; DATA
015B EE                         OUT     DX,AL
015C 4A                         DEC     DX
015D 8A C4                      MOV     AL,AH
015F FE C0                      INC     AL                              ; POINT TO OTHER DATA REGISTER
0161 EE                         OUT     DX,AL                           ; SET FOR SECOND REGISTER
0162 42                         INC     DX
0163 8A C1                      MOV     AL,CL                           ; SECOND DATA VALUE
```

SECTION 5

```
0165 EE                        OUT     DX,AL
0166 C3                        RET                             ; ALL DONE
0167                   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
                       ;------------------------------------------
0167                   SET_CPOS        PROC    NEAR
0167 8A C7                     MOV     AL,BH                   ; MOVE PAGE NUMBER TO WORK REGISTER
0169 98                        CBW                             ; CONVERT PAGE TO WORD VALUE
016A D1 E0                     SAL     AX,1                    ; WORD OFFSET
016C 96                        XCHG    AX,SI                   ; USE INDEX REGISTER
016D 89 94 0050 R              MOV     [SI+OFFSET @CURSOR_POSN],DX     ; SAVE THE POINTER
0171 38 3E 0062 R              CMP     @ACTIVE_PAGE,BH
0175 75 05                     JNZ     M17                     ; SET_CPOS_RETURN
0177 8B C2                     MOV     AX,DX                   ; GET ROW/COLUMN TO AX
0179 E8 017E R                 CALL    M18                     ; CURSOR_SET
017C                   M17:                                    ; SET_CPOS_RETURN
017C EB BF                     JMP     VIDEO_RETURN
017E                   SET_CPOS        ENDP

                       ;-----  SET CURSOR POSITION, AX HAS ROW/COLUMN FOR CURSOR

017E                   M18     PROC    NEAR
017E E8 0200 R                 CALL    POSITION                ; DETERMINE LOCATION IN REGEN BUFFER
0181 8B C8                     MOV     CX,AX
0183 03 0E 004E R              ADD     CX,@CRT_START           ; ADD IN THE START ADDRESS FOR THIS PAGE
0187 D1 F9                     SAR     CX,1                    ; DIVIDE BY 2 FOR CHAR ONLY COUNT
0189 B4 0E                     MOV     AH,14                   ; REGISTER NUMBER FOR CURSOR
018B E8 0151 R                 CALL    M16                     ; OUTPUT THE VALUE TO THE 6845
018E C3                        RET
018F                   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
                       ;------------------------------------------
018F                   READ_CURSOR     PROC    NEAR
018F 8A DF                     MOV     BL,BH
0191 32 FF                     XOR     BH,BH
0193 D1 E3                     SAL     BX,1                    ; WORD OFFSET
0195 8B 97 0050 R              MOV     DX,[BX+OFFSET @CURSOR_POSN]
0199 8B 0E 0060 R              MOV     CX,@CURSOR_MODE
019D 5D                        POP     BP
019E 5F                        POP     DI
019F 5E                        POP     SI
01A0 5B                        POP     BX
01A1 58                        POP     AX                      ; DISCARD SAVED CX AND DX
01A2 58                        POP     AX
01A3 1F                        POP     DS
01A4 07                        POP     ES
01A5 CF                        IRET
01A6                   READ_CURSOR     ENDP
                       ;----------------------------------------------
                       ; ACT_DISP_PAGE
                       ;       THIS ROUTINE SETS THE ACTIVE DISPLAY PAGE, ALLOWING
                       ;       THE FULL USE OF THE MEMORY SET ASIDE FOR THE VIDEO ATTACHMENT
                       ; INPUT
                       ;       AL HAS THE NEW ACTIVE DISPLAY PAGE
                       ; OUTPUT
                       ;       THE 6845 IS RESET TO DISPLAY THAT PAGE
                       ;----------------------------------------------
01A6                   ACT_DISP_PAGE   PROC    NEAR
01A6 A2 0062 R                 MOV     @ACTIVE_PAGE,AL         ; SAVE ACTIVE PAGE VALUE
01A9 98                        CBW                             ; CONVERT (AL) TO WORD
01AA 50                        PUSH    AX                      ; SAVE PAGE VALUE
01AB F7 26 004C R              MUL     WORD PTR @CRT_LEN       ; DISPLAY PAGE TIMES REGEN LENGTH
01AF A3 004E R                 MOV     @CRT_START,AX           ; SAVE START ADDRESS FOR LATER
01B2 8B C8                     MOV     CX,AX                   ; START ADDRESS TO CX
01B4 D1 F9                     SAR     CX,1                    ; DIVIDE BY 2 FOR 6845 HANDLING
01B6 B4 0C                     MOV     AH,12                   ; 6845 REGISTER FOR START ADDRESS
01B8 E8 0151 R                 CALL    M16
01BB 5B                        POP     BX                      ; RECOVER PAGE VALUE
01BC D1 E3                     SAL     BX,1                    ; *2 FOR WORD OFFSET
01BE 8B 87 0050 R              MOV     AX,[BX + OFFSET @CURSOR_POSN]   ; GET CURSOR FOR THIS PAGE
01C2 E8 017E R                 CALL    M18                     ; SET THE CURSOR POSITION
01C5 E9 013D R                 JMP     VIDEO_RETURN
01C8                   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 PALETTE 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
                       ;------------------------------------------
01C8                   SET_COLOR       PROC    NEAR
01C8 8B 16 0063 R              MOV     DX,@ADDR_6845           ; I/O PORT FOR PALETTE
01CC 83 C2 05                  ADD     DX,5                    ; OVERSCAN PORT
01CF A0 0066 R                 MOV     AL,@CRT_PALETTE         ; GET THE CURRENT PALETTE VALUE
01D2 0A FF                     OR      BH,BH                   ; IS THIS COLOR 0?
01D4 75 0E                     JNZ     M20                     ; OUTPUT COLOR 1

                       ;-----  HANDLE COLOR 0 BY SETTING THE BACKGROUND COLOR
```

```
01D6 24 E0                    AND     AL,0E0H                 ; TURN OFF LOW 5 BITS OF CURRENT
01D8 80 E3 1F                 AND     BL,01FH                 ; TURN OFF HIGH 3 BITS OF INPUT VALUE
01DB 0A C3                    OR      AL,BL                   ; PUT VALUE INTO REGISTER
01DD                    M19:                                  ; OUTPUT THE PALETTE
01DD EE                       OUT     DX,AL                   ; OUTPUT COLOR SELECTION TO 3D9 PORT
01DE A2 0066 R                MOV     @CRT_PALETTE,AL         ; SAVE THE COLOR VALUE
01E1 E9 013D R                JMP     VIDEO_RETURN

                        ;-----  HANDLE COLOR 1 BY SELECTING THE PALETTE TO BE USED

01E4                    M20:
01E4 24 DF                    AND     AL,0DFH                 ; TURN OFF PALETTE SELECT BIT
01E6 D0 EB                    SHR     BL,1                    ; TEST THE LOW ORDER BIT OF BL
01E8 73 F3                    JNC     M19                     ; ALREADY DONE
01EA 0C 20                    OR      AL,20H                  ; TURN ON PALETTE SELECT BIT
01EC EB EF                    JMP     M19                     ; GO DO IT
01EE                    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
                        ;------------------------------------------------
01EE                    VIDEO_STATE     PROC    NEAR
01EE 8A 26 004A R             MOV     AH,BYTE PTR @CRT_COLS   ; GET NUMBER OF COLUMNS
01F2 A0 0049 R                MOV     AL,@CRT_MODE            ; CURRENT MODE
01F5 8A 3E 0062 R             MOV     BH,@ACTIVE_PAGE         ; GET CURRENT ACTIVE PAGE
01F9 5D                       POP     BP                      ; RECOVER REGISTERS
01FA 5F                       POP     DI
01FB 5E                       POP     SI
01FC 59                       POP     CX                      ; DISCARD SAVED BX
01FD E9 0141 R                JMP     M15                     ; RETURN TO CALLER
0200                    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
                        ;---------------------------------------
0200                    POSITION        PROC    NEAR
0200 53                       PUSH    BX                      ; SAVE REGISTER
0201 93                       XCHG    BX,AX                   ; SAVE ROW/COLUNM POSITION IN (BX)
0202 A0 004A R                MOV     AL,BYTE PTR @CRT_COLS   ; GET COLUMNS PER ROW COUNT
0205 F6 E7                    MUL     BH                      ; DETERMINE BYTES TO ROW
0207 32 FF                    XOR     BH,BH
0209 03 C3                    ADD     AX,BX                   ; ADD IN COLUMN VALUE
020B D1 E0                    SAL     AX,1                    ; * 2 FOR ATTRIBUTE BYTES
020D 5B                       POP     BX
020E C3                       RET
020F                    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  DS:DATA,ES:DATA
020F                    SCROLL_UP       PROC    NEAR

020F E8 02EA R                CALL    TEST_LINE_COUNT
0212 80 FC 04                 CMP     AH,4                    ; TEST FOR GRAPHICS MODE
0215 72 08                    JC      N1                      ; HANDLE SEPARATELY
0217 80 FC 07                 CMP     AH,7                    ; TEST FOR BW CARD
021A 74 03                    JE      N1
021C E9 04AC R                JMP     GRAPHICS_UP
021F                    N1:                                   ; UP_CONTINUE
021F 53                       PUSH    BX                      ; SAVE FILL ATTRIBUTE IN BH
0220 8B C1                    MOV     AX,CX                   ; UPPER LEFT POSITION
0222 E8 025C R                CALL    SCROLL_POSITION         ; DO SETUP FOR SCROLL
0225 74 31                    JZ      N7                      ; BLANK_FIELD
0227 03 F0                    ADD     SI,AX                   ; FROM ADDRESS
0229 8A E6                    MOV     AH,DH                   ; # ROWS IN BLOCK
022B 2A E3                    SUB     AH,BL                   ; # ROWS TO BE MOVED
022D                    N2:                                   ; ROW_LOOP
022D E8 029D R                CALL    N10                     ; MOVE ONE ROW
0230 03 F5                    ADD     SI,BP
0232 03 FD                    ADD     DI,BP                   ; POINT TO NEXT LINE IN BLOCK
0234 FE CC                    DEC     AH                      ; COUNT OF LINES TO MOVE
0236 75 F5                    JNZ     N2                      ; ROW_LOOP
0238                    N3:                                   ; CLEAR_ENTRY
0238 58                       POP     AX                      ; RECOVER ATTRIBUTE IN AH
0239 B0 20                    MOV     AL,' '                  ; FILL WITH BLANKS
023B                    N4:                                   ; CLEAR_LOOP
023B E8 02A6 R                CALL    N11                     ; CLEAR THE ROW
023E 03 FD                    ADD     DI,BP                   ; POINT TO NEXT LINE
0240 FE CB                    DEC     BL                      ; COUNTER OF LINES TO SCROLL
0242 75 F7                    JNZ     N4                      ; CLEAR_LOOP
0244                    N5:                                   ; SCROLL_END
0244 E8 0000 E                CALL    DDS
0247 80 3E 0049 R 07          CMP     @CRT_MODE,7             ; IS THIS THE BLACK AND WHITE CARD
024C 74 07                    JE      N6                      ; IF SO, SKIP THE MODE RESET
024E A0 0065 R                MOV     AL,@CRT_MODE_SET        ; GET THE VALUE OF THE MODE SET
0251 BA 03D8                  MOV     DX,03D8H                ; ALWAYS SET COLOR CARD PORT
0254 EE                       OUT     DX,AL
0255                    N6:                                   ; VIDEO_RET_HERE
0255 E9 013D R                JMP     VIDEO_RETURN
0258                    N7:                                   ; BLANK_FIELD
0258 8A DE                    MOV     BL,DH                   ; GET ROW COUNT
025A EB DC                    JMP     N3                      ; GO CLEAR THAT AREA
025C                    SCROLL_UP       ENDP
```

SECTION 5

```
                               ;----- HANDLE COMMON SCROLL SET UP HERE

 025C                          SCROLL_POSITION PROC    NEAR
 025C E8 0200 R                        CALL    POSITION                ; CONVERT TO REGEN POINTER
 025F 03 06 004E R                     ADD     AX,@CRT_START           ; OFFSET OF ACTIVE PAGE
 0263 8B F8                            MOV     DI,AX                   ; TO ADDRESS FOR SCROLL
 0265 8B F0                            MOV     SI,AX                   ; FROM ADDRESS FOR SCROLL
 0267 2B D1                            SUB     DX,CX                   ; DX = #ROWS, #COLS IN BLOCK
 0269 FE C6                            INC     DH
 026B FE C2                            INC     DL                      ; INCREMENT FOR 0 ORIGIN
 026D 32 ED                            XOR     CH,CH                   ; SET HIGH BYTE OF COUNT TO ZERO
 026F 8B 2E 004A R                     MOV     BP,@CRT_COLS            ; GET NUMBER OF COLUMNS IN DISPLAY
 0273 03 ED                            ADD     BP,BP                   ; TIMES 2 FOR ATTRIBUTE BYTE
 0275 A0 004A R                        MOV     AL,BYTE PTR @CRT_COLS   ; GET CHARACTERS PER LINE COUNT
 0278 F6 E3                            MUL     BL                      ; DETERMINE OFFSET TO FROM ADDRESS
 027A 03 C0                            ADD     AX,AX                   ; *2 FOR ATTRIBUTE BYTE
 027C 50                               PUSH    AX                      ; SAVE LINE COUNT
 027D A0 0049 R                        MOV     AL,@CRT_MODE            ; GET CURRENT MODE
 0280 06                               PUSH    ES                      ; ESTABLISH ADDRESSING TO REGEN BUFFER
 0281 1F                               POP     DS                      ;  FOR BOTH POINTERS
 0282 3C 02                            CMP     AL,2                    ; TEST FOR COLOR CARD SPECIAL CASES HERE
 0284 72 13                            JB      N9                      ; HAVE TO HANDLE 80X25 SEPARATELY
 0286 3C 03                            CMP     AL,3
 0288 77 0F                            JA      N9
                               ;-----                                  ; 80X25 COLOR CARD SCROLL
 028A 52                               PUSH    DX
 028B BA 03DA                          MOV     DX,3DAH                 ; GUARANTEED TO BE COLOR CARD HERE
 028E                          N8:                                     ; WAIT_DISP_ENABLE
 028E EC                               IN      AL,DX                   ; GET PORT
 028F A8 08                            TEST    AL,RVRT                 ; WAIT FOR VERTICAL RETRACE
 0291 74 FB                            JZ      N8                      ; WAIT_DISP_ENABLE
 0293 B0 25                            MOV     AL,25H
 0295 B2 D8                            MOV     DL,0D8H                 ; ADDRESS CONTROL PORT
 0297 EE                               OUT     DX,AL                   ; TURN OFF VIDEO DURING VERTICAL RETRACE
 0298 5A                               POP     DX
 0299                          N9:
 0299 58                               POP     AX                      ; RESTORE LINE COUNT
 029A 0A DB                            OR      BL,BL                   ; 0 SCROLL MEANS BLANK FIELD
 029C C3                               RET                             ; RETURN WITH FLAGS SET
 029D                          SCROLL_POSITION ENDP

                               ;-----  MOVE_ROW
 029D                          N10     PROC    NEAR
 029D 8A CA                            MOV     CL,DL                   ; GET # OF COLS TO MOVE
 029F 56                               PUSH    SI
 02A0 57                               PUSH    DI                      ; SAVE START ADDRESS
 02A1 F3/ A5                           REP     MOVSW                   ; MOVE THAT LINE ON SCREEN
 02A3 5F                               POP     DI
 02A4 5E                               POP     SI                      ; RECOVER ADDRESSES
 02A5 C3                               RET
 02A6                          N10     ENDP

                               ;-----  CLEAR_ROW
 02A6                          N11     PROC    NEAR
 02A6 8A CA                            MOV     CL,DL                   ; GET # COLUMNS TO CLEAR
 02A8 57                               PUSH    DI
 02A9 F3/ AB                           REP     STOSW                   ; STORE THE FILL CHARACTER
 02AB 5F                               POP     DI
 02AC C3                               RET
 02AD                          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
                               ; OUTPUT
                               ;       NONE -- SCREEN IS SCROLLED
                               ;----------------------------------------
 02AD                          SCROLL_DOWN     PROC    NEAR
 02AD FD                               STD                             ; DIRECTION FOR SCROLL DOWN
 02AE E8 02EA R                        CALL    TEST_LINE_COUNT         ;
 02B1 80 FC 04                         CMP     AH,4                    ; TEST FOR GRAPHICS
 02B4 72 08                            JC      N12
 02B6 80 FC 07                         CMP     AH,7                    ; TEST FOR BW CARD
 02B9 74 03                            JE      N12
 02BB E9 0503 R                        JMP     GRAPHICS_DOWN
 02BE                          N12:                                    ; CONTINUE_DOWN
 02BE 53                               PUSH    BX                      ; SAVE ATTRIBUTE IN BH
 02BF 8B C2                            MOV     AX,DX                   ; LOWER RIGHT CORNER
 02C1 E8 025C R                        CALL    SCROLL_POSITION         ; GET REGEN LOCATION
 02C4 74 20                            JZ      N16
 02C6 2B F0                            SUB     SI,AX                   ; SI IS FROM ADDRESS
 02C8 8A E6                            MOV     AH,DH                   ; GET TOTAL # ROWS
 02CA 2A E3                            SUB     AH,BL                   ; COUNT TO MOVE IN SCROLL
 02CC                          N13:
 02CC E8 029D R                        CALL    N10                     ; MOVE ONE ROW
 02CF 2B F5                            SUB     SI,BP
 02D1 2B FD                            SUB     DI,BP
 02D3 FE CC                            DEC     AH
 02D5 75 F5                            JNZ     N13
 02D7                          N14:
 02D7 58                               POP     AX                      ; RECOVER ATTRIBUTE IN AH
 02D8 B0 20                            MOV     AL,' '
 02DA                          N15:
 02DA E8 02A6 R                        CALL    N11                     ; CLEAR ONE ROW
 02DD 2B FD                            SUB     DI,BP                   ; GO TO NEXT ROW
 02DF FE CB                            DEC     BL
 02E1 75 F7                            JNZ     N15
 02E3 E9 0244 R                        JMP     N5                      ; SCROLL_END
 02E6                          N16:
 02E6 8A DE                            MOV     BL,DH
 02E8 EB ED                            JMP     N14
 02EA                          SCROLL_DOWN     ENDP
```

```
                                PAGE
                                ;-----  IF  AMOUNT OF LINES TO BE SCROLLED = AMOUNT OF LINES IN WINDOW
                                ;-----   THEN  ADJUST AL; ELSE  RETURN;

 02EA                           TEST_LINE_COUNT PROC    NEAR

 02EA 8A D8                             MOV     BL,AL                   ; SAVE LINE COUNT IN BL
 02EC 0A C0                             OR      AL,AL                   ; TEST IF AL IS ALREADY ZERO
 02EE 74 0E                             JZ      BL_SET                  ; IF IT IS THEN RETURN...
 02F0 50                                PUSH    AX                      ; SAVE AX
 02F1 8A C6                             MOV     AL,DH                   ; SUBTRACT LOWER ROW FROM UPPER ROW
 02F3 2A C5                             SUB     AL,CH
 02F5 FE C0                             INC     AL                      ; ADJUST DIFFERENCE BY 1
 02F7 3A C3                             CMP     AL,BL                   ; LINE COUNT = AMOUNT OF ROWS IN WINDOW?
 02F9 58                                POP     AX                      ; RESTORE AX
 02FA 75 02                             JNE     BL_SET                  ; IF NOT THEN WE'RE ALL SET
 02FC 2A DB                             SUB     BL,BL                   ; OTHERWISE SET BL TO ZERO
 02FE                           BL_SET:
 02FE C3                                RET                             ; RETURN
 02FF                           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) = CHARACTER READ                                               :
                                ;       (AH) = ATTRIBUTE READ                                               :
                                ;---------------------------------------------------------------------------
                                        ASSUME  DS:DATA,ES:DATA

 02FF                           READ_AC_CURRENT PROC    NEAR
 02FF 80 FC 04                          CMP     AH,4                    ; IS THIS GRAPHICS
 0302 72 08                             JC      P10

 0304 80 FC 07                          CMP     AH,7                    ; IS THIS BW CARD
 0307 74 03                             JE      P10

 0309 E9 063E R                         JMP     GRAPHICS_READ
 030C                           P10:                                    ;  READ_AC_CONTINUE
 030C E8 0328 R                         CALL    FIND_POSITION           ; GET REGEN LOCATION AND PORT ADDRESS
 030F 8B F7                             MOV     SI,DI                   ; ESTABLISH ADDRESSING IN SI
 0311 06                                PUSH    ES                      ; GET REGEN SEGMENT FOR QUICK ACCESS
 0312 1F                                POP     DS

                                ;-----  WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80

 0313 0A DB                             OR      BL,BL                   ; CHECK MODE FLAG FOR COLOR CARD IN 80
 0315 75 0D                             JNZ     P13                     ; ELSE SKIP RETRACE WAIT - DO FAST READ
 0317                           P11:                                    ;  WAIT FOR HORZ RETRACE LOW OR VERTICAL
 0317 FB                                STI                             ; ENABLE INTERRUPTS FIRST
 0318 90                                NOP                             ; ALLOW FOR SMALL INTERRUPT WINDOW
 0319 FA                                CLI                             ; BLOCK INTERRUPTS FOR SINGLE LOOP
 031A EC                                IN      AL,DX                   ; GET STATUS FROM THE ADAPTER
 031B A8 01                             TEST    AL,RHRZ                 ; IS HORIZONTAL RETRACE LOW
 031D 75 F8                             JNZ     P11                     ; WAIT UNTIL IT IS
 031F                           P12:                                    ;  NOW WAIT FOR EITHER RETRACE HIGH
 031F EC                                IN      AL,DX                   ; GET STATUS
 0320 A8 09                             TEST    AL,RVRT+RHRZ            ; IS HORIZONTAL OR VERTICAL RETRACE HIGH
 0322 74 FB                             JZ      P12                     ; WAIT UNTIL EITHER IS ACTIVE
 0324                           P13:
 0324 AD                                LODSW                           ; GET THE CHARACTER AND ATTRIBUTE
 0325 E9 013D R                         JMP     VIDEO_RETURN            ;  EXIT WITH (AX)

 0328                           READ_AC_CURRENT ENDP



 0328                           FIND_POSITION   PROC    NEAR            ;       SETUP FOR BUFFER READ OR WRITE
 0328 86 E3                             XCHG    AH,BL                   ; SWAP MODE TYPE WITH ATTRIBUTE
 032A 8B E8                             MOV     BP,AX                   ; SAVE CHARACTER/ATTR IN (BP) REGISTER
 032C 80 EB 02                          SUB     BL,2                    ; CONVERT DISPLAY MODE TYPE TO A
 032F D0 EB                             SHR     BL,1                    ;  ZERO VALUE FOR COLOR IN 80 COLUMN
 0331 8A C7                             MOV     AL,BH                   ; MOVE DISPLAY PAGE TO LOW BYTE
 0333 98                                CBW                             ; CLEAR HIGH BYTE FOR BYTE OFFSET
 0334 8B F8                             MOV     DI,AX                   ; MOVE DISPLAY PAGE (COUNT) TO WORK REG
 0336 D1 E7                             SAL     DI,1                    ; TIMES 2 FOR WORD OFFSET
 0338 8B 95 0050 R                      MOV     DX,[DI+OFFSET @CURSOR_POSN]     ; GET ROW/COLUMN OF THAT PAGE
 033C 74 09                             JZ      P21                     ; SKIP BUFFER ADJUSTMENT IF PAGE ZERO

 033E 33 FF                             XOR     DI,DI                   ; ELSE SET BUFFER START ADDRESS TO ZERO
 0340                           P20:
 0340 03 3E 004C R                      ADD     DI,@CRT_LEN             ; ADD LENGTH OF BUFFER FOR ONE PAGE
 0344 48                                DEC     AX                      ; DECREMENT PAGE COUNT
 0345 75 F9                             JNZ     P20                     ; LOOP TILL PAGE COUNT EXHAUSTED

 0347                           P21:                                    ; DETERMINE LOCATION IN REGEN IN PAGE
 0347 A0 004A R                         MOV     AL,BYTE PTR @CRT_COLS   ; GET COLUMNS PER ROW COUNT
 034A F6 E6                             MUL     DH                      ; DETERMINE BYTES TO ROW
 034C 32 F6                             XOR     DH,DH
 034E 03 C2                             ADD     AX,DX                   ; ADD IN COLUMN VALUE
 0350 D1 E0                             SAL     AX,1                    ; * 2 FOR ATTRIBUTE BYTES
 0352 03 F8                             ADD     DI,AX                   ; ADD LOCATION TO START OF REGEN PAGE
 0354 8B 16 0063 R                      MOV     DX,@ADDR_6845           ; GET BASE ADDRESS OF ACTIVE DISPLAY
 0358 83 C2 06                          ADD     DX,6                    ;  DX= STATUS PORT ADDRESS OF ADAPTER
 035B C3                                RET                             ;  BP= ATTRIBUTE/CHARACTER (FROM BL/AL)
                                                                        ;  DI= POSITION (OFFSET IN REGEN BUFFER)
 035C                           FIND_POSITION   ENDP                    ;  BL= MODE FLAG (ZERO FOR 80X25 COLOR)
```

SECTION 5

```
                                PAGE
                                ;--------------------------------------------------------------------------------
                                ; 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                                                                         :
                                ;       DISPLAY REGEN BUFFER UPDATED                                             :
                                ;--------------------------------------------------------------------------------

 035C                           WRITE_AC_CURRENT        PROC    NEAR
 035C 80 FC 04                          CMP     AH,4                    ; IS THIS GRAPHICS
 035F 72 08                             JC      P30
 0361 80 FC 07                          CMP     AH,7                    ; IS THIS BW CARD
 0364 74 03                             JE      P30
 0366 E9 058A R                         JMP     GRAPHICS_WRITE
 0369                           P30:                                    ;  WRITE_AC_CONTINUE
 0369 E8 0328 R                         CALL    FIND_POSITION           ; GET REGEN LOCATION AND PORT ADDRESS
                                                                        ; ADDRESS IN (DI) REGISTER
 036C 0A DB                             OR      BL,BL                   ; CHECK MODE FLAG FOR COLOR CARD AT 80
 036E 74 06                             JZ      P32                     ; SKIP TO RETRACE WAIT IF COLOR AT 80

 0370 95                                XCHG    AX,BP                   ; GET THE ATTR/CHAR SAVED FOR FAST WRITE
 0371 F3/ AB                            REP     STOSW                   ; STRING WRITE THE ATTRIBUTE & CHARACTER
 0373 EB 16                             JMP     SHORT  P35              ; EXIT FAST WRITE ROUTINE

                                ;-----  WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80

 0375                           P31:                                    ;  LOOP FOR EACH ATTR/CHAR WRITE
 0375 95                                XCHG    BP,AX                   ; PLACE ATTR/CHAR BACK IN SAVE REGISTER
 0376                           P32:                                    ;  WAIT FOR HORZ RETRACE LOW OR VERTICAL
 0376 FB                                STI                             ; ENABLE INTERRUPTS FIRST
 0377 90                                NOP                             ; ALLOW FOR INTERRUPT WINDOW
 0378 FA                                CLI                             ; BLOCK INTERRUPTS FOR SINGLE LOOP
 0379 EC                                IN      AL,DX                   ; GET STATUS FROM THE ADAPTER
 037A A8 08                             TEST    AL,RVRT                 ; CHECK FOR VERTICAL RETRACE FIRST
 037C 75 09                             JNZ     P34                     ; DO FAST WRITE NOW IF VERTICAL RETRACE
 037E A8 01                             TEST    AL,RHRZ                 ; IS HORIZONTAL RETRACE LOW THEN
 0380 75 F4                             JNZ     P32                     ; WAIT UNTIL IT IS
 0382                           P33:                                    ;  WAIT FOR EITHER RETRACE HIGH
 0382 EC                                IN      AL,DX                   ; GET STATUS AGAIN
 0383 A8 09                             TEST    AL,RVRT+RHRZ            ; IS HORIZONTAL OR VERTICAL RETRACE HIGH
 0385 74 FB                             JZ      P33                     ; WAIT UNTIL EITHER IS ACTIVE
 0387                           P34:
 0387 95                                XCHG    AX,BP                   ; GET THE ATTR/CHAR SAVED IN (BP)
 0388 AB                                STOSW                           ; WRITE THE ATTRIBUTE AND CHARACTER
 0389 E2 EA                             LOOP    P31                     ; AS MANY TIMES AS REQUESTED - TILL CX=0
 038B                           P35:
 038B E9 013D R                         JMP     VIDEO_RETURN            ; EXIT

 038E                           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                                                                         :
                                ;       DISPLAY REGEN BUFFER UPDATED                                             :
                                ;--------------------------------------------------------------------------------

 038E                           WRITE_C_CURRENT PROC    NEAR
 038E 80 FC 04                          CMP     AH,4                    ; IS THIS GRAPHICS
 0391 72 08                             JC      P40
 0393 80 FC 07                          CMP     AH,7                    ; IS THIS BW CARD
 0396 74 03                             JE      P40
 0398 E9 058A R                         JMP     GRAPHICS_WRITE
 039B                           P40:
 039B E8 0328 R                         CALL    FIND_POSITION           ; GET REGEN LOCATION AND PORT ADDRESS
                                                                        ; ADDRESS OF LOCATION IN (DI)

                                ;-----  WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80

 039E                           P41:                                    ;  WAIT FOR HORZ RETRACE LOW OR VERTICAL
 039E FB                                STI                             ; ENABLE INTERRUPTS FIRST
 039F 0A DB                             OR      BL,BL                   ; CHECK MODE FLAG FOR COLOR CARD IN 80
 03A1 75 0F                             JNZ     P43                     ; ELSE SKIP RETRACE WAIT - DO FAST WRITE
 03A3 FA                                CLI                             ; BLOCK INTERRUPTS FOR SINGLE LOOP
 03A4 EC                                IN      AL,DX                   ; GET STATUS FROM THE ADAPTER
 03A5 A8 08                             TEST    AL,RVRT                 ; CHECK FOR VERTICAL RETRACE FIRST
 03A7 75 09                             JNZ     P43                     ; DO FAST WRITE NOW IF VERTICAL RETRACE
 03A9 A8 01                             TEST    AL,RHRZ                 ; IS HORIZONTAL RETRACE LOW THEN
 03AB 75 F1                             JNZ     P41                     ; WAIT UNTIL IT IS
 03AD                           P42:                                    ;  WAIT FOR EITHER RETRACE HIGH
 03AD EC                                IN      AL,DX                   ; GET STATUS AGAIN
 03AE A8 09                             TEST    AL,RVRT+RHRZ            ; IS HORIZONTAL OR VERTICAL RETRACE HIGH
 03B0 74 FB                             JZ      P42                     ; WAIT UNTIL EITHER RETRACE ACTIVE
 03B2                           P43:
 03B2 8B C5                             MOV     AX,BP                   ; GET THE CHARACTER SAVE IN (BP)
 03B4 AA                                STOSB                           ; PUT THE CHARACTER INTO REGEN BUFFER
 03B5 47                                INC     DI                      ; BUMP POINTER PAST ATTRIBUTE
 03B6 E2 E6                             LOOP    P41                     ;  AS MANY TIMES AS REQUESTED

 03B8 E9 013D R                         JMP     VIDEO_RETURN

 03BB                           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 (ACTIVE PAGE)                                        :
                              ;       (CX) = COUNT OF CHARACTERS TO WRITE, IF (CX) = 0 THEN RETURN             :
                              ;       (DX) = CURSOR POSITION FOR START OF STRING WRITE                         :
                              ;       (BL) = ATTRIBUTE OF CHARACTER TO WRITE IF (AL) = 0  OR  (AL) = 1         :
                              ;       (BP) = SOURCE STRING OFFSET                                              :
                              ;       [0E] = SOURCE STRING SEGMENT (FOR USE IN (ES) IN STACK +14)              :
                              ; OUTPUT                                                                         :
                              ;       NONE                                                                     :
                              ;--------------------------------------------------------------------------------
 03BB                         WRITE_STRING    PROC    NEAR
 03BB 55                              PUSH    BP                      ; SAVE BUFFER OFFSET (BP) IN STACK
 03BC 8B EC                           MOV     BP,SP                   ; GET POINTER TO STACKED REGISTERS
 03BE 8E 46 10                        MOV     ES,[BP]+14+2            ; RECOVER ENTRY (ES) SEGMENT REGISTER
 03C1 5D                              POP     BP                      ; RESTORE BUFFER OFFSET
 03C2 98                              CBW                             ; CLEAR (AH) REGISTER
 03C3 8B F8                           MOV     DI,AX                   ; SAVE (AL) COMMAND IN (DI) REGISTER
 03C5 3C 04                           CMP     AL,04                   ; TEST FOR INVALID WRITE STRING OPTION
 03C7 73 73                           JNB     P59                     ; IF OPTION INVALID THEN RETURN

 03C9 E3 71                           JCXZ    P59                     ; IF ZERO LENGTH STRING THEN RETURN

 03CB 8B F3                           MOV     SI,BX                   ; SAVE CURRENT CURSOR PAGE
 03CD 8A DF                           MOV     BL,BH                   ; MOVE PAGE TO LOW BYTE
 03CF 32 FF                           XOR     BH,BH                   ; CLEAR HIGH BYTE
 03D1 87 F3                           XCHG    SI,BX                   ; MOVE OFFSET AND RESTORE PAGE REGISTER
 03D3 D1 E6                           SAL     SI,1                    ; CONVERT TO PAGE OFFSET  (SI= PAGE)
 03D5 FF B4 0050 R                    PUSH    [SI+OFFSET @CURSOR_POSN]; SAVE CURRENT CURSOR POSITION IN STACK
 03D9 B8 0200                         MOV     AX,0200H                ; SET NEW CURSOR POSITION
 03DC CD 10                           INT     10H
 03DE                         P50:
 03DE 26: 8A 46 00                    MOV     AL,ES:[BP]              ; GET CHARACTER FROM INPUT STRING
 03E2 45                              INC     BP                      ; BUMP POINTER TO CHARACTER

                              ;-----  TEST FOR SPECIAL CHARACTER'S

 03E3 3C 08                           CMP     AL,08H                  ; IS IT A BACKSPACE
 03E5 74 0C                           JE      P51                     ; BACK_SPACE
 03E7 3C 0D                           CMP     AL,CR                   ; IS IT CARRIAGE RETURN
 03E9 74 08                           JE      P51                     ; CAR_RET
 03EB 3C 0A                           CMP     AL,LF                   ; IS IT A LINE FEED
 03ED 74 04                           JE      P51                     ; LINE_FEED
 03EF 3C 07                           CMP     AL,07H                  ; IS IT A BELL
 03F1 75 0A                           JNE     P52                     ; IF NOT THEN DO WRITE CHARACTER
 03F3                         P51:
 03F3 B4 0E                           MOV     AH,0EH                  ; TTY_CHARACTER WRITE
 03F5 CD 10                           INT     10H                     ;  WRITE TTY CHARACTER TO THE CRT
 03F7 8B 94 0050 R                    MOV     DX,[SI+OFFSET @CURSOR_POSN] ; GET CURRENT CURSOR POSITION
 03FB EB 2D                           JMP     SHORT P54                   ; SET CURSOR POSITION AND CONTINUE

 03FD                         P52:
 03FD 51                              PUSH    CX
 03FE 53                              PUSH    BX
 03FF B9 0001                         MOV     CX,1                    ; SET CHARACTER WRITE AMOUNT TO ONE
 0402 83 FF 02                        CMP     DI,2                    ; IS THE ATTRIBUTE IN THE STRING
 0405 72 05                           JB      P53                     ; IF NOT THEN SKIP
 0407 26: 8A 5E 00                    MOV     BL,ES:[BP]              ; ELSE GET NEW ATTRIBUTE
 040B 45                              INC     BP                      ; BUMP STRING POINTER
 040C                         P53:
 040C B4 09                           MOV     AH,09H                  ;  GOT_CHARACTER
 040E CD 10                           INT     10H                     ; WRITE CHARACTER TO THE CRT
 0410 5B                              POP     BX                      ; RESTORE REGISTERS
 0411 59                              POP     CX
 0412 FE C2                           INC     DL                      ; INCREMENT COLUMN COUNTER
 0414 3A 16 004A R                    CMP     DL,BYTE PTR @CRT_COLS   ; IF COLS ARE WITHIN RANGE FOR THIS MODE
 0418 72 10                           JB      P54                     ;    THEN GO TO COLUMNS SET
 041A FE C6                           INC     DH                      ; BUMP ROW COUNTER BY ONE
 041C 2A D2                           SUB     DL,DL                   ; SET COLUMN COUNTER TO ZERO
 041E 80 FE 19                        CMP     DH,25                   ; IF ROWS ARE LESS THAN 25 THEN
 0421 72 07                           JB      P54                     ;    GO TO ROWS_COLUMNS_SET

 0423 B8 0E0A                         MOV     AX,0E0AH                ; ELSE SCROLL SCREEN ONE LINE
 0426 CD 10                           INT     10H                     ; RESET ROW COUNTER TO 24
 0428 FE CE                           DEC     DH
 042A                         P54:                                    ;  ROW_COLUMNS_SET
 042A B8 0200                         MOV     AX,0200H                ; SET NEW CURSOR POSITION COMMAND
 042D CD 10                           INT     10H                     ; ESTABLISH NEW CURSOR POSITION
 042F E2 AD                           LOOP    P50                     ; DO IT ONCE MORE UNTIL (CX) = ZERO

 0431 5A                              POP     DX                      ; RESTORE OLD CURSOR COORDINATES
 0432 97                              XCHG    AX,DI                   ; RECOVER WRITE STRING COMMAND
 0433 A8 01                           TEST    AL,01H                  ; IF CURSOR WAS NOT TO BE MOVED THEN
 0435 75 05                           JNZ     P59                     ;  THEN EXIT WITHOUT RESETTING OLD VALUE
 0437 B8 0200                         MOV     AX,0200H                ; ELSE RESTORE OLD CURSOR POSITION
 043A CD 10                           INT     10H
 043C                         P59:                                    ;  DONE - EXIT WRITE STRING
 043C E9 013D R                       JMP     VIDEO_RETURN            ; RETURN TO CALLER

 043F                         WRITE_STRING    ENDP
```

SECTION 5

```
                        PAGE
                        ;--------------------------------------------
                        ; READ DOT  -- WRITE DOT
                        ; THESE ROUTINES WILL WRITE A DOT, OR READ THE
                        ;  DOT AT THE INDICATED LOCATION
                        ; ENTRY --
                        ;   DX = ROW (0-199)     (THE ACTUAL VALUE DEPENDS ON THE MODE)
                        ;   CX = COLUMN ( 0-639) ( THE VALUES ARE NOT RANGE CHECKED )
                        ;   AL = DOT VALUE TO WRITE (1,2 OR 4 BITS DEPENDING ON MODE,
                        ;        REQUIRED FOR WRITE DOT ONLY, RIGHT JUSTIFIED)
                        ;        BIT 7 OF AL = 1 INDICATES XOR THE VALUE INTO THE LOCATION
                        ;   DS = DATA SEGMENT
                        ;   ES = REGEN SEGMENT
                        ;
                        ; EXIT
                        ;        AL = DOT VALUE READ, RIGHT JUSTIFIED, READ ONLY
                        ;--------------------------------------------
                                ASSUME  DS:DATA,ES:DATA
043F                    READ_DOT        PROC    NEAR
043F E8 0473 R                  CALL    R3                      ; DETERMINE BYTE POSITION OF DOT
0442 26: 8A 04                  MOV     AL,ES:[SI]              ; GET THE BYTE
0445 22 C4                      AND     AL,AH                   ; MASK OFF THE OTHER BITS IN THE BYTE
0447 D2 E0                      SHL     AL,CL                   ; LEFT JUSTIFY THE VALUE
0449 8A CE                      MOV     CL,DH                   ; GET NUMBER OF BITS IN RESULT
044B D2 C0                      ROL     AL,CL                   ; RIGHT JUSTIFY THE RESULT
044D E9 013D R                  JMP     VIDEO_RETURN            ; RETURN FROM VIDEO I/O
0450                    READ_DOT        ENDP

0450                    WRITE_DOT       PROC    NEAR
0450 50                         PUSH    AX                      ; SAVE DOT VALUE
0451 50                         PUSH    AX                      ;  TWICE
0452 E8 0473 R                  CALL    R3                      ; DETERMINE BYTE POSITION OF THE DOT
0455 D2 E8                      SHR     AL,CL                   ; SHIFT TO SET UP THE BITS FOR OUTPUT
0457 22 C4                      AND     AL,AH                   ; STRIP OFF THE OTHER BITS
0459 26: 8A 0C                  MOV     CL,ES:[SI]              ; GET THE CURRENT BYTE
045C 5B                         POP     BX                      ; RECOVER XOR FLAG
045D F6 C3 80                   TEST    BL,80H                  ; IS IT ON
0460 75 0D                      JNZ     R2                      ; YES,cXOR THE DOT
0462 F6 D4                      NOT     AH                      ; SET MASK TO REMOVE THE INDICATED BITS
0464 22 CC                      AND     CL,AH
0466 0A C1                      OR      AL,CL                   ; OR IN THE NEW VALUE OF THOSE BITS
0468                    R1:                                     ; FINISH_DOT
0468 26: 88 04                  MOV     ES:[SI],AL              ; RESTORE THE BYTE IN MEMORY
046B 58                         POP     AX
046C E9 013D R                  JMP     VIDEO_RETURN            ; RETURN FROM VIDEO I/O
046F                    R2:                                     ; XOR_DOT
046F 32 C1                      XOR     AL,CL                   ; EXCLUSIVE OR THE DOTS
0471 EB F5                      JMP     R1                      ; FINISH UP THE WRITING
0473                    WRITE_DOT       ENDP
                        ;--------------------------------------------
                        ; THIS SUBROUTINE DETERMINES THE REGEN BYTE LOCATION OF THE
                        ; INDICATED ROW COLUMN VALUE IN GRAPHICS MODE.
                        ; ENTRY --
                        ;  DX = ROW VALUE (0-199)
                        ;  CX = COLUMN VALUE (0-639)
                        ; EXIT --
                        ;  SI = OFFSET INTO REGEN BUFFER FOR BYTE OF INTEREST
                        ;  AH = MASK TO STRIP OFF THE BITS OF INTEREST
                        ;  CL = BITS TO SHIFT TO RIGHT JUSTIFY THE MASK IN AH
                        ;  DH = # BITS IN RESULT
                        ;  BX = MODIFIED
                        ;--------------------------------------------
0473                    R3      PROC    NEAR

                        ;-----  DETERMINE 1ST BYTE IN INDICATED ROW BY MULTIPLYING ROW VALUE BY 40
                        ;-----   ( LOW BIT OF ROW DETERMINES EVEN/ODD, 80 BYTES/ROW )

0473 96                         XCHG    SI,AX                   ; WILL SAVE AL AND AH DURING OPERATION
0474 B0 28                      MOV     AL,40
0476 F6 E2                      MUL     DL                      ; AX= ADDRESS OF START OF INDICATED ROW
0478 A8 08                      TEST    AL,008H                 ; TEST FOR EVEN/ODD ROW CALCULATED
047A 74 03                      JZ      R4                      ; JUMP IF EVEN ROW
047C 05 1FD8                    ADD     AX,2000H-40             ; OFFSET TO LOCATION OF ODD ROWS ADJUST
047F                    R4:                                     ; EVEN_ROW
047F 96                         XCHG    SI,AX                   ; MOVE POINTER TO (SI) AND RECOVER (AX)
0480 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 )

0482 BB 02C0                    MOV     BX,2C0H
0485 B9 0302                    MOV     CX,302H                 ; SET PARMS FOR MED RES
0488 80 3E 0049 R 06            CMP     @CRT_MODE,6
048D 72 06                      JC      R5                      ; HANDLE IF MED RES
048F BB 0180                    MOV     BX,180H
0492 B9 0703                    MOV     CX,703H                 ; SET PARMS FOR HIGH RES

                        ;-----  DETERMINE BIT OFFSET IN BYTE FROM COLUMN MASK
0495                    R5:
0495 22 EA                      AND     CH,DL                   ; ADDRESS OF PEL WITHIN BYTE TO CH

                        ;-----  DETERMINE BYTE OFFSET FOR THIS LOCATION IN COLUMN

0497 D3 EA                      SHR     DX,CL                   ; SHIFT BY CORRECT AMOUNT
0499 03 F2                      ADD     SI,DX                   ; INCREMENT THE POINTER
049B 8A F7                      MOV     DH,BH                   ; GET THE # OF BITS IN RESULT TO DH

                        ;-----  MULTIPLY BH (VALID BITS IN BYTE) BY CH (BIT OFFSET)

049D 2A C9                      SUB     CL,CL                   ; ZERO INTO STORAGE LOCATION
049F                    R6:
049F D0 C8                      ROR     AL,1                    ; LEFT JUSTIFY VALUE IN AL (FOR WRITE)
04A1 02 CD                      ADD     CL,CH                   ; ADD IN THE BIT OFFSET VALUE
04A3 FE CF                      DEC     BH                      ; LOOP CONTROL
04A5 75 F8                      JNZ     R6                      ; ON EXIT, CL HAS COUNT TO RESTORE BITS
04A7 8A E3                      MOV     AH,BL                   ;  GET MASK TO AH
04A9 D2 EC                      SHR     AH,CL                   ;  MOVE THE MASK TO CORRECT LOCATION
04AB C3                         RET                             ;  RETURN WITH EVERYTHING SET UP
04AC                    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
                                ;-----------------------------------------------------
04AC                            GRAPHICS_UP     PROC    NEAR
04AC 8A D8                              MOV     BL,AL                   ; SAVE LINE COUNT IN BL
04AE 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

04B0 E8 06EC R                          CALL    GRAPH_POSN
04B3 8B F8                              MOV     DI,AX                   ; SAVE RESULT AS DESTINATION ADDRESS

                                ;-----  DETERMINE SIZE OF WINDOW

04B5 2B D1                              SUB     DX,CX
04B7 81 C2 0101                         ADD     DX,101H                 ; ADJUST VALUES
04BB D0 E6                              SAL     DH,1                    ; MULTIPLY ROWS BY 4 AT 8 VERT DOTS/CHAR
04BD D0 E6                              SAL     DH,1                    ;  AND EVEN/ODD ROWS

                                ;-----  DETERMINE CRT MODE

04BF 80 3E 0049 R 06                    CMP     @CRT_MODE,6             ; TEST FOR MEDIUM RES
04C4 73 04                              JNC     R7                      ; FIND_SOURCE

                                ;-----  MEDIUM RES UP
04C6 D0 E2                              SAL     DL,1                    ; # COLUMNS * 2, SINCE 2 BYTES/CHAR
04C8 D1 E7                              SAL     DI,1                    ; OFFSET *2 SINCE 2 BYTES/CHAR

                                ;-----  DETERMINE THE SOURCE ADDRESS IN THE BUFFER
04CA                            R7:                                     ; FIND_SOURCE
04CA 06                                 PUSH    ES                      ; GET SEGMENTS BOTH POINTING TO REGEN
04CB 1F                                 POP     DS
04CC 2A ED                              SUB     CH,CH                   ; ZERO TO HIGH OF COUNT REGISTER
04CE D0 E3                              SAL     BL,1                    ; MULTIPLY NUMBER OF LINES BY 4
04D0 D0 E3                              SAL     BL,1
04D2 74 2B                              JZ      R11                     ; IF ZERO, THEN BLANK ENTIRE FIELD
04D4 B0 50                              MOV     AL,80                   ; 80 BYTES/ROW
04D6 F6 E3                              MUL     BL                      ; DETERMINE OFFSET TO SOURCE
04D8 8B F7                              MOV     SI,DI                   ; SET UP SOURCE
04DA 03 F0                              ADD     SI,AX                   ;  ADD IN OFFSET TO IT
04DC 8A E6                              MOV     AH,DH                   ; NUMBER OF ROWS IN FIELD
04DE 2A E3                              SUB     AH,BL                   ; DETERMINE NUMBER TO MOVE

                                ;-----  LOOP THROUGH, MOVING ONE ROW AT A TIME, BOTH EVEN AND ODD FIELDS
04E0                            R8:                                     ; ROW_LOOP
04E0 E8 0560 R                          CALL    R17                     ; MOVE ONE ROW
04E3 81 EE 1FB0                         SUB     SI,2000H-80             ; MOVE TO NEXT ROW
04E7 81 EF 1FB0                         SUB     DI,2000H-80
04EB FE CC                              DEC     AH                      ; NUMBER OF ROWS TO MOVE
04ED 75 F1                              JNZ     R8                      ; CONTINUE TILL ALL MOVED

                                ;-----  FILL IN THE VACATED LINE(S)
04EF                            R9:                                     ; CLEAR_ENTRY
04EF 8A C7                              MOV     AL,BH                   ; ATTRIBUTE TO FILL WITH
04F1                            R10:
04F1 E8 0579 R                          CALL    R18                     ; CLEAR THAT ROW
04F4 81 EF 1FB0                         SUB     DI,2000H-80             ; POINT TO NEXT LINE
04F8 FE CB                              DEC     BL                      ; NUMBER OF LINES TO FILL
04FA 75 F5                              JNZ     R10                     ; CLEAR_LOOP
04FC E9 013D R                          JMP     VIDEO_RETURN            ; EVERYTHING DONE

04FF                            R11:                                    ; BLANK_FIELD
04FF 8A DE                              MOV     BL,DH                   ; SET BLANK COUNT TO EVERYTHING IN FIELD
0501 EB EC                              JMP     R9                      ; CLEAR THE FIELD
0503                            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
                                ;-----------------------------------------------------

0503                            GRAPHICS_DOWN   PROC    NEAR
0503 FD                                 STD                             ; SET DIRECTION
0504 8A D8                              MOV     BL,AL                   ; SAVE LINE COUNT IN BL
0506 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

0508 E8 06EC R                          CALL    GRAPH_POSN
050B 8B F8                              MOV     DI,AX                   ; SAVE RESULT AS DESTINATION ADDRESS

                                ;-----  DETERMINE SIZE OF WINDOW

050D 2B D1                              SUB     DX,CX
050F 81 C2 0101                         ADD     DX,101H                 ; ADJUST VALUES
0513 D0 E6                              SAL     DH,1                    ; MULTIPLY ROWS BY 4 AT 8 VERT DOTS/CHAR
0515 D0 E6                              SAL     DH,1                    ;  AND EVEN/ODD ROWS

                                ;-----  DETERMINE CRT MODE

0517 80 3E 0049 R 06                    CMP     @CRT_MODE,6             ; TEST FOR MEDIUM RES
051C 73 05                              JNC     R12                     ; FIND_SOURCE_DOWN
```

SECTION 5

```
                            ;-----  MEDIUM RES DOWN
 051E D0 E2                         SAL     DL,1                    ; # COLUMNS * 2, SINCE 2 BYTES/CHAR
 0520 D1 E7                         SAL     DI,1                    ; OFFSET *2 SINCE 2 BYTES/CHAR
 0522 47                            INC     DI                      ; POINT TO LAST BYTE

                            ;-----  DETERMINE THE SOURCE ADDRESS IN THE BUFFER
 0523                       R12:                                    ; FIND_SOURCE_DOWN
 0523 06                            PUSH    ES                      ; BOTH SEGMENTS TO REGEN
 0524 1F                            POP     DS
 0525 2A ED                         SUB     CH,CH                   ; ZERO TO HIGH OF COUNT REGISTER
 0527 81 C7 00F0                    ADD     DI,240                  ; POINT TO LAST ROW OF PIXELS
 052B D0 E3                         SAL     BL,1                    ; MULTIPLY NUMBER OF LINES BY 4
 052D D0 E3                         SAL     BL,1
 052F 74 2B                         JZ      R16                     ; IF ZERO, THEN BLANK ENTIRE FIELD
 0531 B0 50                         MOV     AL,80                   ; 80 BYTES/ROW
 0533 F6 E3                         MUL     BL                      ; DETERMINE OFFSET TO SOURCE
 0535 8B F7                         MOV     SI,DI                   ; SET UP SOURCE
 0537 2B F0                         SUB     SI,AX                   ;  SUBTRACT THE OFFSET
 0539 8A E6                         MOV     AH,DH                   ; NUMBER OF ROWS IN FIELD
 053B 2A E3                         SUB     AH,BL                   ; DETERMINE NUMBER TO MOVE

                            ;-----  LOOP THROUGH, MOVING ONE ROW AT A TIME, BOTH EVEN AND ODD FIELDS
 053D                       R13:                                    ; ROW_LOOP_DOWN
 053D E8 0560 R                     CALL    R17                     ; MOVE ONE ROW
 0540 81 EE 2050                    SUB     SI,2000H+80             ; MOVE TO NEXT ROW
 0544 81 EF 2050                    SUB     DI,2000H+80
 0548 FE CC                         DEC     AH                      ; NUMBER OF ROWS TO MOVE
 054A 75 F1                         JNZ     R13                     ; CONTINUE TILL ALL MOVED

                            ;-----  FILL IN THE VACATED LINE(S)
 054C                       R14:                                    ; CLEAR_ENTRY_DOWN
 054C 8A C7                         MOV     AL,BH                   ; ATTRIBUTE TO FILL WITH
 054E                       R15:                                    ; CLEAR_LOOP_DOWN
 054E E8 0579 R                     CALL    R18                     ; CLEAR A ROW
 0551 81 EF 2050                    SUB     DI,2000H+80             ; POINT TO NEXT LINE
 0555 FE CB                         DEC     BL                      ; NUMBER OF LINES TO FILL
 0557 75 F5                         JNZ     R15                     ; CLEAR_LOOP_DOWN

 0559 E9 013D R                     JMP     VIDEO_RETURN            ; EVERYTHING DONE

 055C                       R16:                                    ; BLANK_FIELD_DOWN
 055C 8A DE                         MOV     BL,DH                   ; SET BLANK COUNT TO EVERYTHING IN FIELD
 055E EB EC                         JMP     R14                     ; CLEAR THE FIELD
 0560                       GRAPHICS_DOWN   ENDP

                            ;-----  ROUTINE TO MOVE ONE ROW OF INFORMATION

 0560                       R17     PROC    NEAR
 0560 8A CA                         MOV     CL,DL                   ; NUMBER OF BYTES IN THE ROW
 0562 56                            PUSH    SI
 0563 57                            PUSH    DI                      ; SAVE POINTERS
 0564 F3/ A4                        REP     MOVSB                   ; MOVE THE EVEN FIELD
 0566 5F                            POP     DI
 0567 5E                            POP     SI
 0568 81 C6 2000                    ADD     SI,2000H
 056C 81 C7 2000                    ADD     DI,2000H                ; POINT TO THE ODD FIELD
 0570 56                            PUSH    SI
 0571 57                            PUSH    DI                      ; SAVE THE POINTERS
 0572 8A CA                         MOV     CL,DL                   ; COUNT BACK
 0574 F3/ A4                        REP     MOVSB                   ; MOVE THE ODD FIELD
 0576 5F                            POP     DI
 0577 5E                            POP     SI                      ; POINTERS BACK
 0578 C3                            RET                             ; RETURN TO CALLER
 0579                       R17     ENDP

                            ;-----  CLEAR A SINGLE ROW

 0579                       R18     PROC    NEAR
 0579 8A CA                         MOV     CL,DL                   ; NUMBER OF BYTES IN FIELD
 057B 57                            PUSH    DI                      ; SAVE POINTER
 057C F3/ AA                        REP     STOSB                   ; STORE THE NEW VALUE
 057E 5F                            POP     DI                      ; POINTER BACK
 057F 81 C7 2000                    ADD     DI,2000H                ; POINT TO ODD FIELD
 0583 57                            PUSH    DI
 0584 8A CA                         MOV     CL,DL
 0586 F3/ AA                        REP     STOSB                   ; FILL THE ODD FIELD
 0588 5F                            POP     DI
 0589 C3                            RET                             ; RETURN TO CALLER
 058A                       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  DS:DATA,ES:DATA
 058A                       GRAPHICS_WRITE  PROC    NEAR
 058A B4 00                         MOV     AH,0                    ; ZERO TO HIGH OF CODE POINT
 058C 50                            PUSH    AX                      ; SAVE CODE POINT VALUE
```

```
                           ;-----  DETERMINE POSITION IN REGEN BUFFER TO PUT CODE POINTS
058D E8 06E9 R                     CALL    S26                       ; FIND LOCATION IN REGEN BUFFER
0590 8B F8                         MOV     DI,AX                     ; REGEN POINTER IN DI
                           ;-----  DETERMINE REGION TO GET CODE POINTS FROM
0592 58                            POP     AX                        ; RECOVER CODE POINT
0593 3C 80                         CMP     AL,80H                    ; IS IT IN SECOND HALF
0595 73 06                         JAE     S1                        ; YES
                           ;-----  IMAGE IS IN FIRST HALF, CONTAINED IN ROM
0597 BE 0000 E                     MOV     SI,OFFSET CRT_CHAR_GEN    ; OFFSET OF IMAGES
059A 0E                            PUSH    CS                        ; SAVE SEGMENT ON STACK
059B EB 18                         JMP     SHORT S2                  ; DETERMINE_MODE
                           ;-----  IMAGE IS IN SECOND HALF, IN USER MEMORY
059D                       S1:                                       ; EXTEND_CHAR
059D 2C 80                         SUB     AL,80H                    ; ZERO ORIGIN FOR SECOND HALF
059F 1E                            PUSH    DS                        ; SAVE DATA POINTER
05A0 2B F6                         SUB     SI,SI
05A2 8E DE                         MOV     DS,SI                     ; ESTABLISH VECTOR ADDRESSING
                                   ASSUME  DS:ABS0
05A4 C5 36 007C R                  LDS     SI,@EXT_PTR               ; GET THE OFFSET OF THE TABLE
05A8 8C DA                         MOV     DX,DS                     ; GET THE SEGMENT OF THE TABLE
                                   ASSUME  DS:DATA
05AA 1F                            POP     DS                        ; RECOVER DATA SEGMENT
05AB 52                            PUSH    DX                        ; SAVE TABLE SEGMENT ON STACK
05AC 0B D6                         OR      DX,SI                     ; CHECK FOR VALID TABLE DEFINED
05AE 75 05                         JNZ     S2                        ; CONTINUE IF DS:SI NOT 0000:0000
05B0 58                            POP     AX                        ; ELSE SET (AX)= 0000 FOR "NULL"
05B1 BE 0000 E                     MOV     SI,OFFSET CRT_CHAR_GEN    ; POINT TO DEFAULT TABLE OFFSET
05B4 0E                            PUSH    CS                        ;  IN THE CODE SEGMENT
                           ;-----  DETERMINE GRAPHICS MODE IN OPERATION
05B5                       S2:                                       ;        DETERMINE_MODE
05B5 D1 E0                         SAL     AX,1                      ; MULTIPLY CODE POINT VALUE BY 8
05B7 D1 E0                         SAL     AX,1
05B9 D1 E0                         SAL     AX,1
05BB 03 F0                         ADD     SI,AX                     ; SI HAS OFFSET OF DESIRED CODES
05BD 80 3E 0049 R 06               CMP     @CRT_MODE,6
05C2 1F                            POP     DS                        ; RECOVER TABLE POINTER SEGMENT
05C3 72 2C                         JC      S7                        ; TEST FOR MEDIUM RESOLUTION MODE
                           ;-----  HIGH RESOLUTION MODE
05C5                       S3:                                       ; HIGH_CHAR
05C5 57                            PUSH    DI                        ; SAVE REGEN POINTER
05C6 56                            PUSH    SI                        ; SAVE CODE POINTER
05C7 B6 04                         MOV     DH,4                      ; NUMBER OF TIMES THROUGH LOOP
05C9                       S4:
05C9 AC                            LODSB                             ; GET BYTE FROM CODE POINTS
05CA F6 C3 80                      TEST    BL,80H                    ; SHOULD WE USE THE FUNCTION
05CD 75 16                         JNZ     S6                        ;  TO PUT CHAR IN
05CF AA                            STOSB                             ; STORE IN REGEN BUFFER
05D0 AC                            LODSB
05D1                       S5:
05D1 26: 88 85 1FFF                MOV     ES:[DI+2000H-1],AL        ; STORE IN SECOND HALF
05D6 83 C7 4F                      ADD     DI,79                     ; MOVE TO NEXT ROW IN REGEN
05D9 FE CE                         DEC     DH                        ; DONE WITH LOOP
05DB 75 EC                         JNZ     S4
05DD 5E                            POP     SI
05DE 5F                            POP     DI                        ; RECOVER REGEN POINTER
05DF 47                            INC     DI                        ; POINT TO NEXT CHAR POSITION
05E0 E2 E3                         LOOP    S3                        ; MORE CHARS TO WRITE
05E2 E9 013D R                     JMP     VIDEO_RETURN
05E5                       S6:
05E5 26: 32 05                     XOR     AL,ES:[DI]                ; EXCLUSIVE OR WITH CURRENT
05E8 AA                            STOSB                             ; STORE THE CODE POINT
05E9 AC                            LODSB                             ; AGAIN FOR ODD FIELD
05EA 26: 32 85 1FFF                XOR     AL,ES:[DI+2000H-1]
05EF EB E0                         JMP     S5                        ; BACK TO MAINSTREAM
                           ;-----  MEDIUM RESOLUTION WRITE
05F1                       S7:                                       ;  MED_RES_WRITE
05F1 8A D3                         MOV     DL,BL                     ; SAVE HIGH COLOR BIT
05F3 D1 E7                         SAL     DI,1                      ; OFFSET*2 SINCE 2 BYTES/CHAR
                                                                     ; EXPAND BL TO FULL WORD OF COLOR
05F5 80 E3 03                      AND     BL,3                      ; ISOLATE THE COLOR BITS ( LOW 2 BITS )
05F8 B0 55                         MOV     AL,055H                   ; GET BIT CONVERSION MULTIPLIER
05FA F6 E3                         MUL     BL                        ; EXPAND 2 COLOR BITS TO 4 REPLICATIONS
05FC 8A D8                         MOV     BL,AL                     ; PLACE BACK IN WORK REGISTER
05FE 8A F8                         MOV     BH,AL                     ; EXPAND TO 8 REPLICATIONS OF COLOR BITS
0600                       S8:                                       ;  MED_CHAR
0600 57                            PUSH    DI                        ; SAVE REGEN POINTER
0601 56                            PUSH    SI                        ; SAVE THE CODE POINTER
0602 B6 04                         MOV     DH,4                      ; NUMBER OF LOOPS
0604                       S9:
0604 AC                            LODSB                             ; GET CODE POINT
0605 E8 06C0 R                     CALL    S21                       ; DOUBLE UP ALL THE BITS
0608 23 C3                         AND     AX,BX                     ; CONVERT TO FOREGROUND COLOR ( 0 BACK )
060A 86 E0                         XCHG    AH,AL                     ; SWAP HIGH/LOW BYTES FOR WORD MOVE
060C F6 C2 80                      TEST    DL,80H                    ; IS THIS XOR FUNCTION
060F 74 03                         JZ      S10                       ; NO, STORE IT IN AS IT IS
0611 26: 33 05                     XOR     AX,ES:[DI]                ; DO FUNCTION WITH LOW/HIGH
0614                       S10:
0614 26: 89 05                     MOV     ES:[DI],AX                ; STORE FIRST BYTE HIGH, SECOND LOW
0617 AC                            LODSB                             ; GET CODE POINT
0618 E8 06C0 R                     CALL    S21
061B 23 C3                         AND     AX,BX                     ; CONVERT TO COLOR
061D 86 E0                         XCHG    AH,AL                     ; SWAP HIGH/LOW BYTES FOR WORD MOVE
061F F6 C2 80                      TEST    DL,80H                    ; AGAIN, IS THIS XOR FUNCTION
0622 74 05                         JZ      S11                       ; NO, JUST STORE THE VALUES
0624 26: 33 85 2000                XOR     AX,ES:[DI+2000H]          ; FUNCTION WITH FIRST HALF LOW
0629                       S11:
0629 26: 89 85 2000                MOV     ES:[DI+2000H],AX          ; STORE SECOND PORTION HIGH
062E 83 C7 50                      ADD     DI,80                     ; POINT TO NEXT LOCATION
0631 FE CE                         DEC     DH
0633 75 CF                         JNZ     S9                        ; KEEP GOING
```

SECTION 5

```
0635 5E                          POP     SI                     ; RECOVER CODE POINTER
0636 5F                          POP     DI                     ; RECOVER REGEN POINTER
0637 47                          INC     DI                     ; POINT TO NEXT CHAR POSITION
0638 47                          INC     DI
0639 E2 C5                       LOOP    S8                     ; MORE TO WRITE
063B E9 013D R                   JMP     VIDEO_RETURN
063E                     GRAPHICS_WRITE  ENDP
                         ;--------------------------------------
                         ; GRAPHICS READ
                         ;--------------------------------------
063E                     GRAPHICS_READ   PROC    NEAR
063E E8 06E9 R                   CALL    S26                    ; CONVERTED TO OFFSET IN REGEN
0641 8B F0                       MOV     SI,AX                  ; SAVE IN SI
0643 83 EC 08                    SUB     SP,8                   ; ALLOCATE SPACE FOR THE READ CODE POINT
0646 8B EC                       MOV     BP,SP                  ; POINTER TO SAVE AREA

                         ;-----  DETERMINE GRAPHICS MODES

0648 80 3E 0049 R 06             CMP     @CRT_MODE,6
064D 06                          PUSH    ES
064E 1F                          POP     DS                     ; POINT TO REGEN SEGMENT
064F 72 19                       JC      S13                    ; MEDIUM RESOLUTION

                         ;-----  HIGH RESOLUTION READ

                         ;-----  GET VALUES FROM REGEN BUFFER AND CONVERT TO CODE POINT
0651 B6 04                       MOV     DH,4                   ; NUMBER OF PASSES
0653                     S12:
0653 8A 04                       MOV     AL,[SI]                ; GET FIRST BYTE
0655 88 46 00                    MOV     [BP],AL                ; SAVE IN STORAGE AREA
0658 45                          INC     BP                     ; NEXT LOCATION
0659 8A 84 2000                  MOV     AL,[SI+2000H]          ; GET LOWER REGION BYTE
065D 88 46 00                    MOV     [BP],AL                ; ADJUST AND STORE
0660 45                          INC     BP
0661 83 C6 50                    ADD     SI,80                  ; POINTER INTO REGEN
0664 FE CE                       DEC     DH                     ; LOOP CONTROL
0666 75 EB                       JNZ     S12                    ; DO IT SOME MORE
0668 EB 16                       JMP     SHORT S15              ; GO MATCH THE SAVED CODE POINTS

                         ;-----  MEDIUM RESOLUTION READ
066A                     S13:                                   ; MED_RES_READ
066A D1 E6                       SAL     SI,1                   ; OFFSET*2 SINCE 2 BYTES/CHAR
066C B6 04                       MOV     DH,4                   ; NUMBER OF PASSES
066E                     S14:
066E E8 06CF R                   CALL    S23                    ; GET BYTES FROM REGEN INTO SINGLE SAVE
0671 81 C6 1FFE                  ADD     SI,2000H-2             ; GO TO LOWER REGION
0675 E8 06CF R                   CALL    S23                    ; GET THIS PAIR INTO SAVE
0678 81 EE 1FB2                  SUB     SI,2000H-80+2          ; ADJUST POINTER BACK INTO UPPER
067C FE CE                       DEC     DH
067E 75 EE                       JNZ     S14                    ; KEEP GOING UNTIL ALL 8 DONE

                         ;-----  SAVE AREA HAS CHARACTER IN IT, MATCH IT
0680                     S15:                                   ; FIND_CHAR
0680 BF 0000 E                   MOV     DI,OFFSET CRT_CHAR_GEN ; ESTABLISH ADDRESSING
0683 0E                          PUSH    CS
0684 07                          POP     ES                     ; CODE POINTS IN CS
0685 83 ED 08                    SUB     BP,8                   ; ADJUST POINTER TO START OF SAVE AREA
0688 8B F5                       MOV     SI,BP
068A B0 00                       MOV     AL,0                   ; CURRENT CODE POINT BEING MATCHED
068C                     S16:
068C 16                          PUSH    SS                     ; ESTABLISH ADDRESSING TO STACK
068D 1F                          POP     DS                     ; FOR THE STRING COMPARE
068E BA 0080                     MOV     DX,128                 ; NUMBER TO TEST AGAINST
0691                     S17:
0691 56                          PUSH    SI                     ; SAVE SAVE AREA POINTER
0692 57                          PUSH    DI                     ; SAVE CODE POINTER
0693 B9 0004                     MOV     CX,4                   ; NUMBER OF WORDS TO MATCH
0696 F3/ A7                      REPE    CMPSW                  ; COMPARE THE 8 BYTES AS WORDS
0698 5F                          POP     DI                     ; RECOVER THE POINTERS
0699 5E                          POP     SI
069A 74 1E                       JZ      S18                    ; IF ZERO FLAG SET, THEN MATCH OCCURRED
069C FE C0                       INC     AL                     ; NO MATCH, MOVE ON TO NEXT
069E 83 C7 08                    ADD     DI,8                   ; NEXT CODE POINT
06A1 4A                          DEC     DX                     ; LOOP CONTROL
06A2 75 ED                       JNZ     S17                    ; DO ALL OF THEM

                         ;-----  CHAR NOT MATCHED, MIGHT BE IN USER SUPPLIED SECOND HALF

06A4 3C 00                       CMP     AL,0                   ;  AL<> 0 IF ONLY 1ST HALF SCANNED
06A6 74 12                       JE      S18                    ;  IF = 0, THEN ALL HAS BEEN SCANNED
06A8 2B C0                       SUB     AX,AX
06AA 8E D8                       MOV     DS,AX                  ;  ESTABLISH ADDRESSING TO VECTOR
                                 ASSUME  DS:ABS0
06AC C4 3E 007C R                LES     DI,@EXT_PTR            ;  GET POINTER
06B0 8C C0                       MOV     AX,ES                  ;  SEE IF THE POINTER REALLY EXISTS
06B2 0B C7                       OR      AX,DI                  ; IF ALL 0, THEN DOESN'T EXIST
06B4 74 04                       JZ      S18                    ; NO SENSE LOOKING
06B6 B0 80                       MOV     AL,128                 ; ORIGIN FOR SECOND HALF
06B8 EB D2                       JMP     S16                    ; GO BACK AND TRY FOR IT
                                 ASSUME  DS:DATA

                         ;-----  CHARACTER IS FOUND ( AL=0 IF NOT FOUND )
06BA                     S18:
06BA 83 C4 08                    ADD     SP,8                   ; READJUST THE STACK, THROW AWAY SAVE
06BD E9 013D R                   JMP     VIDEO_RETURN           ; ALL DONE
06C0                     GRAPHICS_READ   ENDP
                         ;------------------------------------------
                         ; EXPAND_BYTE
                         ;  THIS ROUTINE TAKES THE BYTE IN AL AND DOUBLES ALL
                         ;  OF THE BITS, TURNING THE 8 BITS INTO 16 BITS.
                         ;  THE RESULT IS LEFT IN AX
                         ;------------------------------------------
06C0                     S21     PROC    NEAR
06C0 51                          PUSH    CX                     ; SAVE REGISTER
06C1 B9 0008                     MOV     CX,8                   ; SHIFT COUNT REGISTER FOR ONE BYTE
06C4                     S22:
06C4 D0 C8                       ROR     AL,1                   ; SHIFT BITS, LOW BIT INTO CARRY FLAG
06C6 D1 DD                       RCR     BP,1                   ; MOVE CARRY FLAG (LOW BIT) INTO RESULTS
06C8 D1 FD                       SAR     BP,1                   ; SIGN EXTEND HIGH BIT (DOUBLE IT)
06CA E2 F8                       LOOP    S22                    ; REPEAT FOR ALL 8 BITS

06CC 95                          XCHG    AX,BP                  ; MOVE RESULTS TO PARAMETER REGISTER
06CD 59                          POP     CX                     ; RECOVER REGISTER
06CE C3                          RET                            ; ALL DONE
```

```
06CF                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 --
                    ;  SI AND BP ARE INCREMENTED
                    ;--------------------------------------------------
06CF                S23     PROC    NEAR
06CF AD                     LODSW                   ; GET FIRST BYTE AND SECOND BYTES
06D0 86 C4                  XCHG    AL,AH           ; SWAP FOR COMPARE
06D2 B9 C000                MOV     CX,0C000H       ; 2 BIT MASK TO TEST THE ENTRIES
06D5 B2 00                  MOV     DL,0            ; RESULT REGISTER
06D7                S24:
06D7 85 C1                  TEST    AX,CX           ; IS THIS SECTION BACKGROUND?
06D9 74 01                  JZ      S25             ; IF ZERO, IT IS BACKGROUND (CARRY=0)
06DB F9                     STC                     ; WASN'T, SO SET CARRY
06DC                S25:
06DC D0 D2                  RCL     DL,1            ; MOVE THAT BIT INTO THE RESULT
06DE D1 E9                  SHR     CX,1
06E0 D1 E9                  SHR     CX,1            ; MOVE THE MASK TO THE RIGHT BY 2 BITS
06E2 73 F3                  JNC     S24             ; DO IT AGAIN IF MASK DIDN'T FALL OUT
06E4 88 56 00               MOV     [BP],DL         ; STORE RESULT IN SAVE AREA
06E7 45                     INC     BP              ; ADJUST POINTER
06E8 C3                     RET                     ; ALL DONE
06E9                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
                    ;----------------------------------------
06E9                S26     PROC    NEAR
06E9 A1 0050 R              MOV     AX,@CURSOR_POSN         ; GET CURRENT CURSOR
06EC                GRAPH_POSN      LABEL   NEAR
06EC 53                     PUSH    BX                      ; SAVE REGISTER
06ED 8B D8                  MOV     BX,AX                   ; SAVE A COPY OF CURRENT CURSOR
06EF A0 004A R              MOV     AL,BYTE PTR @CRT_COLS   ; GET BYTES PER COLUMN
06F2 F6 E4                  MUL     AH                      ; MULTIPLY BY ROWS
06F4 D1 E0                  SHL     AX,1
06F6 D1 E0                  SHL     AX,1                    ; MULTIPLY * 4 SINCE 4 ROWS/BYTE
06F8 2A FF                  SUB     BH,BH                   ; ISOLATE COLUMN VALUE
06FA 03 C3                  ADD     AX,BX                   ; DETERMINE OFFSET
06FC 5B                     POP     BX                      ; RECOVER POINTER
06FD C3                     RET                             ; ALL DONE
06FE                S26     ENDP
                    ;--- WRITE_TTY ------------------------------------------------------------------------
                    ;                                                                                    :
                    ;   THIS INTERFACE PROVIDES A TELETYPE LIKE INTERFACE TO THE                         :
                    ;     VIDEO CARDS.  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, CARRIAGE RETURN, BELL AND LINE FEED ARE           :
                    ;            HANDLED AS COMMANDS RATHER THAN AS DISPLAY GRAPHICS CHARACTERS          :
                    ;     (BL) = FOREGROUND COLOR FOR CHAR WRITE IF CURRENTLY IN A GRAPHICS MODE         :
                    ;   EXIT --                                                                          :
                    ;     ALL REGISTERS SAVED THROUGH VIDEO_EXIT (INCLUDING (AX))                        :
                    ;------------------------------------------------------------------------------------
                            ASSUME  DS:DATA
06FE                WRITE_TTY       PROC    NEAR
06FE 97                     XCHG    DI,AX                   ; SAVE (AX) REGISTER IN (DI) FOR EXIT
06FF B4 03                  MOV     AH,03H                  ; READ CURSOR POSITION
0701 8A 3E 0062 R           MOV     BH,@ACTIVE_PAGE         ; GET CURRENT PAGE SETTING
0705 CD 10                  INT     10H                     ; READ THE CURRENT CURSOR POSITION
0707 8B C7                  MOV     AX,DI                   ; RECOVER CHARACTER FROM (DI) REGISTER

                    ;-----  DX NOW HAS THE CURRENT CURSOR POSITION

0709 3C 0D                  CMP     AL,CR                   ; IS IT CARRIAGE RETURN OR CONTROL
070B 76 46                  JBE     U8                      ; GO TO CONTROL CHECKS IF IT IS

                    ;-----  WRITE THE CHAR TO THE SCREEN
070D                U0:
070D B4 0A                  MOV     AH,0AH                  ; WRITE CHARACTER ONLY COMMAND
070F B9 0001                MOV     CX,1                    ; ONLY ONE CHARACTER
0712 CD 10                  INT     10H                     ; WRITE THE CHARACTER

                    ;-----  POSITION THE CURSOR FOR NEXT CHAR

0714 FE C2                  INC     DL
0716 3A 16 004A R           CMP     DL,BYTE PTR @CRT_COLS   ; TEST FOR COLUMN OVERFLOW
071A 75 33                  JNZ     U7                      ;  SET_CURSOR
071C B2 00                  MOV     DL,0                    ; COLUMN FOR CURSOR
071E 80 FE 18               CMP     DH,25-1                 ; CHECK FOR LAST ROW
0721 75 2A                  JNZ     U6                      ;  SET_CURSOR_INC

                    ;-----  SCROLL REQUIRED
0723                U1:
0723 B4 02                  MOV     AH,02H
0725 CD 10                  INT     10H                     ; SET THE CURSOR

                    ;-----  DETERMINE VALUE TO FILL WITH DURING SCROLL

```

SECTION 5

```
0727 A0 0049 R                      MOV     AL,@CRT_MODE            ; GET THE CURRENT MODE
072A 3C 04                          CMP     AL,4
072C 72 06                          JC      U2                      ; READ-CURSOR
072E 3C 07                          CMP     AL,7
0730 B7 00                          MOV     BH,0                    ; FILL WITH BACKGROUND
0732 75 06                          JNE     U3                      ; SCROLL-UP
0734                        U2:                                     ; READ-CURSOR
0734 B4 08                          MOV     AH,08H                  ; GET READ CURSOR COMMAND
0736 CD 10                          INT     10H                     ; READ CHAR/ATTR AT CURRENT CURSOR
0738 8A FC                          MOV     BH,AH                   ; STORE IN BH
073A                        U3:                                     ; SCROLL-UP
073A B8 0601                        MOV     AX,0601H                ; SCROLL ONE LINE
073D 2B C9                          SUB     CX,CX                   ; UPPER LEFT CORNER
073F B6 18                          MOV     DH,25-1                 ; LOWER RIGHT ROW
0741 8A 16 004A R                   MOV     DL,BYTE PTR @CRT_COLS   ; LOWER RIGHT COLUMN
0745 FE CA                          DEC     DL
0747                        U4:                                     ; VIDEO-CALL-RETURN
0747 CD 10                          INT     10H                     ; SCROLL UP THE SCREEN
0749                        U5:                                     ; TTY-RETURN
0749 97                             XCHG    AX,DI                   ; RESTORE THE ENTRY CHARACTER FROM (DI)
074A E9 013D R                      JMP     VIDEO_RETURN            ; RETURN TO CALLER

074D                        U6:                                     ; SET-CURSOR-INC
074D FE C6                          INC     DH                      ; NEXT ROW
074F                        U7:                                     ; SET-CURSOR
074F B4 02                          MOV     AH,02H
0751 EB F4                          JMP     U4                      ; ESTABLISH THE NEW CURSOR

                            ;-----  CHECK FOR CONTROL CHARACTERS
0753                        U8:
0753 74 13                          JE      U9                      ; WAS IT A CARRIAGE RETURN
0755 3C 0A                          CMP     AL,LF                   ; IS IT A LINE FEED
0757 74 13                          JE      U10                     ; GO TO LINE FEED
0759 3C 07                          CMP     AL,07H                  ; IS IT A BELL
075B 74 16                          JE      U11                     ; GO TO BELL
075D 3C 08                          CMP     AL,08H                  ; IS IT A BACKSPACE
075F 75 AC                          JNE     U0                      ; IF NOT A CONTROL, DISPLAY IT

                            ;-----  BACK SPACE FOUND

0761 0A D2                          OR      DL,DL                   ; IS IT ALREADY AT START OF LINE
0763 74 EA                          JE      U7                      ; SET_CURSOR
0765 4A                             DEC     DX                      ; NO -- JUST MOVE IT BACK
0766 EB E7                          JMP     U7                      ; SET_CURSOR

                            ;-----  CARRIAGE RETURN FOUND

0768                        U9:
0768 B2 00                          MOV     DL,0                    ; MOVE TO FIRST COLUMN
076A EB E3                          JMP     U7                      ; SET_CURSOR

                            ;-----  LINE FEED FOUND

076C                        U10:
076C 80 FE 18                       CMP     DH,25-1                 ; BOTTOM OF SCREEN
076F 75 DC                          JNE     U6                      ; YES, SCROLL THE SCREEN
0771 EB B0                          JMP     U1                      ; NO, JUST SET THE CURSOR

                            ;-----  BELL FOUND

0773                        U11:
0773 B9 0533                        MOV     CX,1331                 ; DIVISOR FOR 896 HZ TONE
0776 B3 1F                          MOV     BL,31                   ; SET COUNT FOR 31/64 SECOND FOR BEEP
0778 E8 0000 E                      CALL    BEEP                    ; SOUND THE POD BELL
077B EB CC                          JMP     U5                      ; TTY_RETURN
077D                        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  DS:DATA
077D 03 03 05 05 03 03      V1      DB      3,3,5,5,3,3,3,4         ; SUBTRACT_TABLE
     03 04
                            ;-----  WAIT FOR LIGHT PEN TO BE DEPRESSED

0785                        READ_LPEN       PROC    NEAR
0785 B4 00                          MOV     AH,0                    ; SET NO LIGHT PEN RETURN CODE
0787 8B 16 0063 R                   MOV     DX,@ADDR_6845           ; GET BASE ADDRESS OF 6845
078B 83 C2 06                       ADD     DX,6                    ; POINT TO STATUS REGISTER
078E EC                             IN      AL,DX                   ; GET STATUS REGISTER
078F A8 04                          TEST    AL,004H                 ; TEST LIGHT PEN SWITCH
0791 74 03                          JZ      V6_A                    ; GO IF YES
0793 E9 0816 R                      JMP     V6                      ; NOT SET, RETURN

                            ;-----  NOW TEST FOR LIGHT PEN TRIGGER

0796 A8 02                  V6_A:   TEST    AL,2                    ; TEST LIGHT PEN TRIGGER
0798 75 03                          JNZ     V7A                     ; RETURN WITHOUT RESETTING TRIGGER
079A E9 0820 R                      JMP     V7

                            ;-----  TRIGGER HAS BEEN SET, READ THE VALUE IN

079D                        V7A:
079D B4 10                          MOV     AH,16                   ; LIGHT PEN REGISTERS ON 6845

                            ;-----  INPUT REGISTERS POINTED TO BY AH, AND CONVERT TO ROW COLUMN IN (DX)

079F 8B 16 0063 R                   MOV     DX,@ADDR_6845           ; ADDRESS REGISTER FOR 6845
07A3 8A C4                          MOV     AL,AH                   ; REGISTER TO READ
07A5 EE                             OUT     DX,AL                   ; SET IT UP
07A6 90                             NOP                             ; I/O DELAY
07A7 42                             INC     DX                      ; DATA REGISTER
07A8 EC                             IN      AL,DX                   ; GET THE VALUE
07A9 8A E8                          MOV     CH,AL                   ; SAVE IN CX
```

```
07AB 4A                     DEC     DX                      ; ADDRESS REGISTER
07AC FE C4                  INC     AH
07AE 8A C4                  MOV     AL,AH                   ; SECOND DATA REGISTER
07B0 EE                     OUT     DX,AL
07B1 42                     INC     DX                      ; POINT TO DATA REGISTER
07B2 90                     NOP                             ; I/O DELAY
07B3 EC                     IN      AL,DX                   ; GET SECOND DATA VALUE
07B4 8A E5                  MOV     AH,CH                   ; AX HAS INPUT VALUE

                    ;-----  AX HAS THE VALUE READ IN FROM THE 6845

07B6 8A 1E 0049 R           MOV     BL,@CRT_MODE
07BA 2A FF                  SUB     BH,BH                   ; MODE VALUE TO BX
07BC 2E: 8A 9F 077D R       MOV     BL,CS:V1[BX]            ; DETERMINE AMOUNT TO SUBTRACT
07C1 2B C3                  SUB     AX,BX                   ; TAKE IT AWAY
07C3 8B 1E 004E R           MOV     BX,@CRT_START
07C7 D1 EB                  SHR     BX,1
07C9 2B C3                  SUB     AX,BX                   ; CONVERT TO CORRECT PAGE ORIGIN
07CB 79 02                  JNS     V2                      ; IF POSITIVE, DETERMINE MODE
07CD 2B C0                  SUB     AX,AX                   ; <0 PLAYS AS 0

                    ;-----  DETERMINE MODE OF OPERATION

07CF                V2:                                     ; DETERMINE_MODE
07CF B1 03                  MOV     CL,3                    ; SET *8 SHIFT COUNT
07D1 80 3E 0049 R 04        CMP     @CRT_MODE,4             ; DETERMINE IF GRAPHICS OR ALPHA
07D6 72 2A                  JB      V4                      ; ALPHA_PEN
07D8 80 3E 0049 R 07        CMP     @CRT_MODE,7
07DD 74 23                  JE      V4                      ; ALPHA_PEN

                    ;-----  GRAPHICS MODE

07DF B2 28                  MOV     DL,40                   ; DIVISOR FOR GRAPHICS
07E1 F6 F2                  DIV     DL                      ; DETERMINE ROW(AL) AND COLUMN(AH)
                                                            ;  AL RANGE 0-99, AH RANGE 0-39
                    ;-----  DETERMINE GRAPHIC ROW POSITION

07E3 8A E8                  MOV     CH,AL                   ; SAVE ROW VALUE IN CH
07E5 02 ED                  ADD     CH,CH                   ; *2 FOR EVEN/ODD FIELD
07E7 8A DC                  MOV     BL,AH                   ; COLUMN VALUE TO BX
07E9 2A FF                  SUB     BH,BH                   ; MULTIPLY BY 8 FOR MEDIUM RES
07EB 80 3E 0049 R 06        CMP     @CRT_MODE,6             ; DETERMINE MEDIUM OR HIGH RES
07F0 75 04                  JNE     V3                      ; NOT_HIGH_RES
07F2 B1 04                  MOV     CL,4                    ; SHIFT VALUE FOR HIGH RES
07F4 D0 E4                  SAL     AH,1                    ; COLUMN VALUE TIMES 2 FOR HIGH RES
07F6                V3:                                     ; NOT_HIGH_RES
07F6 D3 E3                  SHL     BX,CL                   ; MULTIPLY *16 FOR HIGH RES

                    ;-----  DETERMINE ALPHA CHAR POSITION

07F8 8A D4                  MOV     DL,AH                   ; COLUMN VALUE FOR RETURN
07FA 8A F0                  MOV     DH,AL                   ; ROW VALUE
07FC D0 EE                  SHR     DH,1                    ; DIVIDE BY 4
07FE D0 EE                  SHR     DH,1                    ;    FOR VALUE IN 0-24 RANGE
0800 EB 12                  JMP     SHORT V5                ; LIGHT_PEN_RETURN_SET

                    ;-----  ALPHA MODE ON LIGHT PEN

0802                V4:                                     ; ALPHA_PEN
0802 F6 36 004A R           DIV     BYTE PTR @CRT_COLS      ; DETERMINE ROW,COLUMN VALUE
0806 8A F0                  MOV     DH,AL                   ; ROWS TO DH
0808 8A D4                  MOV     DL,AH                   ; COLS TO DL
080A D2 E0                  SAL     AL,CL                   ; MULTIPLY ROWS * 8
080C 8A E8                  MOV     CH,AL                   ; GET RASTER VALUE TO RETURN REGISTER
080E 8A DC                  MOV     BL,AH                   ; COLUMN VALUE
0810 32 FF                  XOR     BH,BH                   ;  TO BX
0812 D3 E3                  SAL     BX,CL
0814                V5:                                     ; LIGHT_PEN_RETURN_SET
0814 B4 01                  MOV     AH,1                    ; INDICATE EVERY THING SET
0816                V6:                                     ; LIGHT_PEN_RETURN
0816 52                     PUSH    DX                      ; SAVE RETURN VALUE (IN CASE)
0817 8B 16 0063 R           MOV     DX,@ADDR_6845           ; GET BASE ADDRESS
081B 83 C2 07               ADD     DX,7                    ; POINT TO RESET PARM
081E EE                     OUT     DX,AL                   ; ADDRESS, NOT DATA, IS IMPORTANT
081F 5A                     POP     DX                      ; RECOVER VALUE
0820                V7:                                     ; RETURN_NO_RESET
0820 5D                     POP     BP
0821 5F                     POP     DI
0822 5E                     POP     SI
0823 1F                     POP     DS                      ; DISCARD SAVED BX,CX,DX
0824 1F                     POP     DS
0825 1F                     POP     DS
0826 1F                     POP     DS
0827 07                     POP     ES
0828 CF                     IRET
0829                READ_LPEN       ENDP
0829                CODE    ENDS
                            END
```

SECTION 5

```
                        PAGE 118,121
                        TITLE BIOS1 ---- 01/10/86   INTERRUPT 15H BIOS ROUTINES
                        .LIST
 0000                   CODE    SEGMENT BYTE PUBLIC

                                PUBLIC  CASSETTE_IO_1

                                EXTRN   CONF_TBL:NEAR           ; SYSTEM/BIOS CONFIGURATION TABLE
                                EXTRN   DDS:NEAR                ; LOAD (DS) WITH DATA SEGMENT SELECTOR

                        ;--- INT 15 H -----------------------------------------------------------
                        ;     INPUT - CASSETTE I/O FUNCTIONS                                   :
                        ;                                                                      :
                        ;          (AH) = 00H                                                  :
                        ;          (AH) = 01H                                                  :
                        ;          (AH) = 02H                                                  :
                        ;          (AH) = 03H                                                  :
                        ;       RETURNS FOR THESE FUNCTIONS ALWAYS (AH) = 86H, CY = 1)         :
                        ;       IF CASSETTE PORT NOT PRESENT                                   :
                        ;----------------------------------------------------------------------:
                        ;     INPUT - UNUSED FUNCTIONS                                         :
                        ;          (AH) = 04H THROUGH 7FH                                      :
                        ;       RETURNS FOR THESE FUNCTIONS ALWAYS (AH) = 86H, CY = 1)         :
                        ;               (UNLESS INTERCEPTED BY SYSTEM HANDLERS)                :
                        ;               NOTE: THE KEYBOARD INTERRUPT HANDLER INTERRUPTS WITH AH=4FH :
                        ;----------------------------------------------------------------------:
                        ; EXTENSIONS                                                           :
                        ;          (AH) = 80H   DEVICE OPEN (NULL)                             :
                        ;                          (BX) = DEVICE ID                            :
                        ;                          (CX) = PROCESS ID                           :
                        ;                                                                      :
                        ;          (AH) = 81H   DEVICE CLOSE (NULL)                            :
                        ;                          (BX) = DEVICE ID                            :
                        ;                          (CX) = PROCESS ID                           :
                        ;                                                                      :
                        ;          (AH) = 82H   PROGRAM TERMINATION (NULL)                     :
                        ;                          (BX) = DEVICE ID                            :
                        ;                                                                      :
                        ;          (AH) = 83H   EVENT WAIT (NULL)                              :
                        ;                                                                      :
                        ;          (AH) = 84H   JOYSTICK SUPPORT                               :
                        ;                          (DX) = 00H - READ THE CURRENT SWITCH SETTINGS :
                        ;                                    RETURNS AL = SWITCH SETTINGS (BITS 7-4) :
                        ;                          (DX) = 01H - READ THE RESISTIVE INPUTS      :
                        ;                                    RETURNS AX = A(x) VALUE           :
                        ;                                            BX = A(y) VALUE           :
                        ;                                            CX = B(x) VALUE           :
                        ;                                            DX = B(y) VALUE           :
                        ;                                                                      :
                        ;          (AH) = 88H   EXTENDED MEMORY SIZE DETERMINE                 :
                        ;                                                                      :
                        ;          (AH) = 91H   INTERRUPT COMPLETE FLAG SET                    :
                        ;                          (AL)    TYPE CODE                           :
                        ;                          00H -> 7FH                                  :
                        ;                                  SERIALLY REUSABLE DEVICES           :
                        ;                                  OPERATING SYSTEM MUST SERIALIZE ACCESS :
                        ;                          80H -> BFH                                  :
                        ;                                  REENTRANT DEVICES; ES:BX IS USED TO :
                        ;                                  DISTINGUISH DIFFERENT CALLS (MULTIPLE I/O :
                        ;                                  CALLS ARE ALLOWED SIMULTANEOUSLY)   :
                        ;                          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             :
                        ;                                                                      :
                        ;          (AH) = C0H   RETURN CONFIGURATION PARAMETERS POINTER        :
                        ;                          RETURNS                                     :
                        ;                           (AH) = 00H AND CY= 0 (IF PRESENT ELSE 86 AND CY= 1) :
                        ;                           (ES:BX) = PARAMETER TABLE ADDRESS POINTER  :
                        ;                                       WHERE:                         :
                        ;                                                                      :
                        ;                     DW   8            LENGTH OF FOLLOWING TABLE      :
                        ;                     DB   MODEL_BYTE   SYSTEM MODEL BYTE              :
                        ;                     DB   TYPE_BYTE    SYSTEM MODEL TYPE BYTE         :
                        ;                     DB   BIOS_LEVEL   BIOS REVISION LEVEL            :
                        ;                     DB   ?            10000000 = DMA CHANNEL 3 USE BY BIOS :
                        ;                                       01000000 = CASCADED INTERRUPT LEVEL 2 :
                        ;                                       00100000 = REAL TIME CLOCK AVAILABLE :
                        ;                                       00010000 = KEYBOARD SCAN CODE HOOK 1AH :
                        ;                     DB   0            RESERVED                       :
                        ;                     DB   0            RESERVED                       :
                        ;                     DB   0            RESERVED                       :
                        ;                     DB   0            RESERVED                       :
                        ;                                                                      :
                        ;-----------------------------------------------------------------------

                                ASSUME  CS:CODE

 0000                   CASSETTE_IO_1   PROC    FAR
 0000  FB                       STI                     ; ENABLE INTERRUPTS
 0001  80 FC 80                 CMP     AH,080H         ; CHECK FOR RANGE OF  00-7FH
 0004  73 06                    JAE     C1_G            ; SKIP AND HANDLE, ELSE RETURN ERROR

 0006                   C1:                             ;       ERROR
 0006  B4 86                    MOV     AH,86H          ; SET BAD COMMAND
 0008  F9                       STC                     ; SET CARRY FLAG ON (CY=1)

 0009                   C1_F:                           ;       COMMON EXIT
 0009  CA 0002                  RET     2               ; FAR RETURN EXIT FROM ROUTINES

 000C                   C1_G:                           ;       CONTINUE CHECKING FOR FUNCTION
 000C  80 FC C0                 CMP     AH,0C0H         ; CHECK FOR CONFIGURATION PARAMETERS
 000F  74 2E                    JE      CONF_PARMS
```

```
0011 80 EC 80                   SUB     AH,080H                 ; BASE ON 0
0014 74 25                      JZ      DEV_OPEN                ; DEVICE OPEN           (80H)
0016 FE CC                      DEC     AH
0018 74 21                      JZ      DEV_CLOSE               ; DEVICE CLOSE          (81H)
001A FE CC                      DEC     AH
001C 74 1D                      JZ      PROG_TERM               ; PROGRAM TERMINATION   (82H)
001E FE CC                      DEC     AH                      ;  IGNORE EVENT WAIT    (83H)
0020 FE CC                      DEC     AH
0022 74 27                      JZ      JOY_STICK               ; JOYSTICK BIOS         (84H)
0024 FE CC                      DEC     AH
0026 74 13                      JZ      SYS_REQ                 ; SYSTEM REQUEST KEY    (85H)
0028 FE CC                      DEC     AH                      ;  IGNORE WAIT          (86H)
002A FE CC                      DEC     AH                      ;  IGNORE BLOCK MOVE    (87H)
002C FE CC                      DEC     AH
002E 74 18                      JZ      EXT_MEMORY              ; EXTENDED MEMORY SIZE (88H)

0030 80 EC 08                   SUB     AH,8                    ; CHECK FOR FUNCTION    (90H)
0033 74 06                      JZ      DEVICE_BUSY
0035 FE CC                      DEC     AH                      ; CHECK FOR FUNCTION    (91H)
0037 74 05                      JZ      INT_COMPLETE            ; GO TO INTERRUPT COMPLETE RETURN
0039 EB CB                      JMP     C1                      ; EXIT IF NOT A VALID FUNCTION

003B                    DEV_OPEN:                               ; NULL HANDLERS
003B                    DEV_CLOSE:
003B                    PROG_TERM:
003B                    SYS_REQ:
003B                    DEVICE_BUSY:
003B F8                         CLC                             ; TURN CARRY OFF
003C EB CB                      JMP     C1_F                    ; RETURN WITH (AH= 00) AND CY=0

003E                    CASSETTE_IO_1   ENDP

                        ;--- INTERRUPT COMPLETE ------------------------
                        ;                                               :
                        ;       THIS ROUTINE IS A TEMPORARY HANDLER     :
                        ;       FOR INTERRUPT COMPLETE                  :
                        ;                                               :
                        ;       INPUT   - SEE PROLOGUE                  :
                        ;-----------------------------------------------

003E                    INT_COMPLETE    PROC    NEAR
003E CF                         IRET                            ; RETURN
003F                    INT_COMPLETE    ENDP

003F                    CONF_PARMS      PROC    NEAR            ;       FUNCTION (C0H)
003F 0E                         PUSH    CS                      ; GET CODE SEGMENT
0040 07                         POP     ES                      ; PLACE IN SELECTOR POINTER
0041 BB 0000 E                  MOV     BX,OFFSET CONF_TBL      ; GET OFFSET OF PARAMETER TABLE
0044 32 E4                      XOR     AH,AH                   ; CLEAR AH AND SET CARRY OFF
0046 EB C1                      JMP     C1_F                    ; EXIT THROUGH COMMON RETURN
0048                    CONF_PARMS      ENDP

                        ;--- INT 15 H -- ( FUNCTION 88 H - I/O 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.                                                    :
                        ; INPUT                                                                       :
                        ;       AH = 88H                                                              :
                        ;                                                                             :
                        ; OUTPUT                                                                      :
                        ;       (AX) = 0                                                              :
                        ;                                                                             :
                        ;------------------------------------------------------------------------------

0048                    EXT_MEMORY      PROC

0048 33 C0                      XOR     AX,AX                   ; SET EXTENDED MEMORY SIZE TO ZERO

004A CF                         IRET                            ; RETURN TO USER

004B                    EXT_MEMORY      ENDP
```

SECTION 5

```
                              PAGE
                              ;--- 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           :
                              ;---------------------------------------------------------------

 004B                         JOY_STICK       PROC    NEAR
 004B FB                              STI                     ; INTERRUPTS BACK ON
 004C 8B C2                           MOV     AX,DX           ; GET SUB FUNCTION CODE
 004E BA 0201                         MOV     DX,201H         ; ADDRESS OF PORT
 0051 0A C0                           OR      AL,AL
 0053 74 09                           JZ      JOY_2           ; READ SWITCHES
 0055 FE C8                           DEC     AL
 0057 74 0A                           JZ      JOY_3           ; READ RESISTIVE INPUTS
 0059 EB AB                           JMP     C1              ; GO TO ERROR RETURN
 005B                         JOY_1:
 005B FB                              STI
 005C EB AB                           JMP     C1_F            ; GO TO COMMON RETURN

 005E                         JOY_2:
 005E EC                              IN      AL,DX
 005F 24 F0                           AND     AL,0F0H         ; STRIP UNWANTED BITS OFF
 0061 EB F8                           JMP     JOY_1           ; FINISHED

 0063                         JOY_3:
 0063 B3 01                           MOV     BL,1
 0065 E8 0081 R                       CALL    TEST_CORD
 0068 51                              PUSH    CX              ; SAVE A(X) VALUE
 0069 B3 02                           MOV     BL,2
 006B E8 0081 R                       CALL    TEST_CORD
 006E 51                              PUSH    CX              ; SAVE A(Y) VALUE
 006F B3 04                           MOV     BL,4
 0071 E8 0081 R                       CALL    TEST_CORD
 0074 51                              PUSH    CX              ; SAVE B(X) VALUE
 0075 B3 08                           MOV     BL,8
 0077 E8 0081 R                       CALL    TEST_CORD
 007A 8B D1                           MOV     DX,CX           ; SAVE B(Y) VALUE
 007C 59                              POP     CX              ; GET B(X) VALUE
 007D 5B                              POP     BX              ; GET A(Y) VALUE
 007E 58                              POP     AX              ; GET A(X) VALUE
 007F EB DA                           JMP     JOY_1           ; FINISHED - RETURN

 0081                         TEST_CORD       PROC    NEAR
 0081 52                              PUSH    DX              ; SAVE
 0082 FA                              CLI                     ; BLOCK INTERRUPTS WHILE READING
 0083 B0 00                           MOV     AL,0            ; SET UP TO LATCH TIMER 0
 0085 E6 43                           OUT     TIMER+3,AL
 0087 EB 00                           JMP     $+2
 0089 E4 40                           IN      AL,TIMER        ; READ LOW BYTE OF TIMER 0
 008B EB 00                           JMP     $+2
 008D 8A E0                           MOV     AH,AL
 008F E4 40                           IN      AL,TIMER        ; READ HIGH BYTE OF TIMER 0
 0091 86 E0                           XCHG    AH,AL           ; REARRANGE TO HIGH,LOW
 0093 50                              PUSH    AX              ; SAVE
 0094 B9 04FF                         MOV     CX,4FFH         ; SET COUNT
 0097 EE                              OUT     DX,AL           ; FIRE TIMER
 0098 EB 00                           JMP     $+2
 009A                         TEST_CORD_1:
 009A EC                              IN      AL,DX           ; READ VALUES
 009B 84 C3                           TEST    AL,BL           ; HAS PULSE ENDED?
 009D E0 FB                           LOOPNZ  TEST_CORD_1
 009F 83 F9 00                        CMP     CX,0
 00A2 59                              POP     CX              ; ORIGINAL COUNT
 00A3 75 04                           JNZ     SHORT TEST_CORD_2
 00A5 2B C9                           SUB     CX,CX           ; SET 0 COUNT FOR RETURN
 00A7 EB 2D                           JMP     SHORT TEST_CORD_3 ; EXIT WITH COUNT = 0
 00A9                         TEST_CORD_2:
 00A9 B0 00                           MOV     AL,0            ; SET UP TO LATCH TIMER 0
 00AB E6 43                           OUT     TIMER+3,AL
 00AD EB 00                           JMP     $+2
 00AF E4 40                           IN      AL,TIMER        ; READ LOW BYTE OF TIMER 0
 00B1 8A E0                           MOV     AH,AL
 00B3 EB 00                           JMP     $+2
 00B5 E4 40                           IN      AL,TIMER        ; READ HIGH BYTE OF TIMER 0
 00B7 86 E0                           XCHG    AH,AL           ; REARRANGE TO HIGH,LOW

 00B9 3B C8                           CMP     CX,AX           ; CHECK FOR COUNTER WRAP
 00BB 73 0B                           JAE     TEST_CORD_4     ; GO IF NO
 00BD 52                              PUSH    DX
 00BE BA FFFF                         MOV     DX,-1

 00C1 2B D0                           SUB     DX,AX           ; ADJUST FOR WRAP
 00C3 03 CA                           ADD     CX,DX
 00C5 5A                              POP     DX
 00C6 EB 02                           JMP     SHORT TEST_CORD_5

 00C8                         TEST_CORD_4:
 00C8 2B C8                           SUB     CX,AX
 00CA                         TEST_CORD_5:
 00CA 81 E1 1FF0                      AND     CX,1FF0H        ; ADJUST
 00CE D1 E9                           SHR     CX,1
 00D0 D1 E9                           SHR     CX,1
 00D2 D1 E9                           SHR     CX,1
 00D4 D1 E9                           SHR     CX,1

 00D6                         TEST_CORD_3:
 00D6 FB                              STI                     ; INTERRUPTS BACK ON
 00D7 BA 0201                         MOV     DX,201H         ; FLUSH OTHER INPUTS
 00DA 51                              PUSH    CX
 00DB 50                              PUSH    AX
 00DC B9 04FF                         MOV     CX,4FFH         ; COUNT
 00DF                         TEST_CORD_6:
 00DF EC                              IN      AL,DX
```

```
00E0 A8 0F                          TEST    AL,0FH
00E2 E0 FB                          LOOPNZ  TEST_CORD_6

00E4 58                             POP     AX
00E5 59                             POP     CX
00E6 5A                             POP     DX              ; SET COUNT

00E7 C3                             RET                     ; RETURN

00E8                     TEST_CORD          ENDP
00E8                     JOY_STICK          ENDP

00E8                     CODE       ENDS
                                    END
```

```
                                PAGE 118,121
                                TITLE POST ----- 01/10/86  SYSTEM POST AND BIOS PROCEDURES

                                PUBLIC  A1
                                PUBLIC  BEEP
                                PUBLIC  CONF_TBL
                                PUBLIC  CRT_CHAR_GEN
                                PUBLIC  DDS
                                PUBLIC  DISK_BASE
                                PUBLIC  M5
                                PUBLIC  M6
                                PUBLIC  M7
                                PUBLIC  MD_TBL1
                                PUBLIC  MD_TBL2
                                PUBLIC  MD_TBL3
                                PUBLIC  MD_TBL4
                                PUBLIC  MD_TBL5
                                PUBLIC  MD_TBL6
                                PUBLIC  P_O_R
                                PUBLIC  RESET
                                PUBLIC  VIDEO_PARMS
                                PUBLIC  WAITF

                                EXTRN   CASSETTE_IO_1:NEAR
                                EXTRN   DISKETTE_IO_1:NEAR
                                EXTRN   DISK_INT_1:NEAR
                                EXTRN   DSKETTE_SETUP:NEAR
                                EXTRN   KB_INT_1:NEAR
                                EXTRN   KEYBOARD_IO_1:NEAR
                                EXTRN   NEC_OUTPUT:NEAR
                                EXTRN   PRINTER_IO_1:NEAR
                                EXTRN   RESULTS:NEAR
                                EXTRN   RS232_IO_1:NEAR
                                EXTRN   SEEK:NEAR
                                EXTRN   VIDEO_IO_1:NEAR

                                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
                                .LIST
                                ;----------------------------------------------------------------
                                ;       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.              :
                                ;----------------------------------------------------------------

                                ;-------------------------------------
                                ;       ROM RESIDENT CODE           :
                                ;-------------------------------------
0000                            CODE    SEGMENT BYTE    PUBLIC

0000  1FFF [                            DB      01FFFH  DUP     (0CCH)  ; FILL UNUSED LOCATIONS WITH INTERRUPT 3
            CC
                ]

                                ;       ORG     0E000H
0000                                    ORG     0
0000  36 32 58 30 38 35                 DB      '62X0851 COPR. IBM 1986'        ; COPYRIGHT NOTICE
      31 20 43 4F 50 52
      2E 20 49 42 4D 20
      31 39 38 36

                                ;----------------------------------------------
                                ;       INITIAL RELIABILITY TESTS -- PHASE 1    :
                                ;----------------------------------------------

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

0016  00D5 R                    C1      DW      C11                     ; RETURN ADDRESS
0018  0181 R                    C2      DW      C24                     ; RETURN ADDRESS FOR DUMMY STACK
001A  20 4B 42 20 4F 4B         F3B     DB      ' KB OK',CR             ; KB FOR MEMORY SIZE
      0D
                                ;----------------------------------------------------------------------
                                ;       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 JUST BELOW       :
                                ;       THE TEST CODE. THIS ROUTINE ASSUMES THAT THE FRIST 2          :
                                ;       BYTES TRANSFERED CONTAIN THE COUNT OF BYTES TO BE LOADED      :
                                ;       (BYTE 1=COUNT LOW, BYTE 2=COUNT HI.)                          :
                                ;----------------------------------------------------------------------

                                ;----- FIRST, GET THE COUNT

0021                            MFG_BOOT:
0021  E8 19F0 R                         CALL    SP_TEST                 ; GET COUNT LOW
0024  8A FB                             MOV     BH,BL                   ; SAVE IT
0026  E8 19F0 R                         CALL    SP_TEST                 ; GET COUNT HI
0029  8A EB                             MOV     CH,BL
002B  8A CF                             MOV     CL,BH                   ; CX NOW HAS COUNT
002D  FC                                CLD                             ; SET DIR. FLAG TO INCRIMENT
002E  FA                                CLI
002F  BF 0500                           MOV     DI,0500H                ; SET TARGET OFFSET (DS=0000)
0032  B0 FD                             MOV     AL,0FDH                 ; UNMASK K/B INTERRUPT
0034  E6 21                             OUT     INTA01,AL
0036  B0 0A                             MOV     AL,0AH                  ; SEND READ INT. REQUEST REG. CMD
0038  E6 20                             OUT     INTA00,AL
003A  BA 0061                           MOV     DX,PORT_B               ; SET UP PORT B ADDRESS
```

```
003D BB 4CCC                  MOV     BX,4CCCH                 ; CONTROL BITS FOR PORT B
0040 B4 02                    MOV     AH,02H                   ; K/B REQUEST PENDING MASK
0042                   TST:
0042 8A C3                    MOV     AL,BL
0044 EE                       OUT     DX,AL                    ; TOGGLE K/B CLOCK
0045 8A C7                    MOV     AL,BH
0047 EE                       OUT     DX,AL
0048 4A                       DEC     DX                       ; POINT DX AT ADDR. 60 (KB DATA)
0049                   TST1:
0049 E4 20                    IN      AL,INTA00                ; GET IRR REG
004B 22 C4                    AND     AL,AH                    ; KB REQUEST PENDING?
004D 74 FA                    JZ      TST1                     ; LOOP TILL DATA PRESENT
004F EC                       IN      AL,DX                    ; GET DATA
0050 AA                       STOSB                            ; STORE IT
0051 42                       INC     DX                       ; POINT DX BACK AT PORT B (61)
0052 E2 EE                    LOOP    TST                      ; LOOP TILL ALL BYTES READ
0054 EA 0500 ---- R           JMP     @MFG_TEST_RTN            ; FAR JUMP TO CODE THAT WAS JUST
                                                               ; LOADED
                       ;----------------------------------------
                       ;       8088 PROCESSOR TEST            :
                       ; DESCRIPTION                          :
                       ;       VERIFY 8088 FLAGS, REGISTERS   :
                       ;       AND CONDITIONAL JUMPS          :
                       ;----------------------------------------
                              ASSUME  CS:CODE,DS:NOTHING,ES:NOTHING,SS:NOTHING
                       ;      ORG     0E05BH
                              ORG     0005BH
005B                   RESET:
005B FA                START: CLI                              ; DISABLE INTERRUPTS
005C B4 D5                    MOV     AH,0D5H                  ; SET SF, CF, ZF, AND AF FLAGS ON
005E 9E                       SAHF
005F 73 4A                    JNC     ERR01                    ; GO TO ERROR ROUTINE IF CF NOT SET
0061 75 48                    JNZ     ERR01                    ; GO TO ERROR ROUTINE IF ZF NOT SET
0063 7B 46                    JNP     ERR01                    ; GO TO ERROR ROUTINE IF PF NOT SET
0065 79 44                    JNS     ERR01                    ; GO TO ERROR ROUTINE IF SF NOT SET
0067 9F                       LAHF                             ; LOAD FLAG IMAGE TO AH
0068 B1 05                    MOV     CL,5                     ; LOAD COUNT REG WITH SHIFT COUNT
006A D2 EC                    SHR     AH,CL                    ; SHIFT AF INTO CARRY BIT POSITION
006C 73 3D                    JNC     ERR01                    ; GO TO ERROR ROUTINE IF AF NOT SET
006E B0 40                    MOV     AL,40H                   ; SET THE OF FLAG ON
0070 D0 E0                    SHL     AL,1                     ; SETUP FOR TESTING
0072 71 37                    JNO     ERR01                    ; GO TO ERROR ROUTINE IF OF NOT SET
0074 32 E4                    XOR     AH,AH                    ; SET AH = 0
0076 9E                       SAHF                             ; CLEAR SF, CF, ZF, AND PF
0077 76 32                    JBE     ERR01                    ; GO TO ERROR ROUTINE IF CF ON
                                                               ; GO TO ERROR ROUTINE IF ZF ON
0079 78 30                    JS      ERR01                    ; GO TO ERROR ROUTINE IF SF ON
007B 7A 2E                    JP      ERR01                    ; GO TO ERROR ROUTINE IF PF ON
007D 9F                       LAHF                             ; LOAD FLAG IMAGE TO AH
007E D2 EC                    SHR     AH,CL                    ; SHIFT 'AF' INTO CARRY BIT POSITION
0080 72 29                    JC      ERR01                    ; GO TO ERROR ROUTINE IF ON
0082 D0 E4                    SHL     AH,1                     ; CHECK THAT 'OF' IS CLEAR
0084 70 25                    JO      ERR01                    ; GO TO ERROR ROUTINE IF ON

                       ;----- READ/WRITE THE 8088 GENERAL AND SEGMENTATION REGISTERS
                       ;      WITH ALL ONE'S AND ZEROES'S.

0086 B8 FFFF                  MOV     AX,0FFFFH                ; SETUP ONE'S PATTERN IN AX
0089 F9                       STC
008A 8E D8             C8:    MOV     DS,AX                    ; WRITE PATTERN TO ALL REGS
008C 8C DB                    MOV     BX,DS
008E 8E C3                    MOV     ES,BX
0090 8C C1                    MOV     CX,ES
0092 8E D1                    MOV     SS,CX
0094 8C D2                    MOV     DX,SS
0096 8B E2                    MOV     SP,DX
0098 8B EC                    MOV     BP,SP
009A 8B F5                    MOV     SI,BP
009C 8B FE                    MOV     DI,SI
009E 73 07                    JNC     C9                       ; TST1A
00A0 33 C7                    XOR     AX,DI                    ; PATTERN MAKE IT THRU ALL REGS
00A2 75 07                    JNZ     ERR01                    ; NO - GO TO ERR ROUTINE
00A4 F8                       CLC
00A5 EB E3                    JMP     C8
00A7                   C9:                                     ; TST1A
00A7 0B C7                    OR      AX,DI                    ; ZERO PATTERN MAKE IT THRU?
00A9 74 01                    JZ      C10                      ; YES - GO TO NEXT TEST
500AB F4               ERR01: HLT                              ; HALT SYSTEM
                       ;----------------------------------------
                       ;       ROS CHECKSUM TEST I            :
                       ; DESCRIPTION                          :
                       ;       A CHECKSUM IS DONE FOR THE 8K  :
                       ;       ROS MODULE CONTAINING POD AND  :
                       ;       BIOS.                          :
                       ;----------------------------------------
00AC                   C10:
                                                               ; ZERO IN AL ALREADY
00AC E6 A0                    OUT     0A0H,AL                  ; DISABLE NMI INTERRUPTS
00AE E6 83                    OUT     83H,AL                   ; INITIALZE DMA PAGE REG
00B0 BA 03D8                  MOV     DX,3D8H
00B3 EE                       OUT     DX,AL                    ; DISABLE COLOR VIDEO
00B4 FE C0                    INC     AL
00B6 B2 B8                    MOV     DL,0B8H
00B8 EE                       OUT     DX,AL                    ; DISABLE B/W VIDEO,EN HIGH RES
00B9 B0 89                    MOV     AL,89H                   ; SET 8255 FOR B,A=OUT, C=IN
00BB E6 63                    OUT     CMD_PORT,AL
00BD B0 A5                    MOV     AL,10100101B
                                                               ; ENABLE PARITY CHECKERS AND
00BF E6 61                    OUT     PORT_B,AL                ; PULL KB CLOCK LOW, TRI-STATE
                                                               ; KEYBOARD INPUTS,ENABLE HIGH
                                                               ; BANK OF SWITCHES->PORT C(0-3)
00C1 B0 01                    MOV     AL,01H                   ; <><><><><><><><><><><><>
00C3 E6 60                    OUT     PORT_A,AL                ; <><><>CHECKPOINT 1<><><>
00C5 8C C8                    MOV     AX,CS                    ; SETUP SS SEG REG
00C7 8E D0                    MOV     SS,AX
00C9 8E D8                    MOV     DS,AX                    ; SET UP DATA SEG TO POINT TO
                                                               ; ROM ADDRESS
00CB FC                       CLD                              ; SET DIRECTION FLAG TO INC.
                              ASSUME  SS:CODE
00CC BB 0000                  MOV     BX,00000H                ; SETUP STARTING ROS ADDR
00CF BC 0016 R                MOV     SP,OFFSET C1             ; SETUP RETURN ADDRESS   (p5-10)
00D2 E9 18BF R                JMP     ROS_CHECKSUM
00D5 75 D4             C11:   JNE     ERR01                    ; HALT SYSTEM IF ERROR
```

SECTION 5

```
                              PAGE
                              ;-------------------------------------------------------
                              ;       8237 DMA INITIALIZATION CHANNEL REGISTER TEST   :
                              ; DESCRIPTION                                           :
                              ;       DISABLE THE 8237 DMA CONTROLLER.  VERIFY THAT   :
                              ;       TIMER 1 FUNCTIONS OK. WRITE/READ THE CURRENT    :
                              ;       ADDRESS AND WORD COUNT REGISTERS FOR ALL        :
                              ;       CHANNELS.  INITIALIZE AND START DMA FOR MEMORY  :
                              ;       REFRESH.                                        :
                              ;-------------------------------------------------------

                              ;----- DISABLE DMA CONTROLLER

 00D7  B0 02                          MOV     AL,02H                  ; <><><><><><><><><><><><>
 00D9  E6 60                          OUT     PORT_A,AL               ; <><><>CHECKPOINT 2<><><>
 00DB  B0 04                          MOV     AL,04                   ; DISABLE DMA CONTROLLER
 00DD  E6 08                          OUT     DMA08,AL

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

 00DF  B0 54                          MOV     AL,54H                  ; SEL TIMER 1,LSB,MODE 2
 00E1  E6 43                          OUT     TIMER+3,AL
 00E3  8A C1                          MOV     AL,CL                   ; SET INITIAL TIMER CNT TO 0
 00E5  E6 41                          OUT     TIMER+1,AL
 00E7                         C12:                                    ; TIMER1_BITS_ON
 00E7  B0 40                          MOV     AL,40H                  ; LATCH TIMER 1 COUNT
 00E9  E6 43                          OUT     TIMER+3,AL
 00EB  80 FB FF                       CMP     BL,0FFH                 ; YES - SEE IF ALL BITS GO OFF
 00EE  74 07                          JE      C13                     ; TIMER1_BITS_OFF
 00F0  E4 41                          IN      AL,TIMER+1              ; READ TIMER 1 COUNT
 00F2  0A D8                          OR      BL,AL                   ; ALL BITS ON IN TIMER
 00F4  E2 F1                          LOOP    C12                     ; TIMER1_BITS_ON
 00F6  F4                             HLT                             ; TIMER 1 FAILURE, HALT SYS
 00F7                         C13:                                    ; TIMER1_BITS_OFF
 00F7  8A C3                          MOV     AL,BL                   ; SET TIMER 1 CNT
 00F9  2B C9                          SUB     CX,CX
 00FB  E6 41                          OUT     TIMER+1,AL
 00FD                         C14:                                    ; TIMER_LOOP
 00FD  B0 40                          MOV     AL,40H                  ; LATCH TIMER 1 COUNT
 00FF  E6 43                          OUT     TIMER+3,AL
 0101  90                             NOP                             ; DELAY FOR TIMER
 0102  90                             NOP
 0103  E4 41                          IN      AL,TIMER+1              ; READ TIMER 1 COUNT
 0105  22 D8                          AND     BL,AL
 0107  74 03                          JZ      C15                     ; WRAP_DMA_REG
 0109  E2 F2                          LOOP    C14                     ; TIMER_LOOP
 010B  F4                             HLT                             ; HALT SYSTEM

                              ;----- INITIALIZE TIMER 1 TO REFRESH MEMORY

 010C  B0 03                  C15:    MOV     AL,03H                  ; <><><><><><><><><><><><>
 010E  E6 60                          OUT     PORT_A,AL               ; <><><>CHECKPOINT 3<><><>
                                                                      ; WRAP_DMA_REG
 0110  E6 0D                          OUT     DMA+0DH,AL              ; SEND MASTER CLEAR TO DMA

                              ;----- WRAP DMA CHANNELS ADDRESS AND COUNT REGISTERS

 0112  B0 FF                          MOV     AL,0FFH                 ; WRITE PATTERN FF TO ALL REGS
 0114  8A D8                  C16:    MOV     BL,AL                   ; SAVE PATTERN FOR COMPARE
 0116  8A F8                          MOV     BH,AL
 0118  B9 0008                        MOV     CX,8                    ; SETUP LOOP CNT
 011B  BA 0000                        MOV     DX,DMA                  ; SETUP I/O PORT ADDR OF REG
 011E  EE                     C17:    OUT     DX,AL                   ; WRITE PATTERN TO REG, LSB
 011F  50                             PUSH    AX                      ; SATISFY 8237 I/O TIMINGS
 0120  EE                             OUT     DX,AL                   ; MSB OF 16 BIT REG
 0121  B0 01                          MOV     AL,01H                  ; AL TO ANOTHER PAT BEFORE RD
 0123  EC                             IN      AL,DX                   ; READ 16-BIT DMA CH REG, LSB
 0124  8A E0                          MOV     AH,AL                   ; SAVE LSB OF 16-BIT REG
 0126  EC                             IN      AL,DX                   ; READ MSB OF DMA CH REG
 0127  3B D8                          CMP     BX,AX                   ; PATTERN READ AS WRITTEN?
 0129  74 01                          JE      C18                     ; YES - CHECK NEXT REG
 012B                         C17A:
 012B  F4                             HLT                             ; NO - HALT THE SYSTEM
 012C                         C18:                                    ; NXT_DMA_CH
 012C  42                             INC     DX                      ; SET I/O PORT TO NEXT CH REG
 012D  F9                             STC
 012E  E2 EE                          LOOP    C17                     ; WRITE PATTERN TO NEXT REG
 0130  73 F9                          JNC     C17A                    ; IF CARRY NOT SET HALT SYSTEM
 0132  FE C0                          INC     AL                      ; SET PATTERN TO 0
 0134  74 DE                          JZ      C16                     ; WRITE TO CHANNEL REGS

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

 0136  8E DB                          MOV     DS,BX                   ; SET UP ABS0 INTO DS AND ES
 0138  8E C3                          MOV     ES,BX
                                      ASSUME  DS:ABS0,ES:ABS0
 013A  B0 FF                          MOV     AL,0FFH                 ; SET CNT OF 64K FOR REFRESH
 013C  E6 01                          OUT     DMA+1,AL
 013E  50                             PUSH    AX
 013F  E6 01                          OUT     DMA+1,AL
 0141  B0 58                          MOV     AL,058H                 ; SET DMA MODE,CH 0,RD.,AUOTINT
 0143  E6 0B                          OUT     DMA+0BH,AL              ; WRITE DMA MODE REG
 0145  B0 00                          MOV     AL,0                    ; ENABLE DMA CONTROLLER
 0147  8A E8                          MOV     CH,AL                   ; SET COUNT HIGH=00
 0149  E6 08                          OUT     DMA+8,AL                ; SETUP DMA COMMAND REG
 014B  50                             PUSH    AX
 014C  E6 0A                          OUT     DMA+10,AL               ; ENABLE DMA CH 0
 014E  B0 12                          MOV     AL,18                   ; START TIMER 1
 0150  E6 41                          OUT     TIMER+1,AL
 0152  B0 41                          MOV     AL,41H                  ; SET MODE FOR CHANNEL 1
 0154  E6 0B                          OUT     DMA+0BH,AL
 0156  50                             PUSH    AX
 0157  E4 08                          IN      AL,DMA+08               ; GET DMA STATUS
 0159  24 10                          AND     AL,00010000B            ; IS TIMER REQUEST THERE?
 015B  74 01                          JZ      C18C                    ; (IT SHOULD'T BE)
 015D  F4                             HLT                             ; HALT SYS.(HOT TIMER 1 OUTPUT)
 015E  B0 42                  C18C:   MOV     AL,42H                  ; SET MODE FOR CHANNEL 2
 0160  E6 0B                          OUT     DMA+0BH,AL
 0162  B0 43                          MOV     AL,43H                  ; SET MODE FOR CHANNEL 3
 0164  E6 0B                          OUT     DMA+0BH,AL
```

```
                                   PAGE
                                   ;-------------------------------------------------------
                                   ;          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.     :
                                   ;-------------------------------------------------------

 0166 AD                                   LODSW                        ; ALLOW RAM CHARGE TIME.
 0167 AD                                   LODSW
 0168 AD                                   LODSW
 0169 AD                                   LODSW

                                   ;----- DETERMINE MEMORY SIZE AND FILL MEMORY WITH DATA

 016A 8B 1E 0472 R                         MOV    BX,DATA_WORD[●RESET_FLAG-DATA40]  ; SAVE 'RESET_FLAG' IN BX
 016E 8B 2E 0496 R                         MOV    BP,DATA_WORD[●KB_FLAG_3-DATA40]  ; SAVE KEYBOARD TYPE
 0172 B9 8000                              MOV    CX,08000H              ; SET FOR 32K WORDS
 0175 81 FB 1234                           CMP    BX,1234H               ; WARM START?
 0179 74 16                                JE     CLR_STG
 017B BC 0018 R                            MOV    SP,OFFSET C2
 017E E9 0CCF R                            JMP    STGTST_CNT
 0181 74 12                        C24:    JE     HOW_BIG                ; STORAGE OK, DETERMINE SIZE
 0183 8A D8                                MOV    BL,AL                  ; SAVE FAILING BIT PATTERN
 0185 B0 04                                MOV    AL,04H                 ; <><><><><><><><><><><><>
 0187 E6 60                        C24A:   OUT    PORT_A,AL              ; <><><>CHECKPOINT 4<><><>
 0189 2B C9                                SUB    CX,CX                  ; BASE RAM FAILURE - HANG
 018B E2 FE                        C24B:   LOOP   C24B                   ; FLIPPING BETWEEN 04 AND
 018D 86 D8                                XCHG   BL,AL                  ; FAILING BIT PATTERN
 018F EB F6                                JMP    C24A
 0191                              CLR_STG:
 0191 2B C0                                SUB    AX,AX                  ; MAKE AX=0000
 0193 F3/ AB                               REP    STOSW                  ; STORE 32K WORDS OF 0000
 0195                              HOW_BIG:
 0195 89 1E 0472 R                         MOV    DATA_WORD[●RESET_FLAG-DATA40],BX       ; RESTORE RESET FLAG
 0199 83 FD 10                             CMP    BP,KBX                 ; IS THE KBX BIT THE ONLY ONE ON?
 019C 74 02                                JE     C24C                   ; IF NOT THEN THIS MUST BE A P.O.R.
 019E 2B ED                                SUB    BP,BP                  ; IF P.O.R. THEN INITIALIZE THIS TO ZERO
 01A0                              C24C:
 01A0 89 2E 0496 R                         MOV    DATA_WORD[●KB_FLAG_3-DATA40],BP  ; RESTORE RESET FLAG
 01A4 2B ED                                SUB    BP,BP                  ; BP IS USED LATER AS AN ERROR INDICATOR
 01A6 BA 0400                              MOV    DX,0400H               ; SET POINTER TO JUST>16KB
 01A9 BB 0010                              MOV    BX,16                  ; BASIC COUNT OF 16K
 01AC                              FILL_LOOP:
 01AC 8E C2                                MOV    ES,DX                  ; SET SEG. REG.
 01AE 2B FF                                SUB    DI,DI
 01B0 B8 AA55                              MOV    AX,0AA55H              ; TEST PATTERN
 01B3 8B C8                                MOV    CX,AX                  ; SAVE PATTERN
 01B5 26: 89 05                            MOV    ES:[DI],AX             ; SEND PATTERN TO MEM.
 01B8 B0 0F                                MOV    AL,0FH                 ; PUT SOMETHING IN AL
 01BA 26: 8B 05                            MOV    AX,ES:[DI]             ; GET PATTERN
 01BD 33 C1                                XOR    AX,CX                  ; COMPARE PATTERNS
 01BF 75 11                                JNZ    HOW_BIG_END            ; GO END IF NO COMPARE
 01C1 B9 2000                              MOV    CX,2000H               ; SET COUNT FOR 8K WORDS
 01C4 F3/ AB                               REP    STOSW                  ; FILL 8K WORDS
 01C6 81 C2 0400                           ADD    DX,400H                ; POINT TO NEXT 16KB BLOCK
 01CA 83 C3 10                             ADD    BX,16                  ; BUMP COUNT BY 16KB
 01CD 80 FE A0                             CMP    DH,0A0H                ; TOP OF RAM AREA YET? (A0000)
 01D0 75 DA                                JNZ    FILL_LOOP
 01D2                              HOW_BIG_END:
 01D2 89 1E 0413 R                         MOV    DATA_WORD[●MEMORY_SIZE-DATA40],BX      ; SAVE MEMORY SIZE

                                   ;----- SETUP STACK SEG AND SP

 01D6 B8 0030                              MOV    AX,STACK_SS            ; GET STACK VALUE
 01D9 8E D0                                MOV    SS,AX                  ; SET THE STACK UP
 01DB BC 0100                              MOV    SP,TOS                 ; STACK IS READY TO GO
                                   ;-------------------------------------------------------
                                   ;       INITIALIZE THE 8259 INTERRUPT CONTROLLER CHIP   :
                                   ;-------------------------------------------------------
 01DE B0 13                        C25:    MOV    AL,13H                 ; ICW1 - EDGE, SNGL, ICW4
 01E0 E6 20                                OUT    INTA00,AL
 01E2 B0 08                                MOV    AL,8                   ; SETUP ICW2 - INT TYPE 8 (8-F)
 01E4 E6 21                                OUT    INTA01,AL
 01E6 B0 09                                MOV    AL,9                   ; SETUP ICW4 - BUFFRD,8086 MODE
 01E8 E6 21                                OUT    INTA01,AL
 01EA B0 FF                                MOV    AL,0FFH                ; MASK ALL INTS. OFF
 01EC E6 21                                OUT    INTA01,AL              ; (VIDEO ROUTINE ENABLES INTS.)

                                   ;----- SET UP THE INTERRUPT VECTORS TO TEMP INTERRUPT

 01EE 1E                                   PUSH   DS
 01EF B9 0020                              MOV    CX,32                  ; FILL ALL 32 INTERRUPTS
 01F2 2B FF                                SUB    DI,DI                  ; FIRST INTERRUPT LOCATION
 01F4 8E C7                                MOV    ES,DI                  ; SET ES=0000 ALSO
 01F6 B8 1F23 R                    D3:     MOV    AX,OFFSET D11          ; MOVE ADDR OF INTR PROC TO TBL
 01F9 AB                                   STOSW
 01FA 8C C8                                MOV    AX,CS                  ; GET ADDR OF INTR PROC SEG
 01FC AB                                   STOSW
 01FD E2 F7                                LOOP   D3                     ; VECTBL0

                                   ;----- ESTABLISH BIOS SUBROUTINE CALL INTERRUPT VECTORS

 01FF BF 0040 R                            MOV    DI,OFFSET ●VIDEO_INT   ; SETUP ADDR TO INTR AREA
 0202 0E                                   PUSH   CS
 0203 1F                                   POP    DS                     ; SETUP ADDR OF VECTOR TABLE
 0204 BE 1F03 R                            MOV    SI,OFFSET VECTOR_TABLE+16      ; START WITH VIDEO ENTRY
 0207 B9 0010                              MOV    CX,16
 020A A5                           D3A:    MOVSW                         ; MOVE VECTOR TABLE TO RAM
 020B 47                                   INC    DI                     ; SKIP SEGMENT POINTER
 020C 47                                   INC    DI
 020D E2 FB                                LOOP   D3A
                                   ;-------------------------------------------------------
                                   ;       DETERMINE CONFIGURATION AND MFG. MODE          :
                                   ;-------------------------------------------------------

 020F 1F                                   POP    DS
 0210 1E                                   PUSH   DS                     ; RECOVER DATA SEG
 0211 E4 62                                IN     AL,PORT_C              ; GET SWITCH INFO
 0213 24 0F                                AND    AL,00001111B           ; ISOLATE SWITCHES
 0215 8A E0                                MOV    AH,AL                  ; SAVE
 0217 B0 AD                                MOV    AL,10101101B           ; ENABLE OTHER BANK OF SWS.
 0219 E6 61                                OUT    PORT_B,AL
```

SECTION 5

```
021B 90                              NOP
021C E4 62                           IN      AL,PORT_C
021E B1 04                           MOV     CL,4
0220 D2 C0                           ROL     AL,CL                      ; ROTATE TO HIGH NIBBLE
0222 24 F0                           AND     AL,11110000B               ; ISOLATE
0224 0A C4                           OR      AL,AH                      ; COMBINE WITH OTHER BANK
0226 2A E4                           SUB     AH,AH
0228 A3 0410 R                       MOV     DATA_WORD[@EQUIP_FLAG-DATA40],AX        ; SAVE SWITCH INFO
022B B0 99                           MOV     AL,99H
022D E6 63                           OUT     CMD_PORT,AL
022F E8 19E3 R                       CALL    KBD_RESET                  ; SEE IF MFG. JUMPER IN
0232 80 FB EA                        CMP     BL,0EAH                    ; IS THIS THE EXTENDED KEYBOARD?
0235 75 08                           JNE     KBX1                       ; IF NOT THEN LEAVE THE FLAG ALONE
0237 C6 06 0496 R 10                 MOV     DATA_AREA[@KB_FLAG_3-DATA40],KBX   ; EXTENDED KEYBOARD
023C EB 22 90                        JMP     E6                         ; DONE WITH KEYBOARD HERE
023F                         KBX1:
023F 80 FB AA                        CMP     BL,0AAH                    ; KEYBOARD PRESENT?
0242 74 1C                           JE      E6
0244 80 FB 65                        CMP     BL,065H                    ; LOAD MFG. TEST REQUEST?
0247 75 03                           JNE     D3B
0249 E9 0021 R                       JMP     MFG_BOOT                   ; GO TO BOOTSTRAP IF SO
024C                         D3B:
024C 0A DB                           OR      BL,BL                      ; MFG PLUG IN?
024E 75 10                           JNZ     E6                         ; NO
0250 B0 38                           MOV     AL,38H
0252 E6 61                           OUT     PORT_B,AL
0254 90                              NOP
0255 90                              NOP
0256 E4 60                           IN      AL,PORT_A
0258 24 FF                           AND     AL,0FFH                    ; WAS DATA LINE GROUNDED
025A 75 04                           JNZ     E6
025C FE 06 0412 R                    INC     DATA_AREA[@MFG_TST-DATA40]       ; SET MANUFACTURING TEST FLAG

                             ;-----------------------------------------------------------
                             ;       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                          :
                             ;-----------------------------------------------------------
0260                         E6:
0260 A1 0410 R                       MOV     AX,DATA_WORD[@EQUIP_FLAG-DATA40]        ; GET SENSE SWITCH INFO
0263 50                              PUSH    AX                         ; SAVE IT
0264 B0 30                           MOV     AL,30H
0266 A3 0410 R                       MOV     DATA_WORD[@EQUIP_FLAG-DATA40],AX
0269 2A E4                           SUB     AH,AH
026B CD 10                           INT     10H                        ; SEND INIT TO B/W CARD
026D B0 20                           MOV     AL,20H
026F A3 0410 R                       MOV     DATA_WORD[@EQUIP_FLAG-DATA40],AX
0272 2A E4                           SUB     AH,AH                      ; AND INIT COLOR CARD
0274 CD 10                           INT     10H
0276 58                              POP     AX                         ; RECOVER REAL SWITCH INFO
0277 A3 0410 R                       MOV     DATA_WORD[@EQUIP_FLAG-DATA40],AX    ; RESTORE IT
                                                                        ; AND CONTINUE
027A 24 30                           AND     AL,30H                     ; ISOLATE VIDEO SWS
027C 75 0A                           JNZ     E7                         ; VIDEO SWS SET TO 0?
027E BF 0040 R                       MOV     DI,OFFSET @VIDEO_INT       ; SET INT 10H TO DUMMY
0281 C7 05 1F49 R                    MOV     WORD PTR [DI],OFFSET DUMMY_RETURN    ; RET IF NO VIDEO CARD
0285 E9 033B R                       JMP     E18_1                      ; BYPASS VIDEO TEST
0288                         E7:                                        ; TEST_VIDEO:
0288 3C 30                           CMP     AL,30H                     ; B/W CARD ATTACHED?
028A 74 08                           JE      E8                         ; YES - SET MODE FOR B/W CARD
028C FE C4                           INC     AH                         ; SET COLOR MODE FOR COLOR CD
028E 3C 20                           CMP     AL,20H                     ; 80X25 MODE SELECTED?
0290 75 02                           JNE     E8                         ; NO - SET MODE FOR 40X25
0292 B4 03                           MOV     AH,3                       ; SET MODE FOR 80X25
0294 86 E0                   E8:     XCHG    AH,AL                      ; SET_MODE:
0296 50                              PUSH    AX                         ; SAVE VIDEO MODE ON STACK
0297 2A E4                           SUB     AH,AH                      ; INITIALIZE TO ALPHANUMERIC MD
0299 CD 10                           INT     10H                        ; CALL VIDEO_IO
029B 58                              POP     AX                         ; RESTORE VIDEO SENSE SWS IN AH
029C 50                              PUSH    AX                         ; RESAVE VALUE
029D BB B000                         MOV     BX,0B000H                  ; BEG VIDEO RAM ADDR B/W CD
02A0 EB 24                           JMP     SHORT E8A

                             ;----- UNNATURAL ACT FOR ADDRESS COMPATIBILITY

                             ;       ORG     0E2C3H
02C3                                 ORG     002C3H
02C3                         NMI_INT:
02C3 E9 185C R                       JMP     NMI_INT_1

02C6                         E8A:
02C6 BA 03B8                         MOV     DX,3B8H                    ; MODE REG FOR B/W
02C9 B9 0800                         MOV     CX,2048                    ; RAM WORD CNT FOR B/W CD
02CC B0 01                           MOV     AL,1                       ; SET MODE FOR BW CARD
02CE 80 FC 30                        CMP     AH,30H                     ; B/W VIDEO CARD ATTACHED?
02D1 74 09                           JE      E9                         ; YES - GO TEST VIDEO STG
02D3 B7 B8                           MOV     BH,0B8H                    ; BEG VIDEO RAM ADDR COLOR CD
02D5 BA 03D8                         MOV     DX,3D8H                    ; MODE REG FOR COLOR CD
02D8 B5 20                           MOV     CH,20H                     ; RAM WORD CNT FOR COLOR CD
02DA FE C8                           DEC     AL                         ; SET MODE TO 0 FOR COLOR CD
02DC                         E9:                                        ; TEST_VIDEO_STG:
02DC EE                              OUT     DX,AL                      ; DISABLE VIDEO FOR COLOR CD
02DD 81 3E 0472 R 1234               CMP     DATA_WORD[@RESET_FLAG-DATA40],1234H ; POD INIT BY KBD RESET?
02E3 8E C3                           MOV     ES,BX                      ; POINT ES TO VIDEO RAM STG
02E5 74 07                           JE      E10                        ; YES - SKIP VIDEO RAM TEST
02E7 8E DB                           MOV     DS,BX                      ; POINT DS TO VIDEO RAM STG
                                     ASSUME  DS:NOTHING,ES:NOTHING
02E9 E8 0CCF R                       CALL    STGTST_CNT                 ; GO TEST VIDEO R/W STG
02EC 75 33                           JNE     E17                        ; R/W STG FAILURE - BEEP SPK
                             ;-----------------------------------------------------
                             ;       SETUP VIDEO DATA ON SCREEN FOR VIDEO          :
                             ;       LINE TEST.                                    :
                             ; DESCRIPTION                                         :
                             ;       ENABLE VIDEO SIGNAL AND SET MODE.             :
                             ;       DISPLAY A HORIZONTAL BAR ON SCREEN.           :
                             ;-----------------------------------------------------
02EE                         E10:
02EE 58                              POP     AX                         ; GET VIDEO SENSE SWS (AH)
02EF 50                              PUSH    AX                         ; SAVE IT
```

```
02F0 B4 00                      MOV     AH,0                    ; ENABLE VIDEO AND SET MODE
02F2 CD 10                      INT     10H                     ; VIDEO
02F4 B8 7020                    MOV     AX,7020H                ; WRT BLANKS IN REVERSE VIDEO

02F7 2B FF                      SUB     DI,DI                   ; SETUP STARTING LOC
02F9 B9 0028                    MOV     CX,40                   ; NO. OF BLANKS TO DISPLAY
02FC F3/ AB                     REP     STOSW                   ; WRITE VIDEO STORAGE
                        ;----------------------------------------
                        ;       CRT INTERFACE LINES TEST        :
                        ; DESCRIPTION                           :
                        ;       SENSE ON/OFF TRANSITION OF THE  :
                        ;       VIDEO ENABLE AND HORIZONTAL     :
                        ;       SYNC LINES.                     :
                        ;----------------------------------------
02FE 58                         POP     AX                      ; GET VIDEO SENSE SW INFO
02FF 50                         PUSH    AX                      ; SAVE IT
0300 80 FC 30                   CMP     AH,30H                  ; B/W CARD ATTACHED?
0303 BA 03BA                    MOV     DX,03BAH                ; SETUP ADDR OF BW STATUS PORT
0306 74 03                      JE      E11                     ; YES - GO TEST LINES
0308 BA 03DA                    MOV     DX,03DAH                ; COLOR CARD IS ATTACHED
030B                    E11:                                    ; LINE_TST:
030B B4 08                      MOV     AH,8
030D                    E12:                                    ; OFLOOP_CNT:
030D 2B C9                      SUB     CX,CX
030F                    E13:
030F EC                         IN      AL,DX                   ; READ CRT STATUS PORT
0310 22 C4                      AND     AL,AH                   ; CHECK VIDEO/HORZ LINE
0312 75 04                      JNZ     E14                     ; ITS ON - CHECK IF IT GOES OFF
0314 E2 F9                      LOOP    E13                     ; LOOP TILL ON OR TIMEOUT
0316 EB 09                      JMP     SHORT E17               ; GO PRINT ERROR MSG
0318                    E14:
0318 2B C9                      SUB     CX,CX
031A                    E15:
031A EC                         IN      AL,DX                   ; READ CRT STATUS PORT
031B 22 C4                      AND     AL,AH                   ; CHECK VIDEO/HORZ LINE
031D 74 11                      JZ      E16                     ; ITS ON - CHECK NEXT LINE
031F E2 F9                      LOOP    E15                     ; LOOP IF OFF TILL IT GOES ON
0321                    E17:                                    ; CRT_ERR:
0321 1F                         POP     DS
0322 1E                         PUSH    DS
0323 C6 06 0015 R 06            MOV     DS:@MFG_ERR_FLAG,06H    ; <><><>CRT ERR CHKPT. 06<><><>
0328 BA 0102                    MOV     DX,102H
032B E8 19A5 R                  CALL    ERR_BEEP                ; GO BEEP SPEAKER
032E EB 06                      JMP     SHORT E18
0330                    E16:                                    ; NXT_LINE:
0330 B1 03                      MOV     CL,3                    ; GET NEXT BIT TO CHECK
0332 D2 EC                      SHR     AH,CL
0334 75 D7                      JNZ     E12                     ; GO CHECK HORIZONTAL LINE
0336                    E18:                                    ; DISPLAY_CURSOR:
0336 58                         POP     AX                      ; GET VIDEO SENSE SWS (AH)
0337 B4 00                      MOV     AH,0                    ; SET MODE AND DISPLAY CURSOR
0339 CD 10                      INT     10H                     ; CALL VIDEO I/O PROCEDURE
033B                    E18_1:
033B BA C000                    MOV     DX,0C000H               ; SEE IF ADVANCED VIDEO CARD
033E                    E18A:
033E 8E DA                      MOV     DS,DX                   ; IS PRESENT
0340 2B DB                      SUB     BX,BX
0342 8B 07                      MOV     AX,[BX]                 ; GET FIRST 2 LOCATIONS
0344 53                         PUSH    BX
0345 5B                         POP     BX                      ; LET BUS SETTLE
0346 3D AA55                    CMP     AX,0AA55H               ; PRESENT?
0349 75 05                      JNZ     E18B                    ; NO? GO LOOK FOR OTHER MODULES
034B E8 1920 R                  CALL    ROM_CHECK               ; GO SCAN MODULE
034E EB 04                      JMP     SHORT E18C
0350                    E18B:
0350 81 C2 0080                 ADD     DX,0080H                ; POINT TO NEXT 2K BLOCK
0354                    E18C:
0354 81 FA C800                 CMP     DX,0C800H               ; TOP OF VIDEO ROM AREA YET?
0358 7C E4                      JL      E18A                    ; GO SCAN FOR ANOTHER MODULE
                        ;----------------------------------------------------------
                        ;       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  DS:ABS0
035A 1F                 C21:    POP     DS

                        ;----- TEST THE IMR REGISTER

035B C6 06 0415 R 05    C21A:   MOV     DATA_AREA[@MFG_ERR_FLAG-DATA40],05H
                                                                ; <><><><><><><><><><><><>
                                                                ; <><><>CHECKPOINT 5<><><>
0360 B0 00                      MOV     AL,0                    ; SET IMR TO ZERO
0362 E6 21                      OUT     INTA01,AL
0364 E4 21                      IN      AL,INTA01               ; READ IMR
0366 0A C0                      OR      AL,AL                   ; IMR = 0?
0368 75 1B                      JNZ     D6                      ; GO TO ERR ROUTINE IF NOT 0
036A B0 FF                      MOV     AL,0FFH                 ; DISABLE DEVICE INTERRUPTS
036C E6 21                      OUT     INTA01,AL               ; WRITE TO IMR
036E E4 21                      IN      AL,INTA01               ; READ IMR
0370 04 01                      ADD     AL,1                    ; ALL IMR BIT ON?
0372 75 11                      JNZ     D6                      ; NO - GO TO ERR ROUTINE

                        ;----- CHECK FOR HOT INTERRUPTS

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

0374 A2 046B R                  MOV     DATA_AREA[@INTR_FLAG-DATA40],AL ; CLEAR INTERRUPT FLAG
0377 FB                         STI                             ; ENABLE EXTERNAL INTERRUPTS
0378 2B C9                      SUB     CX,CX                   ; WAIT 1 SEC FOR ANY INTRS THAT
037A                    D4:
037A E2 FE                      LOOP    D4                      ; MIGHT OCCUR
037C                    D5:
037C E2 FE                      LOOP    D5
037E 80 3E 046B R 00            CMP     DATA_AREA[@INTR_FLAG-DATA40],00H        ; ANY INTERRUPTS OCCUR?
0383 74 08                      JZ      D7                      ; NO - GO TO NEXT TEST
0385                    D6:
0385 BE 18CC R                  MOV     SI,OFFSET E0            ; DISPLAY 101 ERROR
0388 E8 1976 R                  CALL    E_MSG
038B FA                         CLI
038C F4                         HLT                             ; HALT THE SYSTEM
```

SECTION 5

```
                                        PAGE
                                        ;--------------------------------------------------------
                                        ;        8253 TIMER CHECKOUT                             :
                                        ; DESCRIPTION                                            :
                                        ;        VERIFY THAT THE SYSTEM TIMER (0) DOESN'T COUNT  :
                                        ;        TOO FAST OR TOO SLOW.                           :
                                        ;--------------------------------------------------------
 038D                                   D7:
 038D C6 06 0415 R 02                           MOV     DATA_AREA[@MFG_ERR_FLAG-DATA40],02H
                                                                        ; <><><><><><><><><><><><><><><>
                                                                        ; <><><>TIMER CHECKPOINT (2)<><><>
 0392 B0 FE                                     MOV     AL,0FEH         ; MASK ALL INTRS EXCEPT LVL 0
 0394 E6 21                                     OUT     INTA01,AL       ; WRITE THE 8259 IMR
 0396 B0 10                                     MOV     AL,00010000B    ; SEL TIM 0, LSB, MODE 0, BINARY
 0398 E6 43                                     OUT     TIM_CTL,AL      ; WRITE TIMER CONTROL MODE REG
 039A B9 0016                                   MOV     CX,16H          ; SET PGM LOOP CNT
 039D 8A C1                                     MOV     AL,CL           ; SET TIMER 0 CNT REG
 039F E6 40                                     OUT     TIMER0,AL       ; WRITE TIMER 0 CNT REG
 03A1                                   D8:
 03A1 F6 06 046B R 01                           TEST    DATA_AREA[@INTR_FLAG-DATA40],01H
                                                                        ; DID TIMER 0 INTERRUPT OCCUR?
 03A6 75 04                                     JNZ     D9              ; YES - CHECK TIMER OP FOR SLOW TIME
 03A8 E2 F7                                     LOOP    D8              ; WAIT FOR INTR FOR SPECIFIED TIME
 03AA EB D9                                     JMP     D6              ; TIMER 0 INTR DIDN'T OCCUR - ERR
 03AC                                   D9:
 03AC B1 0C                                     MOV     CL,12           ; SET PGM LOOP CNT
 03AE B0 FF                                     MOV     AL,0FFH         ; WRITE TIMER 0 CNT REG
 03B0 E6 40                                     OUT     TIMER0,AL
 03B2 C6 06 046B R 00                           MOV     DATA_AREA[@INTR_FLAG-DATA40],0  ; RESET INTR RECEIVED FLAG
 03B7 B0 FE                                     MOV     AL,0FEH         ; REENABLE TIMER 0 INTERRUTS
 03B9 E6 21                                     OUT     INTA01,AL
 03BB                                   D10:
 03BB F6 06 046B R 01                           TEST    DATA_AREA[@INTR_FLAG-DATA40],01H     ; TIMER 0 INTERRUPT OCCUR?
 03C0 75 C3                                     JNZ     D6              ; YES - TIMER CNTING TOO FAST, ERR
 03C2 E2 F7                                     LOOP    D10             ; WAIT FOR INTR FOR SPECIFIED TIME

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

 03C4 B0 FF                                     MOV     AL,0FFH         ; DISABLE ALL DEVICE INTERRUPTS
 03C6 E6 21                                     OUT     INTA01,AL
 03C8 B0 36                                     MOV     AL,36H          ; SEL TIM 0,LSB,MSB,MODE 3
 03CA E6 43                                     OUT     TIMER+3,AL      ; WRITE TIMER MODE REG
 03CC B0 00                                     MOV     AL,0
 03CE E6 40                                     OUT     TIMER,AL        ; WRITE LSB TO TIMER 0 REG
 03D0 E6 40                                     OUT     TIMER,AL        ; WRITE MSB TO TIMER 0 REG
                                        ;---------------------------------------------------
                                        ;        KEYBOARD TEST                              :
                                        ; DESCRIPTION                                       :
                                        ;        RESET THE KEYBOARD AND CHECK THAT SCAN     :
                                        ;        CODE 'AA' IS RETURNED TO THE CPU.          :
                                        ;        CHECK FOR STUCK KEYS.                      :
                                        ;---------------------------------------------------
 03D2                                   TST12:
 03D2 B0 99                                     MOV     AL,99H          ; SET 8255 MODE A,C=IN B=OUT
 03D4 E6 63                                     OUT     CMD_PORT,AL
 03D6 A0 0410 R                                 MOV     AL,DATA_AREA[@EQUIP_FLAG-DATA40]
 03D9 24 01                                     AND     AL,01           ; TEST CHAMBER?
 03DB 74 30                                     JZ      F7              ; BYPASS IF SO
 03DD 80 3E 0412 R 01                           CMP     DATA_AREA[@MFG_TST-DATA40],1    ; MANUFACTURING TEST MODE?
 03E2 74 29                                     JE      F7              ; YES - SKIP KEYBOARD TEST
 03E4 E8 19E3 R                                 CALL    KBD_RESET       ; ISSUE RESET TO KEYBRD
 03E7 E3 1E                                     JCXZ    F6              ; PRINT ERR MSG IF NO INTERRUPT
 03E9 B0 49                                     MOV     AL,49H          ; ENABLE KEYBOARD
 03EB E6 61                                     OUT     PORT_B,AL
 03ED 80 FB AA                                  CMP     BL,0AAH         ; SCAN CODE AS EXPECTED?
 03F0 75 15                                     JNE     F6              ; NO - DISPLAY ERROR MSG

                                        ;----- CHECK FOR STUCK KEYS

 03F2 B0 C8                                     MOV     AL,0C8H         ; CLR KBD, SET CLK LINE HIGH
 03F4 E6 61                                     OUT     PORT_B,AL
 03F6 B0 48                                     MOV     AL,48H          ; ENABLE KBD,CLK IN NEXT BYTE
 03F8 E6 61                                     OUT     PORT_B,AL
 03FA 2B C9                                     SUB     CX,CX
 03FC                                   F5:                             ; KBD_WAIT:
 03FC E2 FE                                     LOOP    F5              ; DELAY FOR A WHILE
 03FE E4 60                                     IN      AL,KB_DATA      ; CHECK FOR STUCK KEYS
 0400 3C 00                                     CMP     AL,0            ; SCAN CODE = 0?
 0402 74 09                                     JE      F7              ; YES - CONTINUE TESTING
 0404 E8 1958 R                                 CALL    XPC_BYTE        ; CONVERT AND PRINT
 0407                                   F6:
 0407 BE 098A R                                 MOV     SI,OFFSET F1    ; GET MSG ADDR
 040A E8 1976 R                                 CALL    E_MSG           ; PRINT MSG ON SCREEN
                                        ;---------------------------------------------------
                                        ;        SETUP HARDWARE INT. VECTOR TABLE           :
                                        ;---------------------------------------------------
 040D                                   F7:
 040D 1E                                        PUSH    DS              ; SETUP_INT_TABLE:
 040E 2B C0                                     SUB     AX,AX
 0410 8E C0                                     MOV     ES,AX
 0412 B9 0008                                   MOV     CX,08           ; GET VECTOR CNT
 0415 0E                                        PUSH    CS              ; SETUP DS SEG REG
 0416 1F                                        POP     DS
 0417 BE 1EF3 R                                 MOV     SI,OFFSET VECTOR_TABLE
 041A BF 0020 R                                 MOV     DI,OFFSET @INT_PTR
 041D                                   F7A:
 041D A5                                        MOVSW
 041E 47                                        INC     DI              ; SKIP OVER SEGMENT
 041F 47                                        INC     DI
 0420 E2 FB                                     LOOP    F7A
 0422 1F                                        POP     DS

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

 0423 C7 06 0008 R 02C3 R                       MOV     WORD PTR @NMI_PTR,OFFSET NMI_INT ; NMI INTERRUPT
 0429 C7 06 0014 R 1F54 R                       MOV     WORD PTR @INT5_PTR,OFFSET PRINT_SCREEN_1 ; PRINT SCREEN
 042F C7 06 0062 R F600                         MOV     WORD PTR @BASIC_PTR+2,0F600H     ; SEGMENT FOR CASSETTE BASIC
 0435 C7 06 007E R 0000                         MOV     WORD PTR @EXT_PTR+2,0000         ;SEGMENT FOR VIDEO EXTENSION

                                        ;----- SETUP TIMER 0 TO BLINK LED IF MANUFACTURING TEST MODE

 043B 80 3E 0412 R 01                           CMP     DATA_AREA[@MFG_TST-DATA40],01H  ; MFG. TEST MODE?
 0440 75 0A                                     JNZ     EXP_TO
 0442 C7 06 0070 1909 R                         MOV     WORD PTR DS:[1CH*4],OFFSET BLINK_INT  ; SETUP TIMER TO BLINK LED
```

```
0448 B0 FE                      MOV     AL,0FEH                 ; ENABLE TIMER INTERRUPT
044A E6 21                      OUT     INTA01,AL
                        ;-------------------------------------------------------------
                        ; EXPANSION I/O BOX TEST                                      :
                        ;       CHECK TO SEE IF EXPANSION BOX PRESENT - IF INSTALLED, :
                        ;       TEST DATA AND ADDRESS BUSES TO I/O BOX                :
                        ;  ERROR='1801'                                               :
                        ;-------------------------------------------------------------

                        ;----- DETERMINE IF BOX IS PRESENT

044C                    EXP_IO:                                 ; (CARD WAS ENABLED EARLIER)
044C BA 0210                    MOV     DX,0210H                ; CONTROL PORT ADDRESS
044F B8 5555                    MOV     AX,5555H                ; SET DATA PATTERN
0452 EE                         OUT     DX,AL
0453 B0 01                      MOV     AL,01H                  ; MAKE AL DIFFERENT
0455 EC                         IN      AL,DX                   ; RECOVER DATA
0456 3A C4                      CMP     AL,AH                   ; REPLY?
0458 75 43                      JNE     E19                     ; NO RESPONSE, GO TO NEXT TEST
045A F7 D0                      NOT     AX                      ; MAKE DATA=AAAA
045C EE                         OUT     DX,AL
045D B0 01                      MOV     AL,01H
045F EC                         IN      AL,DX                   ; RECOVER DATA
0460 3A C4                      CMP     AL,AH
0462 75 39                      JNE     E19

                        ;----- CHECK ADDRESS BUS

0464                    EXP2:
0464 BB 0001                    MOV     BX,0001H
0467 BA 0215                    MOV     DX,0215H                ; LOAD HI ADDR. REG ADDRESS
046A B9 0010                    MOV     CX,0016                 ; GO ACROSS 16 BITS
046D                    EXP3:
046D 2E: 88 07                  MOV     CS:[BX],AL              ; WRITE ADDRESS F0000+BX
0470 90                         NOP
0471 EC                         IN      AL,DX                   ; READ ADDR. HIGH
0472 3A C7                      CMP     AL,BH
0474 75 21                      JNE     EXP_ERR                 ; GO ERROR IF MISCOMPARE
0476 42                         INC     DX                      ; DX=216H (ADDR. LOW REG)
0477 EC                         IN      AL,DX
0478 3A C3                      CMP     AL,BL                   ; COMPARE TO LOW ADDRESS
047A 75 1B                      JNE     EXP_ERR
047C 4A                         DEC     DX                      ; DX BACK TO 215H
047D D1 E3                      SHL     BX,1
047F E2 EC                      LOOP    EXP3                    ; LOOP TILL '1' WALKS ACROSS BX

                        ;----- CHECK DATA BUS

0481 B9 0008                    MOV     CX,0008                 ; DO 8 TIMES
0484 B0 01                      MOV     AL,01
0486 4A                         DEC     DX                      ; MAKE DX=214H (DATA BUS REG)
0487                    EXP4:
0487 8A E0                      MOV     AH,AL                   ; SAVE DATA BUS VALUE
0489 EE                         OUT     DX,AL                   ; SEND VALUE TO REG
048A B0 01                      MOV     AL,01H
048C EC                         IN      AL,DX                   ; RETRIVE VALUE FROM REG
048D 3A C4                      CMP     AL,AH                   ; = TO SAVED VALUE
048F 75 06                      JNE     SHORT EXP_ERR
0491 D0 E0                      SHL     AL,1                    ; FORM NEW DATA PATTERN
0493 E2 F2                      LOOP    EXP4                    ; LOOP TILL BIT WALKS ACROSS AL
0495 EB 06                      JMP     SHORT E19               ; GO ON TO NEXT TEST
0497                    EXP_ERR:
0497 BE 18DC R                  MOV     SI,OFFSET F3C           ('1801')
049A E8 1976 R                  CALL    E_MSG
                        ;-----------------------------------------------------
                        ;       ADDITIONAL READ/WRITE STORAGE TEST            :
                        ; DESCRIPTION                                         :
                        ;       WRITE/READ DATA PATTERNS TO ANY READ/WRITE    :
                        ;       STORAGE AFTER THE FIRST 64K.  STORAGE         :
                        ;       ADDRESSABILITY IS CHECKED.                    :
                        ;-----------------------------------------------------
                                ASSUME  DS:DATA
049D                    E19:
049D E8 1A12 R                  CALL    DDS
04A0 1E                         PUSH    DS
04A1                    E20:
04A1 81 3E 0072 R 1234          CMP     @RESET_FLAG,1234H       ; WARM START?
04A7 75 03                      JNE     E20A                    ; CONTINUE TEST IF NOT
04A9 E9 054A R                  JMP     ROM_SCAN                ; GO TO NEXT ROUTINE IF SO
04AC                    E20A:
04AC B8 0040                    MOV     AX,64                   ; STARTING AMT. OF MEMORY OK
04AF EB 28                      JMP     SHORT PRT_SIZ           ; POST MESSAGE
04B1                    E20B:
04B1 8B 1E 0013 R               MOV     BX,@MEMORY_SIZE         ; GET MEM. SIZE WORD
04B5 83 EB 40                   SUB     BX,64                   ; 1ST 64K ALREADY DONE
04B8 B1 04                      MOV     CL,4
04BA D3 EB                      SHR     BX,CL                   ; DIVIDE BY 16
04BC 8B CB                      MOV     CX,BX                   ; SAVE COUNT OF 16K BLOCKS
04BE BB 1000                    MOV     BX,1000H                ; SET PTR. TO RAM SEGMENT>64K
04C1                    E21:
04C1 8E DB                      MOV     DS,BX                   ; SET SEG. REG
04C3 8E C3                      MOV     ES,BX
04C5 81 C3 0400                 ADD     BX,0400H                ; POINT TO NEXT 16K
04C9 52                         PUSH    DX
04CA 51                         PUSH    CX                      ; SAVE WORK REGS
04CB 53                         PUSH    BX
04CC 50                         PUSH    AX
04CD B9 2000                    MOV     CX,02000H               ; SET COUNT FOR 8K WORDS
04D0 E8 0CCF R                  CALL    STGTST_CNT
04D3 75 4C                      JNZ     E21A                    ; GO PRINT ERROR
04D5 58                         POP     AX                      ; RECOVER TESTED MEM NUMBER
04D6 05 0010                    ADD     AX,16
04D9                    PRT_SIZ:
04D9 50                         PUSH    AX
04DA BB 000A                    MOV     BX,10                   ; SET UP FOR DECIMAL CONVERT
04DD B9 0003                    MOV     CX,3                    ; OF 3 NIBBLES
04E0                    DECIMAL_LOOP:
04E0 33 D2                      XOR     DX,DX
04E2 F7 F3                      DIV     BX                      ; DIVIDE BY 10
04E4 80 CA 30                   OR      DL,30H                  ; MAKE INTO ASCII
04E7 52                         PUSH    DX                      ; SAVE
04E8 E2 F6                      LOOP    DECIMAL_LOOP
04EA B9 0003                    MOV     CX,3
04ED                    PRT_DEC_LOOP:
```

SECTION 5

```
04ED 58                         POP     AX                      ; RECOVER A NUMBER
04EE E8 1969 R                  CALL    PRT_HEX
04F1 E2 FA                      LOOP    PRT_DEC_LOOP
04F3 B9 0007                    MOV     CX,7
04F6 BE 001A R                  MOV     SI,OFFSET F3B           ; PRINT ' KB OK'
04F9                    KB_LOOP:
04F9 2E: 8A 04                  MOV     AL,CS:[SI]
04FC 46                         INC     SI
04FD E8 1969 R                  CALL    PRT_HEX
0500 E2 F7                      LOOP    KB_LOOP
0502 58                         POP     AX                      ; RECOVER WORK REGS
0503 3D 0040                    CMP     AX,64                   ; FIRST PASS?
0506 74 A9                      JE      E20B
0508 5B                         POP     BX
0509 59                         POP     CX
050A 5A                         POP     DX
050B E2 B4                      LOOP    E21                     ; LOOP TILL ALL MEM. CHECKED
050D B0 0A                      MOV     AL,10
050F E8 1969 R                  CALL    PRT_HEX                 ; LINE FEED

                        ;----- DMA TC0 SHOULD BE ON BY NOW - SEE IF IT IS

0512 E4 08                      IN      AL,DMA+08H
0514 24 01                      AND     AL,00000001B            ; TC0 STATUS BIT ON?
0516 75 32                      JNZ     ROM_SCAN                ; GO ON WITH NEXT TEST IF OK
0518 1F                         POP     DS
0519 C6 06 0015 R 03            MOV     @MFG_ERR_FLAG,03H       ; <><><><><><><><><><><><><><>
051E E9 0385 R                  JMP     D6                      ; POST 101 ERROR MSG AND HALT

                        ;----- PRINT FAILING ADDRESS AND XOR'ED PATTERN IF DATA COMPARE ERROR

0521 8A E8              E21A:   MOV     CH,AL                   ; SAVE FAILING BIT PATTERN
0523 B0 0D                      MOV     AL,CR                   ; CARRAGE RETURN
0525 E8 1969 R                  CALL    PRT_HEX
0528 B0 0A                      MOV     AL,LF                   ; LINE FEED
052A E8 1969 R                  CALL    PRT_HEX
052D 58                         POP     AX                      ; RECOVER AMT. OF GOOD MEM.
052E 83 C4 06                   ADD     SP,6                    ; BALANCE STACK
0531 8C DA                      MOV     DX,DS                   ; GET FAILING SEGMENT
0533 1F                         POP     DS
0534 1E                         PUSH    DS
0535 A3 0013 R                  MOV     @MEMORY_SIZE,AX         ; LOAD MEM. SIZE WORD TO SHOW
                                                                ; HOW MUCH MEM. WORKING
0538 88 36 0015 R               MOV     @MFG_ERR_FLAG,DH        ; <><><><><><><><><><><><><><>
                                                                ; <><>CHECKPOINTS 08->A0<><>
053C E8 0CBA R                  CALL    PRT_SEG                 ; PRINT IT
053F 8A C5                      MOV     AL,CH                   ; GET FAILING BIT PATTERN
0541 E8 1958 R                  CALL    XPC_BYTE                ; CONVERT AND PRINT CODE
0544 BE 18D1 R                  MOV     SI,OFFSET E1            ; SETUP ADDRESS OF ERROR MSG    (201)
0547 E8 1976 R                  CALL    E_MSG                   ; PRINT ERROR MSG
                        ;-----------------------------------------------------------------
                        ; CHECK FOR OPTIONAL ROM FROM C8000->F0000 IN 2K BLOCKS           :
                        ;       (A VALID MODULE HAS '55AA' IN THE FIRST 2 LOCATIONS,      :
                        ;       LENGTH INDICATOR (LENGTH/512) IN THE 3D LOCATION AND      :
                        ;       TEST/INIT. CODE STARTING IN THE 4TH LOCATION.)            :
                        ;-----------------------------------------------------------------
054A                    ROM_SCAN:
054A BA C800                    MOV     DX,0C800H               ; SET BEGINNING ADDRESS
054D                    ROM_SCAN_1:
054D 8E DA                      MOV     DS,DX
054F 2B DB                      SUB     BX,BX                   ; SET BX=0000
0551 8B 07                      MOV     AX,[BX]                 ; GET 1ST WORD FROM MODULE
0553 53                         PUSH    BX
0554 5B                         POP     BX                      ; BUS SETTLING
0555 3D AA55                    CMP     AX,0AA55H               ; = TO ID WORD?
0558 75 06                      JNZ     NEXT_ROM                ; PROCEED TO NEXT ROM IF NOT
055A E8 1920 R                  CALL    ROM_CHECK               ; GO CHECK OUT MODULE
055D EB 05 90                   JMP     ARE_WE_DONE             ; CHECK FOR END OF ROM SPACE
0560                    NEXT_ROM:
0560 81 C2 0080                 ADD     DX,0080H                ; POINT TO NEXT 2K ADDRESS
0564                    ARE_WE_DONE:
0564 81 FA F000                 CMP     DX,0F000H               ; AT F0000 YET?
0568 7C E3                      JL      ROM_SCAN_1              ; GO CHECK ANOTHER ADD. IF NOT
                        ;-----------------------------------------------------------------
                        ;       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.                                           :
                        ;-----------------------------------------------------------------
056A                    F9:
056A 1F                         POP     DS
056B A0 0010 R                  MOV     AL,BYTE PTR @EQUIP_FLAG ; DISKETTE PRESENT?
056E 24 01                      AND     AL,01H                  ; NO - BYPASS DISKETTE TEST
0570 74 5E                      JZ      F15
0572                    F10:                                    ; DISK_TEST:
0572 BA 03F1                    MOV     DX,3F1H                 ; I.D. PORT
0575 EC                         IN      AL,DX
0576 90                         NOP
0577 BB FFFF                    MOV     BX,0FFFFH               ; BUS PRECHARGE
057A 24 F8                      AND     AL,0F8H                 ; KEEP I.D. BITS
057C 80 26 008F R FE            AND     @HF_CNTRL,11111110B     ; RESET DUAL BIT
0581 3C 50                      CMP     AL,CARD_ID
0583 75 05                      JNE     NO_ID
0585 80 0E 008F R 01            OR      @HF_CNTRL,1             ; SET DUAL BIT
058A                    NO_ID:
058A E4 21                      IN      AL,INTA01
058C 24 BF                      AND     AL,0BFH                 ; ENABLE DISKETTE INTERRUPTS
058E E6 21                      OUT     INTA01,AL
0590 B4 00                      MOV     AH,0                    ; RESET NEC FDC
0592 8A D4                      MOV     DL,AH                   ; SET FOR DRIVE 0
0594 CD 13                      INT     13H                     ; VERIFY STATUS AFTER RESET
0596 F6 C4 FF                   TEST    AH,0FFH                 ; STATUS OK?
0599 75 19                      JNZ     F13                     ; NO - FDC FAILED

                        ;----- TURN DRIVE 0 MOTOR ON

059B BA 03F2                    MOV     DX,03F2H                ; GET ADDR OF FDC CARD
059E B0 1C                      MOV     AL,1CH                  ; TURN MOTOR ON, EN DMA/INT
05A0 EE                         OUT     DX,AL                   ; WRITE FDC CONTROL REG
05A1 2B C9                      SUB     CX,CX
05A3                    F11:                                    ; MOTOR_WAIT:
```

```
05A3 E2 FE                          LOOP   F11                       ; WAIT FOR 1 SECOND
05A5                        F12:                                     ; MOTOR_WAIT1:
05A5 E2 FE                          LOOP   F12
05A7 33 D2                          XOR    DX,DX                     ; SELECT DRIVE 0
05A9 B5 22                          MOV    CH,34                     ; SELECT TRACK 34
05AB 88 16 003E R                   MOV    @SEEK_STATUS,DL
05AF E8 0000 E                      CALL   SEEK                      ; RECALIBRATE DISKETTE AND SEEK TO 34
05B2 73 05                          JNC    F14                       ; OK--> GO TURN OF MOTOR
05B4                        F13:                                     ; DISKETTE ERROR
05B4 BE 0990 R                      MOV    SI,OFFSET F3              ; GET ADDR OF MSG
05B7 EB 02                          JMP    SHORT F14A                ; DISPLAY MESSAGE AFTER DISKETTE SETUP

                            ;----- TURN DRIVE 0 MOTOR OFF

05B9                        F14:                                     ; SEQUENCE END ENTRY IF NO ERROR
05B9 33 F6                          XOR    SI,SI                     ; ZERO SI IF NO ERROR
05BB                        F14A:                                    ; SEQUENCE END ENTRY IF ERROR
05BB B0 0C                          MOV    AL,0CH                    ; TURN DRIVE 0 MOTOR OFF
05BD BA 03F2                        MOV    DX,03F2H                  ; FDC CTL ADDRESS
05C0 EE                             OUT    DX,AL

                            ;-----SETUP DISKETTE STATES

05C1 E8 0000 E                      CALL DSKETTE_SETUP               ; INITIALIZE DISKETTE PARMS
05C4 72 04                          JC     F14B                      ; CY-->DISKETTE SETUP ERROR
05C6 0B F6                          OR     SI,SI                     ; PREVIOUS DISKETTE ERROR
05C8 74 06                          JZ     F15                       ; NZ-->DISKETTE ERROR BEFORE SETUP
05CA                        F14B:
05CA BE 0990 R                      MOV    SI,OFFSET F3              ; GET ADDR OF MSG
05CD E8 1976 R                      CALL   E_MSG                     ; GO PRINT ERROR MSG

                            ;----- SETUP PRINTER AND RS232 BASE ADDRESSES IF DEVICE ATTACHED

05D0                        F15:
05D0 C6 06 006B R 00                MOV    @INTR_FLAG,00H            ; SET STRAY INTERRUPT FLAG = 00
05D5 BE 001E R                      MOV    SI,OFFSET @KB_BUFFER      ; SETUP KEYBOARD PARAMETERS
05D8 89 36 001A R                   MOV    @BUFFER_HEAD,SI
05DC 89 36 001C R                   MOV    @BUFFER_TAIL,SI
05E0 89 36 0080 R                   MOV    @BUFFER_START,SI
05E4 83 C6 20                       ADD    SI,32                     ;DEFAULT BUFFER OF 32 BYTES
05E7 89 36 0082 R                   MOV    @BUFFER_END,SI
05EB BF 0078 R                      MOV    DI,OFFSET @PRINT_TIM_OUT  ;SET DEFAULT PRINTER TIMEOUT
05EE 1E                             PUSH   DS
05EF 07                             POP    ES
05F0 B8 1414                        MOV    AX,1414H                  ; DEFAULT=20
05F3 AB                             STOSW
05F4 AB                             STOSW
05F5 B8 0101                        MOV    AX,0101H                  ;RS232 DEFAULT=01
05F8 AB                             STOSW
05F9 AB                             STOSW
05FA E4 21                          IN     AL,INTA01
05FC 24 FC                          AND    AL,0FCH                   ; ENABLE TIMER AND KB INTS
05FE E6 21                          OUT    INTA01,AL

0600 83 FD 00                       CMP    BP,0000                   ; CHECK FOR BP= NON ZERO
                                                                     ; (ERROR HAPPENED)
0603 74 18                          JE     F15A_0                    ; CONTINUE IF NO ERROR
0605 BA 0002                        MOV    DX,2                      ; 2 SHORT BEEPS (ERROR)
0608 E8 19A5 R                      CALL   ERR_BEEP
060B BE 0769 R                      MOV    SI,OFFSET F3D             ; LOAD ERROR MSG
060E E8 1997 R                      CALL   P_MSG
0611                        ERR_WAIT:
0611 B4 00                          MOV    AH,00
0613 CD 16                          INT    16H                       ; WAIT FOR 'F1' KEY
0615 80 FC 3B                       CMP    AH,3BH
0618 75 F7                          JNE    ERR_WAIT
061A EB 0E 90                       JMP    F15A                      ; BYPASS ERROR
061D                        F15A_0:
061D 80 3E 0012 R 01                CMP    @MFG_TST,1                ; MFG MODE
0622 74 06                          JE     F15A                      ; BYPASS BEEP
0624 BA 0001                        MOV    DX,1                      ; 1 SHORT BEEP (NO ERRORS)
0627 E8 19A5 R                      CALL   ERR_BEEP
062A A0 0010 R              F15A:   MOV    AL,BYTE PTR @EQUIP_FLAG   ; GET SWITCHES
062D 24 01                          AND    AL,00000001B              ; 'LOOP POST' SWITCH ON
062F 75 03                          JNZ    F15B                      ; CONTINUE WITH BRING-UP
0631 E9 005B R                      JMP    START
0634 2A E4                  F15B:   SUB    AH,AH
0636 A0 0049 R                      MOV    AL,@CRT_MODE
0639 CD 10                          INT    10H                       ; CLEAR SCREEN
063B                        F15C:
063B BD 1970 R                      MOV    BP,OFFSET F4              ; PRT_SRC_TBL
063E BE 0000                        MOV    SI,0
0641                        F16:                                     ; PRT_BASE:
0641 2E: 8B 56 00                   MOV    DX,CS:[BP]                ; GET PRINTER BASE ADDR
0645 B0 AA                          MOV    AL,0AAH                   ; WRITE DATA TO PORT A
0647 EE                             OUT    DX,AL
0648 1E                             PUSH   DS                        ; BUS SETTLEING
0649 EC                             IN     AL,DX                     ; READ PORT A
064A 1F                             POP    DS
064B 3C AA                          CMP    AL,0AAH                   ; DATA PATTERN SAME
064D 75 05                          JNE    F17                       ; NO - CHECK NEXT PRT CD
064F 89 54 08                       MOV    [@PRINTER_BASE-DATA40][SI],DX    ; YES - STORE PRT BASE ADDR
0652 46                             INC    SI                        ; INCREMENT TO NEXT WORD
0653 46                             INC    SI
0654                        F17:
0654 45                             INC    BP                        ; POINT TO NEXT BASE ADDR
0655 45                             INC    BP
0656 81 FD 1976 R                   CMP    BP,OFFSET F4E             ; ALL POSSIBLE ADDRS CHECKED?
065A 75 E5                          JNE    F16                       ; PRT_BASE
065C BB 0000                        MOV    BX,0                      ; POINTER TO RS232 TABLE
065F BA 03FA                        MOV    DX,3FAH                   ; CHECK IF RS232 CD 1 ATTCH?
0662 EC                             IN     AL,DX                     ; READ INTR ID REG
0663 A8 F8                          TEST   AL,0F8H
0665 75 06                          JNZ    F18
0667 C7 07 03F8                     MOV    [@RS232_BASE-DATA40][BX],3F8H    ; SETUP RS232 CD #1 ADDR
066B 43                             INC    BX
066C 43                             INC    BX
066D                        F18:
066D BA 02FA                        MOV    DX,2FAH                   ; CHECK IF RS232 CD 2 ATTCH
0670 EC                             IN     AL,DX                     ; READ INTERRUPT ID REG
0671 A8 F8                          TEST   AL,0F8H
0673 75 06                          JNZ    F19                       ; BASE_END
0675 C7 07 02F8                     MOV    [@RS232_BASE-DATA40][BX],2F8H    ; SETUP RS232 CD #2
0679 43                             INC    BX
```

SECTION 5

```
067A 43                         INC     BX
                        ;----- SET UP EQUIP FLAG TO INDICATE NUMBER OF PRINTERS AND RS232 CARDS
067B                    F19:                                    ; BASE_END:
067B 8B C6                      MOV     AX,SI                   ; SI HAS 2* NUMBER OF RS232
067D B1 03                      MOV     CL,3                    ; SHIFT COUNT
067F D2 C8                      ROR     AL,CL                   ; ROTATE RIGHT 3 POSITIONS
0681 0A C3                      OR      AL,BL                   ; OR IN THE PRINTER COUNT
0683 A2 0011 R                  MOV     BYTE PTR @EQUIP_FLAG+1,AL       ; STORE AS SECOND BYTE
0686 BA 0201                    MOV     DX,201H
0689 EC                         IN      AL,DX
068A 90                         NOP
068B 90                         NOP
068C 90                         NOP
068D A8 0F                      TEST    AL,0FH
068F 75 05                      JNZ     F20                     ; NO_GAME_CARD
0691 80 0E 0011 R 10            OR      BYTE PTR @EQUIP_FLAG+1,16
0696                    F20:                                    ; NO_GAME_CARD:
                        ;----- ENABLE NMI INTERRUPTS
0696 E4 61                      IN      AL,PORT_B               ; RESET CHECK ENABLES
0698 0C 30                      OR      AL,30H
069A E6 61                      OUT     PORT_B,AL
069C 24 CF                      AND     AL,0CFH
069E E6 61                      OUT     PORT_B,AL
06A0 B0 80                      MOV     AL,80H                  ; ENABLE NMI INTERRUPTS
06A2 E6 A0                      OUT     0A0H,AL
06A4                    F21:                                    ; LOAD_BOOT_STRAP:
06A4 CD 19                      INT     19H                     ; GO TO THE BOOT LOADER
                        ;--- INT 19 -------------------------------------------------
                        ; BOOT STRAP LOADER                                         :
                        ;       TRACK 0, SECTOR 1 IS READ INTO THE                  :
                        ;       BOOT LOCATION (SEGMENT 0, OFFSET 7C00)              :
                        ;       AND CONTROL IS TRANSFERRED THERE.                   :
                        ;                                                           :
                        ;       IF THERE IS A HARDWARE ERROR CONTROL IS             :
                        ;       TRANSFERRED TO THE ROM BASIC ENTRY POINT.           :
                        ;------------------------------------------------------------
                                ASSUME  CS:CODE,DS:ABS0
                        ;       ORG     0E6F2H
06F2                            ORG     006F2H
06F2                    BOOT_STRAP      PROC    NEAR
06F2 FB                         STI                             ; ENABLE INTERRUPTS
06F3 2B C0                      SUB     AX,AX                   ; ESTABLISH ADDRESSING
06F5 8E D8                      MOV     DS,AX
                        ;----- RESET THE DISK PARAMETER TABLE VECTOR
06F7 C7 06 0078 R 0FC7 R        MOV     WORD PTR @DISK_POINTER,OFFSET DISK_BASE
06FD 8C 0E 007A R               MOV     WORD PTR @DISK_POINTER+2,CS
                        ; -----  LOAD SYSTEM FROM DISKETTE -- CX HAS RETRY COUNT
0701 B9 0004                    MOV     CX,4                    ; SET RETRY COUNT
0704                    H1:                                     ; IPL_SYSTEM
0704 51                         PUSH    CX                      ; SAVE RETRY COUNT
0705 B4 00                      MOV     AH,0                    ; RESET THE DISKETTE SYSTEM
0707 CD 13                      INT     13H                     ; DISKETTE_IO
0709 72 0F                      JC      H2                      ; IF ERROR, TRY AGAIN
070B B8 0201                    MOV     AX,201H                 ; READ IN THE SINGLE SECTOR
070E 2B D2                      SUB     DX,DX                   ; TO THE BOOT LOCATION
0710 8E C2                      MOV     ES,DX
0712 BB 7C00 R                  MOV     BX,OFFSET @BOOT_LOCN
                                                                ; DRIVE 0, HEAD 0
0715 B9 0001                    MOV     CX,1                    ; SECTOR 1, TRACK 0
0718 CD 13                      INT     13H                     ; DISKETTE_IO
071A                    H2:
071A 59                         POP     CX                      ; RECOVER RETRY COUNT
071B 73 04                      JNC     H4                      ; CF SET BY UNSUCCESSFUL READ
071D E2 E5                      LOOP    H1                      ; DO IT FOR RETRY TIMES
                        ;----- UNABLE TO IPL FROM THE DISKETTE
071F                    H3:
071F CD 18                      INT     18H                     ; GO TO RESIDENT BASIC
                        ;----- IPL WAS SUCCESSFUL
0721                    H4:
0721 EA 7C00 ---- R             JMP     @BOOT_LOCN
0726                    BOOT_STRAP      ENDP
                        ;;-     ORG     0E729H
0729                            ORG     00729H
0729 0417               A1      DW      1047                    ; 110 BAUD      ; TABLE OF VALUES
072B 0300                       DW      768                     ; 150           ; FOR INITIALIZATION
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
0739                    RS232_IO:
0739 E9 0000 E                  JMP     RS232_IO_1
                                                                ;       USE INT 15 H  AH= 0C0H
073C                    CONF_TBL:                               ; CONFIGURATION TABLE FOR THIS SYSTEM
073C 0008                       DW      CONF_E-CONF_TBL-2       ; LENGTH OF FOLLOWING TABLE
073E FB                         DB      MODEL_BYTE              ; SYSTEM MODEL BYTE
073F 00                         DB      SUB_MODEL_BYTE          ; SYSTEM SUB MODEL TYPE BYTE
0740 01                         DB      BIOS_LEVEL              ; BIOS REVISION LEVEL
0741 50                         DB      01010000B               ; 10000000 = DMA CHANNEL 3 USE BY BIOS
                                                                ; 01000000 = CASCADED INTERRUPT LEVEL 2
                                                                ; 00100000 = REAL TIME CLOCK AVAILABLE
                                                                ; 00010000 = KEYBOARD SCAN CODE HOOK 1AH
0742 00                         DB      0                       ; RESERVED
0743 00                         DB      0                       ; RESERVED
0744 00                         DB      0                       ; RESERVED
0745 00                         DB      0                       ; RESERVED
= 0746                  CONF_E  EQU     $                       ; RESERVED FOR EXPANSION
```

```

                                   ;--------------------------------------------------------------------------
                                   ;  PRINT ADDRESS AND ERROR MESSAGE FOR ROM CHECKSUM ERRORS                :
                                   ;--------------------------------------------------------------------------
0746                               ROM_ERR PROC     NEAR
0746 52                                    PUSH     DX                      ; SAVE POINTER
0747 50                                    PUSH     AX
0748 8C DA                                 MOV      DX,DS                   ; GET ADDRESS POINTER
074A 26: 88 36 0015 R                      MOV      ES:@MFG_ERR_FLAG,DH     ; <><><><><><><><><><><><><><><><>
                                                                            ; <><><>CHECKPOINTS C0->F4<><><>
074F 81 FA C800                            CMP      DX,0C800H               ; CRT CARD IN ERROR?
0753 7C 0C                                 JL       ROM_ERR_BEEP            ; GIVE CRT CARD FAIL BEEP
0755 E8 0CBA R                             CALL     PRT_SEG                 ; PRINT SEGEMENT IN ERROR
0758 BE 18D7 R                             MOV      SI,OFFSET F3A           ; DISPLAY ERROR MSG
075B E8 1976 R                             CALL     E_MSG
075E                               ROM_ERR_END:
075E 58                                    POP      AX
075F 5A                                    POP      DX
0760 C3                                    RET
0761                               ROM_ERR_BEEP:
0761 BA 0102                               MOV      DX,0102H                ; BEEP 1 LONG, 2 SHORT
0764 E8 19A5 R                             CALL     ERR_BEEP
0767 EB F5                                 JMP      SHORT ROM_ERR_END
0769                               ROM_ERR ENDP

0769 45 52 52 4F 52 2E             F3D     DB       'ERROR. (RESUME = "F1" KEY)',CR,LF      ; ERROR PROMPT
     20 28 52 45 53 55
     4D 45 20 3D 20 22
     46 31 22 20 4B 45
     59 29 0D 0A

                                   ;       ORG      0E82EH
082E                                       ORG      0082EH
082E                               KEYBOARD_IO:
082E E9 0000 E                             JMP      KEYBOARD_IO_1

                                   ;       ORG      0E987H
0987                                       ORG      00987H
0987                               KB_INT:
0987 E9 0000 E                             JMP      KB_INT_1

098A 20 33 30 31 0D 0A             F1      DB       ' 301',CR,LF            ; KEYBOARD ERROR
0990 36 30 31 0D 0A                F3      DB       '601',CR,LF             ; DISKETTE ERROR

                                   ;--- INT  1A H -- SYSTEM AND REAL TIME CLOCK SERVICES -------------------------
                                   ;                                                                             :
                                   ;       THIS BIOS ROUTINE ALLOWS THE CLOCKS TO BE SET OR READ                 :
                                   ;                                                                             :
                                   ; PARAMETERS:                                                                 :
                                   ;    (AH) = 00H  READ THE CURRENT CLOCK SETTING AND RETURN WITH,              :
                                   ;                   (CX) = HIGH PORTION OF COUNT                              :
                                   ;                   (DX) = LOW PORTION OF COUNT                               :
                                   ;                   (AL) = 0 TIMER HAS NOT PASSED 24 HOURS SINCE LAST READ,   :
                                   ;                          1 IF ON ANOTHER DAY. (RESET TO ZERO AFTER READ)    :
                                   ;                                                                             :
                                   ;    (AH) = 01H  SET THE CURRENT CLOCK USING,                                 :
                                   ;                   (CX) = HIGH PORTION OF COUNT                              :
                                   ;                   (DX) = LOW PORTION OF COUNT.                              :
                                   ;                                                                             :
                                   ;            NOTE: COUNTS OCCUR AT THE RATE OF 1193180/65536 COUNTS/SECOND    :
                                   ;                  (OR ABOUT 18.2 PER SECOND -- SEE EQUATES)                  :
                                   ;                                                                             :
                                   ;    (AH) = 0AH  READ THE CURRENT COUNT OF DAYS AND RETURN WITH,              :
                                   ;                   (CX) = COUNT OF ELAPSED DAYS                              :
                                   ;                                                                             :
                                   ;    (AH) = 0BH  SET THE CURRENT COUNT OF DAYS USING,                         :
                                   ;                   (CX) = COUNT OF ELAPSED DAYS                              :
                                   ;                                                                             :
                                   ; NOTES: FOR ALL RETURNS CY= 0 FOR SUCCESSFUL OPERATION.                      :
                                   ;        INTERRUPTS ARE DISABLED DURING DATA MODIFICATION.                    :
                                   ;        AH & AL ARE RETURNED MODIFIED AND NOT DEFINED EXCEPT WHERE INDICATED. :
                                   ;-----------------------------------------------------------------------------
                                           ASSUME   CS:CODE,DS:DATA

0995                               TIME_OF_DAY_1    PROC    FAR
0995                               TIME_OF_DAY_11:
0995 FB                                    STI                              ; INTERRUPTS BACK ON
0996 80 FC 0C                              CMP      AH,(RTC_TBE-RTC_TB)/2   ; CHECK IF COMMAND IN VALID RANGE
0999 F5                                    CMC                              ; COMPLEMENT CARRY FOR ERROR EXIT
099A 72 17                                 JC       TIME_9                  ; EXIT WITH CARRY = 1 IF NOT VALID

099C 1E                                    PUSH     DS                      ; SAVE USERS (DS) SEGMENT
099D E8 1A12 R                             CALL     DDS                     ; GET DATA SEGMENT SELECTOR
09A0 56                                    PUSH     SI                      ; SAVE WORK REGISTER
09A1 8A C4                                 MOV      AL,AH                   ; MOVE FUNCTION TO (AL) REGISTER
09A3 98                                    CBW                              ; CONVERT FUNCTION TO BYTE OFFSET
09A4 03 C0                                 ADD      AX,AX                   ; CONVERT FUNCTION TO WORD OFFSET (CY=0)
09A6 8B F0                                 MOV      SI,AX                   ; PLACE INTO ADDRESSING REGISTER
09A8 FA                                    CLI                              ; NO INTERRUPTS DURING TIME FUNCTIONS
09A9 2E: FF 94 09B6 R                      CALL     CS:[SI]+OFFSET RTC_TB   ; VECTOR TO FUNCTION REQUESTED WITH CY=0
                                                                            ;  RETURN WITH CARRY FLAG SET FOR RESULT
09AE FB                                    STI                              ; INTERRUPTS BACK ON
09AF B4 00                                 MOV      AH,0                    ; CLEAR (AH) TO ZERO
09B1 5E                                    POP      SI                      ; RECOVER USERS REGISTER
09B2 1F                                    POP      DS                      ; RECOVER USERS SEGMENT SELECTOR
09B3                               TIME_9:                                  ; RETURN WITH CY= 0 IF NO ERROR
09B3 CA 0002                               RET      2
                                                                            ;       ROUTINE VECTOR TABLE (AH)=
09B6 09CE R                        RTC_TB  DW       RTC_00                  ; 00H = READ CURRENT CLOCK COUNT
09B8 09DF R                                DW       RTC_10                  ; 01H = SET CLOCK COUNT
09BA 09ED R                                DW       RTC_NS                  ; 02H     INVALID
09BC 09ED R                                DW       RTC_NS                  ; 03H     INVALID
09BE 09ED R                                DW       RTC_NS                  ; 04H     INVALID
09C0 09ED R                                DW       RTC_NS                  ; 05H     INVALID
09C2 09ED R                                DW       RTC_NS                  ; 06H     INVALID
09C4 09ED R                                DW       RTC_NS                  ; 07H     INVALID
09C6 09ED R                                DW       RTC_NS                  ; 08H     INVALID
09C8 09ED R                                DW       RTC_NS                  ; 09H     INVALID
09CA 09EF R                                DW       RTC_A0                  ; 0AH = READ SYSTEM DAY COUNTER
09CC 09F4 R                                DW       RTC_B0                  ; 0BH = WRITE SYSTEM DAY COUNTER
= 09CE                             RTC_TBE EQU      $

09CE                               TIME_OF_DAY_1    ENDP

```

SECTION 5

```
09CE                RTC_00   PROC   NEAR              ;       READ TIME COUNT
09CE A0 0070 R               MOV    AL,@TIMER_OFL     ; GET THE OVERFLOW FLAG
09D1 C6 06 0070 R 00         MOV    @TIMER_OFL,0      ; AND THEN RESET THE OVERFLOW FLAG
09D6 8B 0E 006E R            MOV    CX,@TIMER_HIGH    ; GET COUNT OF TIME HIGH WORD
09DA 8B 16 006C R            MOV    DX,@TIMER_LOW     ; GET COUNT OF TIME LOW WORD
09DE C3                      RET                      ; RETURN WITH NO CARRY

09DF                RTC_10:                           ;       SET TIME COUNT
09DF 89 16 006C R            MOV    @TIMER_LOW,DX     ; SET TIME COUNT LOW WORD
09E3 89 0E 006E R            MOV    @TIMER_HIGH,CX    ; SET THE TIME COUNT HIGH WORD
09E7 C6 06 0070 R 00         MOV    @TIMER_OFL,0      ; RESET OVERFLOW FLAG
09EC C3                      RET                      ; RETURN WITH NO CARRY

09ED                RTC_NS:                           ;      INVALID FUNCTION (NOT SUPPORTED)
09ED F9                      STC                      ; SET CARRY FLAG FOR ERROR (CY=1)
09EE C3                      RET                      ; EXIT THROUGH COMMON RETURN

09EF                RTC_A0:                           ;       READ SYSTEM DAY COUNTER
09EF 8B 0E 00CE R            MOV    CX,@DAY_COUNT     ; GET COUNT OF DAYS
09F3 C3                      RET                      ; EXIT THROUGH COMMON RETURN WITH CY=0

09F4                RTC_B0:                           ;       SET SYSTEM DAY COUNTER
09F4 89 0E 00CE R            MOV    @DAY_COUNT,CX     ; SET COUNT OF DAYS
09F8 C3                      RET                      ; EXIT THROUGH COMMON RETURN WITH CY=0

09F9                RTC_00   ENDP

                    ;        ORG    0EC59H
0C59                         ORG    00C59H
0C59 E9 0000 E      DISKETTE_IO:    JMP     DISKETTE_IO_1

                    ;--- BEEP -----------------------------------------------------------
                    ;        ROUTINE TO SOUND THE BEEPER USING TIMER 2 FOR TONE            :
                    ; ENTRY:                                                             :
                    ;        (BL) = DURATION COUNTER  ( 1 FOR 1/64 SECOND )                :
                    ;        (CX) = FREQUENCY DIVISOR (1193180/FREQUENCY) (1331 FOR 886 HZ) :
                    ; EXIT:                                                              :
                    ;        (AX),(BL),(CX) MODIFIED.                                      :
                    ;--------------------------------------------------------------------

0C5C                BEEP     PROC   NEAR              ;       SETUP TIMER 2
0C5C 9C                      PUSHF                    ; SAVE INTERRUPT STATUS
0C5D FA                      CLI                      ; BLOCK INTERRUPTS DURING UPDATE
0C5E B0 B6                   MOV    AL,10110110B      ; SELECT TIMER 2,LSB,MSB,BINARY
0C60 E6 43                   OUT    TIMER+3,AL        ; WRITE THE TIMER MODE REGISTER
0C62 90                      NOP                      ; I/O DELAY
0C63 8A C1                   MOV    AL,CL             ; DIVISOR FOR HZ (LOW)
0C65 E6 42                   OUT    TIMER+2,AL        ; WRITE TIMER 2 COUNT - LSB
0C67 90                      NOP                      ; I/O DELAY
0C68 8A C5                   MOV    AL,CH             ; DIVISOR FOR HZ (HIGH)
0C6A E6 42                   OUT    TIMER+2,AL        ; WRITE TIMER 2 COUNT - MSB
0C6C E4 61                   IN     AL,PORT_B         ; GET CURRENT SETTING OF PORT
0C6E 8A E0                   MOV    AH,AL             ; SAVE THAT SETTING
0C70 0C 03                   OR     AL,GATE2+SPK2     ; GATE TIMER 2 AND TURN SPEAKER ON
0C72 E6 61                   OUT    PORT_B,AL         ; AND RESTORE INTERRUPT STATUS
0C74 9D                      POPF
0C75                G7:                               ;      1/64 SECOND PER COUNT (BL)
0C75 B9 040B                 MOV    CX,1035           ; DELAY COUNT FOR 1/64 OF A SECOND
0C78 E8 0CA0 R               CALL   WAITF             ; GO TO BEEP DELAY 1/64 COUNT
0C7B FE CB                   DEC    BL                ; (BL) LENGTH COUNT EXPIRED?
0C7D 75 F6                   JNZ    G7                ; NO - CONTINUE BEEPING SPEAKER

0C7F 9C                      PUSHF                    ; SAVE INTERRUPT STATUS
0C80 FA                      CLI                      ; BLOCK INTERRUPTS DURING UPDATE
0C81 E4 61                   IN     AL,PORT_B         ; GET CURRENT PORT VALUE
0C83 0C FC                   OR     AL,NOT (GATE2+SPK2) ; ISOLATE CURRENT SPEAKER BITS IN CASE
0C85 22 E0                   AND    AH,AL             ;  SOMEONE TURNED THEM OFF DURING BEEP
0C87 8A C4                   MOV    AL,AH             ; RECOVER VALUE OF PORT
0C89 24 FC                   AND    AL,NOT (GATE2+SPK2) ; FORCE SPEAKER DATA OFF
0C8B E6 61                   OUT    PORT_B,AL         ; AND STOP SPEAKER TIMER
0C8D 9D                      POPF                     ; RESTORE INTERRUPT FLAG STATE
0C8E B9 040B                 MOV    CX,1035           ; FORCE 1/64 SECOND DELAY (SHORT)
0C91 E8 0CA0 R               CALL   WAITF             ; MINIMUM DELAY BETWEEN ALL BEEPS
0C94 9C                      PUSHF                    ; SAVE INTERRUPT STATUS
0C95 FA                      CLI                      ; BLOCK INTERRUPTS DURING UPDATE
0C96 E4 61                   IN     AL,PORT_B         ; GET CURRENT PORT VALUE IN CASE
0C98 24 03                   AND    AL,GATE2+SPK2     ;  SOMEONE TURNED THEM ON
0C9A 0A C4                   OR     AL,AH             ; RECOVER VALUE OF PORT_B
0C9C E6 61                   OUT    PORT_B,AL         ; RESTORE SPEAKER STATUS
0C9E 9D                      POPF                     ; RESTORE INTERRUPT FLAG STATE
0C9F C3                      RET

0CA0                BEEP     ENDP

                    ;--- WAITF ----------------------------------------------------------
                    ;        FIXED TIME WAIT ROUTINE (HARDWARE CONTROLLED - NOT PROCESSOR)  :
                    ;                                                                     :
                    ; ENTRY:                                                              :
                    ;        (CX) = COUNT OF 15.085737 MICROSECOND INTERVALS TO WAIT        :
                    ;               MEMORY REFRESH TIMER 1 OUTPUT AT THE DMA CHANNEL 0     :
                    ;               ADDRESS REGISTER USED AS REFERENCE.                    :
                    ; EXIT:                                                               :
                    ;               AFTER (CX) TIME COUNT (PLUS OR MINUS 31 MICROSECONDS)  :
                    ;        (CX) = 0                                                      :
                    ;--------------------------------------------------------------------

0CA0                WAITF    PROC   NEAR              ;       DELAY FOR  (CX)*15.085737 US
0CA0 50                      PUSH   AX                ; SAVE WORK REGISTER (AH)
0CA1 D1 E9                   SHR    CX,1              ; DIVIDE 15us COUNT DOWN TO 30us COUNT
0CA3 E3 13                   JCXZ   WAITF9            ; EXIT IF COUNT WAS ZERO OR ONE

0CA5 E6 0C                   OUT    DMA+12,AL         ; CLEAR THE DMA BYTE POINTER FLIP/FLOP
0CA7                WAITF1:
0CA7 9C                      PUSHF                    ; SAVE INTERRUPT STATE
0CA8 FA                      CLI                      ; BLOCK INTERRUPTS TILL NEXT CHANGE
0CA9                WAITF3:                           ;  WAIT FOR REFRESH ADDRESS CHANGE
0CA9 E4 00                   IN     AL,DMA            ; READ CURRENT ADDRESS LOW BYTE
0CAB 24 FE                   AND    AL,11111110B      ; DISCARD LOW BIT (30us)
0CAD 3A E0                   CMP    AH,AL             ; DID VALUE JUST CHANGE
0CAF 8A E0                   MOV    AH,AL             ; SAVE NEW/OLD VALUE INCASE IT DID
0CB1 E4 00                   IN     AL,DMA            ; READ HIGH BYTE (AND IGNORE)
0CB3 74 F4                   JE     WAITF3            ; WAIT FOR A CHANGE IN ADDRESS BITS

0CB5 9D                      POPF                     ; RESTORE INTERRUPTS
```

```
0CB6 E2 EF                      LOOP    WAITF1              ; DECREMENT CYCLES COUNT TILL COUNT END
0CB8                    WAITF9:
0CB8 58                         POP     AX                  ; RESTORE (AH)
0CB9 C3                         RET                         ; RETURN  (CX)= 0

0CBA                    WAITF   ENDP


                        ;---------------------------------------------------------------
                        ;       PRINT A SEGMENT VALUE TO LOOK LIKE A 20 BIT ADDRESS    :
                        ;       DX MUST CONTAIN SEGMENT VALUE TO BE PRINTED            :
                        ;---------------------------------------------------------------
0CBA                    PRT_SEG PROC    NEAR
0CBA 8A C6                      MOV     AL,DH               ;GET MSB
0CBC E8 1958 R                  CALL    XPC_BYTE
0CBF 8A C2                      MOV     AL,DL               ;LSB
0CC1 E8 1958 R                  CALL    XPC_BYTE
0CC4 B0 30                      MOV     AL,'0'              ; PRINT A '0 '
0CC6 E8 1969 R                  CALL    PRT_HEX
0CC9 B0 20                      MOV     AL,' '              ;SPACE
0CCB E8 1969 R                  CALL    PRT_HEX
0CCE C3                         RET
0CCF                    PRT_SEG ENDP


                        ;---------------------------------------------------------------
                        ; 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.                     :
                        ;---------------------------------------------------------------

0CCF                    STGTST_CNT      PROC    NEAR
0CCF 8B D9                      MOV     BX,CX               ; SAVE WORD COUNT OF BLOCK TO TEST
0CD1 FC                         CLD                         ; SET DIR FLAG TO INCREMENT
0CD2 2B FF                      SUB     DI,DI               ; SET DI=OFFSET 0 REL TO ES REG
0CD4 2B C0                      SUB     AX,AX               ; SETUP FOR 0->FF PATTERN TEST
0CD6                    C2_1:
0CD6 88 05                      MOV     [DI],AL             ; ON FIRST BYTE
0CD8 8A 05                      MOV     AL,[DI]
0CDA 32 C4                      XOR     AL,AH               ; O.K.?
0CDC 75 79                      JNZ     C7                  ; GO ERROR IF NOT
0CDE FE C4                      INC     AH
0CE0 8A C4                      MOV     AL,AH
0CE2 75 F2                      JNZ     C2_1                ; LOOP TILL WRAP THROUGH FF
0CE4 B8 55AA                    MOV     AX,055AAH           ; GET INITIAL DATA PATTERN TO WRITE
0CE7 8B D0                      MOV     DX,AX               ; SET INITIAL COMPARE PATTERN.
0CE9 F3/ AB                     REP     STOSW               ; FILL STORAGE LOCATIONS IN BLOCK
0CEB E4 61                      IN      AL,PORT_B
0CED 0C 30                      OR      AL,030H             ; TOGGLE PARITY CHECK LATCHES
0CEF E6 61                      OUT     PORT_B,AL
0CF1 90                         NOP
0CF2 24 CF                      AND     AL,0CFH
0CF4 E6 61                      OUT     PORT_B,AL
                        ;
0CF6 4F                         DEC     DI                  ; POINT TO LAST WORD JUST WRITTEN
0CF7 4F                         DEC     DI
0CF8 FD                         STD                         ; SET DIR FLAG TO GO BACKWARDS
0CF9 8B F7                      MOV     SI,DI               ; INITIALIZE DESTINATION POINTER
0CFB 8B CB                      MOV     CX,BX               ; SETUP WORD COUNT FOR LOOP
0CFD                    C3:                                 ;       INNER TEST LOOP
0CFD AD                         LODSW                       ; READ OLD TEST WORD FROM STORAGE
0CFE 33 C2                      XOR     AX,DX               ; DATA READ AS EXPECTED ?
0D00 75 57                      JNE     C7X                 ; NO - GO TO ERROR ROUTINE
0D02 B8 AA55                    MOV     AX,0AA55H           ; GET NEXT DATA PATTERN TO WRITE
0D05 AB                         STOSW                       ; WRITE INTO LOCATION JUST READ
0D06 E2 F5                      LOOP    C3                  ; DECREMENT WORD COUNT AND LOOP
                        ;
0D08 FC                         CLD                         ; SET DIR FLAG TO GO FORWARD
0D09 47                         INC     DI                  ; SET POINTER TO BEG LOCATION
0D0A 47                         INC     DI
0D0B 8B F7                      MOV     SI,DI               ; INITIALIZE DESTINATION POINTER
0D0D 8B CB                      MOV     CX,BX               ; SETUP WORD COUNT FOR LOOP
0D0F 8B D0                      MOV     DX,AX               ; SETUP COMPARE PATTERN OF "0AA55H".
0D11                    C4:                                 ;       INNER TEST LOOP
0D11 AD                         LODSW                       ; READ OLD TEST WORD FROM STORAGE
0D12 33 C2                      XOR     AX,DX               ; DATA READ AS EXPECTED ?
0D14 75 43                      JNE     C7X                 ; NO - GO TO ERROR ROUTINE
0D16 B8 FFFF                    MOV     AX,0FFFFH           ; GET NEXT DATA PATTERN TO WRITE
0D19 AB                         STOSW                       ; WRITE INTO LOCATION JUST READ
0D1A E2 F5                      LOOP    C4                  ; DECREMENT WORD COUNT AND LOOP
                        ;
0D1C 4F                         DEC     DI                  ; POINT TO LAST WORD JUST WRITTEN
0D1D 4F                         DEC     DI
0D1E FD                         STD                         ; SET DIR FLAG TO GO BACKWARDS
0D1F 8B F7                      MOV     SI,DI               ; INITIALIZE DESTINATION POINTER
0D21 8B CB                      MOV     CX,BX               ; SETUP WORD COUNT FOR LOOP
0D23 8B D0                      MOV     DX,AX               ; SETUP COMPARE PATTERN "0FFFFH"
0D25                    C5:                                 ;       INNER TEST LOOP
0D25 AD                         LODSW                       ; READ OLD TEST WORD FROM STORAGE
0D26 33 C2                      XOR     AX,DX               ; DATA READ AS EXPECTED ?
0D28 75 2F                      JNE     C7X                 ; NO - GO TO ERROR ROUTINE
0D2A B8 0101                    MOV     AX,00101H           ; GET NEXT DATA PATTERN TO WRITE
0D2D AB                         STOSW                       ; WRITE INTO LOCATION JUST READ
0D2E E2 F5                      LOOP    C5                  ; DECREMENT WORD COUNT AND LOOP
                        ;
0D30 FC                         CLD                         ; SET DIR FLAG TO GO FORWARD
0D31 47                         INC     DI                  ; SET POINTER TO BEG LOCATION
0D32 47                         INC     DI
0D33 8B F7                      MOV     SI,DI               ; INITIALIZE DESTINATION POINTER
0D35 8B CB                      MOV     CX,BX               ; SETUP WORD COUNT FOR LOOP
0D37 8B D0                      MOV     DX,AX               ; SETUP COMPARE PATTERN "00101H".
0D39                    C6:                                 ;       INNER TEST LOOP
0D39 AD                         LODSW                       ; READ OLD TEST WORD FROM STORAGE
0D3A 33 C2                      XOR     AX,DX               ; DATA READ AS EXPECTED ?
0D3C 75 1B                      JNE     C7X                 ; NO - GO TO ERROR ROUTINE
0D3E AB                         STOSW                       ; WRITE ZERO INTO LOCATION READ
0D3F E2 F8                      LOOP    C6                  ; DECREMENT WORD COUNT AND LOOP
```

SECTION 5

```
                                ;
0D41 4F                                 DEC     DI                      ; POINT TO LAST WORD JUST WRITTEN
0D42 4F                                 DEC     DI
0D43 FD                                 STD                             ; SET DIR FLAG TO GO BACKWARDS
0D44 8B F7                              MOV     SI,DI                   ; INITIALIZE DESTINATION POINTER
0D46 8B CB                              MOV     CX,BX                   ; SETUP WORD COUNT FOR LOOP
0D48 8B D0                              MOV     DX,AX                   ; SETUP COMPARE PATTERN "00000H"
0D4A                            C6X:
0D4A AD                                 LODSW                           ; VERIFY MEMORY IS ZERO.
0D4B 33 C2                              XOR     AX,DX                   ; DATA READ AS EXPECTED ?
0D4D 75 0A                              JNE     C7X                     ; NO - GO TO ERROR ROUTINE
0D4F E2 F9                              LOOP    C6X                     ; DECREMENT WORD COUNT AND LOOP
                                ;
0D51 E4 62                              IN      AL,PORT_C               ; DID A PARITY ERROR OCCUR ?
0D53 24 C0                              AND     AL,0C0H                 ; ZERO FLAG WILL BE OFF, IF PARITY ERROR
0D55 B0 00                              MOV     AL,0                    ; AL=0 DATA COMPARE OK
0D57                            C7:
0D57 FC                                 CLD                             ; SET DIRECTION FLAG TO INC
0D58 C3                                 RET
0D59                            C7X:
0D59 3C 00                              CMP     AL,0                    ; FIND BYTE THAT FAILED.
0D5B 75 FA                              JNZ     C7
0D5D 8A C4                              MOV     AL,AH
0D5F EB F6                              JMP     SHORT C7
0D61                            STGTST_CNT      ENDP

                                ;       ORG     0EF57H
0F57                                    ORG     00F57H
0F57 E9 0000 E                  DISK_INT:       JMP     DISK_INT_1

                                ;       ORG     0EF79H
0F79                                    ORG     00F79H
                                ;------------------------------------------------------------------
                                ;       MEDIA/DRIVE PARAMETER TABLES                              :
                                ;------------------------------------------------------------------
                                ;------------------------------------------------------------------
                                ;       40 TRACK LOW DATA RATE MEDIA IN 40 TRACK LOW DATA RATE DRIVE :
                                ;------------------------------------------------------------------
0F79                            MD_TBL1         LABEL BYTE
0F79 DF                                 DB      11011111B       ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
0F7A 02                                 DB      2               ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
0F7B 25                                 DB      MOTOR_WAIT      ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
0F7C 02                                 DB      2               ; 512 BYTES/SECTOR
0F7D 09                                 DB      09              ; EOT ( LAST SECTOR ON TRACK)
0F7E 2A                                 DB      02AH            ; GAP LENGTH
0F7F FF                                 DB      0FFH            ; DTL
0F80 50                                 DB      050H            ; GAP LENGTH FOR FORMAT
0F81 F6                                 DB      0F6H            ; FILL BYTE FOR FORMAT
0F82 0F                                 DB      15              ; HEAD SETTLE TIME (MILLISECONDS)
0F83 08                                 DB      8               ; MOTOR START TIME (1/8 SECONDS)
0F84 27                                 DB      39              ; MAX. TRACK NUMBER
0F85 80                                 DB      RATE_250        ; DATA TRANSFER RATE
                                ;------------------------------------------------------------------
                                ;       40 TRACK LOW DATA RATE MEDIA IN 80 TRACK HI DATA RATE DRIVE  :
                                ;------------------------------------------------------------------
0F86                            MD_TBL2         LABEL BYTE
0F86 DF                                 DB      11011111B       ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
0F87 02                                 DB      2               ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
0F88 25                                 DB      MOTOR_WAIT      ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
0F89 02                                 DB      2               ; 512 BYTES/SECTOR
0F8A 09                                 DB      09              ; EOT ( LAST SECTOR ON TRACK)
0F8B 2A                                 DB      02AH            ; GAP LENGTH
0F8C FF                                 DB      0FFH            ; DTL
0F8D 50                                 DB      050H            ; GAP LENGTH FOR FORMAT
0F8E F6                                 DB      0F6H            ; FILL BYTE FOR FORMAT
0F8F 0F                                 DB      15              ; HEAD SETTLE TIME (MILLISECONDS)
0F90 08                                 DB      8               ; MOTOR START TIME (1/8 SECONDS)
0F91 27                                 DB      39              ; MAX. TRACK NUMBER
0F92 40                                 DB      RATE_300        ; DATA TRANSFER RATE
                                ;------------------------------------------------------------------
                                ;       80 TRACK HI DATA RATE MEDIA IN 80 TRACK HI DATA RATE DRIVE   :
                                ;------------------------------------------------------------------
0F93                            MD_TBL3         LABEL BYTE
0F93 DF                                 DB      11011111B       ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
0F94 02                                 DB      2               ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
0F95 25                                 DB      MOTOR_WAIT      ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
0F96 02                                 DB      2               ; 512 BYTES/SECTOR
0F97 0F                                 DB      15              ; EOT ( LAST SECTOR ON TRACK)
0F98 1B                                 DB      01BH            ; GAP LENGTH
0F99 FF                                 DB      0FFH            ; DTL
0F9A 54                                 DB      054H            ; GAP LENGTH FOR FORMAT
0F9B F6                                 DB      0F6H            ; FILL BYTE FOR FORMAT
0F9C 0F                                 DB      15              ; HEAD SETTLE TIME (MILLISECONDS)
0F9D 08                                 DB      8               ; MOTOR START TIME (1/8 SECONDS)
0F9E 4F                                 DB      79              ; MAX. TRACK NUMBER
0F9F 00                                 DB      RATE_500        ; DATA TRANSFER RATE
                                ;------------------------------------------------------------------
                                ;       80 TRACK LOW DATA RATE MEDIA IN 80 TRACK LOW DATA RATE DRIVE :
                                ;------------------------------------------------------------------
0FA0                            MD_TBL4         LABEL BYTE
0FA0 DF                                 DB      11011111B       ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
0FA1 02                                 DB      2               ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
0FA2 25                                 DB      MOTOR_WAIT      ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
0FA3 02                                 DB      2               ; 512 BYTES/SECTOR
0FA4 09                                 DB      09              ; EOT ( LAST SECTOR ON TRACK)
0FA5 2A                                 DB      02AH            ; GAP LENGTH
0FA6 FF                                 DB      0FFH            ; DTL
0FA7 50                                 DB      050H            ; GAP LENGTH FOR FORMAT
0FA8 F6                                 DB      0F6H            ; FILL BYTE FOR FORMAT
0FA9 0F                                 DB      15              ; HEAD SETTLE TIME (MILLISECONDS)
0FAA 08                                 DB      8               ; MOTOR START TIME (1/8 SECONDS)
0FAB 4F                                 DB      79              ; MAX. TRACK NUMBER
0FAC 80                                 DB      RATE_250        ; DATA TRANSFER RATE
                                ;------------------------------------------------------------------
                                ;       80 TRACK LOW DATA RATE MEDIA IN 80 TRACK HI DATA RATE DRIVE  :
                                ;------------------------------------------------------------------
0FAD                            MD_TBL5         LABEL BYTE
0FAD DF                                 DB      11011111B       ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
0FAE 02                                 DB      2               ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
0FAF 25                                 DB      MOTOR_WAIT      ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
0FB0 02                                 DB      2               ; 512 BYTES/SECTOR
0FB1 09                                 DB      09              ; EOT ( LAST SECTOR ON TRACK)
0FB2 2A                                 DB      02AH            ; GAP LENGTH
0FB3 FF                                 DB      0FFH            ; DTL
```

```
0FB4 50                         DB      050H            ; GAP LENGTH FOR FORMAT
0FB5 F6                         DB      0F6H            ; FILL BYTE FOR FORMAT
0FB6 0F                         DB      15              ; HEAD SETTLE TIME (MILLISECONDS)
0FB7 08                         DB      8               ; MOTOR START TIME (1/8 SECONDS)
0FB8 4F                         DB      79              ; MAX. TRACK NUMBER
0FB9 80                         DB      RATE_250        ; DATA TRANSFER RATE
                        ;----------------------------------------------------------------
                        ;       80 TRACK HI DATA RATE MEDIA IN 80 TRACK HI DATA RATE DRIVE   :
                        ;----------------------------------------------------------------
0FBA                    MD_TBL6         LABEL BYTE
0FBA AF                         DB      10101111B       ; SRT=A, HD UNLOAD=0F - 1ST SPECIFY BYTE
0FBB 02                         DB      2               ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
0FBC 25                         DB      MOTOR_WAIT      ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
0FBD 02                         DB      2               ; 512 BYTES/SECTOR
0FBE 12                         DB      18              ; EOT ( LAST SECTOR ON TRACK)
0FBF 1B                         DB      01BH            ; GAP LENGTH
0FC0 FF                         DB      0FFH            ; DTL
0FC1 6C                         DB      06CH            ; GAP LENGTH FOR FORMAT
0FC2 F6                         DB      0F6H            ; FILL BYTE FOR FORMAT
0FC3 0F                         DB      15              ; HEAD SETTLE TIME (MILLISECONDS)
0FC4 08                         DB      8               ; MOTOR START TIME (1/8 SECONDS)
0FC5 4F                         DB      79              ; MAX. TRACK NUMBER
0FC6 00                         DB      RATE_500        ; DATA TRANSFER RATE

                        ;----------------------------------------------------------------
                        ; 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 :
                        ;       DISK_POINTER TO IT.                                            :
                        ;----------------------------------------------------------------
                        ;       ORG     0EFC7H
0FC7                            ORG     00FC7H
0FC7                    DISK_BASE       LABEL   BYTE
0FC7 CF                         DB      11001111B       ; SRT=C, 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 08                         DB      8               ; EOT ( LAST SECTOR ON TRACK)
0FCC 2A                         DB      02AH            ; GAP LENGTH
0FCD FF                         DB      0FFH            ; DTL
0FCE 50                         DB      050H            ; GAP LENGTH FOR FORMAT
0FCF F6                         DB      0F6H            ; FILL BYTE FOR FORMAT
0FD0 19                         DB      25              ; HEAD SETTLE TIME (MILLISECONDS)
0FD1 04                         DB      4               ; MOTOR START TIME (1/8 SECONDS)

                        ;       ORG     0EFD2H
0FD2                            ORG     00FD2H
0FD2                    PRINTER_IO:
0FD2 E9 0000 E                  JMP     PRINTER_IO_1

                        ;       ORG     0F045H
1045                            ORG     01045H
1045 0000 E             M1      DW      OFFSET  SET_MODE        ;  TABLE OF ROUTINES WITHIN VIDEO I/O
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:
1065 E9 0000 E                  JMP     VIDEO_IO_1

                        ;-----  VIDEO PARAMETERS --- INIT_TABLE

                        ;;-     ORG     0F0A4H
10A4                            ORG     010A4H

10A4                    VIDEO_PARMS     LABEL   BYTE
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
                                                        ; TABLE OF REGEN LENGTHS
10E4 0800               M5      DW      2048            ; 40X25
10E6 1000                       DW      4096            ; 80X25
10E8 4000                       DW      16384           ; GRAPHICS
10EA 4000                       DW      16384

                        ;-----  COLUMNS
10EC 28 28 50 50 28 28  M6      DB      40,40,80,80,40,40,80,80
     50 50

                        ;-----  C_REG_TAB
10F4 2C 28 2D 29 2A 2E  M7      DB      2CH,28H,2DH,29H,2AH,2EH,1EH,29H ; TABLE OF MODE SETS
     1E 29
```

SECTION 5

```
                                PAGE
                                ;--- INT 12 -------------------------------------------------------------
                                ; MEMORY_SIZE_DET
                                ;       THIS ROUTINE DETERMINES THE AMOUNT OF MEMORY IN THE SYSTEM          :
                                ;       AS REPRESENTED BY THE SWITCHES ON THE PLANAR.  NOTE THAT THE       :
                                ;       SYSTEM MAY NOT BE ABLE TO USE I/O MEMORY UNLESS THERE IS A FULL :
                                ;       COMPLEMENT OF 64K BYTES ON THE PLANAR.                              :
                                ; INPUT                                                                     :
                                ;       NO REGISTERS                                                        :
                                ;       THE MEMORY_SIZE VARIABLE IS SET DURING POWER ON DIAGNOSTICS        :
                                ;        ACCORDING TO THE FOLLOWING HARDWARE ASSUMPTIONS:                   :
                                ;       PORT 60 BITS 3,2 = 00 - 256K BASE RAM                               :
                                ;                          01 - 512K BASE RAM                               :
                                ;                          10 - 576K BASE RAM                               :
                                ;                          11 - 640K BASE RAM                               :
                                ;       PORT 62 BITS 3-0 INDICATE AMOUNT OF I/O RAM IN 32K INCREMENTS      :
                                ;               E.G., 0000 - NO RAM IN I/O CHANNEL                          :
                                ;                     0010 - 64K RAM IN I/O CHANNEL, ETC.                   :
                                ; OUTPUT                                                                    :
                                ;       (AX) = NUMBER OF CONTIGUOUS 1K BLOCKS OF MEMORY                     :
                                ;------------------------------------------------------------------------
                                        ASSUME  CS:CODE,DS:DATA
                                ;       ORG     0F841H
     1841                               ORG     01841H
     1841                       MEMORY_SIZE_DET PROC    FAR
     1841 FB                            STI                             ; INTERRUPTS BACK ON
     1842 1E                            PUSH    DS                      ; SAVE SEGMENT
     1843 E8 1A12 R                     CALL    DDS
     1846 A1 0013 R                     MOV     AX,@MEMORY_SIZE         ; GET VALUE
     1849 1F                            POP     DS                      ; RECOVER SEGMENT
     184A CF                            IRET                            ; RETURN TO CALLER
     184B                       MEMORY_SIZE_DET  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 60  = LOW ORDER BYTE OF EQUPMENT                           :
                                ;       PORT 3FA = INTERRUPT ID REGISTER OF 8250                        :
                                ;               BITS 7-3 ARE ALWAYS 0                                   :
                                ;       PORT 378 = OUTPUT PORT OF PRINTER -- 8255 PORT THAT             :
                                ;               CAN BE READ AS WELL AS WRITTEN                          :
                                ; OUTPUT                                                                :
                                ;       (AX) IS SET, BIT SIGNIFICANT, TO INDICATE ATTACHED I/O          :
                                ;       BIT 15,14 = NUMBER OF PRINTERS ATTACHED                         :
                                ;       BIT 13 NOT USED                                                 :
                                ;       BIT 12 = GAME I/O ATTACHED                                      :
                                ;       BIT 11,10,9 = NUMBER OF RS232 CARDS ATTACHED                    :
                                ;       BIT 8 UNUSED                                                    :
                                ;       BIT 7,6 = NUMBER OF DISKETTE DRIVES                             :
                                ;               00=1, 01=2, 10=3, 11=4 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 3,2 = PLANAR RAM SIZE (00=256K,01=512K,10=576K,11=640K)     :
                                ;       BIT 1 = MATH COPROCESSOR                                        :
                                ;       BIT 0 = IPL FROM DISKETTE -- THIS BIT INDICATES THAT            :
                                ;               THERE ARE DISKETTE DRIVES ON THE SYSTEM                 :
                                ;                                                                       :
                                ;       NO OTHER REGISTERS AFFECTED                                     :
                                ;-----------------------------------------------------------------------:
                                        ASSUME  CS:CODE,DS:DATA
                                ;       ORG     0F84DH
     184D                               ORG     0184DH
     184D                       EQUIPMENT       PROC    FAR
     184D FB                            STI                             ; INTERRUPTS BACK ON
     184E 1E                            PUSH    DS                      ; SAVE SEGMENT REGISTER
     184F E8 1A12 R                     CALL    DDS
     1852 A1 0010 R                     MOV     AX,@EQUIP_FLAG          ; GET THE CURRENT SETTINGS
     1855 1F                            POP     DS                      ; RECOVER SEGMENT
     1856 CF                            IRET                            ; RETURN TO CALLER
     1857                       EQUIPMENT       ENDP

                                ;--- INT 15 -------------------------------------------------------------
                                ;
                                ;       ORG     0F859H
     1859                               ORG     01859H
     1859                       CASSETTE_IO:
     1859 E9 0000 E                     JMP     CASSETTE_IO_1


                                ;--------------------------------------------------------------------
                                ; NON-MASKABLE INTERRUPT ROUTINE:                                    :
                                ;       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 (INTERMITTANT :
                                ;       READ PROBLEM) ?????<-WILL BE PRINTED WHERE THE ADDRESS    :
                                ;       WOULD NORMALLY GO.                                           :
                                ;--------------------------------------------------------------------
     185C                       NMI_INT_1       PROC    NEAR
                                        ASSUME  DS:DATA
     185C 50                            PUSH    AX                      ; SAVE ORIG CONTENTS OF AX
     185D E4 62                         IN      AL,PORT_C
     185F A8 C0                         TEST    AL,0C0H                 ; PARITY CHECK?
     1861 75 03                         JNZ     NMI_1
     1863 EB 58 90                      JMP     D14                     ; NO, EXIT FROM ROUTINE
     1866                       NMI_1:
     1866 BA ---- R                     MOV     DX,DATA
     1869 8E DA                         MOV     DS,DX
     186B BE 18E2 R                     MOV     SI,OFFSET D1            ; ADDR OF ERROR MSG
     186E A8 40                         TEST    AL,40H                  ; I/O PARITY CHECK
     1870 75 03                         JNZ     D13                     ; DISPLAY ERROR MSG
     1872 BE 18F2 R                     MOV     SI,OFFSET D2            ; MUST BE PLANAR
     1875                       D13:
     1875 B4 00                         MOV     AH,0                    ; INIT AND SET MODE FOR VIDEO
     1877 A0 0049 R                     MOV     AL,@CRT_MODE
     187A CD 10                         INT     10H                     ; CALL VIDEO_IO PROCEDURE
     187C E8 1997 R                     CALL    P_MSG                   ; PRINT ERROR MSG
```

```

                                ;----- SEE IF LOCATION THAT CAUSED PARITY CHECK CAN BE FOUND

187F B0 00                              MOV     AL,00H                  ; DISABLE TRAP
1881 E6 A0                              OUT     0A0H,AL
1883 E4 61                              IN      AL,PORT_B
1885 0C 30                              OR      AL,00110000B            ; TOGGLE PARITY CHECK ENABLES
1887 E6 61                              OUT     PORT_B,AL
1889 24 CF                              AND     AL,11001111B
188B E6 61                              OUT     PORT_B,AL
188D 8B 1E 0013 R                       MOV     BX,@MEMORY_SIZE         ; GET MEMORY SIZE WORD
1891 FC                                 CLD                             ; SET DIR FLAG TO INCRIMENT
1892 2B D2                              SUB     DX,DX                   ; POINT DX AT START OF MEM
1894                            NMI_LOOP:
1894 8E DA                              MOV     DS,DX
1896 8E C2                              MOV     ES,DX
1898 B9 4000                            MOV     CX,4000H                ; SET FOR 16KB SCAN
189B 2B F6                              SUB     SI,SI                   ; SET SI TO BE REALTIVE TO
                                                                        ; START OF ES
189D F3/ AC                             REP     LODSB                   ; READ 16KB OF MEMORY
189F E4 62                              IN      AL,PORT_C               ; SEE IF PARITY CHECK HAPPENED
18A1 24 C0                              AND     AL,11000000B
18A3 75 11                              JNZ     PRT_NMI                 ; GO PRINT ADDRESS IF IT DID
18A5 81 C2 0400                         ADD     DX,0400H                ; POINT TO NEXT 16K BLOCK
18A9 83 EB 10                           SUB     BX,16D
18AC 75 E6                              JNZ     NMI_LOOP
18AE BE 1902 R                          MOV     SI,(OFFSET D2A)         ; PRINT ROW OF ????? IF PARITY
18B1 E8 1997 R                          CALL    P_MSG                   ; CHECK COULD NOT BE RE-CREATED
18B4 FA                                 CLI
18B5 F4                                 HLT                             ; HALT SYSTEM
18B6                            PRT_NMI:
18B6 8C DA                              MOV     DX,DS
18B8 E8 0CBA R                          CALL    PRT_SEG                 ; PRINT SEGMENT VALUE
18BB FA                                 CLI                             ; HALT SYSTEM
18BC F4                                 HLT
18BD                            D14:
18BD 58                                 POP     AX                      ; RESTORE ORIG CONTENTS OF AX
18BE CF                                 IRET
18BF                            NMI_INT_1       ENDP


                                ;---------------------------------------
                                ;       ROS CHECKSUM SUBROUTINE         :
                                ;---------------------------------------
18BF                            ROS_CHECKSUM    PROC    NEAR            ; NEXT_ROS_MODULE
18BF B9 0000                            MOV     CX,0                    ; NUMBER OF BYTES TO ADD
18C2                            ROS_CHECKSUM_CNT:                       ; ENTRY FOR OPTIONAL ROS TEST
18C2 32 C0                              XOR     AL,AL
18C4                            C26:
18C4 02 07                              ADD     AL,DS:[BX]
18C6 43                                 INC     BX                      ; POINT TO NEXT BYTE
18C7 E2 FB                              LOOP    C26                     ; ADD ALL BYTES IN ROS MODULE
18C9 0A C0                              OR      AL,AL                   ; SUM = 0?
18CB C3                                 RET
18CC                            ROS_CHECKSUM    ENDP
                                ;---------------------------------------
                                ;       MESSAGE AREA FOR POST           :
                                ;---------------------------------------
18CC 31 30 31 0D 0A             E0      DB      '101',CR,LF             ; SYSTEM BOARD ERROR
18D1 20 32 30 31 0D 0A          E1      DB      ' 201',CR,LF            ; MEMORY ERROR
18D7 52 4F 4D 0D 0A             F3A     DB      'ROM',CR,LF             ; ROM CHECKSUM ERROR
18DC 31 38 30 31 0D 0A          F3C     DB      '1801',CR,LF            ; EXPANSION IO BOX ERROR
18E2 50 41 52 49 54 59          D1      DB      'PARITY CHECK 2',CR,LF
     20 43 48 45 43 4B
     20 32 0D 0A
18F2 50 41 52 49 54 59          D2      DB      'PARITY CHECK 1',CR,LF
     20 43 48 45 43 4B
     20 31 0D 0A
1902 3F 3F 3F 3F 3F 0D          D2A     DB      '?????',CR,LF
     0A


                                ;------------------------------------------------------------------------
                                ;       BLINK LED PROCEDURE FOR MFG RUN-IN TESTS
                                ;        IF LED IS ON, TURN IT OFF. IF OFF, TURN ON.
                                ;------------------------------------------------------------------------
                                        ASSUME  DS:DATA
1909                            BLINK_INT       PROC    NEAR
1909 FB                                 STI
190A 50                                 PUSH    AX                      ; SAVE AX REG CONTENTS
190B E4 61                              IN      AL,PORT_B               ; READ CURRENT VAL OF PORT B
190D 8A E0                              MOV     AH,AL
190F F6 D0                              NOT     AL                      ; FLIP ALL BITS
1911 24 40                              AND     AL,01000000B            ; ISOLATE CONTROL BIT
1913 80 E4 BF                           AND     AH,10111111B            ; MASK OUT OF ORIGINAL VAL
1916 0A C4                              OR      AL,AH                   ; OR NEW CONTROL BIT IN
1918 E6 61                              OUT     PORT_B,AL
191A B0 20                              MOV     AL,EOI
191C E6 20                              OUT     INTA00,AL
191E 58                                 POP     AX                      ; RESTORE AX REG
191F CF                                 IRET
1920                            BLINK_INT       ENDP


                                ;-------------------------------------------------------
                                ; THIS ROUTINE CHECKSUMS OPTIONAL ROM MODULES AND      :
                                ; IF CHECKSUM IS OK, CALLS INIT/TEST CODE IN MODULE    :
                                ;-------------------------------------------------------
1920                            ROM_CHECK       PROC    NEAR
1920 B8 ---- R                          MOV     AX,DATA                 ; POINT ES TO DATA AREA
1923 8E C0                              MOV     ES,AX
1925 2A E4                              SUB     AH,AH                   ; ZERO OUT AH
1927 8A 47 02                           MOV     AL,[BX+2]               ; GET LENGTH INDICATOR
192A B1 09                              MOV     CL,09H                  ; MULTIPLY BY 512
192C D3 E0                              SHL     AX,CL
192E 8B C8                              MOV     CX,AX                   ; SET COUNT
1930 51                                 PUSH    CX                      ; SAVE COUNT
1931 B9 0004                            MOV     CX,4                    ; ADJUST
1934 D3 E8                              SHR     AX,CL
1936 03 D0                              ADD     DX,AX                   ; SET POINTER TO NEXT MODULE
1938 59                                 POP     CX                      ; RETRIVE COUNT
1939 E8 18C2 R                          CALL    ROS_CHECKSUM_CNT        ; DO CHECKSUM
193C 74 06                              JZ      ROM_CHECK_1
193E E8 0746 R                          CALL    ROM_ERR                 ; POST CHECKSUM ERROR
1941 EB 14 90                           JMP     ROM_CHECK_END           ; AND EXIT
1944                            ROM_CHECK_1:
1944 52                                 PUSH    DX                      ; SAVE POINTER
```

SECTION 5

```
1945 26: C7 06 0067 R 0003        MOV     ES:@IO_ROM_INIT,0003H   ; LOAD OFFSET
194C 26: 8C 1E 0069 R             MOV     ES:@IO_ROM_SEG,DS       ; LOAD SEGMENT
1951 26: FF 1E 0067 R             CALL    DWORD PTR ES:@IO_ROM_INIT        ; CALL INIT./TEST ROUTINE
1956 5A                           POP     DX
1957                      ROM_CHECK_END:
1957 C3                           RET                             ; RETURN TO CALLER
1958                      ROM_CHECK       ENDP

                          ;----------------------------------------------
                          ; CONVERT AND PRINT ASCII CODE                 :
                          ;       AL MUST CONTAIN NUMBER TO BE CONVERTED. :
                          ;       AX AND BX DESTROYED.                    :
                          ;----------------------------------------------
1958                      XPC_BYTE        PROC    NEAR
1958 50                           PUSH    AX                      ; SAVE FOR LOW NIBBLE DISPLAY
1959 B1 04                        MOV     CL,4                    ; SHIFT COUNT
195B D2 E8                        SHR     AL,CL                   ; NYBBLE SWAP
195D E8 1963 R                    CALL    XLAT_PR                 ; DO THE HIGH NIBBLE DISPLAY
1960 58                           POP     AX                      ; RECOVER THE NIBBLE
1961 24 0F                        AND     AL,0FH                  ; ISOLATE TO LOW NIBBLE
                                                                  ; FALL INTO LOW NIBBLE CONVERSION
1963                      XLAT_PR PROC    NEAR                    ;  CONVERT 00-0F TO ASCII CHARACTER
1963 04 90                        ADD     AL,090H                 ; ADD FIRST CONVERSION FACTOR
1965 27                           DAA                             ; ADJUST FOR NUMERIC AND ALPHA RANGE
1966 14 40                        ADC     AL,040H                 ; ADD CONVERSION AND ADJUST LOW NIBBLE
1968 27                           DAA                             ; ADJUST HIGH NIBBLE TO ASCHI RANGE
1969                      PRT_HEX PROC    NEAR
1969 B4 0E                        MOV     AH,14                   ; DISPLAY CHARACTER IN AL
196B B7 00                        MOV     BH,0
196D CD 10                        INT     10H                     ; CALL VIDEO_IO
196F C3                           RET
1970                      PRT_HEX ENDP
1970                      XLAT_PR ENDP
1970                      XPC_BYTE        ENDP

1970                      F4      LABEL   WORD                    ; PRINTER SOURCE TABLE
1970 03BC                         DW      3BCH
1972 0378                         DW      378H
1974 0278                         DW      278H
1976                      F4E     LABEL   WORD

                          ;-------------------------------------------------------
                          ; 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         :
                          ;-------------------------------------------------------
1976                      E_MSG   PROC    NEAR
1976 8B EE                        MOV     BP,SI                   ; SET BP NON-ZERO TO FLAG ERR
1978 E8 1997 R                    CALL    P_MSG                   ; PRINT MESSAGE
197B 1E                           PUSH    DS
197C E8 1A12 R                    CALL    DDS
197F A0 0010 R                    MOV     AL,BYTE PTR @EQUIP_FLAG ; LOOP/HALT ON ERROR
1982 24 01                        AND     AL,01H                  ; SWITCH ON?
1984 75 0F                        JNZ     G12                     ; NO - RETURN
1986                      MFG_HALT:
1986 FA                           CLI                             ; YES -  HALT SYSTEM
1987 B0 89                        MOV     AL,89H
1989 E6 63                        OUT     CMD_PORT,AL
198B B0 85                        MOV     AL,10000101B            ; DISABLE KB
198D E6 61                        OUT     PORT_B,AL
198F A0 0015 R                    MOV     AL,@MFG_ERR_FLAG        ; RECOVER ERROR INDICATOR
1992 E6 60                        OUT     PORT_A,AL               ; SET INTO 8255 REG
1994 F4                           HLT                             ; HALT SYS
1995                      G12:
1995 1F                           POP     DS                      ; WRITE_MSG:
1996 C3                           RET
1997                      E_MSG   ENDP

1997                      P_MSG   PROC    NEAR
1997                      G12A:
1997 2E: 8A 04                    MOV     AL,CS:[SI]              ; PUT CHAR IN AL
199A 46                           INC     SI                      ; POINT TO NEXT CHAR
199B 50                           PUSH    AX                      ; SAVE PRINT CHAR
199C E8 1969 R                    CALL    PRT_HEX                 ; CALL VIDEO_IO
199F 58                           POP     AX                      ; RECOVER PRINT CHAR
19A0 3C 0A                        CMP     AL,10                   ; WAS IT LINE FEED?
19A2 75 F3                        JNE     G12A                            ; NO,KEEP PRINTING STRING
19A4 C3                           RET
19A5                      P_MSG   ENDP
                                  ASSUME  CS:CODE,DS:DATA
                          ;------------------------------------------------------------------------
                          ;       THIS PROCEDURE WILL ISSUE LONG TONES (1-3/4 SECONDS) AND ONE OR :
                          ;       MORE SHORT TONES (9/32 SECOND) TO INDICATE A FAILURE ON THE     :
                          ;       PLANAR BOARD, A BAD MEMORY MODULE, OR A PROBLEM WITH THE CRT.   :
                          ; ENTRY PARAMETERS:                                                     :
                          ;       DH = NUMBER OF LONG TONES TO BEEP.                              :
                          ;       DL = NUMBER OF SHORT TONES TO BEEP.                             :
                          ;------------------------------------------------------------------------

19A5                      ERR_BEEP        PROC    NEAR
19A5 9C                           PUSHF                           ; SAVE FLAGS
19A6 FA                           CLI                             ; DISABLE SYSTEM INTERRUPTS
19A7 0A F6                        OR      DH,DH                   ; ANY LONG ONES TO BEEP
19A9 74 1E                        JZ      G3                      ; NO, DO THE SHORT ONES
19AB                      G1:                                     ;       LONG BEEPS
19AB B3 70                        MOV     BL,112                  ; COUNTER FOR LONG BEEPS (1-3/4 SECONDS)
19AD B9 0500                      MOV     CX,1280                 ; DIVISOR FOR 932 HZ
19B0 E8 0C5C R                    CALL    BEEP                    ; DO THE BEEP
19B3 B9 C233                      MOV     CX,49715                ; 2/3 SECOND DELAY AFTER LONG BEEP
19B6 E8 0CA0 R                    CALL    WAITF                   ; DELAY BETWEEN BEEPS
19B9 FE CE                        DEC     DH                      ; ANY MORE LONG BEEPS TO DO
19BB 75 EE                        JNZ     G1                      ; LOOP TILL DONE
19BD 1E                           PUSH    DS                      ; SAVE DS REGISTER CONTENTS
19BE E8 1A12 R                    CALL    DDS
19C1 80 3E 0012 R 01              CMP     @MFG_TST,01H            ; MANUFACTURING TEST MODE?
19C6 1F                           POP     DS                      ; RESTORE ORIGINAL CONTENTS OF (DS)
19C7 74 BD                        JE      MFG_HALT                ; YES - STOP BLINKING LED
19C9                      G3:                                     ;       SHORT BEEPS
19C9 B3 12                        MOV     BL,18                   ; COUNTER FOR A SHORT BEEP (9/32)
19CB B9 04B8                      MOV     CX,1208                 ; DIVISOR FOR 987 HZ
19CE E8 0C5C R                    CALL    BEEP                    ; DO THE SOUND
```

```
19D1 B9 8178                   MOV    CX,33144          ; 1/2 SECOND DELAY AFTER SHORT BEEP
19D4 E8 0CA0 R                 CALL   WAITF             ; DELAY BETWEEN BEEPS
19D7 FE CA                     DEC    DL                ; DONE WITH SHORT BEEPS COUNT
19D9 75 EE                     JNZ    G3                ; LOOP TILL DONE
19DB B9 8178                   MOV    CX,33144          ; 1/2 SECOND DELAY AFTER LAST BEEP
19DE E8 0CA0 R                 CALL   WAITF             ; MAKE IT ONE SECOND DELAY BEFORE RETURN
19E1 9D                        POPF                     ; RESTORE FLAGS TO ORIGINAL SETTINGS
19E2 C3                        RET                      ; RETURN TO CALLER
19E3                   ERR_BEEP       ENDP

                       ;---------------------------------------------------------------------
                       ;      THIS PROCEDURE WILL SEND A SOFTWARE RESET TO THE KEYBOARD.    :
                       ;      SCAN CODE 'AA' SHOULD BE RETURNED TO THE CPU.                 :
                       ;---------------------------------------------------------------------
19E3                   KBD_RESET      PROC   NEAR
                              ASSUME  DS:ABS0
19E3 B0 08                    MOV     AL,08H            ; SET KBD CLK LINE LOW
19E5 E6 61                    OUT     PORT_B,AL         ; WRITE 8255 PORT B
19E7 B9 2956                  MOV     CX,10582          ; HOLD KBD CLK LOW FOR 20 MS
19EA                   G8:
19EA E2 FE                    LOOP    G8                ; LOOP FOR 20 MS
19EC B0 C8                    MOV     AL,0C8H           ; SET CLK, ENABLE LINES HIGH
19EE E6 61                    OUT     PORT_B,AL
19F0                   SP_TEST:                         ; ENTRY FOR MANUFACTURING TEST 2
19F0 B0 48                    MOV     AL,48H            ; SET KBD CLK HIGH, ENABLE LOW
19F2 E6 61                    OUT     PORT_B,AL
19F4 B0 FD                    MOV     AL,0FDH           ; ENABLE KEYBOARD INTERRUPTS
19F6 E6 21                    OUT     INTA01,AL         ; WRITE 8259 IMR
19F8 C6 06 046B R 00          MOV     DATA_AREA[@INTR_FLAG-DATA40],0  ; RESET INTERRUPT INDICATOR
19FD FB                       STI                       ; ENABLE INTERRUPTS
19FE 2B C9                    SUB     CX,CX             ; SETUP INTERRUPT TIMEOUT CNT
1A00                   G9:
1A00 F6 06 046B R 02          TEST    DATA_AREA[@INTR_FLAG-DATA40],02H ; DID A KEYBOARD INTR OCCUR?
1A05 75 02                    JNZ     G10               ; YES - READ SCAN CODE RETURNED
1A07 E2 F7                    LOOP    G9                ; NO - LOOP TILL TIMEOUT
1A09                   G10:
1A09 E4 60                    IN      AL,PORT_A         ; READ KEYBOARD SCAN CODE
1A0B 8A D8                    MOV     BL,AL             ; SAVE SCAN CODE JUST READ
1A0D B0 C8                    MOV     AL,0C8H           ; CLEAR KEYBOARD
1A0F E6 61                    OUT     PORT_B,AL
1A11 C3                       RET                       ; RETURN TO CALLER
1A12                   KBD_RESET      ENDP

1A12                   DDS    PROC    NEAR              ;       LOAD (DS) TO DATA AREA
1A12 2E: 8E 1E 1A18 R         MOV     DS,CS:DDSDATA     ; PUT SEGMENT VALUE OF DATA AREA INTO DS
1A17 C3                       RET                       ; RETURN TO USER WITH (DS)= DATA

1A18 ---- R            DDSDATA DW     DATA              ; SEGMENT SELECTOR VALUE FOR DATA AREA

1A1A                   DDS    ENDP

                       ;-- HARDWARE INT  08 H -- ( IRQ LEVEL 0 ) ------------------------------------
                       ;                                                                           :
                       ;      THIS ROUTINE HANDLES THE TIMER INTERRUPT FROM FROM CHANNEL 0 OF      :
                       ;      THE 8254 TIMER.  INPUT FREQUENCY IS 1.19318 MHZ AND THE DIVISOR      :
                       ;      IS 65536, RESULTING IN APPROXIMATELY 18.2 INTERRUPTS EVERY SECOND.   :
                       ;                                                                           :
                       ;      THE INTERRUPT HANDLER MAINTAINS A COUNT (40:6C) OF INTERRUPTS SINCE  :
                       ;      POWER ON TIME, WHICH MAY BE USED TO ESTABLISH TIME OF DAY.           :
                       ;      THE INTERRUPT HANDLER ALSO DECREMENTS THE MOTOR CONTROL COUNT (40:40) :
                       ;      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.           :
                       ;----------------------------------------------------------------------------
                              ASSUME  CS:CODE,DS:DATA

1A1A                   TIMER_INT_1    PROC   NEAR
1A1A FB                       STI                       ; INTERRUPTS BACK ON
1A1B 1E                       PUSH    DS
1A1C 50                       PUSH    AX
1A1D 52                       PUSH    DX                ; SAVE MACHINE STATE
1A1E B8 ---- R                MOV     AX,DATA           ; GET ADDRESS OF DATA SEGMENT
1A21 8E D8                    MOV     DS,AX             ; ESTABLISH ADDRESSABILITY
1A23 FF 06 006C R             INC     @TIMER_LOW        ; INCREMENT TIME
1A27 75 04                    JNZ     T4                ; GO TO TEST_DAY
1A29 FF 06 006E R             INC     @TIMER_HIGH       ; INCREMENT HIGH WORD OF TIME
1A2D                   T4:                              ;       TEST_DAY
1A2D 83 3E 006E R 18          CMP     @TIMER_HIGH,018H  ; TEST FOR COUNT EQUALING 24 HOURS
1A32 75 19                    JNZ     T5                ; GO TO DISKETTE_CTL
1A34 81 3E 006C R 00B0        CMP     @TIMER_LOW,0B0H
1A3A 75 11                    JNZ     T5                ; GO TO DISKETTE_CTL

                       ;-----  TIMER HAS GONE 24 HOURS

1A3C 2B C0                    SUB     AX,AX
1A3E A3 006E R                MOV     @TIMER_HIGH,AX    ; CLEAR TIMER COUNT HIGH
1A41 A3 006C R                MOV     @TIMER_LOW,AX     ;  AND LOW
1A44 C6 06 0070 R 01          MOV     @TIMER_OFL,1      ; SET TIMER ELAPSED 24 HOURS FLAG
1A49 FF 06 00CE R             INC     @DAY_COUNT        ; INCREMENT ELAPSED DAY COUNTER

                       ;-----  TEST FOR DISKETTE TIME OUT

1A4D                   T5:
1A4D FE 0E 0040 R             DEC     @MOTOR_COUNT      ; DECREMENT DISKETTE MOTOR CONTROL
1A51 75 0B                    JNZ     T6                ; RETURN IF COUNT NOT OUT
1A53 80 26 003F R F0          AND     @MOTOR_STATUS,0F0H ; TURN OFF MOTOR RUNNING BITS
1A58 B0 0C                    MOV     AL,0CH
1A5A BA 03F2                  MOV     DX,03F2H          ; FDC CTL PORT
1A5D EE                       OUT     DX,AL             ; TURN OFF THE MOTOR

1A5E                   T6:                              ;       TIMER TICK INTERRUPT
1A5E CD 1C                    INT     1CH               ; TRANSFER CONTROL TO A USER ROUTINE

1A60 FA                       CLI                       ; DISABLE INTERRUPTS TILL STACK CLEARED
1A61 B0 20                    MOV     AL,EOI            ; GET END OF INTERRUPT MASK
1A63 E6 20                    OUT     INTA00,AL         ; END OF INTERRUPT TO 8259 - 1
1A65 5A                       POP     DX                ; RESTORE (DX)
1A66 58                       POP     AX
1A67 1F                       POP     DS                ; RESET MACHINE STATE
1A68 CF                       IRET                      ;  RETURN FROM INTERRUPT

1A69                   TIMER_INT_1    ENDP
```

SECTION 5

```
                               PAGE
                               ;--------------------------------------------------------------------
                               ;        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    BLANK
      00 00
 1A76 7E 81 A5 81 BD 99                   DB        07EH,081H,0A5H,081H,0BDH,099H,081H,07EH ; D_01    SMILING FACE
      81 7E
 1A7E 7E FF DB FF C3 E7                   DB        07EH,0FFH,0DBH,0FFH,0C3H,0E7H,0FFH,07EH ; D_02    SMILING FACE N
      FF 7E
 1A86 6C FE FE FE 7C 38                   DB        06CH,0FEH,0FEH,0FEH,07CH,038H,010H,000H ; D_03    HEART
      10 00
 1A8E 10 38 7C FE 7C 38                   DB        010H,038H,07CH,0FEH,07CH,038H,010H,000H ; D_04    DIAMOND
      10 00
 1A96 38 7C 38 FE FE 7C                   DB        038H,07CH,038H,0FEH,0FEH,07CH,038H,07CH ; D_05    CLUB
      38 7C
 1A9E 10 10 38 7C FE 7C                   DB        010H,010H,038H,07CH,0FEH,07CH,038H,07CH ; D_06    SPADE
      38 7C
 1AA6 00 00 18 3C 3C 18                   DB        000H,000H,018H,03CH,03CH,018H,000H,000H ; D_07    BULLET
      00 00
 1AAE FF FF E7 C3 C3 E7                   DB        0FFH,0FFH,0E7H,0C3H,0C3H,0E7H,0FFH,0FFH ; D_08    BULLET NEG
      FF FF
 1AB6 00 3C 66 42 42 66                   DB        000H,03CH,066H,042H,042H,066H,03CH,000H ; D_09    CIRCLE
      3C 00
 1ABE FF C3 99 BD BD 99                   DB        0FFH,0C3H,099H,0BDH,0BDH,099H,0C3H,0FFH ; D_0A    CIRCLE NEG
      C3 FF
 1AC6 0F 07 0F 7D CC CC                   DB        00FH,007H,00FH,07DH,0CCH,0CCH,0CCH,078H ; D_0B    MALE
      CC 78
 1ACE 3C 66 66 66 3C 18                   DB        03CH,066H,066H,066H,03CH,018H,07EH,018H ; D_0C    FEMALE
      7E 18
 1AD6 3F 33 3F 30 30 70                   DB        03FH,033H,03FH,030H,030H,070H,0F0H,0E0H ; D_0D    EIGHTH NOTE
      F0 E0
 1ADE 7F 63 7F 63 63 67                   DB        07FH,063H,07FH,063H,063H,067H,0E6H,0C0H ; D_0E    TWO 1/16 NOTE
      E6 C0
 1AE6 99 5A 3C E7 E7 3C                   DB        099H,05AH,03CH,0E7H,0E7H,03CH,05AH,099H ; D_0F    SUN
      5A 99

 1AEE 80 E0 F8 FE F8 E0                   DB        080H,0E0H,0F8H,0FEH,0F8H,0E0H,080H,000H ; D_10    R ARROWHEAD
      80 00
 1AF6 02 0E 3E FE 3E 0E                   DB        002H,00EH,03EH,0FEH,03EH,00EH,002H,000H ; D_11    L ARROWHEAD
      02 00
 1AFE 18 3C 7E 18 18 7E                   DB        018H,03CH,07EH,018H,018H,07EH,03CH,018H ; D_12    ARROW 2 VERT
      3C 18
 1B06 66 66 66 66 66 00                   DB        066H,066H,066H,066H,066H,000H,066H,000H ; D_13    2 EXCLAMATIONS
      66 00
 1B0E 7F DB DB 7B 1B 1B                   DB        07FH,0DBH,0DBH,07BH,01BH,01BH,01BH,000H ; D_14    PARAGRAPH
      1B 00
 1B16 3E 63 38 6C 6C 38                   DB        03EH,063H,038H,06CH,06CH,038H,0CCH,078H ; D_15    SECTION
      CC 78
 1B1E 00 00 00 00 7E 7E                   DB        000H,000H,000H,000H,07EH,07EH,07EH,000H ; D_16    RECTANGLE
      7E 00
 1B26 18 3C 7E 18 7E 3C                   DB        018H,03CH,07EH,018H,07EH,03CH,018H,0FFH ; D_17    ARROW 2 VRT UP
      18 FF
 1B2E 18 3C 7E 18 18 18                   DB        018H,03CH,07EH,018H,018H,018H,018H,000H ; D_18    ARROW VRT UP
      18 00
 1B36 18 18 18 18 7E 3C                   DB        018H,018H,018H,018H,07EH,03CH,018H,000H ; D_19    ARROW VRT DOWN
      18 00
 1B3E 00 18 0C FE 0C 18                   DB        000H,018H,00CH,0FEH,00CH,018H,000H,000H ; D_1A    ARROW RIGHT
      00 00
 1B46 00 30 60 FE 60 30                   DB        000H,030H,060H,0FEH,060H,030H,000H,000H ; D_1B    ARROW LEFT
      00 00
 1B4E 00 00 C0 C0 C0 FE                   DB        000H,000H,0C0H,0C0H,0C0H,0FEH,000H,000H ; D_1C    NOT INVERTED
      00 00
 1B56 00 24 66 FF 66 24                   DB        000H,024H,066H,0FFH,066H,024H,000H,000H ; D_1D    ARROW 2 HORZ
      00 00
 1B5E 00 18 3C 7E FF FF                   DB        000H,018H,03CH,07EH,0FFH,0FFH,000H,000H ; D_1E    ARROWHEAD UP
      00 00
 1B66 00 FF FF 7E 3C 18                   DB        000H,0FFH,0FFH,07EH,03CH,018H,000H,000H ; D_1F    ARROWHEAD DOWN
      00 00

 1B6E 00 00 00 00 00 00                   DB        000H,000H,000H,000H,000H,000H,000H,000H ; D_20    SPACE
      00 00
 1B76 30 78 78 30 30 00                   DB        030H,078H,078H,030H,030H,000H,030H,000H ; D_21 !  EXCLAMATION
      30 00
 1B7E 6C 6C 6C 00 00 00                   DB        06CH,06CH,06CH,000H,000H,000H,000H,000H ; D_22 "  QUOTATION
      00 00
 1B86 6C 6C FE 6C FE 6C                   DB        06CH,06CH,0FEH,06CH,0FEH,06CH,06CH,000H ; D_23 #  LB.
      6C 00
 1B8E 30 7C C0 78 0C F8                   DB        030H,07CH,0C0H,078H,00CH,0F8H,030H,000H ; D_24 $  DOLLAR SIGN
      30 00
 1B96 00 C6 CC 18 30 66                   DB        000H,0C6H,0CCH,018H,030H,066H,0C6H,000H ; D_25 %  PERCENT
      C6 00
 1B9E 38 6C 38 76 DC CC                   DB        038H,06CH,038H,076H,0DCH,0CCH,076H,000H ; D_26 &  AMPERSAND
      76 00
 1BA6 60 60 C0 00 00 00                   DB        060H,060H,0C0H,000H,000H,000H,000H,000H ; D_27 '  APOSTROPHE
      00 00
 1BAE 18 30 60 60 60 30                   DB        018H,030H,060H,060H,060H,030H,018H,000H ; D_28 (  L. PARENTHESIS
      18 00
 1BB6 60 30 18 18 18 30                   DB        060H,030H,018H,018H,018H,030H,060H,000H ; D_29 )  R. PARENTHESIS
      60 00
 1BBE 00 66 3C FF 3C 66                   DB        000H,066H,03CH,0FFH,03CH,066H,000H,000H ; D_2A *  ASTERISK
      00 00
 1BC6 00 30 30 FC 30 30                   DB        000H,030H,030H,0FCH,030H,030H,000H,000H ; D_2B +  PLUS
      00 00
 1BCE 00 00 00 00 00 30                   DB        000H,000H,000H,000H,000H,030H,030H,060H ; D_2C ,  COMMA
      30 60
 1BD6 00 00 00 FC 00 00                   DB        000H,000H,000H,0FCH,000H,000H,000H,000H ; D_2D -  DASH
      00 00
 1BDE 00 00 00 00 00 30                   DB        000H,000H,000H,000H,000H,030H,030H,000H ; D_2E .  PERIOD
      30 00
 1BE6 06 0C 18 30 60 C0                   DB        006H,00CH,018H,030H,060H,0C0H,080H,000H ; D_2F /  SLASH
      80 00

 1BEE 7C C6 CE DE F6 E6                   DB        07CH,0C6H,0CEH,0DEH,0F6H,0E6H,07CH,000H ; D_30 0
      7C 00
 1BF6 30 70 30 30 30 30                   DB        030H,070H,030H,030H,030H,030H,0FCH,000H ; D_31 1
      FC 00
 1BFE 78 CC 0C 38 60 CC                   DB        078H,0CCH,00CH,038H,060H,0CCH,0FCH,000H ; D_32 2
      FC 00
 1C06 78 CC 0C 38 0C CC                   DB        078H,0CCH,00CH,038H,00CH,0CCH,078H,000H ; D_33 3
      78 00
```

```
1C0E 1C 3C 6C CC FE 0C     DB   01CH,03CH,06CH,0CCH,0FEH,00CH,01EH,000H ; D_34 4
     1E 00
1C16 FC C0 F8 0C 0C CC     DB   0FCH,0C0H,0F8H,00CH,00CH,0CCH,078H,000H ; D_35 5
     78 00
1C1E 38 60 C0 F8 CC CC     DB   038H,060H,0C0H,0F8H,0CCH,0CCH,078H,000H ; D_36 6
     78 00
1C26 FC CC 0C 18 30 30     DB   0FCH,0CCH,00CH,018H,030H,030H,030H,000H ; D_37 7
     30 00
1C2E 78 CC CC 78 CC CC     DB   078H,0CCH,0CCH,078H,0CCH,0CCH,078H,000H ; D_38 8
     78 00
1C36 78 CC CC 7C 0C 18     DB   078H,0CCH,0CCH,07CH,00CH,018H,070H,000H ; D_39 9
     70 00
1C3E 00 30 30 00 00 30     DB   000H,030H,030H,000H,000H,030H,030H,000H ; D_3A :  COLON
     30 00
1C46 00 30 30 00 00 30     DB   000H,030H,030H,000H,000H,030H,030H,060H ; D_3B ;  SEMICOLON
     30 60
1C4E 18 30 60 C0 60 30     DB   018H,030H,060H,0C0H,060H,030H,018H,000H ; D_3C <  LESS THAN
     18 00
1C56 00 00 FC 00 00 FC     DB   000H,000H,0FCH,000H,000H,0FCH,000H,000H ; D_3D =  EQUAL
     00 00
1C5E 60 30 18 0C 18 30     DB   060H,030H,018H,00CH,018H,030H,060H,000H ; D_3E >  GREATER THAN
     60 00
1C66 78 CC 0C 18 30 00     DB   078H,0CCH,00CH,018H,030H,000H,030H,000H ; D_3F ?  QUESTION MARK
     30 00

1C6E 7C C6 DE DE DE C0     DB   07CH,0C6H,0DEH,0DEH,0DEH,0C0H,078H,000H ; D_40 @  AT
     78 00
1C76 30 78 CC CC FC CC     DB   030H,078H,0CCH,0CCH,0FCH,0CCH,0CCH,000H ; D_41 A
     CC 00
1C7E FC 66 66 7C 66 66     DB   0FCH,066H,066H,07CH,066H,066H,0FCH,000H ; D_42 B
     FC 00
1C86 3C 66 C0 C0 C0 66     DB   03CH,066H,0C0H,0C0H,0C0H,066H,03CH,000H ; D_43 C
     3C 00
1C8E F8 6C 66 66 66 6C     DB   0F8H,06CH,066H,066H,066H,06CH,0F8H,000H ; D_44 D
     F8 00
1C96 FE 62 68 78 68 62     DB   0FEH,062H,068H,078H,068H,062H,0FEH,000H ; D_45 E
     FE 00
1C9E FE 62 68 78 68 60     DB   0FEH,062H,068H,078H,068H,060H,0F0H,000H ; D_46 F
     F0 00
1CA6 3C 66 C0 C0 CE 66     DB   03CH,066H,0C0H,0C0H,0CEH,066H,03EH,000H ; D_47 G
     3E 00
1CAE CC CC CC FC CC CC     DB   0CCH,0CCH,0CCH,0FCH,0CCH,0CCH,0CCH,000H ; D_48 H
     CC 00
1CB6 78 30 30 30 30 30     DB   078H,030H,030H,030H,030H,030H,078H,000H ; D_49 I
     78 00
1CBE 1E 0C 0C 0C CC CC     DB   01EH,00CH,00CH,00CH,0CCH,0CCH,078H,000H ; D_4A J
     78 00
1CC6 E6 66 6C 78 6C 66     DB   0E6H,066H,06CH,078H,06CH,066H,0E6H,000H ; D_4B K
     E6 00
1CCE F0 60 60 60 62 66     DB   0F0H,060H,060H,060H,062H,066H,0FEH,000H ; D_4C L
     FE 00
1CD6 C6 EE FE FE D6 C6     DB   0C6H,0EEH,0FEH,0FEH,0D6H,0C6H,0C6H,000H ; D_4D M
     C6 00
1CDE C6 E6 F6 DE CE C6     DB   0C6H,0E6H,0F6H,0DEH,0CEH,0C6H,0C6H,000H ; D_4E N
     C6 00
1CE6 38 6C C6 C6 C6 6C     DB   038H,06CH,0C6H,0C6H,0C6H,06CH,038H,000H ; D_4F O
     38 00

1CEE FC 66 66 7C 60 60     DB   0FCH,066H,066H,07CH,060H,060H,0F0H,000H ; D_50 P
     F0 00
1CF6 78 CC CC CC DC 78     DB   078H,0CCH,0CCH,0CCH,0DCH,078H,01CH,000H ; D_51 Q
     1C 00
1CFE FC 66 66 7C 6C 66     DB   0FCH,066H,066H,07CH,06CH,066H,0E6H,000H ; D_52 R
     E6 00
1D06 78 CC E0 70 1C CC     DB   078H,0CCH,0E0H,070H,01CH,0CCH,078H,000H ; D_53 S
     78 00
1D0E FC B4 30 30 30 30     DB   0FCH,0B4H,030H,030H,030H,030H,078H,000H ; D_54 T
     78 00
1D16 CC CC CC CC CC CC     DB   0CCH,0CCH,0CCH,0CCH,0CCH,0CCH,0FCH,000H ; D_55 U
     FC 00
1D1E CC CC CC CC CC 78     DB   0CCH,0CCH,0CCH,0CCH,0CCH,078H,030H,000H ; D_56 V
     30 00
1D26 C6 C6 C6 D6 FE EE     DB   0C6H,0C6H,0C6H,0D6H,0FEH,0EEH,0C6H,000H ; D_57 W
     C6 00
1D2E C6 C6 6C 38 38 6C     DB   0C6H,0C6H,06CH,038H,038H,06CH,0C6H,000H ; D_58 X
     C6 00
1D36 CC CC CC 78 30 30     DB   0CCH,0CCH,0CCH,078H,030H,030H,078H,000H ; D_59 Y
     78 00
1D3E FE C6 8C 18 32 66     DB   0FEH,0C6H,08CH,018H,032H,066H,0FEH,000H ; D_5A Z
     FE 00
1D46 78 60 60 60 60 60     DB   078H,060H,060H,060H,060H,060H,078H,000H ; D_5B [  LEFT BRACKET
     78 00
1D4E C0 60 30 18 0C 06     DB   0C0H,060H,030H,018H,00CH,006H,002H,000H ; D_5C \  BACKSLASH
     02 00
1D56 78 18 18 18 18 18     DB   078H,018H,018H,018H,018H,018H,078H,000H ; D_5D ]  RIGHT BRACKET
     78 00
1D5E 10 38 6C C6 00 00     DB   010H,038H,06CH,0C6H,000H,000H,000H,000H ; D_5E ^  CIRCUMFLEX
     00 00
1D66 00 00 00 00 00 00     DB   000H,000H,000H,000H,000H,000H,000H,0FFH ; D_5F _  UNDERSCORE
     00 FF

1D6E 30 30 18 00 00 00     DB   030H,030H,018H,000H,000H,000H,000H,000H ; D_60 `  APOSTROPHE REV
     00 00
1D76 00 00 78 0C 7C CC     DB   000H,000H,078H,00CH,07CH,0CCH,076H,000H ; D_61 a
     76 00
1D7E E0 60 60 7C 66 66     DB   0E0H,060H,060H,07CH,066H,066H,0DCH,000H ; D_62 b
     DC 00
1D86 00 00 78 CC C0 CC     DB   000H,000H,078H,0CCH,0C0H,0CCH,078H,000H ; D_63 c
     78 00
1D8E 1C 0C 0C 7C CC CC     DB   01CH,00CH,00CH,07CH,0CCH,0CCH,076H,000H ; D_64 d
     76 00
1D96 00 00 78 CC FC C0     DB   000H,000H,078H,0CCH,0FCH,0C0H,078H,000H ; D_65 e
     78 00
1D9E 38 6C 60 F0 60 60     DB   038H,06CH,060H,0F0H,060H,060H,0F0H,000H ; D_66 f
     F0 00
1DA6 00 00 76 CC CC 7C     DB   000H,000H,076H,0CCH,0CCH,07CH,00CH,0F8H ; D_67 g
     0C F8
1DAE E0 60 6C 76 66 66     DB   0E0H,060H,06CH,076H,066H,066H,0E6H,000H ; D_68 h
     E6 00
1DB6 30 00 70 30 30 30     DB   030H,000H,070H,030H,030H,030H,078H,000H ; D_69 i
     78 00
1DBE 0C 00 0C 0C 0C CC     DB   00CH,000H,00CH,00CH,00CH,0CCH,0CCH,078H ; D_6A j
     CC 78
1DC6 E0 60 66 6C 78 6C     DB   0E0H,060H,066H,06CH,078H,06CH,0E6H,000H ; D_6B k
```

SECTION 5

```
           E6 00
     1DCE  70 30 30 30 30 30      DB    070H,030H,030H,030H,030H,030H,078H,000H ; D_6C l
           78 00
     1DD6  00 00 CC FE FE D6      DB    000H,000H,0CCH,0FEH,0FEH,0D6H,0C6H,000H ; D_6D m
           C6 00
     1DDE  00 00 F8 CC CC CC      DB    000H,000H,0F8H,0CCH,0CCH,0CCH,0CCH,000H ; D_6E n
           CC 00
     1DE6  00 00 78 CC CC CC      DB    000H,000H,078H,0CCH,0CCH,0CCH,078H,000H ; D_6F o
           78 00

     1DEE  00 00 DC 66 66 7C      DB    000H,000H,0DCH,066H,066H,07CH,060H,0F0H ; D_70 p
           60 F0
     1DF6  00 00 76 CC CC 7C      DB    000H,000H,076H,0CCH,0CCH,07CH,00CH,01EH ; D_71 q
           0C 1E
     1DFE  00 00 DC 76 66 60      DB    000H,000H,0DCH,076H,066H,060H,0F0H,000H ; D_72 r
           F0 00
     1E06  00 00 7C C0 78 0C      DB    000H,000H,07CH,0C0H,078H,00CH,0F8H,000H ; D_73 s
           F8 00
     1E0E  10 30 7C 30 30 34      DB    010H,030H,07CH,030H,030H,034H,018H,000H ; D_74 t
           18 00
     1E16  00 00 CC CC CC CC      DB    000H,000H,0CCH,0CCH,0CCH,0CCH,076H,000H ; D_75 u
           76 00
     1E1E  00 00 CC CC CC 78      DB    000H,000H,0CCH,0CCH,0CCH,078H,030H,000H ; D_76 v
           30 00
     1E26  00 00 C6 D6 FE FE      DB    000H,000H,0C6H,0D6H,0FEH,0FEH,06CH,000H ; D_77 w
           6C 00
     1E2E  00 00 C6 6C 38 6C      DB    000H,000H,0C6H,06CH,038H,06CH,0C6H,000H ; D_78 x
           C6 00
     1E36  00 00 CC CC CC 7C      DB    000H,000H,0CCH,0CCH,0CCH,07CH,00CH,0F8H ; D_79 y
           0C F8
     1E3E  00 00 FC 98 30 64      DB    000H,000H,0FCH,098H,030H,064H,0FCH,000H ; D_7A z
           FC 00
     1E46  1C 30 30 E0 30 30      DB    01CH,030H,030H,0E0H,030H,030H,01CH,000H ; D_7B {  LEFT BRACE
           1C 00
     1E4E  18 18 18 00 18 18      DB    018H,018H,018H,000H,018H,018H,018H,000H ; D_7C |  BROKEN STROKE
           18 00
     1E56  E0 30 30 1C 30 30      DB    0E0H,030H,030H,01CH,030H,030H,0E0H,000H ; D_7D }  RIGHT BRACE
           E0 00
     1E5E  76 DC 00 00 00 00      DB    076H,0DCH,000H,000H,000H,000H,000H,000H ; D_7E ~  TILDE
           00 00
     1E66  00 10 38 6C C6 C6      DB    000H,010H,038H,06CH,0C6H,0C6H,0FEH,000H ; D_7F    DELTA
           FE 00
```

```
                           PAGE
                           ;--- 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 BELOW) :
                           ;-----------------------------------------------------
                                  ASSUME  CS:CODE,DS:DATA
                           ;      ORG     0FE6EH
 1E6E                             ORG     01E6EH
 1E6E                      TIME_OF_DAY:
 1E6E E9 0995 R                   JMP     TIME_OF_DAY_1

                           ;      ORG     0FEA5H
 1EA5                             ORG     01EA5H
 1EA5                      TIMER_INT:
 1EA5 E9 1A1A R                   JMP     TIMER_INT_1

                           ;---------------------------------------------
                           ; THESE ARE THE VECTORS WHICH ARE MOVED INTO  :
                           ; THE 8086 INTERRUPT AREA DURING POWER ON.    :
                           ; ONLY THE OFFSETS ARE DISPLAYED HERE, CODE   :
                           ; SEGMENT WILL BE ADDED FOR ALL OF THEM, EXCEPT :
                           ; WHERE NOTED.                                :
                           ;---------------------------------------------
                                  ASSUME  CS:CODE
                           ;      ORG     0FEF3H
 1EF3                             ORG     01EF3H
 1EF3                      VECTOR_TABLE   LABEL   WORD    ; VECTOR TABLE VALUES FOR POST TESTS
 1EF3 1EA5 R                      DW      OFFSET TIMER_INT ; INT 08H - HARDWARE TIMER 0     IRQ 0
 1EF5 0987 R                      DW      OFFSET KB_INT    ; INT 09H - KEYBOARD             IRQ 1
 1EF7 1F23 R                      DW      OFFSET  D11      ; INT 0AH                        IRQ 2
 1EF9 1F23 R                      DW      OFFSET  D11      ; INT 0BH -                      IRQ 3
 1EFB 1F23 R                      DW      OFFSET  D11      ; INT 0CH -                      IRQ 4
 1EFD 1F23 R                      DW      OFFSET  D11      ; INT 0DH -                      IRQ 5
 1EFF 0F57 R                      DW      OFFSET DISK_INT  ; INT 0EH - DISKETTE             IRQ 6
 1F01 1F23 R                      DW      OFFSET  D11      ; INT 0FH -                      IRQ 7

                           ;----- SOFTWARE INTERRUPTS ( BIOS CALLS AND POINTERS )

 1F03 1065 R                      DW      OFFSET VIDEO_IO        ; INT 10H -- VIDEO DISPLAY
 1F05 184D R                      DW      OFFSET EQUIPMENT       ; INT 11H -- GET EQUIPMENT FLAG WORD
 1F07 1841 R                      DW      OFFSET MEMORY_SIZE_DET ; INT 12H -- GET REAL MODE MEMORY SIZE
 1F09 0C59 R                      DW      OFFSET DISKETTE_IO     ; INT 13H -- DISKETTE
 1F0B 0739 R                      DW      OFFSET RS232_IO        ; INT 14H -- COMMUNICATION ADAPTER
 1F0D 1859 R                      DW      OFFSET CASSETTE_IO     ; INT 15H -- EXPANDED BIOS FUNCTION CALL
 1F0F 082E R                      DW      OFFSET KEYBOARD_IO     ; INT 16H -- KEYBOARD INPUT
 1F11 0FD2 R                      DW      OFFSET PRINTER_IO      ; INT 17H -- PRINTER OUTPUT
 1F13 0000                        DW      00000H                 ; INT 18H -- 0F600H INSERTED FOR BASIC
 1F15 06F2 R                      DW      OFFSET BOOT_STRAP      ; INT 19H -- BOOT FROM SYSTEM MEDIA
 1F17 1E6E R                      DW      OFFSET TIME_OF_DAY     ; INT 1AH -- TIME OF DAY
 1F19 1F49 R                      DW      OFFSET DUMMY_RETURN    ; INT 1BH -- KEYBOARD BREAK ADDRESS
 1F1B 1F49 R                      DW      OFFSET DUMMY_RETURN    ; INT 1CH -- TIMER BREAK ADDRESS
 1F1D 10A4 R                      DW      OFFSET VIDEO_PARMS     ; INT 1DH -- VIDEO PARAMETERS
 1F1F 0FC7 R                      DW      OFFSET DISK_BASE       ; INT 1EH -- DISKETTE PARAMETERS
 1F21 0000                        DW      00000H                 ; INT 1FH -- POINTER TO VIDEO EXTENSION

                           ;------------------------------------------------------
                           ; 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.                           :
                           ;------------------------------------------------------
 1F23                      D11    PROC    NEAR
                                  ASSUME  DS:DATA
 1F23 1E                          PUSH    DS
 1F24 E8 1A12 R                   CALL    DDS
 1F27 50                          PUSH    AX                ; SAVE REG AX CONTENTS
 1F28 B0 0B                       MOV     AL,0BH            ; READ IN-SERVICE REG
 1F2A E6 20                       OUT     INTA00,AL         ; (FIND OUT WHAT LEVEL BEING
 1F2C 90                          NOP                       ; SERVICED)
 1F2D E4 20                       IN      AL,INTA00         ; GET LEVEL
 1F2F 8A E0                       MOV     AH,AL             ; SAVE IT
 1F31 0A C4                       OR      AL,AH             ; 00? (NO HARDWARE ISR ACTIVE)
 1F33 75 04                       JNZ     HW_INT
 1F35 B4 FF                       MOV     AH,0FFH
 1F37 EB 0A                       JMP     SHORT SET_INTR_FLAG ; SET FLAG TO FF IF NON-HDWARE
 1F39                      HW_INT:
 1F39 E4 21                       IN      AL,INTA01         ; GET MASK VALUE
 1F3B 0A C4                       OR      AL,AH             ; MASK OFF LVL BEING SERVICED
 1F3D E6 21                       OUT     INTA01,AL
 1F3F B0 20                       MOV     AL,EOI
 1F41 E6 20                       OUT     INTA00,AL
 1F43                      SET_INTR_FLAG:
 1F43 88 26 006B R                MOV     @INTR_FLAG,AH     ; SET FLAG
 1F47 58                          POP     AX                ; RESTORE REG AX CONTENTS
 1F48 1F                          POP     DS
 1F49                      DUMMY_RETURN:                    ; NEED IRET FOR VECTOR TABLE
 1F49 CF                          IRET
 1F4A                      D11    ENDP

                           ;---------------------------------------------
                           ; DUMMY RETURN FOR ADDRESS COMPATIBILITY      :
                           ;---------------------------------------------
                           ;      ORG     0FF53H
 1F53                             ORG     01F53H
 1F53 CF                          IRET
```

SECTION 5

```
                                PAGE
                                ;--- INT  05 H -----------------------------------------------------------
                                ; PRINT_SCREEN                                                           :
                                ;       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 WHILE THIS ROUTINE IS PRINTING IT WILL BE IGNORED.   :
                                ;       THE BASE PRINTERS STATUS IS CHECKED FOR NOT BUSY AND NOT OUT OF   :
                                ;       PAPER.  AN INITIAL STATUS ERROR WILL ABEND THE PRINT REQUEST.     :
                                ;       ADDRESS  0050:0000  CONTAINS THE STATUS OF THE PRINT SCREEN:      :
                                ;                                                                         :
                                ;       50:0     = 0      PRINT SCREEN HAS NOT BEEN CALLED OR UPON RETURN :
                                ;                          FROM A CALL THIS INDICATES A SUCCESSFUL OPERATION.  :
                                ;                = 1      PRINT SCREEN IS IN PROGRESS - IGNORE THIS REQUEST.  :
                                ;                = 255    ERROR ENCOUNTERED DURING PRINTING.              :
                                ;-------------------------------------------------------------------------
                                ;       ORG     0FF54H
 1F54                                   ORG     01F54H

 1F54                           PRINT_SCREEN_1  PROC    FAR
                                                                ; DELAY INTERRUPT ENABLE TILL FLAG SET
 1F54 1E                                PUSH    DS              ; USE 0040:0100 FOR STATUS AREA STORAGE
 1F55 E8 1A12 R                         CALL    DDS             ; GET STATUS_BYTE DATA SEGMENT
 1F58 80 3E 0100 R 01                   CMP     @STATUS_BYTE,1  ; SEE IF PRINT ALREADY IN PROGRESS
 1F5D 74 7C                             JE      PR190           ; EXIT IF PRINT ALREADY IN PROGRESS
 1F5F C6 06 0100 R 01                   MOV     @STATUS_BYTE,1  ; INDICATE PRINT NOW IN PROGRESS
 1F64 FB                                STI                     ; MUST RUN WITH INTERRUPTS ENABLED
 1F65 50                                PUSH    AX              ; SAVE WORK REGISTERS
 1F66 53                                PUSH    BX
 1F67 51                                PUSH    CX
 1F68 52                                PUSH    DX
 1F69 B4 0F                             MOV     AH,0FH          ; WILL REQUEST THE CURRENT SCREEN MODE
 1F6B CD 10                             INT     10H             ;       (AL)= MODE
                                                                ;       (AH)= NUMBER COLUMNS/LINE
                                                                ;       (BH)= VISUAL PAGE
 1F6D 8A CC                             MOV     CL,AH           ; WILL MAKE USE OF (CX) REGISTER TO
 1F6F 8A 2E 0084 R                      MOV     CH,@ROWS        ;  CONTROL ROWS ON SCREEN & COLUMNS
 1F73 FE C5                             INC     CH              ; ADJUST ROWS ON DISPLAY COUNT
                                                                ;       (CL)= NUMBER COLUMNS/LINE
                                                                ;       (CH)= NUMBER OF ROWS ON DISPLAY
                                        ;--------------------------------------------------------------
                                        ;       AT THIS POINT WE KNOW THE COLUMNS/LINE COUNT IS IN (CL) :
                                        ;       AND THE NUMBER OF ROWS ON THE DISPLAY IS IN (CH).       :
                                        ;       THE PAGE IF APPLICABLE IS IN (BH).  THE STACK HAS       :
                                        ;       (DS),(AX),(BX),(CX),(DX)  PUSHED.                       :
                                        ;--------------------------------------------------------------
 1F75 33 D2                             XOR     DX,DX           ; FIRST PRINTER
 1F77 B4 02                             MOV     AH,02H          ; SET PRINTER STATUS REQUEST COMMAND
 1F79 CD 17                             INT     17H             ; REQUEST CURRENT PRINTER STATUS
 1F7B 80 F4 80                          XOR     AH,080H         ; CHECK FOR PRINTER BUSY (NOT CONNECTED)
 1F7E F6 C4 A0                          TEST    AH,0A0H         ;  OR  OUT OF PAPER
 1F81 75 4E                             JNZ     PR170           ; ERROR EXIT IF PRINTER STATUS ERROR

 1F83 E8 1FDD R                         CALL    CRLF            ; CARRIAGE RETURN LINE FEED TO PRINTER
 1F86 51                                PUSH    CX              ; SAVE SCREEN BOUNDS
 1F87 B4 03                             MOV     AH,03H          ; NOW READ THE CURRENT CURSOR POSITION
 1F89 CD 10                             INT     10H             ; AND RESTORE AT END OF ROUTINE
 1F8B 59                                POP     CX              ; RECALL SCREEN BOUNDS
 1F8C 52                                PUSH    DX              ; PRESERVE THE ORIGINAL POSITION
 1F8D 33 D2                             XOR     DX,DX           ; INITIAL CURSOR (0,0) AND FIRST PRINTER
                                        ;--------------------------------------------------------------
                                        ;       THIS LOOP IS TO READ EACH CURSOR POSITION FROM THE     :
                                        ;       SCREEN AND PRINT IT.  (BH)= VISUAL PAGE  (CH)= ROWS    :
                                        ;--------------------------------------------------------------
 1F8F                           PR110:
 1F8F B4 02                             MOV     AH,02H          ; INDICATE CURSOR SET REQUEST
 1F91 CD 10                             INT     10H             ; NEW CURSOR POSITION ESTABLISHED
 1F93 B4 08                             MOV     AH,08H          ; INDICATE READ CHARACTER FROM DISPLAY
 1F95 CD 10                             INT     10H             ; CHARACTER NOW IN (AL)
 1F97 0A C0                             OR      AL,AL           ; SEE IF VALID CHAR
 1F99 75 02                             JNZ     PR120           ; JUMP IF VALID CHAR
 1F9B B0 20                             MOV     AL,' '          ; ELSE MAKE IT A BLANK
 1F9D                           PR120:
 1F9D 52                                PUSH    DX              ; SAVE CURSOR POSITION
 1F9E 33 D2                             XOR     DX,DX           ; INDICATE FIRST PRINTER (DX= 0)
 1FA0 32 E4                             XOR     AH,AH           ; INDICATE PRINT CHARACTER IN (AL)
 1FA2 CD 17                             INT     17H             ; PRINT THE CHARACTER
 1FA4 5A                                POP     DX              ; RECALL CURSOR POSITION
 1FA5 F6 C4 29                          TEST    AH,29H          ; TEST FOR PRINTER ERROR
 1FA8 75 22                             JNZ     PR160           ; EXIT IF ERROR DETECTED
 1FAA FE C2                             INC     DL              ; ADVANCE TO NEXT COLUMN
 1FAC 3A CA                             CMP     CL,DL           ; SEE IF AT END OF LINE
 1FAE 75 DF                             JNZ     PR110           ; IF NOT LOOP FOR NEXT COLUMN
 1FB0 32 D2                             XOR     DL,DL           ; BACK TO COLUMN 0
 1FB2 8A E2                             MOV     AH,DL           ; (AH)=0
 1FB4 52                                PUSH    DX              ; SAVE NEW CURSOR POSITION
 1FB5 E8 1FDD R                         CALL    CRLF            ; LINE FEED CARRIAGE RETURN
 1FB8 5A                                POP     DX              ; RECALL CURSOR POSITION
 1FB9 FE C6                             INC     DH              ; ADVANCE TO NEXT LINE
 1FBB 3A EE                             CMP     CH,DH           ; FINISHED?
 1FBD 75 D0                             JNZ     PR110           ; IF NOT LOOP FOR NEXT LINE

 1FBF 5A                                POP     DX              ; GET CURSOR POSITION
 1FC0 B4 02                             MOV     AH,02H          ; INDICATE REQUEST CURSOR SET
 1FC2 CD 10                             INT     10H             ; CURSOR POSITION RESTORED
 1FC4 FA                                CLI                     ; BLOCK INTERRUPTS TILL STACK CLEARED
 1FC5 C6 06 0100 R 00                   MOV     @STATUS_BYTE,0  ; MOVE OK RESULTS FLAG TO STATUS_BYTE
 1FCA EB 0B                             JMP     SHORT PR180     ; EXIT PRINTER ROUTINE
```

```
                              PAGE
1FCC                          PRI60:                                    ;       ERROR EXIT
1FCC 5A                                 POP     DX                      ; GET CURSOR POSITION
1FCD B4 02                              MOV     AH,02H                  ; INDICATE REQUEST CURSOR SET
1FCF CD 10                              INT     10H                     ; CURSOR POSITION RESTORED
1FD1                          PRI70:
1FD1 FA                                 CLI                             ; BLOCK INTERRUPTS TILL STACK CLEARED
1FD2 C6 06 0100 R FF                    MOV     @STATUS_BYTE,0FFH       ; SET ERROR FLAG
1FD7                          PRI80:
1FD7 5A                                 POP     DX                      ;       EXIT ROUTINE
1FD8 59                                 POP     CX                      ; RESTORE ALL THE REGISTERS USED
1FD9 5B                                 POP     BX
1FDA 58                                 POP     AX
1FDB                          PRI90:                                    ;       ROUTINE BUSY EXIT
1FDB 1F                                 POP     DS
1FDC CF                                 IRET                            ; RETURN WITH INITIAL INTERRUPT MASK
1FDD                          PRINT_SCREEN_1  ENDP

                              ;-----   CARRIAGE RETURN, LINE FEED SUBROUTINE

1FDD                          CRLF      PROC    NEAR
                                                                        ;       SEND CR,LF TO FIRST PRINTER
1FDD 33 D2                              XOR     DX,DX                   ; ASSUME FIRST PRINTER (DX= 0)
1FDF B8 000D                            MOV     AX,CR                   ; GET THE PRINT CHARACTER COMMAND AND
1FE2 CD 17                              INT     17H                     ;   THE CARRIAGE RETURN CHARACTER
1FE4 B8 000A                            MOV     AX,LF                   ; NOW GET THE LINE FEED AND
1FE7 CD 17                              INT     17H                     ;   SEND IT TO THE BIOS PRINTER ROUTINE
1FE9 C3                                 RET
1FEA                          CRLF      ENDP


                              ;--------------------------------
                              ;         POWER ON RESET VECTOR  :
                              ;--------------------------------
                              ;         ORG     0FFF0H
1FF0                                    ORG     01FF0H

                              ;----- POWER ON RESET
1FF0                          P_O_R     LABEL   FAR
1FF0 EA                                 DB      0EAH
1FF1 E05B                               DW      0E05BH                  ; LOW WORD OF RESET
1FF3 F000                               DW      0F000H                  ; SEGMENT

1FF5 30 31 2F 31 30 2F                  DB      '01/10/86'              ; RELEASE MARKER
     38 36

                              ;         ORG     0FFFEH
1FFE                                    ORG     01FFEH
1FFE                          MODEL:
1FFE FB                                 DB      MODEL_BYTE

1FFF                          CODE      ENDS
                                        END
```

SECTION 5

Notes:

System BIOS Listing - 11/8/82

Quick Reference - 64/256K Board

Equates **5-113**

8088 Interupt Locations **5-113**

Stack **5-113**

Data Areas **5-113**

Power-On-Self-Test **5-115**

Boot Strap Loader **5-127**

I/O Support
- RS-232C 5-128
- Keyboard 5-131
- Diskette 5-138
- Printer 5-146
- Display 5-148

System Configuration Analysis
- Memory Size Determine 5-167
- Equipment Determination 5-167

Graphics Character Generator **5-171**

Time of Day 5-172

Print Screen 5-175

Notes:

```
LOC OBJECT               SOURCE (BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82

                         $TITLE(BIOS FOR THE IBM PERSONAL COMPUTER XT)

                         ;----------------------------------------------------------------
                         ;       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.                 :
                         ;----------------------------------------------------------------

                         ;-------------------------------------
                         ;               EQUATES              :
                         ;-------------------------------------
 0060                    PORT_A          EQU     60H             ; 8255 PORT A ADDR
 0061                    PORT_B          EQU     61H             ; 8255 PORT B ADDR
 0062                    PORT_C          EQU     62H             ; 8255 PORT C ADDR
 0063                    CMD_PORT        EQU     63H
 0020                    INTA00          EQU     20H             ; 8259 PORT
 0021                    INTA01          EQU     21H             ; 8259 PORT
 0020                    EOI             EQU     20H
 0040                    TIMER           EQU     40H
 0043                    TIM_CTL         EQU     43H             ; 8253 TIMER CONTROL PORT ADDR
 0040                    TIMER0          EQU     40H             ; 8253 TIMER/CNTER 0 PORT ADDR
 0001                    TMINT           EQU     01              ; TIMER 0 INTR RECVD MASK
 0008                    DMA08           EQU     08              ; DMA STATUS REG PORT ADDR
 0000                    DMA             EQU     00              ; DMA CH.0 ADDR. REG PORT ADDR
 0540                    MAX_PERIOD      EQU     540H
 0410                    MIN_PERIOD      EQU     410H
 0060                    KBD_IN          EQU     60H             ; KEYBOARD DATA IN ADDR PORT
 0002                    KBDINT          EQU     02              ; KEYBOARD INTR MASK
 0060                    KB_DATA         EQU     60H             ; KEYBOARD SCAN CODE PORT
 0061                    KB_CTL          EQU     61H             ; CONTROL BITS FOR KEYBOARD SENSE DATA

                         ;-------------------------------------
                         ;       8088 INTERRUPT LOCATIONS      :
                         ;-------------------------------------

----                     ABS0     SEGMENT AT 0
0000                     STG_LOC0        LABEL   BYTE
0008                             ORG     2*4
0008                     NMI_PTR         LABEL    WORD
0014                             ORG     5*4
0014                     INT5_PTR        LABEL   WORD
0020                             ORG     8*4
0020                     INT_ADDR        LABEL   WORD
0020                     INT_PTR         LABEL   DWORD
0040                             ORG     10H*4
0040                     VIDEO_INT       LABEL   WORD
0074                             ORG     1DH*4
0074                     PARM_PTR        LABEL   DWORD           ; POINTER TO VIDEO PARMS
0060                             ORG     18H*4
0060                     BASIC_PTR       LABEL   WORD            ; ENTRY POINT FOR CASSETTE BASIC
0078                             ORG     01EH*4                  ; INTERRUPT 1EH
0078                     DISK_POINTER    LABEL   DWORD
007C                             ORG     01FH*4                  ; LOCATION OF POINTER
007C                     EXT_PTR LABEL   DWORD                   ; POINTER TO EXTENSION
0400                             ORG     400H
0400                     DATA_AREA       LABEL   BYTE            ; ABSOLUTE LOCATION OF DATA SEGMENT
0400                     DATA_WORD       LABEL   WORD
0500                             ORG     0500H
0500                     MFG_TEST_RTN    LABEL   FAR
7C00                             ORG     7C00H
7C00                     BOOT_LOCN       LABEL   FAR
----                     ABS0    ENDS

                         ;---------------------------------------------
                         ; STACK -- USED DURING INITIALIZATION ONLY     :
                         ;---------------------------------------------

----                     STACK   SEGMENT AT 30H
0000 (128                        DW      128 DUP(?)
      ????
     )
0100                     TOS     LABEL   WORD
----                     STACK   ENDS

                         ;-------------------------------------
                         ;       ROM BIOS DATA AREAS           :
                         ;-------------------------------------

----                     DATA    SEGMENT AT 40H
0000 (4                  RS232_BASE      DW      4 DUP(?)        ; ADDRESSES OF RS232 ADAPTERS
      ????
     )
0008 (4                  PRINTER_BASE    DW      4 DUP(?)        ; ADDRESSES OF PRINTERS
      ????
     )
0010 ????                EQUIP_FLAG      DW      ?               ; INSTALLED HARDWARE
0012 ??                  MFG_TST         DB      ?               ; INITIALIZATION FLAG
0013 ????                MEMORY_SIZE     DW      ?               ; MEMORY SIZE IN K BYTES
0015 ??                  MFG_ERR_FLAG    DB      ?               ; SCRATCHPAD FOR MANUFACTURING
0016 ??                                  DB      ?               ; ERROR CODES

                         ;-------------------------------------
                         ;       KEYBOARD DATA AREAS           :
                         ;-------------------------------------

0017 ??                  KB_FLAG         DB      ?

                         ;----- SHIFT FLAG EQUATES WITHIN KB_FLAG

 0080                    INS_STATE       EQU     80H             ; INSERT STATE IS ACTIVE
 0040                    CAPS_STATE      EQU     40H             ; CAPS LOCK STATE HAS BEEN TOGGLED
 0020                    NUM_STATE       EQU     20H             ; NUM LOCK STATE HAS BEEN TOGGLED
 0010                    SCROLL_STATE    EQU     10H             ; SCROLL LOCK STATE HAS BEEN TOGGLED
 0008                    ALT_SHIFT       EQU     08H             ; ALTERNATE SHIFT KEY DEPRESSED
 0004                    CTL_SHIFT       EQU     04H             ; CONTROL SHIFT KEY DEPRESSED
 0002                    LEFT_SHIFT      EQU     02H             ; LEFT SHIFT KEY DEPRESSED
 0001                    RIGHT_SHIFT     EQU     01H             ; RIGHT SHIFT KEY DEPRESSED
```

```
LOC OBJECT                LINE  SOURCE (BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82

0018 ??                    KB_FLAG_1       DB      ?                ; SECOND BYTE OF KEYBOARD STATUS

  0080                     INS_SHIFT       EQU     80H              ; INSERT KEY IS DEPRESSED
  0040                     CAPS_SHIFT      EQU     40H              ; CAPS LOCK KEY IS DEPRESSED
  0020                     NUM_SHIFT       EQU     20H              ; NUM LOCK KEY IS DEPRESSED
  0010                     SCROLL_SHIFT    EQU     10H              ; SCROLL LOCK KEY IS DEPRESSED
  0008                     HOLD_STATE      EQU     08H              ; SUSPEND KEY HAS BEEN TOGGLED

0019 ??                    ALT_INPUT       DB      ?                ; STORAGE FOR ALTERNATE KEYPAD ENTRY
001A ????                  BUFFER_HEAD     DW      ?                ; POINTER TO HEAD OF KEYBOARD BUFFER
001C ????                  BUFFER_TAIL     DW      ?                ; POINTER TO TAIL OF KEYBOARD BUFFER
001E (16                   KB_BUFFER       DW      16 DUP(?)        ; ROOM FOR 15 ENTRIES
      ????
      )
003E                       KB_BUFFER_END   LABEL   WORD

                           ;------ HEAD = TAIL INDICATES THAT THE BUFFER IS EMPTY

  0045                     NUM_KEY         EQU     69               ; SCAN CODE FOR NUMBER LOCK
  0046                     SCROLL_KEY      EQU     70               ; SCROLL LOCK KEY
  0038                     ALT_KEY         EQU     56               ; ALTERNATE SHIFT KEY SCAN CODE
  001D                     CTL_KEY         EQU     29               ; SCAN CODE FOR CONTROL KEY
  003A                     CAPS_KEY        EQU     58               ; SCAN CODE FOR SHIFT LOCK
  002A                     LEFT_KEY        EQU     42               ; SCAN CODE FOR LEFT SHIFT
  0036                     RIGHT_KEY       EQU     54               ; SCAN CODE FOR RIGHT SHIFT
  0052                     INS_KEY         EQU     82               ; SCAN CODE FOR INSERT KEY
  0053                     DEL_KEY         EQU     83               ; SCAN CODE FOR DELETE KEY

                           ;----------------------------------------
                           ;        DISKETTE DATA AREAS            :
                           ;----------------------------------------
003E ??                    SEEK_STATUS     DB      ?                ; DRIVE RECALIBRATION STATUS
                                                                    ; BIT 3-0 = DRIVE 3-0 NEEDS RECAL
                                                                    ; BEFORE NEXT SEEK IF BIT IS = 0

  0080                     INT_FLAG        EQU     080H             ; INTERRUPT OCCURRENCE FLAG
003F ??                    MOTOR_STATUS    DB      ?                ; MOTOR STATUS
                                                                    ; BIT 3-0 = DRIVE 3-0 IS CURRENTLY
                                                                    ;   RUNNING
                                                                    ; BIT 7 = CURRENT OPERATION IS A WRITE,
                                                                    ;   REQUIRES DELAY

0040 ??                    MOTOR_COUNT     DB      ?                ; TIME OUT COUNTER FOR DRIVE TURN OFF
  0025                     MOTOR_WAIT      EQU     37               ; 2 SECS OF COUNTS FOR MOTOR TURN OFF

0041 ??                    DISKETTE_STATUS DB      ?                ; RETURN CODE STATUS BYTE
  0080                     TIME_OUT        EQU     80H              ; ATTACHMENT FAILED TO RESPOND
  0040                     BAD_SEEK        EQU     40H              ; SEEK OPERATION FAILED
  0020                     BAD_NEC         EQU     20H              ; NEC CONTROLLER HAS FAILED
  0010                     BAD_CRC         EQU     10H              ; BAD CRC ON DISKETTE READ
  0009                     DMA_BOUNDARY    EQU     09H              ; ATTEMPT TO DMA ACROSS 64K BOUNDARY
  0008                     BAD_DMA         EQU     08H              ; DMA OVERRUN ON OPERATION
  0004                     RECORD_NOT_FND  EQU     04H              ; REQUESTED SECTOR NOT FOUND
  0003                     WRITE_PROTECT   EQU     03H              ; WRITE ATTEMPTED ON WRITE PROT DISK
  0002                     BAD_ADDR_MARK   EQU     02H              ; ADDRESS MARK NOT FOUND
  0001                     BAD_CMD         EQU     01H              ; BAD COMMAND PASSED TO DISKETTE I/O

0042 (7                    NEC_STATUS      DB      7 DUP(?)         ; STATUS BYTES FROM NEC
      ??
      )

                           ;----------------------------------------
                           ;        VIDEO DISPLAY DATA AREA        :
                           ;----------------------------------------
0049 ??                    CRT_MODE        DB      ?                ; CURRENT CRT MODE
004A ????                  CRT_COLS        DW      ?                ; NUMBER OF COLUMNS ON SCREEN
004C ????                  CRT_LEN         DW      ?                ; LENGTH OF REGEN IN BYTES
004E ????                  CRT_START       DW      ?                ; STARTING ADDRESS IN REGEN BUFFER
0050 (8                    CURSOR_POSN     DW      8 DUP(?)         ; CURSOR FOR EACH OF UP TO 8 PAGES
      ????
      )
0060 ????                  CURSOR_MODE     DW      ?                ; CURRENT CURSOR MODE SETTING
0062 ??                    ACTIVE_PAGE     DB      ?                ; CURRENT PAGE BEING DISPLAYED
0063 ????                  ADDR_6845       DW      ?                ; BASE ADDRESS FOR ACTIVE DISPLAY CARD
0065 ??                    CRT_MODE_SET    DB      ?                ; CURRENT SETTING OF THE 3X8 REGISTER
0066 ??                    CRT_PALETTE     DB      ?                ; CURRENT PALETTE SETTING COLOR CARD

                           ;----------------------------------------
                           ;        POST DATA AREA                 :
                           ;----------------------------------------
0067 ????                  IO_ROM_INIT     DW      ?                ; PNTR TO OPTIONAL I/O ROM INIT ROUTINE
0069 ????                  IO_ROM_SEG      DW      ?                ; POINTER TO IO ROM SEGMENT
006B ??                    INTR_FLAG       DB      ?                ; FLAG TO INDICATE AN INTERRUPT HAPPEND

                           ;----------------------------------------
                           ;        TIMER DATA AREA                :
                           ;----------------------------------------
006C ????                  TIMER_LOW       DW      ?                ; LOW WORD OF TIMER COUNT
006E ????                  TIMER_HIGH      DW      ?                ; HIGH WORD OF TIMER COUNT
0070 ??                    TIMER_OFL       DB      ?                ; TIMER HAS ROLLED OVER SINCE LAST READ
                           ; COUNTS_SEC    EQU     18
                           ; COUNTS_MIN    EQU     1092
                           ; COUNTS_HOUR   EQU     65543
                           ; COUNTS_DAY    EQU     1573040 = 1800B0H

                           ;----------------------------------------
                           ;        SYSTEM DATA AREA               :
                           ;----------------------------------------
0071 ??                    BIOS_BREAK      DB      ?                ; BIT 7=1 IF BREAK KEY HAS BEEN HIT
0072 ????                  RESET_FLAG      DW      ?                ; WORD=1234H IF KEYBOARD RESET UNDERWAY
                           ;----------------------------------------
                           ;        FIXED DISK DATA AREAS          :
                           ;----------------------------------------
0074 ????                                  DW      ?
0076 ????                                  DW      ?
                           ;--------------------------------------------------------
                           ;        PRINTER AND RS232 TIME-OUT VARIABLES          :
                           ;--------------------------------------------------------
0078 (4                    PRINT_TIM_OUT   DB      4 DUP(?)
      ??
      )
007C (4                    RS232_TIM_OUT   DB      4 DUP(?)
      ??
      )
```

```
                        ;----------------------------------------
                        ;       ADDITIONAL KEYBOARD DATA AREA   :
                        ;----------------------------------------
0080 ????               BUFFER_START    DW      ?
0082 ????               BUFFER_END      DW      ?
----                    DATA    ENDS
                        ;----------------------------------------
                        ;       EXTRA DATA AREA                 :
                        ;----------------------------------------
----                    XXDATA  SEGMENT AT 50H
0000 ??                 STATUS_BYTE     DB      ?
----                    XXDATA  ENDS
                        ;----------------------------------------
                        ;       VIDEO DISPLAY BUFFER            :
                        ;----------------------------------------
----                    VIDEO_RAM       SEGMENT AT 0B800H
0000                    REGEN   LABEL   BYTE
0000                    REGENW  LABEL   WORD
0000 (16384                     DB      16384 DUP(?)
     ??
     )
----                    VIDEO_RAM       ENDS
                        ;----------------------------------------
                        ;       ROM RESIDENT CODE               :
                        ;----------------------------------------
----                    CODE    SEGMENT AT 0F000H
0000 (57344                     DB      57344 DUP(?)            ; FILL LOWEST 56K
     ??
     )

E000 31353031353132             DB      '1501512 COPR. IBM 1982'        ; COPYRIGHT NOTICE
     20434F50522E20
     49424D20313938
     32



                        ;----------------------------------------------
                        ;       INITIAL RELIABILITY TESTS -- PHASE 1   :
                        ;----------------------------------------------

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

                        ;----------------------------------------
                        ;       DATA DEFINITIONS                :
                        ;----------------------------------------

E016 D7E0               C1      DW      C11                     ; RETURN ADDRESS
E018 7EE1               C2      DW      C24                     ; RETURN ADDRESS FOR DUMMY STACK

E01A 204B42204F4B       F3B     DB      ' KB OK',13             ; KB FOR MEMORY SIZE
E020 0D

                        ;---------------------------------------------------------------
                        ;       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 JUST BELOW :
                        ;       THE TEST CODE. THIS ROUTINE ASSUMES THAT THE FRIST 2   :
                        ;       BYTES TRANSFERED CONTAIN THE COUNT OF BYTES TO BE LOADED :
                        ;       (BYTE 1=COUNT LOW, BYTE 2=COUNT HI.)                   :
                        ;---------------------------------------------------------------

                        ;----- FIRST, GET THE COUNT

E021                    MFG_BOOT:
E021 E8131A                     CALL    SP_TEST                 ; GET COUNT LOW
E024 8AFB                       MOV     BH,BL                   ; SAVE IT
E026 E80E1A                     CALL    SP_TEST                 ; GET COUNT HI
E029 8AEB                       MOV     CH,BL
E02B 8ACF                       MOV     CL,BH                   ; CX NOW HAS COUNT
E02D FC                         CLD                             ; SET DIR. FLAG TO INCRIMENT
E02E FA                         CLI
E02F BF0005                     MOV     DI,0500H                ; SET TARGET OFFSET (DS=0000)
E032 B0FD                       MOV     AL,0FDH                 ; UNMASK K/B INTERRUPT
E034 E621                       OUT     INTA01,AL
E036 B00A                       MOV     AL,0AH                  ; SEND READ INT. REQUEST REG. CMD
E038 E620                       OUT     INTA00,AL
E03A BA6100                     MOV     DX,61H                  ; SET UP PORT B ADDRESS
E03D BBCC4C                     MOV     BX,4CCCH                ; CONTROL BITS FOR PORT B
E040 B402                       MOV     AH,02H                  ; K/B REQUEST PENDING MASK
E042                    TST:
E042 8AC3                       MOV     AL,BL
E044 EE                         OUT     DX,AL                   ; TOGGLE K/B CLOCK
E045 8AC7                       MOV     AL,BH
E047 EE                         OUT     DX,AL
E048 4A                         DEC     DX                      ; POINT DX AT ADDR. 60 (KB DATA)
E049                    TST1:
E049 E420                       IN      AL,INTA00               ; GET IRR REG
E04B 22C4                       AND     AL,AH                   ; KB REQUEST PENDING?
E04D 74FA                       JZ      TST1                    ; LOOP TILL DATA PRESENT
E04F EC                         IN      AL,DX                   ; GET DATA
E050 AA                         STOSB                           ; STORE IT
E051 42                         INC     DX                      ; POINT DX BACK AT PORT B (61)
E052 E2EE                       LOOP    TST                     ; LOOP TILL ALL BYTES READ

E054 EA00050000                 JMP     MFG_TEST_RTN            ; FAR JUMP TO CODE THAT WAS JUST
                                                                ; LOADED
```

SECTION 5

```
                ;----------------------------------------
                ;       8088 PROCESSOR TEST            :
                ; DESCRIPTION                          :
                ;       VERIFY 8088 FLAGS, REGISTERS   :
                ;       AND CONDITIONAL JUMPS          :
                ;----------------------------------------
                        ASSUME  CS:CODE,DS:NOTHING,ES:NOTHING,SS:NOTHING
E05B                    ORG     0E05BH
E05B            RESET   LABEL   FAR
E05B FA         START:  CLI                     ; DISABLE INTERRUPTS
E05C B4D5               MOV     AH,0D5H         ; SET SF, CF, ZF, AND AF FLAGS ON
E05E 9E                 SAHF
E05F 734C               JNC     ERR01           ; GO TO ERR ROUTINE IF CF NOT SET
E061 754A               JNZ     ERR01           ; GO TO ERR ROUTINE IF ZF NOT SET
E063 7B48               JNP     ERR01           ; GO TO ERR ROUTINE IF PF NOT SET
E065 7946               JNS     ERR01           ; GO TO ERR ROUTINE IF SF NOT SET
E067 9F                 LAHF                    ; LOAD FLAG IMAGE TO AH
E068 B105               MOV     CL,5            ; LOAD CNT REG WITH SHIFT CNT
E06A D2EC               SHR     AH,CL           ; SHIFT AF INTO CARRY BIT POS
E06C 733F               JNC     ERR01           ; GO TO ERR ROUTINE IF AF NOT SET
E06E B040               MOV     AL,40H          ; SET THE OF FLAG ON
E070 D0E0               SHL     AL,1            ; SETUP FOR TESTING
E072 7139               JNO     ERR01           ; GO TO ERR ROUTINE IF OF NOT SET
E074 32E4               XOR     AH,AH           ; SET AH = 0
E076 9E                 SAHF                    ; CLEAR SF, CF, ZF, AND PF
E077 7634               JBE     ERR01           ; GO TO ERR ROUTINE IF CF ON
                                                ; GO TO ERR ROUTINE IF ZF ON
E079 7832               JS      ERR01           ; GO TO ERR ROUTINE IF SF ON
E07B 7A30               JP      ERR01           ; GO TO ERR ROUTINE IF PF ON
E07D 9F                 LAHF                    ; LOAD FLAG IMAGE TO AH
E07E B105               MOV     CL,5            ; LOAD CNT REG WITH SHIFT CNT
E080 D2EC               SHR     AH,CL           ; SHIFT 'AF' INTO CARRY BIT POS
E082 7229               JC      ERR01           ; GO TO ERR ROUTINE IF ON
E084 D0E4               SHL     AH,1            ; CHECK THAT 'OF' IS CLEAR
E086 7025               JO      ERR01           ; GO TO ERR ROUTINE IF ON

                ;----- READ/WRITE THE 8088 GENERAL AND SEGMENTATION REGISTERS
                ;       WITH ALL ONE'S AND ZEROES'S.

E088 B8FFFF             MOV     AX,0FFFFH       ; SETUP ONE'S PATTERN IN AX
E08B F9                 STC
E08C 8ED8       C8:     MOV     DS,AX           ; WRITE PATTERN TO ALL REGS
E08E 8CDB               MOV     BX,DS
E090 8EC3               MOV     ES,BX
E092 8CC1               MOV     CX,ES
E094 8ED1               MOV     SS,CX
E096 8CD2               MOV     DX,SS
E098 8BE2               MOV     SP,DX
E09A 8BEC               MOV     BP,SP
E09C 8BF5               MOV     SI,BP
E09E 8BFE               MOV     DI,SI
E0A0 7307               JNC     C9              ; TST1A
E0A2 33C7               XOR     AX,DI           ; PATTERN MAKE IT THRU ALL REGS
E0A4 7507               JNZ     ERR01           ; NO - GO TO ERR ROUTINE
E0A6 F8                 CLC
E0A7 EBE3               JMP     C8
E0A9            C9:                             ; TST1A
E0A9 0BC7               OR      AX,DI           ; ZERO PATTERN MAKE IT THRU?
E0AB 7401               JZ      C10             ; YES - GO TO NEXT TEST
E0AD F4         ERR01:  HLT                     ; HALT SYSTEM
                ;----------------------------------------
                ;       ROS CHECKSUM TEST I            :
                ; DESCRIPTION                          :
                ;       A CHECKSUM IS DONE FOR THE 8K  :
                ;       ROS MODULE CONTAINING POD AND  :
                ;       BIOS.                          :
                ;----------------------------------------
E0AE            C10:
                                                ; ZERO IN AL ALREADY
E0AE E6A0               OUT     0A0H,AL         ; DISABLE NMI INTERRUPTS
E0B0 E683               OUT     83H,AL          ; INITIALZE DMA PAGE REG
E0B2 BAD803             MOV     DX,3D8H
E0B5 EE                 OUT     DX,AL           ; DISABLE COLOR VIDEO
E0B6 FEC0               INC     AL
E0B8 B2B8               MOV     DL,0B8H
E0BA EE                 OUT     DX,AL           ; DISABLE B/W VIDEO,EN HIGH RES
E0BB B089               MOV     AL,89H          ; SET 8255 FOR B,A=OUT, C=IN
E0BD E663               OUT     CMD_PORT,AL
E0BF B0A5               MOV     AL,10100101B
                                                ; ENABLE PARITY CHECKERS AND
E0C1 E661               OUT     PORT_B,AL       ; PULL KB CLOCK HI, TRI-STATE
                                                ; KEYBOARD INPUTS,ENABLE HIGH
                                                ; BANK OF SWITCHES->PORT C(0-3)
E0C3 B001               MOV     AL,01H          ; <><><><><><><><><><><><>
E0C5 E660               OUT     PORT_A,AL       ; <><><>CHECKPOINT 1<><><>
E0C7 8CC8               MOV     AX,CS           ; SETUP SS SEG REG
E0C9 8ED0               MOV     SS,AX
E0CB 8ED8               MOV     DS,AX           ; SET UP DATA SEG TO POINT TO
                                                ; ROM ADDRESS
E0CD FC                 CLD                     ; SET DIRECTION FLAG TO INC.
                        ASSUME  SS:CODE
E0CE BB00E0             MOV     BX,0E000H       ; SETUP STARTING ROS ADDR
E0D1 BC16E0             MOV     SP,OFFSET C1    ; SETUP RETURN ADDRESS
E0D4 E91B18             JMP     ROS_CHECKSUM
E0D7 75D4       C11:    JNE     ERR01           ; HALT SYSTEM IF ERROR
                ;-------------------------------------------------------
                ;       8237 DMA INITIALIZATION CHANNEL REGISTER TEST  :
                ; DESCRIPTION                                          :
                ;       DISABLE THE 8237 DMA CONTROLLER.  VERIFY THAT  :
                ;       TIMER 1 FUNCTIONS OK. WRITE/READ THE CURRENT   :
                ;       ADDRESS AND WORD COUNT REGISTERS FOR ALL       :
                ;       CHANNELS.  INITIALIZE AND START DMA FOR MEMORY :
                ;       REFRESH.                                       :
                ;-------------------------------------------------------

                ;----- DISABLE DMA CONTROLLER

E0D9 B002               MOV     AL,02H          ; <><><><><><><><><><><><>
E0DB E660               OUT     PORT_A,AL       ; <><><>CHECKPOINT 2<><><>
E0DD B004               MOV     AL,04           ; DISABLE DMA CONTROLLER
E0DF E608               OUT     DMA08,AL

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

E0E1 B054               MOV     AL,54H          ; SEL TIMER 1,LSB,MODE 2
E0E3 E643               OUT     TIMER+3,AL
```

```
LOC  OBJECT          LINE SOURCE (BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82

E0E5 8AC1                          MOV     AL,CL                   ; SET INITIAL TIMER CNT TO 0
E0E7 E641                          OUT     TIMER+1,AL
E0E9                          C12:                                 ; TIMER1_BITS_ON
E0E9 B040                          MOV     AL,40H                  ; LATCH_TIMER_1 COUNT
E0EB E643                          OUT     TIMER+3,AL
E0ED 80FBFF                        CMP     BL,0FFH                 ; YES - SEE IF ALL BITS GO OFF
E0F0 7407                          JE      C13                     ; TIMER1_BITS_OFF
E0F2 E441                          IN      AL,TIMER+1              ; READ TIMER_1 COUNT
E0F4 0AD8                          OR      BL,AL                   ; ALL BITS ON IN TIMER
E0F6 E2F1                          LOOP    C12                     ; TIMER1_BITS_ON
E0F8 F4                            HLT                             ; TIMER_1 FAILURE, HALT SYS
E0F9                          C13:                                 ; TIMER1_BITS_OFF
E0F9 8AC3                          MOV     AL,BL                   ; SET TIMER_1 CNT
E0FB 2BC9                          SUB     CX,CX
E0FD E641                          OUT     TIMER+1,AL
E0FF                          C14:                                 ; TIMER_LOOP
E0FF B040                          MOV     AL,40H                  ; LATCH_TIMER 1 COUNT
E101 E643                          OUT     TIMER+3,AL
E103 90                            NOP                             ; DELAY FOR TIMER
E104 90                            NOP
E105 E441                          IN      AL,TIMER+1              ; READ TIMER 1 COUNT
E107 22D8                          AND     BL,AL
E109 7403                          JZ      C15                     ; WRAP_DMA_REG
E10B E2F2                          LOOP    C14                     ; TIMER_LOOP
E10D F4                            HLT                             ; HALT_SYSTEM

                              ;----- INITIALIZE TIMER 1 TO REFRESH MEMORY

E10E B003                     C15: MOV     AL,03H                  ; <><><><><><><><><><><><><>
E110 E660                          OUT     PORT_A,AL               ; <><><>CHECKPOINT 3<><><>
                                                                   ; WRAP_DMA_REG
E112 E60D                          OUT     DMA+0DH,AL              ; SEND_MASTER CLEAR TO DMA

                              ;----- WRAP DMA CHANNELS ADDRESS AND COUNT REGISTERS

E114 B0FF                          MOV     AL,0FFH                 ; WRITE PATTERN FF TO ALL REGS
E116 8AD8                     C16: MOV     BL,AL                   ; SAVE PATTERN FOR COMPARE
E118 8AF8                          MOV     BH,AL
E11A B90800                        MOV     CX,8                    ; SETUP LOOP CNT
E11D BA0000                        MOV     DX,DMA                  ; SETUP I/O PORT ADDR OF REG
E120 EE                       C17: OUT     DX,AL                   ; WRITE PATTERN TO REG, LSB
E121 50                            PUSH    AX                      ; SATISIFY 8237 I/O TIMINGS
E122 EE                            OUT     DX,AL                   ; MSB OF 16 BIT REG
E123 B001                          MOV     AL,01H                  ; AL TO ANOTHER PAT BEFORE RD
E125 EC                            IN      AL,DX                   ; READ 16-BIT DMA CH REG, LSB
E126 8AE0                          MOV     AH,AL                   ; SAVE LSB OF 16-BIT REG
E128 EC                            IN      AL,DX                   ; READ MSB OF DMA CH REG
E129 3BD8                          CMP     BX,AX                   ; PATTERN READ AS WRITTEN?
E12B 7401                          JE      C18                     ; YES - CHECK NEXT REG
E12D F4                            HLT                             ; NO - HALT THE SYSTEM
E12E                          C18:                                 ; NXT_DMA_CH
E12E 42                            INC     DX                      ; SET_I/O_PORT TO NEXT CH REG
E12F E2EF                          LOOP    C17                     ; WRITE PATTERN TO NEXT REG
E131 FEC0                          INC     AL                      ; SET PATTERN TO 0
E133 74E1                          JZ      C16                     ; WRITE TO CHANNEL REGS

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

E135 8EDB                          MOV     DS,BX                   ; SET UP ABS0 INTO DS AND ES
E137 8EC3                          MOV     ES,BX
                                   ASSUME  DS:ABS0,ES:ABS0
E139 B0FF                          MOV     AL,0FFH                 ; SET CNT OF 64K FOR REFRESH
E13B E601                          OUT     DMA+1,AL
E13D 50                            PUSH    AX
E13E E601                          OUT     DMA+1,AL
E140 B058                          MOV     AL,058H                 ; SET DMA MODE,CH 0,RD.,AUOTINT
E142 E60B                          OUT     DMA+0BH,AL              ; WRITE DMA MODE REG
E144 B000                          MOV     AL,0                    ; ENABLE DMA CONTROLLER
E146 8AE8                          MOV     CH,AL                   ; SET COUNT HIGH=00
E148 E608                          OUT     DMA+8,AL                ; SETUP DMA COMMAND REG
E14A 50                            PUSH    AX
E14B E60A                          OUT     DMA+10,AL               ; ENABLE DMA CH 0
E14D B012                          MOV     AL,18                   ; START TIMER 1
E14F E641                          OUT     TIMER+1,AL
E151 B041                          MOV     AL,41H                  ; SET MODE FOR CHANNEL 1
E153 E60B                          OUT     DMA+0BH,AL
E155 50                            PUSH    AX
E156 E408                          IN      AL,DMA+08               ; GET DMA STATUS
E158 2410                          AND     AL,00010000B            ; IS TIMER REQUEST THERE?
E15A 7401                          JZ      C18C                    ; (IT SHOULD'T BE)
E15C F4                            HLT                             ; HALT SYS.(HOT TIMER 1 OUTPUT)
E15D B042                     C18C: MOV    AL,42H                  ; SET MODE FOR CHANNEL 2
E15F E60B                          OUT     DMA+0BH,AL
E161 B043                          MOV     AL,43H                  ; SET MODE FOR CHANNEL 3
E163 E60B                          OUT     DMA+0BH,AL
                              ;--------------------------------------------------
                              ;        BASE 16K READ/WRITE STORAGE TEST        :
                              ; DESCRIPTION                                    :
                              ;        WRITE/READ/VERIFY DATA PATTERNS         :
                              ;        AA,55,FF,01, AND 00 TO 1ST 32K OF       :
                              ;        STORAGE. VERIFY STORAGE ADDRESSABILITY. :
                              ;--------------------------------------------------

                              ;----- DETERMINE MEMORY SIZE AND FILL MEMORY WITH DATA

E165 BA1302                        MOV     DX,0213H                ; ENABLE I/O EXPANSION BOX
E168 B001                          MOV     AL,01H
E16A EE                            OUT     DX,AL

E16B 8B1E7204                      MOV     BX,DATA_WORD[OFFSET RESET_FLAG] ; SAVE 'RESET_FLAG' IN BX
E16F B90020                        MOV     CX,2000H                ; SET FOR 16K WORDS
E172 81FB3412                      CMP     BX,1234H                ; WARM START?
E176 7416                          JE      CLR_STG
E178 BC18E0                        MOV     SP,OFFSET C2
E17B E9F104                        JMP     STGTST_CNT
E17E 7412                     C24: JE      HOW_BIG                 ; STORAGE OK, DETERMINE SIZE
E180 8AD8                          MOV     BL,AL                   ; SAVE FAILING BIT PATTERN
E182 B004                          MOV     AL,04H                  ; <><><><><><><><><><><><><>
E184 E660                     C24A: OUT    PORT_A,AL               ; <><><>CHECKPOINT 4<><><>
E186 2BC9                          SUB     CX,CX                   ; BASE RAM FAILURE - HANG
E188 E2FE                     C24B: LOOP   C24B                    ; FLIPPING BETWEEN 04 AND
E18A 86D8                          XCHG    BL,AL                   ; FAILING BIT PATTERN
E18C EBF6                          JMP     C24A
```

SECTION 5

```
E18E                   CLR_STG:
E18E 2BC0                      SUB     AX,AX                   ; MAKE AX=0000
E190 F3                        REP     STOSW                   ; STORE 8K WORDS OF 0000
E191 AB
E192                   HOW_BIG:
E192 891E7204                  MOV     DATA_WORD[OFFSET RESET_FLAG],BX ; RESTORE RESET FLAG
E196 BA0004                    MOV     DX,0400H                ; SET POINTER TO JUST>16KB
E199 BB1000                    MOV     BX,16                   ; BASIC COUNT OF 16K
E19C                   FILL_LOOP:
E19C 8EC2                      MOV     ES,DX                   ; SET SEG. REG.
E19E 2BFF                      SUB     DI,DI
E1A0 B855AA                    MOV     AX,0AA55H               ; TEST PATTERN
E1A3 8BC8                      MOV     CX,AX                   ; SAVE PATTERN
E1A5 268905                    MOV     ES:[DI],AX              ; SEND PATTERN TO MEM.
E1A8 B00F                      MOV     AL,0FH                  ; PUT SOMETHING IN AL
E1AA 268B05                    MOV     AX,ES:[DI]              ; GET PATTERN
E1AD 33C1                      XOR     AX,CX                   ; COMPARE PATTERNS
E1AF 7511                      JNZ     HOW_BIG_END             ; GO END IF NO COMPARE
E1B1 B90020                    MOV     CX,2000H                ; SET COUNT FOR 8K WORDS
E1B4 F3                        REP     STOSW                   ; FILL 8K WORDS
E1B5 AB
E1B6 81C20004                  ADD     DX,400H                 ; POINT TO NEXT 16KB BLOCK
E1BA 83C310                    ADD     BX,16                   ; BUMP COUNT BY 16KB
E1BD 80FEA0                    CMP     DH,0A0H                 ; TOP OF RAM AREA YET? (A0000)
E1C0 75DA                      JNZ     FILL_LOOP
E1C2                   HOW_BIG_END:
E1C2 891E1304                  MOV     DATA_WORD[OFFSET MEMORY_SIZE],BX        ; SAVE MEMORY SIZE

                       ;----- SETUP STACK SEG AND SP

E1C6 B83000                    MOV     AX,STACK                ; GET STACK VALUE
E1C9 8ED0                      MOV     SS,AX                   ; SET THE STACK UP
E1CB BC0001                    MOV     SP,OFFSET TOS           ; STACK IS READY TO GO
                       ;------------------------------------------------------
                       ;       INITIALIZE THE 8259 INTERRUPT CONTROLLER CHIP   :
                       ;------------------------------------------------------
E1CE B013              C25:    MOV     AL,13H                  ; ICW1 - EDGE, SNGL, ICW4
E1D0 E620                      OUT     INTA00,AL
E1D2 B008                      MOV     AL,8                    ; SETUP ICW2 - INT TYPE 8 (8-F)
E1D4 E621                      OUT     INTA01,AL
E1D6 B009                      MOV     AL,9                    ; SETUP ICW4 - BUFFRD,8086 MODE
E1D8 E621                      OUT     INTA01,AL
E1DA B0FF                      MOV     AL,0FFH                 ; MASK ALL INTS. OFF
E1DC E621                      OUT     INTA01,AL               ; (VIDEO ROUTINE ENABLES INTS.)

                       ;----- SET UP THE INTERRUPT VECTORS TO TEMP INTERRUPT

E1DE 1E                        PUSH    DS
E1DF B92000                    MOV     CX,32                   ; FILL ALL 32 INTERRUPTS
E1E2 2BFF                      SUB     DI,DI                   ; FIRST INTERRUPT LOCATION
E1E4 8EC7                      MOV     ES,DI                   ; SET ES=0000 ALSO
E1E6 B823FF            D3:     MOV     AX,OFFSET D11           ; MOVE ADDR OF INTR PROC TO TBL
E1E9 AB                        STOSW
E1EA 8CC8                      MOV     AX,CS                   ; GET ADDR OF INTR PROC SEG
E1EC AB                        STOSW
E1ED E2F7                      LOOP    D3                      ; VECTBL0

                       ;----- ESTABLISH BIOS SUBROUTINE CALL INTERRUPT VECTORS

E1EF BF4000                    MOV     DI,OFFSET VIDEO_INT     ; SETUP ADDR TO INTR AREA
E1F2 0E                        PUSH    CS
E1F3 1F                        POP     DS                      ; SETUP ADDR OF VECTOR TABLE
E1F4 8CD8                      MOV     AX,DS                   ; SET AX=SEGMENT
E1F6 BE03FF90                  MOV     SI,OFFSET VECTOR_TABLE+16       ; START WITH VIDEO ENTRY
E1FA B91000                    MOV     CX,16
E1FD A5                D3A:    MOVSW                           ; MOVE VECTOR TABLE TO RAM
E1FE 47                        INC     DI                      ; SKIP SEGMENT POINTER
E1FF 47                        INC     DI
E200 E2FB                      LOOP    D3A
                       ;----------------------------------------------
                       ;       DETERMINE CONFIGURATION AND MFG. MODE   :
                       ;----------------------------------------------

E202 1F                        POP     DS
E203 1E                        PUSH    DS                      ; RECOVER DATA SEG
E204 E462                      IN      AL,PORT_C               ; GET SWITCH INFO
E206 240F                      AND     AL,00001111B            ; ISOLATE SWITCHES
E208 8AE0                      MOV     AH,AL                   ; SAVE
E20A B0AD                      MOV     AL,10101101B            ; ENABLE OTHER BANK OF SWS.
E20C E661                      OUT     PORT_B,AL
E20E 90                        NOP
E20F E462                      IN      AL,PORT_C
E211 B104                      MOV     CL,4
E213 D2C0                      ROL     AL,CL                   ; ROTATE TO HIGH NIBBLE
E215 24F0                      AND     AL,11110000B            ; ISOLATE
E217 0AC4                      OR      AL,AH                   ; COMBINE WITH OTHER BANK
E219 2AE4                      SUB     AH,AH
E21B A31004                    MOV     DATA_WORD[OFFSET EQUIP_FLAG],AX ; SAVE SWITCH INFO
E21E B099                      MOV     AL,99H
E220 E663                      OUT     CMD_PORT,AL
E222 E80518                    CALL    KBD_RESET               ; SEE IF MFG. JUMPER IN
E225 80FBAA                    CMP     BL,0AAH                 ; KEYBOARD PRESENT?
E228 7418                      JE      E6
E22A 80FB65                    CMP     BL,065H                 ; LOAD MFG. TEST REQUEST?
E22D 7503                      JNE     D3B
E22F E9EFFD                    JMP     MFG_BOOT                ; GO TO BOOTSTRAP IF SO
E232 B038              D3B:    MOV     AL,38H
E234 E661                      OUT     PORT_B,AL
E236 90                        NOP
E237 90                        NOP
E238 E460                      IN      AL,PORT_A
E23A 24FF                      AND     AL,0FFH                 ; WAS DATA LINE GROUNDED
E23C 7504                      JNZ     E6
E23E FE061204                  INC     DATA_AREA[OFFSET MFG_TST]       ; SET MANUFACTURING TEST FLAG
```

```
                       ;-----------------------------------------------------
                       ;       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                    :
                       ;-----------------------------------------------------
E242                   E6:
E242 A11004                    MOV     AX,DATA_WORD[OFFSET EQUIP_FLAG] ; GET SENSE SWITCH INFO
E245 50                        PUSH    AX                      ; SAVE IT
E246 B030                      MOV     AL,30H
E248 A31004                    MOV     DATA_WORD[OFFSET EQUIP_FLAG],AX
E24B 2AE4                      SUB     AH,AH
E24D CD10                      INT     10H                     ; SEND INIT TO B/W CARD
E24F B020                      MOV     AL,20H
E251 A31004                    MOV     DATA_WORD[OFFSET EQUIP_FLAG],AX
E254 2AE4                      SUB     AH,AH                   ; AND INIT COLOR CARD
E256 CD10                      INT     10H
E258 58                        POP     AX                      ; RECOVER REAL SWITCH INFO
E259 A31004                    MOV     DATA_WORD[OFFSET EQUIP_FLAG],AX ; RESTORE IT
                                                               ; AND CONTINUE
E25C 2430                      AND     AL,30H                  ; ISOLATE VIDEO SWS
E25E 750A                      JNZ     E7                      ; VIDEO SWS SET TO 0?
E260 BF4000                    MOV     DI,OFFSET VIDEO_INT     ; SET INT 10H TO DUMMY
E263 C7054BFF                  MOV     [DI],OFFSET DUMMY_RETURN        ; RETURN IF NO VIDEO CARD
E267 E9A000                    JMP     E18_1                   ; BYPASS VIDEO TEST
E26A                   E7:                                     ; TEST_VIDEO:
E26A 3C30                      CMP     AL,30H                  ; B/W CARD ATTACHED?
E26C 7408                      JE      E8                      ; YES - SET MODE FOR B/W CARD
E26E FEC4                      INC     AH                      ; SET COLOR MODE FOR COLOR CD
E270 3C20                      CMP     AL,20H                  ; 80X25 MODE SELECTED?
E272 7502                      JNE     E8                      ; NO - SET MODE FOR 40X25
E274 B403                      MOV     AH,3                    ; SET MODE FOR 80X25
E276 86E0              E8:     XCHG    AH,AL                   ; SET_MODE:
E278 50                        PUSH    AX                      ; SAVE VIDEO MODE ON STACK
E279 2AE4                      SUB     AH,AH                   ; INITIALIZE TO ALPHANUMERIC MD
E27B CD10                      INT     10H                     ; CALL VIDEO_IO
E27D 58                        POP     AX                      ; RESTORE VIDEO SENSE SWS IN AH
E27E 50                        PUSH    AX                      ; RESAVE VALUE
E27F BB00B0                    MOV     BX,0B000H               ; BEG VIDEO RAM ADDR B/W CD
E282 BAB803                    MOV     DX,3B8H                 ; MODE REG FOR B/W
E285 B90008                    MOV     CX,2048                 ; RAM WORD CNT FOR B/W CD
E288 B001                      MOV     AL,1                    ; SET MODE FOR BW CARD
E28A 80FC30                    CMP     AH,30H                  ; B/W VIDEO CARD ATTACHED?
E28D 7409                      JE      E9                      ; YES - GO TEST VIDEO STG
E28F B7B8                      MOV     BH,0B8H                 ; BEG VIDEO RAM ADDR COLOR CD
E291 BAD803                    MOV     DX,3D8H                 ; MODE REG FOR COLOR CD
E294 B520                      MOV     CH,20H                  ; RAM WORD CNT FOR COLOR CD
E296 FEC8                      DEC     AL                      ; SET MODE TO 0 FOR COLOR CD
E298                   E9:                                     ; TEST_VIDEO_STG:
E298 EE                        OUT     DX,AL                   ; DISABLE VIDEO FOR COLOR CD
E299 813E72043412              CMP     DATA_WORD[OFFSET RESET_FLAG],1234H ; POD INIT BY KBD RESET?
E29F 8EC3                      MOV     ES,BX                   ; POINT ES TO VIDEO RAM STG
E2A1 7407                      JE      E10                     ; YES - SKIP VIDEO RAM TEST
E2A3 8EDB                      MOV     DS,BX                   ; POINT DS TO VIDEO RAM STG
                               ASSUME  DS:NOTHING,ES:NOTHING
E2A5 E8C703                    CALL    STGTST_CNT              ; GO TEST VIDEO R/W STG
E2A8 7546                      JNE     E17                     ; R/W STG FAILURE - BEEP SPK
                       ;-------------------------------------------------
                       ;       SETUP VIDEO DATA ON SCREEN FOR VIDEO     :
                       ;       LINE TEST.                               :
                       ; DESCRIPTION                                    :
                       ;       ENABLE VIDEO SIGNAL AND SET MODE.        :
                       ;       DISPLAY A HORIZONTAL BAR ON SCREEN.      :
                       ;-------------------------------------------------
E2AA                   E10:
E2AA 58                        POP     AX                      ; GET VIDEO SENSE SWS (AH)
E2AB 50                        PUSH    AX                      ; SAVE IT
E2AC B400                      MOV     AH,0                    ; ENABLE VIDEO AND SET MODE
E2AE CD10                      INT     10H                     ; VIDEO
E2B0 B82070                    MOV     AX,7020H                ; WRT BLANKS IN REVERSE VIDEO

                       ;----- UNNATURAL ACT FOR ADDRESS COMPATIBILITY

E2B3 EB11                      JMP     SHORT E10A
E2C3                           ORG     0E2C3H
E2C3 E99915                    JMP     NMI_INT

E2C6                   E10A:
E2C6 2BFF                      SUB     DI,DI                   ; SETUP STARTING LOC
E2C8 B92800                    MOV     CX,40                   ; NO. OF BLANKS TO DISPLAY
E2CB F3                        REP     STOSW                   ; WRITE VIDEO STORAGE
E2CC AB
                       ;----------------------------------------
                       ;       CRT INTERFACE LINES TEST        :
                       ; DESCRIPTION                           :
                       ;       SENSE ON/OFF TRANSITION OF THE  :
                       ;       VIDEO ENABLE AND HORIZONTAL     :
                       ;       SYNC LINES.                     :
                       ;----------------------------------------
E2CD 58                        POP     AX                      ; GET VIDEO SENSE SW INFO
E2CE 50                        PUSH    AX                      ; SAVE IT
E2CF 80FC30                    CMP     AH,30H                  ; B/W CARD ATTACHED?
E2D2 BABA03                    MOV     DX,03BAH                ; SETUP ADDR OF BW STATUS PORT
E2D5 7403                      JE      E11                     ; YES - GO TEST LINES
E2D7 BADA03                    MOV     DX,03DAH                ; COLOR CARD IS ATTACHED
E2DA                   E11:                                    ; LINE_TST:
E2DA B408                      MOV     AH,8
E2DC                   E12:                                    ; OFLOOP_CNT:
E2DC 2BC9                      SUB     CX,CX
E2DE                   E13:
E2DE EC                        IN      AL,DX                   ; READ CRT STATUS PORT
E2DF 22C4                      AND     AL,AH                   ; CHECK VIDEO/HORZ LINE
E2E1 7504                      JNZ     E14                     ; ITS ON - CHECK IF IT GOES OFF
E2E3 E2F9                      LOOP    E13                     ; LOOP TILL ON OR TIMEOUT
E2E5 EB09                      JMP     SHORT E17               ; GO PRINT ERROR MSG
E2E7                   E14:
E2E7 2BC9                      SUB     CX,CX
E2E9                   E15:
E2E9 EC                        IN      AL,DX                   ; READ CRT STATUS PORT
E2EA 22C4                      AND     AL,AH                   ; CHECK VIDEO/HORZ LINE
E2EC 7411                      JZ      E16                     ; ITS ON - CHECK NEXT LINE
E2EE E2F9                      LOOP    E15                     ; LOOP IF OFF TILL IT GOES ON
```

```
E2F0                 E17:                                   ; CRT_ERR:
E2F0 1F                        POP     DS
E2F1 1E                        PUSH    DS
E2F2 C606150006                MOV     DS:MFG_ERR_FLAG,06H      ; <><><>CRT ERR CHKPT. 06<><><>
E2F7 BA0201                    MOV     DX,102H
E2FA E8DB16                    CALL    ERR_BEEP                 ; GO BEEP SPEAKER
E2FD EB06                      JMP     SHORT E18
E2FF                 E16:                                   ; NXT_LINE:
E2FF B103                      MOV     CL,3                     ; GET NEXT BIT TO CHECK
E301 D2EC                      SHR     AH,CL
E303 75D7                      JNZ     E12                      ; GO CHECK HORIZONTAL LINE
E305                 E18:                                   ; DISPLAY_CURSOR:
E305 58                        POP     AX                       ; GET VIDEO SENSE SWS (AH)
E306 B400                      MOV     AH,0                     ; SET MODE AND DISPLAY CURSOR
E308 CD10                      INT     10H                      ; CALL VIDEO I/O PROCEDURE
E30A                 E18_1:
E30A BA00C0                    MOV     DX,0C000H                ; SEE IF ADVANCED VIDEO CARD
E30D                 E18A:
E30D 8EDA                      MOV     DS,DX                    ; IS PRESENT
E30F 2BDB                      SUB     BX,BX
E311 8B07                      MOV     AX,[BX]                  ; GET FIRST 2 LOCATIONS
E313 53                        PUSH    BX
E314 5B                        POP     BX                       ; LET BUS SETTLE
E315 3D55AA                    CMP     AX,0AA55H                ; PRESENT?
E318 7505                      JNZ     E18B                     ; NO? GO LOOK FOR OTHER MODULES
E31A E83616                    CALL    ROM_CHECK                ; GO SCAN MODULE
E31D EB04                      JMP     SHORT E18C
E31F                 E18B:
E31F 81C28000                  ADD     DX,0080H                 ; POINT TO NEXT 2K BLOCK
E323                 E18C:
E323 81FA00C8                  CMP     DX,0C800H                ; TOP OF VIDEO ROM AREA YET?
E327 7CE4                      JL      E18A                     ; GO SCAN FOR ANOTHER MODULE
                     ;--------------------------------------------------------
                     ;         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  DS:ABS0
E329 1F              C21:      POP     DS

                     ;----- TEST THE IMR REGISTER

E32A C606150405      C21A:     MOV     DATA_AREA[OFFSET MFR_ERR_FLAG],05H
                                                                ; <><><><><><><><><><><><>
                                                                ; <><><>CHECKPOINT 5<><><>
E32F B000                      MOV     AL,0                     ; SET IMR TO ZERO
E331 E621                      OUT     INTA01,AL
E333 E421                      IN      AL,INTA01                ; READ IMR
E335 0AC0                      OR      AL,AL                    ; IMR = 0?
E337 751B                      JNZ     D6                       ; GO TO ERR ROUTINE IF NOT 0
E339 B0FF                      MOV     AL,0FFH                  ; DISABLE DEVICE INTERRUPTS
E33B E621                      OUT     INTA01,AL                ; WRITE TO IMR
E33D E421                      IN      AL,INTA01                ; READ IMR
E33F 0401                      ADD     AL,1                     ; ALL IMR BIT ON?
E341 7511                      JNZ     D6                       ; NO - GO TO ERR ROUTINE

                     ;----- CHECK FOR HOT INTERRUPTS

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

E343 A26B04                    MOV     DATA_AREA[OFFSET INTR_FLAG],AL   ; CLEAR INTERRUPT FLAG
E346 FB                        STI                              ; ENABLE EXTERNAL INTERRUPTS
E347 2BC9                      SUB     CX,CX                    ; WAIT 1 SEC FOR ANY INTRS THAT
E349                 D4:
E349 E2FE                      LOOP    D4                       ; MIGHT OCCUR
E34B                 D5:
E34B E2FE                      LOOP    D5
E34D 803E6B0400                CMP     DATA_AREA[OFFSET INTR_FLAG],00H ; DID ANY INTERRUPTS OCCUR?
E352 7409                      JZ      D7                       ; NO - GO TO NEXT TEST
E354                 D6:
E354 BEFFF890                  MOV     SI,OFFSET E0             ; DISPLAY 101 ERROR
E358 E84E16                    CALL    E_MSG
E35B FA                        CLI
E35C F4                        HLT                              ; HALT THE SYSTEM
                     ;--------------------------------------------------------
                     ;         8253 TIMER CHECKOUT                           :
                     ; DESCRIPTION                                           :
                     ;         VERIFY THAT THE SYSTEM TIMER (0) DOESN'T COUNT :
                     ;         TOO FAST OR TOO SLOW.                         :
                     ;--------------------------------------------------------
E35D                 D7:
E35D C606150402                MOV     DATA_AREA[OFFSET MFR_ERR_FLAG],02H
                                                                ; <><><><><><><><><><><><><><>
                                                                ; <><><>TIMER CHECKPOINT (2)<><><>
E362 B0FE                      MOV     AL,0FEH                  ; MASK ALL INTRS EXCEPT LVL 0
E364 E621                      OUT     INTA01,AL                ; WRITE THE 8259 IMR
E366 B010                      MOV     AL,00010000B             ; SEL TIM 0, LSB, MODE 0, BINARY
E368 E643                      OUT     TIM_CTL,AL               ; WRITE TIMER CONTROL MODE REG
E36A B91600                    MOV     CX,16H                   ; SET PGM LOOP CNT
E36D 8AC1                      MOV     AL,CL                    ; SET TIMER 0 CNT REG
E36F E640                      OUT     TIMER0,AL                ; WRITE TIMER 0 CNT REG
E371                 D8:
E371 F6066B0401                TEST    DATA_AREA[OFFSET INTR_FLAG],01H
                                                                ; DID TIMER 0 INTERRUPT OCCUR?
E376 7504                      JNZ     D9                       ; YES - CHECK TIMER OP FOR SLOW TIME
E378 E2F7                      LOOP    D8                       ; WAIT FOR INTR FOR SPECIFIED TIME
E37A EBD8                      JMP     D6                       ; TIMER 0 INTR DIDN'T OCCUR - ERR
E37C                 D9:
E37C B10C                      MOV     CL,12                    ; SET PGM LOOP CNT
E37E B0FF                      MOV     AL,0FFH                  ; WRITE TIMER 0 CNT REG
E380 E640                      OUT     TIMER0,AL
E382 C6066B0400                MOV     DATA_AREA[OFFSET INTR_FLAG],0    ; RESET INTR RECEIVED FLAG
E387 B0FE                      MOV     AL,0FEH                  ; REENABLE TIMER 0 INTERRUTS
E389 E621                      OUT     INTA01,AL
E38B                 D10:
E38B F6066B0401                TEST    DATA_AREA[OFFSET INTR_FLAG],01H ; DID TIMER 0 INTERRUPT OCCUR?
E390 75C2                      JNZ     D6                       ; YES - TIMER CNTING TOO FAST, ERR
E392 E2F7                      LOOP    D10                      ; WAIT FOR INTR FOR SPECIFIED TIME
```

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

E394 B0FF                     MOV     AL,0FFH                 ; DISABLE ALL DEVICE INTERRUPTS
E396 E621                     OUT     INTA01,AL
E398 B036                     MOV     AL,36H                  ; SEL TIM 0,LSB,MSB,MODE 3
E39A E643                     OUT     TIMER+3,AL              ; WRITE TIMER MODE REG
E39C B000                     MOV     AL,0
E39E E640                     OUT     TIMER,AL                ; WRITE LSB TO TIMER 0 REG
E3A0 E640                     OUT     TIMER,AL                ; WRITE MSB TO TIMER 0 REG
                      ;--------------------------------------------------
                      ;       KEYBOARD TEST                               :
                      ; DESCRIPTION                                       :
                      ;       RESET THE KEYBOARD AND CHECK THAT SCAN      :
                      ;       CODE 'AA' IS RETURNED TO THE CPU.           :
                      ;       CHECK FOR STUCK KEYS.                       :
                      ;--------------------------------------------------
E3A2                  TST12:
E3A2 B099                     MOV     AL,99H                  ; SET 8255 MODE A,C=IN B=OUT
E3A4 E663                     OUT     CMD_PORT,AL
E3A6 A01004                   MOV     AL,DATA_AREA[OFFSET EQUIP_FLAG]
E3A9 2401                     AND     AL,01                   ; TEST CHAMBER?
E3AB 7431                     JZ      F7                      ; BYPASS IF SO
E3AD 803E120401               CMP     DATA_AREA[OFFSET MFG_TST],1     ; MANUFACTURING TEST MODE?
E3B2 742A                     JE      F7                      ; YES - SKIP KEYBOARD TEST
E3B4 E87316                   CALL    KBD_RESET               ; ISSUE RESET TO KEYBRD
E3B7 E31E                     JCXZ    F6                      ; PRINT ERR MSG IF NO INTERRUPT
E3B9 B049                     MOV     AL,49H                  ; ENABLE KEYBOARD
E3BB E661                     OUT     PORT_B,AL
E3BD 80FBAA                   CMP     BL,0AAH                 ; SCAN CODE AS EXPECTED?
E3C0 7515                     JNE     F6                      ; NO - DISPLAY ERROR MSG

                      ;----- CHECK FOR STUCK KEYS

E3C2 B0C8                     MOV     AL,0C8H                 ; CLR KBD, SET CLK LINE HIGH
E3C4 E661                     OUT     PORT_B,AL
E3C6 B048                     MOV     AL,48H                  ; ENABLE KBD,CLK IN NEXT BYTE
E3C8 E661                     OUT     PORT_B,AL
E3CA 2BC9                     SUB     CX,CX
E3CC                  F5:                                     ; KBD_WAIT:
E3CC E2FE                     LOOP    F5                      ; DELAY FOR A WHILE
E3CE E460                     IN      AL,KBD_IN               ; CHECK FOR STUCK KEYS
E3D0 3C00                     CMP     AL,0                    ; SCAN CODE = 0?
E3D2 740A                     JE      F7                      ; YES - CONTINUE TESTING
E3D4 E88415                   CALL    XPC_BYTE                ; CONVERT AND PRINT
E3D7                  F6:
E3D7 BE4CEC90                 MOV     SI,OFFSET F1            ; GET MSG ADDR
E3DB E8CB15                   CALL    E_MSG                   ; PRINT MSG ON SCREEN
                      ;--------------------------------------------------
                      ;       SETUP HARDWARE INT. VECTOR TABLE          :
                      ;--------------------------------------------------
E3DE                  F7:
E3DE 1E                       PUSH    DS                      ; SETUP_INT_TABLE:
E3DF 2BC0                     SUB     AX,AX
E3E1 8EC0                     MOV     ES,AX
E3E3 B90800                   MOV     CX,08                   ; GET VECTOR CNT
E3E6 0E                       PUSH    CS                      ; SETUP DS SEG REG
E3E7 1F                       POP     DS
E3E8 BEF3FE90                 MOV     SI,OFFSET VECTOR_TABLE
E3EC BF2000                   MOV     DI,OFFSET INT_PTR
E3EF                  F7A:
E3EF A5                       MOVSW
E3F0 47                       INC     DI                      ; SKIP OVER SEGMENT
E3F1 47                       INC     DI
E3F2 E2FB                     LOOP    F7A
E3F4 1F                       POP     DS

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

E3F5 C70608005FF8             MOV     NMI_PTR,OFFSET NMI_INT  ; NMI INTERRUPT
E3FB C706140054FF             MOV     INT5_PTR,OFFSET PRINT_SCREEN    ; PRINT SCREEN
E401 C706620000F6             MOV     BASIC_PTR+2,0F600H      ; SEGMENT FOR CASSETTE BASIC

                      ;----- SETUP TIMER 0 TO BLINK LED IF MANUFACTURING TEST MODE

E407 803E120401               CMP     DATA_AREA[OFFSET MFG_TST],01H   ; MFG. TEST MODE?
E40C 750A                     JNZ     EXP_IO
E40E C70670003CF9             MOV     WORD PRT(1CH*4),OFFSET BLINK_INT; SETUP TIMER INTR TO BLINK LED
E414 B0FE                     MOV     AL,0FEH                 ; ENABLE TIMER INTERRUPT
E416 E621                     OUT     INTA01,AL
```

SECTION 5

```
LOC OBJECT           LINE    SOURCE (BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82

                     ;----------------------------------------------------------------
                     ; EXPANSION I/O BOX TEST                                         :
                     ;        CHECK TO SEE IF EXPANSION BOX PRESENT - IF INSTALLED,   :
                     ;        TEST DATA AND ADDRESS BUSES TO I/O BOX                  :
                     ;  ERROR='1801'                                                  :
                     ;----------------------------------------------------------------

                     ;----- DETERMINE IF BOX IS PRESENT

E418                 EXP_IO:                                 ; (CARD WAS ENABLED EARLIER)
E418 BA1002                  MOV     DX,0210H                ; CONTROL PORT ADDRESS
E41B B85555                  MOV     AX,5555H                ; SET DATA PATTERN
E41E EE                      OUT     DX,AL
E41F B001                    MOV     AL,01H                  ; MAKE AL DIFFERENT
E421 EC                      IN      AL,DX                   ; RECOVER DATA
E422 3AC4                    CMP     AL,AH                   ; REPLY?
E424 7544                    JNE     E19                     ; NO RESPONSE, GO TO NEXT TEST
E426 F7D0                    NOT     AX                      ; MAKE DATA=AAAA
E428 EE                      OUT     DX,AL
E429 B001                    MOV     AL,01H
E42B EC                      IN      AL,DX                   ; RECOVER DATA
E42C 3AC4                    CMP     AL,AH
E42E 753A                    JNE     E19

                     ;----- CHECK ADDRESS BUS

E430                 EXP2:
E430 BB0100                  MOV     BX,0001H
E433 BA1502                  MOV     DX,0215H                ; LOAD HI ADDR. REG ADDRESS
E436 B91000                  MOV     CX,0016                 ; GO ACROSS 16 BITS
E439                 EXP3:
E439 2E8807                  MOV     CS:[BX],AL              ; WRITE ADDRESS F0000+BX
E43C 90                      NOP
E43D EC                      IN      AL,DX                   ; READ ADDR. HIGH
E43E 3AC7                    CMP     AL,BH
E440 7521                    JNE     EXP_ERR                 ; GO ERROR IF MISCOMPARE
E442 42                      INC     DX                      ; DX=216H (ADDR. LOW REG)
E443 EC                      IN      AL,DX
E444 3AC3                    CMP     AL,BL                   ; COMPARE TO LOW ADDRESS
E446 751B                    JNE     EXP_ERR
E448 4A                      DEC     DX                      ; DX BACK TO 215H
E449 D1E3                    SHL     BX,1
E44B E2EC                    LOOP    EXP3                    ; LOOP TILL '1' WALKS ACROSS BX

                     ;----- CHECK DATA BUS

E44D B90800                  MOV     CX,0008                 ; DO 8 TIMES
E450 B001                    MOV     AL,01
E452 4A                      DEC     DX                      ; MAKE DX=214H (DATA BUS REG)
E453                 EXP4:
E453 8AE0                    MOV     AH,AL                   ; SAVE DATA BUS VALUE
E455 EE                      OUT     DX,AL                   ; SEND VALUE TO REG
E456 B001                    MOV     AL,01H
E458 EC                      IN      AL,DX                   ; RETRIVE VALUE FROM REG
E459 3AC4                    CMP     AL,AH                   ; = TO SAVED VALUE
E45B 7506                    JNE     SHORT EXP_ERR
E45D D0E0                    SHL     AL,1                    ; FORM NEW DATA PATTERN
E45F E2F2                    LOOP    EXP4                    ; LOOP TILL BIT WALKS ACROSS AL
E461 EB07                    JMP     SHORT E19               ; GO ON TO NEXT TEST
E463                 EXP_ERR:
E463 BE0FF990                MOV     SI,OFFSET F3C
E467 E83F15                  CALL    E_MSG
```

```
                      ;-------------------------------------------------------
                      ;         ADDITIONAL READ/WRITE STORAGE TEST            :
                      ; DESCRIPTION                                           :
                      ;         WRITE/READ DATA PATTERNS TO ANY READ/WRITE    :
                      ;         STORAGE AFTER THE FIRST 32K.  STORAGE         :
                      ;         ADDRESSABILITY IS CHECKED.                    :
                      ;-------------------------------------------------------
                                ASSUME  DS:DATA
E46A                  E19:
E46A E8EC15                     CALL    DDS
E46D 1E                         PUSH    DS
E46E                  E20:
E46E 813E72003412               CMP     RESET_FLAG,1234H        ; WARM START?
E474 7503                       JNE     E20A                    ; CONTINUE TEST IF NOT
E476 E99F00                     JMP     ROM_SCAN                ; GO TO NEXT ROUTINE IF SO
E479                  E20A:
E479 B81000                     MOV     AX,16                   ; STARTING AMT. OF MEMORY OK
E47C EB28                       JMP     SHORT PRT_SIZ           ; POST MESSAGE
E47E                  E20B:
E47E 8B1E1300                   MOV     BX,MEMORY_SIZE          ; GET MEM. SIZE WORD
E482 83EB10                     SUB     BX,16                   ; 1ST 16K ALREADY DONE
E485 B104                       MOV     CL,04H
E487 D3EB                       SHR     BX,CL                   ; DIVIDE BY 16
E489 8BCB                       MOV     CX,BX                   ; SAVE COUNT OF 16K BLOCKS
E48B BB0004                     MOV     BX,0400H                ; SET PTR. TO RAM SEGMENT>16K
E48E                  E21:
E48E 8EDB                       MOV     DS,BX                   ; SET SEG. REG
E490 8EC3                       MOV     ES,BX
E492 81C30004                   ADD     BX,0400H                ; POINT TO NEXT 16K
E496 52                         PUSH    DX
E497 51                         PUSH    CX                      ; SAVE WORK REGS
E498 53                         PUSH    BX
E499 50                         PUSH    AX
E49A B90020                     MOV     CX,2000H                ; SET COUNT FOR 8K WORDS
E49D E8CF01                     CALL    STGTST_CNT
E4A0 754C                       JNZ     E21A                    ; GO PRINT ERROR
E4A2 58                         POP     AX                      ; RECOVER TESTED MEM NUMBER
E4A3 051000                     ADD     AX,16
E4A6                  PRT_SIZ:
E4A6 50                         PUSH    AX
E4A7 BB0A00                     MOV     BX,10                   ; SET UP FOR DECIMAL CONVERT
E4AA B90300                     MOV     CX,3                    ; OF 3 NIBBLES
E4AD                  DECIMAL_LOOP:
E4AD 33D2                       XOR     DX,DX
E4AF F7F3                       DIV     BX                      ; DIVIDE BY 10
E4B1 80CA30                     OR      DL,30H                  ; MAKE INTO ASCII
E4B4 52                         PUSH    DX                      ; SAVE
E4B5 E2F6                       LOOP    DECIMAL_LOOP
E4B7 B90300                     MOV     CX,3
E4BA                  PRT_DEC_LOOP:
E4BA 58                         POP     AX                      ; RECOVER A NUMBER
E4BB E8DE14                     CALL    PRT_HEX
E4BE E2FA                       LOOP    PRT_DEC_LOOP
E4C0 B90700                     MOV     CX,7
E4C3 BE1AE0                     MOV     SI,OFFSET F3B           ; PRINT ' KB OK'
E4C6                  KB_LOOP:
E4C6 2E8A04                     MOV     AL,CS:[SI]
E4C9 46                         INC     SI
E4CA E8CF14                     CALL    PRT_HEX
E4CD E2F7                       LOOP    KB_LOOP
E4CF 58                         POP     AX                      ; RECOVER WORK REGS
E4D0 3D1000                     CMP     AX,16                   ; FIRST PASS?
E4D3 74A9                       JE      E20B
E4D5 5B                         POP     BX
E4D6 59                         POP     CX
E4D7 5A                         POP     DX
E4D8 E2B4                       LOOP    E21                     ; LOOP TILL ALL MEM. CHECKED
E4DA B00A                       MOV     AL,10
E4DC E8BD14                     CALL    PRT_HEX                 ; LINE FEED

                      ;----- DMA TC0 SHOULD BE ON BY NOW - SEE IF IT IS

E4DF E408                       IN      AL,DMA+08H
E4E1 2401                       AND     AL,00000001B            ; TC0 STATUS BIT ON?
E4E3 7533                       JNZ     ROM_SCAN                ; GO ON WITH NEXT TEST IF OK
E4E5 1F                         POP     DS
E4E6 C606150003                 MOV     MFG_ERR_FLAG,03H        ; <><><><><><><><><><><><><><>
E4EB E966FE                     JMP     D6                      ; POST 101 ERROR MSG AND HALT

                      ;----- PRINT FAILING ADDRESS AND XOR'ED PATTERN IF DATA COMPARE ERROR

E4EE 8AE8             E21A:     MOV     CH,AL                   ; SAVE FAILING BIT PATTERN
E4F0 B00D                       MOV     AL,13                   ; CARRAGE RETURN
E4F2 E8A714                     CALL    PRT_HEX
E4F5 B00A                       MOV     AL,10
E4F7 E8A214                     CALL    PRT_HEX
E4FA 58                         POP     AX                      ; RECOVER AMT. OF GOOD MEM.
E4FB 83C406                     ADD     SP,6                    ; BALANCE STACK
E4FE 8CDA                       MOV     DX,DS                   ; GET FAILING SEGMENT
E500 1F                         POP     DS
E501 1E                         PUSH    DS
E502 A31300                     MOV     MEMORY_SIZE,AX          ; LOAD MEM. SIZE WORD TO SHOW
                                                                ; HOW MUCH MEM. WORKING
E505 88361500                   MOV     MFG_ERR_FLAG,DH         ; <><><><><><><><><><><><><><>
                                                                ; <><>CHECKPOINTS 08->A0<><>
E509 E8CE1A                     CALL    PRT_SEG                 ; PRINT IT
E50C 8AC5                       MOV     AL,CH                   ; GET FAILING BIT PATTERN
E50E E87A14                     CALL    XPC_BYTE                ; CONVERT AND PRINT CODE
E511 BE04F990                   MOV     SI,OFFSET E1            ; SETUP ADDRESS OF ERROR MSG
E515 E89114                     CALL    E_MSG                   ; PRINT ERROR MSG
```

SECTION 5

```
LOC OBJECT              LINE  SOURCE (BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82

                              ;----------------------------------------------------------------
                              ; CHECK FOR OPTIONAL ROM FROM C8000->F4000 IN 2K BLOCKS         :
                              ;       (A VALID MODULE HAS '55AA' IN THE FIRST 2 LOCATIONS,     :
                              ;       LENGTH INDICATOR (LENGTH/512) IN THE 3D LOCATION AND    :
                              ;       TEST/INIT. CODE STARTING IN THE 4TH LOCATION.)           :
                              ;----------------------------------------------------------------
E518                          ROM_SCAN:
E518 BA00C8                           MOV     DX,0C800H               ; SET BEGINNING ADDRESS
E51B                          ROM_SCAN_1:
E51B 8EDA                             MOV     DS,DX
E51D 2BDB                             SUB     BX,BX                   ; SET BX=0000
E51F 8B07                             MOV     AX,[BX]                 ; GET 1ST WORD FROM MODULE
E521 53                               PUSH    BX
E522 5B                               POP     BX                      ; BUS SETTLING
E523 3D55AA                           CMP     AX,0AA55H               ; = TO ID WORD?
E526 7506                             JNZ     NEXT_ROM                ; PROCEED TO NEXT ROM IF NOT
E528 E82814                           CALL    ROM_CHECK               ; GO CHECK OUT MODULE
E52B EB0590                           JMP     ARE_WE_DONE             ; CHECK FOR END OF ROM SPACE
E52E                          NEXT_ROM:
E52E 81C28000                         ADD     DX,0080H                ; POINT TO NEXT 2K ADDRESS
E532                          ARE_WE_DONE:
E532 81FA00F6                         CMP     DX,0F600H               ; AT F6000 YET?
E536 7CE3                             JL      ROM_SCAN_1              ; GO CHECK ANOTHER ADD. IF NOT
E538 EB0190                           JMP     BASE_ROM_CHK            ; GO CHECK BASIC ROM
                              ;-------------------------------------------------------------------------
                              ; A CHECKSUM IS DONE FOR THE 4 ROS MODULES CONTAINING BASIC CODE          :
                              ;-------------------------------------------------------------------------
E53B                          BASE_ROM_CHK:
E53B B404                             MOV     AH,4                    ; NO. OF ROS MODULES TO CHECK
E53D                          E4:
E53D 2BDB                             SUB     BX,BX                   ; SETUP STARTING ROS ADDR
E53F 8EDA                             MOV     DS,DX
                                                                      ; CHECK ROS
E541 E8AE13                           CALL    ROS_CHECKSUM
E544 7403                             JE      E5                      ; CONTINUE IF OK
E546 E88201                           CALL    ROM_ERR                 ; POST ERROR
E549                          E5:
E549 81C20002                         ADD     DX,0200H                ; POINT TO NEXT 8K MODULE
E54D FECC                             DEC     AH                      ; ANY MORE TO DO?
E54F 75EC                             JNZ     E4                      ; YES - CONTINUE
                              ;----------------------------------------------------------------
                              ;       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.                                          :
                              ;----------------------------------------------------------------
E551                          F9:
E551 1F                               POP     DS
E552 A01000                           MOV     AL,BYTE PTR EQUIP_FLAG  ; DISKETTE PRESENT?
E555 2401                             AND     AL,01H                  ; NO - BYPASS DISKETTE TEST
E557 743E                             JZ      F15
E559                          F10:                                    ; DISK_TEST:
E559 E421                             IN      AL,INTA01
E55B 24BF                             AND     AL,0BFH                 ; ENABLE DISKETTE INTERRUPTS
E55D E621                             OUT     INTA01,AL
E55F B400                             MOV     AH,0                    ; RESET NEC FDC
E561 8AD4                             MOV     DL,AH                   ; SET FOR DRIVE 0
E563 CD13                             INT     13H                     ; VERIFY STATUS AFTER RESET
E565 F6C4FF                           TEST    AH,0FFH                 ; STATUS OK?
E568 7520                             JNZ     F13                     ; NO - FDC FAILED

                              ;----- TURN DRIVE 0 MOTOR ON

E56A BAF203                           MOV     DX,03F2H                ; GET ADDR OF FDC CARD
E56D B01C                             MOV     AL,1CH                  ; TURN MOTOR ON, EN DMA/INT
E56F EE                               OUT     DX,AL                   ; WRITE FDC CONTROL REG
E570 2BC9                             SUB     CX,CX
E572                          F11:                                    ; MOTOR_WAIT:
E572 E2FE                             LOOP    F11                     ; WAIT FOR 1 SECOND
E574                          F12:                                    ; MOTOR_WAIT1:
E574 E2FE                             LOOP    F12
E576 33D2                             XOR     DX,DX                   ; SELECT DRIVE 0
E578 B501                             MOV     CH,1                    ; SELECT TRACK 1
E57A 88163E00                         MOV     SEEK_STATUS,DL
E57E E8FC08                           CALL    SEEK                    ; RECALIBRATE DISKETTE
E581 7207                             JC      F13                     ; GO TO ERR SUBROUTINE IF ERR
E583 B522                             MOV     CH,34                   ; SELECT TRACK 34
E585 E8F508                           CALL    SEEK                    ; SEEK TO TRACK 34
E588 7307                             JNC     F14                     ; OK, TURN MOTOR OFF
E58A                          F13:                                    ; DSK_ERR:
E58A BE52EC90                         MOV     SI,OFFSET F3            ; GET ADDR OF MSG
E58E E81814                           CALL    E_MSG                   ; GO PRINT ERROR MSG

                              ;----- TURN DRIVE 0 MOTOR OFF

E591                          F14:                                    ; DR0_OFF:
E591 B00C                             MOV     AL,0CH                  ; TURN DRIVE 0 MOTOR OFF
E593 BAF203                           MOV     DX,03F2H                ; FDC CTL ADDRESS
E596 EE                               OUT     DX,AL

                              ;----- SETUP PRINTER AND RS232 BASE ADDRESSES IF DEVICE ATTACHED

E597                          F15:
E597 C6066B0000                       MOV     INTR_FLAG,00H           ; SET STRAY INTERRUPT FLAG = 00
E59C BE1E00                           MOV     SI,OFFSET KB_BUFFER     ; SETUP KEYBOARD PARAMETERS
E59F 89361A00                         MOV     BUFFER_HEAD,SI
E5A3 89361C00                         MOV     BUFFER_TAIL,SI
E5A7 89368000                         MOV     BUFFER_START,SI
E5AB 83C620                           ADD     SI,32                   ;DEFAULT BUFFER OF 32 BYTES
E5AE 89368200                         MOV     BUFFER_END,SI
E5B2 BF7800                           MOV     DI,OFFSET PRINT_TIM_OUT ;SET DEFAULT PRINTER TIMEOUT
E5B5 1E                               PUSH    DS
E5B6 07                               POP     ES
E5B7 B81414                           MOV     AX,1414H                ; DEFAULT=20
E5BA AB                               STOSW
E5BB AB                               STOSW
E5BC B80101                           MOV     AX,0101H                ;RS232 DEFAULT=01
E5BF AB                               STOSW
E5C0 AB                               STOSW
E5C1 E421                             IN      AL,INTA01
E5C3 24FC                             AND     AL,0FCH                 ; ENABLE TIMER AND KB INTS
E5C5 E621                             OUT     INTA01,AL
```

```
LOC OBJECT                 LINE  SOURCE (BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82

E5C7 83FD00                              CMP     BP,0000H                ; CHECK FOR BP= NON-ZERO
                                                                         ; (ERROR HAPPENED)
E5CA 7419                                JE      F15A_0                  ; CONTINUE IF NO ERROR
E5CC BA0200                              MOV     DX,2                    ; 2 SHORT BEEPS (ERROR)
E5CF E80614                              CALL    ERR_BEEP
E5D2 BE09E890                            MOV     SI,OFFSET F3D           ; LOAD ERROR MSG
E5D6 E8F113                              CALL    P_MSG
E5D9                             ERR_WAIT:
E5D9 B400                                MOV     AH,00
E5DB CD16                                INT     16H                     ; WAIT FOR 'F1' KEY
E5DD 80FC3B                              CMP     AH,3BH
E5E0 75F7                                JNE     ERR_WAIT
E5E2 EB0E90                              JMP     F15A                    ; BYPASS ERROR
E5E5                             F15A_0:
E5E5 803E120001                          CMP     MFG_TST,1               ; MFG MODE
E5EA 7406                                JE      F15A                    ; BYPASS BEEP
E5EC BA0100                              MOV     DX,1                    ; 1 SHORT BEEP (NO ERRORS)
E5EF E8E613                              CALL    ERR_BEEP
E5F2 A01000                      F15A:   MOV     AL,BYTE PTR EQUIP_FLAG  ; GET SWITCHES
E5F5 2401                                AND     AL,00000001B            ; 'LOOP POST' SWITCH ON
E5F7 7503                                JNZ     F15B                    ; CONTINUE WITH BRING-UP
E5F9 E95FFA                              JMP     START
E5FC 2AE4                        F15B:   SUB     AH,AH
E5FE A04900                              MOV     AL,CRT_MODE
E601 CD10                                INT     10H                     ; CLEAR SCREEN
E603                             F15C:
E603 BDA3F990                            MOV     BP,OFFSET F4            ; PRT_SRC_TBL
E607 BE0000                              MOV     SI,0
E60A                             F16:                                    ; PRT_BASE:
E60A 2E8B5600                            MOV     DX,CS:[BP]              ; GET PRINTER BASE ADDR
E60E B0AA                                MOV     AL,0AAH                 ; WRITE DATA TO PORT A
E610 EE                                  OUT     DX,AL
E611 1E                                  PUSH    DS                      ; BUS SETTLEING
E612 EC                                  IN      AL,DX                   ; READ PORT A
E613 1F                                  POP     DS
E614 3CAA                                CMP     AL,0AAH                 ; DATA PATTERN SAME
E616 7505                                JNE     F17                     ; NO - CHECK NEXT PRT CD
E618 895408                              MOV     PRINTER_BASE[SI],DX     ; YES - STORE PRT BASE ADDR
E61B 46                                  INC     SI                      ; INCREMENT TO NEXT WORD
E61C 46                                  INC     SI
E61D                             F17:
E61D 45                                  INC     BP                      ; POINT TO NEXT BASE ADDR
E61E 45                                  INC     BP
E61F 81FDA9F9                            CMP     BP,OFFSET F4E           ; ALL POSSIBLE ADDRS CHECKED?
E623 75E5                                JNE     F16                     ; PRT_BASE
E625 BB0000                              MOV     BX,0                    ; POINTER TO RS232 TABLE
E628 BAFA03                              MOV     DX,3FAH                 ; CHECK IF RS232 CD 1 ATTCH?
E62B EC                                  IN      AL,DX                   ; READ INTR ID REG
E62C A8F8                                TEST    AL,0F8H
E62E 7506                                JNZ     F18
E630 C707F803                            MOV     RS232_BASE[BX],3F8H     ; SETUP RS232 CD #1 ADDR
E634 43                                  INC     BX
E635 43                                  INC     BX
E636                             F18:
E636 BAFA02                              MOV     DX,2FAH                 ; CHECK IF RS232 CD 2 ATTCH
E639 EC                                  IN      AL,DX                   ; READ INTERRUPT ID REG
E63A A8F8                                TEST    AL,0F8H
E63C 7506                                JNZ     F19                     ; BASE_END
E63E C707F802                            MOV     RS232_BASE[BX],2F8H     ; SETUP RS232 CD #2
E642 43                                  INC     BX
E643 43                                  INC     BX

                                 ;----- SET UP EQUIP FLAG TO INDICATE NUMBER OF PRINTERS AND RS232 CARDS

E644                             F19:                                    ; BASE_END:
E644 8BC6                                MOV     AX,SI                   ; SI HAS 2* NUMBER OF RS232
E646 B103                                MOV     CL,3                    ; SHIFT COUNT
E648 D2C8                                ROR     AL,CL                   ; ROTATE RIGHT 3 POSITIONS
E64A 0AC3                                OR      AL,BL                   ; OR IN THE PRINTER COUNT
E64C A21100                              MOV     BYTE PTR EQUIP_FLAG+1,AL        ; STORE AS SECOND BYTE
E64F BA0102                              MOV     DX,201H
E652 EC                                  IN      AL,DX
E653 90                                  NOP
E654 90                                  NOP
E655 90                                  NOP
E656 A80F                                TEST    AL,0FH
E658 7505                                JNZ     F20                     ; NO_GAME_CARD
E65A 800E110010                          OR      BYTE PTR EQUIP_FLAG+1,16
E65F                             F20:                                    ; NO_GAME_CARD:

                                 ;----- ENABLE NMI INTERRUPTS

E65F E461                                IN      AL,PORT_B               ; RESET CHECK ENABLES
E661 0C30                                OR      AL,30H
E663 E661                                OUT     PORT_B,AL
E665 24CF                                AND     AL,0CFH
E667 E661                                OUT     PORT_B,AL
E669 B080                                MOV     AL,80H                  ; ENABLE NMI INTERRUPTS
E66B E6A0                                OUT     0A0H,AL
E66D                             F21:                                    ; LOAD_BOOT_STRAP:
E66D CD19                                INT     19H                     ; GO TO THE BOOT LOADER
```

SECTION 5

```
                        ;---------------------------------------------------------------
                        ; 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.                    :
                        ;---------------------------------------------------------------

E66F                    STGTST_CNT      PROC    NEAR
E66F FC                         CLD                     ; SET DIR FLAG TO INCREMENT
E670 2BFF                       SUB     DI,DI           ; SET DI=OFFSET 0 REL TO ES REG
E672 2BC0                       SUB     AX,AX           ; SETUP FOR 0->FF PATTERN TEST
E674                    C2_1:
E674 8805                       MOV     [DI],AL         ; ON FIRST BYTE
E676 8A05                       MOV     AL,[DI]
E678 32C4                       XOR     AL,AH           ; O.K.?
E67A 754D                       JNZ     C7              ; GO ERROR IF NOT
E67C FEC4                       INC     AH
E67E 8AC4                       MOV     AL,AH
E680 75F2                       JNZ     C2_1            ; LOOP TILL WRAP THROUGH FF
E682 8BD9                       MOV     BX,CX           ; SAVE WORD COUNT OF BLOCK TO TEST
E684 D1E3                       SHL     BX,1            ; CONVERT TO A BYTE COUNT
E686 B8AAAA                     MOV     AX,0AAAAH       ; GET INITIAL DATA PATTERN TO WRITE
E689 BA55FF                     MOV     DX,0FF55H       ; SETUP OTHER DATA PATTERNS TO USE
E68C F3                         REP     STOSW           ; FILL STORAGE LOCATIONS IN BLOCK
E68D AB
E68E E461                       IN      AL,PORT_B
E690 0C30                       OR      AL,00110000B    ; TOGGLE PARITY CHECK LATCHES
E692 E661                       OUT     PORT_B,AL
E694 90                         NOP
E695 24CF                       AND     AL,11001111B
E697 E661                       OUT     PORT_B,AL
E699                    C3:
E699 4F                         DEC     DI              ; POINT TO LAST BYTE JUST WRITTEN
E69A FD                         STD                     ; SET DIR FLAG TO GO BACKWARDS
E69B                    C4:
E69B 8BF7                       MOV     SI,DI           ; INITIALIZE DESTINATION POINTER
E69D 8BCB                       MOV     CX,BX           ; SETUP BYTE COUNT FOR LOOP
E69F                    C5:                             ;       INNER TEST LOOP
E69F AC                         LODSB                   ; READ OLD TEST BYTE FROM STG [SI]+'
E6A0 32C4                       XOR     AL,AH           ; DATA READ AS EXPECTED ?
E6A2 7525                       JNE     C7              ; NO - GO TO ERROR ROUTINE
E6A4 8AC2                       MOV     AL,DL           ; GET NEXT DATA PATTERN TO WRITE
E6A6 AA                         STOSB                   ; WRITE INTO LOC JUST READ [DI]+'
E6A7 E2F6                       LOOP    C5              ; DECREMENT BYTE COUNT AND LOOP   CX

E6A9 22E4                       AND     AH,AH           ; ENDING ZERO PATTERN WRITTEN TO STG ?
E6AB 7416                       JZ      C6X             ; YES - RETURN TO CALLER WITH AL=0
E6AD 8AE0                       MOV     AH,AL           ; SETUP NEW VALUE FOR COMPARE
E6AF 86F2                       XCHG    DH,DL           ; MOVE NEXT DATA PATTERN TO DL
E6B1 22E4                       AND     AH,AH           ; READING ZERO PATTERN THIS PASS ?
E6B3 7504                       JNZ     C6              ; CONTINUE TEST SEQUENCE TILL ZERO DATA
E6B5 8AD4                       MOV     DL,AH           ; ELSE SET ZERO FOR END READ PATTERN
E6B7 EBE0                       JMP     C3              ; AND MAKE FINAL BACKWARDS PASS
E6B9                    C6:
E6B9 FC                         CLD                     ; SET DIR FLAG TO GO FORWARD
E6BA 47                         INC     DI              ; SET POINTER TO BEG LOCATION
E6BB 74DE                       JZ      C4              ; READ/WRITE FORWARD IN STG
E6BD 4F                         DEC     DI              ; ADJUST POINTER
E6BE BA0100                     MOV     DX,00001H       ; SETUP 01 FOR PARITY BIT AND 00 FOR END
E6C1 EBD6                       JMP     C3              ; READ/WRITE BACKWARD IN STG
E6C3                    C6X:
E6C3 E462                       IN      AL,PORT_C       ; DID A PARITY ERROR OCCUR ?
E6C5 24C0                       AND     AL,0C0H         ; ZERO FLAG WILL BE OFF PARITY ERROR
E6C7 B000                       MOV     AL,000H         ; AL=0 DATA COMPARE OK
E6C9                    C7:
E6C9 FC                         CLD                     ; SET DIRECTION FLAG TO INC
E6CA C3                         RET
                        STGTST_CNT      ENDP
                        ;----------------------------------------------------------------------
                        ;  PRINT ADDRESS AND ERROR MESSAGE FOR ROM CHECKSUM ERRORS            :
                        ;----------------------------------------------------------------------
E6CB                    ROM_ERR PROC    NEAR
E6CB 52                         PUSH    DX              ; SAVE POINTER
E6CC 50                         PUSH    AX
E6CD 8CDA                       MOV     DX,DS           ; GET ADDRESS POINTER
E6CF 2688361500                 MOV     ES:MFG_ERR_FLAG,DH ; <><><><><><><><><><><><><><><><>
                                                        ; <><><>CHECKPOINTS C0->F4<><><>
E6D4 81FA00C8                   CMP     DX,0C800H       ; CRT CARD IN ERROR?
E6D8 7C0D                       JL      ROM_ERR_BEEP    ; GIVE CRT CARD FAIL BEEP
E6DA E8FD18                     CALL    PRT_SEG         ; PRINT SEGEMENT IN ERROR
E6DD BE0AF990                   MOV     SI,OFFSET F3A   ; DISPLAY ERROR MSG
E6E1 E8C512                     CALL    E_MSG
E6E4                    ROM_ERR_END:
E6E4 58                         POP     AX
E6E5 5A                         POP     DX
E6E6 C3                         RET
E6E7                    ROM_ERR_BEEP:
E6E7 BA0201                     MOV     DX,0102H        ; BEEP 1 LONG, 2 SHORT
E6EA E8EB12                     CALL    ERR_BEEP
E6ED EBF5                       JMP     SHORT ROM_ERR_END
                        ROM_ERR ENDP
```

```
                        ;--- INT 19 -------------------------------------------
                        ; BOOT STRAP LOADER                                   :
                        ;       TRACK 0, SECTOR 1 IS READ INTO THE            :
                        ;       BOOT LOCATION (SEGMENT 0, OFFSET 7C00)        :
                        ;       AND CONTROL IS TRANSFERRED THERE.             :
                        ;                                                     :
                        ;       IF THERE IS A HARDWARE ERROR CONTROL IS       :
                        ;       TRANSFERRED TO THE ROM BASIC ENTRY POINT.     :
                        ;------------------------------------------------------
                                ASSUME  CS:CODE,DS:ABS0
E6F2                            ORG     0E6F2H

E6F2                    BOOT_STRAP      PROC    NEAR
E6F2 FB                         STI                             ; ENABLE INTERRUPTS
E6F3 2BC0                       SUB     AX,AX                   ; ESTABLISH ADDRESSING
E6F5 8ED8                       MOV     DS,AX

                        ;----- RESET THE DISK PARAMETER TABLE VECTOR

E6F7 C7067800C7EF               MOV     WORD PTR DISK_POINTER, OFFSET DISK_BASE
E6FD 8C0E7A00                   MOV     WORD PTR DISK_POINTER+2,CS

                        ;----- LOAD SYSTEM FROM DISKETTE -- CX HAS RETRY COUNT

E701 B90400                     MOV     CX,4                    ; SET RETRY COUNT
E704                    H1:                                     ; IPL_SYSTEM
E704 51                         PUSH    CX                      ; SAVE RETRY COUNT
E705 B400                       MOV     AH,0                    ; RESET THE DISKETTE SYSTEM
E707 CD13                       INT     13H                     ; DISKETTE_IO
E709 720F                       JC      H2                      ; IF ERROR, TRY AGAIN
E70B B80102                     MOV     AX,201H                 ; READ IN THE SINGLE SECTOR
E70E 2BD2                       SUB     DX,DX                   ; TO THE BOOT LOCATION
E710 8EC2                       MOV     ES,DX
E712 BB007C                     MOV     BX,OFFSET BOOT_LOCN
                                                                ; DRIVE 0, HEAD 0
E715 B90100                     MOV     CX,1                    ; SECTOR 1, TRACK 0
E718 CD13                       INT     13H                     ; DISKETTE_IO
E71A                    H2:
E71A 59                         POP     CX                      ; RECOVER RETRY COUNT
E71B 7304                       JNC     H4                      ; CF SET BY UNSUCCESSFUL READ
E71D E2E5                       LOOP    H1                      ; DO IT FOR RETRY TIMES

                        ;----- UNABLE TO IPL FROM THE DISKETTE

E71F                    H3:
E71F CD18                       INT     18H                     ; GO TO RESIDENT BASIC

                        ;----- IPL WAS SUCCESSFUL

E721                    H4:
E721 EA007C0000                 JMP     BOOT_LOCN
                        BOOT_STRAP      ENDP
```

SECTION 5

```
                ;-----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 PARAMETERS FOR INITIALIZATION            :
                ;                                                                 :
                ;       7       6       5       4       3       2       1       0 :
                ;       ----- BAUD RATE --      -PARITY--      STOPBIT  --WORD LENGTH-- :
                ;       000 - 110               X0 - NONE       0 - 1    10 - 7 BITS :
                ;       001 - 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 TRANSMIT THE BYTE OF DATA OVER THE LINE. :
                ;                       IF BIT 7 OF AH IS NOT SET, THE REMAINDER OF AH :
                ;                       IS SET AS IN A STATUS REQUEST, REFLECTING THE :
                ;                       CURRENT STATUS OF THE LINE.               :
                ;       (AH)=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 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 = RECEIVED 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 LABEL 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
E729                    ORG     0E729H
E729            A1      LABEL   WORD            ; TABLE OF INIT VALUES
E729 1704               DW      1047            ; 110 BAUD
E72B 0003               DW      768             ; 150
E72D 8001               DW      384             ; 300
E72F C000               DW      192             ; 600
E731 6000               DW      96              ; 1200
E733 3000               DW      48              ; 2400
E735 1800               DW      24              ; 4800
E737 0C00               DW      12              ; 9600

E739            RS232_IO        PROC    FAR

                ;----- VECTOR TO APPROPRIATE ROUTINE

E739 FB                 STI                             ; INTERRUPTS BACK ON
E73A 1E                 PUSH    DS                      ; SAVE SEGMENT
E73B 52                 PUSH    DX
E73C 56                 PUSH    SI
E73D 57                 PUSH    DI
E73E 51                 PUSH    CX
E73F 53                 PUSH    BX
E740 8BF2               MOV     SI,DX                   ; RS232 VALUE TO SI
E742 8BFA               MOV     DI,DX
E744 D1E6               SHL     SI,1                    ; WORD OFFSET
E746 E81013             CALL    DDS
E7 49 8B14              MOV     DX,RS232_BASE[SI]       ; GET BASE ADDRESS
E74B 0BD2               OR      DX,DX                   ; TEST FOR 0 BASE ADDRESS
E74D 7413               JZ      A3                      ; RETURN
E74F 0AE4               OR      AH,AH                   ; TEST FOR (AH)=0
E751 7416               JZ      A4                      ; COMMUN INIT
E753 FECC               DEC     AH                      ; TEST FOR (AH)=1
E755 7445               JZ      A5                      ; SEND AL
E757 FECC               DEC     AH                      ; TEST FOR (AH)=2
E759 746A               JZ      A12                     ; RECEIVE INTO AL
E75B            A2:
E75B FECC               DEC     AH                      ; TEST FOR (AH)=3
E75D 7503               JNZ     A3
E75F E98300             JMP     A18                     ; COMMUNICATION STATUS
E762            A3:                                     ; RETURN FROM RS232
E762 5B                 POP     BX
E763 59                 POP     CX
E764 5F                 POP     DI
E765 5E                 POP     SI
E766 5A                 POP     DX
E767 1F                 POP     DS
E768 CF                 IRET                            ; RETURN TO CALLER, NO ACTION
```

```
                   ;----- INITIALIZE THE COMMUNICATIONS PORT

E769               A4:
E769 8AE0                  MOV     AH,AL                   ; SAVE INIT PARMS IN AH
E76B 83C203                ADD     DX,3                    ; POINT TO 8250 CONTROL REGISTER
E76E B080                  MOV     AL,80H
E770 EE                    OUT     DX,AL                   ; SET DLAB=1

                   ;----- DETERMINE BAUD RATE DIVISOR

E771 8AD4                  MOV     DL,AH                   ; GET PARMS TO DL
E773 B104                  MOV     CL,4
E775 D2C2                  ROL     DL,CL
E777 81E20E00              AND     DX,0EH                  ; ISOLATE THEM
E77B BF29E7                MOV     DI,OFFSET A1            ; BASE OF TABLE
E77E 03FA                  ADD     DI,DX                   ; PUT INTO INDEX REGISTER
E780 8B14                  MOV     DX,RS232_BASE[SI]       ; POINT TO HIGH ORDER OF DIVISOR
E782 42                    INC     DX
E783 2E8A4501              MOV     AL,CS:[DI]+1            ; GET HIGH ORDER OF DIVISOR
E787 EE                    OUT     DX,AL                   ; SET MS OF DIV TO 0
E788 4A                    DEC     DX
E789 2E8A05                MOV     AL,SC:[DI]              ; GET LOW ORDER OF DIVISOR
E78C EE                    OUT     DX,AL                   ; SET LOW OF DIVISOR
E78D 83C203                ADD     DX,3
E790 8AC4                  MOV     AL,AH                   ; GET PARMS BACK
E792 241F                  AND     AL,01FH                 ; STRIP OFF THE BAUD BITS
E794 EE                    OUT     DX,AL                   ; LINE CONTROL TO 8 BITS
E795 4A                    DEC     DX
E796 4A                    DEC     DX
E797 B000                  MOV     AL,0
E799 EE                    OUT     DX,AL                   ; INTERRUPT ENABLES ALL OFF
E79A EB49                  JMP     SHORT A18               ; COM_STATUS

                   ;----- SEND CHARACTER IN (AL) OVER COMMO LINE

E79C               A5:
E79C 50                    PUSH    AX                      ; SAVE CHAR TO SEND
E79D 83C204                ADD     DX,4                    ; MODEM CONTROL REGISTER
E7A0 B003                  MOV     AL,3                    ; DTR AND RTS
E7A2 EE                    OUT     DX,AL                   ; DATA TERMINAL READY, REQUEST TO SEND
E7A3 42                    INC     DX                      ; MODEM STATUS REGISTER
E7A4 42                    INC     DX
E7A5 B730                  MOV     BH,30H                  ; DATA SET READY & CLEAR TO SEND
E7A7 E84800                CALL    WAIT_FOR_STATUS         ; ARE BOTH TRUE
E7AA 7408                  JE      A9                      ; YES, READY TO TRANSMIT CHAR
E7AC               A7:
E7AC 59                    POP     CX
E7AD 8AC1                  MOV     AL,CL                   ; RELOAD DATA BYTE
E7AF               A8:
E7AF 80CC80                OR      AH,80H                  ; INDICATE TIME OUT
E7B2 EBAE                  JMP     A3                      ; RETURN
E7B4               A9:                                     ; CLEAR_TO_SEND
E7B4 4A                    DEC     DX                      ; LINE STATUS REGISTER
E7B5               A10:                                    ; WAIT_SEND
E7B5 B720                  MOV     BH,20H                  ; IS TRANSMITTER READY
E7B7 E83800                CALL    WAIT_FOR_STATUS         ; TEST FOR TRANSMITTER READY
E7BA 75F0                  JNZ     A7                      ; RETURN WITH TIME OUT SET
E7BC               A11:                                    ; OUT_CHAR
E7BC 83EA05                SUB     DX,5                    ; DATA PORT
E7BF 59                    POP     CX                      ; RECOVER IN CX TEMPORARILY
E7C0 8AC1                  MOV     AL,CL                   ; MOVE CHAR TO AL FOR OUT, STATUS IN AH
E7C2 EE                    OUT     DX,AL                   ; OUTPUT CHARACTER
E7C3 EB9D                  JMP     A3                      ; RETURN

                   ;----- RECEIVE CHARACTER FROM COMMO LINE

E7C5               A12:
E7C5 83C204                ADD     DX,4                    ; MODEM CONTROL REGISTER
E7C8 B001                  MOV     AL,1                    ; DATA TERMINAL READY
E7CA EE                    OUT     DX,AL
E7CB 42                    INC     DX                      ; MODEM STATUS REGISTER
E7CC 42                    INC     DX
E7CD               A13:                                    ; WAIT_DSR
E7CD B720                  MOV     BH,20H                  ; DATA SET READY
E7CF E82000                CALL    WAIT_FOR_STATUS         ; TEST FOR DSR
E7D2 75DB                  JNZ     A8                      ; RETURN WITH ERROR
E7D4               A15:                                    ; WAIT_DSR_END
E7D4 4A                    DEC     DX                      ; LINE STATUS REGISTER
E7D5               A16:                                    ; WAIT_RECV
E7D5 B701                  MOV     BH,1                    ; RECEIVE BUFFER FULL
E7D7 E81800                CALL    WAIT_FOR_STATUS         ; TEST FOR REC. BUFF. FULL
E7DA 75D3                  JNZ     A8                      ; SET TIME OUT ERROR
E7DC               A17:                                    ; GET_CHAR
E7DC 80E41E                AND     AH,00011110B            ; TEST FOR ERR CONDITIONS ON RECV CHAR
E7DF 8B14                  MOV     DX,RS232_BASE[SI]       ; DATA PORT
E7E1 EC                    IN      AL,DX                   ; GET CHARACTER FROM LINE
E7E2 E97DFF                JMP     A3                      ; RETURN

                   ;----- COMMO PORT STATUS ROUTINE

E7E5               A18:
E7E5 8B14                  MOV     DX,RS232_BASE[SI]
E7E7 83C205                ADD     DX,5                    ; CONTROL PORT
E7EA EC                    IN      AL,DX                   ; GET LINE CONTROL STATUS
E7EB 8AE0                  MOV     AH,AL                   ; PUT IN AH FOR RETURN
E7ED 42                    INC     DX                      ; POINT TO MODEM STATUS REGISTER
E7EE EC                    IN      AL,DX                   ; GET MODEM CONTROL STATUS
E7EF E970FF                JMP     A3                      ; RETURN
```

SECTION 5

```
LOC OBJECT              LINE  SOURCE (BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82

                              ;----------------------------------------
                              ; 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              :
                              ;----------------------------------------
E7F2                          WAIT_FOR_STATUS PROC     NEAR
E7F2 8A5D7C                           MOV     BL,RS232_TIM_OUT[DI]    ; LOAD OUTER LOOP COUNT
E7F5                          WFS0:
E7F5 2BC9                             SUB     CX,CX
E7F7                          WFS1:
E7F7 EC                               IN      AL,DX                   ; GET STATUS
E7F8 8AE0                             MOV     AH,AL                   ; MOVE TO AH
E7FA 22C7                             AND     AL,BH                   ; ISOLATE BITS TO TEST
E7FC 3AC7                             CMP     AL,BH                   ; EXACTLY = TO MASK
E7FE 7408                             JE      WFS_END                 ; RETURN WITH ZERO FLAG ON
E800 E2F5                             LOOP    WFS1                    ; TRY AGAIN
E802 FECB                             DEC     BL
E804 75EF                             JNZ     WFS0

E806 0AFF                             OR      BH,BH                   ; SET ZERO FLAG OFF
E808                          WFS_END:
E808 C3                               RET
                              WAIT_FOR_STATUS ENDP
                              RS232_IO        ENDP

E809 4552524F522E20           F3D     DB      'ERROR. (RESUME = F1 KEY)',13,10       ; ERROR PROMPT
     28524553554D45
     203D2022463122
     204B455929
E823 0D
E824 0A
```

```
LOC OBJECT                  LINE  SOURCE (BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82

                                  ;---- 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 PRESERVED                                       :
                                  ;---------------------------------------------------------------------
                                          ASSUME  CS:CODE,DS:DATA
E82E                                      ORG     0E82EH
E82E                              KEYBOARD_IO     PROC    FAR
E82E FB                                   STI                             ; INTERRUPTS BACK ON
E82F 1E                                   PUSH    DS                      ; SAVE CURRENT DS
E830 53                                   PUSH    BX                      ; SAVE BX TEMPORARILY
E831 E82512                               CALL    DDS
E834 0AE4                                 OR      AH,AH                   ; AH=0
E836 740A                                 JZ      K1                      ; ASCII_READ
E838 FECC                                 DEC     AH                      ; AH=1
E83A 741E                                 JZ      K2                      ; ASCII_STATUS
E83C FECC                                 DEC     AH                      ; AH=2
E83E 742B                                 JZ      K3                      ; SHIFT_STATUS
E840 EB2C                                 JMP     SHORT INT10_END         ; EXIT

                                  ;----- READ THE KEY TO FIGURE OUT WHAT TO DO

E842                              K1:                                     ; ASCII READ
E842 FB                                   STI                             ; INTERRUPTS BACK ON DURING LOOP
E843 90                                   NOP                             ; ALLOW AN INTERRUPT TO OCCUR
E844 FA                                   CLI                             ; INTERRUPTS BACK OFF
E845 8B1E1A00                             MOV     BX,BUFFER_HEAD          ; GET POINTER TO HEAD OF BUFFER
E849 3B1E1C00                             CMP     BX,BUFFER_TAIL          ; TEST END OF BUFFER
E84D 74F3                                 JZ      K1                      ; LOOP UNTIL SOMETHING IN BUFFER
E84F 8B07                                 MOV     AX,[BX]                 ; GET SCAN CODE AND ASCII CODE
E851 E81D00                               CALL    K4                      ; MOVE POINTER TO NEXT POSITION
E854 891E1A00                             MOV     BUFFER_HEAD,BX          ; STORE VALUE IN VARIABLE
E858 EB14                                 JMP     SHORT INT10_END         ; RETURN

                                  ;----- ASCII STATUS

E85A                              K2:
E85A FA                                   CLI                             ; INTERRUPTS OFF
E85B 8B1E1A00                             MOV     BX,BUFFER_HEAD          ; GET HEAD POINTER
E85F 3B1E1C00                             CMP     BX,BUFFER_TAIL          ; IF EQUAL (Z=1) THEN NOTHING THERE
E863 8B07                                 MOV     AX,[BX]
E865 FB                                   STI                             ; INTERRUPTS BACK ON
E866 5B                                   POP     BX                      ; RECOVER REGISTER
E867 1F                                   POP     DS                      ; RECOVER SEGMENT
E868 CA0200                               RET     2                       ; THROW AWAY FLAGS

                                  ;----- SHIFT STATUS

E86B                              K3:
E86B A01700                               MOV     AL,KB_FLAG              ; GET THE SHIFT STATUS FLAGS
E86E                              INT10_END:
E86E 5B                                   POP     BX                      ; RECOVER REGISTER
E86F 1F                                   POP     DS                      ; RECOVER REGISTERS
E870 CF                                   IRET                            ; RETURN TO CALLER
                                  KEYBOARD_IO     ENDP

                                  ;----- INCREMENT A BUFFER POINTER

E871                              K4      PROC    NEAR
E871 43                                   INC     BX                      ; MOVE TO NEXT WORD IN LIST
E872 43                                   INC     BX
E873 3B1E8200                             CMP     BX,BUFFER_END           ; AT END OF BUFFER?
E877 7504                                 JNE     K5                      ; NO, CONTINUE
E879 8B1E8000                             MOV     BX,BUFFER_START         ; YES, RESET TO BUFFER BEGINNING
E87D                              K5:
E87D C3                                   RET
                                  K4      ENDP

                                  ;----- TABLE OF SHIFT KEYS AND MASK VALUES

E87E                              K6      LABEL   BYTE
E87E 52                                   DB      INS_KEY                 ; INSERT KEY
E87F 3A                                   DB      CAPS_KEY,NUM_KEY,SCROLL_KEY,ALT_KEY,CTL_KEY
E880 45
E881 46
E882 38
E883 1D
E884 2A                                   DB      LEFT_KEY,RIGHT_KEY
E885 36
  0008                            K6L     EQU     $-K6

                                  ;----- SHIFT_MASK_TABLE

E886                              K7      LABEL   BYTE
E886 80                                   DB      INS_SHIFT               ; INSERT MODE SHIFT
E887 40                                   DB      CAPS_SHIFT,NUM_SHIFT,SCROLL_SHIFT,ALT_SHIFT,CTL_SHIFT
E888 20
E889 10
E88A 08
E88B 04
E88C 02                                   DB      LEFT_SHIFT,RIGHT_SHIFT
E88D 01

                                  ;----- SCAN CODE TABLES

E88E 1B                           K8              DB      27,-1,0,-1,-1,-1,30,-1
E88F FF
E890 00
E891 FF
E892 FF
E893 FF
E894 1E
```

SECTION 5

```
E895 FF
E896 FF                 1806                DB      -1,-1,-1,31,-1,127,-1,17
E897 FF
E898 FF
E899 1F
E89A FF
E89B 7F
E89C FF
E89D 11
E89E 17                 1807                DB      23,5,18,20,25,21,9,15
E89F 05
E8A0 12
E8A1 14
E8A2 19
E8A3 15
E8A4 09
E8A5 0F
E8A6 10                 1808                DB      16,27,29,10,-1,1,19
E8A7 1B
E8A8 1D
E8A9 0A
E8AA FF
E8AB 01
E8AC 13
E8AD 04                 1809                DB      4,6,7,8,10,11,12,-1,-1
E8AE 06
E8AF 07
E8B0 08
E8B1 0A
E8B2 0B
E8B3 0C
E8B4 FF
E8B5 FF
E8B6 FF                 1810                DB      -1,-1,28,26,24,3,22,2
E8B7 FF
E8B8 1C
E8B9 1A
E8BA 18
E8BB 03
E8BC 16
E8BD 02
E8BE 0E                 1811                DB      14,13,-1,-1,-1,-1,-1,-1
E8BF 0D
E8C0 FF
E8C1 FF
E8C2 FF
E8C3 FF
E8C4 FF
E8C5 FF
E8C6 20                 1812                DB      ' ',-1
E8C7 FF
                        1813  ;----- CTL TABLE SCAN
E8C8                    1814  K9      LABEL   BYTE
E8C8 5E                 1815                DB      94,95,96,97,98,99,100,101
E8C9 5F
E8CA 60
E8CB 61
E8CC 62
E8CD 63
E8CE 64
E8CF 65
E8D0 66                 1816                DB      102,103,-1,-1,119,-1,132,-1
E8D1 67
E8D2 FF
E8D3 FF
E8D4 77
E8D5 FF
E8D6 84
E8D7 FF
E8D8 73                 1817                DB      115,-1,116,-1,117,-1,118,-1
E8D9 FF
E8DA 74
E8DB FF
E8DC 75
E8DD FF
E8DE 76
E8DF FF
E8E0 FF                 1818                DB      -1
                        1819  ;----- LC TABLE
E8E1                    1820  K10     LABEL   BYTE
E8E1 1B                 1821                DB      01BH,'1234567890-=',08H,09H
E8E2 31323334353637
     3839302D3D
E8EE 08
E8EF 09
E8F0 71776572747975     1822                DB      'qwertyuiop[]',0DH,-1,'asdfghjkl;',027H
     696F705B5D
E8FC 0D
E8FD FF
E8FE 6173646667686A
     6B6C3B
E908 27
E909 60                 1823                DB      60H,-1,5CH,'zxcvbnm,./',-1,'*',-1,' '
E90A FF
E90B 5C
E90C 7A786376626E6D
     2C2E2F
E916 FF
E917 2A
E918 FF
E919 20
E91A FF                 1824                DB      -1
                        1825  ;----- UC TABLE
E91B                    1826  K11     LABEL   BYTE
E91B 1B                 1827                DB      27,'!@#$',37,05EH,'&*()_+',08H,0
E91C 21402324
E920 25
E921 5E
E922 262A28295F2B
E928 08
E929 00
E92A 51574552545955     1828                DB      'QWERTYUIOP{}',0DH,-1,'ASDFGHJKL:"'
     494F507B7D
E936 0D
E937 FF
E938 4153444647484A
```

```
     4B4C3A22
E943 7E                                 DB      07EH,-1,'|ZXCVBNM<>?',-1,0,-1,' ',-1
E944 FF
E945 7C5A584356424E
     4D3C3E3F
E950 FF
E951 00
E952 FF
E953 20
E954 FF
                        ;----- UC TABLE SCAN
E955                    K12     LABEL   BYTE
E955 54                                 DB      84,85,86,87,88,89,90
E956 55
E957 56
E958 57
E959 58
E95A 59
E95B 5A
E95C 5B                                 DB      91,92,93
E95D 5C
E95E 5D
                        ;----- ALT TABLE SCAN
E95F                    K13     LABEL   BYTE
E95F 68                                 DB      104,105,106,107,108
E960 69
E961 6A
E962 6B
E963 6C
E964 6D                                 DB      109,110,111,112,113
E965 6E
E966 6F
E967 70
E968 71
                        ;----- NUM STATE TABLE
E969                    K14     LABEL   BYTE
E969 3738392D343536                     DB      '789-456+1230.'
     2B313233302E
                        ;----- BASE CASE TABLE
E976                    K15     LABEL   BYTE
E976 47                                 DB      71,72,73,-1,75,-1,77
E977 48
E978 49
E979 FF
E97A 4B
E97B FF
E97C 4D
E97D FF                                 DB      -1,79,80,81,82,83
E97E 4F
E97F 50
E980 51
E981 52
E982 53

                        ;----- KEYBOARD INTERRUPT ROUTINE

E987                            ORG     0E987H
E987                    KB_INT  PROC    FAR
E987 FB                         STI                     ; ALLOW FURTHER INTERRUPTS
E988 50                         PUSH    AX
E989 53                         PUSH    BX
E98A 51                         PUSH    CX
E98B 52                         PUSH    DX
E98C 56                         PUSH    SI
E98D 57                         PUSH    DI
E98E 1E                         PUSH    DS
E98F 06                         PUSH    ES
E990 FC                         CLD                     ; FORWARD DIRECTION
E991 E8C510                     CALL    DDS
E994 E460                       IN      AL,KB_DATA      ; READ IN THE CHARACTER
E996 50                         PUSH    AX              ; SAVE IT
E997 E461                       IN      AL,KB_CTL       ; GET THE CONTROL PORT
E999 8AE0                       MOV     AH,AL           ; SAVE VALUE
E99B 0C80                       OR      AL,80H          ; RESET BIT FOR KEYBOARD
E99D E661                       OUT     KB_CTL,AL
E99F 86E0                       XCHG    AH,AL           ; GET BACK ORIGINAL CONTROL
E9A1 E661                       OUT     KB_CTL,AL       ; KB HAS BEEN RESET
E9A3 58                         POP     AX              ; RECOVER SCAN CODE
E9A4 8AE0                       MOV     AH,AL           ; SAVE SCAN CODE IN AH ALSO

                        ;----- TEST FOR OVERRUN SCAN CODE FROM KEYBOARD

E9A6 3CFF                       CMP     AL,0FFH         ; IS THIS AN OVERRUN CHAR
E9A8 7503                       JNZ     K16             ; NO, TEST FOR SHIFT KEY
E9AA E97A02                     JMP     K62             ; BUFFER_FULL_BEEP

                        ;----- TEST FOR SHIFT KEYS

E9AD                    K16:                            ; TEST_SHIFT
E9AD 247F                       AND     AL,07FH         ; TURN OFF THE BREAK BIT
E9AF 0E                         PUSH    CS
E9B0 07                         POP     ES              ; ESTABLISH ADDRESS OF SHIFT TABLE
E9B1 BF7EE8                     MOV     DI,OFFSET K6    ; SHIFT KEY TABLE
E9B4 B90800                     MOV     CX,K6L          ; LENGTH
E9B7 F2                         REPNE   SCASB           ; LOOK THROUGH THE TABLE FOR A MATCH
E9B8 AE
E9B9 8AC4                       MOV     AL,AH           ; RECOVER SCAN CODE
E9BB 7403                       JE      K17             ; JUMP IF MATCH FOUND
E9BD E98500                     JMP     K25             ; IF NO MATCH, THEN SHIFT NOT FOUND

                        ;----- SHIFT KEY FOUND

E9C0 81EF7FE8           K17:    SUB     DI,OFFSET K6+1  ; ADJUST PTR TO SCAN CODE MTCH
E9C4 2E8AA586E8                 MOV     AH,CS:K7[DI]    ; GET MASK INTO AH
E9C9 A880                       TEST    AL,80H          ; TEST FOR BREAK KEY
E9CB 7551                       JNZ     K23             ; BREAK_SHIFT_FOUND

                        ;----- SHIFT MAKE FOUND, DETERMINE SET OR TOGGLE

E9CD 80FC10                     CMP     AH,SCROLL_SHIFT
E9D0 7307                       JAE     K18             ; IF SCROLL SHIFT OR ABOVE, TOGGLE KEY

                        ;----- PLAIN SHIFT KEY, SET SHIFT ON

E9D2 08261700                   OR      KB_FLAG,AH      ; TURN ON SHIFT BIT
E9D6 E98000                     JMP     K26             ; INTERRUPT_RETURN
```

SECTION 5

```
                       ;----- TOGGLED SHIFT KEY, TEST FOR 1ST MAKE OR NOT

E9D9                   K18:                                      ; SHIFT-TOGGLE
E9D9 F606170004                TEST    KB_FLAG, CTL_SHIFT        ; CHECK CTL SHIFT STATE
E9DE 7565                      JNZ     K25                       ; JUMP IF CTL STATE
E9E0 3C52                      CMP     AL, INS_KEY               ; CHECK FOR INSERT KEY
E9E2 7522                      JNZ     K22                       ; JUMP IF NOT INSERT KEY
E9E4 F606170008                TEST    KB_FLAG, ALT_SHIFT        ; CHECK FOR ALTERNATE SHIFT
E9E9 755A                      JNZ     K25                       ; JUMP IF ALTERNATE SHIFT
E9EB F606170020        K19:    TEST    KB_FLAG, NUM_STATE        ; CHECK FOR BASE STATE
E9F0 750D                      JNZ     K21                       ; JUMP IF NUM LOCK IS ON
E9F2 F606170003                TEST    KB_FLAG, LEFT_SHIFT+ RIGHT_SHIFT
E9F7 740D                      JZ      K22                       ; JUMP IF BASE STATE

E9F9                   K20:                                      ; NUMERIC ZERO, NOT INSERT KEY
E9F9 B83052                    MOV     AX, 5230H                 ; PUT OUT AN ASCII ZERO
E9FC E9D601                    JMP     K57                       ; BUFFER_FILL
E9FF                   K21:                                      ; MIGHT BE NUMERIC
E9FF F606170003                TEST    KB_FLAG, LEFT_SHIFT+ RIGHT_SHIFT
EA04 74F3                      JZ      K20                       ; JUMP NUMERIC, NOT INSERT

EA06                   K22:                                      ; SHIFT TOGGLE KEY HIT; PROCESS IT
EA06 84261800                  TEST    AH,KB_FLAG_1              ; IS KEY ALREADY DEPRESSED
EA0A 754D                      JNZ     K26                       ; JUMP IF KEY ALREADY DEPRESSED
EA0C 08261800                  OR      KB_FLAG_1,AH              ; INDICATE THAT THE KEY IS DEPRESSED
EA10 30261700                  XOR     KB_FLAG,AH                ; TOGGLE THE SHIFT STATE
EA14 3C52                      CMP     AL,INS_KEY                ; TEST FOR 1ST MAKE OF INSERT KEY
EA16 7541                      JNE     K26                       ; JUMP IF NOT INSERT KEY
EA18 B80052                    MOV     AX,INS_KEY*256            ; SET SCAN CODE INTO AH, 0 INTO AL
EA1B E9B701                    JMP     K57                       ; PUT INTO OUTPUT BUFFER

                       ;----- BREAK SHIFT FOUND

EA1E                   K23:                                      ; BREAK-SHIFT-FOUND
EA1E 80FC10                    CMP     AH,SCROLL_SHIFT           ; IS THIS A TOGGLE KEY
EA21 731A                      JAE     K24                       ; YES, HANDLE BREAK TOGGLE
EA23 F6D4                      NOT     AH                        ; INVERT MASK
EA25 20261700                  AND     KB_FLAG,AH                ; TURN OFF SHIFT BIT
EA29 3CB8                      CMP     AL,ALT_KEY+80H            ; IS THIS ALTERNATE SHIFT RELEASE
EA2B 752C                      JNE     K26                       ; INTERRUPT_RETURN
                       ;----- ALTERNATE SHIFT KEY RELEASED, GET THE VALUE INTO BUFFER

EA2D A01900                    MOV     AL,ALT_INPUT
EA30 B400                      MOV     AH,0                      ; SCAN CODE OF 0
EA32 88261900                  MOV     ALT_INPUT,AH              ; ZERO OUT THE FIELD
EA36 3C00                      CMP     AL,0                      ; WAS THE INPUT=0
EA38 741F                      JE      K26                       ; INTERRUPT_RETURN
EA3A E9A101                    JMP     K58                       ; IT WASN'T, SO PUT IN BUFFER
EA3D                   K24:                                      ; BREAK-TOGGLE
EA3D F6D4                      NOT     AH                        ; INVERT MASK
EA3F 20261800                  AND     KB_FLAG_1,AH              ; INDICATE NO LONGER DEPRESSED
EA43 EB14                      JMP     SHORT K26                 ; INTERRUPT_RETURN

                       ;----- TEST FOR HOLD STATE

EA45                   K25:                                      ; NO-SHIFT-FOUND
EA45 3C80                      CMP     AL,80H                    ; TEST FOR BREAK KEY
EA47 7310                      JAE     K26                       ; NOTHING FOR BREAK CHARS FROM HERE ON
EA49 F606180008                TEST    KB_FLAG_1,HOLD_STATE      ; ARE WE IN HOLD STATE
EA4E 7417                      JZ      K28                       ; BRANCH AROUND TEST IF NOT
EA50 3C45                      CMP     AL,NUM_KEY
EA52 7405                      JE      K26                       ; CAN'T END HOLD ON NUM_LOCK
EA54 80261800F7                AND     KB_FLAG_1,NOT HOLD_STATE  ; TURN OFF THE HOLD STATE BIT
EA59                   K26:                                      ; INTERRUPT-RETURN
EA59 FA                        CLI                               ; TURN OFF INTERRUPTS
EA5A B020                      MOV     AL,EOI                    ; END OF INTERRUPT COMMAND
EA5C E620                      OUT     020H,AL                   ; SEND COMMAND TO INT CONTROL PORT
EA5E                   K27:                                      ; INTERRUPT-RETURN-NO-EOI
EA5E 07                        POP     ES
EA5F 1F                        POP     DS
EA60 5F                        POP     DI
EA61 5E                        POP     SI
EA62 5A                        POP     DX
EA63 59                        POP     CX
EA64 5B                        POP     BX
EA65 58                        POP     AX                        ; RESTORE STATE
EA66 CF                        IRET                              ; RETURN, INTERRUPTS BACK ON
                                                                 ;  WITH FLAG CHANGE

                       ;----- NOT IN  HOLD STATE, TEST FOR SPECIAL CHARS

EA67                   K28:                                      ; NO-HOLD-STATE
EA67 F606170008                TEST    KB_FLAG,ALT_SHIFT         ; ARE WE IN ALTERNATE SHIFT
EA6C 7503                      JNZ     K29                       ; JUMP IF ALTERNATE SHIFT
EA6E E99100                    JMP     K38                       ; JUMP IF NOT ALTERNATE

                       ;----- TEST FOR RESET KEY SEQUENCE (CTL ALT DEL)

EA71                   K29:                                      ; TEST-RESET
EA71 F606170004                TEST    KB_FLAG,CTL_SHIFT         ; ARE WE IN CONTROL SHIFT ALSO
EA76 7433                      JZ      K31                       ; NO_RESET
EA78 3C53                      CMP     AL,DEL_KEY                ; SHIFT STATE IS THERE, TEST KEY
EA7A 752F                      JNE     K31                       ; NO_RESET

                       ;----- CTL-ALT-DEL HAS BEEN FOUND, DO I/O CLEANUP

EA7C C70672003412              MOV     RESET_FLAG, 1234H         ; SET FLAG FOR RESET FUNCTION
EA82 EA5BE000F0                JMP     RESET                     ; JUMP TO POWER ON DIAGNOSTICS

                       ;----- ALT-INPUT-TABLE
EA87                   K30     LABEL   BYTE
EA87 52                        DB      82,79,80,81,75,76,77
EA88 4F
EA89 50
EA8A 51
EA8B 4B
EA8C 4C
EA8D 4D
EA8E 47                        DB      71,72,73                  ; 10 NUMBERS ON KEYPAD
EA8F 48
EA90 49
                       ;----- SUPER-SHIFT-TABLE
EA91 10                        DB      16,17,18,19,20,21,22,23 ; A-Z TYPEWRITER CHARS
EA92 11
```

```
LOC OBJECT           LINE  SOURCE (BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82

EA93 12
EA94 13
EA95 14
EA96 15
EA97 16
EA98 17
EA99 18                      DB      24,25,30,31,32,33,34,35
EA9A 19
EA9B 1E
EA9C 1F
EA9D 20
EA9E 21
EA9F 22
EAA0 23
EAA1 24                      DB      36,37,38,44,45,46,47,48
EAA2 25
EAA3 26
EAA4 2C
EAA5 2D
EAA6 2E
EAA7 2F
EAA8 30
EAA9 31                      DB      49,50
EAAA 32

                       ;----- IN ALTERNATE SHIFT, RESET NOT FOUND

EAAB                   K31:                                    ; NO-RESET
EAAB 3C39                    CMP     AL,57                     ; TEST FOR SPACE KEY
EAAD 7505                    JNE     K32                       ; NOT THERE
EAAF B020                    MOV     AL,' '                    ; SET SPACE CHAR
EAB1 E92101                  JMP     K57                       ; BUFFER_FILL

                       ;----- LOOK FOR KEY PAD ENTRY

EAB4                   K32:                                    ; ALT-KEY-PAD
EAB4 BF87EA                  MOV     DI,OFFSET K30             ; ALT-INPUT-TABLE
EAB7 B90A00                  MOV     CX,10                     ; LOOK FOR ENTRY USING KEYPAD
EABA F2                      REPNE   SCASB                     ; LOOK FOR MATCH
EABB AE
EABC 7512                    JNE     K33                       ; NO_ALT_KEYPAD
EABE 81EF88EA                SUB     DI,OFFSET K30+1           ; DI NOW HAS ENTRY VALUE
EAC2 A01900                  MOV     AL,ALT_INPUT              ; GET THE CURRENT BYTE
EAC5 B40A                    MOV     AH,10                     ; MULTIPLY BY 10
EAC7 F6E4                    MUL     AH
EAC9 03C7                    ADD     AX,DI                     ; ADD IN THE LATEST ENTRY
EACB A21900                  MOV     ALT_INPUT,AL              ; STORE IT AWAY
EACE EB89                    JMP     K26                       ; THROW AWAY THAT KEYSTROKE

                       ;----- LOOK FOR SUPERSHIFT ENTRY

EAD0                   K33:                                    ; NO-ALT-KEYPAD
EAD0 C606190000              MOV     ALT_INPUT,0               ; ZERO ANY PREVIOUS ENTRY INTO INPUT
EAD5 B91A00                  MOV     CX,26                     ; DI,ES ALREADY POINTING
EAD8 F2                      REPNE   SCASB                     ; LOOK FOR MATCH IN ALPHABET
EAD9 AE
EADA 7505                    JNE     K34                       ; NOT FOUND, FUNCTION KEY OR OTHER
EADC B000                    MOV     AL,0                      ; ASCII CODE OF ZERO
EADE E9F400                  JMP     K57                       ; PUT IT IN THE BUFFER

                       ;----- LOOK FOR TOP ROW OF ALTERNATE SHIFT

EAE1                   K34:                                    ; ALT-TOP-ROW
EAE1 3C02                    CMP     AL,2                      ; KEY WITH '1' ON IT
EAE3 720C                    JB      K35                       ; NOT ONE OF INTERESTING KEYS
EAE5 3C0E                    CMP     AL,14                     ; IS IT IN THE REGION
EAE7 7308                    JAE     K35                       ; ALT-FUNCTION
EAE9 80C476                  ADD     AH,118                    ; CONVERT PSUEDO SCAN CODE TO RANGE
EAEC B000                    MOV     AL,0                      ; INDICATE AS SUCH
EAEE E9E400                  JMP     K57                       ; BUFFER_FILL

                       ;----- TRANSLATE ALTERNATE SHIFT PSEUDO SCAN CODES

EAF1                   K35:                                    ; ALT-FUNCTION
EAF1 3C3B                    CMP     AL,59                     ; TEST FOR IN TABLE
EAF3 7303                    JAE     K37                       ; ALT-CONTINUE
EAF5                   K36:                                    ; CLOSE-RETURN
EAF5 E961FF                  JMP     K26                       ; IGNORE THE KEY
EAF8                   K37:                                    ; ALT-CONTINUE
EAF8 3C47                    CMP     AL,71                     ; IN KEYPAD REGION
EAFA 73F9                    JAE     K36                       ; IF SO, IGNORE
EAFC BB5FE9                  MOV     BX,OFFSET K13             ; ALT SHIFT PSEUDO SCAN TABLE
EAFF E91B01                  JMP     K63                       ; TRANSLATE THAT

                       ;----- NOT IN ALTERNATE SHIFT

EB02                   K38:                                    ; NOT-ALT-SHIFT
EB02 F606170004              TEST    KB_FLAG,CTL_SHIFT         ; ARE WE IN CONTROL SHIFT
EB07 7458                    JZ      K44                       ; NOT-CTL-SHIFT

                       ;----- CONTROL SHIFT, TEST SPECIAL CHARACTERS
                       ;----- TEST FOR BREAK AND PAUSE KEYS

EB09 3C46                    CMP     AL,SCROLL_KEY             ; TEST FOR BREAK
EB0B 7518                    JNE     K39                       ; NO-BREAK
EB0D 8B1E8000                MOV     BX,BUFFER_START           ; RESET BUFFER TO EMPTY
EB11 891E1A00                MOV     BUFFER_HEAD,BX
EB15 891E1C00                MOV     BUFFER_TAIL,BX
EB19 C606710080              MOV     BIOS_BREAK,80H            ; TURN ON BIOS_BREAK BIT
EB1E CD1B                    INT     1BH                       ; BREAK INTERRUPT VECTOR
EB20 2BC0                    SUB     AX,AX                     ; PUT OUT DUMMY CHARACTER
EB22 E9B000                  JMP     K57                       ; BUFFER_FILL
EB25                   K39:                                    ; NO-BREAK
EB25 3C45                    CMP     AL,NUM_KEY                ; LOOK FOR PAUSE KEY
EB27 7521                    JNE     K41                       ; NO-PAUSE
EB29 800E180008              OR      KB_FLAG_1,HOLD_STATE      ; TURN ON THE HOLD FLAG
EB2E B020                    MOV     AL,EOI                    ; END OF INTERRUPT TO CONTROL PORT
EB30 E620                    OUT     020H,AL                   ; ALLOW FURTHER KEYSTROKE INTS

                       ;----- DURING PAUSE INTERVAL, TURN CRT BACK ON

EB32 803E490007              CMP     CRT_MODE,7                ; IS THIS BLACK AND WHITE CARD
EB37 7407                    JE      K40                       ; YES, NOTHING TO DO
EB39 BAD803                  MOV     DX,03D8H                  ; PORT FOR COLOR CARD
EB3C A06500                  MOV     AL,CRT_MODE_SET           ; GET THE VALUE OF THE CURRENT MODE
EB3F EE                      OUT     DX,AL                     ; SET THE CRT MODE, SO THAT CRT IS ON
```

SECTION 5

```
EB40                     K40:                                 ; PAUSE-LOOP
EB40 F606180008                  TEST    KB_FLAG_1,HOLD_STATE
EB45 75F9                        JNZ     K40                  ; LOOP UNTIL FLAG TURNED OFF
EB47 E914FF                      JMP     K27                  ; INTERRUPT_RETURN_NO_EOI
EB4A                     K41:                                 ; NO-PAUSE
                         ;----- TEST SPECIAL CASE KEY 55
EB4A 3C37                        CMP     AL,55
EB4C 7506                        JNE     K42                  ; NOT-KEY-55
EB4E B80072                      MOV     AX,114*256           ; START/STOP PRINTING SWITCH
EB51 E98100                      JMP     K57                  ; BUFFER_FILL
                         ;----- SET UP TO TRANSLATE CONTROL SHIFT
EB54                     K42:                                 ; NOT-KEY-55
EB54 BB8EE8                      MOV     BX,OFFSET K8         ; SET UP TO TRANSLATE CTL
EB57 3C3B                        CMP     AL,59                ; IS IT IN TABLE
                                                              ; CTL-TABLE-TRANSLATE
EB59 7276                        JB      K56                  ; YES, GO TRANSLATE CHAR
EB5B                     K43:                                 ; CTL-TABLE-TRANSLATE
EB5B BBC8E8                      MOV     BX,OFFSET K9         ; CTL TABLE SCAN
EB5E E9BC00                      JMP     K63                  ; TRANSLATE_SCAN
                         ;----- NOT IN CONTROL SHIFT
EB61                     K44:                                 ; NOT-CTL-SHIFT
EB61 3C47                        CMP     AL,71                ; TEST FOR KEYPAD REGION
EB63 732C                        JAE     K48                  ; HANDLE KEYPAD REGION
EB65 F606170003                  TEST    KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT
EB6A 745A                        JZ      K54                  ; TEST FOR SHIFT STATE
                         ;----- UPPER CASE, HANDLE SPECIAL CASES
EB6C 3C0F                        CMP     AL,15                ; BACK TAB KEY
EB6E 7505                        JNE     K45                  ; NOT-BACK-TAB
EB70 B8000F                      MOV     AX,15*256            ; SET PSEUDO SCAN CODE
EB73 EB60                        JMP     SHORT K57            ; BUFFER_FILL
EB75                     K45:                                 ; NOT-BACK-TAB
EB75 3C37                        CMP     AL,55                ; PRINT SCREEN KEY
EB77 7509                        JNE     K46                  ; NOT-PRINT-SCREEN
                         ;----- ISSUE INTERRUPT TO INDICATE PRINT SCREEN FUNCTION
EB79 B020                        MOV     AL,EOI               ; END OF CURRENT INTERRUPT
EB7B E620                        OUT     020H,AL              ;  SO FURTHER THINGS CAN HAPPEN
EB7D CD05                        INT     5H                   ; ISSUE PRINT SCREEN INTERRUPT
EB7F E9DCFE                      JMP     K27                  ; GO BACK WITHOUT EOI OCCURRING
EB82                     K46:                                 ; NOT-PRINT-SCREEN
EB82 3C3B                        CMP     AL,59                ; FUNCTION KEYS
EB84 7206                        JB      K47                  ; NOT-UPPER-FUNCTION
EB86 BB55E9                      MOV     BX,OFFSET K12        ; UPPER CASE PSEUDO SCAN CODES
EB89 E99100                      JMP     K63                  ; TRANSLATE_SCAN
EB8C                     K47:                                 ; NOT-UPPER-FUNCTION
EB8C BB1BE9                      MOV     BX,OFFSET K11        ; POINT TO UPPER CASE TABLE
EB8F EB40                        JMP     SHORT K56            ; OK, TRANSLATE THE CHAR
                         ;----- KEYPAD KEYS, MUST TEST NUM LOCK FOR DETERMINATION
EB91                     K48:                                 ; KEYPAD-REGION
EB91 F606170020                  TEST    KB_FLAG,NUM_STATE    ; ARE WE IN NUM_LOCK
EB96 7520                        JNZ     K52                  ; TEST FOR SURE
EB98 F606170003                  TEST    KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT  ; ARE WE IN SHIFT STATE
EB9D 7520                        JNZ     K53                  ; IF SHIFTED, REALLY NUM STATE
                         ;----- BASE CASE FOR KEYPAD
EB9F                     K49:                                 ; BASE-CASE
EB9F 3C4A                        CMP     AL,74                ; SPECIAL CASE FOR A COUPLE OF KEYS
EBA1 740B                        JE      K50                  ; MINUS
EBA3 3C4E                        CMP     AL,78
EBA5 740C                        JE      K51
EBA7 2C47                        SUB     AL,71                ; CONVERT ORIGIN
EBA9 BB76E9                      MOV     BX,OFFSET K15        ; BASE CASE TABLE
EBAC EB71                        JMP     SHORT K64            ; CONVERT TO PSEUDO SCAN
EBAE                     K50:
EBAE B82D4A                      MOV     AX,74*256+'-'        ; MINUS
EBB1 EB22                        JMP     SHORT K57            ; BUFFER_FILL
EBB3                     K51:
EBB3 B82B4E                      MOV     AX,78*256+'+'        ; PLUS
EBB6 EB1D                        JMP     SHORT K57            ; BUFFER_FILL
                         ;----- MIGHT BE NUM LOCK, TEST SHIFT STATUS
EBB8                     K52:                                 ; ALMOST-NUM-STATE
EBB8 F606170003                  TEST    KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT
EBBD 75E0                        JNZ     K49                  ; SHIFTED TEMP OUT OF NUM STATE
EBBF                     K53:                                 ; REALLY_NUM_STATE
EBBF 2C46                        SUB     AL,70                ; CONVERT ORIGIN
EBC1 BB69E9                      MOV     BX,OFFSET K14        ; NUM STATE TABLE
EBC4 EB0B                        JMP     SHORT K56            ; TRANSLATE_CHAR
                         ;----- PLAIN OLD LOWER CASE
EBC6                     K54:                                 ; NOT-SHIFT
EBC6 3C3B                        CMP     AL,59                ; TEST FOR FUNCTION KEYS
EBC8 7204                        JB      K55                  ; NOT-LOWER-FUNCTION
EBCA B000                        MOV     AL,0                 ; SCAN CODE IN AH ALREADY
EBCC EB07                        JMP     SHORT K57            ; BUFFER_FILL
EBCE                     K55:                                 ; NOT-LOWER-FUNCTION
EBCE BBE1E8                      MOV     BX,OFFSET K10        ; LC TABLE
                         ;----- TRANSLATE THE CHARACTER
EBD1                     K56:                                 ; TRANSLATE-CHAR
EBD1 FEC8                        DEC     AL                   ; CONVERT ORIGIN
EBD3 2ED7                        XLAT    CS:K11               ; CONVERT THE SCAN CODE TO ASCII
                         ;----- PUT CHARACTER INTO BUFFER
EBD5                     K57:                                 ; BUFFER-FILL
EBD5 3CFF                        CMP     AL,-1                ; IS THIS AN IGNORE CHAR
EBD7 741F                        JE      K59                  ; YES, DO NOTHING WITH IT
EBD9 80FCFF                      CMP     AH,-1                ; LOOK FOR -1 PSEUDO SCAN
EBDC 741A                        JE      K59                  ; NEAR_INTERRUPT_RETURN
```

```
                     ;----- HANDLE THE CAPS LOCK PROBLEM
EBDE                 K58:                                       ; BUFFER-FILL-NOTEST
EBDE F606170040              TEST    KB_FLAG,CAPS_STATE         ; ARE WE IN CAPS LOCK STATE
EBE3 7420                    JZ      K61                        ; SKIP IF NOT
                     ;----- IN CAPS LOCK STATE
EBE5 F606170003              TEST    KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT  ; TEST FOR SHIFT STATE
EBEA 740F                    JZ      K60                        ; IF NOT SHIFT, CONVERT LOWER TO UPPER
                     ;----- CONVERT ANY UPPER CASE TO LOWER CASE
EBEC 3C41                    CMP     AL,'A'                     ; FIND OUT IF ALPHABETIC
EBEE 7215                    JB      K61                        ; NOT_CAPS_STATE
EBF0 3C5A                    CMP     AL,'Z'
EBF2 7711                    JA      K61                        ; NOT_CAPS_STATE
EBF4 0420                    ADD     AL,'a'-'A'                 ; CONVERT TO LOWER CASE
EBF6 EB0D                    JMP     SHORT K61                  ; NOT_CAPS_STATE
EBF8                 K59:                                       ; NEAR-INTERRUPT-RETURN
EBF8 E95EFE                  JMP     K26                        ; INTERRUPT_RETURN
                     ;----- CONVERT ANY LOWER CASE TO UPPER CASE
EBFB                 K60:                                       ; LOWER-TO-UPPER
EBFB 3C61                    CMP     AL,'a'                     ; FIND OUT IF ALPHABETIC
EBFD 7206                    JB      K61                        ; NOT_CAPS_STATE
EBFF 3C7A                    CMP     AL,'z'
EC01 7702                    JA      K61                        ; NOT_CAPS_STATE
EC03 2C20                    SUB     AL,'a'-'A'                 ; CONVERT TO UPPER CASE
EC05                 K61:                                       ; NOT-CAPS-STATE
EC05 8B1E1C00                MOV     BX,BUFFER_TAIL             ; GET THE END POINTER TO THE BUFFER
EC09 8BF3                    MOV     SI,BX                      ; SAVE THE VALUE
EC0B E863FC                  CALL    K4                         ; ADVANCE THE TAIL
EC0E 3B1E1A00                CMP     BX,BUFFER_HEAD             ; HAS THE BUFFER WRAPPED AROUND
EC12 7413                    JE      K62                        ; BUFFER_FULL_BEEP
EC14 8904                    MOV     [SI],AX                    ; STORE THE VALUE
EC16 891E1C00                MOV     BUFFER_TAIL,BX             ; MOVE THE POINTER UP
EC1A E93CFE                  JMP     K26                        ; INTERRUPT_RETURN
                     ;----- TRANSLATE SCAN FOR PSEUDO SCAN CODES
EC1D                 K63:                                       ; TRANSLATE-SCAN
EC1D 2C3B                    SUB     AL,59                      ; CONVERT ORIGIN TO FUNCTION KEYS
EC1F                 K64:                                       ; TRANSLATE-SCAN-ORGD
EC1F 2ED7                    XLAT    CS:K9                      ; CTL TABLE SCAN
EC21 8AE0                    MOV     AH,AL                      ; PUT VALUE INTO AH
EC23 B000                    MOV     AL,0                       ; ZERO ASCII CODE
EC25 EBAE                    JMP     K57                        ; PUT IT INTO THE BUFFER
                     KB_INT  ENDP
                     ;----- BUFFER IS FULL, SOUND THE BEEPER
EC27                 K62:                                       ; BUFFER-FULL-BEEP
EC27 B020                    MOV     AL,EOI                     ; END OF INTERRUPT COMMAND
EC29 E620                    OUT     20H,AL                     ; SEND COMMAND TO INT CONTROL PORT
EC2B BB8000                  MOV     BX,080H                    ; NUMBER OF CYCLES FOR 1/12 SECOND TONE
EC2E E461                    IN      AL,KB_CTL                  ; GET CONTROL INFORMATION
EC30 50                      PUSH    AX                         ; SAVE
EC31                 K65:                                       ; BEEP-CYCLE
EC31 24FC                    AND     AL,0FCH                    ; TURN OFF TIMER GATE AND SPEAKER DATA
EC33 E661                    OUT     KB_CTL,AL                  ; OUTPUT TO CONTROL
EC35 B94800                  MOV     CX,48H                     ; HALF CYCLE TIME FOR TONE
EC38                 K66:
EC38 E2FE                    LOOP    K66                        ; SPEAKER OFF
EC3A 0C02                    OR      AL,2                       ; TURN ON SPEAKER BIT
EC3C E661                    OUT     KB_CTL,AL                  ; OUTPUT TO CONTROL
EC3E B94800                  MOV     CX,48H                     ; SET UP COUNT
EC41                 K67:
EC41 E2FE                    LOOP    K67                        ; ANOTHER HALF CYCLE
EC43 4B                      DEC     BX                         ; TOTAL TIME COUNT
EC44 75EB                    JNZ     K65                        ; DO ANOTHER CYCLE
EC46 58                      POP     AX                         ; RECOVER CONTROL
EC47 E661                    OUT     KB_CTL,AL                  ; OUTPUT THE CONTROL
EC49 E912FE                  JMP     K27
EC4C 20333031        F1      DB      ' 301',13,10               ; KEYBOARD ERROR
EC50 0D
EC51 0A
EC52 363031          F3      DB      '601',13,10                ; DISKETTE ERROR
EC55 0D
EC56 0A
```

SECTION 5

```
                    ;-- INT 13 -----------------------------------------------------------
                    ; DISKETTE I/O                                                       :
                    ;       THIS INTERFACE PROVIDES ACCESS TO THE 5 1/4 DISKETTE DRIVES  :
                    ; INPUT                                                              :
                    ;       (AH)=0  RESET DISKETTE SYSTEM                                :
                    ;               HARD RESET TO NEC, PREPARE COMMAND, RECAL REQUIRED   :
                    ;               ON ALL DRIVES                                        :
                    ;       (AH)=1  READ THE STATUS OF THE SYSTEM INTO (AL)              :
                    ;               DISKETTE_STATUS FROM LAST OPERATION IS USED          :
                    ;                                                                    :
                    ; REGISTERS FOR READ/WRITE/VERIFY/FORMAT                             :
                    ;       (DL) - DRIVE NUMBER (0-3 ALLOWED, VALUE CHECKED)             :
                    ;       (DH) - HEAD NUMBER (0-1 ALLOWED, NOT VALUE CHECKED)          :
                    ;       (CH) - TRACK NUMBER (0-39, NOT VALUE CHECKED)                :
                    ;       (CL) - SECTOR NUMBER (1-8, NOT VALUE CHECKED,                :
                    ;                               NOT USED FOR FORMAT)                 :
                    ;       (AL) - NUMBER OF SECTORS ( MAX = 8, NOT VALUE CHECKED, NOT USED :
                    ;                                       FOR FORMAT)                  :
                    ;       (ES:BX) - ADDRESS OF BUFFER ( NOT 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.                                              :
                    ;                                                                    :
                    ; 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)                :
                    ;       CY = 1  FAILED OPERATION (AH HAS ERROR REASON)               :
                    ;       FOR READ/WRITE/VERIFY                                        :
                    ;               DS,BX,DX,CH,CL PRESERVED                             :
                    ;               AL = NUMBER OF SECTORS ACTUALLY READ                 :
                    ;               ***** AL MAY NOT BE CORRECT IF TIME OUT ERROR OCCURS :
                    ;       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.                                            :
                    ;--------------------------------------------------------------------
                            ASSUME  CS:CODE,DS:DATA,ES:DATA
EC59                        ORG     0EC59H
EC59                DISKETTE_IO     PROC    FAR
EC59 FB                     STI                             ; INTERRUPTS BACK ON
EC5A 53                     PUSH    BX                      ; SAVE ADDRESS
EC5B 51                     PUSH    CX
EC5C 1E                     PUSH    DS                      ; SAVE SEGMENT REGISTER VALUE
EC5D 56                     PUSH    SI                      ; SAVE ALL REGISTERS DURING OPERATION
EC5E 57                     PUSH    DI
EC5F 55                     PUSH    BP
EC60 52                     PUSH    DX
EC61 8BEC                   MOV     BP,SP                   ; SET UP POINTER TO HEAD PARM
EC63 E8F30D                 CALL    DDS
EC66 E81C00                 CALL    J1                      ; CALL THE REST TO ENSURE DS RESTORED
EC69 BB0400                 MOV     BX,4                    ; GET THE MOTOR WAIT PARAMETER
EC6C E8FD01                 CALL    GET_PARM
EC6F 88264000               MOV     MOTOR_COUNT,AH          ; SET THE TIMER COUNT FOR THE MOTOR
EC73 8A264100               MOV     AH,DISKETTE_STATUS      ; GET STATUS OF OPERATION
EC77 80FC01                 CMP     AH,1                    ; SET THE CARRY FLAG TO INDICATE
EC7A F5                     CMC                             ;  SUCCESS OR FAILURE
EC7B 5A                     POP     DX                      ; RESTORE ALL REGISTERS
EC7C 5D                     POP     BP
EC7D 5F                     POP     DI
EC7E 5E                     POP     SI
EC7F 1F                     POP     DS
EC80 59                     POP     CX
EC81 5B                     POP     BX                      ; RECOVER ADDRESS
EC82 CA0200                 RET     2                       ; THROW AWAY SAVED FLAGS
                    DISKETTE_IO     ENDP

EC85                J1      PROC    NEAR
EC85 8AF0                   MOV     DH,AL                   ; SAVE # SECTORS IN DH
EC87 80263F007F             AND     MOTOR_STATUS,07FH       ; INDICATE A READ OPERATION
EC8C 0AE4                   OR      AH,AH                   ; AH=0
EC8E 7427                   JZ      DISK_RESET
EC90 FECC                   DEC     AH                      ; AH=1
EC92 7473                   JZ      DISK_STATUS
EC94 C606410000             MOV     DISKETTE_STATUS,0       ; RESET THE STATUS INDICATOR
EC99 80FA04                 CMP     DL,4                    ; TEST FOR DRIVE IN 0-3 RANGE
EC9C 7313                   JAE     J3                      ; ERROR IF ABOVE
EC9E FECC                   DEC     AH                      ; AH=2
ECA0 7469                   JZ      DISK_READ
ECA2 FECC                   DEC     AH                      ; AH=3
ECA4 7503                   JNZ     J2                      ; TEST_DISK_VERF
ECA6 E99500                 JMP     DISK_WRITE
ECA9                J2:                                     ; TEST_DISK_VERF
ECA9 FECC                   DEC     AH                      ; AH=4
ECAB 7467                   JZ      DISK_VERF
ECAD FECC                   DEC     AH                      ; AH=5
ECAF 7467                   JZ      DISK_FORMAT
ECB1                J3:                                     ; BAD_COMMAND
ECB1 C606410001             MOV     DISKETTE_STATUS,BAD_CMD ; ERROR CODE, NO SECTORS TRANSFERRED
ECB6 C3                     RET                             ; UNDEFINED OPERATION
                    J1      ENDP
```

```
                  ;----- RESET THE DISKETTE SYSTEM

ECB7              DISK_RESET      PROC    NEAR
ECB7 BAF203               MOV     DX,03F2H                ; ADAPTER CONTROL PORT
ECBA FA                   CLI                             ; NO INTERRUPTS
ECBB A03F00               MOV     AL,MOTOR_STATUS         ; WHICH MOTOR IS ON
ECBE B104                 MOV     CL,4                    ; SHIFT COUNT
ECC0 D2E0                 SAL     AL,CL                   ; MOVE MOTOR VALUE TO HIGH NYBBLE
ECC2 A820                 TEST    AL, 20H                 ; SELECT CORRESPONDING DRIVE
ECC4 750C                 JNZ     J5                      ; JUMP IF MOTOR ONE IS ON
ECC6 A840                 TEST    AL, 40H
ECC8 7506                 JNZ     J4                      ; JUMP IF MOTOR TWO IS ON
ECCA A880                 TEST    AL, 80H
ECCC 7406                 JZ      J6                      ; JUMP IF MOTOR ZERO IS ON
ECCE FEC0                 INC     AL
ECD0              J4:
ECD0 FEC0                 INC     AL
ECD2              J5:
ECD2 FEC0                 INC     AL
ECD4              J6:
ECD4 0C08                 OR      AL,8                    ; TURN ON INTERRUPT ENABLE
ECD6 EE                   OUT     DX,AL                   ; RESET THE ADAPTER
ECD7 C6063E0000           MOV     SEEK_STATUS,0           ; SET RECAL REQUIRED ON ALL DRIVES
ECDC C606410000           MOV     DISKETTE_STATUS,0       ; SET OK STATUS FOR DISKETTE
ECE1 0C04                 OR      AL,4                    ; TURN OFF RESET
ECE3 EE                   OUT     DX,AL                   ; TURN OFF THE RESET
ECE4 FB                   STI                             ; REENABLE THE INTERRUPTS
ECE5 E82A02               CALL    CHK_STAT_2              ; DO SENSE INTERRUPT STATUS
                                                          ;  FOLLOWING RESET
ECE8 A04200               MOV     AL,NEC_STATUS           ; IGNORE ERROR RETURN AND DO OWN TEST
ECEB 3CC0                 CMP     AL,0C0H                 ; TEST FOR DRIVE READY TRANSITION
ECED 7406                 JZ      J7                      ; EVERYTHING OK
ECEF 800E410020           OR      DISKETTE_STATUS,BAD_NEC ; SET ERROR CODE
ECF4 C3                   RET

                  ;----- SEND SPECIFY COMMAND TO NEC

ECF5              J7:                                     ; DRIVE_READY
ECF5 B403                 MOV     AH,03H                  ; SPECIFY COMMAND
ECF7 E84701               CALL    NEC_OUTPUT              ; OUTPUT THE COMMAND
ECFA BB0100               MOV     BX,1                    ; FIRST BYTE PARM IN BLOCK
ECFD E86C01               CALL    GET_PARM                ;  TO THE NEC CONTROLLER
ED00 BB0300               MOV     BX,3                    ; SECOND BYTE PARM IN BLOCK
ED03 E86601               CALL    GET_PARM                ;  TO THE NEC CONTROLLER
ED06              J8:                                     ; RESET_RET
ED06 C3                   RET                             ; RETURN TO CALLER
                  DISK_RESET      ENDP

                  ;----- DISKETTE STATUS ROUTINE

ED07              DISK_STATUS     PROC    NEAR
ED07 A04100               MOV     AL,DISKETTE_STATUS
ED0A C3                   RET
                  DISK_STATUS     ENDP

                  ;----- DISKETTE READ

ED0B              DISK_READ       PROC    NEAR
ED0B B046                 MOV     AL,046H                 ; READ COMMAND FOR DMA
ED0D              J9:                                     ; DISK_READ_CONT
ED0D E8B801               CALL    DMA_SETUP               ; SET UP THE DMA
ED10 B4E6                 MOV     AH,0E6H                 ; SET UP RD COMMAND FOR NEC CONTROLLER
ED12 EB36                 JMP     SHORT RW_OPN            ; GO DO THE OPERATION
                  DISK_READ       ENDP

                  ;----- DISKETTE VERIFY

ED14              DISK_VERF       PROC    NEAR
ED14 B042                 MOV     AL,042H                 ; VERIFY COMMAND FOR DMA
ED16 EBF5                 JMP     J9                      ; DO AS IF DISK READ
                  DISK_VERF       ENDP

                  ;----- DISKETTE FORMAT

ED18              DISK_FORMAT     PROC    NEAR
ED18 800E3F0080           OR      MOTOR_STATUS,80H        ; INDICATE WRITE OPERATION
ED1D B04A                 MOV     AL,04AH                 ; WILL WRITE TO THE DISKETTE
ED1F E8A601               CALL    DMA_SETUP               ; SET UP THE DMA
ED22 B44D                 MOV     AH,04DH                 ; ESTABLISH THE FORMAT COMMAND
ED24 EB24                 JMP     SHORT RW_OPN            ; DO THE OPERATION
ED26              J10:                                    ; CONTINUATION OF RW_OPN FOR FMT
ED26 BB0700               MOV     BX,7                    ; GET THE
ED29 E84001               CALL    GET_PARM                ;  BYTES/SECTOR VALUE TO NEC
ED2C BB0900               MOV     BX,9                    ; GET THE
ED2F E83A01               CALL    GET_PARM                ;  SECTORS/TRACK VALUE TO NEC
ED32 BB0F00               MOV     BX,15                   ; GET THE
ED35 E83401               CALL    GET_PARM                ;  GAP LENGTH VALUE TO NEC
ED38 BB1100               MOV     BX,17                   ; GET THE FILLER BYTE
ED3B E9AB00               JMP     J16                     ;  TO THE CONTROLLER
                  DISK_FORMAT     ENDP

                  ;----- DISKETTE WRITE ROUTINE

ED3E              DISK_WRITE      PROC    NEAR
ED3E 800E3F0080           OR      MOTOR_STATUS,80H        ; INDICATE WRITE OPERATION
ED43 B04A                 MOV     AL,04AH                 ; DMA WRITE COMMAND
ED45 E88001               CALL    DMA_SETUP
ED48 B4C5                 MOV     AH,0C5H                 ; NEC COMMAND TO WRITE TO DISKETTE
                  DISK_WRITE      ENDP

                  ;----- ALLOW WRITE ROUTINE TO FALL INTO RW_OPN
```

```
                      ;-------------------------------------------------------------
                      ; RW_OPN                                                      :
                      ;       THIS ROUTINE PERFORMS THE READ/WRITE/VERIFY OPERATION :
                      ;-------------------------------------------------------------
ED4A                  RW_OPN  PROC    NEAR
ED4A 7308                     JNC     J11                         ; TEST FOR DMA ERROR
ED4C C606410009               MOV     DISKETTE_STATUS,DMA_BOUNDARY     ; SET ERROR
ED51 B000                     MOV     AL,0                        ; NO SECTORS TRANSFERRED
ED53 C3                       RET                                 ; RETURN TO MAIN ROUTINE
ED54                  J11:                                        ; DO_RW_OPN
ED54 50                       PUSH    AX                          ; SAVE THE COMMAND

                      ;----- TURN ON THE MOTOR AND SELECT THE DRIVE

ED55 51                       PUSH    CX                          ; SAVE THE T/S PARMS
ED56 8ACA                     MOV     CL,DL                       ; GET DRIVE NUMBER AS SHIFT COUNT
ED58 B001                     MOV     AL,1                        ; MASK FOR DETERMINING MOTOR BIT
ED5A D2E0                     SAL     AL,CL                       ; SHIFT THE MASK BIT
ED5C FA                       CLI                                 ; NO INTERRUPTS WHILE DETERMINING
                                                                  ;  MOTOR STATUS
ED5D C6064000FF               MOV     MOTOR_COUNT,0FFH            ; SET LARGE COUNT DURING OPERATION
ED62 84063F00                 TEST    AL,MOTOR_STATUS             ; TEST THAT MOTOR FOR OPERATING
ED66 7531                     JNZ     J14                         ; IF RUNNING, SKIP THE WAIT
ED68 80263F00F0               AND     MOTOR_STATUS,0F0H           ; TURN OFF ALL MOTOR BITS
ED6D 08063F00                 OR      MOTOR_STATUS,AL             ; TURN ON THE CURRENT MOTOR
ED71 FB                       STI                                 ; INTERRUPTS BACK ON
ED72 B010                     MOV     AL,10H                      ; MASK BIT
ED74 D2E0                     SAL     AL,CL                       ; DEVELOP BIT MASK FOR MOTOR ENABLE
ED76 0AC2                     OR      AL,DL                       ; GET DRIVE SELECT BITS IN
ED78 0C0C                     OR      AL,0CH                      ; NO RESET, ENABLE DMA/INT
ED7A 52                       PUSH    DX                          ; SAVE REG
ED7B BAF203                   MOV     DX,03F2H                    ; CONTROL PORT ADDRESS
ED7E EE                       OUT     DX,AL
ED7F 5A                       POP     DX                          ; RECOVER REGISTERS

                      ;----- WAIT FOR MOTOR IF WRITE OPERATION

ED80 F6063F0080               TEST    MOTOR_STATUS,80H            ; IS THIS A WRITE
ED85 7412                     JZ      J14                         ; NO, CONTINUE WITHOUT WAIT
ED87 BB1400                   MOV     BX,20                       ; GET THE MOTOR WAIT
ED8A E8DF00                   CALL    GET_PARM                    ;  PARAMETER
ED8D 0AE4                     OR      AH,AH                       ; TEST FOR NO WAIT
ED8F                  J12:                                        ; TEST_WAIT_TIME
ED8F 7408                     JZ      J14                         ; EXIT WITH TIME EXPIRED
ED91 2BC9                     SUB     CX,CX                       ; SET UP 1/8 SECOND LOOP TIME
ED93                  J13:
ED93 E2FE                     LOOP    J13                         ; WAIT FOR THE REQUIRED TIME
ED95 FECC                     DEC     AH                          ; DECREMENT TIME VALUE
ED97 EBF6                     JMP     J12                         ; ARE WE DONE YET
ED99                  J14:                                        ; MOTOR_RUNNING
ED99 FB                       STI                                 ; INTERRUPTS BACK ON FOR BYPASS WAIT
ED9A 59                       POP     CX

                      ;----- DO THE SEEK OPERATION

ED9B E8DF00                   CALL    SEEK                        ; MOVE TO CORRECT TRACK
ED9E 58                       POP     AX                          ; RECOVER COMMAND
ED9F 8AFC                     MOV     BH,AH                       ; SAVE COMMAND IN BH
EDA1 B600                     MOV     DH,0                        ; SET NO SECTORS READ IN CASE OF ERROR
EDA3 724B                     JC      J17                         ; IF ERROR, THEN EXIT AFTER MOTOR OFF
EDA5 BEF0ED90                 MOV     SI,OFFSET J17               ; DUMMY RETURN ON STACK FOR NEC_OUTPUT
EDA9 56                       PUSH    SI                          ;  SO THAT IT WILL RETURN TO MOTOR OFF
                                                                  ;  LOCATION

                      ;----- SEND OUT THE PARAMETERS TO THE CONTROLLER

EDAA E89400                   CALL    NEC_OUTPUT                  ; OUTPUT THE OPERATION COMMAND
EDAD 8A6601                   MOV     AH,[BP+1]                   ; GET THE CURRENT HEAD NUMBER
EDB0 D0E4                     SAL     AH,1                        ; MOVE IT TO BIT 2
EDB2 D0E4                     SAL     AH,1
EDB4 80E404                   AND     AH,4                        ; ISOLATE THAT BIT
EDB7 0AE2                     OR      AH,DL                       ; OR IN THE DRIVE NUMBER
EDB9 E88500                   CALL    NEC_OUTPUT

                      ;----- TEST FOR FORMAT COMMAND

EDBC 80FF4D                   CMP     BH,04DH                     ; IS THIS A FORMAT OPERATION
EDBF 7503                     JNE     J15                         ; NO. CONTINUE WITH R/W/V
EDC1 E962FF                   JMP     J10                         ; IF SO, HANDLE SPECIAL
EDC4                  J15:
EDC4 8AE5                     MOV     AH,CH                       ; CYLINDER NUMBER
EDC6 E87800                   CALL    NEC_OUTPUT
EDC9 8A6601                   MOV     AH,[BP+1]                   ; HEAD NUMBER FROM STACK
EDCC E87200                   CALL    NEC_OUTPUT
EDCF 8AE1                     MOV     AH,CL                       ; SECTOR NUMBER
EDD1 E86D00                   CALL    NEC_OUTPUT
EDD4 BB0700                   MOV     BX,7                        ; BYTES/SECTOR PARM FROM BLOCK
EDD7 E89200                   CALL    GET_PARM                    ;  TO THE NEC
EDDA BB0900                   MOV     BX,9                        ; EOT PARM FROM BLOCK
EDDD E88C00                   CALL    GET_PARM                    ;  TO THE NEC
EDE0 BB0B00                   MOV     BX,11                       ; GAP LENGTH PARM FROM BLOCK
EDE3 E88600                   CALL    GET_PARM                    ;  TO THE NEC
EDE6 BB0D00                   MOV     BX,13                       ; DTL PARM FROM BLOCK
EDE9                  J16:                                        ; RW_OPN_FINISH
EDE9 E88000                   CALL    GET_PARM                    ;  TO THE NEC
EDEC 5E                       POP     SI                          ; CAN NOW DISCARD THAT DUMMY
                                                                  ;  RETURN ADDRESS

                      ;----- LET THE OPERATION HAPPEN

EDED E84301                   CALL    WAIT_INT                    ; WAIT FOR THE INTERRUPT
EDF0                  J17:                                        ; MOTOR_OFF
EDF0 7245                     JC      J21                         ; LOOK FOR ERROR
EDF2 E87401                   CALL    RESULTS                     ; GET THE NEC STATUS
EDF5 723F                     JC      J20                         ; LOOK FOR ERROR

                      ;----- CHECK THE RESULTS RETURNED BY THE CONTROLLER

EDF7 FC                       CLD                                 ; SET THE CORRECT DIRECTION
EDF8 BE4200                   MOV     SI,OFFSET NEC_STATUS        ; POINT TO STATUS FIELD
EDFB AC                       LODS    NEC_STATUS                  ; GET ST0
EDFC 24C0                     AND     AL,0C0H                     ; TEST FOR NORMAL TERMINATION
EDFE 743B                     JZ      J22                         ; OPN_OK
EE00 3C40                     CMP     AL,040H                     ; TEST FOR ABNORMAL TERMINATION
EE02 7529                     JNZ     J18                         ; NOT ABNORMAL, BAD NEC

```

```
                   ;----- ABNORMAL TERMINATION, FIND OUT WHY

EE04 AC                    LODS    NEC_STATUS              ; GET ST1
EE05 D0E0                  SAL     AL,1                    ; TEST FOR EOT FOUND
EE07 B404                  MOV     AH,RECORD_NOT_FND
EE09 7224                  JC      J19                     ; RW_FAIL
EE0B D0E0                  SAL     AL,1
EE0D D0E0                  SAL     AL,1                    ; TEST FOR CRC ERROR
EE0F B410                  MOV     AH,BAD_CRC
EE11 721C                  JC      J19                     ; RW_FAIL
EE13 D0E0                  SAL     AL,1                    ; TEST FOR DMA OVERRUN
EE15 B408                  MOV     AH,BAD_DMA
EE17 7216                  JC      J19                     ; RW_FAIL
EE19 D0E0                  SAL     AL,1
EE1B D0E0                  SAL     AL,1                    ; TEST FOR RECORD NOT FOUND
EE1D B404                  MOV     AH,RECORD_NOT_FND
EE1F 720E                  JC      J19                     ; RW_FAIL
EE21 D0E0                  SAL     AL,1
EE23 B403                  MOV     AH,WRITE_PROTECT        ; TEST FOR WRITE_PROTECT
EE25 7208                  JC      J19                     ; RW_FAIL
EE27 D0E0                  SAL     AL,1                    ; TEST MISSING ADDRESS MARK
EE29 B402                  MOV     AH,BAD_ADDR_MARK
EE2B 7202                  JC      J19                     ; RW_FAIL

                   ;----- NEC MUST HAVE FAILED

EE2D               J18:                                    ; RW-NEC-FAIL
EE2D B420                  MOV     AH,BAD_NEC
EE2F               J19:                                    ; RW-FAIL
EE2F 08264100              OR      DISKETTE_STATUS,AH
EE33 E87801                CALL    NUM_TRANS               ; HOW MANY WERE REALLY TRANSFERRED
EE36               J20:                                    ; RW_ERR
EE36 C3                    RET                             ; RETURN TO CALLER
EE37               J21:                                    ; RW_ERR_RES
EE37 E82F01                CALL    RESULTS                 ; FLUSH THE RESULTS BUFFER
EE3A C3                    RET

                   ;----- OPERATION WAS SUCCESSFUL

EE3B               J22:                                    ; OPN_OK
EE3B E87001                CALL    NUM_TRANS               ; HOW MANY GOT MOVED
EE3E 32E4                  XOR     AH,AH                   ; NO ERRORS
EE40 C3                    RET
                   RW_OPN  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                                           :
                   ;----------------------------------------------------------------
EE41               NEC_OUTPUT      PROC    NEAR
EE41 52                    PUSH    DX                      ; SAVE REGISTERS
EE42 51                    PUSH    CX
EE43 BAF403                MOV     DX,03F4H                ; STATUS PORT
EE46 33C9                  XOR     CX,CX                   ; COUNT FOR TIME OUT
EE48               J23:
EE48 EC                    IN      AL,DX                   ; GET STATUS
EE49 A840                  TEST    AL,040H                 ; TEST DIRECTION BIT
EE4B 740C                  JZ      J25                     ; DIRECTION OK
EE4D E2F9                  LOOP    J23
EE4F               J24:                                    ; TIME_ERROR
EE4F 800E410080            OR      DISKETTE_STATUS,TIME_OUT
EE54 59                    POP     CX
EE55 5A                    POP     DX                      ; SET ERROR CODE AND RESTORE REGS
EE56 58                    POP     AX                      ; DISCARD THE RETURN ADDRESS
EE57 F9                    STC                             ; INDICATE ERROR TO CALLER
EE58 C3                    RET
EE59               J25:
EE59 33C9                  XOR     CX,CX                   ; RESET THE COUNT
EE5B               J26:
EE5B EC                    IN      AL,DX                   ; GET THE STATUS
EE5C A880                  TEST    AL,080H                 ; IS IT READY
EE5E 7504                  JNZ     J27                     ; YES, GO OUTPUT
EE60 E2F9                  LOOP    J26                     ; COUNT DOWN AND TRY AGAIN
EE62 EBEB                  JMP     J24                     ; ERROR CONDITION
EE64               J27:                                    ; OUTPUT
EE64 8AC4                  MOV     AL,AH                   ; GET BYTE TO OUTPUT
EE66 B2F5                  MOV     DL,0F5H                 ; DATA PORT (3F5)
EE68 EE                    OUT     DX,AL                   ; OUTPUT THE BYTE
EE69 59                    POP     CX                      ; RECOVER REGISTERS
EE6A 5A                    POP     DX
EE6B C3                    RET                             ; CY = 0 FROM TEST INSTRUCTION
                   NEC_OUTPUT      ENDP
```

SECTION 5

```
                  ;----------------------------------------------------------------
                  ; 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                                       :
                  ;----------------------------------------------------------------
EE6C              GET_PARM        PROC    NEAR
EE6C 1E                   PUSH    DS                      ; SAVE SEGMENT
EE6D 2BC0                 SUB     AX,AX                   ; ZERO TO AX
EE6F 8ED8                 MOV     DS,AX
                          ASSUME  DS:ABS0
EE71 C5367800             LDS     SI,DISK_POINTER         ; POINT TO BLOCK
EE75 D1EB                 SHR     BX,1                    ; DIVIDE BX BY 2, AND SET FLAG
                                                          ; FOR EXIT
EE77 8A20                 MOV     AH,[SI+BX]              ; GET THE WORD
EE79 1F                   POP     DS                      ; RESTORE SEGMENT
                          ASSUME  DS:DATA
EE7A 72C5                 JC      NEC_OUTPUT              ; IF FLAG SET, OUTPUT TO CONTROLLER
EE7C C3                   RET                             ; RETURN TO CALLER
                  GET_PARM        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                                              :
                  ;----------------------------------------------------------------
EE7D              SEEK    PROC    NEAR
EE7D B001                 MOV     AL,1                    ; ESTABLISH MASK FOR RECAL TEST
EE7F 51                   PUSH    CX                      ; SAVE INPUT VALUES
EE80 8ACA                 MOV     CL,DL                   ; GET DRIVE VALUE INTO CL
EE82 D2C0                 ROL     AL,CL                   ; SHIFT IT BY THE DRIVE VALUE
EE84 59                   POP     CX                      ; RECOVER TRACK VALUE
EE85 84063E00             TEST    AL,SEEK_STATUS          ; TEST FOR RECAL REQUIRED
EE89 7513                 JNZ     J28                     ; NO_RECAL
EE8B 08063E00             OR      SEEK_STATUS,AL          ; TURN ON THE NO RECAL BIT IN FLAG
EE8F B407                 MOV     AH,07H                  ; RECALIBRATE COMMAND
EE91 E8ADFF               CALL    NEC_OUTPUT
EE94 8AE2                 MOV     AH,DL
EE96 E8A8FF               CALL    NEC_OUTPUT              ; OUTPUT THE DRIVE NUMBER
EE99 E87600               CALL    CHK_STAT_2              ; GET THE INTERUPT AND SENSE INT STATUS
EE9C 7229                 JC      J32                     ; SEEK_ERROR

                  ;----- DRIVE IS IN SYNCH WITH CONTROLLER, SEEK TO TRACK

EE9E              J28:
EE9E B40F                 MOV     AH,0FH                  ; SEEK COMMAND TO NEC
EEA0 E89EFF               CALL    NEC_OUTPUT
EEA3 8AE2                 MOV     AH,DL                   ; DRIVE NUMBER
EEA5 E899FF               CALL    NEC_OUTPUT
EEA8 8AE5                 MOV     AH,CH                   ; TRACK NUMBER
EEAA E894FF               CALL    NEC_OUTPUT
EEAD E86200               CALL    CHK_STAT_2              ; GET ENDING INTERRUPT AND
                                                          ;  SENSE STATUS

                  ;----- WAIT FOR HEAD SETTLE

EEB0 9C                   PUSHF                           ; SAVE STATUS FLAGS
EEB1 BB1200               MOV     BX,18                   ; GET HEAD SETTLE PARAMETER
EEB4 E8B5FF               CALL    GET_PARM
EEB7 51                   PUSH    CX                      ; SAVE REGISTER
EEB8              J29:                                    ; HEAD_SETTLE
EEB8 B92602               MOV     CX,550                  ; 1 MS LOOP
EEBB 0AE4                 OR      AH,AH                   ; TEST FOR TIME EXPIRED
EEBD 7406                 JZ      J31
EEBF              J30:
EEBF E2FE                 LOOP    J30                     ; DELAY FOR 1 MS
EEC1 FECC                 DEC     AH                      ; DECREMENT THE COUNT
EEC3 EBF3                 JMP     J29                     ; DO IT SOME MORE
EEC5              J31:
EEC5 59                   POP     CX                      ; RECOVER STATE
EEC6 9D                   POPF
EEC7              J32:                                    ; SEEK_ERROR
EEC7 C3                   RET                             ; RETURN TO CALLER
                  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                                                  :
                ;-----------------------------------------------------------------------
EEC8            DMA_SETUP       PROC    NEAR
EEC8 51                 PUSH    CX                      ; SAVE THE REGISTER
EEC9 FA                 CLI                             ; NO MORE INTERRUPTS
EECA E60C               OUT     DMA+12,AL               ; SET THE FIRST/LAST F/F
EECC 50                 PUSH    AX
EECD 58                 POP     AX
EECE E60B               OUT     DMA+11,AL               ; OUTPUT THE MODE BYTE
EED0 8CC0               MOV     AX,ES                   ; GET THE ES VALUE
EED2 B104               MOV     CL,4                    ; SHIFT COUNT
EED4 D3C0               ROL     AX,CL                   ; ROTATE LEFT
EED6 8AE8               MOV     CH,AL                   ; GET HIGHEST NYBLE OF ES TO CH
EED8 24F0               AND     AL,0F0H                 ; ZERO THE LOW NYBBLE FROM SEGMENT
EEDA 03C3               ADD     AX,BX                   ; TEST FOR CARRY FROM ADDITION
EEDC 7302               JNC     J33
EEDE FEC5               INC     CH                      ; CARRY MEANS HIGH 4 BITS MUST BE INC
EEE0            J33:
EEE0 50                 PUSH    AX                      ; SAVE START ADDRESS
EEE1 E604               OUT     DMA+4,AL                ; OUTPUT LOW ADDRESS
EEE3 8AC4               MOV     AL,AH
EEE5 E604               OUT     DMA+4,AL                ; OUTPUT HIGH ADDRESS
EEE7 8AC5               MOV     AL,CH                   ; GET HIGH 4 BITS
EEE9 240F               AND     AL,0FH
EEEB E681               OUT     081H,AL                 ; OUTPUT THE HIGH 4 BITS TO
                                                        ;  THE PAGE REGISTER

                ;----- DETERMINE COUNT

EEED 8AE6               MOV     AH,DH                   ; NUMBER OF SECTORS
EEEF 2AC0               SUB     AL,AL                   ;  TIMES 256 INTO AX
EEF1 D1E8               SHR     AX,1                    ; SECTORS * 128 INTO AX
EEF3 50                 PUSH    AX
EEF4 BB0600             MOV     BX,6                    ; GET THE BYTES/SECTOR PARM
EEF7 E872FF             CALL    GET_PARM
EEFA 8ACC               MOV     CL,AH                   ; USE AS SHIFT COUNT (0=128, 1=256 ETC)
EEFC 58                 POP     AX
EEFD D3E0               SHL     AX,CL                   ; MULTIPLY BY CORRECT AMOUNT
EEFF 48                 DEC     AX                      ; -1 FOR DMA VALUE
EF00 50                 PUSH    AX                      ; SAVE COUNT VALUE
EF01 E605               OUT     DMA+5,AL                ; LOW BYTE OF COUNT
EF03 8AC4               MOV     AL,AH
EF05 E605               OUT     DMA+5,AL                ; HIGH BYTE OF COUNT
EF07 FB                 STI                             ; INTERRUPTS BACK ON
EF08 59                 POP     CX                      ; RECOVER COUNT VALUE
EF09 58                 POP     AX                      ; RECOVER ADDRESS VALUE
EF0A 03C1               ADD     AX,CX                   ; ADD, TEST FOR 64K OVERFLOW
EF0C 59                 POP     CX                      ; RECOVER REGISTER
EF0D B002               MOV     AL,2                    ; MODE FOR 8237
EF0F E60A               OUT     DMA+10,AL               ; INITIALIZE THE DISKETTE CHANNEL
EF11 C3                 RET                             ; RETURN TO CALLER,
                                                        ;  CFL SET BY ABOVE IF ERROR
                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                                                  :
                ;-----------------------------------------------------------------------
EF12            CHK_STAT_2      PROC    NEAR
EF12 E81E00             CALL    WAIT_INT                ; WAIT FOR THE INTERRUPT
EF15 7214               JC      J34                     ; IF ERROR, RETURN IT
EF17 B408               MOV     AH,08H                  ; SENSE INTERRUPT STATUS COMMAND
EF19 E825FF             CALL    NEC_OUTPUT
EF1C E84A00             CALL    RESULTS                 ; READ IN THE RESULTS
EF1F 720A               JC      J34                     ; CHK2_RETURN
EF21 A04200             MOV     AL,NEC_STATUS           ; GET THE FIRST STATUS BYTE
EF24 2460               AND     AL,060H                 ; ISOLATE THE BITS
EF26 3C60               CMP     AL,060H                 ; TEST FOR CORRECT VALUE
EF28 7402               JZ      J35                     ; IF ERROR, GO MARK IT
EF2A F8                 CLC                             ; GOOD RETURN
EF2B            J34:
EF2B C3                 RET                             ; RETURN TO CALLER
EF2C            J35:                                    ; CHK2_ERROR
EF2C 800E410040         OR      DISKETTE_STATUS,BAD_SEEK
EF31 F9                 STC                             ; ERROR RETURN CODE
EF32 C3                 RET
                CHK_STAT_2      ENDP
```

SECTION 5

```
                      ;----------------------------------------------------------------------
                      ; 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                                                 :
                      ;----------------------------------------------------------------------
EF33                  WAIT_INT        PROC    NEAR
EF33 FB                       STI                             ; TURN ON INTERRUPTS, JUST IN CASE
EF34 53                       PUSH    BX
EF35 51                       PUSH    CX                      ; SAVE REGISTERS
EF36 B302                     MOV     BL,2                    ; CLEAR THE COUNTERS
EF38 33C9                     XOR     CX,CX                   ; FOR 2 SECOND WAIT
EF3A                  J36:
EF3A F6063E0080               TEST    SEEK_STATUS,INT_FLAG    ; TEST FOR INTERRUPT OCCURRING
EF3F 750C                     JNZ     J37
EF41 E2F7                     LOOP    J36                     ; COUNT DOWN WHILE WAITING
EF43 FECB                     DEC     BL                      ; SECOND LEVEL COUNTER
EF45 75F3                     JNZ     J36
EF47 800E410080               OR      DISKETTE_STATUS,TIME_OUT        ; NOTHING HAPPENED
EF4C F9                       STC                             ; ERROR RETURN
EF4D                  J37:
EF4D 9C                       PUSHF                           ; SAVE CURRENT CARRY
EF4E 80263E007F               AND     SEEK_STATUS,NOT INT_FLAG        ; TURN OFF INTERRUPT FLAG
EF53 9D                       POPF                            ; RECOVER CARRY
EF54 59                       POP     CX
EF55 5B                       POP     BX                      ; RECOVER REGISTERS
EF56 C3                       RET                             ; GOOD RETURN CODE COMES
                                                              ;  FROM TEST INST
                      WAIT_INT        ENDP
                      ;-------------------------------------------------------
                      ; DISK_INT                                              :
                      ;       THIS ROUTINE HANDLES THE DISKETTE INTERRUPT     :
                      ;INPUT                                                  :
                      ;       NONE                                            :
                      ; OUTPUT                                                :
                      ;       THE INTERRUPT FLAG IS SET IS SEEK_STATUS        :

                      ;-------------------------------------------------------
EF57                          ORG     0EF57H
EF57                  DISK_INT        PROC    FAR
EF57 FB                       STI                             ; RE ENABLE INTERRUPTS
EF58 1E                       PUSH    DS
EF59 50                       PUSH    AX
EF5A E8FC0A                   CALL    DDS
EF5D 800E3E0080               OR      SEEK_STATUS,INT_FLAG
EF62 B020                     MOV     AL,20H                  ; END OF INTERRUPT MARKER
EF64 E620                     OUT     20H,AL                  ; INTERRUPT CONTROL PORT
EF66 58                       POP     AX
EF67 1F                       POP     DS                      ; RECOVER SYSTEM
EF68 CF                       IRET                            ; RETURN FROM INTERRUPT
                      DISK_INT        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                                                 :
                      ;----------------------------------------------------------------------
EF69                  RESULTS PROC    NEAR
EF69 FC                       CLD
EF6A BF4200                   MOV     DI,OFFSET NEC_STATUS    ; POINTER TO DATA AREA
EF6D 51                       PUSH    CX                      ; SAVE COUNTER
EF6E 52                       PUSH    DX
EF6F 53                       PUSH    BX
EF70 B307                     MOV     BL,7                    ; MAX STATUS BYTES

                      ;----- WAIT FOR REQUEST FOR MASTER

EF72                  J38:                                    ; INPUT_LOOP
EF72 33C9                     XOR     CX,CX                   ; COUNTER
EF74 BAF403                   MOV     DX,03F4H                ; STATUS PORT
EF77                  J39:                                    ; WAIT FOR MASTER
EF77 EC                       IN      AL,DX                   ; GET STATUS
EF78 A880                     TEST    AL,080H                 ; MASTER READY
EF7A 750C                     JNZ     J40A                    ; TEST_DIR
EF7C E2F9                     LOOP    J39                     ; WAIT_MASTER
EF7E 800E410080               OR      DISKETTE_STATUS,TIME_OUT
EF83                  J40:                                    ; RESULTS_ERROR
EF83 F9                       STC                             ; SET ERROR RETURN
EF84 5B                       POP     BX
EF85 5A                       POP     DX
EF86 59                       POP     CX
EF87 C3                       RET

                      ;----- TEST THE DIRECTION BIT

EF88                  J40A:
EF88 EC                       IN      AL,DX                   ; GET STATUS REG AGAIN
EF89 A840                     TEST    AL,040H                 ; TEST DIRECTION BIT
EF8B 7507                     JNZ     J42                     ; OK TO READ STATUS
EF8D                  J41:                                    ; NEC_FAIL
EF8D 800E410020               OR      DISKETTE_STATUS,BAD_NEC
EF92 EBEF                     JMP     J40                     ; RESULTS_ERROR

                      ;----- READ IN THE STATUS

EF94                  J42:                                    ; INPUT_STAT
EF94 42                       INC     DX                      ; POINT AT DATA PORT
EF95 EC                       IN      AL,DX                   ; GET THE DATA
EF96 8805                     MOV     [DI],AL                 ; STORE THE BYTE
EF98 47                       INC     DI                      ; INCREMENT THE POINTER
EF99 B90A00                   MOV     CX,10                   ; LOOP TO KILL TIME FOR NEC
EF9C E2FE             J43:    LOOP    J43
EF9E 4A                       DEC     DX                      ; POINT AT STATUS PORT
EF9F EC                       IN      AL,DX                   ; GET STATUS
EFA0 A810                     TEST    AL,010H                 ; TEST FOR NEC STILL BUSY
```

```
EFA2 7406               JZ      J44                     ; RESULTS DONE
EFA4 FECB               DEC     BL                      ; DECREMENT THE STATUS COUNTER
EFA6 75CA               JNZ     J38                     ; GO BACK FOR MORE
EFA8 EBE3               JMP     J41                     ; CHIP HAS FAILED

                ;----- RESULT OPERATION IS DONE

EFAA            J44:
EFAA 5B                 POP     BX
EFAB 5A                 POP     DX
EFAC 59                 POP     CX                      ; RECOVER REGISTERS
EFAD 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                              :
                ;-----------------------------------------------------------------
EFAE            NUM_TRANS       PROC    NEAR
EFAE A04500             MOV     AL,NEC_STATUS+3         ; GET CYLINDER ENDED UP ON
EFB1 3AC5               CMP     AL,CH                   ; SAME AS WE STARTED
EFB3 A04700             MOV     AL,NEC_STATUS+5         ; GET ENDING SECTOR
EFB6 740A               JZ      J45                     ; IF ON SAME CYL, THEN NO ADJUST
EFB8 BB0800             MOV     BX,8
EFBB E8AEFE             CALL    GET_PARM                ; GET EOT VALUE
EFBE 8AC4               MOV     AL,AH                   ;  INTO AL
EFC0 FEC0               INC     AL                      ; USE EOT+1 FOR CALCULATION
EFC2            J45:
EFC2 2AC1               SUB     AL,CL                   ; SUBTRACT START FROM END
EFC4 C3                 RET
                NUM_TRANS       ENDP
                RESULTS ENDP
                ;--------------------------------------------------------------------------
                ; 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     :
                ;       DISK_POINTER TO IT.                                                :
                ;--------------------------------------------------------------------------
EFC7                    ORG     0EFC7H
EFC7            DISK_BASE       LABEL   BYTE
EFC7 CF                 DB      11001111B       ; SRT=C, HD UNLOAD=0F - 1ST SPECIFY BYTE
EFC8 02                 DB      2               ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
EFC9 25                 DB      MOTOR_WAIT      ; WAIT AFTER OPN TIL MOTOR OFF
EFCA 02                 DB      2               ; 512 BYTES/SECTOR
EFCB 08                 DB      8               ; EOT ( LAST SECTOR ON TRACK)
EFCC 2A                 DB      02AH            ; GAP LENGTH
EFCD FF                 DB      0FFH            ; DTL
EFCE 50                 DB      050H            ; GAP LENGTH FOR FORMAT
EFCF F6                 DB      0F6H            ; FILL BYTE FOR FORMAT
EFD0 19                 DB      25              ; HEAD SETTLE TIME (MILLISECONDS)
EFD1 04                 DB      4               ; MOTOR START TIME (1/8 SECONDS)
```

SECTION 5

```
LOC  OBJECT          LINE   SOURCE (BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82

                            ;--- 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 CHANGED TO CAUSE DIFFERENT         :
                            ; TIME-OUT WAITS. DEFAULT=20                                               :
                            ;                                                                          :
                            ; REGISTERS       AH IS MODIFIED                                           :
                            ;                 ALL OTHERS UNCHANGED                                     :
                            ;---------------------------------------------------------------------------
                                    ASSUME  CS:CODE,DS:DATA
EFD2                                ORG     0EFD2H
EFD2                        PRINTER_IO      PROC    FAR
EFD2 FB                             STI                             ; INTERRUPTS BACK ON
EFD3 1E                             PUSH    DS                      ; SAVE SEGMENT
EFD4 52                             PUSH    DX
EFD5 56                             PUSH    SI
EFD6 51                             PUSH    CX
EFD7 53                             PUSH    BX
EFD8 E87E0A                         CALL    DDS
EFDB 8BF2                           MOV     SI,DX                   ; GET PRINTER PARM
EFDD 8A5C78                         MOV     BL,PRINT_TIM_OUT[SI]    ; LOAD TIME-OUT PARM
EFE0 D1E6                           SHL     SI,1                    ; WORD OFFSET INTO TABLE
EFE2 8B5408                         MOV     DX,PRINTER_BASE[SI]     ; GET BASE ADDRESS FOR PRINTER CARD
EFE5 0BD2                           OR      DX,DX                   ; TEST DX FOR ZERO,
                                                                    ;  INDICATING NO PRINTER
EFE7 740C                           JZ      B1                      ; RETURN
EFE9 0AE4                           OR      AH,AH                   ; TEST FOR (AH)=0
EFEB 740E                           JZ      B2                      ; PRINT_AL
EFED FECC                           DEC     AH                      ; TEST FOR (AH)=1
EFEF 743F                           JZ      B8                      ; INIT_PRT
EFF1 FECC                           DEC     AH                      ; TEST FOR (AH)=2
EFF3 7428                           JZ      B5                      ; PRINTER STATUS
EFF5                        B1:                                     ; RETURN
EFF5 5B                             POP     BX
EFF6 59                             POP     CX
EFF7 5E                             POP     SI                      ; RECOVER REGISTERS
EFF8 5A                             POP     DX                      ; RECOVER REGISTERS
EFF9 1F                             POP     DS
EFFA CF                             IRET

                            ;------ PRINT THE CHARACTER IN (AL)

EFFB                        B2:
EFFB 50                             PUSH    AX                      ; SAVE VALUE TO PRINT
EFFC EE                             OUT     DX,AL                   ; OUTPUT CHAR TO PORT
EFFD 42                             INC     DX                      ; POINT TO STATUS PORT
EFFE                        B3:
EFFE 2BC9                           SUB     CX,CX                   ; WAIT_BUSY
F000                        B3_1:
F000 EC                             IN      AL,DX                   ; GET STATUS
F001 8AE0                           MOV     AH,AL                   ; STATUS TO AH ALSO
F003 A880                           TEST    AL,80H                  ; IS THE PRINTER CURRENTLY BUSY
F005 750E                           JNZ     B4                      ; OUT_STROBE
F007 E2F7                           LOOP    B3_1                    ; TRY AGAIN
F009 FECB                           DEC     BL                      ; DROP LOOP COUNT
F00B 75F1                           JNZ     B3                      ; GO TILL TIMEOUT ENDS
F00D 80CC01                         OR      AH,1                    ; SET ERROR FLAG
F010 80E4F9                         AND     AH,0F9H                 ; TURN OFF THE OTHER BITS
F013 EB13                           JMP     SHORT B7                ; RETURN WITH ERROR FLAG SET
F015                        B4:                                     ; OUT_STROBE
F015 B00D                           MOV     AL,0DH                  ; SET THE STROBE HIGH
F017 42                             INC     DX                      ; STROBE IS BIT 0 OF PORT C OF 8255
F018 EE                             OUT     DX,AL
F019 B00C                           MOV     AL,0CH                  ; SET THE STROBE LOW
F01B EE                             OUT     DX,AL
F01C 58                             POP     AX                      ; RECOVER THE OUTPUT CHAR

                            ;------ PRINTER STATUS

F01D                        B5:
F01D 50                             PUSH    AX                      ; SAVE AL REG
F01E                        B6:
F01E 8B5408                         MOV     DX,PRINTER_BASE[SI]
F021 42                             INC     DX
F022 EC                             IN      AL,DX                   ; GET PRINTER STATUS
F023 8AE0                           MOV     AH,AL
F025 80E4F8                         AND     AH,0F8H                 ; TURN OFF UNUSED BITS
F028                        B7:                                     ; STATUS_SET
F028 5A                             POP     DX                      ; RECOVER AL REG
F029 8AC2                           MOV     AL,DL                   ; GET CHARACTER INTO AL
F02B 80F448                         XOR     AH,48H                  ; FLIP A COUPLE OF BITS
F02E EBC5                           JMP     B1                      ; RETURN FROM ROUTINE
```

```
                 
                 ;------ INITIALIZE THE PRINTER PORT
                 
F030             B8:
F030 50                  PUSH    AX              ; SAVE AL
F031 42                  INC     DX              ; POINT TO OUTPUT PORT
F032 42                  INC     DX
F033 B008                MOV     AL,8            ; SET INIT LINE LOW
F035 EE                  OUT     DX,AL
F036 B8E803              MOV     AX,1000
F039             B9:                             ; INIT_LOOP
F039 48                  DEC     AX              ; LOOP FOR RESET TO TAKE
F03A 75FD                JNZ     B9              ; INIT_LOOP
F03C B00C                MOV     AL,0CH          ; NO INTERRUPTS, NON AUTO LF,
                                                 ;  INIT HIGH
F03E EE                  OUT     DX,AL
F03F EBDD                JMP     B6              ; PRT_STATUS_1
                 PRINTER_IO      ENDP
```


SECTION 5

```

;--- 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)       :
;               *** NOTE BW MODES OPERATE SAME AS COLOR MODES, BUT            :
;                       COLOR BURST IS NOT ENABLED                            :
;       (AH)=1  SET CURSOR TYPE                                               :
;               (CH) =  BITS 4-0 = START LINE FOR CURSOR                      :
;                       ** HARDWARE WILL ALWAYS CAUSE BLIN                    :
;                       ** 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 VAL (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) = PALETTE 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 PALETTE 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 PALETTE 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                                     :
```

```
                 ;                                                              :
                 ; 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             :
                 ;                                                              :
                 ;       CS,SS,DS,ES,BX,CX,DX PRESERVED DURING CALL             :
                 ;       ALL OTHERS DESTROYED                                   :
                 ;--------------------------------------------------------------
                         ASSUME  CS:CODE,DS:DATA,ES:VIDEO_RAM
F045                     ORG     0F045H
F045             M1      LABEL   WORD                    ; TABLE OF ROUTINES WITHIN VIDEO I/O
F045 FCF0                DW      OFFSET  SET_MODE
F047 CDF1                DW      OFFSET  SET_CTYPE
F049 EEF1                DW      OFFSET  SET_CPOS
F04B 39F2                DW      OFFSET  READ_CURSOR
F04D 9CF7                DW      OFFSET  READ_LPEN
F04F 17F2                DW      OFFSET  ACT_DISP_PAGE
F051 96F2                DW      OFFSET  SCROLL_UP
F053 38F3                DW      OFFSET  SCROLL_DOWN
F055 74F3                DW      OFFSET  READ_AC_CURRENT
F057 B9F3                DW      OFFSET  WRITE_AC_CURRENT
F059 ECF3                DW      OFFSET  WRITE_C_CURRENT
F05B 4EF2                DW      OFFSET  SET_COLOR
F05D 2FF4                DW      OFFSET  WRITE_DOT
F05F 1EF4                DW      OFFSET  READ_DOT
F061 18F7                DW      OFFSET  WRITE_TTY
F063 74F2                DW      OFFSET  VIDEO_STATE
  0020           M1L     EQU     $-M1

F065                     ORG     0F065H
F065             VIDEO_IO        PROC    NEAR
F065 FB                  STI                             ; INTERRUPTS BACK ON
F066 FC                  CLD                             ; SET DIRECTION FORWARD
F067 06                  PUSH    ES
F068 1E                  PUSH    DS                      ; SAVE SEGMENT REGISTERS
F069 52                  PUSH    DX
F06A 51                  PUSH    CX
F06B 53                  PUSH    BX
F06C 56                  PUSH    SI
F06D 57                  PUSH    DI
F06E 50                  PUSH    AX                      ; SAVE AX VALUE
F06F 8AC4                MOV     AL,AH                   ; GET INTO LOW BYTE
F071 32E4                XOR     AH,AH                   ; ZERO TO HIGH BYTE
F073 D1E0                SAL     AX,1                    ; *2 FOR TABLE LOOKUP
F075 8BF0                MOV     SI,AX                   ; PUT INTO SI FOR BRANCH
F077 3D2000              CMP     AX,M1L                  ; TEST FOR WITHIN RANGE
F07A 7204                JB      M2                      ; BRANCH AROUND BRANCH
F07C 58                  POP     AX                      ; THROW AWAY THE PARAMETER
F07D E94501              JMP     VIDEO_RETURN            ; DO NOTHING IF NOT IN RANGE
F080             M2:
F080 E8D609              CALL    DDS
F083 B800B8              MOV     AX,0B800H               ; SEGMENT FOR COLOR CARD
F086 8B3E1000            MOV     DI,EQUIP_FLAG           ; GET EQUIPMENT SETTING
F08A 81E73000            AND     DI,30H                  ; ISOLATE CRT SWITCHES
F08E 83FF30              CMP     DI,30H                  ; IS SETTING FOR BW CARD?
F091 7502                JNE     M3
F093 B4B0                MOV     AH,0B0H                 ; SEGMENT FOR BW CARD
F095             M3:
F095 8EC0                MOV     ES,AX                   ; SET UP TO POINT AT VIDEO RAM AREAS
F097 58                  POP     AX                      ; RECOVER VALUE
F098 8A264900            MOV     AH,CRT_MODE             ; GET CURRENT MODE INTO AH
F09C 2EFFA445F0          JMP     WORD PRT CS:[SI+OFFSET M1]
                 VIDEO_IO        ENDP
```

SECTION 5

```
LOC OBJECT            LINE  SOURCE (BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82

                            ;----------------------------------------------------
                            ; SET_MODE                                          :
                            ;       THIS ROUTINE INITIALIZES THE ATTACHMENT TO   :
                            ;       THE SELECTED MODE.  THE SCREEN IS BLANKED.   :
                            ; INPUT                                             :
                            ;       (AL) = MODE SELECTED (RANGE 0-9)            :
                            ; OUTPUT                                            :
                            ;       NONE                                        :
                            ;----------------------------------------------------

                            ;----- TABLES FOR USE IN SETTING OF MODE

F0A4                                ORG     0F0A4H
F0A4                        VIDEO_PARMS     LABEL   BYTE
                            ;----- INIT_TABLE
F0A4 38                             DB      38H,28H,2DH,0AH,1FH,6,19H      ; SET UP FOR 40X25
F0A5 28
F0A6 2D
F0A7 0A
F0A8 1F
F0A9 06
F0AA 19
F0AB 1C                             DB      1CH,2,7,6,7
F0AC 02
F0AD 07
F0AE 06
F0AF 07
F0B0 00                             DB      0,0,0,0
F0B1 00
F0B2 00
F0B3 00
  0010                      M4      EQU     $-VIDEO_PARMS

F0B4 71                             DB      71H,50H,5AH,0AH,1FH,6,19H      ; SET UP FOR 80X25
F0B5 50
F0B6 5A
F0B7 0A
F0B8 1F
F0B9 06
F0BA 19
F0BB 1C                             DB      1CH,2,7,6,7
F0BC 02
F0BD 07
F0BE 06
F0BF 07
F0C0 00                             DB      0,0,0,0
F0C1 00
F0C2 00
F0C3 00

F0C4 38                             DB      38H,28H,2DH,0AH,7FH,6,64H      ; SET UP FOR GRAPHICS
F0C5 28
F0C6 2D
F0C7 0A
F0C8 7F
F0C9 06
F0CA 64
F0CB 70                             DB      70H,2,1,6,7
F0CC 02
F0CD 01
F0CE 06
F0CF 07
F0D0 00                             DB      0,0,0,0
F0D1 00
F0D2 00
F0D3 00

F0D4 61                             DB      61H,50H,52H,0FH,19H,6,19H      ; SET UP FOR 80X25 B&W CARD
F0D5 50
F0D6 52
F0D7 0F
F0D8 19
F0D9 06
F0DA 19
F0DB 19                             DB      19H,2,0DH,0BH,0CH
F0DC 02
F0DD 0D
F0DE 0B
F0DF 0C
F0E0 00                             DB      0,0,0,0
F0E1 00
F0E2 00
F0E3 00

F0E4                        M5      LABEL   WORD                   ; TABLE OF REGEN LENGTHS
F0E4 0008                           DW      2048                   ; 40X25
F0E6 0010                           DW      4096                   ; 80X25
F0E8 0040                           DW      16384                  ; GRAPHICS
F0EA 0040                           DW      16384

                            ;----- COLUMNS

F0EC                        M6      LABEL   BYTE
F0EC 28                             DB      40,40,80,80,40,40,80,80
F0ED 28
F0EE 50
F0EF 50
F0F0 28
F0F1 28
F0F2 50
F0F3 50

                            ;----- C_REG_TAB

F0F4                        M7      LABEL   BYTE                   ; TABLE OF MODE SETS
F0F4 2C                             DB      2CH,28H,2DH,29H,2AH,2EH,1EH,29H
F0F5 28
F0F6 2D
F0F7 29
F0F8 2A
F0F9 2E
F0FA 1E
F0FB 29
```

```
F0FC                  SET_MODE        PROC    NEAR
F0FC BAD403                   MOV     DX,03D4H                ; ADDRESS OF COLOR CARD
F0FF B300                     MOV     BL,0                    ; MODE SET FOR COLOR CARD
F101 83FF30                   CMP     DI,30H                  ; IS BW CARD INSTALLED
F104 7506                     JNE     M8                      ; OK WITH COLOR
F106 B007                     MOV     AL,7                    ; INDICATE BW CARD MODE
F108 B2B4                     MOV     DL,0B4H                 ; ADDRESS OF BW CARD (3B4)
F10A FEC3                     INC     BL                      ; MODE SET FOR BW CARD
F10C                  M8:
F10C 8AE0                     MOV     AH,AL                   ; SAVE MODE IN AH
F10E A24900                   MOV     CRT_MODE,AL             ; SAVE IN GLOBAL VARIABLE
F111 89166300                 MOV     ADDR_6845,DX            ; SAVE ADDRESS OF BASE
F115 1E                       PUSH    DS                      ; SAVE POINTER TO DATA SEGMENT
F116 50                       PUSH    AX                      ; SAVE MODE
F117 52                       PUSH    DX                      ; SAVE OUTPUT PORT VALUE
F118 83C204                   ADD     DX,4                    ; POINT TO CONTROL REGISTER
F11B 8AC3                     MOV     AL,BL                   ; GET MODE SET FOR CARD
F11D EE                       OUT     DX,AL                   ; RESET VIDEO
F11E 5A                       POP     DX                      ; BACK TO BASE REGISTER
F11F 2BC0                     SUB     AX,AX                   ; SET UP FOR ABS0 SEGMENT
F121 8ED8                     MOV     DS,AX                   ; ESTABLISH VECTOR TABLE ADDRESSING
                              ASSUME  DS:ABS0
F123 C51E7400                 LDS     BX,PARM_PTR             ; GET POINTER TO VIDEO PARMS
F127 58                       POP     AX                      ; RECOVER PARMS
                              ASSUME  DS:CODE
F128 B91000                   MOV     CX,M4                   ; LENGTH OF EACH ROW OF TABLE
F12B 80FC02                   CMP     AH,2                    ; DETERMINE WHICH ONE TO USE
F12E 7210                     JC      M9                      ; MODE IS 0 OR 1
F130 03D9                     ADD     BX,CX                   ; MOVE TO NEXT ROW OF INIT TABLE
F132 80FC04                   CMP     AH,4
F135 7209                     JC      M9                      ; MODE IS 2 OR 3
F137 03D9                     ADD     BX,CX                   ; MOVE TO GRAPHICS ROW OF INIT_TABLE
F139 80FC07                   CMP     AH,7
F13C 7202                     JC      M9                      ; MODE IS 4,5, OR 6
F13E 03D9                     ADD     BX,CX                   ; MOVE TO BW CARD ROW OF INIT_TABLE

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

F140                  M9:                                     ; OUT_INIT
F140 50                       PUSH    AX                      ; SAVE MODE IN AH
F141 32E4                     XOR     AH,AH                   ; AH WILL SERVE AS REGISTER
                                                              ;  NUMBER DURING LOOP

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

F143                  M10:                                    ; INIT LOOP
F143 8AC4                     MOV     AL,AH                   ; GET 6845 REGISTER NUMBER
F145 EE                       OUT     DX,AL
F146 42                       INC     DX                      ; POINT TO DATA PORT
F147 FEC4                     INC     AH                      ; NEXT REGISTER VALUE
F149 8A07                     MOV     AL,[BX]                 ; GET TABLE VALUE
F14B EE                       OUT     DX,AL                   ; OUT TO CHIP
F14C 43                       INC     BX                      ; NEXT IN TABLE
F14D 4A                       DEC     DX                      ; BACK TO POINTER REGISTER
F14E E2F3                     LOOP    M10                     ; DO THE WHOLE TABLE
F150 58                       POP     AX                      ; GET MODE BACK
F151 1F                       POP     DS                      ; RECOVER SEGMENT VALUE
                              ASSUME  DS:DATA

                      ;----- FILL REGEN AREA WITH BLANK

F152 33FF                     XOR     DI,DI                   ; SET UP POINTER FOR REGEN
F154 893E4E00                 MOV     CRT_START,DI            ; START ADDRESS SAVED IN GLOBAL
F158 C606620000               MOV     ACTIVE_PAGE,0           ; SET PAGE VALUE
F15D B90020                   MOV     CX,8192                 ; NUMBER OF WORDS IN COLOR CARD
F160 80FC04                   CMP     AH,4                    ; TEST FOR GRAPHICS
F163 720B                     JC      M12                     ; NO_GRAPHICS_INIT
F165 80FC07                   CMP     AH,7                    ; TEST FOR BW_CARD
F168 7404                     JE      M11                     ; BW_CARD_INIT
F16A 33C0                     XOR     AX,AX                   ; FILL FOR GRAPHICS MODE
F16C EB05                     JMP     SHORT M13               ; CLEAR_BUFFER
F16E                  M11:                                    ; BW_CARD_INIT
F16E B508                     MOV     CH,08H                  ; BUFFER SIZE ON BW CARD
F170                  M12:                                    ; NO_GRAPHICS_INIT
F170 B82007                   MOV     AX,' '+7*256            ; FILL CHAR FOR ALPHA
F173                  M13:                                    ; CLEAR_BUFFER
F173 F3                       REP     STOSW                   ; FILL THE REGEN BUFFER WITH BLANKS
F174 AB

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

F175 C70660000706             MOV     CURSOR_MODE,607H        ; SET CURRENT CURSOR MODE
F17B A04900                   MOV     AL,CRT_MODE             ; GET THE MODE
F17E 32E4                     XOR     AH,AH                   ;  INTO AX REGISTER
F180 8BF0                     MOV     SI,AX                   ; TABLE POINTER, INDEXED BY MODE
F182 8B166300                 MOV     DX,ADDR_6845            ; PREPARE TO OUTPUT TO
                                                              ;  VIDEO ENABLE PORT
F186 83C204                   ADD     DX,4
F189 2E8A84F4F0               MOV     AL,CS:[SI+OFFSET M7]
F18E EE                       OUT     DX,AL                   ; SET VIDEO ENABLE PORT
F18F A26500                   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

F192 2E8A84ECF0               MOV     AL,CS:[SI+OFFSET M6]
F197 32E4                     XOR     AH,AH
F199 A34A00                   MOV     CRT_COLS,AX             ; NUMBER OF COLUMNS IN THIS SCREEN

                      ;----- SET CURSOR POSITIONS

F19C 81E60E00                 AND     SI,0EH                  ; WORD OFFSET INTO CLEAR LENGTH TABLE
F1A0 2E8B8CE4F0               MOV     CX,CS:[SI+OFFSET M5]    ; LENGTH TO CLEAR
F1A5 890E4C00                 MOV     CRT_LEN,CX              ; SAVE LENGTH OF CRT -- NOT USED FOR BW
F1A9 B90800                   MOV     CX,8                    ; CLEAR ALL CURSOR POSITIONS
F1AC BF5000                   MOV     DI,OFFSET CURSOR_POSN
F1AF 1E                       PUSH    DS                      ; ESTABLISH SEGMENT
F1B0 07                       POP     ES                      ;   ADDRESSING
F1B1 33C0                     XOR     AX,AX
F1B3 F3                       REP     STOSW                   ; FILL WITH ZEROES
F1B4 AB
```

SECTION 5

```

                ;----- SET UP OVERSCAN REGISTER

F1B5 42                 INC     DX                      ; SET OVERSCAN PORT TO A DEFAULT
F1B6 B030               MOV     AL,30H                  ; VALUE OF 30H FOR ALL MODES
                                                        ;  EXCEPT 640X200
F1B8 803E490006         CMP     CRT_MODE,6              ; SEE IF THE MODE IS 640X200 BW
F1BD 7502               JNZ     M14                     ; IF IT ISNT 640X200, THEN GOTO REGULAR
F1BF B03F               MOV     AL,3FH                  ; IF IT IS 640X200, THEN PUT IN 3FH
F1C1            M14:
F1C1 EE                 OUT     DX,AL                   ; OUTPUT THE CORRECT VALUE TO 3D9 PORT
F1C2 A26600             MOV     CRT_PALETTE,AL          ; SAVE THE VALUE FOR FUTURE USE

                ;----- NORMAL RETURN FROM ALL VIDEO RETURNS

F1C5            VIDEO_RETURN:
F1C5 5F                 POP     DI
F1C6 5E                 POP     SI
F1C7 5B                 POP     BX
F1C8            M15:                                    ; VIDEO_RETURN_C
F1C8 59                 POP     CX
F1C9 5A                 POP     DX
F1CA 1F                 POP     DS
F1CB 07                 POP     ES                      ; RECOVER SEGMENTS
F1CC CF                 IRET                            ; ALL DONE
                SET_MODE        ENDP
                ;--------------------------------------------------------------
                ; SET_CTYPE                                                    :
                ;       THIS ROUTINE SETS THE CURSOR VALUE                     :
                ; INPUT                                                        :
                ;       (CX) HAS CURSOR VALUE CH-START LINE, CL-STOP LINE      :
                ; OUTPUT                                                       :
                ;       NONE                                                   :
                ;--------------------------------------------------------------
F1CD            SET_CTYPE       PROC    NEAR
F1CD B40A               MOV     AH,10                   ; 6845 REGISTER FOR CURSOR SET
F1CF 890E6000           MOV     CURSOR_MODE,CX          ; SAVE IN DATA AREA
F1D3 E80200             CALL    M16                     ; OUTPUT CX REG
F1D6 EBED               JMP     VIDEO_RETURN

                ;----- THIS ROUTINE OUTPUTS THE CX REGISTER TO THE 6845 REGS NAMED IN AH

F1D8            M16:
F1D8 8B166300           MOV     DX,ADDR_6845            ; ADDRESS REGISTER
F1DC 8AC4               MOV     AL,AH                   ; GET VALUE
F1DE EE                 OUT     DX,AL                   ; REGISTER SET
F1DF 42                 INC     DX                      ; DATA REGISTER
F1E0 8AC5               MOV     AL,CH                   ; DATA
F1E2 EE                 OUT     DX,AL
F1E3 4A                 DEC     DX
F1E4 8AC4               MOV     AL,AH
F1E6 FEC0               INC     AL                      ; POINT TO OTHER DATA REGISTER
F1E8 EE                 OUT     DX,AL                   ; SET FOR SECOND REGISTER
F1E9 42                 INC     DX
F1EA 8AC1               MOV     AL,CL                   ; SECOND DATA VALUE
F1EC EE                 OUT     DX,AL
F1ED C3                 RET                             ; ALL DONE
                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                     :
                ;----------------------------------------------
F1EE            SET_CPOS        PROC    NEAR
F1EE 8ACF               MOV     CL,BH
F1F0 32ED               XOR     CH,CH                   ; ESTABLISH LOOP COUNT
F1F2 D1E1               SAL     CX,1                    ; WORD OFFSET
F1F4 8BF1               MOV     SI,CX                   ; USE INDEX REGISTER
F1F6 895450             MOV     [SI+OFFSET CURSOR_POSN],DX      ; SAVE THE POINTER
F1F9 383E6200           CMP     ACTIVE_PAGE,BH
F1FD 7505               JNZ     M17                     ; SET_CPOS_RETURN
F1FF 8BC2               MOV     AX,DX                   ; GET ROW/COLUMN TO AX
F201 E80200             CALL    M18                     ; CURSOR_SET
F204            M17:                                    ; SET_CPOS_RETURN
F204 EBBF               JMP     VIDEO_RETURN
                SET_CPOS        ENDP

                ;----- SET CURSOR POSITION, AX HAS ROW/COLUMN FOR CURSOR

F206            M18     PROC    NEAR
F206 E87C00             CALL    POSITION                ; DETERMINE LOCATION IN REGEN BUFFER
F209 8BC8               MOV     CX,AX
F20B 030E4E00           ADD     CX,CRT_START            ; ADD IN THE START ADDR FOR THIS PAGE
F20F D1F9               SAR     CX,1                    ; DIVIDE BY 2 FOR CHAR ONLY COUNT
F211 B40E               MOV     AH,14                   ; REGISTER NUMBER FOR CURSOR
F213 E8C2FF             CALL    M16                     ; OUTPUT THE VALUE TO THE 6845
F216 C3                 RET
                M18     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                 :
                   ;--------------------------------------------------------------
F217               ACT_DISP_PAGE   PROC    NEAR
F217 A26200                MOV     ACTIVE_PAGE,AL          ; SAVE ACTIVE PAGE VALUE
F21A 8B0E4C00              MOV     CX,CRT_LEN              ; GET SAVED LENGTH OF REGEN BUFFER
F21E 98                    CBW                             ; CONVERT AL TO WORD
F21F 50                    PUSH    AX                      ; SAVE PAGE VALUE
F220 F7E1                  MUL     CX                      ; DISPLAY PAGE TIMES REGEN LENGTH
F222 A34E00                MOV     CRT_START,AX            ; SAVE START ADDRESS FOR
                                                           ;  LATER REQUIREMENTS
F225 8BC8                  MOV     CX,AX                   ; START ADDRESS TO CX
F227 D1F9                  SAR     CX,1                    ; DIVIDE BY 2 FOR 6845 HANDLING
F229 B40C                  MOV     AH,12                   ; 6845 REGISTER FOR START ADDRESS
F22B E8AAFF                CALL    M16
F22E 5B                    POP     BX                      ; RECOVER PAGE VALUE
F22F D1E3                  SAL     BX,1                    ; *2 FOR WORD OFFSET
F231 8B4750                MOV     AX,[BX + OFFSET CURSOR_POSN]    ; GET CURSOR FOR THIS PAGE
F234 E8CFFF                CALL    M18                     ; SET THE CURSOR POSITION
F237 EB8C                  JMP     SHORT VIDEO_RETURN
                   ACT_DISP_PAGE   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                               :
                   ;--------------------------------------------------------------
F239               READ_CURSOR     PROC    NEAR
F239 8ADF                  MOV     BL,BH
F23B 32FF                  XOR     BH,BH
F23D D1E3                  SAL     BX,1                    ; WORD OFFSET
F23F 8B5750                MOV     DX,[BX+OFFSET CURSOR_POSN]
F242 8B0E6000              MOV     CX,CURSOR_MODE
F246 5F                    POP     DI
F247 5E                    POP     SI
F248 5B                    POP     BX
F249 58                    POP     AX                      ; DISCARD SAVED CX AND DX
F24A 58                    POP     AX
F24B 1F                    POP     DS
F24C 07                    POP     ES
F24D CF                    IRET
                   READ_CURSOR     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 PALETTE 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                                      :
                   ;---------------------------------------------------------------------------
F24E               SET_COLOR       PROC    NEAR
F24E 8B166300              MOV     DX,ADDR_6845            ; I/O PORT FOR PALETTE
F252 83C205                ADD     DX,5                    ; OVERSCAN PORT
F255 A06600                MOV     AL,CRT_PALETTE          ; GET THE CURRENT PALETTE VALUE
F258 0AFF                  OR      BH,BH                   ; IS THIS COLOR 0?
F25A 750E                  JNZ     M20                     ; OUTPUT COLOR 1

                   ;----- HANDLE COLOR 0 BY SETTING THE BACKGROUND COLOR

F25C 24E0                  AND     AL,0E0H                 ; TURN OFF LOW 5 BITS OF CURRENT
F25E 80E31F                AND     BL,01FH                 ; TURN OFF HIGH 3 BITS OF INPUT VALUE
F261 0AC3                  OR      AL,BL                   ; PUT VALUE INTO REGISTER
F263               M19:                                    ; OUTPUT THE PALETTE
F263 EE                    OUT     DX,AL                   ; OUTPUT COLOR SELECTION TO 3D9 PORT
F264 A26600                MOV     CRT_PALETTE,AL          ; SAVE THE COLOR VALUE
F267 E95BFF                JMP     VIDEO_RETURN

                   ;----- HANDLE COLOR 1 BY SELECTING THE PALETTE TO BE USED

F26A               M20:
F26A 24DF                  AND     AL,0DFH                 ; TURN OFF PALETTE SELECT BIT
F26C D0EB                  SHR     BL,1                    ; TEST THE LOW ORDER BIT OF BL
F26E 73F3                  JNC     M19                     ; ALREADY DONE
F270 0C20                  OR      AL,20H                  ; TURN ON PALETTE SELECT BIT
F272 EBEF                  JMP     M19                     ; GO DO IT
                   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                    :
                   ;----------------------------------------------
F274               VIDEO_STATE     PROC    NEAR
F274 8A264A00              MOV     AH,BYTE PTR CRT_COLS    ; GET NUMBER OF COLUMNS
F278 A04900                MOV     AL,CRT_MODE             ; CURRENT MODE
F27B 8A3E6200              MOV     BH,ACTIVE_PAGE          ; GET CURRENT ACTIVE PAGE
F27F 5F                    POP     DI                      ; RECOVER REGISTERS
F280 5E                    POP     SI
F281 59                    POP     CX                      ; DISCARD SAVED BX
F282 E943FF                JMP     M15                     ; RETURN TO CALLER
                   VIDEO_STATE     ENDP
```

```
LOC OBJECT              LINE  SOURCE (BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82

                              ;--------------------------------------------------------
                              ; 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     :
                              ;--------------------------------------------------------
F285                          POSITION        PROC    NEAR
F285 53                               PUSH    BX                      ; SAVE REGISTER
F286 8BD8                             MOV     BX,AX
F288 8AC4                             MOV     AL,AH                   ; ROWS TO AL
F28A F6264A00                         MUL     BYTE PTR CRT_COLS       ; DETERMINE BYTES TO ROW
F28E 32FF                             XOR     BH,BH
F290 03C3                             ADD     AX,BX                   ; ADD IN COLUMN VALUE
F292 D1E0                             SAL     AX,1                    ; * 2 FOR ATTRIBUTE BYTES
F294 5B                               POP     BX
F295 C3                               RET
                              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
F296                          SCROLL_UP       PROC    NEAR
F296 8AD8                             MOV     BL,AL                   ; SAVE LINE COUNT IN BL
F298 80FC04                           CMP     AH,4                    ; TEST FOR GRAPHICS MODE
F29B 7208                             JC      N1                      ; HANDLE SEPARATELY
F29D 80FC07                           CMP     AH,7                    ; TEST FOR BW CARD
F2A0 7403                             JE      N1
F2A2 E9F001                           JMP     GRAPHICS_UP
F2A5                          N1:                                     ; UP_CONTINUE
F2A5 53                               PUSH    BX                      ; SAVE FILL ATTRIBUTE IN BH
F2A6 8BC1                             MOV     AX,CX                   ; UPPER LEFT POSITION
F2A8 E83700                           CALL    SCROLL_POSITION         ; DO SETUP FOR SCROLL
F2AB 7431                             JZ      N7                      ; BLANK_FIELD
F2AD 03F0                             ADD     SI,AX                   ; FROM ADDRESS
F2AF 8AE6                             MOV     AH,DH                   ; # ROWS IN BLOCK
F2B1 2AE3                             SUB     AH,BL                   ; # ROWS TO BE MOVED
F2B3                          N2:                                     ; ROW_LOOP
F2B3 E87200                           CALL    N10                     ; MOVE ONE ROW
F2B6 03F5                             ADD     SI,BP
F2B8 03FD                             ADD     DI,BP                   ; POINT TO NEXT LINE IN BLOCK
F2BA FECC                             DEC     AH                      ; COUNT OF LINES TO MOVE
F2BC 75F5                             JNZ     N2                      ; ROW_LOOP
F2BE                          N3:                                     ; CLEAR_ENTRY
F2BE 58                               POP     AX                      ; RECOVER ATTRIBUTE IN AH
F2BF B020                             MOV     AL,' '                  ; FILL WITH BLANKS
F2C1                          N4:                                     ; CLEAR_LOOP
F2C1 E86D00                           CALL    N11                     ; CLEAR THE ROW
F2C4 03FD                             ADD     DI,BP                   ; POINT TO NEXT LINE
F2C6 FECB                             DEC     BL                      ; COUNTER OF LINES TO SCROLL
F2C8 75F7                             JNZ     N4                      ; CLEAR_LOOP
F2CA                          N5:                                     ; SCROLL_END
F2CA E88C07                           CALL    DDS
F2CD 803E490007                       CMP     CRT_MODE,7              ; IS THIS THE BLACK AND WHITE CARD
F2D2 7407                             JE      N6                      ; IF SO, SKIP THE MODE RESET
F2D4 A06500                           MOV     AL,CRT_MODE_SET         ; GET THE VALUE OF THE MODE SET
F2D7 BAD803                           MOV     DX,03D8H                ; ALWAYS SET COLOR CARD PORT
F2DA EE                               OUT     DX,AL
F2DB                          N6:                                     ; VIDEO_RET_HERE
F2DB E9E7FE                           JMP     VIDEO_RETURN
F2DE                          N7:                                     ; BLANK_FIELD
F2DE 8ADE                             MOV     BL,DH                   ; GET ROW COUNT
F2E0 EBDC                             JMP     N3                      ; GO CLEAR THAT AREA
                              SCROLL_UP       ENDP

                              ;----- HANDLE COMMON SCROLL SET UP HERE

F2E2                          SCROLL_POSITION PROC    NEAR
F2E2 803E490002                       CMP     CRT_MODE,2              ; TEST FOR SPECIAL CASE HERE
F2E7 7218                             JB      N9                      ; HAVE TO HANDLE 80X25 SEPARATELY
F2E9 803E490003                       CMP     CRT_MODE,3
F2EE 7711                             JA      N9

                              ;----- 80X25 COLOR CARD SCROLL

F2F0 52                               PUSH    DX
F2F1 BADA03                           MOV     DX,3DAH                 ; GUARANTEED TO BE COLOR CARD HERE
F2F4 50                               PUSH    AX
F2F5                          N8:                                     ; WAIT_DISP_ENABLE
F2F5 EC                               IN      AL,DX                   ; GET PORT
F2F6 A808                             TEST    AL,8                    ; WAIT FOR VERTICAL RETRACE
F2F8 74FB                             JZ      N8                      ; WAIT_DISP_ENABLE
F2FA B025                             MOV     AL,25H
F2FC B2D8                             MOV     DL,0D8H                 ; DX=3D8
F2FE EE                               OUT     DX,AL                   ; TURN OFF VIDEO
F2FF 58                               POP     AX                      ; DURING VERTICAL RETRACE
F300 5A                               POP     DX
F301                          N9:
F301 E881FF                           CALL    POSITION                ; CONVERT TO REGEN POINTER
F304 03064E00                         ADD     AX,CRT_START            ; OFFSET OF ACTIVE PAGE
F308 8BF8                             MOV     DI,AX                   ; TO ADDRESS FOR SCROLL
F30A 8BF0                             MOV     SI,AX                   ; FROM ADDRESS FOR SCROLL
F30C 2BD1                             SUB     DX,CX                   ; DX = # ROWS, #COLS IN BLOCK
F30E FEC6                             INC     DH
F310 FEC2                             INC     DL                      ; INCREMENT FOR 0 ORIGIN
F312 32ED                             XOR     CH,CH                   ; SET HIGH BYTE OF COUNT TO ZERO
F314 8B2E4A00                         MOV     BP,CRT_COLS             ; GET NUMBER OF COLUMNS IN DISPLAY
F318 03ED                             ADD     BP,BP                   ; TIMES 2 FOR ATTRIBUTE BYTE
F31A 8AC3                             MOV     AL,BL                   ; GET LINE COUNT
F31C F6264A00                         MUL     BYTE PTR CRT_COLS       ; DETERMINE OFFSET TO FROM ADDRESS
F320 03C0                             ADD     AX,AX                   ; *2 FOR ATTRIBUTE BYTE
F322 06                               PUSH    ES                      ; ESTABLISH ADDRESSING TO REGEN BUFFER
```

```
F323 1F                     POP     DS                      ; FOR BOTH POINTERS
F324 80FB00                 CMP     BL,0                    ; 0 SCROLL MEANS BLANK FIELD
F327 C3                     RET                             ; RETURN WITH FLAGS SET
                  SCROLL_POSITION ENDP

                  ;----- MOVE_ROW

F328              N10       PROC    NEAR
F328 8ACA                   MOV     CL,DL                   ; GET # OF COLS TO MOVE
F32A 56                     PUSH    SI
F32B 57                     PUSH    DI                      ; SAVE START ADDRESS
F32C F3                     REP     MOVSW                   ; MOVE THAT LINE ON SCREEN
F32D A5
F32E 5F                     POP     DI
F32F 5E                     POP     SI                      ; RECOVER ADDRESSES
F330 C3                     RET
                  N10       ENDP

                  ;----- CLEAR_ROW

F331              N11       PROC    NEAR
F331 8ACA                   MOV     CL,DL                   ; GET # COLUMNS TO CLEAR
F333 57                     PUSH    DI
F334 F3                     REP     STOSW                   ; STORE THE FILL CHARACTER
F335 AB
F336 5F                     POP     DI
F337 C3                     RET
                  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                            :
                  ; OUTPUT                                                  :
                  ;         NONE -- SCREEN IS SCROLLED                      :
                  ;---------------------------------------------------------
F338              SCROLL_DOWN       PROC    NEAR
F338 FD                     STD                             ; DIRECTION FOR SCROLL DOWN
F339 8AD8                   MOV     BL,AL                   ; LINE COUNT TO BL
F33B 80FC04                 CMP     AH,4                    ; TEST FOR GRAPHICS
F33E 7208                   JC      N12
F340 80FC07                 CMP     AH,7                    ; TEST FOR BW CARD
F343 7403                   JE      N12
F345 E9A601                 JMP     GRAPHICS_DOWN
F348              N12:                                      ; CONTINUE_DOWN
F348 53                     PUSH    BX                      ; SAVE ATTRIBUTE IN BH
F349 8BC2                   MOV     AX,DX                   ; LOWER RIGHT CORNER
F34B E894FF                 CALL    SCROLL_POSITION         ; GET REGEN LOCATION
F34E 7420                   JZ      N16
F350 2BF0                   SUB     SI,AX                   ; SI IS FROM ADDRESS
F352 8AE6                   MOV     AH,DH                   ; GET TOTAL # ROWS
F354 2AE3                   SUB     AH,BL                   ; COUNT TO MOVE IN SCROLL
F356              N13:
F356 E8CFFF                 CALL    N10                     ; MOVE ONE ROW
F359 2BF5                   SUB     SI,BP
F35B 2BFD                   SUB     DI,BP
F35D FECC                   DEC     AH
F35F 75F5                   JNZ     N13
F361              N14:
F361 58                     POP     AX                      ; RECOVER ATTRIBUTE IN AH
F362 B020                   MOV     AL,' '
F364              N15:
F364 E8CAFF                 CALL    N11                     ; CLEAR ONE ROW
F367 2BFD                   SUB     DI,BP                   ; GO TO NEXT ROW
F369 FECB                   DEC     BL
F36B 75F7                   JNZ     N15
F36D E95AFF                 JMP     N5                      ; SCROLL_END
F370              N16:
F370 8ADE                   MOV     BL,DH
F372 EBED                   JMP     N14
                  SCROLL_DOWN       ENDP
```

SECTION 5

```
LOC OBJECT            LINE  SOURCE (BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82

                            ;--------------------------------------------------------
                            ; 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
F374                        READ_AC_CURRENT PROC    NEAR
F374 80FC04                         CMP     AH,4                    ; IS THIS GRAPHICS
F377 7208                           JC      P1
F379 80FC07                         CMP     AH,7                    ; IS THIS BW CARD
F37C 7403                           JE      P1
F37E E9A802                         JMP     GRAPHICS_READ
F381                        P1:                                     ; READ_AC_CONTINUE
F381 E81A00                         CALL    FIND_POSITION
F384 8BF3                           MOV     SI,BX                   ; ESTABLISH ADDRESSING IN SI

                            ;----- WAIT FOR HORIZONTAL RETRACE

F386 8B166300                       MOV     DX,ADDR_6845            ; GET BASE ADDRESS
F38A 83C206                         ADD     DX,6                    ; POINT AT STATUS PORT
F38D 06                             PUSH    ES
F38E 1F                             POP     DS                      ; GET SEGMENT FOR QUICK ACCESS
F38F                        P2:                                     ; WAIT FOR RETRACE LOW
F38F EC                             IN      AL,DX                   ; GET STATUS
F390 A801                           TEST    AL,1                    ; IS HORZ RETRACE LOW
F392 75FB                           JNZ     P2                      ; WAIT UNTIL IT IS
F394 FA                             CLI                             ; NO MORE INTERRUPTS
F395                        P3:                                     ; WAIT FOR RETRACE HIGH
F395 EC                             IN      AL,DX                   ; GET STATUS
F396 A801                           TEST    AL,1                    ; IS IT HIGH
F398 74FB                           JZ      P3                      ; WAIT UNTIL IT IS
F39A AD                             LODSW                           ; GET THE CHAR/ATTR
F39B E927FE                         JMP     VIDEO_RETURN
                            READ_AC_CURRENT ENDP

F39E                        FIND_POSITION   PROC    NEAR
F39E 8ACF                           MOV     CL,BH                   ; DISPLAY PAGE TO CX
F3A0 32ED                           XOR     CH,CH
F3A2 8BF1                           MOV     SI,CX                   ; MOVE TO SI FOR INDEX
F3A4 D1E6                           SAL     SI,1                    ; * 2 FOR WORD OFFSET
F3A6 8B4450                         MOV     AX,[SI+OFFSET CURSOR_POSN]      ; GET ROW/COLUMN OF THAT PAGE
F3A9 33DB                           XOR     BX,BX                   ; SET START ADDRESS TO ZERO
F3AB E306                           JCXZ    P5                      ; NO_PAGE
F3AD                        P4:                                     ; PAGE_LOOP
F3AD 031E4C00                       ADD     BX,CRT_LEN              ; LENGTH OF BUFFER
F3B1 E2FA                           LOOP    P4
F3B3                        P5:                                     ; NO_PAGE
F3B3 E8CFFE                         CALL    POSITION                ; DETERMINE LOCATION IN REGEN
F3B6 03D8                           ADD     BX,AX                   ; ADD TO START OF REGEN
F3B8 C3                             RET
                            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                                    :
                            ;------------------------------------------------
F3B9                        WRITE_AC_CURRENT        PROC    NEAR
F3B9 80FC04                         CMP     AH,4                    ; IS THIS GRAPHICS
F3BC 7208                           JC      P6
F3BE 80FC07                         CMP     AH,7                    ; IS THIS BW CARD
F3C1 7403                           JE      P6
F3C3 E9B201                         JMP     GRAPHICS_WRITE
F3C6                        P6:                                     ; WRITE_AC_CONTINUE
F3C6 8AE3                           MOV     AH,BL                   ; GET ATTRIBUTE TO AH
F3C8 50                             PUSH    AX                      ; SAVE ON STACK
F3C9 51                             PUSH    CX                      ; SAVE WRITE COUNT
F3CA E8D1FF                         CALL    FIND_POSITION
F3CD 8BFB                           MOV     DI,BX                   ; ADDRESS TO DI REGISTER
F3CF 59                             POP     CX                      ; WRITE COUNT
F3D0 5B                             POP     BX                      ; CHARACTER IN BX REG
F3D1                        P7:                                     ; WRITE_LOOP

                            ;----- WAIT FOR HORIZONTAL RETRACE

F3D1 8B166300                       MOV     DX,ADDR_6845            ; GET BASE ADDRESS
F3D5 83C206                         ADD     DX,6                    ; POINT AT STATUS PORT
F3D8                        P8:
F3D8 EC                             IN      AL,DX                   ; GET STATUS
F3D9 A801                           TEST    AL,1                    ; IS IT LOW
F3DB 75FB                           JNZ     P8                      ; WAIT UNTIL IT IS
F3DD FA                             CLI                             ; NO MORE INTERRUPTS
F3DE                        P9:
F3DE EC                             IN      AL,DX                   ; GET STATUS
F3DF A801                           TEST    AL,1                    ; IS IT HIGH
F3E1 74FB                           JZ      P9                      ; WAIT UNTIL IT IS
F3E3 8BC3                           MOV     AX,BX                   ; RECOVER THE CHAR/ATTR
F3E5 AB                             STOSW                           ; PUT THE CHAR/ATTR
F3E6 FB                             STI                             ; INTERRUPTS BACK ON
F3E7 E2E8                           LOOP    P7                      ;  AS MANY TIMES AS REQUESTED
F3E9 E9D9FD                         JMP     VIDEO_RETURN
                            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                                     :
                   ;------------------------------------------------
F3EC               WRITE_C_CURRENT PROC    NEAR
F3EC 80FC04                CMP     AH,4                    ; IS THIS GRAPHICS
F3EF 7208                  JC      P10
F3F1 80FC07                CMP     AH,7                    ; IS THIS BW CARD
F3F4 7403                  JE      P10
F3F6 E97F01                JMP     GRAPHICS_WRITE
F3F9               P10:
F3F9 50                    PUSH    AX                      ; SAVE ON STACK
F3FA 51                    PUSH    CX                      ; SAVE WRITE COUNT
F3FB E8A0FF                CALL    FIND_POSITION
F3FE 8BFB                  MOV     DI,BX                   ; ADDRESS TO DI
F400 59                    POP     CX                      ; WRITE COUNT
F401 5B                    POP     BX                      ; BL HAS CHAR TO WRITE
F402               P11:                                    ; WRITE_LOOP

                   ;----- WAIT FOR HORIZONTAL RETRACE

F402 8B166300              MOV     DX,ADDR_6845            ; GET BASE ADDRESS
F406 83C206                ADD     DX,6                    ; POINT AT STATUS PORT
F409               P12:
F409 EC                    IN      AL,DX                   ; GET STATUS
F40A A801                  TEST    AL,1                    ; IS IT LOW
F40C 75FB                  JNZ     P12                     ; WAIT UNTIL IT IS
F40E FA                    CLI                             ; NO MORE INTERRUPTS
F40F               P13:
F40F EC                    IN      AL,DX                   ; GET STATUS
F410 A801                  TEST    AL,1                    ; IS IT HIGH
F412 74FB                  JZ      P13                     ; WAIT UNTIL IT IS
F414 8AC3                  MOV     AL,BL                   ; RECOVER CHAR
F416 AA                    STOSB                           ; PUT THE CHAR/ATTR
F417 FB                    STI                             ; INTERRUPTS BACK ON
F418 47                    INC     DI                      ; BUMP POINTER PAST ATTRIBUTE
F419 E2E7                  LOOP    P11                     ;  AS MANY TIMES AS REQUESTED
F41B E9A7FD                JMP     VIDEO_RETURN
                   WRITE_C_CURRENT ENDP
                   ;----------------------------------------------------------------
                   ; 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
F41E               READ_DOT        PROC    NEAR
F41E E83100                CALL    R3                      ; DETERMINE BYTE POSITION OF DOT
F421 268A04                MOV     AL,ES:[SI]              ; GET THE BYTE
F424 22C4                  AND     AL,AH                   ; MASK OFF THE OTHER BITS IN THE BYTE
F426 D2E0                  SHL     AL,CL                   ; LEFT JUSTIFY THE VALUE
F428 8ACE                  MOV     CL,DH                   ; GET NUMBER OF BITS IN RESULT
F42A D2C0                  ROL     AL,CL                   ; RIGHT JUSTIFY THE RESULT
F42C E996FD                JMP     VIDEO_RETURN            ; RETURN FROM VIDEO IO
                   READ_DOT        ENDP

F42F               WRITE_DOT       PROC    NEAR
F42F 50                    PUSH    AX                      ; SAVE DOT VALUE
F430 50                    PUSH    AX                      ;  TWICE
F431 E81E00                CALL    R3                      ; DETERMINE BYTE POSITION OF THE DOT
F434 D2E8                  SHR     AL,CL                   ; SHIFT TO SET UP THE BITS FOR OUTPUT
F436 22C4                  AND     AL,AH                   ; STRIP OFF THE OTHER BITS
F438 268A0C                MOV     CL,ES:[SI]              ; GET THE CURRENT BYTE
F43B 5B                    POP     BX                      ; RECOVER XOR FLAG
F43C F6C380                TEST    BL,80H                  ; IS IT ON
F43F 750D                  JNZ     R2                      ; YES, XOR THE DOT
F441 F6D4                  NOT     AH                      ; SET THE MASK TO REMOVE THE
F443 22CC                  AND     CL,AH                   ;  INDICATED BITS
F445 0AC1                  OR      AL,CL                   ; OR IN THE NEW VALUE OF THOSE BITS
F447               R1:                                     ; FINISH_DOT
F447 268804                MOV     ES:[SI],AL              ; RESTORE THE BYTE IN MEMORY
F44A 58                    POP     AX
F44B E977FD                JMP     VIDEO_RETURN            ; RETURN FROM VIDEO IO
F44E               R2:                                     ; XOR_DOT
F44E 32C1                  XOR     AL,CL                   ; EXCLUSIVE OR THE DOTS
F450 EBF5                  JMP     R1                      ; FINISH UP THE WRITING
                   WRITE_DOT       ENDP
```

SECTION 5

```
                  ;--------------------------------------------------------
                  ; 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                                :
                  ;--------------------------------------------------------
F452              R3      PROC    NEAR
F452 53                   PUSH    BX                  ; SAVE BX DURING OPERATION
F453 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

F454 B028                 MOV     AL,40
F456 52                   PUSH    DX                  ; SAVE ROW VALUE
F457 80E2FE               AND     DL,0FEH             ; STRIP OFF ODD/EVEN BIT
F45A F6E2                 MUL     DL                  ; AX HAS ADDRESS OF 1ST BYTE
                                                      ;  OF INDICATED ROW
F45C 5A                   POP     DX                  ; RECOVER IT
F45D F6C201               TEST    DL,1                ; TEST FOR EVEN/ODD
F460 7403                 JZ      R4                  ; JUMP IF EVEN ROW
F462 050020               ADD     AX,2000H            ; OFFSET TO LOCATION OF ODD ROWS
F465              R4:                                 ; EVEN_ROW
F465 8BF0                 MOV     SI,AX               ; MOVE POINTER TO SI
F467 58                   POP     AX                  ; RECOVER AL VALUE
F468 8BD1                 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)    :
                  ;----------------------------------------------------------------
F46A BBC002               MOV     BX,2C0H
F46D B90203               MOV     CX,302H             ; SET PARMS FOR MED RES
F470 803E490006           CMP     CRT_MODE,6
F475 7206                 JC      R5                  ; HANDLE IF MED ARES
F477 BB8001               MOV     BX,180H
F47A B90307               MOV     CX,703H             ; SET PARMS FOR HIGH RES

                  ;----- DETERMINE BIT OFFSET IN BYTE FROM COLUMN MASK

F47D              R5:
F47D 22EA                 AND     CH,DL               ; ADDRESS OF PEL WITHIN BYTE TO CH

                  ;----- DETERMINE BYTE OFFSET FOR THIS LOCATION IN COLUMN

F47F D3EA                 SHR     DX,CL               ; SHIFT BY CORRECT AMOUNT
F481 03F2                 ADD     SI,DX               ; INCREMENT THE POINTER
F483 8AF7                 MOV     DH,BH               ; GET THE # OF BITS IN RESULT TO DH

                  ;----- MULTIPLY BH (VALID BITS IN BYTE) BY CH (BIT OFFSET)

F485 2AC9                 SUB     CL,CL               ; ZERO INTO STORAGE LOCATION
F487              R6:
F487 D0C8                 ROR     AL,1                ; LEFT JUSTIFY THE VALUE
                                                      ;  IN AL (FOR WRITE)
F489 02CD                 ADD     CL,CH               ; ADD IN THE BIT OFFSET VALUE
F48B FECF                 DEC     BH                  ; LOOP CONTROL
F48D 75F8                 JNZ     R6                  ; ON EXIT, CL HAS SHIFT COUNT
                                                      ;  TO RESTORE BITS
F48F 8AE3                 MOV     AH,BL               ; GET MASK TO AH
F491 D2EC                 SHR     AH,CL               ; MOVE THE MASK TO CORRECT LOCATION
F493 5B                   POP     BX                  ; RECOVER REG
F494 C3                   RET                         ; RETURN WITH EVERYTHING SET UP
                  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                          :
                  ;----------------------------------------------------------------
F495              GRAPHICS_UP     PROC    NEAR
F495 8AD8                 MOV     BL,AL               ; SAVE LINE COUNT IN BL
F497 8BC1                 MOV     AX,CX               ; GET UPPER LEFT POSITION INTO AX REG

                  ;----- USE CHARACTER SUBROUTINE FOR POSITIONING
                  ;----- ADDRESS RETURNED IS MULTIPLIED BY 2 FROM CORRECT VALUE

F499 E86902               CALL    GRAPH_POSN
F49C 8BF8                 MOV     DI,AX               ; SAVE RESULT AS DESTINATION ADDRESS

                  ;----- DETERMINE SIZE OF WINDOW

F49E 2BD1                 SUB     DX,CX
F4A0 81C20101             ADD     DX,101H             ; ADJUST VALUES
F4A4 D0E6                 SAL     DH,1                ; MULTIPLY # ROWS BY 4
                                                      ;  SINCE 8 VERT DOTS/CHAR
F4A6 D0E6                 SAL     DH,1                ;  AND EVEN/ODD ROWS

                  ;----- DETERMINE CRT MODE

F4A8 803E490006           CMP     CRT_MODE,6          ; TEST FOR MEDIUM RES
F4AD 7304                 JNC     R7                  ; FIND_SOURCE
```

```

                     ;----- MEDIUM RES UP

F4AF D0E2                    SAL     DL,1                    ; # COLUMNS * 2, SINCE 2 BYTES/CHAR
F4B1 D1E7                    SAL     DI,1                    ; OFFSET *2 SINCE 2 BYTES/CHAR

                     ;----- DETERMINE THE SOURCE ADDRESS IN THE BUFFER

F4B3                 R7:                                     ; FIND_SOURCE
F4B3 06                      PUSH    ES                      ; GET SEGMENTS BOTH POINTING TO REGEN
F4B4 1F                      POP     DS
F4B5 2AED                    SUB     CH,CH                   ; ZERO TO HIGH OF COUNT REG
F4B7 D0E3                    SAL     BL,1                    ; MULTIPLY NUMBER OF LINES BY 4
F4B9 D0E3                    SAL     BL,1
F4BB 742D                    JZ      R11                     ; IF ZERO, THEN BLANK ENTIRE FIELD
F4BD 8AC3                    MOV     AL,BL                   ; GET NUMBER OF LINES IN AL
F4BF B450                    MOV     AH,80                   ; 80 BYTES/ROW
F4C1 F6E4                    MUL     AH                      ; DETERMINE OFFSET TO SOURCE
F4C3 8BF7                    MOV     SI,DI                   ; SET UP SOURCE
F4C5 03F0                    ADD     SI,AX                   ;  ADD IN OFFSET TO IT
F4C7 8AE6                    MOV     AH,DH                   ; NUMBER OF ROWS IN FIELD
F4C9 2AE3                    SUB     AH,BL                   ; DETERMINE NUMBER TO MOVE

                     ;----- LOOP THROUGH, MOVING ONE ROW AT A TIME, BOTH EVEN AND ODD FIELDS

F4CB                 R8:                                     ; ROW_LOOP
F4CB E88000                  CALL    R17                     ; MOVE ONE ROW
F4CE 81EEB01F                SUB     SI,2000H-80             ; MOVE TO NEXT ROW
F4D2 81EFB01F                SUB     DI,2000H-80
F4D6 FECC                    DEC     AH                      ; NUMBER OF ROWS TO MOVE
F4D8 75F1                    JNZ     R8                      ; CONTINUE TILL ALL MOVED

                     ;----- FILL IN THE VACATED LINE(S)

F4DA                 R9:                                     ; CLEAR_ENTRY
F4DA 8AC7                    MOV     AL,BH                   ; ATTRIBUTE TO FILL WITH
F4DC                 R10:
F4DC E88800                  CALL    R18                     ; CLEAR THAT ROW
F4DF 81EFB01F                SUB     DI,2000H-80             ; POINT TO NEXT LINE
F4E3 FECB                    DEC     BL                      ; NUMBER OF LINES TO FILL
F4E5 75F5                    JNZ     R10                     ; CLEAR_LOOP
F4E7 E9DBFC                  JMP     VIDEO_RETURN            ; EVERYTHING DONE
F4EA                 R11:                                    ; BLANK_FIELD
F4EA 8ADE                    MOV     BL,DH                   ; SET BLANK COUNT TO
                                                             ;  EVERYTHING IN FIELD
F4EC EBEC                    JMP     R9                      ; CLEAR THE FIELD
                     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                        :
                     ;--------------------------------------------------------------
F4EE                 GRAPHICS_DOWN   PROC    NEAR
F4EE FD                      STD                             ; SET DIRECTION
F4EF 8AD8                    MOV     BL,AL                   ; SAVE LINE COUNT IN BL
F4F1 8BC2                    MOV     AX,DX                   ; GET LOWER RIGHT POSITION INTO AX REG

                     ;----- USE CHARACTER SUBROUTINE FOR POSITIONING
                     ;----- ADDRESS RETURNED IS MULTIPLIED BY 2 FROM CORRECT VALUE

F4F3 E80F02                  CALL    GRAPH_POSN
F4F6 8BF8                    MOV     DI,AX                   ; SAVE RESULT AS DESTINATION ADDRESS

                     ;----- DETERMINE SIZE OF WINDOW

F4F8 2BD1                    SUB     DX,CX
F4FA 81C20101                ADD     DX,101H                 ; ADJUST VALUES
F4FE D0E6                    SAL     DH,1                    ; MULTIPLY # ROWS BY 4
                                                             ;  SINCE 8 VERT DOTS/CHAR
F500 D0E6                    SAL     DH,1                    ;  AND EVEN/ODD ROWS

                     ;----- DETERMINE CRT MODE

F502 803E490006              CMP     CRT_MODE,6              ; TEST FOR MEDIUM RES
F507 7305                    JNC     R12                     ; FIND_SOURCE_DOWN

                     ;----- MEDIUM RES DOWN

F509 D0E2                    SAL     DL,1                    ; # COLUMNS * 2, SINCE
                                                             ;  2 BYTES/CHAR (OFFSET OK)
F50B D1E7                    SAL     DI,1                    ; OFFSET *2 SINCE 2 BYTES/CHAR
F50D 47                      INC     DI                      ; POINT TO LAST BYTE

                     ;----- DETERMINE THE SOURCE ADDRESS IN THE BUFFER

F50E                 R12:                                    ; FIND_SOURCE_DOWN
F50E 06                      PUSH    ES                      ; BOTH SEGMENTS TO REGEN
F50F 1F                      POP     DS
F510 2AED                    SUB     CH,CH                   ; ZERO TO HIGH OF COUNT REG
F512 81C7F000                ADD     DI,240                  ; POINT TO LAST ROW OF PIXELS
F516 D0E3                    SAL     BL,1                    ; MULTIPLY NUMBER OF LINES BY 4
F518 D0E3                    SAL     BL,1
F51A 742E                    JZ      R16                     ; IF ZERO, THEN BLANK ENTIRE FIELD
F51C 8AC3                    MOV     AL,BL                   ; GET NUMBER OF LINES IN AL
F51E B450                    MOV     AH,80                   ; 80 BYTES/ROW
F520 F6E4                    MUL     AH                      ; DETERMINE OFFSET TO SOURCE
F522 8BF7                    MOV     SI,DI                   ; SET UP SOURCE
F524 2BF0                    SUB     SI,AX                   ;  SUBTRACT THE OFFSET
F526 8AE6                    MOV     AH,DH                   ; NUMBER OF ROWS IN FIELD
F528 2AE3                    SUB     AH,BL                   ; DETERMINE NUMBER TO MOVE
```

SECTION 5

```
                     ;----- LOOP THROUGH, MOVING ONE ROW AT A TIME, BOTH EVEN AND ODD FIELDS

F52A                 R13:                                      ; ROW_LOOP_DOWN
F52A E82100                  CALL    R17                       ; MOVE ONE ROW
F52D 81EE5020                SUB     SI,2000H+80               ; MOVE TO NEXT ROW
F531 81EF5020                SUB     DI,2000H+80
F535 FECC                    DEC     AH                        ; NUMBER OF ROWS TO MOVE
F537 75F1                    JNZ     R13                       ; CONTINUE TILL ALL MOVED

                     ;----- FILL IN THE VACATED LINE(S)

F539                 R14:                                      ; CLEAR_ENTRY_DOWN
F539 8AC7                    MOV     AL,BH                     ; ATTRIBUTE TO FILL WITH
F53B                 R15:                                      ; CLEAR_LOOP_DOWN
F53B E82900                  CALL    R18                       ; CLEAR A ROW
F53E 81EF5020                SUB     DI,2000H+80               ; POINT TO NEXT LINE
F542 FECB                    DEC     BL                        ; NUMBER OF LINES TO FILL
F544 75F5                    JNZ     R15                       ; CLEAR_LOOP_DOWN
F546 FC                      CLD                               ; RESET THE DIRECTION FLAG
F547 E97BFC                  JMP     VIDEO_RETURN              ; EVERYTHING DONE
F54A                 R16:                                      ; BLANK_FIELD_DOWN
F54A 8ADE                    MOV     BL,DH                     ; SET BLANK COUNT TO EVERYTHING
                                                               ;  IN FIELD
F54C EBEB                    JMP     R14                       ; CLEAR THE FIELD
                     GRAPHICS_DOWN   ENDP

                     ;----- ROUTINE TO MOVE ONE ROW OF INFORMATION

F54E                 R17     PROC    NEAR
F54E 8ACA                    MOV     CL,DL                     ; NUMBER OF BYTES IN THE ROW
F550 56                      PUSH    SI
F551 57                      PUSH    DI                        ; SAVE POINTERS
F552 F3                      REP     MOVSB                     ; MOVE THE EVEN FIELD
F553 A4
F554 5F                      POP     DI
F555 5E                      POP     SI
F556 81C60020                ADD     SI,2000H
F55A 81C70020                ADD     DI,2000H                  ; POINT TO THE ODD FIELD
F55E 56                      PUSH    SI
F55F 57                      PUSH    DI                        ; SAVE THE POINTERS
F560 8ACA                    MOV     CL,DL                     ; COUNT BACK
F562 F3                      REP     MOVSB                     ; MOVE THE ODD FIELD
F563 A4
F564 5F                      POP     DI
F565 5E                      POP     SI                        ; POINTERS BACK
F566 C3                      RET                               ; RETURN TO CALLER
                     R17     ENDP

                     ;----- CLEAR A SINGLE ROW

F567                 R18     PROC    NEAR
F567 8ACA                    MOV     CL,DL                     ; NUMBER OF BYTES IN FIELD
F569 57                      PUSH    DI                        ; SAVE POINTER
F56A F3                      REP     STOSB                     ; STORE THE NEW VALUE
F56B AA
F56C 5F                      POP     DI                        ; POINTER BACK
F56D 81C70020                ADD     DI,2000H                  ; POINT TO ODD FIELD
F571 57                      PUSH    DI
F572 8ACA                    MOV     CL,DL
F574 F3                      REP     STOSB                     ; FILL THE ODD FILELD
F575 AA
F576 5F                      POP     DI
F577 C3                      RET                               ; RETURN TO CALLER
                     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
F578                 GRAPHICS_WRITE  PROC    NEAR
F578 B400                    MOV     AH,0                      ; ZERO TO HIGH OF CODE POINT
F57A 50                      PUSH    AX                        ; SAVE CODE POINT VALUE

                     ;----- DETERMINE POSITION IN REGEN BUFFER TO PUT CODE POINTS

F57B E88401                  CALL    S26                       ; FIND LOCATION IN REGEN BUFFER
F57E 8BF8                    MOV     DI,AX                     ; REGEN POINTER IN DI

                     ;----- DETERMINE REGION TO GET CODE POINTS FROM

F580 58                      POP     AX                        ; RECOVER CODE POINT
F581 3C80                    CMP     AL,80H                    ; IS IT IN SECOND HALF
F583 7306                    JAE     S1                        ; YES
```

```
LOC  OBJECT          LINE   SOURCE (BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82

                            ;----- IMAGE IS IN FIRST HALF, CONTAINED IN ROM

F585 BE6EFA                         MOV     SI,0FA6EH               ; CRT_CHAR_GEN (OFFSET OF IMAGES)
F588 0E                             PUSH    CS                      ; SAVE SEGMENT ON STACK
F589 EB0F                           JMP     SHORT S2                ; DETERMINE_MODE

                            ;----- IMAGE IS IN SECOND HALF, IN USER RAM

F58B                        S1:                                     ; EXTEND_CHAR
F58B 2C80                           SUB     AL,80H                  ; ZERO ORIGIN FOR SECOND HALF
F58D 1E                             PUSH    DS                      ; SAVE DATA POINTER
F58E 2BF6                           SUB     SI,SI
F590 8EDE                           MOV     DS,SI                   ; ESTABLISH VECTOR ADDRESSING
                                    ASSUME  DS:ABS0
F592 C5367C00                       LDS     SI,EXT_PTR              ; GET THE OFFSET OF THE TABLE
F596 8CDA                           MOV     DX,DS                   ; GET THE SEGMENT OF THE TABLE
                                    ASSUME  DS:DATA
F598 1F                             POP     DS                      ; RECOVER DATA SEGMENT
F599 52                             PUSH    DX                      ; SAVE TABLE SEGMENT ON STACK

                            ;----- DETERMINE GRAPHICS MODE IN OPERATION

F59A                        S2:                                     ; DETERMINE_MODE
F59A D1E0                           SAL     AX,1                    ; MULTIPLY CODE POINT
F59C D1E0                           SAL     AX,1                    ;  VALUE BY 8
F59E D1E0                           SAL     AX,1
F5A0 03F0                           ADD     SI,AX                   ; SI HAS OFFSET OF DESIRED CODES
F5A2 803E490006                     CMP     CRT_MODE,6
F5A7 1F                             POP     DS                      ; RECOVER TABLE POINTER SEGMENT
F5A8 722C                           JC      S7                      ; TEST FOR MEDIUM RESOLUTION MODE

                            ;----- HIGH RESOLUTION MODE

F5AA                        S3:                                     ; HIGH_CHAR
F5AA 57                             PUSH    DI                      ; SAVE REGEN POINTER
F5AB 56                             PUSH    SI                      ; SAVE CODE POINTER
F5AC B604                           MOV     DH,4                    ; NUMBER OF TIMES THROUGH LOOP
F5AE                        S4:
F5AE AC                             LODSB                           ; GET BYTE FROM CODE POINTS
F5AF F6C380                         TEST    BL,80H                  ; SHOULD WE USE THE FUNCTION
F5B2 7516                           JNZ     S6                      ;  TO PUT CHAR IN
F5B4 AA                             STOSB                           ; STORE IN REGEN BUFFER
F5B5 AC                             LODSB
F5B6                        S5:
F5B6 268885FF1F                     MOV     ES:[DI+2000H-1],AL      ; STORE IN SECOND HALF
F5BB 83C74F                         ADD     DI,79                   ; MOVE TO NEXT ROW IN REGEN
F5BE FECE                           DEC     DH                      ; DONE WITH LOOP
F5C0 75EC                           JNZ     S4
F5C2 5E                             POP     SI
F5C3 5F                             POP     DI                      ; RECOVER REGEN POINTER
F5C4 47                             INC     DI                      ; POINT TO NEXT CHAR POSITION
F5C5 E2E3                           LOOP    S3                      ; MORE CHARS TO WRITE
F5C7 E9FBFB                         JMP     VIDEO_RETURN
F5CA                        S6:
F5CA 263205                         XOR     AL,ES:[DI]              ; EXCLUSIVE OR WITH CURRENT
F5CD AA                             STOSB                           ; STORE THE CODE POINT
F5CE AC                             LODSB                           ; AGAIN FOR ODD FIELD
F5CF 263285FF1F                     XOR     AL,ES:[DI+2000H-1]
F5D4 EBE0                           JMP     S5                      ; BACK TO MAINSTREAM

                            ;----- MEDIUM RESOLUTION WRITE

F5D6                        S7:                                     ; MED_RES_WRITE
F5D6 8AD3                           MOV     DL,BL                   ; SAVE HIGH COLOR BIT
F5D8 D1E7                           SAL     DI,1                    ; OFFSET*2 SINCE 2 BYTES/CHAR
F5DA E8D100                         CALL    S19                     ; EXPAND BL TO FULL WORD OF COLOR
F5DD                        S8:                                     ; MED_CHAR
F5DD 57                             PUSH    DI                      ; SAVE REGEN POINTER
F5DE 56                             PUSH    SI                      ; SAVE THE CODE POINTER
F5DF B604                           MOV     DH,4                    ; NUMBER OF LOOPS
F5E1                        S9:
F5E1 AC                             LODSB                           ; GET CODE POINT
F5E2 E8DE00                         CALL    S21                     ; DOUBLE UP ALL THE BITS
F5E5 23C3                           AND     AX,BX                   ; CONVERT THEM TO FOREGROUND
                                                                    ;  COLOR ( 0 BACK )
F5E7 F6C280                         TEST    DL,80H                  ; IS THIS XOR FUNCTION
F5EA 7407                           JZ      S10                     ; NO, STORE IT IN AS IT IS
F5EC 263225                         XOR     AH,ES:[DI]              ; DO FUNCTION WITH HALF
F5EF 26324501                       XOR     AL,ES:[DI+1]            ; AND WITH OTHER HALF
F5F3                        S10:
F5F3 268825                         MOV     ES:[DI],AH              ; STORE FIRST BYTE
F5F6 26884501                       MOV     ES:[DI+1],AL            ; STORE SECOND BYTE
F5FA AC                             LODSB                           ; GET CODE POINT
F5FB E8C500                         CALL    S21
F5FE 23C3                           AND     AX,BX                   ; CONVERT TO COLOR
F600 F6C280                         TEST    DL,80H                  ; AGAIN, IS THIS XOR FUNCTION
F603 740A                           JZ      S11                     ; NO, JUST STORE THE VALUES
F605 2632A50020                     XOR     AH,ES:[DI+2000H]        ; FUNCTION WITH FIRST HALF
F60A 2632850120                     XOR     AL,ES:[DI+2001H]        ; AND WITH SECOND HALF
F60F                        S11:
F60F 2688A50020                     MOV     ES:[DI+2000H ,AH]
F614 2688850120                     MOV     ES:[DI+2000H+1],AL      ; STORE IN SECOND PORTION OF BUFFER
F619 83C750                         ADD     DI,80                   ; POINT TO NEXT LOCATION
F61C FECE                           DEC     DH
F61E 75C1                           JNZ     S9                      ; KEEP GOING
F620 5E                             POP     SI                      ; RECOVER CODE PONTER
F621 5F                             POP     DI                      ; RECOVER REGEN POINTER
F622 47                             INC     DI                      ; POINT TO NEXT CHAR POSITION
F623 47                             INC     DI
F624 E2B7                           LOOP    S8                      ; MORE TO WRITE
F626 E99CFB                         JMP     VIDEO_RETURN
                            GRAPHICS_WRITE  ENDP
```

SECTION 5

```
                  ;------------------------
                  ; GRAPHICS READ          :
                  ;------------------------
F629              GRAPHICS_READ   PROC    NEAR
F629 E8D600               CALL    S26                     ; CONVERTED TO OFFSET IN REGEN
F62C 8BF0                 MOV     SI,AX                   ; SAVE IN SI
F62E 83EC08               SUB     SP,8                    ; ALLOCATE SPACE TO SAVE THE
                                                          ;  READ CODE POINT
F631 8BEC                 MOV     BP,SP                   ; POINTER TO SAVE AREA

                  ;----- DETERMINE GRAPHICS MODES

F633 803E490006           CMP     CRT_MODE,6
F638 06                   PUSH    ES
F639 1F                   POP     DS                      ; POINT TO REGEN SEGMENT
F63A 721A                 JC      S13                     ; MEDIUM RESOLUTION

                  ;----- HIGH RESOLUTION READ

                  ;----- GET VALUES FROM REGEN BUFFER AND CONVERT TO CODE POINT

F63C B604                 MOV     DH,4                    ; NUMBER OF PASSES
F63E              S12:
F63E                      MOV     AL,[SI]                 ; GET FIRST BYTE
F640 884600               MOV     [BP],AL                 ; SAVE IN STORAGE AREA
F643 45                   INC     BP                      ; NEXT LOCATION
F644 8A840020             MOV     AL,[SI+2000H]           ; GET LOWER REGION BYTE
F648 884600               MOV     [BP],AL                 ; ADJUST AND STORE
F64B 45                   INC     BP
F64C 83C650               ADD     SI,80                   ; POINTER INTO REGEN
F64F FECE                 DEC     DH                      ; LOOP CONTROL
F651 75EB                 JNZ     S12                     ; DO IT SOME MORE
F653 EB1790               JMP     S15                     ; GO MATCH THE SAVED CODE POINTS

                  ;----- MEDIUM RESOLUTION READ

F656              S13:                                    ; MED_RES_READ
F656 D1E6                 SAL     SI,1                    ; OFFSET*2 SINCE 2 BYTES/CHAR
F658 B604                 MOV     DH,4                    ; NUMBER OF PASSES
F65A              S14:
F65A E88800               CALL    S23                     ; GET PAIR BYTES FROM REGEN
                                                          ;  INTO SINGLE SAVE

F65D 81C60020             ADD     SI,2000H                ; GO TO LOWER REGION
F661 E88100               CALL    S23                     ; GET THIS PAIR INTO SAVE
F664 81EEB01F             SUB     SI,2000H-80             ; ADJUST POINTER BACK INTO UPPER
F668 FECE                 DEC     DH
F66A 75EE                 JNZ     S14                     ; KEEP GOING UNTIL ALL 8 DONE

                  ;----- SAVE AREA HAS CHARACTER IN IT, MATCH IT

F66C              S15:                                    ; FIND_CHAR
F66C BF6EFA90             MOV     DI,OFFSET CRT_CHAR_GEN  ; ESTABLISH ADDRESSING
F670 0E                   PUSH    CS
F671 07                   POP     ES                      ; CODE POINTS IN CS
F672 83ED08               SUB     BP,8                    ; ADJUST POINTER TO BEGINNING
                                                          ;  OF SAVE AREA
F675 8BF5                 MOV     SI,BP
F677 FC                   CLD                             ; ENSURE DIRECTION
F678 B000                 MOV     AL,0                    ; CURRENT CODE POINT BEING MATCHED
F67A              S16:
F67A 16                   PUSH    SS                      ; ESTABLISH ADDRESSING TO STACK
F67B 1F                   POP     DS                      ; FOR THE STRING COMPARE
F67C BA8000               MOV     DX,128                  ; NUMBER TO TEST AGAINST
F67F              S17:
F67F 56                   PUSH    SI                      ; SAVE SAVE AREA POINTER
F680 57                   PUSH    DI                      ; SAVE CODE POINTER
F681 B90800               MOV     CX,8                    ; NUMBER OF BYTES TO MATCH
F684 F3                   REPE    CMPSB                   ; COMPARE THE 8 BYTES
F685 A6
F686 5F                   POP     DI                      ; RECOVER THE POINTERS
F687 5E                   POP     SI
F688 741E                 JZ      S18                     ; IF ZERO FLAG SET, THEN MATCH OCCURRED
F68A FEC0                 INC     AL                      ; NO MATCH, MOVE ON TO NEXT
F68C 83C708               ADD     DI,8                    ; NEXT CODE POINT
F68F 4A                   DEC     DX                      ; LOOP CONTROL
F690 75ED                 JNZ     S17                     ; DO ALL OF THEM

                  ;----- CHAR NOT MATCHED, MIGHT BE IN USER SUPPLIED SECOND HALF

F692 3C00                 CMP     AL,0                    ; AL <> 0 IF ONLY 1ST HALF SCANNED
F694 7412                 JE      S18                     ; IF = 0, THEN ALL HAS BEEN SCANNED
F696 2BC0                 SUB     AX,AX
F698 8ED8                 MOV     DS,AX                   ; ESTABLISH ADDRESSING TO VECTOR
                          ASSUME  DS:ABS0
F69A C43E7C00             LES     DI,EXT_PTR              ; GET POINTER
F69E 8CC0                 MOV     AX,ES                   ; SEE IF THE POINTER REALLY EXISTS
F6A0 0BC7                 OR      AX,DI                   ; IF ALL 0, THEN DOESN'T EXIST
F6A2 7404                 JZ      S18                     ; NO SENSE LOOKING
F6A4 B080                 MOV     AL,128                  ; ORIGIN FOR SECOND HALF
F6A6 EBD2                 JMP     S16                     ; GO BACK AND TRY FOR IT
                          ASSUME  DS:DATA

                  ;----- CHARACTER IS FOUND ( AL=0 IF NOT FOUND )

F6A8              S18:
F6A8 83C408               ADD     SP,8                    ; READJUST THE STACK, THROW AWAY SAVE
F6AB E917FB               JMP     VIDEO_RETURN            ; ALL DONE
                  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 )                                 :
                ;------------------------------------------------------
F6AE            S19     PROC    NEAR
F6AE 80E303             AND     BL,3                ; ISOLATE THE COLOR BITS
F6B1 8AC3               MOV     AL,BL               ; COPY TO AL
F6B3 51                 PUSH    CX                  ; SAVE REGISTER
F6B4 B90300             MOV     CX,3                ; NUMBER OF TIMES TO DO THIS
F6B7            S20:
F6B7 D0E0               SAL     AL,1
F6B9 D0E0               SAL     AL,1                ; LEFT SHIFT BY 2
F6BB 0AD8               OR      BL,AL               ; ANOTHER COLOR VERSION INTO BL
F6BD E2F8               LOOP    S20                 ; FILL ALL OF BL
F6BF 8AFB               MOV     BH,BL               ; FILL UPPER PORTION
F6C1 59                 POP     CX                  ; REGISTER BACK
F6C2 C3                 RET                         ; ALL DONE
                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              :
                ;------------------------------------------------------
F6C3            S21     PROC    NEAR
F6C3 52                 PUSH    DX                  ; SAVE REGISTERS
F6C4 51                 PUSH    CX
F6C5 53                 PUSH    BX
F6C6 2BD2               SUB     DX,DX               ; RESULT REGISTER
F6C8 B90100             MOV     CX,1                ; MASK REGISTER
F6CB            S22:
F6CB 8BD8               MOV     BX,AX               ; BASE INTO TEMP
F6CD 23D9               AND     BX,CX               ; USE MASK TO EXTRACT A BIT
F6CF 0BD3               OR      DX,BX               ; PUT INTO RESULT REGISTER
F6D1 D1E0               SHL     AX,1
F6D3 D1E1               SHL     CX,1                ; SHIFT BASE AND MASK BY 1
F6D5 8BD8               MOV     BX,AX               ; BASE TO TEMP
F6D7 23D9               AND     BX,CX               ; EXTRACT THE SAME BIT
F6D9 0BD3               OR      DX,BX               ; PUT INTO RESULT
F6DB D1E1               SHL     CX,1                ; SHIFT ONLY MASK NOW,
                                                    ;  MOVING TO NEXT BASE
F6DD 73EC               JNC     S22                 ; USE MASK BIT COMING OUT TO TERMINATE
F6DF 8BC2               MOV     AX,DX               ; RESULT TO PARM REGISTER
F6E1 5B                 POP     BX
F6E2 59                 POP     CX                  ; RECOVER REGISTERS
F6E3 5A                 POP     DX
F6E4 C3                 RET                         ; ALL DONE
                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                     :
                ;------------------------------------------------------
F6E5            S23     PROC    NEAR
F6E5 8A24               MOV     AH,[SI]             ; GET FIRST BYTE
F6E7 8A4401             MOV     AL,[SI+1]           ; GET SECOND BYTE
F6EA B900C0             MOV     CX,0C000H           ; 2 BIT MASK TO TEST THE ENTRIES
F6ED B200               MOV     DL,0                ; RESULT REGISTER
F6EF            S24:
F6EF 85C1               TEST    AX,CX               ; IS THIS SECTION BACKGROUND?
F6F1 F8                 CLC                         ; CLEAR CARRY IN HOPES THAT IT IS
F6F2 7401               JZ      S25                 ; IF ZERO, IT IS BACKGROUND
F6F4 F9                 STC                         ; WASN'T, SO SET CARRY
F6F5 D0D2       S25:    RCL     DL,1                ; MOVE THAT BIT INTO THE RESULT
F6F7 D1E9               SHR     CX,1
F6F9 D1E9               SHR     CX,1                ; MOVE THE MASK TO THE RIGHT BY 2 BITS
F6FB 73F2               JNC     S24                 ; DO IT AGAIN IF MASK DIDN'T FALL OUT
F6FD 885600             MOV     [BP],DL             ; STORE RESULT IN SAVE AREA
F700 45                 INC     BP                  ; ADJUST POINTER
F701 C3                 RET                         ; ALL DONE
                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 :
                ;-------------------------------------------
F702            S26     PROC    NEAR
F702 A15000             MOV     AX,CURSOR_POSN      ; GET CURRENT CURSOR
F705            GRAPH_POSN      LABEL   NEAR
F705 53                 PUSH    BX                  ; SAVE REGISTER
F706 8BD8               MOV     BX,AX               ; SAVE A COPY OF CURRENT CURSOR
F708 8AC4               MOV     AL,AH               ; GET ROWS TO AL
F70A F6264A00           MUL     BYTE PTR CRT_COLS   ; MULTIPLY BY BYTES/COLUMN
F70E D1E0               SHL     AX,1                ; MULTIPLY * 4 SINCE 4 ROWS/BYTE
F710 D1E0               SHL     AX,1
F712 2AFF               SUB     BH,BH               ; ISOLATE COLUMN VALUE
F714 03C3               ADD     AX,BX               ; DETERMINE OFFSET
F716 5B                 POP     BX                  ; RECOVER POINTER
F717 C3                 RET                         ; ALL DONE
                S26     ENDP
```

SECTION 5

```
                ;-----------------------------------------------------------------------
                ; 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 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
F718            WRITE_TTY       PROC    NEAR
F718 50                 PUSH    AX                      ; SAVE REGISTERS
F719 50                 PUSH    AX                      ; SAVE CHAR TO WRITE
F71A B403               MOV     AH,3
F71C 8A3E6200           MOV     BH,ACTIVE_PAGE          ; GET THE CURRENT ACTIVE PAGE
F720 CD10               INT     10H                     ; READ THE CURRENT CURSOR POSITION
F722 58                 POP     AX                      ; RECOVER CHAR

                ;----- DX NOW HAS THE CURRENT CURSOR POSITION

F723 3C08               CMP     AL,8                    ; IS IT A BACKSPACE
F725 7452               JE      U8                      ; BACK_SPACE
F727 3C0D               CMP     AL,0DH                  ; IS IT CARRIAGE RETURN
F729 7457               JE      U9                      ; CAR_RET
F72B 3C0A               CMP     AL,0AH                  ; IS IT A LINE FEED
F72D 7457               JE      U10                     ; LINE_FEED
F72F 3C07               CMP     AL,07H                  ; IS IT A BELL
F731 745A               JE      U11                     ; BELL

                ;----- WRITE THE CHAR TO THE SCREEN


F733 B40A               MOV     AH,10                   ; WRITE CHAR ONLY
F735 B90100             MOV     CX,1                    ; ONLY ONE CHAR
F738 CD10               INT     10H                     ; WRITE THE CHAR

                ;----- POSITION THE CURSOR FOR NEXT CHAR

F73A FEC2               INC     DL
F73C 3A164A00           CMP     DL,BYTE PTR CRT_COLS    ; TEST FOR COLUMN OVERFLOW
F740 7533               JNZ     U7                      ; SET_CURSOR
F742 B200               MOV     DL,0                    ; COLUMN FOR CURSOR
F744 80FE18             CMP     DH,24
F747 752A               JNZ     U6                      ; SET_CURSOR_INC

                ;----- SCROLL REQUIRED

F749            U1:
F749 B402               MOV     AH,2
F74B CD10               INT     10H                     ; SET THE CURSOR

                ;----- DETERMINE VALUE TO FILL WITH DURING SCROLL

F74D A04900             MOV     AL,CRT_MODE             ; GET THE CURRENT MODE
F750 3C04               CMP     AL,4
F752 7206               JC      U2                      ; READ-CURSOR
F754 3C07               CMP     AL,7
F756 B700               MOV     BH,0                    ; FILL WITH BACKGROUND
F758 7506               JNE     U3                      ; SCROLL-UP
F75A            U2:                                     ; READ-CURSOR
F75A B408               MOV     AH,8
F75C CD10               INT     10H                     ; READ CHAR/ATTR AT CURRENT CURSOR
F75E 8AFC               MOV     BH,AH                   ; STORE IN BH
F760            U3:                                     ; SCROLL-UP
F760 B80106             MOV     AX,601H                 ; SCROLL ONE LINE
F763 2BC9               SUB     CX,CX                   ; UPPER LEFT CORNER
F765 B618               MOV     DH,24                   ; LOWER RIGHT ROW
F767 8A164A00           MOV     DL,BYTE PTR CRT_COLS    ; LOWER RIGHT COLUMN
F76B FECA               DEC     DL
F76D            U4:                                     ; VIDEO-CALL-RETURN
F76D CD10               INT     10H                     ; SCROLL UP THE SCREEN
F76F            U5:                                     ; TTY-RETURN
F76F 58                 POP     AX                      ; RESTORE THE CHARACTER
F770 E952FA             JMP     VIDEO_RETURN            ; RETURN TO CALLER
F773            U6:                                     ; SET-CURSOR-INC
F773 FEC6               INC     DH                      ; NEXT ROW
F775            U7:                                     ; SET-CURSOR
F775 B402               MOV     AH,2
F777 EBF4               JMP     U4                      ; ESTABLISH THE NEW CURSOR

                ;----- BACK SPACE FOUND

F779            U8:
F779 80FA00             CMP     DL,0                    ; ALREADY AT END OF LINE
F77C 74F7               JE      U7                      ; SET_CURSOR
F77E FECA               DEC     DL                      ; NO -- JUST MOVE IT BACK
F780 EBF3               JMP     U7                      ; SET_CURSOR

                ;----- CARRIAGE RETURN FOUND

F782            U9:
F782 B200               MOV     DL,0                    ; MOVE TO FIRST COLUMN
F784 EBEF               JMP     U7                      ; SET_CURSOR

                ;----- LINE FEED FOUND

F786            U10:
F786 80FE18             CMP     DH,24                   ; BOTTOM OF SCREEN
F789 75E8               JNE     U6                      ; YES, SCROLL THE SCREEN
F78B EBBC               JMP     U1                      ; NO, JUST SET THE CURSOR
```

```
                        ;----- BELL FOUND

F78D                    U11:
F78D B302                       MOV     BL,2                    ; SET UP COUNT FOR BEEP
F78F E87602                     CALL    BEEP                    ; SOUND THE POD BELL
F792 EBDB                       JMP     U5                      ; TTY_RETURN
                        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
F794                    V1      LABEL   BYTE
F794 03                         DB      3,3,5,5,3,3,3,4 ;
F795 03
F796 05
F797 05
F798 03
F799 03
F79A 03
F79B 04
F79C                    READ_LPEN       PROC    NEAR

                        ;----- WAIT FOR LIGHT PEN TO BE DEPRESSED

F79C B400                       MOV     AH,0                    ; SET NO LIGHT PEN RETURN CODE
F79E 8B166300                   MOV     DX,ADDR_6845            ; GET BASE ADDRESS OF 6845
F7A2 83C206                     ADD     DX,6                    ; POINT TO STATUS REGISTER
F7A5 EC                         IN      AL,DX                   ; GET STATUS REGISTER
F7A6 A804                       TEST    AL,4                    ; TEST LIGHT PEN SWITCH
F7A8 757E                       JNZ     V6                      ; NOT SET, RETURN

                        ;----- NOW TEST FOR LIGHT PEN TRIGGER

F7AA A802                       TEST    AL,2                    ; TEST LIGHT PEN TRIGGER
F7AC 7503                       JNZ     V7A                     ; RETURN WITHOUT RESETTING TRIGGER
F7AE E98100                     JMP     V7

                        ;----- TRIGGER HAS BEEN SET, READ THE VALUE IN

F7B1                    V7A:
F7B1 B410                       MOV     AH,16                   ; LIGHT PEN REGISTERS ON 6845

                        ;----- INPUT REGS POINTED TO BY AH, AND CONVERT TO ROW COLUMN IN DX

F7B3 8B166300                   MOV     DX,ADDR_6845            ; ADDRESS REGISTER FOR 6845
F7B7 8AC4                       MOV     AL,AH                   ; REGISTER TO READ
F7B9 EE                         OUT     DX,AL                   ; SET IT UP
F7BA 42                         INC     DX                      ; DATA REGISTER
F7BB EC                         IN      AL,DX                   ; GET THE VALUE
F7BC 8AE8                       MOV     CH,AL                   ; SAVE IN CX
F7BE 4A                         DEC     DX                      ; ADDRESS REGISTER
F7BF FEC4                       INC     AH
F7C1 8AC4                       MOV     AL,AH                   ; SECOND DATA REGISTER
F7C3 EE                         OUT     DX,AL
F7C4 42                         INC     DX                      ; POINT TO DATA REGISTER
F7C5 EC                         IN      AL,DX                   ; GET SECOND DATA VALUE
F7C6 8AE5                       MOV     AH,CH                   ; AX HAS INPUT VALUE

                        ;----- AX HAS THE VALUE READ IN FROM THE 6845

F7C8 8A1E4900                   MOV     BL,CRT_MODE
F7CC 2AFF                       SUB     BH,BH                   ; MODE VALUE TO BX
F7CE 2E8A9F94F7                 MOV     BL,CS:V1[BX]            ; DETERMINE AMOUNT TO SUBTRACT
F7D3 2BC3                       SUB     AX,BX                   ; TAKE IT AWAY
F7D5 8B1E4E00                   MOV     BX,CRT_START
F7D9 D1EB                       SHR     BX,1
F7DB 2BC3                       SUB     AX,BX
F7DD 7902                       JNS     V2                      ; IF POSITIVE, DETERMINE MODE
F7DF 2BC0                       SUB     AX,AX                   ; <0 PLAYS AS 0

                        ;----- DETERMINE MODE OF OPERATION

F7E1                    V2:                                     ; DETERMINE_MODE
F7E1 B103                       MOV     CL,3                    ; SET *8 SHIFT COUNT
F7E3 803E490004                 CMP     CRT_MODE,4              ; DETERMINE IF GRAPHICS OR ALPHA
F7E8 722A                       JB      V4                      ; ALPHA_PEN
F7EA 803E490007                 CMP     CRT_MODE,7
F7EF 7423                       JE      V4                      ; ALPHA_PEN

                        ;----- GRAPHICS MODE

F7F1 B228                       MOV     DL,40                   ; DIVISOR FOR GRAPHICS
F7F3 F6F2                       DIV     DL                      ; DETERMINE ROW(AL) AND COLUMN(AH)
                                                                ;  AL RANGE 0-99, AH RANGE 0-39

                        ;----- DETERMINE GRAPHIC ROW POSITION

F7F5 8AE8                       MOV     CH,AL                   ; SAVE ROW VALUE IN CH
F7F7 02ED                       ADD     CH,CH                   ; *2 FOR EVEN/ODD FIELD
F7F9 8ADC                       MOV     BL,AH                   ; COLUMN VALUE TO BX
F7FB 2AFF                       SUB     BH,BH                   ; MULTIPLY BY 8 FOR MEDIUM RES
F7FD 803E490006                 CMP     CRT_MODE,6              ; DETERMINE MEDIUM OR HIGH RES
F802 7504                       JNE     V3                      ; NOT_HIGH_RES
F804 B104                       MOV     CL,4                    ; SHIFT VALUE FOR HIGH RES
F806 D0E4                       SAL     AH,1                    ; COLUMN VALUE TIMES 2 FOR HIGH RES
F808                    V3:                                     ; NOT_HIGH_RES
F808 D3E3                       SHL     BX,CL                   ; MULTIPLY *16 FOR HIGH RES
```

SECTION 5

```
                    ;----- DETERMINE ALPHA CHAR POSITION

F80A 8AD4                   MOV     DL,AH                   ; COLUMN VALUE FOR RETURN
F80C 8AF0                   MOV     DH,AL                   ; ROW VALUE
F80E D0EE                   SHR     DH,1                    ; DIVIDE BY 4
F810 D0EE                   SHR     DH,1                    ;  FOR VALUE IN 0-24 RANGE
F812 EB12                   JMP     SHORT V5                ; LIGHT_PEN_RETURN_SET

                    ;----- ALPHA MODE ON LIGHT PEN

F814                V4:                                     ; ALPHA_PEN
F814 F6364A00               DIV     BYTE PTR CRT_COLS       ; DETERMINE ROW,COLUMN VALUE
F818 8AF0                   MOV     DH,AL                   ; ROWS TO DH
F81A 8AD4                   MOV     DL,AH                   ; COLS TO DL
F81C D2E0                   SAL     AL,CL                   ; MULTIPLY ROWS * 8
F81E 8AE8                   MOV     CH,AL                   ; GET RASTER VALUE TO RETURN REG
F820 8ADC                   MOV     BL,AH                   ; COLUMN VALUE
F822 32FF                   XOR     BH,BH                   ;  TO BX
F824 D3E3                   SAL     BX,CL
F826                V5:                                     ; LIGHT_PEN_RETURN_SET
F826 B401                   MOV     AH,1                    ; INDICATE EVERTHING SET
F828                V6:                                     ; LIGHT_PEN_RETURN
F828 52                     PUSH    DX                      ; SAVE RETURN VALUE (IN CASE)
F829 8B166300               MOV     DX,ADDR_6845            ; GET BASE ADDRESS
F82D 83C207                 ADD     DX,7                    ; POINT TO RESET PARM
F830 EE                     OUT     DX,AL                   ; ADDRESS, NOT DATA, IS IMPORTANT
F831 5A                     POP     DX                      ; RECOVER VALUE
F832                V7:                                     ; RETURN_NO_RESET
F832 5F                     POP     DI
F833 5E                     POP     SI
F834 1F                     POP     DS                      ; DISCARD SAVED BX,CX,DX
F835 1F                     POP     DS
F836 1F                     POP     DS
F837 1F                     POP     DS
F838 07                     POP     ES
F839 CF                     IRET
                    READ_LPEN       ENDP
```

```
LOC OBJECT

                ;--- INT 12 -------------------------------------------------------------
                ; MEMORY_SIZE_DET                                                        :
                ;       THIS ROUTINE DETERMINES THE AMOUNT OF MEMORY IN THE SYSTEM       :
                ;       AS REPRESENTED BY THE SWITCHES ON THE PLANAR.  NOTE THAT THE     :
                ;       SYSTEM MAY NOT BE ABLE TO USE I/O MEMORY UNLESS THERE IS A FULL  :
                ;       COMPLEMENT OF 64K BYTES ON THE PLANAR.                           :
                ; INPUT                                                                  :
                ;       NO REGISTERS                                                     :
                ;       THE MEMORY_SIZE VARIABLE IS SET DURING POWER ON DIAGNOSTICS      :
                ;        ACCORDING TO THE FOLLOWING HARDWARE ASSUMPTIONS:                :
                ;       PORT 60 BITS 3,2 = 00 - 16K BASE RAM                             :
                ;                          01 - 32K BASE RAM                             :
                ;                          10 - 48K BASE RAM                             :
                ;                          11 - 64K BASE RAM                             :
                ;       PORT 62 BITS 3-0 INDICATE AMOUNT OF I/O RAM IN 32K INCREMENTS    :
                ;               E.G., 0000 - NO RAM IN I/O CHANNEL                       :
                ;                     0010 - 64K RAM IN I/O CHANNEL, ETC.                :
                ; OUTPUT                                                                 :
                ;       (AX) = NUMBER OF CONTIGUOUS 1K BLOCKS OF MEMORY                  :
                ;-------------------------------------------------------------------------
                        ASSUME  CS:CODE,DS:DATA
F841                    ORG     0F841H
F841            MEMORY_SIZE_DET PROC    FAR
F841 FB                 STI                             ; INTERRUPTS BACK ON
F842 1E                 PUSH    DS                      ; SAVE SEGMENT
F843 E81302             CALL    DDS
F846 A11300             MOV     AX,MEMORY_SIZE          ; GET VALUE
F849 1F                 POP     DS                      ; RECOVER SEGMENT
F84A CF                 IRET                            ; RETURN TO CALLER
                MEMORY_SIZE_DET  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 60  = LOW ORDER BYTE OF EQUPMENT                    :
                ;       PORT 3FA = INTERRUPT ID REGISTER OF 8250                 :
                ;               BITS 7-3 ARE ALWAYS 0                            :
                ;       PORT 378 = OUTPUT PORT OF PRINTER -- 8255 PORT THAT      :
                ;               CAN BE READ AS WELL AS WRITTEN                   :
                ; OUTPUT                                                         :
                ;       (AX) IS SET, BIT SIGNIFICANT, TO INDICATE ATTACHED I/O   :
                ;       BIT 15,14 = NUMBER OF PRINTERS ATTACHED                  :
                ;       BIT 13 NOT USED                                          :
                ;       BIT 12 = GAME I/O ATTACHED                               :
                ;       BIT 11,10,9 = NUMBER OF RS232 CARDS ATTACHED             :
                ;       BIT 8 UNUSED                                             :
                ;       BIT 7,6 = NUMBER OF DISKETTE DRIVES                      :
                ;               00=1, 01=2, 10=3, 11=4 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 3,2 = PLANAR RAM SIZE (00=16K,01=32K,10=48K,11=64K)  :
                ;       BIT 1 NOT USED                                           :
                ;       BIT 0 = IPL FROM DISKETTE -- THIS BIT INDICATES THAT     :
                ;               THERE ARE DISKETTE DRIVES ON THE SYSTEM          :
                ;                                                                :
                ;       NO OTHER REGISTERS AFFECTED                              :
                ;-----------------------------------------------------------------
                        ASSUME  CS:CODE,DS:DATA
F84D                    ORG     0F84DH
F84D            EQUIPMENT       PROC    FAR
F84D FB                 STI                             ; INTERRUPTS BACK ON
F84E 1E                 PUSH    DS                      ; SAVE SEGMENT REGISTER
F84F E80702             CALL    DDS
F852 A11000             MOV     AX,EQUIP_FLAG           ; GET THE CURRENT SETTINGS
F855 1F                 POP     DS                      ; RECOVER SEGMENT
F856 CF                 IRET                            ; RETURN TO CALLER
                EQUIPMENT       ENDP

                ;--- INT 15 --------------------------------------------------------------
                ;     DUMMY CASSETTE IO ROUTINE-RETURNS 'INVALID CMD' IF THE ROUTINE IS   :
                ;     IS EVER CALLED BY ACCIDENT (AH=86H, CARRY FLAG=1)                   :
                ;--------------------------------------------------------------------------
F859                    ORG     0F859H
F859            CASSETTE_IO     PROC    FAR
F859 F9                 STC                             ; CARRY INDICATOR=1
F85A B486               MOV     AH,86H
F85C CA0200             RET     2
                CASSETTE_IO     ENDP
```

SECTION 5

```
                         ;-----------------------------------------------------------------
                         ; NON-MASKABLE INTERRUPT ROUTINE:                                  :
                         ;          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 (INTERMITTANT :
                         ;          READ PROBLEM) ?????<-WILL BE PRINTED WHERE THE ADDRESS  :
                         ;          WOULD NORMALLY GO.                                      :
                         ;          IF ADDRESS IN ERROR IS IN THE I/O EXPANSION BOX, THE    :
                         ;          ADDRESS WILL BE FOLLOWED BY A '(E)', IF IN SYSTEM UNIT, :
                         ;          A '(S)' WILL FOLLOW THE ADDRESS                          :
                         ;-----------------------------------------------------------------
F85F                     NMI_INT PROC   NEAR
                                 ASSUME DS:DATA
F85F 50                          PUSH   AX                   ; SAVE ORIG CONTENTS OF AX
F860 E462                        IN     AL,PORT_C
F862 A8C0                        TEST   AL,0C0H              ; PARITY CHECK?
F864 7503                        JNZ    NMI_1
F866 E98700                      JMP    D14                  ; NO, EXIT FROM ROUTINE
F869                     NMI_1:
F869 BA4000                      MOV    DX,DATA
F86C 8EDA                        MOV    DS,DX
F86E BE15F990                    MOV    SI,OFFSET D1         ; ADDR OF ERROR MSG
F872 A840                        TEST   AL,40H               ; I/O PARITY CHECK
F874 7504                        JNZ    D13                  ; DISPLAY ERROR MSG
F876 BE25F990                    MOV    SI,OFFSET D2         ; MUST BE PLANAR
F87A                     D13:
F87A B400                        MOV    AH,0                 ; INIT AND SET MODE FOR VIDEO
F87C A04900                      MOV    AL,CRT_MODE
F87F CD10                        INT    10H                  ; CALL VIDEO_IO PROCEDURE
F881 E84601                      CALL   P_MSG                ; PRINT ERROR MSG

                         ;----- SEE IF LOCATION THAT CAUSED PARITY CHECK CAN BE FOUND

F884 B000                        MOV    AL,00H               ; DISABLE TRAP
F886 E6A0                        OUT    0A0H,AL
F888 E461                        IN     AL,PORT_B
F88A 0C30                        OR     AL,00110000B         ; TOGGLE PARITY CHECK ENABLES
F88C E661                        OUT    PORT_B,AL
F88E 24CF                        AND    AL,11001111B
F890 E661                        OUT    PORT_B,AL
F892 8B1E1300                    MOV    BX,MEMORY_SIZE       ; GET MEMORY SIZE WORD
F896 FC                          CLD                         ; SET DIR FLAG TO INCRIMENT
F897 2BD2                        SUB    DX,DX                ; POINT DX AT START OF MEM
F899                     NMI_LOOP:
F899 8EDA                        MOV    DS,DX
F89B 8EC2                        MOV    ES,DX
F89D B90040                      MOV    CX,4000H             ; SET FOR 16KB SCAN
F8A0 2BF6                        SUB    SI,SI                ; SET SI TO BE REALTIVE TO
                                                             ; START OF ES
F8A2 F3                          REP    LODSB                ; READ 16KB OF MEMORY
F8A3 AC
F8A4 E462                        IN     AL,PORT_C            ; SEE IF PARITY CHECK HAPPENED
F8A6 24C0                        AND    AL,11000000B
F8A8 7512                        JNZ    PRT_NMI              ; GO PRINT ADDRESS IF IT DID
F8AA 81C20004                    ADD    DX,0400H             ; POINT TO NEXT 16K BLOCK
F8AE 83EB10                      SUB    BX,16D
F8B1 75E6                        JNZ    NMI_LOOP
F8B3 BE35F990                    MOV    SI,(OFFSET D2A)      ; PRINT ROW OF ????? IF PARITY
F8B7 E81001                      CALL   P_MSG                ; CHECK COULD NOT BE RE-CREATED
F8BA FA                          CLI
F8BB F4                          HLT                         ; HALT SYSTEM
F8BC                     PRT_NMI:
F8BC 8CDA                        MOV    DX,DS
F8BE E81907                      CALL   PRT_SEG              ; PRINT SEGMENT VALUE
F8C1 BA1302                      MOV    DX,0213H
F8C4 B000                        MOV    AL,00                ; DISABLE EXPANSION BOX
F8C6 EE                          OUT    DX,AL                ; (CAN'T WRITE TO MEM)
F8C7 B028                        MOV    AL,'('
F8C9 E8D000                      CALL   PRT_HEX
F8CC B85AA5                      MOV    AX,0A55AH
F8CF 8BC8                        MOV    CX,AX
F8D1 2BDB                        SUB    BX,BX
F8D3 8907                        MOV    [BX],AX              ; WRITE A WORD TO SEGMENT THAT
F8D5 90                          NOP
F8D6 90                          NOP
F8D7 8B07                        MOV    AX,[BX]              ; HAD THE ERROR
F8D9 3BC1                        CMP    AX,CX                ; IS IT THERE?
F8DB 7407                        JE     SYS_BOX_ERR          ; YES- MUST BE SYS UNIT
F8DD B045                        MOV    AL,'E'               ; NO-MUST BE IN EXP. BOX
F8DF E8BA00                      CALL   PRT_HEX
F8E2 EB05                        JMP    SHORT HLT_NMI
F8E4                     SYS_BOX_ERR:
F8E4 B053                        MOV    AL,'S'
F8E6 E8B300                      CALL   PRT_HEX
F8E9                     HLT_NMI:
F8E9 B029                        MOV    AL,')'
F8EB E8AE00                      CALL   PRT_HEX
F8EE FA                          CLI                         ; HALT SYSTEM
F8EF F4                          HLT
F8F0                     D14:
F8F0 58                          POP    AX                   ; RESTORE ORIG CONTENTS OF AX
F8F1 CF                          IRET
                         NMI_INT ENDP

                         ;--------------------------------------
                         ;        ROS CHECKSUM SUBROUTINE       :
                         ;--------------------------------------
F8F2                     ROS_CHECKSUM    PROC    NEAR        ; NEXT_ROS_MODULE
F8F2 B90020                      MOV    CX,8192              ; NUMBER OF BYTES TO ADD
F8F5                     ROS_CHECKSUM_CNT:                   ; ENTRY FOR OPTIONAL ROS TEST
F8F5 32C0                        XOR    AL,AL
F8F7                     C26:
F8F7 0207                        ADD    AL,DS:[BX]
F8F9 43                          INC    BX                   ; POINT TO NEXT BYTE
F8FA E2FB                        LOOP   C26                  ; ADD ALL BYTES IN ROS MODULE
F8FC 0AC0                        OR     AL,AL                ; SUM = 0?
F8FE C3                          RET
                         ROS_CHECKSUM    ENDP
```

```
                        ;----------------------------------------
                        ;       MESSAGE AREA FOR POST           :
                        ;----------------------------------------
F8FF 313031             E0      DB      '101',13,10             ; SYSTEM BOARD ERROR
F902 0D
F903 0A
F904 20323031           E1      DB      ' 201',13,10            ; MEMORY ERROR
F908 0D
F909 0A
F90A 524F4D             F3A     DB      'ROM',13,10             ; ROM CHECKSUM ERROR
F90D 0D
F90E 0A
F90F 31383031           F3C     DB      '1801',13,10            ; EXPANSION IO BOX ERROR
F913 0D
F914 0A
F915 50415249545920     D1      DB      'PARITY CHECK 2',13,10
     434845434B2032
F923 0D
F924 0A
F925 50415249545920     D2      DB      'PARITY CHECK 1',13,10
     434845434B2031
F933 0D
F934 0A
F935 3F3F3F3F3F         D2A     DB      '?????',13,10
F93A 0D
F93B 0A

                        ;-----------------------------------------------------------------------
                        ;       BLINK LED PROCEDURE FOR MFG RUN-IN TESTS
                        ;        IF LED IS ON, TURN IT OFF. IF OFF, TURN ON.
                        ;-----------------------------------------------------------------------
                                ASSUME  DS:DATA
F93C                    BLINK_INT       PROC    NEAR
F93C FB                         STI
F93D 50                         PUSH    AX                      ; SAVE AX REG CONTENTS
F93E E461                       IN      AL,PORT_B               ; READ CURRENT VAL OF PORT B
F940 8AE0                       MOV     AH,AL
F942 F6D0                       NOT     AL                      ; FLIP ALL BITS
F944 2440                       AND     AL,01000000B            ; ISOLATE CONTROL BIT
F946 80E4BF                     AND     AH,10111111B            ; MASK OUT OF ORIGINAL VAL
F949 0AC4                       OR      AL,AH                   ; OR NEW CONTROL BIT IN
F94B E661                       OUT     PORT_B,AL
F94D B020                       MOV     AL,EOI
F94F E620                       OUT     INTA00,AL
F951 58                         POP     AX                      ; RESTORE AX REG
F952 CF                         IRET
                        BLINK_INT       ENDP

                        ;-------------------------------------------------------
                        ; THIS ROUTINE CHECKSUMS OPTIONAL ROM MODULES AND      :
                        ; IF CHECKSUM IS OK, CALLS INIT/TEST CODE IN MODULE    :
                        ;-------------------------------------------------------
F953                    ROM_CHECK       PROC    NEAR
F953 B84000                     MOV     AX,DATA                 ; POINT ES TO DATA AREA
F956 8EC0                       MOV     ES,AX
F958 2AE4                       SUB     AH,AH                   ; ZERO OUT AH
F95A 8A4702                     MOV     AL,[BX+2]               ; GET LENGTH INDICATOR
F95D B109                       MOV     CL,09H                  ; MULTIPLY BY 512
F95F D3E0                       SHL     AX,CL
F961 8BC8                       MOV     CX,AX                   ; SET COUNT
F963 51                         PUSH    CX                      ; SAVE COUNT
F964 B90400                     MOV     CX,4                    ; ADJUST
F967 D3E8                       SHR     AX,CL
F969 03D0                       ADD     DX,AX                   ; SET POINTER TO NEXT MODULE
F96B 59                         POP     CX                      ; RETRIVE COUNT
F96C E886FF                     CALL    ROS_CHECKSUM_CNT        ; DO CHECKSUM
F96F 7406                       JZ      ROM_CHECK_1
F971 E857ED                     CALL    ROM_ERR                 ; POST CHECKSUM ERROR
F974 EB1490                     JMP     ROM_CHECK_END           ; AND EXIT
F977                    ROM_CHECK_1:
F977 52                         PUSH    DX                      ; SAVE POINTER
F978 26C70667000300             MOV     ES:IO_ROM_INIT,0003H    ; LOAD OFFSET
F97F 268C1E6900                 MOV     ES:IO_ROM_SEG,DS        ; LOAD SEGMENT
F984 26FF1E6700                 CALL    DWORD PTR ES:IO_ROM_INIT        ; CALL INIT./TEST ROUTINE
F989 5A                         POP     DX
F98A                    ROM_CHECK_END:
F98A C3                         RET                             ; RETURN TO CALLER
                        ROM_CHECK       ENDP

                        ;-----------------------------------------------
                        ; CONVERT AND PRINT ASCII CODE                 :
                        ;       AL MUST CONTAIN NUMBER TO BE CONVERTED. :
                        ;       AX AND BX DESTROYED.                   :
                        ;-----------------------------------------------
F98B                    XPC_BYTE        PROC    NEAR
F98B 50                         PUSH    AX                      ; SAVE FOR LOW NIBBLE DISPLAY
F98C B104                       MOV     CL,4                    ; SHIFT COUNT
F98E D2E8                       SHR     AL,CL                   ; NYBBLE SWAP
F990 E80300                     CALL    XLAT_PR                 ; DO THE HIGH NIBBLE DISPLAY
F993 58                         POP     AX                      ; RECOVER THE NIBBLE
F994 240F                       AND     AL,0FH                  ; ISOLATE TO LOW NIBBLE
                                                                ; FALL INTO LOW NIBBLE CONVERSION
F996                    XLAT_PR PROC    NEAR                    ;  CONVERT 00-0F TO ASCII CHARACTER
F996 0490                       ADD     AL,090H                 ; ADD FIRST CONVERSION FACTOR
F998 27                         DAA                             ; ADJUST FOR NUMERIC AND ALPHA RANGE
F999 1440                       ADC     AL,040H                 ; ADD CONVERSION AND ADJUST LOW NIBBLE
F99B 27                         DAA                             ; ADJUST HIGH NIBBLE TO ASCHI RANGE
F99C                    PRT_HEX PROC    NEAR
F99C B40E                       MOV     AH,14                   ; DISPLAY CHARACTER IN AL
F99E B700                       MOV     BH,0
F9A0 CD10                       INT     10H                     ; CALL VIDEO_IO
F9A2 C3                         RET
                        PRT_HEX ENDP
                        XLAT_PR ENDP
                        XPC_BYTE        ENDP

F9A3                    F4      LABEL   WORD                    ; PRINTER SOURCE TABLE
F9A3 BC03                       DW      3BCH
F9A5 7803                       DW      378H
F9A7 7802                       DW      278H
F9A9                    F4E     LABEL   WORD

```


SECTION 5

```
                 ;-----------------------------------------------------------
                 ; 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            :
                 ;-----------------------------------------------------------
F9A9             E_MSG   PROC    NEAR
F9A9 8BEE                MOV     BP,SI                   ; SET BP NON-ZERO TO FLAG ERR
F9AB E81C00              CALL    P_MSG                   ; PRINT MESSAGE
F9AE 1E                  PUSH    DS
F9AF E8A700              CALL    DDS
F9B2 A01000              MOV     AL,BYTE PTR EQUIP_FLAG  ; LOOP/HALT ON ERROR
F9B5 2401                AND     AL,01H                  ; SWITCH ON?
F9B7 750F                JNZ     G12                     ; NO - RETURN
F9B9             MFG_HALT:
F9B9 FA                  CLI                             ; YES -  HALT SYSTEM
F9BA B089                MOV     AL,89H
F9BC E663                OUT     CMD_PORT,AL
F9BE B085                MOV     AL,10000101B            ; DISABLE KB
F9C0 E661                OUT     PORT_B,AL
F9C2 A01500              MOV     AL,MFG_ERR_FLAG         ; RECOVER ERROR INDICATOR
F9C5 E660                OUT     PORT_A,AL               ; SET INTO 8255 REG
F9C7 F4                  HLT                             ; HALT SYS
F9C8             G12:
F9C8 1F                  POP     DS                      ; WRITE_MSG:
F9C9 C3                  RET
                 E_MSG   ENDP

F9CA             P_MSG   PROC    NEAR
F9CA             G12A:
F9CA 2E8A04              MOV     AL,CS:[SI]              ; PUT CHAR IN AL
F9CD 46                  INC     SI                      ; POINT TO NEXT CHAR
F9CE 50                  PUSH    AX                      ; SAVE PRINT CHAR
F9CF E8CAFF              CALL    PRT_HEX                 ; CALL VIDEO_IO
F9D2 58                  POP     AX                      ; RECOVER PRINT CHAR
F9D3 3C0A                CMP     AL,10                   ; WAS IT LINE FEED?
F9D5 75F3                JNE     G12A                            ; NO,KEEP PRINTING STRING
F9D7 C3                  RET
                 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.                             :
                 ;-------------------------------------------------------------------------
F9D8             ERR_BEEP  PROC  NEAR
F9D8 9C                  PUSHF                           ; SAVE FLAGS
F9D9 FA                  CLI                             ; DISABLE SYSTEM INTERRUPTS
F9DA 1E                  PUSH    DS                      ; SAVE DS REG CONTENTS
F9DB E87B00              CALL    DDS
F9DE 0AF6                OR      DH,DH                   ; ANY LONG ONES TO BEEP
F9E0 7414                JZ      G3                      ; NO, DO THE SHORT ONES
F9E2             G1:                                     ; LONG_BEEP:
F9E2 B306                MOV     BL,6                    ; COUNTER FOR BEEPS
F9E4 E82100              CALL    BEEP                    ; DO THE BEEP
F9E7             G2:
F9E7 E2FE                LOOP    G2                      ; DELAY BETWEEN BEEPS
F9E9 FECE                DEC     DH                      ; ANY MORE TO DO
F9EB 75F5                JNZ     G1                      ; DO IT
F9ED 803E120001          CMP     MFG_TST,1               ; MFG TEST MODE?
F9F2 7502                JNE     G3                      ; YES - CONTINUE BEEPING SPEAKER
F9F4 EBC3                JMP     MFG_HALT                ; STOP BLINKING LED
F9F6             G3:                                     ; SHORT_BEEP:
F9F6 B301                MOV     BL,1                    ; COUNTER FOR A SHORT BEEP
F9F8 E80D00              CALL    BEEP                    ; DO THE SOUND
F9FB             G4:
F9FB E2FE                LOOP    G4                      ; DELAY BETWEEN BEEPS
F9FD FECA                DEC     DL                      ; DONE WITH SHORTS
F9FF 75F5                JNZ     G3                      ; DO SOME MORE
FA01             G5:
FA01 E2FE                LOOP    G5                      ; LONG DELAY BEFORE RETURN
FA03             G6:
FA03 E2FE                LOOP    G6
FA05 1F                  POP     DS                      ; RESTORE ORIG CONTENTS OF DS
FA06 9D                  POPF                            ; RESTORE FLAGS TO ORIG SETTINGS
FA07 C3                  RET                             ; RETURN TO CALLER
                 ERR_BEEP        ENDP

                 ;----- ROUTINE TO SOUND BEEPER

FA08             BEEP    PROC    NEAR
FA08 B0B6                MOV     AL,10110110B            ; SEL TIM 2,LSB,MSB,BINARY
FA0A E643                OUT     TIMER+3,AL              ; WRITE THE TIMER MODE REG
FA0C B83305              MOV     AX,533H                 ; DIVISOR FOR 1000 HZ
FA0F E642                OUT     TIMER+2,AL              ; WRITE TIMER 2 CNT - LSB
FA11 8AC4                MOV     AL,AH
FA13 E642                OUT     TIMER+2,AL              ; WRITE TIMER 2 CNT - MSB
FA15 E461                IN      AL,PORT_B               ; GET CURRENT SETTING OF PORT
FA17 8AE0                MOV     AH,AL                   ; SAVE THAT SETTINGH
FA19 0C03                OR      AL,03                   ; TURN SPEAKER ON
FA1B E661                OUT     PORT_B,AL
FA1D 2BC9                SUB     CX,CX                   ; SET CNT TO WAIT 500 MS
FA1F             G7:
FA1F E2FE                LOOP    G7                      ; DELAY BEFORE TURNING OFF
FA21 FECB                DEC     BL                      ; DELAY CNT EXPIRED?
FA23 75FA                JNZ     G7                      ; NO - CONTINUE BEEPING SPK
FA25 8AC4                MOV     AL,AH                   ; RECOVER VALUE OF PORT
FA27 E661                OUT     PORT_B,AL
FA29 C3                  RET                             ; RETURN TO CALLER
                 BEEP    ENDP
```

```
LOC OBJECT                 LINE  SOURCE (BIOS FOR THE IBM PERSONAL COMPUTER XT) 11/08/82

                                 ;-----------------------------------------------------------------------
                                 ;         THIS PROCEDURE WILL SEND A SOFTWARE RESET TO THE KEYBOARD.   :
                                 ;         SCAN CODE 'AA' SHOULD BE RETURNED TO THE CPU.                :
                                 ;-----------------------------------------------------------------------
FA2A                             KBD_RESET      PROC     NEAR
                                          ASSUME  DS:ABS0
FA2A B008                                 MOV     AL,08H                 ; SET KBD CLK LINE LOW
FA2C E661                                 OUT     PORT_B,AL              ; WRITE 8255 PORT B
FA2E B95629                               MOV     CX,10582               ; HOLD KBD CLK LOW FOR 20 MS
FA31                             G8:
FA31 E2FE                                 LOOP    G8                     ; LOOP FOR 20 MS
FA33 B0C8                                 MOV     AL,0C8H                ; SET CLK, ENABLE LINES HIGH
FA35 E661                                 OUT     PORT_B,AL
FA37                             SP_TEST:                                ; ENTRY FOR MANUFACTURING TEST 2
FA37 B048                                 MOV     AL,48H                 ; SET KBD CLK HIGH, ENABLE LOW
FA39 E661                                 OUT     PORT_B,AL
FA3B B0FD                                 MOV     AL,0FDH                ; ENABLE KEYBOARD INTERRUPTS
FA3D E621                                 OUT     INTA01,AL              ; WRITE 8259 IMR
FA3F C6066B0400                           MOV     DATA_AREA[OFFSET INTR_FLAG]     ; RESET INTERRUPT INDICATOR
FA44 FB                                   STI                            ; ENABLE INTERRUPTS
FA45 2BC9                                 SUB     CX,CX                  ; SETUP INTERRUPT TIMEOUT CNT
FA47                             G9:
FA47 F6066B0402                           TEST    DATA_AREA[OFFSET INT R_FLAG],02H ; DID A KEYBOARD INTR OCCUR?
FA4C 7502                                 JNZ     G10                    ; YES - READ SCAN CODE RETURNED
FA4E E2F7                                 LOOP    G9                     ; NO - LOOP TILL TIMEOUT
FA50                             G10:
FA50 E460                                 IN      AL,PORT_A              ; READ KEYBOARD SCAN CODE
FA52 8AD8                                 MOV     BL,AL                  ; SAVE SCAN CODE JUST READ
FA54 B0C8                                 MOV     AL,0C8H                ; CLEAR KEYBOARD
FA56 E661                                 OUT     PORT_B,AL
FA58 C3                                   RET                            ; RETURN TO CALLER
                                 KBD_RESET        ENDP

FA59                             DDS      PROC    NEAR
FA59 50                                   PUSH    AX                     ; SAVE AX
FA5A B84000                               MOV     AX,DATA
FA5D 8ED8                                 MOV     DS,AX                  ; SET SEGMENT
FA5F 58                                   POP     AX                     ; RESTORE AX
FA60 C3                                   RET
                                 DDS      ENDP

                                 ;-----------------------------------------------------------------------
                                 ;        CHARACTER GENERATOR GRAPHICS FOR 320X200 AND 640X200 GRAPHICS  :
                                 ;-----------------------------------------------------------------------
FA6E                                      ORG     0FA6EH
FA6E                             CRT_CHAR_GEN     LABEL   BYTE
FA6E 0000000000000000                     DB      000H,000H,000H,000H,000H,000H,000H,000H ; D_00
FA76 7E81A581BD99817E                     DB      07EH,081H,0A5H,081H,0BDH,099H,081H,07EH ; D_01
FA7E 7EFFDBFFC3E7FF7E                     DB      07EH,0FFH,0DBH,0FFH,0C3H,0E7H,0FFH,07EH ; D_02
FA86 6CFEFEFE7C381000                     DB      06CH,0FEH,0FEH,0FEH,07CH,038H,010H,000H ; D_03
FA8E 10387CFE7C381000                     DB      010H,038H,07CH,0FEH,07CH,038H,010H,000H ; D_04
FA96 387C38FEFE7C387C                     DB      038H,07CH,038H,0FEH,0FEH,07CH,038H,07CH ; D_05
FA9E 1010387CFE7C387C                     DB      010H,010H,038H,07CH,0FEH,07CH,038H,07CH ; D_06
FAA6 0000183C3C180000                     DB      000H,000H,018H,03CH,03CH,018H,000H,000H ; D_07
FAAE FFFFE7C3C3E7FFFF                     DB      0FFH,0FFH,0E7H,0C3H,0C3H,0E7H,0FFH,0FFH ; D_08
FAB6 003C664242663C00                     DB      000H,03CH,066H,042H,042H,066H,03CH,000H ; D_09
FABE FFC399BDBD99C3FF                     DB      0FFH,0C3H,099H,0BDH,0BDH,099H,0C3H,0FFH ; D_0A
FAC6 0F070F7DCCCCCC78                     DB      00FH,007H,00FH,07DH,0CCH,0CCH,0CCH,078H ; D_0B
FACE 3C6666663C187E18                     DB      03CH,066H,066H,066H,03CH,018H,07EH,018H ; D_0C
FAD6 3F333F303070F0E0                     DB      03FH,033H,03FH,030H,030H,070H,0F0H,0E0H ; D_0D
FADE 7F637F636367E6C0                     DB      07FH,063H,07FH,063H,063H,067H,0E6H,0C0H ; D_0E
FAE6 995A3CE7E73C5A99                     DB      099H,05AH,03CH,0E7H,0E7H,03CH,05AH,099H ; D_0F
FAEE 80E0F8FEF8E08000                     DB      080H,0E0H,0F8H,0FEH,0F8H,0E0H,080H,000H ; D_10
FAF6 020E3EFE3E0E0200                     DB      002H,00EH,03EH,0FEH,03EH,00EH,002H,000H ; D_11
FAFE 183C7E18187E3C18                     DB      018H,03CH,07EH,018H,018H,07EH,03CH,018H ; D_12
FB06 6666666666006600                     DB      066H,066H,066H,066H,066H,000H,066H,000H ; D_13
FB0E 7FDBDB7B1B1B1B00                     DB      07FH,0DBH,0DBH,07BH,01BH,01BH,01BH,000H ; D_14
FB16 3E63386C6C38CC78                     DB      03EH,063H,038H,06CH,06CH,038H,0CCH,078H ; D_15
FB1E 000000007E7E7E00                     DB      000H,000H,000H,000H,07EH,07EH,07EH,000H ; D_16
FB26 183C7E187E3C18FF                     DB      018H,03CH,07EH,018H,07EH,03CH,018H,0FFH ; D_17
FB2E 183C7E1818181800                     DB      018H,03CH,07EH,018H,018H,018H,018H,000H ; D_18
FB36 181818187E3C1800                     DB      018H,018H,018H,018H,07EH,03CH,018H,000H ; D_19
FB3E 00180CFE0C180000                     DB      000H,018H,00CH,0FEH,00CH,018H,000H,000H ; D_1A
FB46 003060FE60300000                     DB      000H,030H,060H,0FEH,060H,030H,000H,000H ; D_1B
FB4E 0000C0C0C0FE0000                     DB      000H,000H,0C0H,0C0H,0C0H,0FEH,000H,000H ; D_1C
FB56 002466FF66240000                     DB      000H,024H,066H,0FFH,066H,024H,000H,000H ; D_1D
FB5E 00183C7EFFFF0000                     DB      000H,018H,03CH,07EH,0FFH,0FFH,000H,000H ; D_1E
FB66 00FFFF7E3C180000                     DB      000H,0FFH,0FFH,07EH,03CH,018H,000H,000H ; D_1F
FB6E 0000000000000000                     DB      000H,000H,000H,000H,000H,000H,000H,000H ; SP D_20
FB76 3078783030003000                     DB      030H,078H,078H,030H,030H,000H,030H,000H ; ! D_21
FB7E 6C6C6C0000000000                     DB      06CH,06CH,06CH,000H,000H,000H,000H,000H ; " D_22
FB86 6C6CFE6CFE6C6C00                     DB      06CH,06CH,0FEH,06CH,0FEH,06CH,06CH,000H ; # D_23
FB8E 307CC0780CF83000                     DB      030H,07CH,0C0H,078H,00CH,0F8H,030H,000H ; $ D_24
FB96 00C6CC183066C600                     DB      000H,0C6H,0CCH,018H,030H,066H,0C6H,000H ; PER CENT D_25
FB9E 386C3876DCCC7600                     DB      038H,06CH,038H,076H,0DCH,0CCH,076H,000H ; & D_26
FBA6 6060C00000000000                     DB      060H,060H,0C0H,000H,000H,000H,000H,000H ; ' D_27
FBAE 1830606060301800                     DB      018H,030H,060H,060H,060H,030H,018H,000H ; ( D_28
FBB6 6030181818306000                     DB      060H,030H,018H,018H,018H,030H,060H,000H ; ) D_29
FBBE 00663CFF3C660000                     DB      000H,066H,03CH,0FFH,03CH,066H,000H,000H ; * D_2A
FBC6 003030FC30300000                     DB      000H,030H,030H,0FCH,030H,030H,000H,000H ; + D_2B
FBCE 0000000000303060                     DB      000H,000H,000H,000H,000H,030H,030H,060H ; , D_2C
FBD6 000000FC00000000                     DB      000H,000H,000H,0FCH,000H,000H,000H,000H ; - D_2D
FBDE 0000000000303000                     DB      000H,000H,000H,000H,000H,030H,030H,000H ; . D_2E
FBE6 060C183060C08000                     DB      006H,00CH,018H,030H,060H,0C0H,080H,000H ; / D_2F
FBEE 7CC6CEDEF6E67C00                     DB      07CH,0C6H,0CEH,0DEH,0F6H,0E6H,07CH,000H ; 0 D_30
FBF6 307030303030FC00                     DB      030H,070H,030H,030H,030H,030H,0FCH,000H ; 1 D_31
FBFE 78CC0C3860CCFC00                     DB      078H,0CCH,00CH,038H,060H,0CCH,0FCH,000H ; 2 D_32
FC06 78CC0C380CCC7800                     DB      078H,0CCH,00CH,038H,00CH,0CCH,078H,000H ; 3 D_33
FC0E 1C3C6CCCFE0C1E00                     DB      01CH,03CH,06CH,0CCH,0FEH,00CH,01EH,000H ; 4 D_34
FC16 FCC0F80C0CCC7800                     DB      0FCH,0C0H,0F8H,00CH,00CH,0CCH,078H,000H ; 5 D_35
FC1E 3860C0F8CCCC7800                     DB      038H,060H,0C0H,0F8H,0CCH,0CCH,078H,000H ; 6 D_36
FC26 FCCC0C1830303000                     DB      0FCH,0CCH,00CH,018H,030H,030H,030H,000H ; 7 D_37
FC2E 78CCCC78CCCC7800                     DB      078H,0CCH,0CCH,078H,0CCH,0CCH,078H,000H ; 8 D_38
FC36 78CCCC7C0C187000                     DB      078H,0CCH,0CCH,07CH,00CH,018H,070H,000H ; 9 D_39
FC3E 0030300000303000                     DB      000H,030H,030H,000H,000H,030H,030H,000H ; : D_3A
FC46 0030300000303060                     DB      000H,030H,030H,000H,000H,030H,030H,060H ; ; D_3B
FC4E 183060C060301800                     DB      018H,030H,060H,0C0H,060H,030H,018H,000H ; < D_3C
FC56 0000FC0000FC0000                     DB      000H,000H,0FCH,000H,000H,0FCH,000H,000H ; = D_3D
FC5E 6030180C18306000                     DB      060H,030H,018H,00CH,018H,030H,060H,000H ; > D_3E
FC66 78CC0C1830003000                     DB      078H,0CCH,00CH,018H,030H,000H,030H,000H ; ? D_3F
```

SECTION 5

```
FC6E 7CC6DEDEDEC07800          DB      07CH,0C6H,0DEH,0DEH,0DEH,0C0H,078H,000H ; @ D_40
FC76 3078CCCCFCCCCC00          DB      030H,078H,0CCH,0CCH,0FCH,0CCH,0CCH,000H ; A D_41
FC7E FC66667C6666FC00          DB      0FCH,066H,066H,07CH,066H,066H,0FCH,000H ; B D_42
FC86 3C66C0C0C0663C00          DB      03CH,066H,0C0H,0C0H,0C0H,066H,03CH,000H ; C D_43
FC8E F86C6666666CF800          DB      0F8H,06CH,066H,066H,066H,06CH,0F8H,000H ; D D_44
FC96 FE6268786862FE00          DB      0FEH,062H,068H,078H,068H,062H,0FEH,000H ; E D_45
FC9E FE6268786860F000          DB      0FEH,062H,068H,078H,068H,060H,0F0H,000H ; F D_46
FCA6 3C66C0C0CE663E00          DB      03CH,066H,0C0H,0C0H,0CEH,066H,03EH,000H ; G D_47
FCAE CCCCCCFCCCCCCC00          DB      0CCH,0CCH,0CCH,0FCH,0CCH,0CCH,0CCH,000H ; H D_48
FCB6 7830303030307800          DB      078H,030H,030H,030H,030H,030H,078H,000H ; I D_49
FCBE 1E0C0C0CCCCC7800          DB      01EH,00CH,00CH,00CH,0CCH,0CCH,078H,000H ; J D_4A
FCC6 E6666C786C66E600          DB      0E6H,066H,06CH,078H,06CH,066H,0E6H,000H ; K D_4B
FCCE F06060606266FE00          DB      0F0H,060H,060H,060H,062H,066H,0FEH,000H ; L D_4C
FCD6 C6EEFEFED6C6C600          DB      0C6H,0EEH,0FEH,0FEH,0D6H,0C6H,0C6H,000H ; M D_4D
FCDE C6E6F6DECEC6C600          DB      0C6H,0E6H,0F6H,0DEH,0CEH,0C6H,0C6H,000H ; N D_4E
FCE6 386CC6C6C66C3800          DB      038H,06CH,0C6H,0C6H,0C6H,06CH,038H,000H ; O D_4F
FCEE FC66667C6060F000          DB      0FCH,066H,066H,07CH,060H,060H,0F0H,000H ; P D_50
FCF6 78CCCCCCDC781C00          DB      078H,0CCH,0CCH,0CCH,0DCH,078H,01CH,000H ; Q D_51
FCFE FC66667C6C66E600          DB      0FCH,066H,066H,07CH,06CH,066H,0E6H,000H ; R D_52
FD06 78CCE0701CCC7800          DB      078H,0CCH,0E0H,070H,01CH,0CCH,078H,000H ; S D_53
FD0E FCB4303030307800          DB      0FCH,0B4H,030H,030H,030H,030H,078H,000H ; T D_54
FD16 CCCCCCCCCCCCFC00          DB      0CCH,0CCH,0CCH,0CCH,0CCH,0CCH,0FCH,000H ; U D_55
FD1E CCCCCCCCCC783000          DB      0CCH,0CCH,0CCH,0CCH,0CCH,078H,030H,000H ; V D_56
FD26 C6C6C6D6FEEEC600          DB      0C6H,0C6H,0C6H,0D6H,0FEH,0EEH,0C6H,000H ; W D_57
FD2E C6C66C38386CC600          DB      0C6H,0C6H,06CH,038H,038H,06CH,0C6H,000H ; X D_58
FD36 CCCCCC7830307800          DB      0CCH,0CCH,0CCH,078H,030H,030H,078H,000H ; Y D_59
FD3E FEC68C183266FE00          DB      0FEH,0C6H,08CH,018H,032H,066H,0FEH,000H ; Z D_5A
FD46 7860606060607800          DB      078H,060H,060H,060H,060H,060H,078H,000H ; [ D_5B
FD4E C06030180C060200          DB      0C0H,060H,030H,018H,00CH,006H,002H,000H ; BACKSLASH D_5C
FD56 7818181818187800          DB      078H,018H,018H,018H,018H,018H,078H,000H ; ] D_5D
FD5E 10386CC600000000          DB      010H,038H,06CH,0C6H,000H,000H,000H,000H ; CIRCUMFLEX D_5E
FD66 00000000000000FF          DB      000H,000H,000H,000H,000H,000H,000H,0FFH ; _ D_5F
FD6E 3030180000000000          DB      030H,030H,018H,000H,000H,000H,000H,000H ; ` D_60
FD76 0000780C7CCC7600          DB      000H,000H,078H,00CH,07CH,0CCH,076H,000H ; LOWER CASE A D_61
FD7E E060607C6666DC00          DB      0E0H,060H,060H,07CH,066H,066H,0DCH,000H ; L.C. B D_62
FD86 000078CCC0CC7800          DB      000H,000H,078H,0CCH,0C0H,0CCH,078H,000H ; L.C. C D_63
FD8E 1C0C0C7CCCCC7600          DB      01CH,00CH,00CH,07CH,0CCH,0CCH,076H,000H ; L.C. D D_64
FD96 000078CCFCC07800          DB      000H,000H,078H,0CCH,0FCH,0C0H,078H,000H ; L.C. E D_65
FD9E 386C60F06060F000          DB      038H,06CH,060H,0F0H,060H,060H,0F0H,000H ; L.C. F D_66
FDA6 000076CCCC7C0CF8          DB      000H,000H,076H,0CCH,0CCH,07CH,00CH,0F8H ; L.C. G D_67
FDAE E0606C766666E600          DB      0E0H,060H,06CH,076H,066H,066H,0E6H,000H ; L.C. H D_68
FDB6 3000703030307800          DB      030H,000H,070H,030H,030H,030H,078H,000H ; L.C. I D_69
FDBE 0C000C0C0CCCCC78          DB      00CH,000H,00CH,00CH,00CH,0CCH,0CCH,078H ; L.C. J D_6A
FDC6 E060666C786CE600          DB      0E0H,060H,066H,06CH,078H,06CH,0E6H,000H ; L.C. K D_6B
FDCE 7030303030307800          DB      070H,030H,030H,030H,030H,030H,078H,000H ; L.C. L D_6C
FDD6 0000CCFEFED6C600          DB      000H,000H,0CCH,0FEH,0FEH,0D6H,0C6H,000H ; L.C. M D_6D
FDDE 0000F8CCCCCCCC00          DB      000H,000H,0F8H,0CCH,0CCH,0CCH,0CCH,000H ; L.C. N D_6E
FDE6 000078CCCCCC7800          DB      000H,000H,078H,0CCH,0CCH,0CCH,078H,000H ; L.C. O D_6F
FDEE 0000DC66667C60F0          DB      000H,000H,0DCH,066H,066H,07CH,060H,0F0H ; L.C. P D_70
FDF6 000076CCCC7C0C1E          DB      000H,000H,076H,0CCH,0CCH,07CH,00CH,01EH ; L.C. Q D_71
FDFE 0000DC766660F000          DB      000H,000H,0DCH,076H,066H,060H,0F0H,000H ; L.C. R D_72
FE06 00007CC0780CF800          DB      000H,000H,07CH,0C0H,078H,00CH,0F8H,000H ; L.C. S D_73
FE0E 10307C3030341800          DB      010H,030H,07CH,030H,030H,034H,018H,000H ; L.C. T D_74
FE16 0000CCCCCCCC7600          DB      000H,000H,0CCH,0CCH,0CCH,0CCH,076H,000H ; L.C. U D_75
FE1E 0000CCCCCC783000          DB      000H,000H,0CCH,0CCH,0CCH,078H,030H,000H ; L.C. V D_76
FE26 0000C6D6FEFE6C00          DB      000H,000H,0C6H,0D6H,0FEH,0FEH,06CH,000H ; L.C. W D_77
FE2E 0000C66C386CC600          DB      000H,000H,0C6H,06CH,038H,06CH,0C6H,000H ; L.C. X D_78
FE36 0000CCCCCC7C0CF8          DB      000H,000H,0CCH,0CCH,0CCH,07CH,00CH,0F8H ; L.C. Y D_79
FE3E 0000FC983064FC00          DB      000H,000H,0FCH,098H,030H,064H,0FCH,000H ; L.C. Z D_7A
FE46 1C3030E030301C00          DB      01CH,030H,030H,0E0H,030H,030H,01CH,000H ; { D_7B
FE4E 1818180018181800          DB      018H,018H,018H,000H,018H,018H,018H,000H ; | D_7C
FE56 E030301C3030E000          DB      0E0H,030H,030H,01CH,030H,030H,0E0H,000H ; } D_7D
FE5E 76DC000000000000          DB      076H,0DCH,000H,000H,000H,000H,000H,000H ; TILDE D_7E
FE66 0010386CC6C6FE00          DB      000H,010H,038H,06CH,0C6H,0C6H,0FEH,000H ; DELTA D_7F

                         ;--- 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 BELOW) :
                         ;------------------------------------------------------
                                 ASSUME  CS:CODE,DS:DATA
FE6E                             ORG     0FE6EH
FE6E                     TIME_OF_DAY     PROC    FAR
FE6E FB                          STI                             ; INTERRUPTS BACK ON
FE6F 1E                          PUSH    DS                      ; SAVE SEGMENT
FE70 E8E6FB                      CALL    DDS
FE73 0AE4                        OR      AH,AH                   ; AH=0
FE75 7407                        JZ      T2                      ; READ_TIME
FE77 FECC                        DEC     AH                      ; AH=1
FE79 7416                        JZ      T3                      ; SET_TIME
FE7B                     T1:                                     ; TOD_RETURN
FE7B FB                          STI                             ; INTERRUPTS BACK ON
FE7C 1F                          POP     DS                      ; RECOVER SEGMENT
FE7D CF                          IRET                            ; RETURN TO CALLER
FE7E                     T2:                                     ; READ_TIME
FE7E FA                          CLI                             ; NO TIMER INTERRUPTS WHILE READING
FE7F A07000                      MOV     AL,TIMER_OFL
FE82 C606700000                  MOV     TIMER_OFL,0             ; GET OVERFLOW, AND RESET THE FLAG
FE87 8B0E6E00                    MOV     CX,TIMER_HIGH
FE8B 8B166C00                    MOV     DX,TIMER_LOW
FE8F EBEA                        JMP     T1                      ; TOD_RETURN
FE91                     T3:                                     ; SET_TIME
FE91 FA                          CLI                             ; NO INTERRUPTS WHILE WRITING
FE92 89166C00                    MOV     TIMER_LOW,DX
FE96 890E6E00                    MOV     TIMER_HIGH,CX           ; SET THE TIME
FE9A C606700000                  MOV     TIMER_OFL,0             ; RESET OVERFLOW
FE9F EBDA                        JMP     T1                      ; TOD_RETURN
                         TIME_OF_DAY     ENDP
```

```
                ;--------------------------------------------------------
                ; 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, 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.                                     :
                ;--------------------------------------------------------
FEA5                            ORG     0FEA5H
FEA5            TIMER_INT       PROC    FAR
FEA5 FB                 STI                             ; INTERRUPTS BACK ON
FEA6 1E                 PUSH    DS
FEA7 50                 PUSH    AX
FEA8 52                 PUSH    DX                      ; SAVE MACHINE STATE
FEA9 E8ADFB             CALL    DDS
FEAC FF066C00           INC     TIMER_LOW               ; INCREMENT TIME
FEB0 7504               JNZ     T4                      ; TEST_DAY
FEB2 FF066E00           INC     TIMER_HIGH              ; INCREMENT HIGH WORD OF TIME
FEB6            T4:                                     ; TEST_DAY
FEB6 833E6E0018         CMP     TIMER_HIGH,018H         ; TEST FOR COUNT EQUALING 24 HOURS
FEBB 7515               JNZ     T5                      ; DISKETTE_CTL
FEBD 813E6C00B000       CMP     TIMER_LOW,0B0H
FEC3 750D               JNZ     T5                      ; DISKETTE_CTL

                ;------ TIMER HAS GONE 24 HOURS

FEC5 2BC0               SUB     AX,AX
FEC7 A36E00             MOV     TIMER_HIGH,AX
FECA A36C00             MOV     TIMER_LOW,AX
FECD C606700001         MOV     TIMER_OFL,1

                ;------ TEST FOR DISKETTE TIME OUT

FED2            T5:                                     ; DISKETTE_CTL
FED2 FE0E4000           DEC     MOTOR_COUNT
FED6 750B               JNZ     T6                      ; RETURN IF COUNT NOT OUT
FED8 80263F00F0         AND     MOTOR_STATUS,0F0H       ; TURN OFF MOTOR RUNNING BITS
FEDD B00C               MOV     AL,0CH
FEDF BAF203             MOV     DX,03F2H                ; FDC CTL PORT
FEE2 EE                 OUT     DX,AL                   ; TURN OFF THE MOTOR
FEE3            T6:                                     ; TIMER_RET:
FEE3 CD1C               INT     1CH                     ; TRANSFER CONTROL TO A USER ROUTINE
FEE5 B020               MOV     AL,EOI
FEE7 E620               OUT     020H,AL                 ; END OF INTERRUPT TO 8259
FEE9 5A                 POP     DX
FEEA 58                 POP     AX
FEEB 1F                 POP     DS                      ; RESET MACHINE STATE
FEEC CF                 IRET                            ; RETURN FROM INTERRUPT
                TIMER_INT       ENDP
```

SECTION 5

```
                  ;----------------------------------------------
                  ; THESE ARE THE VECTORS WHICH ARE MOVED INTO    :
                  ; THE 8086 INTERRUPT AREA DURING POWER ON.      :
                  ; ONLY THE OFFSETS ARE DISPLAYED HERE, CODE     :
                  ; SEGMENT WILL BE ADDED FOR ALL OF THEM, EXCEPT :
                  ; WHERE NOTED.                                  :
                  ;----------------------------------------------
                          ASSUME  CS:CODE
FEF3                      ORG     0FEF3H
FEF3              VECTOR_TABLE    LABEL   WORD          ; VECTOR TABLE FOR MOVE TO INTERRUPTS
FEF3 A5FE                 DW      OFFSET TIMER_INT      ; INTERRUPT 8
FEF5 87E9                 DW      OFFSET  KB_INT        ; INTERRUPT 9
FEF7 23FF                 DW      OFFSET  D11           ; INTERRUPT A
FEF9 23FF                 DW      OFFSET  D11           ; INTERRUPT B
FEFB 23FF                 DW      OFFSET  D11           ; INTERRUPT C
FEFD 23FF                 DW      OFFSET  D11           ; INTERRUPT D
FEFF 57EF                 DW      OFFSET DISK_INT       ; INTERRUPT E
FF01 23FF                 DW      OFFSET  D11           ; INTERRUPT F
FF03 65F0                 DW      OFFSET VIDEO_IO       ; INTERRUPT 10H
FF05 4DF8                 DW      OFFSET EQUIPMENT      ; INTERRUPT 11H
FF07 41F8                 DW      OFFSET MEMORY_SIZE_DET ; INTERRUPT 12H
FF09 59EC                 DW      OFFSET DISKETTE_IO    ; INTERRUPT 13H
FF0B 39E7                 DW      OFFSET RS232_IO       ; INTERRUPT 14H
FF0D 59F8                 DW      CASSETTE_IO           ; INTERRUPT 15H(FORMER CASSETTE IO)
FF0F 2EE8                 DW      OFFSET KEYBOARD_IO    ; INTERRUPT 16H
FF11 D2EF                 DW      OFFSET PRINTER_IO     ; INTERRUPT 17H

FF13 0000                 DW      00000H                ; INTERRUPT 18H
                  ;       DW      0F600H                ;  MUST BE INSERTED INTO TABLE LATER

FF15 F2E6                 DW      OFFSET BOOT_STRAP     ; INTERRUPT 19H
FF17 6EFE                 DW      TIME_OF_DAY           ; INTERRUPT 1AH -- TIME OF DAY
FF19 4BFF                 DW      DUMMY_RETURN          ; INTERRUPT 1BH -- KEYBOARD BREAK ADDR
FF1B 4BFF                 DW      DUMMY_RETURN          ; INTERRUPT 1C -- TIMER BREAK ADDR
FF1D A4F0                 DW      VIDEO_PARMS           ; INTERRUPT 1D -- VIDEO PARAMETERS
FF1F C7EF                 DW      OFFSET DISK_BASE      ; INTERRUPT 1E -- DISK PARMS
FF21 0000                 DW      0                     ; INTERRUPT 1F -- POINTER TO VIDEO EXT

                  ;-------------------------------------------------------
                  ; 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.                             :
                  ;-------------------------------------------------------
FF23              D11     PROC    NEAR
                          ASSUME  DS:DATA
FF23 1E                   PUSH    DS
FF24 52                   PUSH    DX
FF25 50                   PUSH    AX                    ; SAVE REG AX CONTENTS
FF26 E830FB               CALL    DDS
FF29 B00B                 MOV     AL,0BH                ; READ IN-SERVICE REG
FF2B E620                 OUT     INTA00,AL             ; (FIND OUT WHAT LEVEL BEING
FF2D 90                   NOP                           ; SERVICED)
FF2E E420                 IN      AL,INTA00             ; GET LEVEL
FF30 8AE0                 MOV     AH,AL                 ; SAVE IT
FF32 0AC4                 OR      AL,AH                 ; 00? (NO HARDWARE ISR ACTIVE)
FF34 7504                 JNZ     HW_INT
FF36 B4FF                 MOV     AH,0FFH
FF38 EB0A                 JMP     SHORT SET_INTR_FLAG   ; SET FLAG TO FF IF NON-HDWARE
FF3A              HW_INT:
FF3A E421                 IN      AL,INTA01             ; GET MASK VALUE
FF3C 0AC4                 OR      AL,AH                 ; MASK OFF LVL BEING SERVICED
FF3E E621                 OUT     INTA01,AL
FF40 B020                 MOV     AL,EOI
FF42 E620                 OUT     INTA00,AL
FF44              SET_INTR_FLAG:
FF44 88266B00             MOV     INTR_FLAG,AH          ; SET FLAG
FF48 58                   POP     AX                    ; RESTORE REG AX CONTENTS
FF49 5A                   POP     DX
FF4A 1F                   POP     DS
FF4B              DUMMY_RETURN:                         ; NEED IRET FOR VECTOR TABLE
FF4B CF                   IRET
                  D11     ENDP

                  ;----------------------------------------------
                  ; DUMMY RETURN FOR ADDRESS COMPATIBILITY        :
                  ;----------------------------------------------
FF53                      ORG     0FF53H
FF53 CF                   IRET
```

```
                        ;-- 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
FF54                            ORG     0FF54H
FF54                    PRINT_SCREEN    PROC    FAR
FF54 FB                         STI                             ; MUST RUN WITH INTERRUPTS ENABLED
FF55 1E                         PUSH    DS                      ; MUST USE 50:0 FOR DATA AREA STORAGE
FF56 50                         PUSH    AX
FF57 53                         PUSH    BX
FF58 51                         PUSH    CX                      ; WILL USE THIS LATER FOR CURSOR LIMITS
FF59 52                         PUSH    DX                      ; WILL HOLD CURRENT CURSOR POSITION
FF5A B85000                     MOV     AX,XXDATA               ; HEX 50
FF5D 8ED8                       MOV     DS,AX
FF5F 803E000001                 CMP     STATUS_BYTE,1           ; SEE IF PRINT ALREADY IN PROGRESS
FF64 745F                       JZ      EXIT                    ; JUMP IF PRINT ALREADY IN PROGRESS
FF66 C606000001                 MOV     STATUS_BYTE,1           ; INDICATE PRINT NOW IN PROGRESS
FF6B B40F                       MOV     AH,15                   ; WILL REQUEST THE CURRENT SCREEN MODE
FF6D CD10                       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. [A] HAS VIDEO MODE           :
                        ;-----------------------------------------------------------------
FF6F 8ACC                       MOV     CL,AH                   ; WILL MAKE USE OF [CX] REGISTER TO
FF71 B519                       MOV     CH,25                   ; CONTROL ROW & COLUMNS
FF73 E85500                     CALL    CRLF                    ; CARRIAGE RETURN LINE FEED ROUTINE
FF76 51                         PUSH    CX                      ; SAVE SCREEN BOUNDS
FF77 B403                       MOV     AH,3                    ; WILL NOW READ THE CURSOR.
FF79 CD10                       INT     10H                     ; AND PRESERVE THE POSITION
FF7B 59                         POP     CX                      ; RECALL SCREEN BOUNDS
FF7C 52                         PUSH    DX                      ; RECALL [BH]=VISUAL PAGE
FF7D 33D2                       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.                                       :
                        ;-----------------------------------------------------------------
FF7F                    PRI10:
FF7F B402                       MOV     AH,2                    ; TO INDICATE CURSOR SET REQUEST
FF81 CD10                       INT     10H                     ; NEW CURSOR POSITION ESTABLISHED
FF83 B408                       MOV     AH,8                    ; TO INDICATE READ CHARACTER
FF85 CD10                       INT     10H                     ; CHARACTER NOW IN [AL]
FF87 0AC0                       OR      AL,AL                   ; SEE IF VALID CHAR
FF89 7502                       JNZ     PRI15                   ; JUMP IF VALID CHAR
FF8B B020                       MOV     AL,' '                  ; MAKE A BLANK
FF8D                    PRI15:
FF8D 52                         PUSH    DX                      ; SAVE CURSOR POSITION
FF8E 33D2                       XOR     DX,DX                   ; INDICATE PRINTER 1
FF90 32E4                       XOR     AH,AH                   ; TO INDICATE PRINT CHAR IN [AL]
FF92 CD17                       INT     17H                     ; PRINT THE CHARACTER
FF94 5A                         POP     DX                      ; RECALL CURSOR POSITION
FF95 F6C425                     TEST    AH, 25H                 ; TEST FOR PRINTER ERROR
FF98 7521                       JNZ     ERR10                   ; JUMP IF ERROR DETECTED
FF9A FEC2                       INC     DL                      ; ADVANCE TO NEXT COLUMN
FF9C 3ACA                       CMP     CL,DL                   ; SEE IF AT END OF LINE
FF9E 75DF                       JNZ     PRI10                   ; IF NOT PROCEED
FFA0 32D2                       XOR     DL,DL                   ; BACK TO COLUMN 0
FFA2 8AE2                       MOV     AH,DL                   ; [AH]=0
FFA4 52                         PUSH    DX                      ; SAVE NEW CURSOR POSITION
FFA5 E82300                     CALL    CRLF                    ; LINE FEED CARRIAGE RETURN
FFA8 5A                         POP     DX                      ; RECALL CURSOR POSITION
FFA9 FEC6                       INC     DH                      ; ADVANCE TO NEXT LINE
FFAB 3AEE                       CMP     CH,DH                   ; FINISHED?
FFAD 75D0                       JNZ     PRI10                   ; IF NOT CONTINUE
FFAF                    PRI20:
FFAF 5A                         POP     DX                      ; RECALL CURSOR POSITION
FFB0 B402                       MOV     AH,2                    ; TO INDICATE CURSOR SET REQUEST
FFB2 CD10                       INT     10H                     ; CURSOR POSITION RESTORED
FFB4 C606000000                 MOV     STATUS_BYTE,0           ; INDICATE FINISHED
FFB9 EB0A                       JMP     SHORT EXIT              ; EXIT THE ROUTINE
FFBB                    ERR10:
FFBB 5A                         POP     DX                      ; GET CURSOR POSITION
FFBC B402                       MOV     AH,2                    ; TO REQUEST CURSOR SET
FFBE CD10                       INT     10H                     ; CURSOR POSITION RESTORED
FFC0                    ERR20:
FFC0 C6060000FF                 MOV     STATUS_BYTE,0FFH        ; INDICATE ERROR
FFC5                    EXIT:
FFC5 5A                         POP     DX                      ; RESTORE ALL THE REGISTERS USED
FFC6 59                         POP     CX
FFC7 5B                         POP     BX
FFC8 58                         POP     AX
FFC9 1F                         POP     DS
FFCA CF                         IRET
                        PRINT_SCREEN    ENDP

                        ;------ CARRIAGE RETURN, LINE FEED SUBROUTINE

FFCB                    CRLF    PROC    NEAR
FFCB 33D2                       XOR     DX,DX                   ; PRINTER 0
FFCD 32E4                       XOR     AH,AH                   ; WILL NOW SEND INITIAL LF,CR
                                                                ;  TO PRINTER
FFCF B00A                       MOV     AL,12Q                  ; LF
FFD1 CD17                       INT     17H                     ; SEND THE LINE FEED
FFD3 32E4                       XOR     AH,AH                   ; NOW FOR THE CR
FFD5 B00D                       MOV     AL,15Q                  ; CR
FFD7 CD17                       INT     17H                     ; SEND THE CARRIAGE RETURN
FFD9 C3                         RET
                        CRLF    ENDP
```

SECTION 5

```
                        ;-------------------------------------------------------------
                        ;       PRINT A SEGMENT VALUE TO LOOK LIKE A 20 BIT ADDRESS    :
                        ;       DX MUST CONTAIN SEGMENT VALUE TO BE PRINTED            :
                        ;-------------------------------------------------------------
FFDA                    PRT_SEG PROC    NEAR
FFDA 8AC6                       MOV     AL,DH                   ;GET MSB
FFDC E8ACF9                     CALL    XPC_BYTE
FFDF 8AC2                       MOV     AL,DL                   ;LSB
FFE1 E8A7F9                     CALL    XPC_BYTE
FFE4 B030                       MOV     AL,'0'                  ; PRINT A '0 '
FFE6 E8B3F9                     CALL    PRT_HEX
FFE9 B020                       MOV     AL,' '                  ;SPACE
FFEB E8AEF9                     CALL    PRT_HEX
FFEE C3                         RET
                        PRT_SEG ENDP

----                    CODE    ENDS

                        ;-------------------------------
                        ;       POWER ON RESET VECTOR   :
                        ;-------------------------------
----                    VECTOR  SEGMENT AT 0FFFFH

                        ;----- POWER ON RESET

0000 EA5BE000F0                 JMP     RESET

0005 31312F30382F38             DB      '11/08/82'              ; RELEASE MARKER
     32
----                    VECTOR  ENDS
                                END
```

SECTION 6. INSTRUCTION SET

8088 Register Model 6-3
Operand Summary 6-4
Second Instruction Byte Summary 6-4
Memory Segmentation Model 6-5
Segment Override Prefix 6-6
Use of Segment Override 6-6
8088 Instruction Set 6-7
Data Transfer 6-7
Arithmetic 6-10
Logic 6-13
String Manipulation 6-15
Control Transfer 6-16
8088 Instruction Set Matrix 6-20
8088 Conditional Transfer Operations 6-22
Processor Control 6-23
8087 Coprocessor Instruction Set 6-24
Data Transfer 6-24
Comparison 6-25
Arithmetic 6-26
Transcendental 6-28
Constants 6-28
Processor Control 6-29

Notes:

8088 Register Model

```
Notes:
if d = 1 then "to"; if d = 0 then "from"
if w = 1 then word instruction; if w = 0 then byte instruction
if s:w = 01 then 16 bits of immediate data from the operand
if s:w = 11 then an immediate data byte is signed extended to form
             the 16-bit operand
if v = 0 the "count" = 1; if v = 1 the "count" is in (CL) or (CX)
x = don't care
z is used for string primitives for comparison with ZF FLAG
AL = 8-bit accumulator
AX = 16-bit accumulator
CX = Count register
DS = Data segment
ES = Extra segment
Above/below refers to unsigned value
Greater = more positive;
Less = less positive (more negative) signed values
```

AX:	AH	AL	Accumulator	General Register File
BX:	BH	BL	Base	
CX:	CH	CL	Count	
DX:	DH	DL	Data	
	SP		Stack Pointer	
	BP		Base Pointer	
	SI		Source Index	
	DI		Destination Index	

IP		Instruction Pointer
FLAGSH	FLAGSL	Status Flags

CS	Code Segment	Segment Register File
DS	Data Segment	
SS	Stack Segment	
ES	Extra Segment	

Instructions which reference the flag register file as a 16-bit object, use the symbol FLAGS to represent the file:

15								7							0
X	X	X	X	OF	DF	IF	TF	SF	ZF	X	AF	X	PF	X	CF

X = Don't Care

AF: Auxiliary Carry - BCD
CF: Carry Flag
PF: Parity Flag
SF: Sign Flag
ZF: Zero Flag

8080 Flags

DF: Direction Flag
IF: Interrupt Enable Flag
OF: Overflow Flag (CF + SF)
TF: Trap-Single Step Flag

8088 Flags

Operand Summary

reg Field Bit Assignments

16-Bit [w = 1]	8-Bit [w = 0]	Segment
000 AX	000 AL	00 ES
001 CX	001 CL	01 CS
010 DX	010 DL	10 SS
011 BX	011 BL	11 DS
100 SP	100 AH	
101 BP	101 CH	
110 SI	110 DH	
111 DI	111 BH	

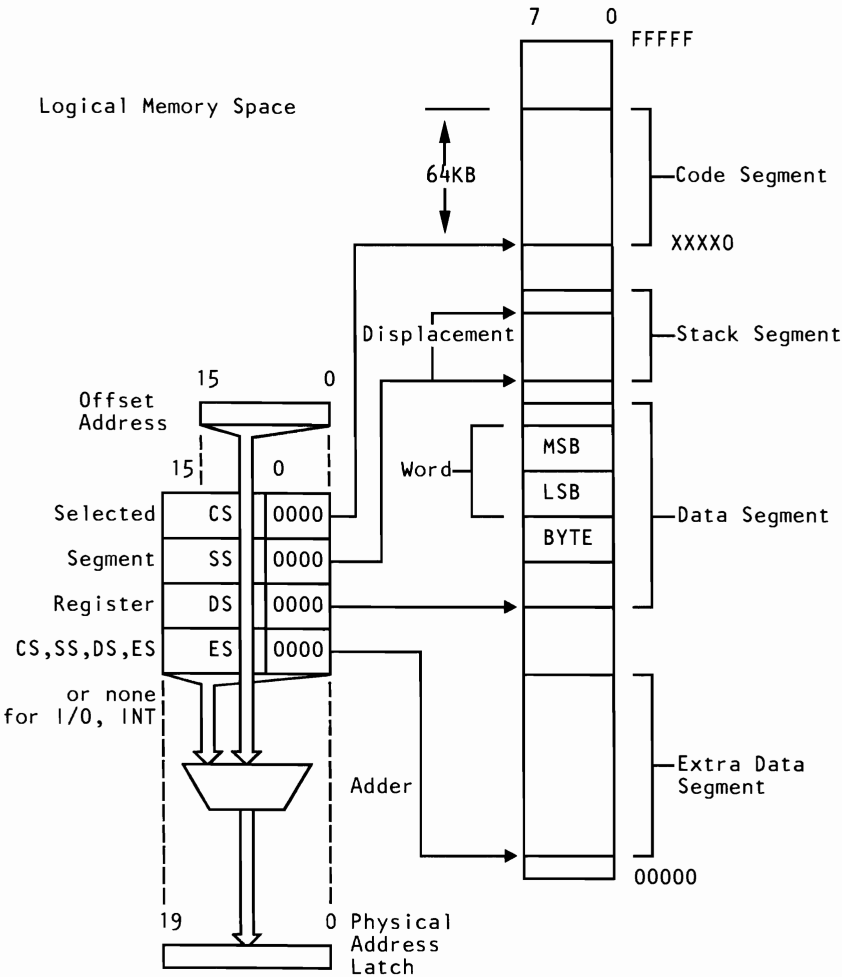
Second Instruction Byte Summary

mod	xxx	r/m

mod	Displacement
00	DISP = 0*, disp-low and disp-high are absent
01	DISP = disp-low sign-extended to 16-bits, disp-high is absent
10	DISP = disp-high: disp-low
11	r/m is treated as a "reg" field

DISP follows 2nd byte of instruction (before data if required)
*except if mod=00 and r/m-110 then EA=disp-high: disp-low.

Memory Segmentation Model



SECTION 6

Segment Override Prefix

001reg110

Use of Segment Override

Operand Register	Default	With Override Prefix
IP (Code Address)	CS	Never
SP (Stack Address)	SS	Never
BP (Stack Address or Stack Marker)	SS	BP + DS or ES, or CS
SI or DI (not including strings)	DS	ES, SS, or CS
SI (Implicit Source Address for strings)	DS	ES, SS, or CS
DI (Implicit Destination Address for strings)	ES	Never

8088 Instruction Set

Data Transfer

MOV = Move

Register/Memory to/from Register

100010dw	mod reg r/m

Immediate to Register/Memory

1100011w	mod 000 r/m	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	mod 0 reg r/m

Segment Register to Register/Memory

10001100	mod 0 reg r/m

PUSH = Push

Register/Memory

11111111	mod 110 r/m

Register

01010 reg

Segment Register

000 reg 110	

POP = Pop

Register/Memory

10001111	mod 000 r/m

Register

01011reg

Segment Register

000 reg 111

XCHG = Exchange

Register/Memory with Register

1000011w	mod reg r/m

Register with Accumulator

10010reg

IN = Input to AL/AX from

Fixed Port

1110010w	port

Variable Port

1110110w

OUT = Output from AL/AX to

Fixed Port

1110011w	port

Variable Port (DX)

1110110w

XLAT = Translate Byte to AL

11010111

LEA = Load EA to Register

10001101	mod reg r/m

LDS = Load Pointer to DS

11000101	mod reg r/m

LES = Load Pointer to ES

11000100	mod reg r/m

LAHF = Load AH with Flags

10011111

SAHF = Store AH with Flags

10011110

PUSHF = Push Flags

10011100

POPF = Pop Flags

10011101

Arithmetic

ADD = Add

Register/Memory with Register to Either

000000dw	mod reg r/m

Immediate to Register Memory

100000sw	mod 000 r/m	data	data if s:w = 01

Immediate to Accumulator

0000010w	data	data if w = 1

ADC = Add with Carry

Register/Memory with Register to Either

000100dw	mod reg r/m

Immediate to Register/Memory

100000sw	mod 010 r/m	data	data if s:w = 01

Immediate to Accumulator

0001010w	data	data if w = 1

INC = Increment

Register/Memory

1111111w	mod 000 r/m

Register

01000reg

AAA = ASCII Adjust for Add

00110111

DAA = Decimal Adjust for Add

00100111

SUB = Subtract

Register/Memory and Register to Either

001010dw	mod reg r/m

Immediate from Register/Memory

100000sw	mod 101 r/m	data	data if s:w = 01

Immediate from Accumulator

0010110w	data	data if w = 1

SBB = Subtract with Borrow

Register/Memory and Register to Either

000110dw	mod reg r/m

Immediate from Register/Memory

100000sw	mod 011 r/m	data	data if s:w = 01

Immediate to Accumulator

0001110w	data	data if w = 1

DEC = Decrement

Register/Memory

1111111w	mod 001 r/m

Register

01001reg

NEG = Change Sign

1111011w	mod 011 r/m

CMP = Compare

Register/Memory and Register

001110dw	mod reg r/m

Immediate with Register/Memory

100000sw	mod 111 r/m	data	data if s:w = 01

Immediate with Accumulator

0011110w	data	data if w = 1

AAS = ASCII Adjust for Subtract

00111111

DAS = Decimal Adjust for Subtract

00101111

MUL = Multiply (Unsigned)

1111011w	mod 100 r/m

IMUL = Integer Multiply (Signed)

1111011w	mod 101 r/m

AAM = ASCII Adjust for Multiply

11010100	00001010

DIV = Divide (Unsigned)

1111011w	mod 110 r/m

IDIV = Integer Divide (Signed)

1111011w	mod 111 r/m

AAD = ASCII Adjust for Divide

11010101	00001010

CBW = Convert Byte to Word

10011000

CWD = Convert Word to Double Word

10011001

Logic

Shift/Rotate Instructions

NOT = Invert Register/Memory

1111011w	mod 010 r/m

SHL/SAL = Shift Logical/Arithmetic Left

110100vw	mod 100 r/m

SHR = Shift Logical Right

110100vw	mod 101 r/m

SAR = Shift Arithmetic Right

110100vw	mod 111 r/m

ROL = Rotate Left

110100vw	mod 000 r/m

ROR = Rotate Right

110100vw	mod 001 r/m

RCL = Rotate through Carry Left

110100vw	mod 010 r/m

RCR = Rotate through Carry Right

110100vw	mod 011 r/m

AND = And

Register/Memory and Register to Either

001000dw	mod reg r/m

Immmediate to Register/Memory

1000000w	mod 100 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	mod 000 r/m	data	data if w = 1

Immediate Data and Accumulator

1010100w	data	data if w = 1

OR = Or

Register/Memory and Register to Either

000010dw	mod reg r/m

Immediate to Register/Memory

1000000w	mod 001 r/m	data	data if w = 1

Immediate to Accumulator

0000110w	data	data if w = 1

XOR = Exclusive OR

Register/Memory and Register to Either

001100dw	mod reg r/m

Immediate to Register/Memory

1000000w	mod 110 r/m	data	data if w = 1

Immediate to Accumulator

0011010w	data	data if w = 1

String Manipulation

REP = Repeat

1111001z

MOVS = Move String

1010010w

CMPS = Compare String

1010011w

SCAS = Scan String

1010111w

LODS = Load String

1010110w

STOS = Store String

1010101w

Control Transfer

CALL = Call

Direct within Segment

11101000	disp-low	disp-high

Indirect within Segment

11111111	mod 010 r/m

Direct Intersegment

10011010	offset-low	offset-high
	seg-low	seg-high

Indirect Intersegment

11111111	mod 011 r/m

JMP = Unconditional Jump

Direct within Segment-Short

11101011	disp

Indirect within Segment

11111111	mod 100 r/m

Direct Intersegment

11101010	offset-low	offset-high
	seg-low	seg-high

Indirect Intersegment

11111111	mod 101 r/m

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

11000010	data-low	data-high

JE/JZ = Jump on Equal/Zero

01110100	disp

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

01111100	disp

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

01111110	disp

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

01110010	disp

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

01110110	disp

JP/JPE = Jump on Parity/Parity Even

01111010	disp

JO = Jump on Overflow

01110000	disp

JS = Jump on Sign

01111000	disp

JNE/JNZ = Jump on Not Equal/Not Zero

01110101	disp

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

01111101	disp

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

01111111	disp

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

01110011	disp

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

01110111	disp

JNP/JPO = Jump on Not Parity/Parity Odd

01111011	disp

JNO = Jump on Not Overflow

01110001	disp

JNS = Jump on Not Sign

01111001	disp

LOOP = Loop CX Times

11100010	disp

LOOPZ/LOOPE = Loop while Zero/Equal

11100001	disp

LOOPNZ/LOOPNE = Loop while Not Zero/Not Equal

11100000	disp

JCXZ = Jump on CX Zero

11100011	disp

8088 Instruction Set Matrix

LO	0	1	2	3	4	5	6	7
HI 0	ADD b,b,r/m	ADD w,f,r/m	ADD b,t,r/m	ADD w,t,r/m	ADD b,ia	ADD w,ia	PUSH ES	POP ES
1	ADC b,f,r/m	ADC w,f,r/m	ADC b,t,r/m	ADC w,t,r/m	ADC b,i	ADC w,i	PUSH SS	POP SS
2	AND b,f,r/m	AND w,f,r/m	AND b,t,r/m	AND w,t,r/m	AND b,i	AND w,i	DEG =ES	DAA
3	XOR b,f,r/m	XOR w,f,r/m	XOR b,t,r/m	XOR w,t,r/m	XOR b,i	XOR w,i	SEG =S+	AAA
4	INC AX	INC CX	INC DX	INC BX	INC SP	INC BP	INC SI	INC DI
5	PUSH AX	PUSH CX	PUSH DX	PUSH BX	PUSH SP	PUSH BP	PUSH SI	PUSH DI
6								
7	JO	JNO	JB/ JNAE	JNB/ JAE	JE/ JZ	JNE/ JNZ	JBE/ JNA	JNBE/ JA
8	Immed b,r/m	Immed w,r/m	Immed b,r/m	Immed is,r/m	TEST b,r/m	TEST w,r/m	XCHG b,r/m	XCHG w,r/m
9	NOP	XCHG CX	XCHG DX	XCHG BX	XCHG SP	XCHG BP	XCHG SI	XCHG DI
A	MOV m AL	MOV m AL	MOV AL m	MOV AL m	MOVS b	MOVS w	CMPS b	CMPS w
B	MOV i AL	MOV i CL	MOV i DL	MOV i BL	MOV i AH	MOV i CH	MOV i DH	MOV i BH
C			RET (I+SP)	RET	LES	LDS	MOV b,i,r/m	MOV w,i,r/m
D	Shift b	Shift w	Shift b,v	Shift w,v	AAM	AAD		XLAT
E	LOOPNZ/ LOOPNE	LOOPZ/ POOPE	LOOP	JCXZ	IN b	IN w	OUT b	OUT w
F	LOCK		REP	REP z	HLT	CMC	Grp 1 b,r/m	Grp 1 w,r/m

b = byte operation
d = direct
f = from CPU reg
i = immediate
ia = immed. to accum.
id = direct
is = immed. byte, sign ext.
l = long ie. intersegment

m = memory
r/m = EA is second byte
si = short intersegment
t = to CPU reg
v = variable
w = word operation
z = zero
sr = segment register

LO	8	9	A	B	C	D	E	F
HI 0	OR b,f,r/m	w,f,r/m	OR b,t,r/m	OR w,t,r/m	OR b,i	OR w,i	PUSH CS	
1	SBB b,f,r/m	SBB w,f,r/m	SBB b,t,r/m	SBB w,t,r/m	SBB b,i	SBB w,i	PUSH DS	POP DS
2	SUB b,f,r/m	SUB w,f,r/m	SUB b,t,r/m	SUB w,t,r/m	SUB b,i	SUB w,i	SEG= CS	DAS
3	CMP b,f,r/m	CMP w,f,r/m	CMP b,t,r/m	CMP w,t,r/m	CMP b,i	CMP w,i	SEG= CS	AAS
4	DEC AX	DEC CX	DEC DX	DEC BX	DEC SP	DEC BP	DEC SI	DEC DI
5	POP AX	POP CX	POP DX	POP BX	POP SP	POP BP	POP SI	POP DI
6								
7	JS	JNS	JP/ JPE	JNP/ JPO	JL/ JNGE	JNL/ JGE	JLE/ JNG	JNLE/ JG
8	MOV b,f,r/m	MOV w,f,r/m	MOV b,t,r/m	MOV w,t,r/m	MOV sr,t,r/m	LEA	MOV sr,f,r/m	POP r/m
9	CBW	CWD CX	CALL l,d	WAIT BX	PUSHF SP	POPF BP	SAHF SI	LAHF DI
A	TEST b,i	TEST w,i	STOS b	STOS w	LODS b	LODS w	SCAS b	SCAS w
B	MOV i AX	MOV i CX	MOV i DX	MOV i BX	MOV i SP	MOV i BP	MOV i SI	MOV i DI
C			RET l,(I+SP)	RET l	INT Type 3	INT (Any)	INTO	IRET
D	ESC 0	ESC 1	ESC 2	ESC 3	ESC 4	ESC 5	ESC 6	ESC 7
E	CALL d	JMP d	JMP l,d	JMP si,d	IN v,b	IN v,w	OUT v,b	OUT v,w
F	CLC	STC	CLI	STI	CLD	STD	Grp 2 b,r/m	Grp 3 w,r/m

where:

mod r/m	000	001	010	011	100	101	110	111
Immed	ADD	OR	ADC	SBB	AND	SUB	XOR	CMP
Shift	ROL	ROR	RCL	RCR	SHL/SAL	SHR	--	SAR
Grp 1	TEST	--	NOT	NEG	MUL	IMUL	DIV	DIV
Grp 2	INC	DEC	CALL id	CALL l,id	JMP id	JMP l,id	PUSH	--

8088 Conditional Transfer Operations

Instruction	Condition	Interpretation
JE or JZ	ZF = 1	"equal" or "zero"
JL or JNGE	(SF xor OF) =1	"less" or "not greater or equal"
JLE or JNG	((SF xor OF) or ZF) = 1	"less or equal" or "not greater"
JB or JNAE or JC	CF = 1	"below" or "not above or equal"
JBE or JNA	(CF or ZF) = 1	"below or equal" or "not above"
JP or JPE	PF = 1	"parity" or "parity even"
JO	OF = 1	"overflow"
JS	SF = 1	"sign"
JNE or JNZ	ZF = 0	"not equal" or "not zero"
JNL or JGE	(SF xor OF) = 0	"not less" or "greater or equal"
JNLE or JG	((SF xor OF) or ZF) = 0	"not less or equal" or "greater"
JNB or JAE or JNC	CF = 0	"not below" or "above or equal"
JNBE or JA	(CF or ZF) = 0	"not below or equal" or "above"
JNP or JPO	PF = 0	"not parity" or "parity odd"
JNO	OF = 0	"not overflow"
JNS	SF = 0	"not sign"

"Above" and "below" refer to the relation between two unsigned values, while "greater" and "less" refer to the relation between two signed values.

INT = Interrupt

Type Specified

11001101	Type

Type 3

11001100

INTO = Interrupt on Overflow

11001110

IRET = Interrupt Return

11001111

Processor Control

CLC = Clear Carry

11111000

STC = Set Carry

11111001

CMC = Complement Carry

11110101

NOP = No Operation

10010000

CLD = Clear Direction

11111100

STD = Set Direction

11111101

CLI = Clear Interrupt

11111010

STI = Set Interrupt

11111011

HLT = Halt

11110100

WAIT = Wait

10011011

LOCK = Bus lock prefix

11110000

ESC = Escape (to 8087)

11011xxx	mod xxx r/m

8087 Coprocessor Instruction Set

The following is an instruction set summary for the 8087 coprocessor. In the following, the bit pattern for escape is 11011.

MF = Memory format
00 - 32-bit Real
01 - 32-bit Integer
10 - 64-bit Real
11 - 64-bit Integer

r/m	Operand Address
000	(BX) + (SI) + DISP
001	(BX) + (DI) + DISP
010	(BP) + (SI) + DISP
011	(BP) + (DI) + DISP
100	(SI) + DISP
101	(DI) + DISP
110	(BP) + DISP*
100	(BX) + DISP

DISP follows 2nd byte of instruction (before data if required)
*except if mod=00 and r/m-110 then EA=disp-high: disp-low.

Data Transfer

FLD = Load

Integer/Real Memory to ST(0)

escape MF 1	mod 000 r/m	disp-low	disp-high

Long Integer Memory to ST(0)

escape 111	mod 101 r/m	disp-low	disp-high

Temporary Real Memory to ST(0)

escape 011	mod 101 r/m	disp-low	disp-high

BCD Memory to ST(0)

escape 111	mod 100 r/m	disp-low	disp-high

ST(i) to ST(0)

escape 001	11000ST(i)

FST = Store

ST(0) to Integer/Real Memory

escape MF 1	mod 010 r/m	disp-low	disp-high

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	disp-low	disp-high

ST(0) to Long Integer Memory

escape 111	mod 111 r/m	disp-low	disp-high

ST(0) to Temporary Real Memory

escape 011	mod 111 r/m	disp-low	disp-high

ST(0) to BCD Memory

escape 111	mod 110 r/m	disp-low	disp-high

ST(0) to ST(i)

escape 101	11011 ST(i)

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

escape 001	11001 ST(i)

Comparison

FCOM = Compare

Integer/Real Memory to ST(0)

escape MF 0	mod 010 r/m	disp-low	disp-high

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	disp-low	disp-high

ST(i) to ST(0)

escape 000	11010 ST(i)

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

escape 110	11011001

FTST = Test ST(0)

escape 001	11100100

FXAM = Examine ST(0)

escape 001	11100101

Arithmetic

FADD = Addition

Integer/Real Memory with ST(0)

escape MF 0	mod 000 r/m	disp-low	disp-high

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	disp-low	disp-high

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	disp-low	disp-high

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	disp-low	disp-high

ST(i) and ST(0)

escape dP0	1111R r/m

FSQRT = Square Root of ST(0)

escape 001	11111010

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

escape 001	11111101

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

escape 001	11111000

FRNDINT = Round ST(0) to Integer

escape 001	11111100

FXTRACT = Extract Components of ST(0)

escape 001	11110100

FABS = Absolute Value of ST(0)

escape 001	11100001

FCHS = Change Sign of ST(0)

escape 001	11100000

Transcendental

FPTAN = Partial Tangent of ST(0)

escape 001	11110010

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

escape 001	11110011

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

escape 001	11110000

FYL2X = ST(1) x Log_2 [ST(0)]

escape 001	11110001

FYL2XP1 = ST(1) x Log_2 [ST(0) + 1]

escape 001	11111001

Constants

FLDZ = Load + 0.0 into ST(0)

escape 001	11101110

FLD1 = Load + 1.0 into ST(0)

escape 001	11101000

FLDP1 = Load π into ST(0)

escape 001	11101011

FLDL2T = Load Log_2 10 into ST(0)

escape 001	11101001

FLDLG2 = Load Log_{10} 2 into ST(0)

escape 001	11101100

FLDLN2 = Load Log_e 2 into ST(0)

escape 001	11101101

Processor Control

FINIT = Initialize NDP

escape 011	11100011

FENI = Enable Interrupts

escape 011	11100000

FDISI = Disable Interrupts

escape 011	11100001

FLDCW = Load Control Word

escape 001	mod101 r/m	disp-low	disp-high

FSTCW = Store Control Word

escape 001	mod111 r/m	disp-low	disp-high

FSTSW = Store Status Word

escape 101	mod111 r/m	disp-low	disp-high

FCLEX = Clear Exceptions

escape 011	11100010

FSTENV = Store Environment

escape 001	mod110 r/m	disp-low	disp-high

FLDENV = Load Environment

escape 001	mod100 r/m	disp-low	disp-high

FSAVE = Save State

escape 101	mod110 r/m	disp-low	disp-high

FRSTOR = Restore State

escape 101	mod100 r/m	disp-low	disp-high

FINCSTP = Increment Stack Pointer

escape 001	11110111

FDECSTP = Decrement Stack Pointer

escape 001	11110110

FFREE = Free ST(i)

escape 001	11000ST(i)

FNOP = No Operation

escape 001	11010000

FWAIT = CPU Wait for NDP

10011011

Notes:

ST(0) = Current Stack top

ST(i) = i^{th} register below Stack top

d = Destination

0—Destination is ST(0)
1—Destination is ST(i)

P = POP

0—No Pop
1—Pop ST(0)

R = Reverse

0—Destination (op) Source
1—Source (op) Destination

For **FSQRT:** $-0 \leq ST(0) \leq +\infty$

For **FSCALE:** $-2^{15} \leq ST(1) < + 2^{15}$ and ST(1) interger

For **F2XM1:** $0 \leq ST(0) \leq 2^{-1}$

For **FYL2X:** $0 < St(0) < \infty \; -\infty < ST(1) < +\infty$

For **FYL2XP1:** $0 < |ST(0)| < (2-\sqrt{2})/2 \; -\infty < ST(1) < \infty$

For **FPTAN:** $0 \leq ST(0) < \pi/4$

For **FPATAN:** $0 \leq ST(0) < ST(1) < +\infty$

Notes:

SECTION 7. CHARACTERS, KEYSTROKES, AND COLORS

Character Codes 7-3
Quick Reference 7-14

Notes:

Character Codes

Value		As Characters			As Text Attributes		
					Color/Graphics Monitor Adapter		IBM Monochrome Display Adapter
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground	
00	0	Blank (Null)	Ctrl 2		Black	Black	Non-Display
01	1	☺	Ctrl A		Black	Blue	Underline
02	2	☻	Ctrl B		Black	Green	Normal
03	3	♥	Ctrl C		Black	Cyan	Normal
04	4	♦	Ctrl D		Black	Red	Normal
05	5	♣	Ctrl E		Black	Magenta	Normal
06	6	♠	Ctrl F		Black	Brown	Normal
07	7	•	Ctrl G		Black	Light Grey	Normal
08	8	◘	Ctrl H, Backspace, Shift Backspace		Black	Dark Grey	Non-Display
09	9	○	Ctrl I		Black	Light Blue	High Intensity Underline
0A	10	◙	Ctrl J, Ctrl ↵		Black	Light Green	High Intensity
0B	11	♂	Ctrl K		Black	Light Cyan	High Intensity
0C	12	♀	Ctrl L		Black	Light Red	High Intensity
0D	13	♪	Ctrl M, ↵, Shift ↵		Black	Light Magenta	High Intensity
0E	14	♫	Ctrl N		Black	Yellow	High Intensity
0F	15	☼	Ctrl O		Black	White	High Intensity
10	16	►	Ctrl P		Blue	Black	Normal
11	17	◄	Ctrl Q		Blue	Blue	Underline
12	18	↕	Ctrl R		Blue	Green	Normal
13	19	‼	Ctrl S		Blue	Cyan	Normal
14	20	¶	Ctrl T		Blue	Red	Normal
15	21	§	Ctrl U		Blue	Magenta	Normal
16	22	▬	Ctrl V		Blue	Brown	Normal
17	23	↨	Ctrl W		Blue	Light Grey	Normal

Value		As Characters			As Text Attributes		
					Color/Graphics Monitor Adapter		IBM Monochrome Display Adapter
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground	
18	24	↑	Ctrl X		Blue	Dark Grey	High Intensity
19	25	↓	Ctrl Y		Blue	Light Blue	High Intensity Underline
1A	26	→	Ctrl Z		Blue	Light Green	High Intensity
1B	27	←	Ctrl [, Esc, Shift Esc, Crtl Esc		Blue	Light Cyan	High Intensity
1C	28	∟	Ctrl \		Blue	Light Red	High Intensity
1D	29	↔	Ctrl]		Blue	Light Magenta	High Intensity
1E	30	▲	Ctrl 6		Blue	Yellow	High Intensity
1F	31	▼	Ctrl —		Blue	White	High Intensity
20	32	Blank Space	Space Bar, Shift, Space, Ctrl Space, Alt Space		Green	Black	Normal
21	33	!	!	Shift	Green	Blue	Underline
22	34	"	"	Shift	Green	Green	Normal
23	35	#	#	Shift	Green	Cyan	Normal
24	36	$	$	Shift	Green	Red	Normal
25	37	%	%	Shift	Green	Magenta	Normal
26	38	&	&	Shift	Green	Brown	Normal
27	39	'	'		Green	Light Grey	Normal
28	40	(	(	Shift	Green	Dark Grey	High Intensity
29	41	)	)	Shift	Green	Light Blue	High Intensity Underline
2A	42	*	*	Note 1	Green	Light Green	High Intensity
2B	43	+	+	Shift	Green	Light Cyan	High Intensity
2C	44	,	,		Green	Light Red	High Intensity
2D	45	-	-		Green	Light Magenta	High Intensity
2E	46	.	.	Note 2	Green	Yellow	High Intensity

Value		As Characters			As Text Attributes		
					Color/Graphics Monitor Adapter		IBM Monochrome Display Adapter
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground	
2F	47	/	/		Green	White	High Intensity
30	48	0	0	Note 3	Cyan	Black	Normal
31	49	1	1	Note 3	Cyan	Blue	Underline
32	50	2	2	Note 3	Cyan	Green	Normal
33	51	3	3	Note 3	Cyan	Cyan	Normal
34	52	4	4	Note 3	Cyan	Red	Normal
35	53	5	5	Note 3	Cyan	Magenta	Normal
36	54	6	6	Note 3	Cyan	Brown	Normal
37	55	7	7	Note 3	Cyan	Light Grey	Normal
38	56	8	8	Note 3	Cyan	Dark Grey	High Intensity
39	57	9	9	Note 3	Cyan	Light Blue	High Intensity Underline
3A	58	:	:	Shift	Cyan	Light Green	High Intensity
3B	59	;	;		Cyan	Light Cyan	High Intensity
3C	60	<	<	Shift	Cyan	Light Red	High Intensity
3D	61	=	=		Cyan	Light Magenta	High Intensity
3E	62	>	>	Shift	Cyan	Yellow	High Intensity
3F	63	?	?	Shift	Cyan	White	High Intensity
40	64	@	@	Shift	Red	Black	Normal
41	65	A	A	Note 4	Red	Blue	Underline
42	66	B	B	Note 4	Red	Green	Normal
43	67	C	C	Note 4	Red	Cyan	Normal
44	68	D	D	Note 4	Red	Red	Normal
45	69	E	E	Note 4	Red	Magenta	Normal
46	70	F	F	Note 4	Red	Brown	Normal
47	71	G	G	Note 4	Red	Light Grey	Normal
48	72	H	H	Note 4	Red	Dark Grey	High Intensity
49	73	I	I	Note 4	Red	Light Blue	High Intensity Underline
4A	74	J	J	Note 4	Red	Light Green	High Intensity

Value		As Characters			As Text Attributes		
					Color/Graphics Monitor Adapter		IBM Monochrome Display Adapter
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground	
4B	75	K	K	Note 4	Red	Light Cyan	High Intensity
4C	76	L	L	Note 4	Red	Light Red	High Intensity
4D	77	M	M	Note 4	Red	Light Magenta	High Intensity
4E	78	N	N	Note 4	Red	Yellow	High Intensity
4F	79	O	O	Note 4	Red	White	High Intensity
50	80	P	P	Note 4	Magenta	Black	Normal
51	81	Q	Q	Note 4	Magenta	Blue	Underline
52	82	R	R	Note 4	Magenta	Green	Normal
53	83	S	S	Note 4	Magenta	Cyan	Normal
54	84	T	T	Note 4	Magenta	Red	Normal
55	85	U	U	Note 4	Magenta	Magenta	Normal
56	86	V	V	Note 4	Magenta	Brown	Normal
57	87	W	W	Note 4	Magenta	Light Grey	Normal
58	88	X	X	Note 4	Magenta	Dark Grey	High Intensity
59	89	Y	Y	Note 4	Magenta	Light Blue	High Intensity Underline
5A	90	Z	Z	Note 4	Magenta	Light Green	High Intensity
5B	91	[	[		Magenta	Light Cyan	High Intensity
5C	92	\	\		Magenta	Light Red	High Intensity
5D	93	]	]		Magenta	Light Magenta	High Intensity
5E	94	^	^	Shift	Magenta	Yellow	High Intensity
5F	95	—	—	Shift	Magenta	White	High Intensity
60	96	'	'		Brown	Black	Normal
61	97	a	a	Note 5	Brown	Blue	Underline
62	98	b	b	Note 5	Brown	Green	Normal
63	99	c	c	Note 5	Brown	Cyan	Normal
64	100	d	d	Note 5	Brown	Red	Normal
65	101	e	e	Note 5	Brown	Magenta	Normal
66	102	f	f	Note 5	Brown	Brown	Normal

Value		As Characters			As Text Attributes		
					Color/Graphics Monitor Adapter		IBM Monochrome Display Adapter
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground	
67	103	g	g	Note 5	Brown	Light Grey	Normal
68	104	h	h	Note 5	Brown	Dark Grey	High Intensity
69	105	i	i	Note 5	Brown	Light Blue	High Intensity Underline
6A	106	j	j	Note 5	Brown	Light Green	High Intensity
6B	107	k	k	Note 5	Brown	Light Cyan	High Intensity
6C	108	l	l	Note 5	Brown	Light Red	High Intensity
6D	109	m	m	Note 5	Brown	Light Magenta	High Intensity
6E	110	n	n	Note 5	Brown	Yellow	High Intensity
6F	111	o	o	Note 5	Brown	White	High Intensity
70	112	p	p	Note 5	Light Grey	Black	Reverse Video
71	113	q	q	Note 5	Light Grey	Blue	Underline
72	114	r	r	Note 5	Light Grey	Green	Normal
73	115	s	s	Note 5	Light Grey	Cyan	Normal
74	116	t	t	Note 5	Light Grey	Red	Normal
75	117	u	u	Note 5	Light Grey	Magenta	Normal
76	118	v	v	Note 5	Light Grey	Brown	Normal
77	119	w	w	Note 5	Light Grey	Light Grey	Normal
78	120	x	x	Note 5	Light Grey	Dark Grey	Reverse Video
79	121	y	y	Note 5	Light Grey	Light Blue	High Intensity Underline
7A	122	z	z	Note 5	Light Grey	Light Green	High Intensity
7B	123	{	{	Shift	Light Grey	Light Cyan	High Intensity
7C	124	¦	¦	Shift	Light Grey	Light Red	High Intensity
7D	125	}	}	Shift	Light Grey	Light Magenta	High Intensity
7E	126	~	~	Shift	Light Grey	Yellow	High Intensity
7F	127	△	Ctrl ←		Light Grey	White	High Intensity

Value		As Characters			As Text Attributes		
					Color/Graphics Monitor Adapter		IBM Monochrome Display Adapter
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground	
* * * * 80 to FF Hex are Flashing in both Color & IBM Monochrome * * * *							
80	128	Ç	Alt 128	Note 6	Black	Black	Non-Display
81	129	ü	Alt 129	Note 6	Black	Blue	Underline
82	130	é	Alt 130	Note 6	Black	Green	Normal
83	131	â	Alt 131	Note 6	Black	Cyan	Normal
84	132	ä	Alt 132	Note 6	Black	Red	Normal
85	133	à	Alt 133	Note 6	Black	Magenta	Normal
86	134	å	Alt 134	Note 6	Black	Brown	Normal
87	135	ç	Alt 135	Note 6	Black	Light Grey	Normal
88	136	ê	Alt 136	Note 6	Black	Dark Grey	Non-Display
89	137	ë	Alt 137	Note 6	Black	Light Blue	High Intensity Underline
8A	138	è	Alt 138	Note 6	Black	Light Green	High Intensity
8B	139	ï	Alt 139	Note 6	Black	Light Cyan	High Intensity
8C	140	î	Alt 140	Note 6	Black	Light Red	High Intensity
8D	141	ì	Alt 141	Note 6	Black	Light Magenta	High Intensity
8E	142	Ä	Alt 142	Note 6	Black	Yellow	High Intensity
8F	143	Å	Alt 143	Note 6	Black	White	High Intensity
90	144	É	Alt 144	Note 6	Blue	Black	Normal
91	145	æ	Alt 145	Note 6	Blue	Blue	Underline
92	146	Æ	Alt 146	Note 6	Blue	Green	Normal
93	147	ô	Alt 147	Note 6	Blue	Cyan	Normal
94	148	ö	Alt 148	Note 6	Blue	Red	Normal
95	149	ò	Alt 149	Note 6	Blue	Magenta	Normal
96	150	û	Alt 150	Note 6	Blue	Brown	Normal
97	151	ù	Alt 151	Note 6	Blue	Light Grey	Normal
98	152	ÿ	Alt 152	Note 6	Blue	Dark Grey	High Intensity
99	153	Ö	Alt 153	Note 6	Blue	Light Blue	High Intensity Underline
9A	154	Ü	Alt 154	Note 6	Blue	Light Green	High Intensity

Value		As Characters			As Text Attributes		
					Color/Graphics Monitor Adapter		IBM Monochrome Display Adapter
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground	
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	ƒ	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	Brown	Black	Normal
E1	225	β	Alt 225	Note 6	Brown	Blue	Underline
E2	226	Γ	Alt 226	Note 6	Brown	Green	Normal
E3	227	π	Alt 227	Note 6	Brown	Cyan	Normal
E4	228	Σ	Alt 228	Note 6	Brown	Red	Normal
E5	229	σ	Alt 229	Note 6	Brown	Magenta	Normal
E6	230	μ	Alt 230	Note 6	Brown	Brown	Normal
E7	231	τ	Alt 231	Note 6	Brown	Light Grey	Normal
E8	232	Φ	Alt 232	Note 6	Brown	Dark Grey	High Intensity
E9	233	θ	Alt 233	Note 6	Brown	Light Blue	High Intensity Underline
EA	234	Ω	Alt 234	Note 6	Brown	Light Green	High Intensity
EB	235	δ	Alt 235	Note 6	Brown	Light Cyan	High Intensity

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

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 001-255 may be entered in this fashion (with Caps Lock activated, character codes 97-122 will display uppercase).

Quick Reference

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

DECIMAL VALUE	➡	128	144	160	176	192	208	224	240
⬇	HEXA-DECIMAL VALUE	8	9	A	B	C	D	E	F
0	0	Ç	É	á	░	└	╨	∝	≡
1	1	ü	æ	í	▒	┴	╤	β	±
2	2	é	Æ	ó	▓	┬	╥	Γ	≥
3	3	â	ô	ú	│	├	╙	π	≤
4	4	ä	ö	ñ	┤	─	╘	Σ	⌠
5	5	à	ò	Ñ	╡	┼	╒	σ	⌡
6	6	å	û	ª	╢	╞	╓	μ	÷
7	7	ç	ù	º	╖	╟	╫	ϒ	≈
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:

Glossary

This glossary includes terms and definitions from the *IBM Vocabulary for Data Processing, Telecommunications, and Office Systems,* GC20-1699.

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 Interchange (ASCII). The standard code, using a coded character set consisting of 7-bit coded characters (8 bits including parity check), used for information exchange between data processing systems, data communication systems, and associated equipment. The ASCII set consists of control characters and graphic characters.

ampere (A). The basic unit of electric current.

A/N. Alphanumeric

analog. (1) Pertaining to data in the form of continuously variable physical quantities. (2) Contrast with digital.

AND. A logic operator having the property that if P is a statement, Q is a statement, R is a statement,..., then the AND of P, Q, R,...is true if all statements are true, false if any statement is false.

AND gate. A logic gate in which the output is 1 only if all inputs are 1.

AND operation. The boolean operation whose result has the boolean value 1, if and only if, each operand has the boolean value 1. Synonymous with conjunction.

APA. All points addressable.

ASCII. American National Standard Code for Information Interchange.

assemble. To translate a program expressed in an assembler language into a computer language.

assembler. A computer program used to assemble.

assembler language. A computer-oriented language whose instructions are usually in one-to-one correspondence with computer instructions.

asynchronous transmission. (1) Transmission in which the time of occurrence of the start of each character, or block of characters, is arbitrary; once started, the time of occurrence of each signal representing a bit within a character, or block, has the same relationship to significant instants of a fixed time frame. (2) Transmission in which each information character is individually transmitted (usually timed by the use of start elements and stop elements).

audio frequencies. Frequencies that can be heard by the human ear (approximately 15 hertz to 20,000 hertz).

auxiliary storage. (1) A storage device that is not main storage. (2) Data storage other than main storage; for example, storage on magnetic disk. (3) Contrast with main storage.

BASIC. Beginner's all-purpose symbolic instruction code.

basic input/output system (BIOS). The feature of the IBM Personal Computer that provides the level control of the major I/O devices, and relieves the programmer from concern about hardware device characteristics.

baud. (1) A unit of signaling speed equal to the number of discrete conditions or signal events per second. For example, one baud equals one bit per second in a train of binary signals, one-half dot cycle per second in Morse code, and one 3-bit value per second in a train of signals each of which can assume one of eight different states. (2) In asynchronous transmission, the unit of modulation rate corresponding to one unit of interval per second; that is, if the duration of the unit interval is 20 milliseconds, the modulation rate is 50 baud.

BCC. Block-check character.

beginner's all-purpose symbolic instruction code (BASIC). A programming language with a small repertoire of commands and a simple syntax, primarily designed for numeric applications.

binary. (1) Pertaining to a selection, choice, or condition that has two possible values or states. (2) Pertaining to a fixed radix numeration system having a radix of 2.

binary digit. (1) In binary notation, either of the characters 0 or 1. (2) Synonymous with bit.

binary notation. Any notation that uses two different characters, usually the binary digits 0 and 1.

binary synchronous communications (BSC). A uniform procedure, using a standardized set of control characters and control character sequences for synchronous transmission of binary–coded data between stations.

BIOS. Basic input/output system.

bit. Synonym for binary digit

bits per second (bps). A unit of measurement representing the number of discrete binary digits transmitted by a device in one second.

block. (1) A string of records, a string of words, or a character string formed for technical or logic reasons to be treated as an entity. (2) A set of things, such as words, characters, or digits, treated as a unit.

block-check character (BCC). In cyclic redundancy checking, a character that is transmitted by the sender after each message block and is compared with a block-check character computed by the receiver to determine if the transmission was successful.

boolean operation. (1) Any operation in which each of the operands and the result take one of two values. (2) An operation that follows the rules of boolean algebra.

bootstrap. A technique or device designed to bring itself into a desired state by means of its own action; for example, a machine routine whose first few instructions are sufficient to bring the rest of itself into the computer from an input device.

bps. Bits per second.

BSC. Binary synchronous communications.

buffer. (1) An area of storage that is temporarily reserved for use in performing an input/output operation, into which data is read or from which data is written. Synonymous with I/O area. (2) A portion of storage for temporarily holding input or output data.

bus. One or more conductors used for transmitting signals or power.

byte. (1) A sequence of eight adjacent binary digits that are operated upon as a unit. (2) A binary character operated upon as a unit. (3) The representation of a character.

C. Celsius.

capacitor. An electronic circuit component that stores an electric charge.

Cartesian coordinates. A system of coordinates for locating a point on a plane by its distance from each of two intersecting lines, or in space by its distance from each of three mutually perpendicular planes.

CAS. Column address strobe.

cathode ray tube (CRT). A vacuum tube in which a stream of electrons is projected onto a fluorescent screen producing a luminous spot. The location of the spot can be controlled.

cathode ray tube display (CRT display). (1) A CRT used for displaying data. For example, the electron beam can be controlled to form alphanumeric data by use of a dot matrix. (2) Synonymous with monitor.

CCITT. International Telegraph and Telephone Consultative Committee.

Celsius (C). A temperature scale. Contrast with Fahrenheit (F).

central processing unit (CPU). Term for processing unit.

channel. A path along which signals can be sent; for example, data channel, output channel.

character generator. (1) In computer graphics, a functional unit that converts the coded representation of a graphic character into the shape of the character for display. (2) In word processing, the means within equipment for generating visual characters or symbols from coded data.

character set. (1) A finite set of different characters upon which agreement has been reached and that is considered complete for some purpose. (2) A set of unique representations called characters. (3) A defined collection of characters.

characters per second (cps). A standard unit of measurement for the speed at which a printer prints.

check key. A group of characters, derived from and appended to a data item, that can be used to detect errors in the data item during processing.

clipping. In computer graphics, removing parts of a display image that lie outside a window.

closed circuit. A continuous unbroken circuit; that is, one in which current can flow. Contrast with open circuit.

CMOS. Complementary metal oxide semiconductor.

code. (1) A set of unambiguous rules specifying the manner in which data may be represented in a discrete form. Synonymous with coding scheme. (2) A set of items, such as abbreviations, representing the members of another set. (3) To represent data or a computer program in a symbolic form that can be accepted by a data processor. (4) Loosely, one or more computer programs, or part of a computer program.

coding scheme. Synonym for code.

collector. An element in a transistor toward which current flows.

color cone. An arrangement of the visible colors on the surface of a double-ended cone where lightness varies along the axis of the cone, and hue varies around the circumference. Lightness includes both the intensity and saturation of color.

column address strobe (CAS). A signal that latches the column addresses in a memory chip.

compile. (1) To translate a computer program expressed in a problem-oriented language into a computer-oriented language. (2) To prepare a machine-language program from a computer program written in another programming language by making use of the overall logic structure of the program, or generating more than one computer instruction for each symbolic statement, or both, as well as performing the function of an assembler.

complement. A number that can be derived from a specified number by subtracting it from a second specified number.

complementary metal oxide semiconductor (CMOS). A logic circuit family that uses very little power. It works with a wide range of power supply voltages.

computer. A functional unit that can perform substantial computation, including numerous arithmetic operations or logic operations, without human intervention during a run.

computer instruction code. A code used to represent the instructions in an instruction set. Synonymous with machine code.

computer program. A sequence of instructions suitable for processing by a computer.

computer word. A word stored in one computer location and capable of being treated as a unit.

configuration. (1) The arrangement of a computer system or network as defined by the nature, number, and the chief characteristics of its functional units. More specifically, the term configuration may refer to a hardware configuration or a software configuration. (2) The devices and programs that make up a system, subsystem, or network.

conjunction. Synonym for AND operation.

contiguous. Touching or joining at the edge or boundary; adjacent.

control character. A character whose occurrence in a particular context initiates, modifies, or stops a control operation.

control operation. An action that affects the recording, processing, transmission, or interpretation of data; for example, starting or stopping a process, carriage return, font change, rewind, and end of transmission.

control storage. A portion of storage that contains microcode.

coordinate space. In computer graphics, a system of Cartesian coordinates in which an object is defined.

cps. Characters per second.

CPU. Central processing unit.

CRC. Cyclic redundancy check.

CRT. Cathode ray tube.

CRT display. Cathode ray tube display.

CTS. Clear to send. Associated with modem control.

cursor. (1) In computer graphics, a movable marker that is used to indicate position on a display. (2) A displayed symbol that acts as a marker to help the user locate a point in text, in a system command, or in storage. (3) A movable spot of light on the screen of a display device, usually indicating where the next character is to be entered, replaced, or deleted.

cyclic redundancy check (CRC). (1) A redundancy check in which the check key is generated by a cyclic algorithm. (2) A system of error checking performed at both the sending and receiving station after a block-check character has been accumulated.

cylinder. (1) The set of all tracks with the same nominal distance from the axis about which the disk rotates. (2) The tracks of a disk storage device that can be accessed without repositioning the access mechanism.

daisy-chained cable. A type of cable that has two or more connectors attached in series.

data. (1) A representation of facts, concepts, or instructions in a formalized manner suitable for communication, interpretation, or processing by human or automatic means. (2) Any representations, such as characters or analog quantities, to which meaning is, or might be assigned.

data base. A collection of data that can be immediately accessed and operated upon by a data processing system for a specific purpose.

data processing system. A system that performs input, processing, storage, output, and control functions to accomplish a sequence of operations on data.

data transmission. Synonym for transmission.

dB. Decibel.

dBa. Adjusted decibels.

dc. Direct current.

debounce. (1) An electronic means of overcoming the make/break bounce of switches to obtain one smooth change of signal level. (2) The elimination of undesired signal variations caused by mechanically generated signals from contacts.

decibel. (1) A unit that expresses the ratio of two power levels on a logarithmic scale. (2) A unit for measuring relative power.

decoupling capacitor. A capacitor that provides a low impedance path to ground to prevent common coupling between circuits.

Deutsche Industrie Norm (DIN). (1) German Industrial Norm. (2) The committee that sets German dimension standards.

digit. (1) A graphic character that represents an integer; for example, one of the characters 0 to 9. (2) A symbol that represents one of the non-negative integers smaller than the radix. For example, in decimal notation, a digit is one of the characters 0 to 9.

digital. (1) Pertaining to data in the form of digits. (2) Contrast with analog.

DIN. Deutsche Industrie Norm.

DIN connector. One of the connectors specified by the DIN committee.

DIP. Dual in-line package.

DIP switch. One of a set of small switches mounted in a dual in-line package.

direct current (dc). A current that always flows in one direction.

direct memory access (DMA). A method of transferring data between main storage and I/O devices that does not require processor intervention.

disable. To stop the operation of a circuit or device.

disabled. Pertaining to a state of a processing unit that prevents the occurrence of certain types of interruptions. Synonymous with masked.

disk. Loosely, a magnetic disk.

diskette. A thin, flexible magnetic disk and a semirigid protective jacket, in which the disk is permanently enclosed. Synonymous with flexible disk.

diskette drive. A device for storing data on and retrieving data from a diskette.

display. (1) A visual presentation of data. (2) A device for visual presentation of information on any temporary character imaging device. (3) To present data visually. (4) See cathode ray tube display.

display attribute. In computer graphics, a particular property that is assigned to all or part of a display; for example, low intensity, green color, blinking status.

display element. In computer graphics, a basic graphic element that can be used to construct a display image; for example, a dot, a line segment, a character.

display group. In computer graphics, a collection of display elements that can be manipulated as a unit and that can be further combined to form larger groups.

display image. In computer graphics, a collection of display elements or display groups that are represented together at any one time in a display space.

display space. In computer graphics, that portion of a display surface available for a display image. The display space may be all or part of a display surface.

display surface. In computer graphics, that medium on which display images may appear; for example, the entire screen of a cathode ray tube.

DMA. Direct memory access.

dot matrix. (1) In computer graphics, a two-dimensional pattern of dots used for constructing a display image. This type of matrix can be used to represent characters by dots. (2) In word processing, a pattern of dots used to form characters. This term normally refers to a small section of a set of addressable points; for example, a representation of characters by dots.

dot printer. Synonym for matrix printer.

dot-matrix character generator. In computer graphics, a character generator that generates character images composed of dots.

drawing primitive. A group of commands that draw defined geometric shapes.

DSR. Data set ready. Associated with modem control.

DTR. In the IBM Personal Computer, data terminal ready. Associated with modem control.

dual in-line package (DIP). A widely used container for an integrated circuit. DIPs have pins in two parallel rows. The pins are spaced 1/10 inch apart. See also DIP switch.

duplex. (1) In data communication, pertaining to a simultaneous two-way independent transmission in both directions.
(2) Contrast with half-duplex.

duty cycle. In the operation of a device, the ratio of on time to idle time. Duty cycle is expressed as a decimal or percentage.

dynamic memory. RAM using transistors and capacitors as the memory elements. This memory requires a refresh (recharge) cycle every few milliseconds. Contrast with static memory.

EBCDIC. Extended binary-coded decimal interchange code.

ECC. Error checking and correction.

edge connector. A terminal block with a number of contacts attached to the edge of a printed-circuit board to facilitate plugging into a foundation circuit.

EIA. Electronic Industries Association.

electromagnet. Any device that exhibits magnetism only while an electric current flows through it.

enable. To initiate the operation of a circuit or device.

end of block (EOB). A code that marks the end of a block of data.

end of file (EOF). An internal label, immediately following the last record of a file, signaling the end of that file. It may include control totals for comparison with counts accumulated during processing.

end-of-text (ETX). A transmission control character used to terminate text.

end-of-transmission (EOT). A transmission control character used to indicate the conclusion of a transmission, which may have included one or more texts and any associated message headings.

end-of-transmission-block (ETB). A transmission control character used to indicate the end of a transmission block of data when data is divided into such blocks for transmission purposes.

EOB. End of block.

EOF. End of file.

EOT. End-of-transmission.

EPROM. Erasable programmable read-only memory.

erasable programmable read-only memory (EPROM). A PROM in which the user can erase old information and enter new information.

error checking and correction (ECC). The detection and correction of all single-bit errors, plus the detection of double-bit and some multiple-bit errors.

ESC. The escape character.

escape character (ESC). A code extension character used, in some cases, with one or more succeeding characters to indicate by some convention or agreement that the coded representations following the character or the group of characters are to be interpreted according to a different code or according to a different coded character set.

ETB. End-of-transmission-block.

ETX. End-of-text.

extended binary-coded decimal interchange code (EBCDIC). A set of 256 characters, each represented by eight bits.

F. Fahrenheit.

Fahrenheit (F). A temperature scale. Contrast with Celsius (C).

falling edge. Synonym for negative-going edge.

FCC. Federal Communications Commission.

fetch. To locate and load a quantity of data from storage.

FF. The form feed character.

field. (1) In a record, a specified area used for a particular category of data. (2) In a data base, the smallest unit of data that can be referred to.

field-programmable logic sequencer (FPLS). An integrated circuit containing a programmable, read-only memory that responds to external inputs and feedback of its own outputs.

FIFO (first-in-first out). A queuing technique in which the next item to be retrieved is the item that has been in the queue for the longest time.

fixed disk drive. In the IBM Personal Computer, a unit consisting of nonremovable magnetic disks, and a device for storing data on and retrieving data from the disks.

flag. (1) Any of various types of indicators used for identification. (2) A character that signals the occurrence of some condition, such as the end of a word. (3) Deprecated term for mark.

flexible disk. Synonym for diskette.

flip-flop. A circuit or device containing active elements, capable of assuming either one of two stable states at a given time.

font. A family or assortment of characters of a given size and style; for example, 10 point Press Roman medium.

foreground. (1) In multiprogramming, the environment in which high-priority programs are executed. (2) On a color display screen, the characters as opposed to the background.

form feed. (1) Paper movement used to bring an assigned part of a form to the printing position. (2) In word processing, a function that advances the typing position to the same character position on a predetermined line of the next form or page.

form feed character. A control character that causes the print or display position to move to the next predetermined first line on the next form, the next page, or the equivalent.

format. The arrangement or layout of data on a data medium.

FPLS. Field-programmable logic sequencer.

frame. (1) In SDLC, the vehicle for every command, every response, and all information that is transmitted using SDLC procedures. Each frame begins and ends with a flag. (2) In data transmission, the sequence of contiguous bits bracketed by and including beginning and ending flag sequences.

g. Gram.

G. (1) Prefix giga; 1,000,000,000. (2) When referring to computer storage capacity, 1,073,741,824. (1,073,741,824 = 2 to the 30th power.)

gate. (1) A combinational logic circuit having one output channel and one or more input channels, such that the output channel state is completely determined by the input channel states. (2) A signal that enables the passage of other signals through a circuit.

Gb. 1,073,741,824 bytes.

general-purpose register. A register, usually explicitly addressable within a set of registers, that can be used for different purposes; for example, as an accumulator, as an index register, or as a special handler of data.

giga (G). Prefix 1,000,000,000.

gram (g). A unit of weight (equivalent to 0.035 ounces).

graphic. A symbol produced by a process such as handwriting, drawing, or printing.

graphic character. A character, other than a control character, that is normally represented by a graphic.

half-duplex. (1) In data communication, pertaining to an alternate, one way at a time, independent transmission. (2) Contrast with duplex.

hardware. (1) Physical equipment used in data processing, as opposed to programs, procedures, rules, and associated documentation. (2) Contrast with software.

head. A device that reads, writes, or erases data on a storage medium; for example, a small electromagnet used to read, write, or erase data on a magnetic disk.

hertz (Hz). A unit of frequency equal to one cycle per second.

hex. Common abbreviation for hexadecimal. Also, hexidecimal can be noted as X'   '.

hexadecimal. (1) Pertaining to a selection, choice, or condition that has 16 possible different values or states. These values or states are usually symbolized by the ten digits 0 through 9 and the six letters A through F. (2) Pertaining to a fixed radix numeration system having a radix of 16.

high impedance state. A state in which the output of a device is effectively isolated from the circuit.

highlighting. In computer graphics, emphasizing a given display group by changing its attributes relative to other display groups in the same display field.

high-order position. The leftmost position in a string of characters. See also most-significant digit.

hither plane. In computer graphics, a plane that is perpendicular to the line joining the viewing reference point and the view point and that lies between these two points. Any part of an object between the hither plane and the view point is not seen. See also yon plane.

housekeeping. Operations or routines that do not contribute directly to the solution of the problem but do contribute directly to the operation of the computer.

Hz. Hertz

image. A fully processed unit of operational data that is ready to be transmitted to a remote unit; when loaded into control storage in the remote unit, the image determines the operations of the unit.

immediate instruction. An instruction that contains within itself an operand for the operation specified, rather than an address of the operand.

index register. A register whose contents may be used to modify an operand address during the execution of computer instructions.

indicator. (1) A device that may be set into a prescribed state, usually according to the result of a previous process or on the occurrence of a specified condition in the equipment, and that usually gives a visual or other indication of the existence of the prescribed state, and that may in some cases be used to determine the selection among alternative processes; for example, an overflow indicator. (2) An item of data that may be interrogated to determine whether a particular condition has been satisfied in the execution of a computer program; for example, a switch indicator, an overflow indicator.

inhibited. (1) Pertaining to a state of a processing unit in which certain types of interruptions are not allowed to occur.
(2) Pertaining to the state in which a transmission control unit or an audio response unit cannot accept incoming calls on a line.

initialize. To set counters, switches, addresses, or contents of storage to 0 or other starting values at the beginning of, or at prescribed points in, the operation of a computer routine.

input/output (I/O). (1) Pertaining to a device or to a channel that may be involved in an input process, and, at a different time, in an output process. In the English language, "input/output" may be used in place of such terms as "input/output data," "input/output signal," and "input/output terminals," when such usage is clear in a given context. (2) Pertaining to a device whose parts can be performing an input process and an output process at the same time. (3) Pertaining to either input or output, or both.

instruction. In a programming language, a meaningful expression that specifies one operation and identifies its operands, if any.

instruction set. The set of instructions of a computer, of a programming language, or of the programming languages in a programming system.

intensity. In computer graphics, the amount of light emitted at a display point

interface. A device that alters or converts actual electrical signals between distinct devices, programs, or systems.

interleave. To arrange parts of one sequence of things or events so that they alternate with parts of one or more other sequences of the same nature and so that each sequence retains its identity.

interrupt. (1) A suspension of a process, such as the execution of a computer program, caused by an event external to that process, and performed in such a way that the process can be resumed. (2) In a data transmission, to take an action at a receiving station that causes the transmitting station to terminate a transmission. (3) Synonymous with interruption.

I/O. Input/output.

I/O area. Synonym for buffer.

irrecoverable error. An error that makes recovery impossible without the use of recovery techniques external to the computer program or run.

joystick. In computer graphics, a lever that can pivot in all directions and that is used as a locator device.

k. Prefix kilo; 1000.

K. When referring to storage capacity, 1024. (1024 = 2 to the 10th power.)

KB. 1024 bytes.

key lock. A device that deactivates the keyboard and locks the cover on for security.

kg. Kilogram; 1000 grams.

kHz. Kilohertz; 1000 hertz.

kilo (k). Prefix 1000

kilogram (kg). 1000 grams.

kilohertz (kHz). 1000 hertz

latch. (1) A simple logic-circuit storage element. (2) A feedback loop in sequential digital circuits used to maintain a state.

least-significant digit. The rightmost digit. See also low-order position.

LED. Light-emitting diode.

light-emitting diode (LED). A semiconductor device that gives off visible or infrared light when activated.

load. In programming, to enter data into storage or working registers.

look-up table (LUT). (1) A technique for mapping one set of values into a larger set of values. (2) In computer graphics, a table that assigns a color value (red, green, blue intensities) to a color index.

low power Schottky TTL. A version (LS series) of TTL giving a good compromise between low power and high speed. See also transistor-transistor logic and Schottky TTL.

low-order position. The rightmost position in a string of characters. See also least-significant digit.

luminance. The luminous intensity per unit projected area of a given surface viewed from a given direction.

GLOSSARY

LUT. Look-up table.

m. (1) Prefix milli; 0.001. (2) Meter.

M. (1) Prefix mega; 1,000,000. (2) When referring to computer storage capacity, 1,048,576. (1,048,576 = 2 to the 20th power.)

mA. Milliampere; 0.001 ampere.

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

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

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

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

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

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

masked. Synonym for disabled.

matrix. (1) A rectangular array of elements, arranged in rows and columns, that may be manipulated according to the rules of matrix algebra. (2) In computers, a logic network in the form of an array of input leads and output leads with logic elements connected at some of their intersections.

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

MB. 1,048,576 bytes.

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

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

memory. Term for main storage.

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

MFM. Modified frequency modulation.

MHz. Megahertz; 1,000,000 hertz.

micro (μ). Prefix 0.000,001.

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

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

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

microsecond (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.

modeling transformation. Operations on the coordinates of an object (usually matrix multiplications) that cause the object to be rotated about any axis, translated (moved without rotating), and/or scaled (changed in size along any or all dimensions). See also viewing transformation.

modem (modulator-demodulator). A device that converts serial (bit by bit) digital signals from a business machine (or data communication equipment) to analog signals that are suitable for transmission in a telephone network. The inverse function is also performed by the modem on reception of analog signals.

modified frequency modulation (MFM). The process of varying the amplitude and frequency of the 'write' signal. MFM pertains to the number of bytes of storage that can be stored on the recording media. The number of bytes is twice the number contained in the same unit area of recording media at single density.

modulation. The process by which some characteristic of one wave (usually high frequency) is varied in accordance with another wave or signal (usually low frequency). This technique is used in modems to make business-machine signals compatible with communication facilities.

modulation rate. The reciprocal of the measure of the shortest nominal time interval between successive significant instants of the modulated signal. If this measure is expressed in seconds, the modulation rate is expressed in baud.

module. (1) A program unit that is discrete and identifiable with respect to compiling, combining with other units, and loading. (2) A packaged functional hardware unit designed for use with other components.

modulo check. A calculation performed on values entered into a system. This calculation is designed to detect errors.

modulo-N check. A check in which an operand is divided by a number N (the modulus) to generate a remainder (check digit) that is retained with the operand. For example, in a modulo-7 check, the remainder will be 0, 1, 2, 3, 4, 5, or 6. The operand is later checked by again dividing it by the modulus; if the remainder is not equal to the check digit, an error is indicated.

modulus. In a modulo-N check, the number by which the operand is divided.

monitor. Synonym for cathode ray tube display (CRT display).

most-significant digit. The leftmost (non-zero) digit. See also high-order position.

ms. Millisecond; 0.001 second.

multiplexer. A device capable of interleaving the events of two or more activities, or capable of distributing the events of an interleaved sequence to the respective activities.

multiprogramming. (1) Pertaining to the concurrent execution of two or more computer programs by a computer. (2) A mode of operation that provides for the interleaved execution of two or more computer programs by a single processor.

n. Prefix nano; 0.000,000,001.

NAND. A logic operator having the property that if P is a statement, Q is a statement, R is a statement,..., then the NAND of P, Q ,R,... is true if at least one statement is false, false if all statements are true.

NAND gate. A gate in which the output is 0 only if all inputs are 1.

nano (n). Prefix 0.000,000,001.

nanosecond (ns). 0.000,000,001 second.

negative true. Synonym for active low.

negative-going edge. The edge of a pulse or signal changing in a negative direction. Synonymous with falling edge.

non-return-to-zero change-on-ones recording (NRZI). A transmission encoding method in which the data terminal equipment changes the signal to the opposite state to send a binary 1 and leaves it in the same state to send a binary 0.

non-return-to-zero (inverted) recording (NRZI). Deprecated term for non-return-to-zero change-on-ones recording.

NOR. A logic operator having the property that if P is a statement, Q is a statement, R is a statement,..., then the NOR of P, Q, R,... is true if all statements are false, false if at least one statement is true.

NOR gate. A gate in which the output is 0 only if at least one input is 1.

NOT. A logical operator having the property that if P is a statement, then the NOT of P is true if P is false, false if P is true.

NRZI. Non-return-to-zero change-on-ones recording.

ns. Nanosecond; 0.000,000,001 second.

NUL. The null character.

null character (NUL). A control character that is used to accomplish media-fill or time-fill, and that may be inserted into or removed from, a sequence of characters without affecting the meaning of the sequence; however, the control of the equipment or the format may be affected by this character.

odd-even check. Synonym for parity check.

offline. Pertaining to the operation of a functional unit without the continual control of a computer.

one-shot. A circuit that delivers one output pulse of desired duration for each input (trigger) pulse.

open circuit. (1) A discontinuous circuit; that is, one that is broken at one or more points and, consequently, cannot conduct current. Contrast with closed circuit. (2) Pertaining to a no-load condition; for example, the open-circuit voltage of a power supply.

open collector. A switching transistor without an internal connection between its collector and the voltage supply. A connection from the collector to the voltage supply is made through an external (pull-up) resistor.

operand. (1) An entity to which an operation is applied. (2) That which is operated upon. An operand is usually identified by an address part of an instruction.

operating system. Software that controls the execution of programs; an operating system may provide services such as resource allocation, scheduling, input/output control, and data management.

OR. A logic operator having the property that if P is a statement, Q is a statement, R is a statement,..., then the OR of P, Q, R,...is true if at least one statement is true, false if all statements are false.

OR gate. A gate in which the output is 1 only if at least one input is 1.

output. Pertaining to a device, process, or channel involved in an output process, or to the data or states involved in an output process.

output process. (1) The process that consists of the delivery of data from a data processing system, or from any part of it. (2) The return of information from a data processing system to an end user, including the translation of data from a machine language to a language that the end user can understand.

overcurrent. A current of higher than specified strength.

overflow indicator. (1) An indicator that signifies when the last line on a page has been printed or passed. (2) An indicator that is set on if the result of an arithmetic operation exceeds the capacity of the accumulator.

overrun. Loss of data because a receiving device is unable to accept data at the rate it is transmitted.

overvoltage. A voltage of higher than specified value.

parallel. (1) Pertaining to the concurrent or simultaneous operation of two or more devices, or to the concurrent performance of two or more activities. (2) Pertaining to the concurrent or simultaneous occurrence of two or more related activities in multiple devices or channels. (3) Pertaining to the simultaneity of two or more processes. (4) Pertaining to the simultaneous processing of the individual parts of a whole, such as the bits of a character and the characters of a word, using separate facilities for the various parts. (5) Contrast with serial.

parameter. (1) A variable that is given a constant value for a specified application and that may denote the application. (2) A name in a procedure that is used to refer to an argument passed to that procedure.

parity bit. A binary digit appended to a group of binary digits to make the sum of all the digits either always odd (odd parity) or always even (even parity).

parity check. (1) A redundancy check that uses a parity bit. (2) Synonymous with odd-even check.

PEL. Picture element.

personal computer. A small home or business computer that has a processor and keyboard and that can be connected to a television or some other monitor. An optional printer is usually available.

phototransistor. A transistor whose switching action is controlled by light shining on it.

picture element (PEL). The smallest displayable unit on a display.

polling. (1) Interrogation of devices for purposes such as to avoid contention, to determine operational status, or to determine readiness to send or receive data. (2) The process whereby stations are invited, one at a time, to transmit.

port. An access point for data entry or exit.

positive true. Synonym for active high.

positive-going edge. The edge of a pulse or signal changing in a positive direction. Synonymous with rising edge.

potentiometer. A variable resistor with three terminals, one at each end and one on a slider (wiper).

power supply. A device that produces the power needed to operate electronic equipment.

printed circuit. A pattern of conductors (corresponding to the wiring of an electronic circuit) formed on a board of insulating material.

printed-circuit board. A usually copper-clad plastic board used to make a printed circuit.

priority. A rank assigned to a task that determines its precedence in receiving system resources.

processing program. A program that performs such functions as compiling, assembling, or translating for a particular programming language.

processing unit. A functional unit that consists of one or more processors and all or part of internal storage.

processor. (1) In a computer, a functional unit that interprets and executes instructions. (2) A functional unit, a part of another unit such as a terminal or a processing unit, that interprets and executes instructions. (3) Deprecated term for processing program. (4) See microprocessor.

program. (1) A series of actions designed to achieve a certain result. (2) A series of instructions telling the computer how to handle a problem or task. (3) To design, write, and test computer programs.

programmable read-only memory (PROM). A read-only memory that can be programmed by the user.

programming language. (1) An artificial language established for expressing computer programs. (2) A set of characters and rules with meanings assigned prior to their use, for writing computer programs.

programming system. One or more programming languages and the necessary software for using these languages with particular automatic data-processing equipment.

PROM. Programmable read-only memory.

propagation delay. (1) The time necessary for a signal to travel from one point on a circuit to another. (2) The time delay between a signal change at an input and the corresponding change at an output.

protocol. (1) A specification for the format and relative timing of information exchanged between communicating parties. (2) The set of rules governing the operation of functional units of a communication system that must be followed if communication is to be achieved.

pulse. A variation in the value of a quantity, short in relation to the time schedule of interest, the final value being the same as the initial value.

radio frequency (RF). An ac frequency that is higher than the highest audio frequency. So called because of the application to radio communication.

radix. (1) In a radix numeration system, the positive integer by which the weight of the digit place is multiplied to obtain the weight of the digit place with the next higher weight; for example, in the decimal numeration system the radix of each digit place is 10. (2) Another term for base.

radix numeration system. A positional representation system in which the ratio of the weight of any one digit place to the weight of the digit place with the next lower weight is a positive integer (the radix). The permissible values of the character in any digit place range from 0 to one less than the radix.

RAM. Random access memory. Read/write memory.

random access memory (RAM). Read/write memory.

RAS. In the IBM Personal Computer, row address strobe.

raster. In computer graphics, a predetermined pattern of lines that provides uniform coverage of a display space.

read. To acquire or interpret data from a storage device, from a data medium, or from another source.

read-only memory (ROM). A storage device whose contents cannot be modified. The memory is retained when power is removed.

read/write memory. A storage device whose contents can be modified. Also called RAM.

recoverable error. An error condition that allows continued execution of a program.

red-green-blue-intensity (RGBI). The description of a direct-drive color monitor that accepts input signals of red, green, blue, and intensity.

redundancy check. A check that depends on extra characters attached to data for the detection of errors. See cyclic redundancy check.

register. (1) A storage device, having a specified storage capacity such as a bit, a byte, or a computer word, and usually intended for a special purpose. (2) A storage device in which specific data is stored.

retry. To resend the current block of data (from the last EOB or ETB) a prescribed number of times, or until it is entered correctly or accepted.

reverse video. A form of highlighting a character, field, or cursor by reversing the color of the character, field, or cursor with its background; for example, changing a red character on a black background to a black character on a red background.

RF. Radio frequency.

RF modulator. The device used to convert the composite video signal to the antenna level input of a home TV.

RGBI. Red-green-blue-intensity.

rising edge. Synonym for positive-going edge.

ROM. Read-only memory.

ROM/BIOS. The ROM resident basic input/output system, which provides the level control of the major I/O devices in the computer system.

row address strobe (RAS). A signal that latches the row address in a memory chip.

RS-232C. A standard by the EIA for communication between computers and external equipment.

RTS. Request to send. Associated with modem control.

run. A single continuous performance of a computer program or routine.

saturation. In computer graphics, the purity of a particular hue. A color is said to be saturated when at least one primary color (red, blue, or green) is completely absent.

scaling. In computer graphics, enlarging or reducing all or part of a display image by multiplying the coordinates of the image by a constant value.

schematic. The representation, usually in a drawing or diagram form, of a logical or physical structure.

Schottky TTL. A version (S series) of TTL with faster switching speed, but requiring more power. See also transistor-transistor logic and low power Schottky TTL.

SDL. Shielded Data Link

SDLC. Synchronous Data Link Control.

sector. That part of a track or band on a magnetic drum, a magnetic disk, or a disk pack that can be accessed by the magnetic heads in the course of a predetermined rotational displacement of the particular device.

SERDES. Serializer/deserializer.

serial. (1) Pertaining to the sequential performance of two or more activities in a single device. In English, the modifiers serial and parallel usually refer to devices, as opposed to sequential and consecutive, which refer to processes. (2) Pertaining to the sequential or consecutive occurrence of two or more related activities in a single device or channel. (3) Pertaining to the sequential processing of the individual parts of a whole, such as the bits of a character or the characters of a word, using the same facilities for successive parts. (4) Contrast with parallel.

GLOSSARY

serializer/deserializer (SERDES). A device that serializes output from, and deserializes input to, a business machine.

setup. (1) In a computer that consists of an assembly of individual computing units, the arrangement of interconnections between the units, and the adjustments needed for the computer to operate. (2) The preparation of a computing system to perform a job or job step. Setup is usually performed by an operator and often involves performing routine functions, such as mounting tape reels. (3) The preparation of the system for normal operation.

short circuit. A low-resistance path through which current flows, rather than through a component or circuit.

signal. A variation of a physical quantity, used to convey data.

sink. A device or circuit into which current drains.

software. (1) Computer programs, procedures, and rules concerned with the operation of a data processing system. (2) Contrast with hardware.

source. The origin of a signal or electrical energy.

square wave. An alternating or pulsating current or voltage whose waveshape is square.

square wave generator. A signal generator delivering an output signal having a square waveform.

SS. Start-stop.

start bit. (1) A signal to a receiving mechanism to get ready to receive data or perform a function. (2) In a start-stop system, a signal preceding a character or block that prepares the receiving device for the reception of the code elements.

start-of-text (STX). A transmission control character that precedes a text and may be used to terminate the message heading.

start-stop system. A data transmission system in which each character is preceded by a start bit and is followed by a stop bit.

start-stop (SS) transmission. (1) Asynchronous transmission such that a group of signals representing a character is preceded by a start bit and followed by a stop bit. (2) Asynchronous transmission in which a group of bits is preceded by a start bit that prepares the receiving mechanism for the reception and registration of a character and is followed by at least one stop bit that enables the receiving mechanism to come to an idle condition pending the reception of the next character.

static memory. RAM using flip-flops as the memory elements. Data is retained as long as power is applied to the flip-flops. Contrast with dynamic memory.

stop bit. (1) A signal to a receiving mechanism to wait for the next signal. (2) In a start-stop system, a signal following a character or block that prepares the receiving device for the reception of a subsequent character or block.

storage. (1) A storage device. (2) A device, or part of a device, that can retain data. (3) The retention of data in a storage device. (4) The placement of data into a storage device.

strobe. An instrument that emits adjustable-rate flashes of light. Used to measure the speed of rotating or vibrating objects.

STX. Start-of-text.

symbol. (1) A conventional representation of a concept. (2) A representation of something by reason of relationship, association, or convention.

synchronization. The process of adjusting the corresponding significant instants of two signals to obtain the desired phase relationship between these instants.

Synchronous Data Link Control (SDLC). A protocol for management of data transfer over a data link.

synchronous transmission. (1) Data transmission in which the time of occurrence of each signal representing a bit is related to a fixed time frame. (2) Data transmission in which the sending and receiving devices are operating continuously at substantially the same frequency and are maintained, by means of correction, in a desired phase relationship.

syntax. (1) The relationship among characters or groups of characters, independent of their meanings or the manner of their interpretation and use. (2) The structure of expressions in a language. (3) The rules governing the structure of a language. (4) The relationships among symbols.

text. In ASCII and data communication, a sequence of characters treated as an entity if preceded and terminated by one STX and one ETX transmission control character, respectively.

time-out. (1) A parameter related to an enforced event designed to occur at the conclusion of a predetermined elapsed time. A time-out condition can be cancelled by the receipt of an appropriate time-out cancellation signal. (2) A time interval allotted for certain operations to occur; for example, response to polling or addressing before system operation is interrupted and must be restarted.

track. (1) The path or one of the set of paths, parallel to the reference edge on a data medium, associated with a single reading or writing component as the data medium moves past the component. (2) The portion of a moving data medium such as a drum, or disk, that is accessible to a given reading head position.

transistor-transistor logic (TTL). A popular logic circuit family that uses multiple-emitter transistors.

translate. To transform data from one language to another.

transmission. (1) The sending of data from one place for reception elsewhere. (2) In ASCII and data communication, a series of characters including headings and text. (3) The dispatching of a signal, message, or other form of intelligence by wire, radio, telephone, or other means. (4) One or more blocks or messages. For BSC and start-stop devices, a transmission is terminated by an EOT character. (5) Synonymous with data transmission.

TTL. Transistor-transistor logic.

typematic key. A keyboard key that repeats its function when held pressed.

V. Volt.

vector. In computer graphics, a directed line segment.

video. Computer data or graphics displayed on a cathode ray tube, monitor, or display.

view point. In computer graphics, the origin from which angles and scales are used to map virtual space into display space.

viewing reference point. In computer graphics, a point in the modeling coordinate space that is a defined distance from the view point.

viewing transformation. Operations on the coordinates of an object (usually matrix multiplications) that cause the view of the object to be rotated about any axis, translated (moved without rotating), and/or scaled (changed in size along any or all dimensions). Viewing transformation differs from modeling transformation in that perspective is considered. See also modeling transformation.

viewplane. The visible plane of a CRT display screen that completely contains a defined window.

viewport. In computer graphics, a predefined part of the CRT display space.

volt. The basic practical unit of electric pressure. The potential that causes electrons to flow through a circuit.

W. Watt.

watt. The practical unit of electric power.

window. (1) A predefined part of the virtual space. (2) The visible area of a viewplane.

word. (1) A character string or a bit string considered as an entity. (2) See computer word.

write. To make a permanent or transient recording of data in a storage device or on a data medium.

write precompensation. The varying of the timing of the head current from the outer tracks to the inner tracks of the diskette to keep a constant 'write' signal.

yon plane. In computer graphics, a plane that is perpendicular to the line joining the viewing reference point and the view point, and that lies beyond the viewing reference point. Any part of an object beyond the yon plane is not seen. See also hither plane.

Notes:

Bibliography

Intel Corporation. *The 8086 Family User's Manual.* This manual introduces the 8086 family of microcomputing components and serves as a reference in system design and implementation.

Intel Corporation. *8086/8087/8088 Macro Assembly Reference Manual for 8088/8085 Based Development System.* This manual describes the 8086/8087/8088 Macro Assembly Language and is intended for persons who are familiar with assembly language.

Intel Corporation. *Component Data Catalog.* This book describes Intel components and their technical specifications.

Motorola, Inc. *The Complete Microcomputer Data Libary.* This book describes Motorola components and their technical specifications.

National Semiconductor Corporation. *250 Asynchronous Communications Element.* This book documents physical and operating characteristics of the INS 8250.

Notes:

Index

A

AAA 6-10, 6-11
AAD 6-13
AAM 6-12
AAS 6-12
adapter card with ROM 5-10
ADC 6-10
ADD 6-10
address
  bits 0 to 19 (A0–A19) 1-20
  enable (AEN), I/O channel 1-20
  latch enable (ALE), I/O channel 1-20
  map, I/O channel 1-25
  map, I/O planar 1-24
AEN (address enable) 1-20
ALE (address latch enable), I/O channel 1-20
alternate key 4-41
AND 6-14
arithmetic instructions 6-10, 6-26
ASCII characters 7-3
ASCII, extended 4-34

B

bandwidth formula 1-14
  specifications I/O channel 1-15
BASIC
  DEF SEG 5-8
  reserved interrupt 5-7, 5-8
basic assurance test 4-25
BASIC reserved interrupts 5-7
BAT (basic assurance test) 4-25
BAT Completion Code command 4-26
BAT Failure Code command 4-26
binary integers (coprocessor) 2-3, 2-4
BIOS
  parameter passing 5-4
  quick reference 5-11, 5-111
  software interrupt 5-5
  system ROM 5-11, 5-111
  use of 5-3
bit map, I/O 8255A 1-27
block diagram
  system timer 1-10
block diagram (coprocessor) 2-6
break 4-11
break code 4-24
break key 4-42
buffer, keyboard 4-24

INDEX

C

cabling 4-23
CALL 6-16
caps lock key 4-10, 4-41
card specifications 1-31
CBW 6-13
CH CK, negative (-channel check), I/O channel 1-22
channel check, negative (-CH CK), I/O channel 1-22
channel, I/O
 pin assignments 1-17
character codes 4-6, 4-34
character codes (keyboard) 4-8
characters 7-3
CLC 6-23
CLD 6-23
CLI 6-23
CLK, I/O channel 1-21
clock (CLK), I/O channel 1-21
clock and data signals 4-32
 data output 4-33
 data stream 4-33
CMC 6-23
CMP 6-12
CMPS 6-15
codes
 character 4-34
 extended 4-38
commands from the system
 Reset 4-26
commands to the system
 BAT (basic assurance test) Completion Code 4-26
 BAT Failure 4-26
 Key Detection Error 4-27
 Overrun 4-27
comparison instructions 6-25
component diagram, system board 1-19
connector specifications 4-19
connectors
 J-1 through J-8 1-16
 keyboard 1-33
 power supply 1-32
 speaker 1-33
 system board 1-32
connectors (power supply) 3-6, 3-9
constants instructions 6-28
control key 4-40
control transfer instructions 6-16
Ctrl state 4-38
CWD 6-13

D

DAS 6-12
data
 bits 0 to 7 (D0–D7) 1-21
 flow, system board diagram 1-6
data output 4-33
data stream 4-33
data transfer instructions 6-7, 6-24
DEC 6-11
decimal integers (coprocessor) 2-3, 2-4
delay, typematic 4-24
description 4-22
 buffer 4-24
 cabling 4-23
 key-code scanning 4-23
 keys 4-24
 sequencing key-code scanning 4-23
description I/O channel 1-20

diagram, system board 1-19
diagrams
logic, 101/102-key keyboard 4-52
logic, 83-key keyboard 4-21
logics, 256/640K 1-46
logics, 64/256K 1-34
DIV 6-12
DMA request 1 to 3 (DRQ1–DRQ3) 1-21
DOS
keyboard function 5-7

E

encoding, keyboard 4-33
ESC 6-23
extended ASCII 4-6, 4-34
extended codes 4-9, 4-38

F

FABS 6-28
FADD 6-26
FCHS 6-28
FCLEX 6-30
FCOM 6-25
FCOMP 6-26
FCOMPP 6-26
FDECSTP 6-30
FDISI 6-29
FDIV 6-27
FENI 6-29
FFREE 6-30
FIFO 4-24
FINCSTP 6-30
FINIT 6-29
FLD 6-24
FLDCW 6-29
FLDENV 6-30
FLDLG2 6-29
FLDLN2 6-29
FLDL2T 6-29
FLDP1 6-29
FLDZ 6-28
FLD1 6-28
FMUL 6-27
FNOP 6-30
FPATAN 6-28
FPREM 6-27
FPTAN 6-28
French keyboard 4-13, 4-45
FRNDINT 6-27
FRSTOR 6-30
FSAVE 6-30
FSCALE 6-27
FSQRT 6-27
FST 6-25
FSTCW 6-29
FSTENV 6-30
FSTP 6-25
FSTSW 6-29
FSUB 6-26
FTST 6-26
FWAIT 6-30
FXAM 6-26
FXCH 6-25
FXTRACT 6-27
FYL2X 6-28
FYL2XP1 6-28
F2XM1 6-28

G

generator, refresh request 1-10
German keyboard 4-14, 4-46

INDEX

H

HLT 6-23

I

I/O channel
address map, channel 1-25
address map, planar 1-24
ALE (address latch enable) 1-20
bit map 8255A 1-27
CH CK (-I/O Channel Check) 1-22
CH RDY (I/O Channel Ready), I/O channel 1-22
check (-CH CK) 1-22
CLK 1-21
description 1-20
I/O channel 1-15
oscillator (OSC) 1-23
pin assignments 1-17
read command (-IOR) 1-22
Reset Drive (RESET DRV) 1-23
Terminal Count (T/C) 1-23
Write Command (-IOW) 1-22
I/O channel connectors 1-17
IDIV 6-12
IMUL 6-12
IN 6-8
INC 6-10
instructions
arithmetic 6-10, 6-26
comparison 6-25
constants 6-28
control transfer 6-16
data transfer 6-7, 6-24
logic 6-13
rotate 6-13
shift 6-13
string manipulation 6-15
INT 6-22
Intel 8048 4-3
Intel 8088 microprocessor,
arithmetic 6-8, 6-19
comparison 6-19
conditional transfer operations 6-15
constants 6-21
control transfer 6-12
data transfer 6-6, 6-17
instruction set index 6-27
instruction set matrix 6-25
instuction set extensions 6-17
logic 6-10
memory segmentation model 6-5
operand summary 6-4
processor control 6-16, 6-22
register model 6-3
second instruction byte summary 6-4
string manipulation 6-11
transcendental 6-21
use of segment override 6-5
interrupt request 2 to 7 (IRQ2–IRQ7) 1-22
INTO 6-22
IRET 6-22
Italian keyboard 4-15, 4-47

J

JB/JNAE 6-17
JBE/JNA 6-17
JCXZ 6-19
JE/JZ 6-17
JL/JNGE 6-17
JLE/JNG 6-17
JMP 6-16
JNB/JAE 6-18
JNBE/JA 6-18
JNE/JNZ 6-18
JNL/JGE 6-18
JNLE/JG 6-18
JNO 6-18
JNP/JPO 6-18
JNS 6-19
JO 6-18
JP/JPE 6-18
JS 6-18

K

key-code scanning 4-23
Key Detection Error command 4-27
keyboard 4-3
  connector 1-33, 4-19
  encoding 4-33
  interface 4-5
  layout 4-12, 4-35
  power-on self test 4-4
  routine 4-43
keyboard buffer 4-24
keyboard data output 4-33
keyboard extended codes
  alt 4-10
  alternate 4-41
  break 4-11, 4-42
  caps lock 4-10, 4-41
  combinations 4-41
  ctrl 4-9, 4-40
  number lock 4-41
  pause 4-11, 4-42
  print screen 4-11, 4-42
  scroll lock 4-10, 4-41
  shift 4-9, 4-40
  system request 4-42
  system reset 4-11
keyboard layouts
  French 4-13, 4-45
  German 4-14, 4-46
  Italian 4-15, 4-47
  Spanish 4-16, 4-48
  UK English 4-17, 4-49
  US English 4-18, 4-50
keyboard scan 4-3
keyboard scan codes 4-6, 4-28
keyboard, French 4-13, 4-45
keyboard, German 4-14, 4-46
keyboard, Italian 4-15, 4-47
keyboard, Spanish 4-16, 4-48
keyboard, UK English 4-17, 4-49
keyboard, US English 4-18, 4-50
keys 4-24
keys, typematic 4-4, 4-24

L

LAHF 6-9
layout, keyboard 4-35
layouts
  French 4-13, 4-45
  German 4-14, 4-46
  Italian 4-15, 4-47
  Spanish 4-16, 4-48
  UK English 4-17, 4-49

INDEX

US English 4-18, 4-50
LDS 6-9
LEA 6-9
LES 6-9
line protocol 4-25
LOCK 6-23
LODS 6-15
logic diagrams 4-52
logic diagrams, system board, 256/640K 1-46
logic diagrams, system board, 64/256K 1-34
logic instructions 6-13
LOOP 6-19
LOOPNZ/LOOPNE 6-19
LOOPZ/LOOPE 6-19

M

make code 4-4, 4-24
make/break 4-24
math coprocessor
binary integers 2-3, 2-4
block diagram 2-6
control word 2-5
decimal integers 2-3, 2-4
hardware interface 2-4
NMI 2-5
QS0 2-4
QS1 2-4
real numbers 2-3, 2-4
memory locations
reserved 5-8
memory map
BIOS 5-8
memory map, system 1-8
memory read command (-MEMR) 1-23
memory write command (-MEMW) 1-23
-MEMR (memory read command) 1-23
-MEMW (memory write command) 1-23
modules, RAM 1-12
modules, ROM/EPROM 1-13
MOV 6-7
MOVS 6-15
MUL 6-12

N

NEG 6-11
NMI (coprocessor) 2-5
NOP 6-23
NOT 6-13
Num Lock key 4-9, 4-11
Num Lock state 4-38
number lock key 4-41

O

OR 6-14
OSC (oscillator), I/O channel 1-23
oscillator (OSC), I/O channel 1-23
OUT 6-8
output, keyboard 4-33
Overrun command 4-27

P

parameter passing (ROM BIOS) 5-4
software interrupt listing 5-5
pause 4-11

pause key 4-42
POP 6-8
POPF 6-9
POR (power-on reset) 4-25
power good signal 3-5, 3-8
power-on reset 4-25
power-on routine 4-25
  basic assurance test 4-25
  BAT (basic assurance test) 4-25
  POR (power-on reset) 4-25
  power-on reset 4-25
power-on self test 4-4
power requirements 4-51
power supply
  connectors 1-32
power supply (system) 3-3
  connectors 3-6, 3-9
  input requirements 3-4, 3-7
  outputs 3-4, 3-8
  overvoltage/overcurrent protection 3-5
  pin assignments 3-6, 3-9
  power good signal 3-5, 3-8
PPI 1-26
print screen key 4-11, 4-42
priorities, shift key 4-41
processor control, 8087 6-29
Programmable Peripheral Interface 1-26
protocol 4-25
PUSH 6-7
PUSHF 6-9

Q

QS0 (coprocessor) 2-4
QS1 (coprocessor) 2-4
quick reference charts 7-14
quick reference, character set 7-14

R

RAM modules 1-12
RAM subsystem 1-12
rate, typematic 4-24
RCL 6-14
RCR 6-14
read command I/O channel 1-22
read memory command (-MEMR) 1-23
ready (RDY), I/O channel 1-22
real numbers (coprocessor) 2-3, 2-4
refresh request generator 1-10
REP 6-15
request interrupt 2 to 7 (IRQ2–IRQ7) 1-22
reserved interrupts
  BASIC and DOS 5-7
Reset command 4-26
RESET DRV, I/O channel 1-23
reset, power-on 4-25
reset, system 4-42
RET 6-17
ROL 6-13
ROM scan codes 4-33
ROM subsystem 1-13

INDEX

ROM/EPROM
modules 1-13
ROR 6-13
rotate instructions 6-13
routine, keyboard 4-6, 4-43

S

SAHF 6-9
SAR 6-13
SBB 6-11
scan code tables 4-28
scan codes 4-28
scan codes, ROM 4-33
scanning, key-code
sequencing 4-23
SCAS 6-15
scroll lock 4-10
scroll lock key 4-10, 4-41
sequencing key-code
scanning 4-23
shift 4-8
shift instructions 6-13
shift key 4-9, 4-40
shift key priorities 4-10, 4-41
shift states 4-9, 4-40
SHL/SAL 6-13
SHR 6-13
signals (I/O)
AEN 1-20
ALE 1-20
A0–A19 1-20
CLK 1-21
-DACK0–DACK3 1-21
DRQ1–DRQ3 1-21
D0–D7 1-21
-I/O CH CK 1-22
I/O CH RDY 1-22
-IOR 1-22
-IOW 1-22
IRQ2–IRQ7 1-22
-MEMR 1-23
-MEMW 1-23
OSC 1-23
RESET DRV 1-23
T/C 1-23
signals, clock and data 4-32
software interrupt listing
(8088) 5-5
Spanish keyboard 4-16, 4-48
speaker circuit 1-26
speaker connector 1-33
speaker drive system 1-26
speaker tone generation 1-10
specifications 4-51
power requirements 4-51
size 4-51
weight 4-51
states
Ctrl 4-9, 4-38
Num Lock 4-9, 4-38
Shift 4-9, 4-38, 4-40
STC 6-23
STD 6-23
STI 6-23
STOS 6-16
stream, data 4-33
string manipulation
instructions 6-15
SUB 6-11
subsystem, RAM 1-12
subsystem, ROM 1-13
switches
dual in-line package (DIP)
switch 1-3
I/O Bit Map 1-27
system board 1-19
system board
data flow diagrams 1-6
diagram 1-19
logic diagrams,
256/640K 1-46
logic diagrams,
64/256K 1-34

system board
 connectors 1-32
system board,
 256/640K 1-13
system board,
 64/256K 1-12, 1-13
system clock (CLK), I/O
 channel 1-21
system memory map 1-8
system request key 4-42
system reset 4-11, 4-42
system ROM BIOS 5-11,
 5-111
system timer block
 diagram 1-10
system timers 1-10

T

terminal count (T/C), I/O
 channel 1-23
TEST 6-14
timer/counters 1-10
timers, system 1-10
tone generation,
 speaker 1-10
typematic delay 4-24
typematic keys 4-4, 4-24
typematic rate 4-24

U

UK English keyboard 4-17,
 4-49
US English keyboard 4-18,
 4-50

V

vectors with special
 meanings 5-5

W

WAIT 6-23
write command (-IOW), I/O
 channel 1-22
write memory command
 (-MEMW) 1-23

X

XCHG 6-8
XLAT 6-9
XOR 6-15

Numerics

8088, (see also Intel 8088
 microprocessor) 1-4
8254-2 1-10
8255A bit map 1-27

INDEX

Reader's Comment Form

Technical Reference **6139821**

Your comments assist us in improving the usefulness of our publication; they are an important part of the input used for revisions.

IBM may use and distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Please do not use this form for technical questions regarding the IBM Personal Computer or programs for the IBM Personal Computer, or for requests for additional publications; this only delays the response. Instead, direct your inquiries or request to your authorized IBM Personal Computer dealer.

Comments:

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS **PERMIT NO. 40** **ARMONK, NEW YORK**

POSTAGE WILL BE PAID BY ADDRESSEE

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

Fold here

Tape

Please do not staple

Tape